Journal of Baku Engineering University

ISSN 2521-635X
E-3107-0531

Volume 9
Number 2
2025

MATHEMATICS AND COMPUTER SCIENCE

Journal is published twice a year
Number-1. June, Number-2. December

An International Journal

http://journal.beu.edu.az

CONTACT ADDRESS

Journal of Baku Engineering University
AZ0102, Khirdalan city, Hasan Aliyev str. 120, Absheron, Baku, Azerbaijan
Tel: 00 994 12 - 349 99 95 Fax: 00 994 12 349-99-90/91

e-mail: *journal@beu.edu.az*
web: *http://journal.qu.edu.az*
facebook: *Journal Of Baku Engineering University*

ISSN 2521-635X | E-3107-0531

ISSN 2521-635X | E-3107-0531

Journal of Baku Engineering University

MATHEMATICS AND COMPUTER SCIENCE

Baku - AZERBAIJAN

Journal of Baku Engineering University

MATHEMATICS AND COMPUTER SCIENCE

2025. Volume 9, Number 2

CONTENTS

JOURNAL OF BAKU ENGINEERING UNIVERSITY- MATHEMATICS AND COMPUTER SCIENCE
2025. Volume 9, Number 2 *Pages 113-136*

UDC: 004.932.2
DOI: https://doi.org/10.30546/09090.2025.510.1002

SENSE: SELF-SUPERVISED NEURAL EMBEDDINGS FOR SPATIAL ENSEMBLES

HAMID GADIROV*, LENNARD MANUEL, STEFFEN FREY

University of Groningen,

Netherlands

** h.gadirov@rug.nl*

ARTICLE INFO

Article history
Received:2025-12-16
Received in revised form:2026-01-12
Accepted:2026-01-13
Available online

Keywords:
Self-supervised learning
Neural embeddings
Autoencoders
Clustering
Latent space analysis

ABSTRACT

Analyzing and visualizing scientific ensemble datasets characterized by high dimensionality and structural complexity remains a significant challenge. While dimensionality reduction methods and autoencoders are widely used for feature extraction, their performance often degrades in high-dimensional settings. In this work, we propose an enhanced autoencoder framework that integrates clustering and contrastive loss functions into the latent space to improve the interpretability and visualization of scientific ensemble data. Clustering is guided by a soft silhouette score, encouraging compact and well-separated latent representations. To address the presence of unlabeled data, EfficientNetV2 is employed to generate pseudo-labels for partially unlabeled ensembles. The model is trained by jointly optimizing reconstruction, clustering, and contrastive objectives, resulting in improved grouping of similar samples and clearer separation of distinct structures in the latent space. UMAP is subsequently applied to the learned embeddings to produce two-dimensional visualizations, which are quantitatively evaluated using silhouette scores. We evaluate multiple autoencoder variants on two scientific ensemble datasets: subsurface channel structures generated via Markov chain Monte Carlo simulations and droplet-impact dynamics on a liquid film. The results show that incorporating clustering or contrastive objectives yields marginal but consistent improvements over baseline autoencoders.

1. Introduction

In recent years, the growth of high-dimensional scientific ensemble datasets has presented both opportunities and challenges for data analysis and visualization [1]. Scientific ensembles, characterized by their complex and multi-dimensional nature, contain valuable insights that can assist in decision-making processes across various domains, from climate modeling to healthcare diagnostics [2]. However, extracting meaningful features from these datasets remains a difficult task due to their complexity.

To make these complex ensemble datasets more understandable, dimensionality reduction techniques can be applied. However, these techniques struggle to uncover the structures in high-dimensional datasets. Thus, we first apply feature extraction through the use of (variational) autoencoders. These models extract the most relevant features from the datasets, after which dimensionality reduction techniques can be used to obtain a more intuitive visualization.

Combining these two methods has shown promising results, but we hope to achieve better clustering within this visualization [3].

In this paper, we propose a novel approach for clustering scientific ensemble datasets by combining the strengths of autoencoder-based feature extraction with a dedicated clustering and contrastive loss function. Our method aims to extract relevant features of scientific data while simultaneously encouraging the forming of distinct clusters in the latent space. By jointly optimizing the reconstruction objective during training as well as the cluster separability enforced by either the clustering or contrastive loss, our approach offers a framework to obtain a better understandable visualization.

We implement a soft silhouette score, which is a differentiable version of the silhouette score. This score is implemented as a clustering loss alongside the reconstruction loss of a (variational) autoencoder during training, ensuring that the data will form more compact clusters while also preserving the original data's information. This clustering loss will also be compared to a contrastive loss function. Contrastive loss aims to bring instances of the same class closer together while pushing apart the instances of different classes, ensuring that similar data will be grouped in the latent space. This clustering during training can be performed on mostly unlabelled datasets because EffNetV2 will be used to first train the model on the manually labelled part of the ensemble datasets, after which pseudo-labels for the unlabelled part of the datasets will be generated, making this a semi-supervised problem.

Once the training is finished, we will perform dimensionality reduction to further reduce the latent space to a 2D visualization by performing dimensionality reduction: specifically by utilizing UMAP. The resulting visualizations will be evaluated by their silhouette score and compared to similar models, with and without a clustering or contrastive loss. In our experiments, we used two ensemble datasets: Markov Chain Monte Carlo and Drop Dynamics [4], [5].

In Section 2, we give a brief overview of related works. Afterwards, we describe the methodology used for this paper in Section 3. Then, we move on to the results and discussion in Section 4. We conclude our findings in Section 5. Finally, we discuss future works in Section 6.

This work is based on the master's internship project by Lennard Manuel titled "Autoencoder-based semi-supervised dimensionality reduction and clustering for scientific ensembles" from the University of Groningen [40].

2. Related Work

In this section, we briefly describe what research has been done previously in the fields of autoencoder-based feature extraction, deep clustering, and contrastive learning.

Autoencoder-based Feature extraction. Autoencoders have become instrumental in the field of feature extraction due to their ability to learn efficient, compressed representations of high-dimensional data. Ardelean et al. propose autoencoders as a feature extraction method for spike sorting, the process of grouping spikes of distinct neurons into their respective clusters [6]. Autoencoders are also widely used in computer vision. Nayak et al. use a deep autoencoder to help detect brain tumors in medical images [7]. Chen et al. propose a convolutional autoencoder to help in detecting and analyzing long nodules [8]. Solomon et al. use autoencoders to develop a face verification system [9]. Furthermore, Variational Autoencoders are also useful in this process. Tian et al. developed the Pyramid-VAE-GAN network to assist in image inpainting [10].

More recently, Yin et al. introduced ENTIRE, an autoencoder-based framework that extracts structure-aware feature representations from time-dependent volumetric data and combines them with rendering parameters to accurately predict volume rendering time, enabling dynamic parameter adaptation and load balancing in visualization pipelines [37].

Deep clustering. Deep clustering refers to the process of integrating deep learning networks with clustering meth- ods. It helps in transforming the input data such that clusters will try to form within the latent space [11]. In the paper "Deep clustering using the soft silhouette score: towards compact and well-separated clusters" Vardakas et al. introduce a probabilistic formulation of the silhouette score to complement their autoencoders' reconstruction loss with a clustering loss [12]. They use a Radial Basis Function model as a clustering network to predict the probabilities with which they calculate the soft silhouette score. They show promising results on the EMNIST datasets. Xie et al. propose the Deep Embedding Clustering (DEC) method, which also optimizes both the reconstruction and clustering objective using deep neural networks [13]. They use the KL divergence as their clustering loss. Guo et al. adapt the DEC method and develop the Improved Deep Embedding Clustering (IDEC) method [14]. This method simultaneously optimizes the reconstruction and clustering objective during the training phase, whereas DEC pre-trains on the reconstruction objective, after which it optimizes the clustering objective. Yang et al. also propose their own method: the Deep Clustering Network (DCN) [15]. This method tries to optimize the clustering objective using k -means on the embedded space.

Contrastive loss. Contrastive learning is a deep learning technique that is effective in creating separation between different classes. Zhou et al. for example propose a contrastive autoencoder (CAE-AD) for anomaly detection in multivariate time series [16]. Luo et al. also combine contrastive learning with an autoencoder to assist in out-of-distribution detection [17]. Lopez-Avila et al. combine a denoising autoencoder with contrastive learning to help fine-tune their transformer models [18]. Contrastive learning also has its place in the medical world: Cao et al. propose ContrastNet, which combines prototypical contrastive learning with an autoencoder to create an unsupervised feature learning network for hyperspectral classification [19].

Ensemble data analysis. Modern scientific simulations and measurements often generate large spatio-temporal ensembles, where each member represents a different realization of the same phenomenon under varying parameters or initial conditions. Extracting meaningful low-dimensional structure from such ensembles is crucial for tasks such as uncertainty quantification, pattern discovery, and interactive visualization. Autoencoder-based methods are particularly well suited for this setting because they can learn compact latent representations that preserve salient spatial and temporal structures while remaining flexible across different simulation domains. In the context of ensemble visualization, autoencoder architectures have been systematically evaluated as feature extractors and dimensionality reduction models for spatial ensembles, with a focus on how architectural choices affect projection quality and the expressiveness of the resulting embeddings [35]. These studies demonstrate that appropriate encoder–decoder configurations can yield latent spaces in which ensemble members cluster according to physically meaningful behaviors, thereby supporting downstream visual analysis and clustering.

Beyond static dimensionality reduction, recent work has explored deep learning for learning flow fields and reconstructing missing temporal information in ensemble data. FLINT introduces a learning-based approach for flow estimation and temporal interpolation that reconstructs

velocity fields and scalar quantities in 2D+time and 3D+time ensembles, enabling high-quality in-between time steps without strong domain assumptions [34]. HyperFLINT extends this idea with a hypernetwork that conditions on simulation parameters, improving generalization across different ensemble configurations and supporting parameter-aware interpolation quality [33]. Together, these methods illustrate how neural networks can capture both spatial and temporal coherence in ensembles and use this knowledge to fill temporal gaps, generate dense time sampling, and support fluid, artifact-free animation and analysis [38].

These developments are part of a broader trend toward machine-learning-driven ensemble data analysis in scientific visualization. A recent comprehensive study on machine learning for scientific visualization discusses autoencoder-based dimensionality reduction, learning-based flow estimation, and hypernetwork-based adaptation as complementary building blocks for analyzing complex spatio-temporal ensembles [36]. In parallel, autoencoders and related deep architectures have been applied to other ensemble-related tasks, such as learning low-dimensional probabilistic representations of ensemble forecast fields and leveraging variational autoencoders for ensemble visualization and uncertainty exploration in meteorology. The usefulness of such learned latent representations is further highlighted by applications beyond "pure" data analysis. For example, ENTIRE uses deep features derived from volumetric data and camera parameters to predict volume rendering time, enabling dynamic parameter adaptation and more stable interactive performance for time-dependent volume data [37]. Collectively, these works underline that autoencoder-based representations and related deep learning techniques provide a powerful and versatile foundation for ensemble data analysis, supporting not only visualization and clustering but also performance prediction and system-level steering in complex simulation workflows.

Recent advances in Artificial Intelligence. Recent advances in large language models (LLMs) and foundation models have begun to influence scientific data analysis workflows, including ensemble-based visualization and simulation analysis. While LLMs are primarily developed for language tasks, their underlying architectural and system-level innovations—such as attention mechanisms, scalable training paradigms, and modular model composition—are increasingly relevant for handling large, heterogeneous scientific datasets. In particular, LLM-driven interfaces and multimodal models show promise for assisting exploratory analysis, metadata reasoning, and interactive steering of ensemble simulations by coupling learned representations with semantic context. As scientific datasets continue to grow in size and complexity, real-time optimization and scalability have become central concerns. Techniques such as mixed-precision training, low-rank adaptation (LoRA), and model pruning significantly reduce computational overhead while preserving model performance [39]. System-level frameworks including vLLM and DeepSpeed further enable efficient training and inference at scale, while sparse expert architectures such as Switch Transformers demonstrate how conditional computation can reduce resource usage without sacrificing expressiveness. Complementary approaches based on reinforcement learning allow models to adapt their complexity dynamically in response to available computational budgets. Moreover, distributed strategies such as federated learning and ZeRO improve scalability and collaboration across institutions handling large-scale ensemble datasets. Finally, emerging work on agentic AI systems—in which autonomous agents coordinate learning, inference, and analysis tasks—opens new directions for ensemble data processing. When combined with containerization technologies such as Docker, these agents can be deployed as modular, reproducible components that manage data ingestion [41, 42], model

execution, and visualization pipelines in a flexible and scalable manner, supporting robust, system-level integration of learning-based ensemble analysis.

3. Methodology

In this section, we discuss and explain all steps of the pipeline of our approach, which is shown in Figure 1. First, the ensemble data is preprocessed, after which we generate the pseudo- labels using EffNetV2. Following this, feature extraction is performed with the (variational) autoencoders using reconstruction loss combined with either clustering or contrastive loss. Then, the latent space is projected to a 2D space using UMAP, which is then evaluated by their silhouette scores. Then, all the results are compared.

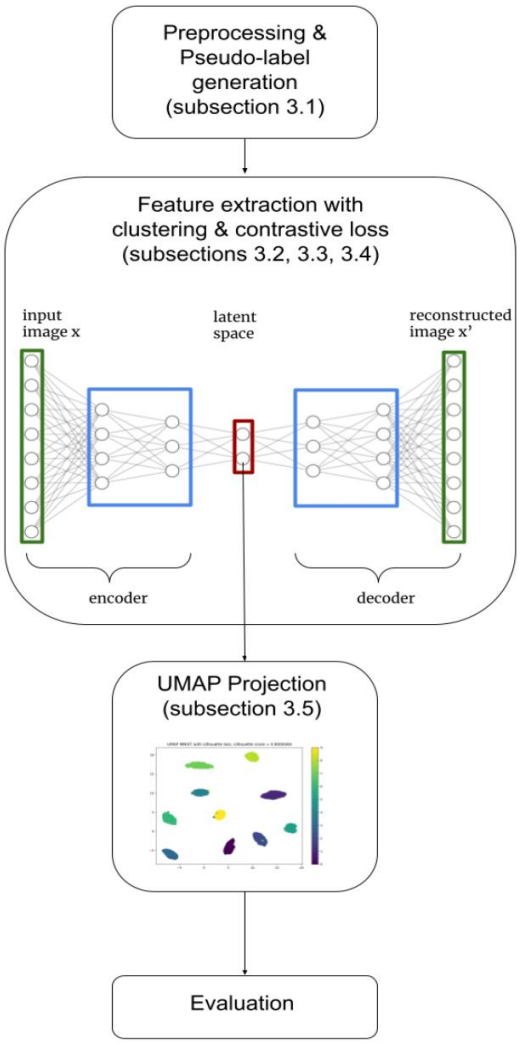


Figure 1: Pipeline of the developed method.

3.1 Scientific ensemble datasets

In this project, we consider two ensemble datasets. The first is the Markov chain Monte Carlo (MCMC) ensemble dataset, which depicts channel structures in soil [4]. This ensemble contains 95K images. The images are monochrome and have a resolution of 50 × 50. An example of the MCMC images is shown in Figure 2.

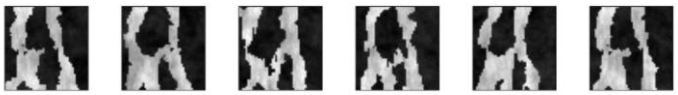


Figure 2: Markov chain Monte Carlo images examples.

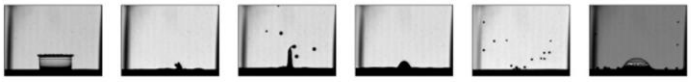


Figure 3: Drop dynamics images examples.

The second ensemble dataset is the drop dynamics (DD) ensemble, which studies the impact of a droplet with a film [5]. These images are also monochrome and have a resolution of 160×224. This ensemble contains 135K images. An example of the DD images is shown in Figure 3.

For both ensembles, a small subset of the datasets is labelled manually. For the MCMC ensemble, 2.5K images are labelled, and 7.2K for the DD ensemble. This labelling was done by observing the different behaviour types shown by the images. For MCMC, there are five different categorical classes. For the DD ensemble, there are seven different categorical classes: bubble, bubble-splash, column, crown, crown-splash, splash, and drop.

Because the ensemble datasets are only partially labelled, we use an EfficientNetV2 model to train on this part, and then generate pseudo-labels for the unlabelled subset of the ensembles, making this a semi-supervised problem. EfficientNetV2 is an image classification convolutional network, and it is known for its small size, speed and performance [20]. For this project, a subset of 30K images for the MCMC ensemble will be used, and 26K images for the DD ensemble. Both datasets are normalized to zero mean and unit standard deviation.

3.2 Architecture

In this subsection, (variational) autoencoders will be explained in detail, and the architectures used for this project will be described.

Autoencoders. Autoencoders and (β)-variational autoencoders will be used for this project. An autoencoder (AE) is a type of artificial neural network that is often used for unsupervised learning problems [21]. An autoencoder aims to learn a representation (encoding) for a dataset by training the network to reconstruct the original input as accurately as possible. It consists of two main parts: an encoder and a decoder.

The encoder compresses the input data into a lower-dimensional representation, also known as the latent space or encoding. It typically consists of one or more hidden layers that progressively reduce the dimensionality of the input data, mapping it to a lower-dimensional representation. The encoder's output represents a compressed version of the input data, capturing its essential features. Mathematically, an encoder with one hidden layer can be expressed as $z = \sigma(Wx + b)$, where z is the latent representation, σ is the activation function, W is the weight matrix, x is the input image, and b is the bias.

The decoder reconstructs the original input data from the encoded representation produced by the encoder. It typically consists of one or more hidden layers that gradually expand the dimensionality of the encoded representation back to the original input dimensionality. The decoder's output should ideally closely match the input data, effectively "decoding" the

compressed representation into a reconstructed version of the original input. Mathematically, a decoder with one hidden layer can be expressed as x′ = σ′(W′z + b′), where x′ is the reconstructed image, σ′ is the activation function, W′ is the weight matrix, z is the latent representation, and b′ is the bias. The general structure of an autoencoder is shown in Figure 4.

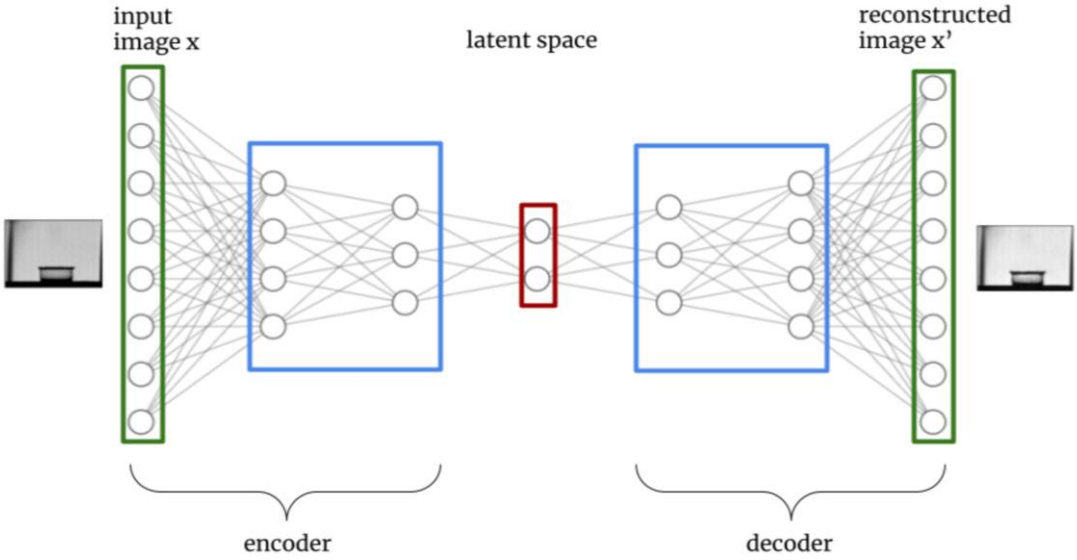


Figure 4: Autoencoder general structure.

During training, an autoencoder is trained to minimize the reconstruction loss function that quantifies the difference between the input data and the reconstructed output. In this project, the Mean Squared Error is chosen as the loss function. Mathematically, MSE is expressed as $L(x, x') = \|x - x'\|_2$.

Variational autoencoders. Variational autoencoders (VAE) are a type of autoencoder and are similar to the previously explained autoencoders in many ways [22]. The key difference between a VAE and a traditional autoencoder lies in how they handle the latent space representation. In a traditional autoencoder, the encoder maps input data to a fixed-dimensional latent space, and the decoder reconstructs the input data from this latent representation. However, in a VAE, the latent space is probabilistic, meaning that instead of encoding data points to a specific point z in the latent space, the encoder outputs the parameters of a probability distribution over the latent space. Then, the decoder takes a sampled point from this distribution and generates a new image. Because of VAEs nature of using a sample from the distribution, they can produce new images. This can help in preventing overfitting, as the model learns a more general structure from the image.

Furthermore, VAEs also add a regularization term to the loss function, called the Kullback-Leibler (KL) divergence. The KL divergence encourages the learned latent space to approximate a unit Gaussian distribution, helping to ensure that the latent space remains interpretable. The training objective of a VAE is to maximize the Evidence Lower Bound (ELBO), which is comprised of two parts:

$$\mathbb{E}_{q_\phi(z|x)}[logp_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p_\theta(z)), \quad (1)$$

where the prior distribution with a unit Gaussian distribution (zero mean and unit standard deviation) and the probability distribution of the parameters μ and σ are outputted by the encoder. This means that expected value is equal to the reconstruction loss, and the second part is equal to the KL divergence, which measures the difference between the probability distribution outputted by the encoder and the prior distribution. The ELBO (evidence lower bound) needs to be minimized.

As in the paper "Evaluation and selection of autoencoders for expressive dimensionality reduction of spatial ensembles" [3], the KL divergence is scaled according to the dimensionality of the latent space according to the following formula:

$$\frac{dim(latent)}{height \cdot width}, \tag{2}$$

where dim(latent) is the size of the latent space, and height and width are the height and width of the input image. The general structure of a VAE is shown in Figure 5.

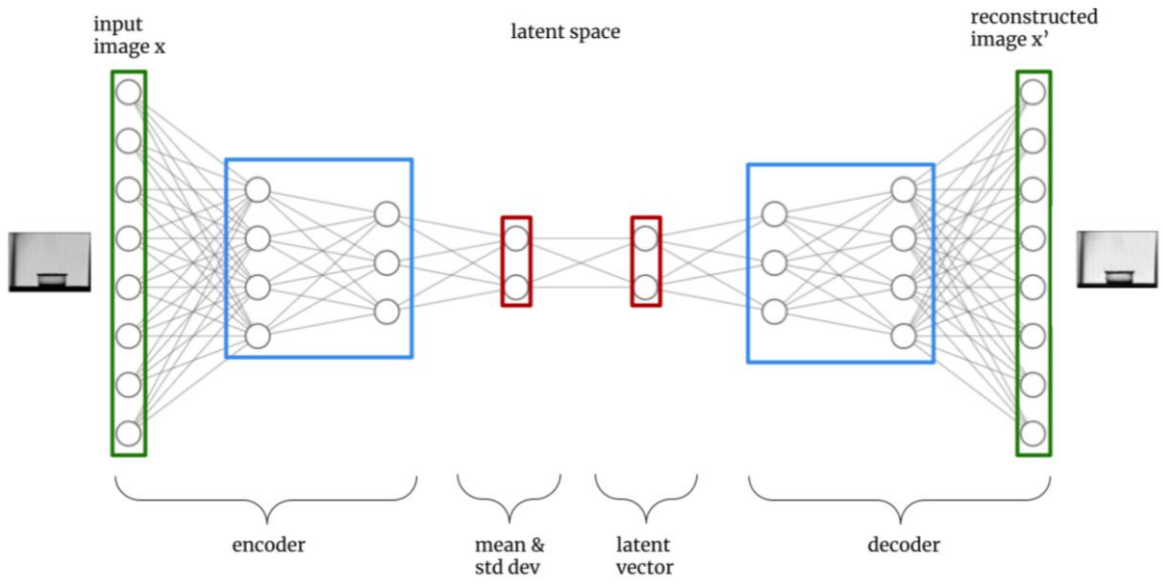


Figure 5: Variational Autoencoder general structure.

A β-VAE is a type of VAE that adds the Lagrangian multiplier β which balances the reconstruction accuracy and impact of the KL divergence factor [23]. Using the Lagrangian multiplier β can help to create an even more expressive latent space. With a β-VAE, the KL divergence coefficient of Equation 2 is multiplied by this Lagrange Multiplier β. This means that a β value of 1 means that it is equal to a VAE.

The architecture used for the AE and (β)-VAE is the same symmetric structure as in [3], meaning that the decoder is reversed to the encoder. First, the resolution of input images is reduced by half after all four convolutional layers. This is done by using a stride of 2. The kernel size for convolution was set to 3, and the number of filters is equal to 64 with zero padding. As an optimizer, Adam was used with a learning rate of 0.0005 [24]. The ReLU activation function was used throughout with random weight initialization. ReLU is linear in the positive dimension but

0 for any negative value [25]. Furthermore, the batch size used in this project is equal to 128. This value was the largest Habrok could reliably handle without running into GPU memory issues.

3.3 Clustering loss

To enhance the performance of autoencoder-based clustering, we complement the reconstruction loss with silhouette score as clustering loss. The silhouette score is a widely used clustering quality metric, which measures how similar an item is to other items in the same cluster (cohesion) as well as how different it is from items from other clusters (separation) [26]. The silhouette score ranges from −1 to 1. A high value (close to 1) indicates that an item has good separation from other clusters while also being similar to items from its cluster (good cohesion). A low value (close to -1) indicates that the item has both bad separation and bad cohesion. The silhouette score is defined as follows:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}, \tag{3}$$

where a(i) is the mean distance of sample i to all the points in its own cluster, and b(i) is the mean distance between i and the nearest cluster. Thus, to achieve a high silhouette score, a(i) must be minimized, meaning that a point's distance to the other points in its cluster must be low (good cohesion), while b(i) must be maximized, meaning that the distance to other clusters must be high (good separation).

However, we cannot simply implement the silhouette score as is, as the function needs to be differentiable to ensure that backpropagation works. This is why we introduce the soft silhouette score: a differentiable variant of the traditional silhouette score [12]. The soft silhouette score is entirely implemented through tensor operations, ensuring its differentiability. We define the *clustering loss* as follows:

$$\mathcal{L}_{cl} = 1 - \mathcal{S}_f, \tag{4}$$

where Sf is the soft silhouette score. 1 is subtracted from the soft silhouette score to ensure that the clustering loss is in the range [0, 2], where 0 is the best achievable clustering loss. By adding this clustering loss alongside the reconstruction loss Lrec, we ensure that the data will form more compact clusters while also preserving the information of the original data. To strike a balance between the clustering and reconstruction objectives, we introduce a clustering coefficient λcl to scale the contribution of the clustering loss. By adjusting the coefficient of the clustering loss, we control the emphasis placed on clustering relative to reconstruction during training. This approach allows us to fine-tune the trade-off between clustering accuracy and reconstruction performance. This gives us the following loss function:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_{cl}\mathcal{L}_{cl} \tag{5}$$

3.4 Contrastive loss

Contrastive loss takes the vectors for a positive example and calculates its distance to an example of the same class and contrasts that with the distance to negative examples [27]. The goal is thus to bring instances with the same class closer together while pushing apart the instances with different classes. It helps ensure that the positive examples are represented by similar vectors

while negative examples are represented by dissimilar vectors. This is accomplished by calculating the distances of the vectors with any distance metric. For this project, the Euclidean distance formula is used. These distances are then either minimized or maximized, based on whether the samples are positive or negative. The contrastive loss is defined as follows:

$$\mathcal{L}_{con} = y \cdot D^2 + (1 - y) \cdot max(0, margin - D)^2, \tag{6}$$

where D is the distance between features, y is 1 if a pair of samples belong to the same class (positive samples), and 0 if not (negative samples). In this formula a margin threshold margin is used. This threshold ensures that negative samples are pushed apart by at least a certain distance. Without the threshold, negative pairs could be arbitrarily close, which would not effectively separate different classes. Moreover, it also helps avoid over-penalizing negative samples that are already far apart. Once the distance between negative samples is larger than the margin, the loss contribution for those pairs becomes zero. In this project, a margin of 1 will be used. Furthermore, to once again strike a balance between the contrastive and reconstruction losses, a contrastive loss coefficient λ will also be used to scale the contribution of the contrastive loss. This results in the following loss function:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_{con}\mathcal{L}_{con} \tag{7}$$

3.5 Projection to 2D Space

Once the ensemble data has been transformed from its original physical space to a feature space through our encoding process, each data sample is represented by a latent vector. These latent vectors are then subjected to dimensionality reduction (DR) techniques. Through DR, the latent vectors are condensed into a lower dimension, in this case, a two-dimensional (2D) representation, while preserving the key information. This transformation to a 2D space helps the visual interpretation of the data, allowing for an informative representation of the patterns present within the dataset.

In this project, we use UMAP (Uniform Manifold Approximation and Projection) to project the latent vectors into a 2D space, as it was shown to outperform other DR techniques such as t-SNE or PCA in the paper "Evaluation and Selection of Autoencoders for Expressive Dimensionality Reduction of Spatial Ensembles" [3]. UMAP does its projection by first determining the similarities between nodes in its original dimension, after which it projects these nodes on a low-dimensional plot [28].

UMAP starts with computing the pairwise distances between the data points in its original dimension. After this, a fuzzy simplicial set is constructed, which captures the local relationship of the data points. Then, the low-dimensional embedding is optimized using stochastic gradient descent. After this optimization process, UMAP provides the 2D representation. Once the projection is done, we evaluate the performance by comparing the silhouette scores. These projections are only done on the manually labelled subset of the ensembles.

3.6 Hyperparameter search

In order to get the best results possible, a hyperparameter search is performed to achieve the best-performing model. An initial search is done on the silhouette and contrastive coefficient on the values of {0.01, 0.1, 0.2, 0.3}. The value with the best results is chosen, after which the hyperparameters shown in Table 1 will be searched on the AE and VAE.

In this grid, adaptive weights work as follows: the coefficient for the reconstruction objective will start at 1, whereas the coefficient for the silhouette or contrastive loss will start at 0. With every epoch, the reconstruction coefficient will decrease by 0.01, whereas the other coefficient will increase by 0.01. In case the latent space size is not searched, its default size is 256. The default setup uses no dropout layers.

Hyperparameter	Values/Used
Latent space size	{32, 64, 128, 256}
Dropout	{0.2, 0.3, 0.4}
β (for β-VAE)	{0.25, 0.5, 0.75, 1, 1.5, 2, 25, 30, 50, 75, 100}
Use of learning rate scheduler	{yes, no}
Adaptive weights	{yes, no}
Train on reconstruction objective, then clustering or contrastive objective	{yes, no}

Table 1: Grid search hyperparameters.

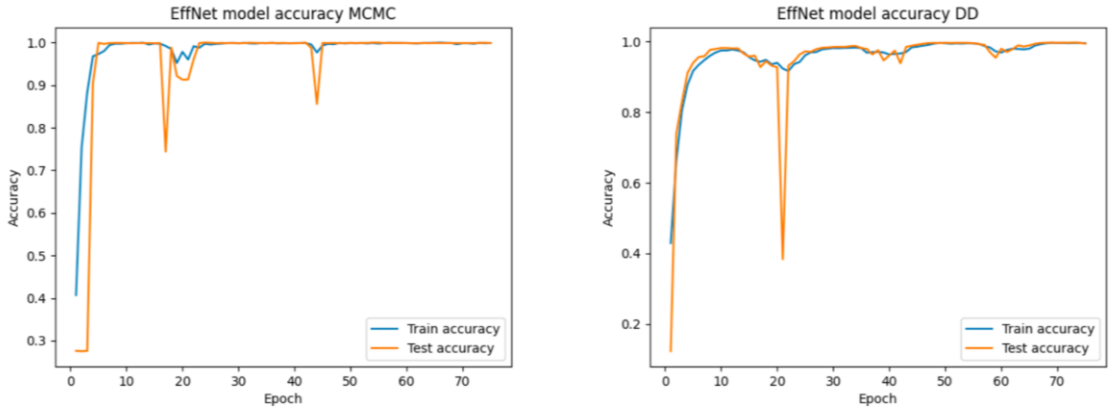
3.7 Setup

All code used for this project was performed using the Python programming language. The (variational) autoencoders were implemented through PyTorch [29]. CUDA was used to train the models on the GPU. The RUG's HPC H´abr´ok was used to perform all experiments on [30]. The amount of epochs is set to 100 for the ensemble datasets training, as we could see both the reconstruction and contrastive or clustering loss converge properly without overfitting. The dataset was split into a training and validation set with an 80%/20% ratio. The training and validation set only use the data that was labelled using the EffNetV2 model. Following training and validation, the UMAP projection is performed on the test set, which is the manually labelled part of the dataset. For the DD ensemble, the categorical classes will be converted into numerical classes to be able to compute the soft silhouette score. After some initial testing, the clustering and contrastive loss coefficients will be set to 0.2, as this value was shown to properly form clusters while succeeding in reconstructing the images.

4. Results & Discussion

In this section, we show all the results that we have produced. First, we examine the EffNetV2 pseudo-label generation, then we test our general pipeline on a simple dataset, the MNIST digits dataset. Following that, we show the obtained results from the two ensemble datasets, both with and without contrastive and clustering loss.

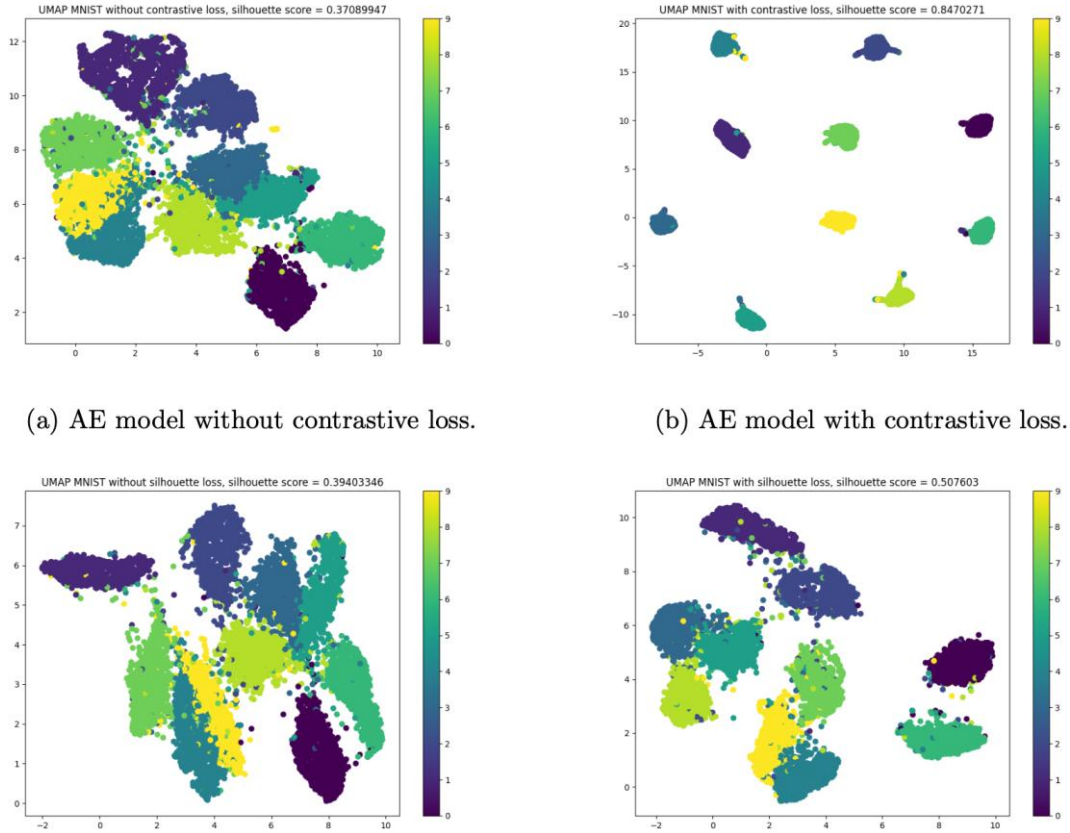
EffNetV2 pseudo-label generation. Firstly, EffNetV2 was trained on the manually labelled subset of the ensembles. Figure 6 shows that EffNetV2 was properly trained on this subset for both MCMC and DD, achieving a test accuracy of over 95%. Thus, we use this model to generate pseudo-labels for a subset of the unlabelled ensemble datasets.



(a) MCMC ensemble dataset. (b) DD ensemble dataset.

Figure 6: EffNetV2 model accuracy for the ensemble datasets.

MNIST results. Following this, the models are constructed and tested on a simple image dataset to see if the task succeeds. We have chosen MNIST: a dataset containing 70000 handwritten digit images of size 28 × 28 [31]. This dataset has 10 classes, representing digits 0 through 9. We compare some results of the AE and the VAE for both with and without silhouette loss and contrastive loss.



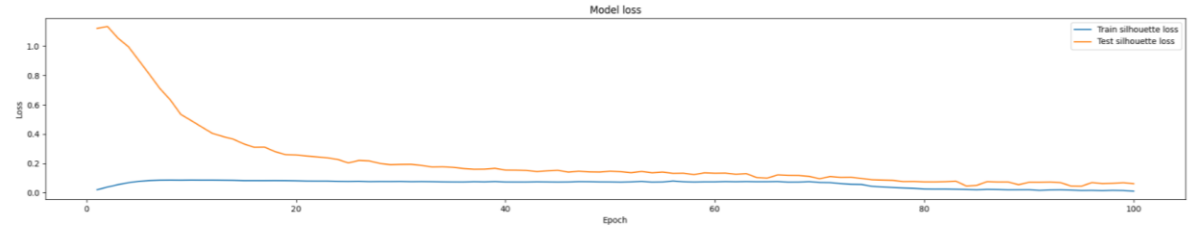
(a) AE model without contrastive loss. (b) AE model with contrastive loss.

(c) VAE model without silhouette loss. (d) VAE model with silhouette loss.

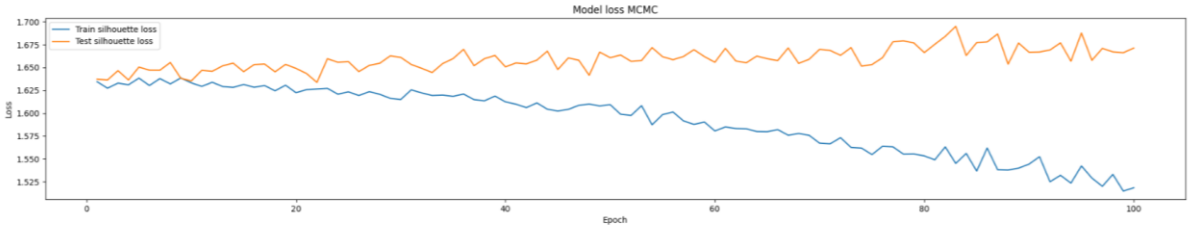
Figure 7: MNIST Autoencoder UMAP projections with and without contrastive loss.

Some of the results for this dataset are shown in Figure 7. As we can see, the models with silhouette or contrastive loss outperform the models without these losses and show good separation. In Figure 7b for example, we see that all clusters have some distance from the other clusters. Only a few mistakes are made, such as a few '9's being in the '4' cluster. We notice that the AE model outperforms the VAE model. This is likely due to the VAE's regularization term, as the MNIST dataset is simple enough not to need this. We can see that the AE model is already capable of creating distinct clusters, and thus no more complication of the problem is necessary.

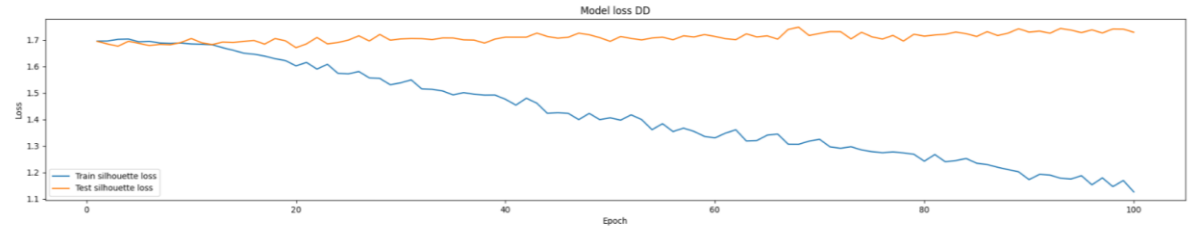
However, we cannot determine which loss or model is superior because no further finetuning has been done yet. We will do this for the scientific ensemble datasets, for which we will perform a hyperparameter search. A note has to be made that the MNIST dataset is a less complicated dataset, for which separation from other clusters is a simple task. This might prove to be more difficult for more complex datasets.



(a) MNIST AE silhouette loss.



(b) MCMC AE silhouette loss.



(c) DD AE silhouette loss.

Figure 8: Autoencoder training and testing silhouette loss.

Clustering loss issues. Before heading on to the scientific ensemble datasets in detail, we will first discuss some problems with the clustering loss. During the testing of the entire pipeline for the simpler MNIST dataset, no problems were encountered. Both the soft silhouette loss and the contrastive loss performed as expected. However, with the ensemble datasets, some issues arose. The issue is shown in Figure 8. We notice that for MNIST, shown in (a), the train and test silhouette losses decrease jointly, showing no signs of overfitting. For both MCMC and DD, shown in (b) and (c) respectively, we notice that the train silhouette loss does decrease over time, however, the testing silhouette loss remains high. This is a classic sign of overfitting: the model learns the training data too well and is incapable of generalizing because of it.

A few solutions to overfitting could be to apply early stopping, however, there is no real point at which the models are good at generalization. Batch normalization is also an option, and is tried unsuccessfully. Variational autoencoders apply a regularisation term, which also helps in stopping overfitting. Another solution we tried is to implement dropout: a regularization technique that drops a node in a neural network with a certain probability [32]. However, none of these solutions worked perfectly for the ensemble datasets, which is why we have chosen to primarily show images with contrastive loss, as this all worked adequately. The results with clustering loss are still supplied in the tables that will follow.

An example of what happens to the UMAP projection when the clustering loss is implemented is shown in Figure 9. In this projection, we can see some groups of points forming. However, never is the majority of these points of the same class. So, the model is capable of forming clusters because of the clustering loss, however, because the model is overfitted, it often chooses the wrong samples to cluster.

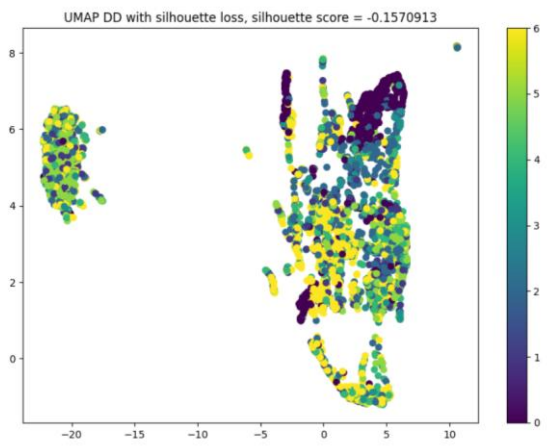


Figure 9: DD Autoencoder UMAP projection with clustering loss.

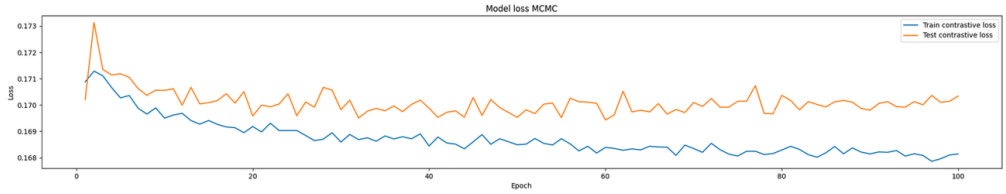
Markov chain Monte Carlo ensemble dataset. Next, we analyze how the models perform on the scientific ensemble datasets. First, we experiment with the Markov chain Monte Carlo ensemble dataset. In Figure 10, we see that both the reconstruction and contrastive objectives converge for the autoencoder models. Figure 10b shows the convergence plots of the model for MCMC, and we notice that the contrastive loss decreases after about 35 epochs and also converges. Compared to Figure 10a the contrastive loss value is significantly higher. We notice that in Figure 10a the test contrastive loss is slightly higher than the training contrastive loss. However, this difference is minimal, so there is no overfitting. Interestingly, the reconstruction losses for both the models with and without con- trastive loss converge in the same way. This shows that the models are capable of performing both the reconstruction and contrastive objectives.

To see if everything works, in Figure 11, the results are shown for a VAE model with and without silhouette loss, with a dropout layer applied with a value of 0.4. We see that the model with the clustering loss does perform better than the model without, albeit marginally. Furthermore, the reconstructed images are very close to the original images. Now, we will observe the results from the hyperparameter search. First, we look at some of the results for MCMC for the autoencoder. These results are shown in Table 2. Please note that in case the latent space size is not searched, its default size is 256. We notice that the best results are produced by

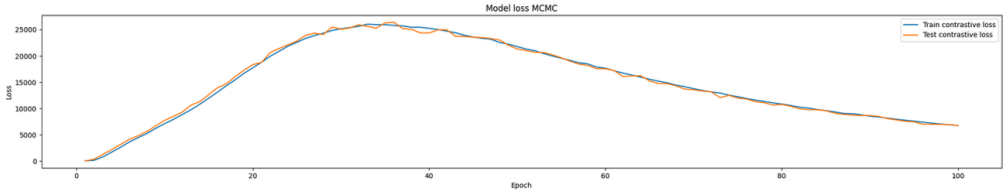
the models with either clustering loss or contrastive loss, in this case primarily when dropout of 0.4 is applied. To ensure, we take a look at its reconstructions. These are shown in Figure 12, where we observe that the reconstructions resemble the input images.

Hyperparameter	Baseline	With clustering loss	With contrastive loss
Latent space=32	0.05	0.08	0.01
Latent space=64	**0.02**	0.02	0.02
Latent space=256	0.03	0.05	−0.03
Dropout=0.3	0.03	0.07	−0.01
Dropout=0.4	0.03	**0.10**	**0.14**
Pretrained	0.03	−0.12	−0.03

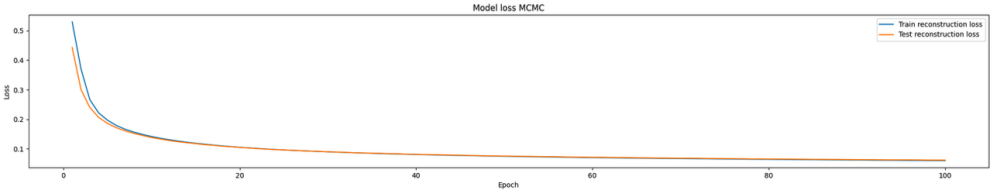
Table 2: Autoencoder hyperparameter search: UMAP silhouette scores for MCMC.



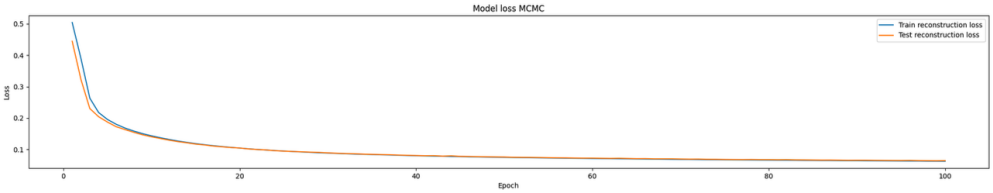
(a) MCMC AE contrastive loss for the model with contrastive loss.



(b) MCMC AE contrastive loss for the model without contrastive loss.



(c) MCMC AE reconstruction loss for the model with contrastive loss.



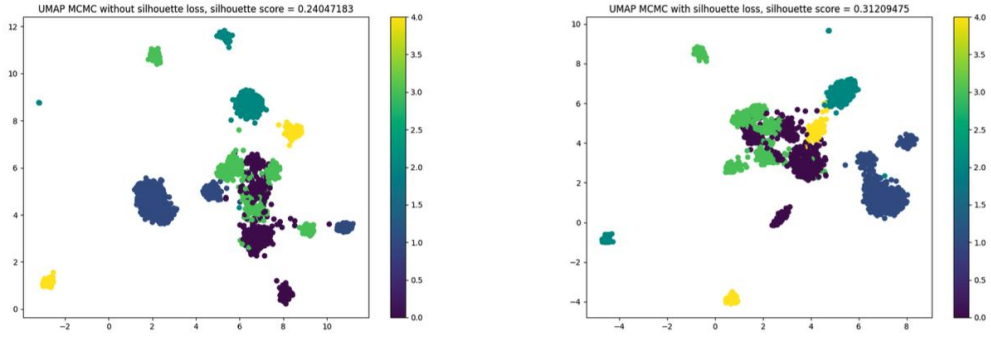
(d) MCMC AE reconstruction loss for the model without contrastive loss.

Figure 10: MCMC Autoencoder reconstruction and contrastive loss convergence plots for models with and without contrastive loss.

Then, we look at some of the results for MCMC for the (β)-variational autoencoder. These results are shown in Table 3. Here, we immediately see that the results are better than the AE variant, but often only slightly better than the baseline without clustering or contrastive loss. We observe the best result for MCMC, the VAE model with clustering loss with dropout of 0.3.

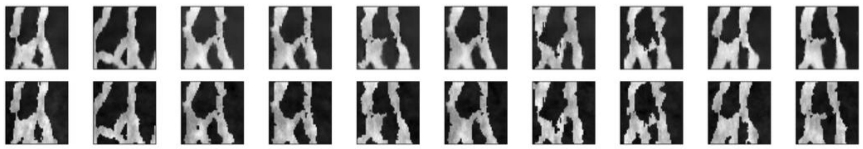
In Figure 13, a selected few projections are shown that are made for the MCMC ensemble. These projections are all by models with contrastive loss. Firstly, we observe the three autoen- coder

model projections, shown in plots (a) through (c). Here, we notice that the difference between the AE model with a latent space of 32 and the model with a latent space of 256 is minimal. The projections show similar clusters and show a minimally different silhouette score. The AE model with a dropout layer with a value of 0.4 shows similar behaviour but shows better cohesion within the clusters.



(a) VAE model without silhouette loss.

(b) VAE model with silhouette loss.



(c) VAE model reconstructions (top row) with silhouette loss with original images (bottom row).

Figure 11: MCMC Variational Autoencoder UMAP projections with and without silhouette loss.

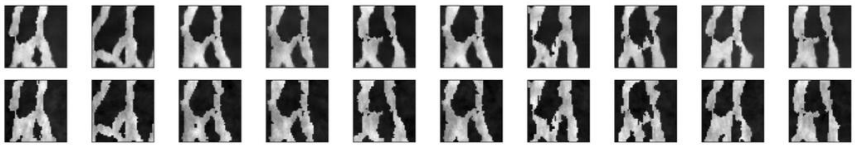


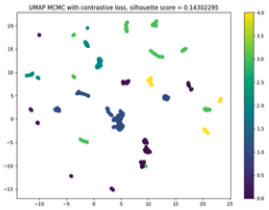
Figure 12: MCMC autoencoder reconstructions (top row) with dropout=0.4 with original images (bottom row).

Then, we analyse the remaining 6 VAE projections. First, we examine plots (d) through (f), the β-VAE models with different Lagrangian multiplier β values. As expected, the β-VAE model with a low value for β, shown in (d), is very similar to the autoencoder projections. This makes sense, as a low value for β decreases the impact of the KL divergence term, causing the clusters to have less of a Gaussian distribution form. However, with a too high value for β, as shown in (e) and ((f)), we notice that all points become scattered together. This means that the KL divergence constraint becomes too important, causing all images to become too general, and thus failing to learn the most relevant features. The latent space should be more constrained.

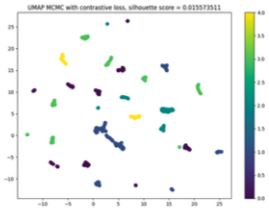
Now, we look at plots (g) through (i). Comparing (g) with its autoencoder counterpart, shown in (a), we see that the VAE model is more successful in forming cohesive clusters and that the clusters also have more of a Gaussian distribution form. Moreover, although in (g) some of the clusters do not have a lot of separation, some do, such as the cluster in the bottom-right with label 1, as opposed to the top-left cluster with label 2. This effect is not observed as much as in (a).

Hyperparameter	Baseline	With clustering loss	With contrastive loss
Latent space=32	−0.01	0.00	0.05
Latent space=256	0.26	0.21	0.23
Dropout=0.3	**0.26**	**0.28**	0.25
Dropout=0.4	0.24	0.21	**0.26**
β=0.25	0.13	0.19	0.21
β=0.5	0.06	0.13	0.19
β=2	0.13	0.17	0.20

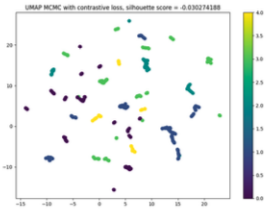
Table 3: Variational autoencoder hyperparameter search: UMAP silhouette scores for MCMC.



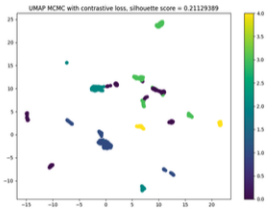
(a) AE model with a dropout layer with value = 0.4.



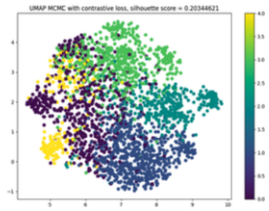
(b) AE model with latent space= 32.



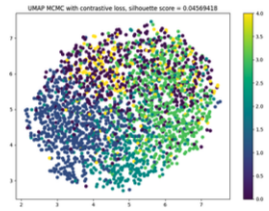
(c) AE model with latent space= 256.



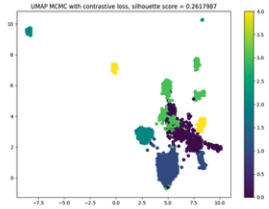
(d) β(0.25)-VAE model.



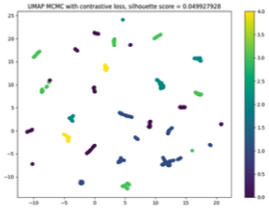
(e) β(2)-VAE model.



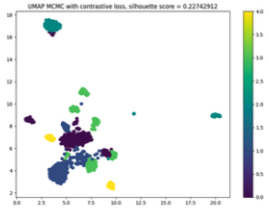
(f) β(4)-VAE model.



(g) VAE model with a dropout layer with value = 0.4.



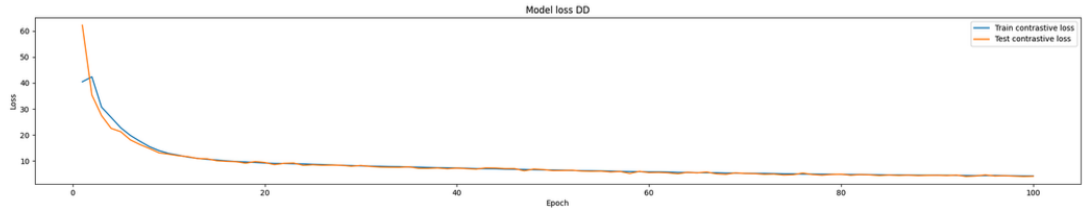
(h) VAE model with latent space= 32.



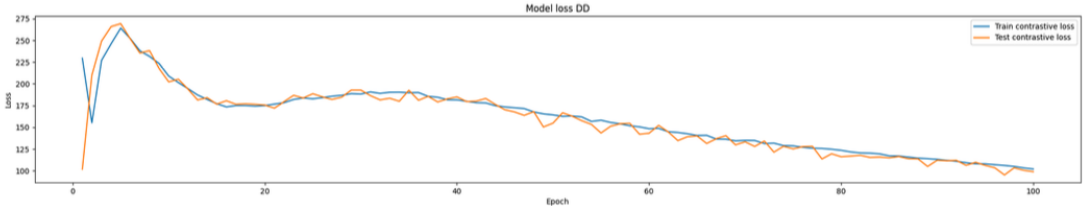
(i) VAE model with latent space= 256.

Figure 13: (β-variational) autoencoder models with contrastive loss UMAP projections on MCMC ensemble dataset.

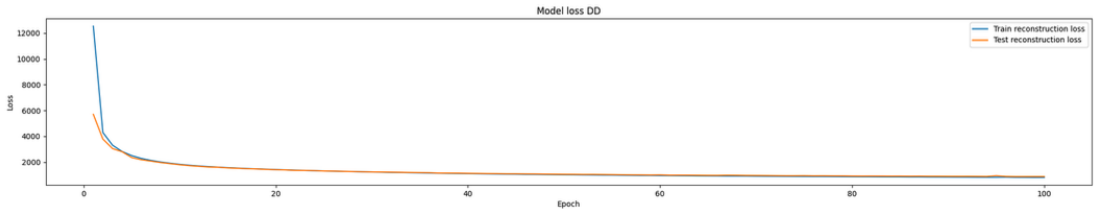
Furthermore, in plots (h) and (i), we compare the effect of the size of the latent space for VAEs. We immediately notice that in (h), the VAE model with a smaller latent space size, the projection is similar to the AE projections. This shows that the bottleneck is too small, causing the VAE struggles to learn relevant features as critical information is lost. The MCMC ensemble dataset is likely too complex to have such a small latent space size. In (i), we see a projection with a higher latent space size of 256. In this projection, we observe the same effect as in (g), with decent cohesion and some separation. Interestingly, comparing (h) and (i) to (b) and (c), we notice that VAEs do benefit from choosing a bigger bottleneck, whereas for AEs, the results are similar no matter the latent space size. This could be because variational autoencoders are more adept at learning complex features due to the KL divergence term.



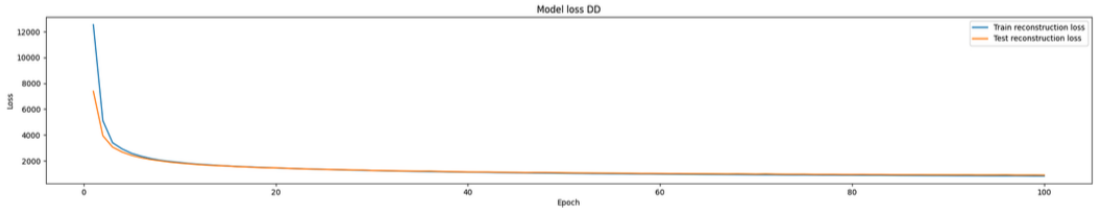
(a) DD VAE contrastive loss for the model with contrastive loss.



(b) DD VAE contrastive loss for the model without contrastive loss.



(c) DD VAE reconstruction loss for the model with contrastive loss.



(d) DD VAE reconstruction loss for the model without contrastive loss.

Figure 14: DD Variational Autoencoder reconstruction and contrastive loss convergence plots for models with and without contrastive loss.

Drop Dynamics ensemble dataset. First, in Figure 14a, we show the reconstruction and contrastive losses for Variational Autoen- coder models trained with and without contrastive loss. A VAE is shown here because we already showed the convergence for AE models in the previous subsection, for MCMC. We observe that the reconstruction and contrastive objectives converge for the variational autoencoder models. We notice in (b) that the contrastive losses do decrease over time, however, they are still significantly higher than the contrastive losses in (a). In (c) and (d), we see that the reconstruction losses also converge properly, and are much alike.

Now, we will observe the results from the hyperparameter search for the drop dynamics ensemble dataset. First, we look at some of the results for DD for the autoencoder models. These results are shown in Table 4. We see that the values are quite close to each other, no matter the configurations chosen for the models, and with or without clustering or contrastive loss.

Hyperparameter	**Baseline**	**With clustering loss**	**With contrastive loss**
Latent space=32	**-0.08**	−0.09	−0.11
Latent space=256	−0.09	−0.09	**-0.10**
Dropout=0.3	−0.09	−0.11	−0.12
Dropout=0.4	−0.12	**-0.07**	−0.14
Pretrained	−0.12	−0.15	−0.10

Table 4: Autoencoder hyperparameter search: UMAP silhouette scores for DD.

best results here are not significantly better than the autoencoder counterparts.

Hyperparameter	**Baseline**	**With clustering loss**	**With contrastive loss**
Latent space=32	−0.09	−0.13	−0.11
Latent space=128	−0.09	**-0.05**	−0.09
Latent space=256	−0.08	−0.11	−0.10
Dropout=0.3	−0.08	−0.12	−0.12
Dropout=0.4	−0.08	−0.08	−0.11
β=0.25	−0.07	−0.08	−0.07
β=1.5	**-0.06**	−0.11	−0.08
β=2	−0.09	−0.11	−0.09
β=25	−0.08	−0.08	**-0.05**
β=75	−0.07	−0.13	−0.07

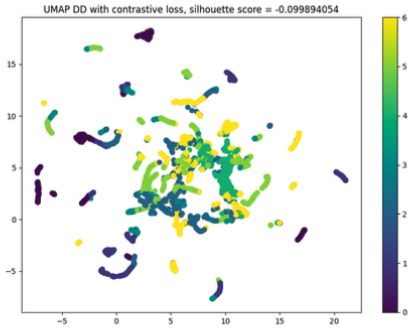
Table 5: Variational autoencoder hyperparameter search: UMAP silhouette scores for DD.

Now, we will observe the results from the hyperparameter search. First, we look at some of the results for DD for the autoencoder. These results are shown in Table 5. Whereas the results improved for MCMC with the VAE, we cannot say the same for the DD dataset. The best results here are not significantly better than the autoencoder counterparts.

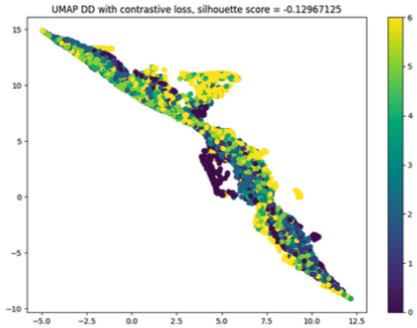
We will analyse the best UMAP projections of both models to see the differences. These projections are shown in Figure 15. We see that the shape of the clusters are quite different, once again most likely that the VAE looks different because of the KL-divergence term causing the clusters to have more of a Gaussian distribution. However, the projection results are not impressive. This is likely due to the extremely complicated nature of the drop dynamics dataset and also due to DD having a large number of classes. The reconstructed images, however, are satisfactory.

In Figure 16, a selected few projections are shown that are made for the DD ensemble. These projections are all by models trained with contrastive loss. In (a) and (b), we compare two autoencoder models. They show that the autoencoder struggles to create clusters, even with the contrastive objective. For the model with the latent space size of 32, we notice that the points are close to being scattered across a 1D line, meaning that the autoencoder possibly simplifies the features too much. This is possible because the number of features in the convolutional layers might be brought down too much, from the images' size of 160 × 224 to a small size of 32 quite quickly, causing too much information to be lost in the process. We see in (b) that this is marginally better for the model with a larger latent space size, however, no real distinct clusters are created either. In (c) through (e), we evaluate the importance of the Lagrangian multiplier β. In (c), we observe more distinct groups of points. However, the complex nature of the dataset makes it difficult to form cohesive clusters which have good separation. This is made even more

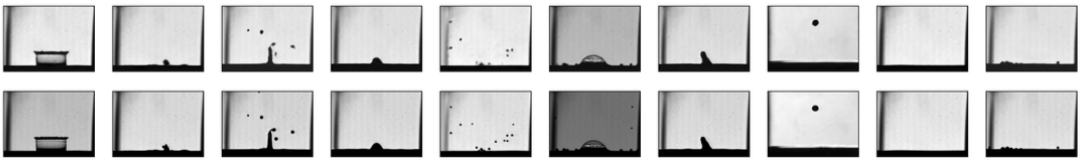
difficult by having seven different classes. By increasing the value of β, we notice in (e), that the points become fairly general, and that the KL divergence term is too involved. One can notice more of a Gaussian distribution in the created clusters. The clusters' cohesion is quite high, although points of the same class are fairly separate, except for, for example, classes 5 and 6. This behaviour is slightly better in (d), but the UMAP projection's silhouette score is not improved. In (f) and (g), we show the importance of the latent space size for the VAE models. The silhouette score of (g) is a bit better than (f), where the points are separated from each other more. Likely, this is due to the number of features decreasing too much within the model, having to simplify it too much in the process.



(a) VAE model with contrastive loss.



(b) AE model with contrastive loss.



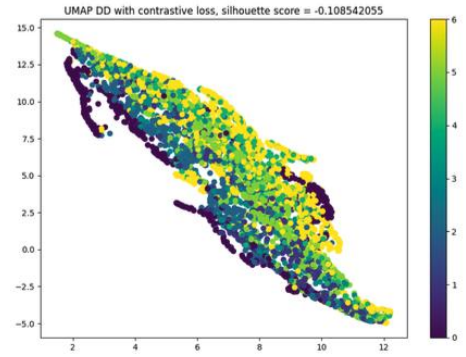
(c) VAE model with contrastive loss reconstructed images (reconstructions in the top row) with original images (bottom row).

Figure 15: DD Variational- and regular Autoencoder UMAP projections with contrastive loss.

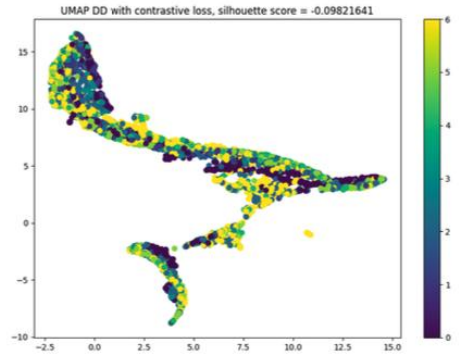
Comparison of results. When comparing the results of the MCMC and DD ensemble datasets, we observe that the results for the MCMC dataset are significantly better than for the DD dataset. We suspect this might be because the DD dataset has a few more classes than MCMC does (7 vs. 5) and because the DD dataset contains larger images than MCMC (160 × 224 vs. 50 × 50). Furthermore, DD's differences between classes are often quite minimal, a small splash in the top of the image could be the difference. Also, the crucial parts of these images are often at the bottom of the image, which might be hard to detect for simple distance functions such as the mean squared error or Euclidean distance functions. This is especially different for a simple dataset such as MNIST, where the crucial part of the images are central, and the difference between the classes is easy to observe.

We also observe the difference between the autoencoder models and the (β)-variational autoencoder models. We notice that the (β)-VAE outperform the AE models for both datasets. However, the difference between the two types of models is bigger for the MCMC dataset, where the (β)-VAE models outperform their AE counterparts fairly significantly. This shows the importance of the Kullback-Leibler divergence term. We also notice that the β value has to be significantly higher for the DD dataset to obtain satisfactory results than for the MCMC dataset. This is because of the scaling done in Equation 2. The denominator in this equation is a lot bigger

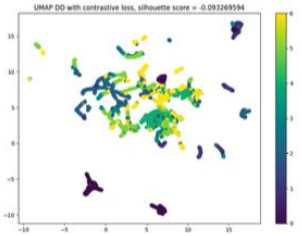
for the DD dataset, as its height and width are larger. This means that a higher β value is necessary to give the KL divergence term the same influence. The KL divergence term helps in creating clusters that have a Gaussian distribution. The importance of the Lagrangian multiplier β is alsoshown multiple times in the figures. As was shown, the models with either clustering or contrastive loss barely outperformed the baselines for both datasets. We suspect, once again, this is due to the datasets' complexity, especially so because it does succeed in the MNIST dataset. For the ensemble datasets, the models succeed in the wanted contrasted effect in the latent space. However, it is too difficult to then project this successfully from a latent space size of 256, for example, to 2D. Unfortunately, too much crucial information is lost in the process.



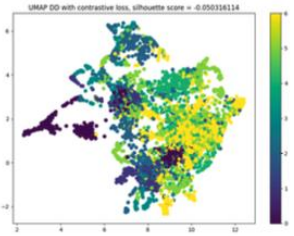
(a) AE model with latent space= 32.



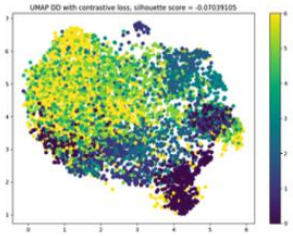
(b) AE model with latent space= 256.



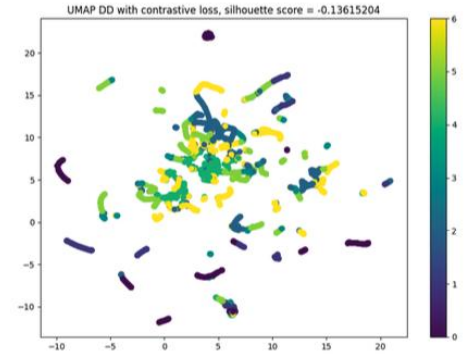
(c) $\beta(2)$-VAE model.



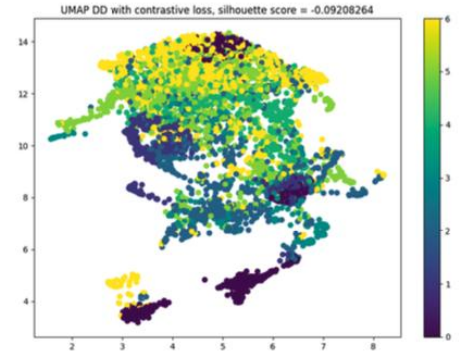
(d) $\beta(25)$-VAE model.



(e) $\beta(75)$-VAE model.



(f) $\beta(30)$-VAE model with latent space= 64.



(g) $\beta(30)$-VAE model with latent space= 256.

Figure 16: (β)-variational) autoencoder models with contrastive loss UMAP projections on DD ensemble dataset.

5. Conclusion

In conclusion, in this project, we have shown different approaches to autoencoder-based semi-supervised dimensionality reduction and clustering for scientific ensembles. We have done this first by generating pseudo-labels for the unlabelled part of the dataset, after which we implemented the soft silhouette score, a clustering loss type, and contrastive loss. Then, with these learning objectives, we compared different configurations of autoencoders and (β)-variational autoencoders, and tested these on two large scientific ensemble datasets, the drop dynamics dataset and the Markov chain Monte Carlo dataset. We projected the latent space to a 2D representation and we obtained results that showed that although it is possible to get better results using clustering or contrastive loss, the improvements are marginal. In general, the β-VAE models outperformed the AE models. For the Markov chain Monte Carlo ensemble, better results were obtained than for the drop dynamics dataset, which is explained by DD's higher number of labels and its respective complexity. Further research has to be done to improve these results.

6. Future work

In this section, we will take a look at improvements that could be made to this project, as well as future work that could be implemented. We note some that we believe could be the most promising ones:

- Stopping overfitting with the clustering loss: Clustering loss showed promising results with the simple MNIST dataset. However, it ran into many issues with the scientific ensemble datasets. Some slight changes to the function might make it perform better.
- Combining clustering and contrastive loss with reconstruction loss: Because both loss functions show promising results, it could be interesting to couple them and perform further optimization.
- Improve on drop dynamics results: In this project, it was difficult to significantly improve the results of the drop dynamics ensemble dataset because of its complicated nature. More research should be done to improve upon this.
- Try out other model architectures: Although the models generally outperformed the baselines, there might be some model architecture that outperforms it all. Furthermore, different types of models could be looked into.
- Utilize a different type of loss function: Another way to improve could be to utilize a different type of loss function than contrastive or clustering loss.
- Further analysis of the β value: The Lagrangian multiplier β showed to be quite influential. A larger search for the best value for this could be done. An adaptive weight for this value could also be implemented, changing the value while training.

Reference list

1. *Junpeng Wang, Subhashis Hazarika, Cheng Li, and Han-Wei Shen. Visualization and visual analysis of ensemble data: A survey. IEEE Transactions on Visualization and Computer Graphics, 25(9):2853–2872, 2019.*
2. *Julie Jebeile and Michel Crucifix. Multi-model ensembles in climate science: Mathematical structures and expert judgements. Studies in History and Philosophy of Science Part A, 83:44–52, 2020.*
3. *Hamid Gadirov, Gleb Tkachev, Thomas Ertl, and Steffen Frey. Evaluation and selection of autoencoders for expressive dimensionality reduction of spatial ensembles. In Advances in Visual Computing, pp. 222–234, 2021. Springer International Publishing.*
4. *Sebastian Reuschen, Teng Xu, and Wolfgang Nowak. Bayesian inversion of hierarchical geostatistical models using a parallel-tempering sequential gibbs MCMC. Advances in Water Resources, 141:103614, 07 2020.*
5. *Anne Geppert, Dimitrios Chatzianagnostou, Christian Meister, Hassan Gomaa, G. Lamanna, and Bernhard Weigand. Classification of impact morphology and splash- ing/deposition limit for n-hexadecane. Atomization and Sprays, 26, 01 2015.*
6. *Eugen-Richard Ardelean, Andreea Coporˆıie, Ana-Maria Ichim, Mihaela Dˆınş, oreanu, and Raul Cristian Mures, an. A study of autoencoders as a feature extraction technique for spike sorting. PLOS ONE, 18(3):1–29, 03 2023.*
7. *Dillip Ranjan Nayak, Neelamadhab Padhy, Pradeep Kumar Mallick, and Ashish Singh. A deep autoencoder approach for detection of brain tumor images. Computers and Electrical Engineering, 102:108238, 2022.*
8. *Min Chen, Xiaobo Shi, Yin Zhang, Di Wu, and Mohsen Guizani. Deep feature learning for medical image analysis with convolutional autoencoder neural network. IEEE Transactions on Big Data, PP:1–1, 06 2017.*
9. *Enoch Solomon, Abraham Woubie, and Eyael Solomon Emiru. Autoencoder based face verification system, 2024.*
10. *Huiyuan Tian, Li Zhang, Shijian Li, Min Yao, and Gang Pan. Pyramid-vae-gan: Trans- ferring hierarchical latent variables for image inpainting. Computational Visual Media, 9:827–841, 07 2023.*
11. *Xiuxi Wei, Zhihui Zhang, Huajuan Huang, and Yongquan Zhou. An overview on deep clustering. Neurocomputing, 590:127761, 2024.*
12. *Georgios Vardakas, Ioannis Papakostas, and Aristidis Likas. Deep clustering using the soft silhouette score: Towards compact and well-separated clusters, 2024.*
13. *Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis, 2016.*
14. *Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. 08 2017.*
15. *Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering, 2017.*
16. *Hao Zhou, Ke Yu, Xuan Zhang, Guanlin Wu, and Anis Yazidi. Contrastive autoencoder for anomaly detection in multivariate time series. Information Sciences, 610:266–280, 2022.*
17. *Dawei Luo, Heng Zhou, Joonsoo Bae, and Bom Yun. Combining contrastive learning with auto-encoder for out-of-distribution detection. Applied Sciences, 13:12930, 12 2023.*
18. *Alejo Lopez-Avila and V´ıctor Su´arez-Paniagua. Combining denoising autoencoders with contrastive learning to fine-tune transformer models, 2024.*
19. *Zeyu Cao, Xiaorun Li, Yueming Feng, Shuhan Chen, Chaoqun Xia, and Liaoying Zhao. Contrastnet: Unsupervised feature learning by autoencoder and prototypical contrastive learning for hyperspectral imagery classification. Neurocomputing, 460:71–83, October 2021.*
20. *Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training, 2021.*
21. *Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. Aiche Journal, 37:233–243, 1991.*
22. *Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.*
23. *Irina Higgins, Lo¨ıc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In International Conference on Learning Representations, 2016.*
24. *Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.*
25. *Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann ma- chines vinod nair. volume 27, pages 807–814, 06 2010.*

26. *Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20:53–65, 1987.*
27. *R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), volume 2, pages 1735–1742, 2006.*
28. *Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction, 2020.*
29. *Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differ- entiation in pytorch. In NIPS-W, 2017.*
30. *RUG. High performance computing cluster. https://www.rug.nl/society-business/ centre-for-information-technology/research/services/hpc/facilities/ peregrine-hpc-cluster, 2023.*
31. *Yann LeCun and Corinna Cortes. The MNIST database of handwritten digits. 2005.*
32. *Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhut- dinov. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(56):1929–1958, 2014.*
33. *H. Gadirov, Q. Wu, D. Bauer, K. L. Ma, J. B. T. M. Roerdink, & S. Frey, (2025). HyperFLINT: Hypernetwork-based Flow Estimation and Temporal Interpolation for Scientific Ensemble Visualization. Computer Graphics Forum, 44(3), Article e70134. https://doi.org/10.1111/cgf.70134*
34. *H. Gadirov, J. B. T. M. Roerdink and S. Frey, FLINT: Learning-Based Flow Estimation and Temporal Interpolation for Scientific Ensemble Visualization, in IEEE Transactions on Visualization and Computer Graphics, vol. 31, no. 10, pp. 7970-7985, Oct. 2025, doi: 10.1109/TVCG.2025.3561091.*
35. *H. Gadirov. Autoencoder-based feature extraction for ensemble visualization. Master's thesis, University of Stuttgart, 2020. url http://dx.doi.org/10.18419/opus-11304.*
36. *Hamid Gadirov, (2025). Machine Learning for Scientific Visualization: Ensemble Data Analysis. [Thesis fully internal (DIV), University of Groningen]. University of Groningen. https://doi.org/10.33612/diss.1402847307*
37. *Z. Yin, H. Gadirov, J. Kosinka, and S. Frey. ENTIRE: Learning-based Volume Rendering Time Prediction. arXiv preprint arXiv:2501.12119, 2025.*
38. *Bauer, D., Wu, Q., Gadirov, H., & Ma, K. L. (2025). GSCache: Real-Time Radiance Caching for Volume Path Tracing using 3D Gaussian Splatting. IEEE transactions on visualization and computer graphics, PP, 10.1109/TVCG.2025.3634634. Advance online publication. https://doi.org/10.1109/TVCG.2025.3634634*
39. *Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). Lora: Low-rank adaptation of large language models. ICLR, 1(2), 3.*
40. *Manuel, L., Gadirov, H., & Frey, S. (2025). Autoencoder-based semi-supervised dimensionality reduction and clustering for scientific ensembles (Research internship project). arXiv. https://arxiv.org/abs/2512.11145*
41. *H. Gadirov. Report on container technology for the ATLAS TDAQ system*
42. *Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. Linux Journal, 2014(239), 2.*

JOURNAL OF BAKU ENGINEERING UNIVERSITY- MATHEMATICS AND COMPUTER SCIENCE
2025. Volume 9, Number 2 *Pages 137-143*

UDC:517.54
DOI: https://doi.org/10.30546/09090.2025.510.1013

PARTIAL SUMS OF MEROMORPHIC FUNCTIONS LINKED WITH $q-$ DIFFERENTIAL OPERATOR

Erhan DENİZ*
Kafkas University,
ediniz36@gmail.com
Kars-Türkiye

ARTICLE INFO

Article history
Received:2026-01-09
Received in revised form:2026-01-12
Accepted:2026-01-16
Available online

Keywords:
Meromorphic functions;
Partial sum,;
q-calculus.
2010 Mathematics Subject Classifications: 30C45

ABSTRACT

The paper presents the introduction of a novel linear differential operator for meromorphic functions associated with $q-$ calculus. By means of this operator, a new subclass of meromorphic functions is defined and investigated in detail. The study focuses on deriving sufficient conditions that guarantee membership in this subclass and on establishing several analytic and geometric properties of the associated functions. Furthermore, the behavior of functions in the defined subclass is examined through the ratios of meromorphic functions to their sequences of partial sums. Various results describing these ratios are obtained, highlighting convergence and structural features. The findings enrich the theory of meromorphic functions and demonstrate the effectiveness of $q-$ calculus-based operators in function theory.

1. Introduction

Quantum calculus known as *q*-calculus is sometimes described as limitless calculus. It substitutes a difference operator for the classical derivative, allowing for the manipulation of sets of non-differentiable functions. Quantum difference operators play an intriguing role in a variety of mathematical fields, including the geometric function theory, calculus of variations, and relativity theory (see [1], [11], [19]). Gasper and Rahman's, and Kac and Cheung's books [9, 12] cover a large number of fundamental aspects of *q*-calculus.

We use the symbol $\sum$ to represent the set of functions f that takes the following form

$$f(z)=\frac{1}{z}+\sum_{n=1}^{\infty}a_n z^n \qquad (1)$$

that are analytic in the punctured open unit disk $\mathrm{U}^*=\{z:z\in\mathrm{C}:0<|z|<1\}=\mathrm{U}\setminus\{0\}$.

Tang et al. [20] introduced the *q*-derivative $D_q(f(z))$ for meromorphic functions, defined as follows:

$$D_q f(z)=\frac{f(z)-f(qz)}{(1-q)z}=-\frac{1}{qz^2}+\sum_{n=1}^{\infty}[n]_q\, a_n z^{n-1},\quad (0<q<1). \quad (2)$$

We start with the definitions and various results from the $q-$analysis, including the $q-$factorial $[n]_q!$ for every non-negative integer $n \in \mathrm{N}$, which is characterized by

$$[n]_q! = \begin{cases} [n]_q [n-1]_q \cdots [2]_q [1]_q; & (n \in \mathrm{N} \setminus \{1\}) \\ 1; & (n=1) \\ 0; & (n=0), \end{cases}$$

where

$$[n]_q = \frac{1-q^n}{1-q}. \qquad (3)$$

If $q \to 1^-$, then $[n]_q \to n$ and $\lim_{q \to 1^-} D_q f = f'$.

More recently, Ali et.al [2] introduced and studied the $q-$operator

$\mathsf{D}_{\rho,\chi}^{r,q} : \Sigma \to \Sigma \quad (r \in \mathbb{N}_0, \rho, \chi \geq 0, 0 < q < 1)$ defined by

$$\mathsf{D}_{\rho,\chi}^{0,q} f(z) := \mathsf{D}_{\rho,\chi}^{q} f(z) = f(z)$$

$$\mathsf{D}_{\rho,\chi}^{1,q} f(z) = (1-\chi) f(z) + \frac{\chi}{[\rho]_q z^{\rho}} D_q (z^{\rho+1} f(z)) = \frac{1}{z} + \sum_{n=1}^{\infty} \frac{[\rho]_q + \chi\left([n+\rho+1]_q - [\rho]_q\right)}{[\rho]_q} a_n z^n$$

……

$$\mathsf{D}_{\rho,\chi}^{r,q} f(z) = (1-\chi) \mathsf{D}_{\rho,\chi}^{r-1,q} f(z) + \frac{\chi}{[\rho]_q z^{\rho}} D_q \left(z^{\rho+1} \mathsf{D}_{\rho,\chi}^{r-1,q} f(z) \right).$$

Thus, we have the power series

$$\mathsf{D}_{\rho,\chi}^{r,q} f(z) = \frac{1}{z} + \sum_{n=1}^{\infty} \left(\frac{[\rho]_q + \chi\left([n+\rho+1]_q - [\rho]_q\right)}{[\rho]_q} \right)^r a_n z^n. \qquad (4)$$

Building on the research conducted in [14,18], we introduce a subclass denoted as $MS_q[r, \rho, \chi; \mathrm{A}, \mathrm{B}]$, which is defined using the operator $\mathsf{D}_{\rho,\chi}^{r,q} f(z)$ in the following manner:

Definition 1.1. A function $f \in \Sigma$ is said to belong to the class $MS_q[r, \rho, \chi; \mathrm{A}, \mathrm{B}]$, if

$$\left| \frac{qzD_q\left(\mathsf{D}_{\rho,\chi}^{r,q} f(z)\right) + \mathsf{D}_{\rho,\chi}^{r,q} f(z)}{\mathrm{B}qzD_q\left(\mathsf{D}_{\rho,\chi}^{r,q} f(z)\right) + \mathrm{A}\mathsf{D}_{\rho,\chi}^{r,q} f(z)} \right| \leq 1 \quad \left(z \in \mathrm{U}^*\right), \quad (5)$$

where $r \in \mathbb{N}_0, \rho, \chi \geq 0, 0 < q < 1$ and $-1 \leq \mathrm{B} < \mathrm{A} \leq 1$.

Furthermore, a function

$$f(z) = \frac{1}{z} + \sum_{n=1}^{\infty} a_n z^n \quad \left(a_n > 0, z \in \mathrm{U}^*\right), \qquad (6)$$

belongs to the class $TMS_q\left[r,\rho,\chi;\mathrm{A},\mathrm{B}\right]$, if it meets the requirement stated in equation (5).

It's worth noting that the previous definition is primarily inspired by the latest research by Morga [14] and Srivastava et al. [18]

This paper's primary aim is to give partial sums of functions that belong to the classes $MS_q\left[r,\rho,\chi;\mathrm{A},\mathrm{B}\right]$ and $TMS_q\left[r,\rho,\chi;\mathrm{A},\mathrm{B}\right]$. Unless specified otherwise, we'll assume that $r\in\mathbb{N}_0$, $\rho,\chi\geq 0$, $0<q<1$ and $-1\leq \mathrm{A}<\mathrm{B}\leq 1$ in this paper.

2. Coefficient Bonds

This section outlines the process of the sufficient conditions based on coefficient estimates for functions f that are part of the subclasses $MS_q\left[r,\rho,\chi;\mathrm{A},\mathrm{B}\right]$ and $TMS_q\left[r,\rho,\chi;\mathrm{A},\mathrm{B}\right]$.

Theorem 2.1. Let $f\in\sum$ as in (1) and satisfies the inequality

$$\sum_{n=1}^{\infty}\left((\mathrm{B}+1)q[n]_q+\mathrm{A}+1\right)\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r|a_n|<\mathrm{A}-\mathrm{B}, \quad (7)$$

then $f\in MS_q\left[r,\rho,\chi;\mathrm{A},\mathrm{B}\right]$.

Proof. To prove that $f\in MS_q\left[r,\rho,\chi;\mathrm{A},\mathrm{B}\right]$, we must show that

$$\left|qzD_q\left(\mathrm{D}_{\rho,\chi}^{r,q}f(z)\right)+\mathrm{D}_{\rho,\chi}^{r,q}f(z)\right|-\left|\mathrm{B}qzD_q\left(\mathrm{D}_{\rho,\chi}^{r,q}f(z)\right)+\mathrm{A}\mathrm{D}_{\rho,\chi}^{r,q}f(z)\right|\leq 0. \quad (8)$$

By using the triangle inequality to the left of side of (8), we have

$$\begin{aligned}
&\left|qzD_q\left(\mathrm{D}_{\rho,\chi}^{r,q}f(z)\right)+\mathrm{D}_{\rho,\chi}^{r,q}f(z)\right|-\left|\mathrm{B}qzD_q\left(\mathrm{D}_{\rho,\chi}^{r,q}f(z)\right)+\mathrm{A}\mathrm{D}_{\rho,\chi}^{r,q}f(z)\right|\\
&=\left|\sum_{n=1}^{\infty}\left(q[n]_q+1\right)\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r a_nz^n\right|\\
&-\left|(\mathrm{A}-\mathrm{B})\frac{1}{z}+\sum_{n=1}^{\infty}\left(\mathrm{A}+\mathrm{B}q[n]_q\right)\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r a_nz^n\right| \quad (9)\\
&\leq\sum_{n=1}^{\infty}\left((\mathrm{B}+1)q[n]_q+\mathrm{A}+1\right)\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r|a_n|-(\mathrm{A}-\mathrm{B}).
\end{aligned}$$

Thus, from (7) and (9), we obtain that

$$\sum_{n=1}^{\infty}\left((\mathrm{B}+1)q[n]_q+\mathrm{A}+1\right)\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r|a_n|-(\mathrm{A}-\mathrm{B})\leq 0,$$

which proves the theorem.

Theorem 2.2. Let $f(z)=\frac{1}{z}+\sum_{n=1}^{\infty}a_nz^n$ $(a_n\geq 0)$ in U^*. Then $f(z)\in TMS_q\left[r,\rho,\chi;\mathrm{A},\mathrm{B}\right]$, if and only if inequality (7) is satisfied. The result is sharp for the function $f(z)$, which is defined as

$$f(z)=\frac{\mathrm{A}-\mathrm{B}}{\left((\mathrm{B}+1)q[n]_q+\mathrm{A}+1\right)\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r}z^n,\quad (n\geq 1). \tag{10}$$

Proof. Considering Theorem 2.1, it's enough to prove the validity of the "if" component.

Assume that $f(z)\in TMS_q[r,\rho,\chi;\mathrm{A},\mathrm{B}]$. Then, we have

$$\mathrm{Re}\left\{\frac{qzD_q\left(\mathrm{D}^{r,q}_{\rho,\chi}f(z)\right)+\mathrm{D}^{r,q}_{\rho,\chi}f(z)}{\mathrm{B}qzD_q\left(\mathrm{D}^{r,q}_{\rho,\chi}f(z)\right)+\mathrm{A}\mathrm{D}^{r,q}_{\rho,\chi}f(z)}\right\}\geq -1\quad \left(z\in \mathrm{U}^*\right). \tag{11}$$

Since $\mathrm{Re}\, f(z)\leq |f(z)|$ for all $z\in \mathrm{U}^*$, then

$$\mathrm{Re}\left\{\frac{\sum_{n=1}^{\infty}\left(q[n]_q+1\right)\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r a_n z^{n+1}}{\mathrm{A}-\mathrm{B}+\sum_{n=1}^{\infty}\left(\mathrm{A}+\mathrm{B}q[n]_q\right)\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r a_n z^{n+1}}\right\}\leq 1, \tag{12}$$

for all z and the above equation is true. By letting $z\to 1^-$ on the real axis, we have the following inequality

$$\sum_{n=1}^{\infty}\left(q[n]_q+1\right)\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r a_n$$

$$\leq \mathrm{A}-\mathrm{B}+\sum_{n=1}^{\infty}\left(\mathrm{A}+\mathrm{B}q[n]_q\right)\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r a_n .$$

Thus, we get the required inequality

$$\sum_{n=1}^{\infty}\left(q[n]_q(\mathrm{B}+1)+(\mathrm{A}+1)\right)\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r a_n\leq \mathrm{A}-\mathrm{B}.$$

This concludes the demonstration of our theorem.

3. Partial Sums

Inspired by previous studies that used the conventional idea of partial sums for analytic functions, such as Goodman [10], Silverman [16, 17], Murugusundaramoorthy and Velayudam [13], Darus and Ibrahim [5], Altıntas and Owa [3], Elhaddad et. al [8], and recently Deniz and co-auhors [4,6,7,15]. Our results related with partial sums as follows:

Theorem 3.1. Let $-1<\mathrm{A}\leq 0$. If $f\in\sum$ of the form (1) and

$s_k(z)=\frac{1}{z}+\sum_{n=1}^{k-1}a_n z^n\quad (k\geq 2)$. Suppose that

$$\sum_{n=1}^{\infty} c_n |a_n| \le 1, \quad (13)$$

where

$$c_n = \frac{\left((1+\mathrm{A})+q[n]_q(1+\mathrm{B})\right)}{(\mathrm{A}-\mathrm{B})}\left(\frac{[\rho]_q+\chi\left([n+\rho+1]_q-[\rho]_q\right)}{[\rho]_q}\right)^r.$$

Then, we have

1) $f(z)\in TMS_q[r,\rho,\chi;\mathrm{A},\mathrm{B}]$.

2) $$\mathrm{Re}\left\{\frac{f(z)}{s_k(z)}\right\} > 1-\frac{1}{c_{k-1}}. \quad (14)$$

3) $$\mathrm{Re}\left\{\frac{s_k(z)}{f(z)}\right\} > \frac{c_{k-1}}{1+c_{k-1}}. \quad (15)$$

The estimates are sharp.

Proof. 1): It is obvious that $\frac{1}{z}\in TMS_q[r,\rho,\chi;\mathrm{A},\mathrm{B}]$. Thus from Theorem 2.1 and the condition (13), we have $N_{\mu,q}\left(\frac{1}{z}\right)\subseteq TMS_q[r,\rho,\chi;\mathrm{A},\mathrm{B}]$. This gives $f(z)\in TMS_q[r,\rho,\chi;\mathrm{A},\mathrm{B}]$.

2): It is easy to see that $1<c_k<c_{k+1}$. Thus

$$\sum_{n=1}^{k-2}|a_n|+c_{k+1}\sum_{n=k-1}^{\infty}|a_n| \le \sum_{n=1}^{\infty} c_n|a_n| \le 1. \quad (16)$$

Let

$$h_1(z)=c_{k-1}\left\{\frac{f(z)}{s_k(z)}-\left(1-\frac{1}{c_{k-1}}\right)\right\}=1+\frac{c_{k-1}\sum_{n=k-1}^{\infty}a_n z^{n+1}}{1+\sum_{n=1}^{k-2}a_n z^{n+1}}.$$

It follows from (16) that

$$\left|\frac{h_1(z)-1}{h_1(z)+1}\right| \le \frac{c_{k-1}\sum_{n=k-1}^{\infty}|a_n|}{2-2\sum_{n=1}^{k-2}|a_n|-c_{k-1}\sum_{n=k-1}^{\infty}|a_n|} \le 1 \quad \left(z\in \mathrm{U}^*\right).$$

Therefore we obtain the inequality (14).

If we take

$$f(z)=\frac{1}{z}-\frac{z^{k-1}}{c_{k-1}}, \quad (17)$$

then

$$f(z)=1-\frac{z^{k}}{c_{k-1}}\to 1-\frac{1}{c_{k-1}} \text{ as } z\to 1^{-}.$$

This demonstrates that the bound in (14) is best possible for any k.

3): Similarly, assuming that we take

$$h_2(z)=\left(1+c_{k+1}\right)\left\{\frac{s_k(z)}{f(z)}-\left(\frac{c_{k-1}}{1+c_{k-1}}\right)\right\}=1+\frac{\left(1+c_{k-1}\right)\sum_{n=k-1}^{\infty}a_n z^{n+1}}{1+\sum_{n=0}^{\infty}a_n z^{n+1}}.$$

Then, we decude that

$$\left|\frac{h_2(z)-1}{h_2(z)+1}\right|\le\frac{\left(1+c_{k-1}\right)\sum_{n=k-1}^{\infty}|a_n|}{2-2\sum_{n=1}^{k-2}|a_n|+\left(1-c_{k-1}\right)\sum_{n=k-1}^{\infty}|a_n|}\le 1,\quad \left(z\in \mathrm{U}^{*}\right),$$

which yields (15). The estimate (15) is sharp with the extremal function $f(z)$ given by (17).

4. Conclusion

In the current study, a new subclass of meromophic functions associated with a $q-$ differential operator has been introduced and investigated. Main results related with coefficient bounds and partial sums for functions belonging to this subclass

REFERENCE LIST

[1] Alatawi, A., Darus, M., Alamri, B. (2023). Applications of Gegenbauer polynomials for subfamilies of bi-univalent functions involving a Borel distribution-type Mittag-Leffler function. *Symmetry*, 15, 785.

[2] Ali, E. E., El-Ashwah, R. M., Albalahi, A. M. (2025). Geometric applications for meromorphic functions involving q-linear operators. Mathematics, 10(12), 30990-31009.

[3] Altintas, O., Owa, S. (2009). Neighborhoods of cartain analytic functions with negative coefficients. *Int. J. Math. Math. Sci.*, 19, 797–800.

[4] Çağlar, M., Deniz, E. (2015). Partial sums of normalized Lommel functions, ***Math. Inequal. Appl.,*** 18(3), 1189–1199.

[5] Darus, M., Ibrahim, R.W. (2009). On partial sums of generalized differential operator. *Proc, Pakistan Acad. Sci.*, 46, 209–215.

[6] Deniz, E., Orhan, H. (2011). Certain subclasses of multivalent functions defined by new multiplier transformations, *Arab. J. Sci. Eng.*, 36(6), 1091-1112.

[7] Deniz, E., Özkan, Y., Kazımoğlu, S., Senger, Ö. (2023). Certain subclasses of p-valent functions defined by multiplier transformations, *Afr. Mat.*, 34 (1), 19pp.

[8] Elhaddad, S., Aldweby H., Darus, M. (2018). Neighborhoods of certain classes of analytic functions defined by a generalized differential operator involving Mittag-Leffler function. *Acta Univ. Apulensis.*, 18, 1–10.

[9] Gasper, G., Rahman, M. (2004). Basic hypergeometric series. *Cambridge university press*.

[10] Goodman, A.W. (1957). Univalent functions and non analytic curves. *Proc. Amer. Math. Soc.*, 8, 598–601.

[11] Hadi, S.H., Darus, M., Alb Lupaş, A. (2023). A class of Janowski-type (p, q)-convex harmonic functions involving a generalized q-Mittag–Leffler function. *Axioms*, 12, 190.

[12] Kac, V., Cheung, P. (2002). Quantum Calculus. *Springer*, New York.

[13] Murugusundaramoorthy, G., Velayudam, S.V.S. (2005). Neighborhoods and Partial sums of meromorphic Univalent Functions. *Mapana Journal of Sciences*, 4, 22–31.

[14] Morga, L.M. (1990). Meromorphic multivalent functions with positive coefficients. *Math. Japon.,* 35, 01–11.

[15] Orhan, H., Raducanu, D., Deniz, E. (2011). Subclasses of meromorphically multivalent functions defined by a differential operator, *Comput. Math. Appl.,* 61(4), 966-979.

[16] Silverman, H. (1995). Neighborhoods of a classes of analytic function. *Far East J. Math.Sci*., 3, 165–169.

[17] Silverman, H. (1997). Partial sums of starlike and convex functions. *J. Math. Anal. Appl.*, 209, 221–227.

[18] Srivastava, H.M., Hossen, H.M., Aouf, M.K. (1996). A unified presentation of some classes of meromorphically multivalent functions. *Comput. Math. Appl.,* 38, 63–70.

[19] Srivastava, H.M., Hadi, S.H., Darus, M. (2023). Some subclasses of p-valent γ-uniformly type q-starlike and q-convex functions defined by using a certain generalized q-Bernardi integral operator. *Rev. Real Acad. Cienc. Exactas Fis. Nat. Ser. A-Mat.,* 117, 65.

[20] Tang, H., Zayed, H., Mostafa, A., Aouf, M. (2018). Fekete-Szego Problems for Certain Classes of Meromorphic Functions Using q-Derivative Operator. *J. Math. Res. Appl.,* 38 (3), 236–246.

2025. Volume 9, Number 2 Pages 144-151

UDC:517.54

DOI: https://doi.org/10.30546/09090.2025.510.1023

MAJORIZATION RESULTS FOR A SUBCLASS OF MEROMORPHIC FUNCTIONS INVOLVING $q-$ LINEAR OPERATORS

Sercan KAZIMOĞLU*

Kafkas University,

sercan.kazimoglu@kafkas.edu.tr

Kars-Türkiye

ARTICLE INFO

Article history
Received:2026-01-09
Received in revised form:2026-01-12
Accepted:2026-01-15
Available online

Keywords:
Meromorphic function;
Majorization;
$q-$ linear operators.

2010 Mathematics Subject Classifications: 30C45

ABSTRACT

In this work, we investigate several majorization results for a subordination class of meromorphic functions of complex order defined by a $q-$linear operator. The study focuses on establishing sufficient conditions under which majorization properties hold within this function class. By employing techniques from geometric function theory and operator theory, new inequalities and relationships are derived. Moreover, we present a number of new results that naturally arise from the main theorems and are stated explicitly in the form of corollaries. These corollaries highlight special cases and further implications of the obtained results. The findings contribute to the broader understanding of subordination, majorization, and $q-$calculus in the theory of meromorphic functions.

1. Introduction

Let M represent the class of meromorphic funtions f in the form of

$$f(z)=\frac{1}{z}+\sum_{n=0}^{\infty}a_n z^n, \quad (18)$$

which are analytic in the punctured disc $\overset{\circ}{\mathrm{U}}=\{z:0<|z|<1\}=\mathrm{U}/\{0\}$, where $\mathrm{U}=\overset{\circ}{\mathrm{U}}\cup\{0\}$. For the two functions $f(z)$ and $g(z)$ belonging to the class M, there exists a Schwartz function w, which is analytic in U with $|w(z)|\le|z|$ and $w(0)=0$, such that $f(z)=g(w(z))$, and the function f is subordinate to g, written as $f\prec g$. The following relationship holds if g is univalent:

$$f\prec g \Leftrightarrow f(0)=g(0), \text{ and } f\left(\overset{\circ}{\mathrm{U}}\right)\subseteq g\left(\overset{\circ}{\mathrm{U}}\right). \quad (19)$$

Because of its use in a variety of mathematical sciences, the study of $q-$calculus (quantum calculus) has fascinated and motivated many scholars. One of the primary contributors among

all the mathematicians who introduced the concept of $q-$ calculus theory was Jackson [1,2]. The formulation of this concept is widely used to investigate the natüre of different structures of function theory, such as $q-$ calculus was used in other branches of mathematics.

Though the authors of the first article [3] discussed the geometric nature $q-$ starlike functions, Srivastava [4] laid a solid foundation fort he use of $q-$ calculus in the context of function theory. Also, in [5], Srivastava provided a brief overview of basic or $q-$ calculus operators and fractional $q-$ calculus operators, as well as their applications in the geometric function theory of complex analysis. Later, the authors [6-8] investigated a number of useful properties for the newly defined $q-$ linear differential operator. While Srivastava et al. [9] introduced a generalized operator for meromorphic harmonic functions by using the idea of convolution.

Let $0<q<1.$ For any nonnegative integer n, the $q-$ integer number n is defined by

$$[n]_q=\frac{1-q^n}{1-q}=1+q+q^2+...+q^{n+1},\ [0]_q=0.$$

In general, we will denote

$$[\delta]_q=\frac{1-q^\delta}{1-q},$$

for a noninteger number δ. Also, the $q-$ number shifted factorial is defined by

$$[n]_q!=[n]_q[n-1]_q...[2]_q[1]_q,[0]_q!=1.$$

Clearly,

$$\lim_{q\to 1^-}[n]_q=n \text{ and } \lim_{q\to 1^-}[n]_q=n!.$$

Let $a,q\in\mathbb{C}\ (|q|<1)$ and $n\in\mathbb{C}_0=\mathbb{C}\cup\{0\}$. Then the $q-$ shifted factorial $(a;q)_n$ is defined by

$$(a;q)_0=1,\ (a;q)_n=\prod_{j=1}^{n}\left(1-aq^{j-1}\right),\ n\in\mathbb{C}.$$

Let $x\in\mathbb{C}-\{-n:n\in\mathbb{C}_0\}$. Then $q-$ gamma function is as follows:

$$\Gamma_q(x)=\frac{(q;q)_\infty}{(q^x;q)_\infty}(1-q)^{1-x},\ 0<q<1.$$

In geometric function theory, operators play an important role. Many authors present differential and integral operators, for example ([10, 11, 12, 13]). For a function $f\in M$ given by (18), the $q-$ derivative (or $q-$ difference) of a function $f(z)$ is defined by [14,15].

$$(D_qf)(z)=\begin{cases}\dfrac{f(z)-f(qz)}{z(1-q)}, & z\neq 0\\ f'(0), & z=0\end{cases} \qquad (20)$$

provided that $f'(0)$ exists. We can easily observe from the definition of (20) that $\lim_{q\to 1^-}\left(D_q f\right)(z) = f'(z)$.

Suppose that $q \in (0,1)$, then $q-$ analog derivative of f as

$$\left(D_q f\right)(z) = -\frac{1}{qz^2} + \sum_{n=1}^{\infty} [n]_q a_n z^n. \qquad (21)$$

Due to its use in numerous fields of mathematics and physics, the q−derivative operator Dq has fascinated and inspired many researchers. Jackson [16] was among the key contributors of all the scientists who introduced and developed the q−calculus theory. In 1991, Ismail [3] was the first to demonstrate a crucial link between geometric function theory and the q−derivative operator, but a solid and comprehensive foundation was provided in 1989 in a book chapter by Srivastava [5].

For $\varepsilon, \nu \geq 0,$ define the meromorphic q-analogue of Ruscheweyh operator $\mathrm{R}_{q,\nu}^{\varepsilon} : \Sigma \to \Sigma$ by Hadamard product (convolution)

$$\mathrm{R}_{q,\nu}^{m,\varepsilon} f(z) = \mathrm{R}_{q,\nu}^{m,\varepsilon}(z) * f(z) = \frac{1}{z} + \sum_{n=1}^{\infty} \left(\frac{[\varepsilon]_q + \nu\left([n+\varepsilon+1]_q - [\varepsilon]_q\right)}{[\varepsilon]_q} \right)^m a_n z^n \quad (r \in \square_0,\ 0 < q < 1), \qquad (22)$$

where

$$\mathrm{R}_{q,\nu}^{m,\varepsilon}(z) = \frac{1}{z} + \sum_{n=1}^{\infty} \left(\frac{[\varepsilon]_q + \nu\left([n+\varepsilon+1]_q - [\varepsilon]_q\right)}{[\varepsilon]_q} \right)^m z^n,$$

was introduced and studied by Ekram et al. [17].

In 1967, Mac Gregor [18] introduced the Notion of majorization as follows.

Definition 1. Let complex-valued functions f and g be analytic in U. We say that f is majorized by g in U and write

$$f(z) \square\ g(z) \quad (z \in \mathrm{U}) \qquad (23)$$

if there exists a function $\varphi(z)$ (complex-valued function in U,) satisfying

$$|\varphi(z)| \leq 1 \text{ and } f(z) = \varphi(z) g(z) \quad (z \in \mathrm{U}). \qquad (24)$$

Majorization (23) is closely related to the concept of quasi-subordination between analytic functions in U. Several researchers have published articles on this topic; for example, Tang et al. [19] gave the concept of majorization for subclasses of starlike functions based on the sine and cosine functions, Arif et al. [20] discussed majorization for various new defned classes, Cho et al. [21] obtained coefficient estimates for majorization, and Tang and Deng [22] defined the majorization problem connected with Liu-Owa integral operator and exponential function. This concept is also defned for $p-$valent function by Altintas and Srivastava [23] and for complex order by Altintas et al. [24].

The basic goal of this article is to examine and explain the idea of majorization in the context of the meromorphic function. Many researchers have shown their interest in this site. Goyal and Goswami [25, 26] studied this concept for majorization for meromorphic function with the integral operator, Tang et al. [19] discussed it for meromorphic sin and cosine functions, Bulut et al, Tang et al, and Janani [27, 28, 29] explained this concept for meromorphic multivalent functions, Rasheed et al. [30] investigated a majorization problem for the class of meromorphic spiral-like functions related with a convolution operator, and Panigrahi and El-Ashwah [31] defined majorization for subclasses of multivalent meromorphic functions through iterations and combinations of the Liu–Srivastava operator and Cho–Kwon–Srivastava operator and much more. In addition, there are several other articles on this topic [26, 32].

Here is the definition of our main function.

Definition 2. A function $f(z)\in \mathrm{M}$ is said to be in the class $\mathrm{M}\,\mathrm{S}_{\mathrm{J}}^{m}(\varepsilon,\nu,q,\gamma)$ of meromorphic functions of complex order $\gamma\neq 0$ in $\overset{\circ}{\mathrm{U}}$, if

$$1-\frac{1}{\gamma}\left[\frac{zqD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon}f(z)\right)}{\mathrm{R}_{q,\nu}^{m,\varepsilon}f(z)}+1\right]\prec\psi(z).$$

Now, we are going to choose a particular function instead of $\psi(z)$. This choice is

$$\psi(z)=\frac{1+Az}{1+Bz},\quad -1\le B<A\le 1,$$

and by applying the above-mentioned concept, we now consider the following class:

$$\mathrm{M}\,\mathrm{S}_{\mathrm{J}}^{m}(\varepsilon,\nu,q,\gamma)=\left\{f(z)\in\mathrm{M}:1-\frac{1}{\gamma}\left[\frac{zqD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon}f(z)\right)}{\mathrm{R}_{q,\nu}^{m,\varepsilon}f(z)}+1\right]\prec\frac{1+Az}{1+Bz}\right\}.$$

This class is related with well-known the Janowski class [33]. In the present work, we discussed majorization problem for the above-defined class of $\mathrm{M}\,\mathrm{S}_{\mathrm{J}}^{m}(\varepsilon,\nu,q,\gamma)$.

2. Main Results

We state the following $q-$ analogue of the result given by Nehari [34] and Salvakumaran et al. [35].

Lemma 1. (See [36]) If the function $\varphi(z)$ is analytic and $|\varphi(z)|<1$ in U, then

$$|D_q\varphi(z)|\le\frac{1-|\varphi(z)|^2}{1-|z|^2}. \tag{25}$$

Theorem 1. Let $-1\le B<A\le 1$, the function $f(z)\in\mathrm{M}$ and suppose $g\in\mathrm{M}\,\mathrm{S}_{\mathrm{J}}^{m}(\varepsilon,\nu,q,\gamma)$ if $\mathrm{R}_{q,\nu}^{m,\varepsilon}f(z)$ is majorized by $\mathrm{R}_{q,\nu}^{m,\varepsilon}g(z)$ in $\overset{\circ}{\mathrm{U}}$, i.e.,

$$\mathrm{R}_{q,\nu}^{m,\varepsilon}f(z)\ \square\ \mathrm{R}_{q,\nu}^{m,\varepsilon}g(z).$$

Then, for $|z| \le r_1$,

$$\left|qzD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} f(z)\right)\right| \le \left|qzD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)\right)\right|, \tag{26}$$

where r_1 is the smallest positive root of the following equation:

$$\left(1-r^2\right)\left(1-\left|\gamma(A-B)+B\right|r\right)-2rq\left(1+|B|r\right)=0. \tag{27}$$

Proof. Since $g \in \mathrm{M}\,\mathrm{S}_{\mathrm{J}}^{m}(\varepsilon,\nu,q,\gamma)$, we readily obtained from Definition 2 that

$$1-\frac{1}{\gamma}\left[\frac{zqD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)\right)}{\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)}+1\right] \prec \psi(z),$$

$z \in \overset{\circ}{\mathrm{U}}$ and

$$\psi(z)=\frac{1+Az}{1+Bz}.$$

By Lemma 1, there exists a bounded analytic function w in U and

$$1-\frac{1}{\gamma}\left[\frac{zqD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)\right)}{\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)}+1\right]=\frac{1+Aw(z)}{1+Bw(z)}, \tag{28}$$

with $w(\infty)=\infty$. From (28), we obtain

$$\frac{zqD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)\right)}{\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)}=-\frac{\left[\gamma(A-B)+B\right]w(z)+1}{1+Bw(z)}, \tag{29}$$

where $w(z)$ is the well-known class of bounded anlytic functions in U such that

$$|w(z)| \le |z| \quad (z \in U). \tag{30}$$

From (29) and making use of (30), we obtain

$$\left|\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)\right| \le \frac{1+|B||z|}{1-\left|\gamma(A-B)+B\right||z|}\left|zqD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)\right)\right|. \tag{31}$$

Since $\mathrm{R}_{q,\nu}^{m,\varepsilon} f(z)$ is majorized by $\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)$ in $\overset{\circ}{\mathrm{U}}$, from (24),

$$\mathrm{R}_{q,\nu}^{m,\varepsilon} f(z)=\varphi(z)\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z).$$

By applying $q-$ derivative on the previous equation write z as in [29] and then multiplying by qz, we have

$$\begin{aligned} qzD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} f(z)\right) &= qzD_q\varphi(z)\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)+qz\varphi(z)D_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)\right) \\ &= qzD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)\right)\left[\varphi(z)+\frac{D_q\varphi(z)\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)}{D_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)\right)}\right]. \end{aligned} \tag{32}$$

Noting that $\varphi(z)$ is the Schwartz function, so $\mathrm{Re}(\varphi(z))>0$ in $\overset{\circ}{\mathrm{U}}$, $\varphi(z)\neq 0$ for all $z\in\overset{\circ}{\mathrm{U}}$, satisfies the $q-$ analogue result given by [18] proved in Lemma 1.

Now, using (31) and (25) in (32), we have

$$\left|qzD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon}f(z)\right)\right|\leq\left|qzD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon}g(z)\right)\right|\left[|\varphi(z)|+\frac{1-|\varphi(z)|^2}{1-|z|^2}\frac{rq(1+|B||z|)}{1-|\gamma(A-B)+B||z|}\right].$$

Let us take $|z|=r<1$ and $|\varphi(z)|=\zeta,(0\leq\zeta\leq 1)$; we obtain

$$\left|qzD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon}f(z)\right)\right|\leq Y(r,\zeta)\left|qzD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon}g(z)\right)\right|.$$

We define

$$Y(r,\zeta)=\zeta+\frac{rq(1-\zeta^2)(1+|B|r)}{(1-r^2)(1-|\gamma(A-B)+B|r)},\quad(0\leq\zeta\leq 1,\ 0<r<1).$$

To determine r_1, it is sufficient to choose

$$r_1=\max\{r\in[0,1):Y(r,\zeta)\leq 1,\quad\forall\zeta\in[0,1]\},$$

equivalently,

$$r_1=\max\{r\in[0,1):Y^*(r,\zeta)\geq 0,\quad\forall\zeta\in[0,1]\},$$

where

$$Y^*(r,\zeta)=(1-r^2)(1-|\gamma(A-B)+B|r)-rq(1+\zeta)(1+|B|r).$$

Clearly, when $\zeta=1$, the above function $Y^*(r,\zeta)$ assumes its minimum value, namely,

$$\min\{Y^*(r,\zeta):\zeta\in[0,1]\}=Y^*(r,1)=\psi^*(r),$$

where

$$\psi^*(r)=(1-r^2)(1-|\gamma(A-B)+B|r)-2rq(1+|B|r).$$

Next, we obtained the following inequalities:

$$\psi^*(0)=1>0\text{ and }\psi^*(1)=-2q(1+|B|)<0,$$

there exists r_1 such that $\psi^*(r)\geq 0$ for all $r\in[0,r_1]$, where r_1 is the smallest positive root of (27). The proof of Theorem 1 is completed.

Putting $A=1$ and $B=-1$ in Theorem 1, we get the following result.

Corollary 1. Let the function $f(z)\in\mathrm{M}$ and suppose $g\in\mathrm{M}\,\mathrm{S}_{\mathrm{J}}^{m}(\varepsilon,\nu,q,\gamma)$ if $\mathrm{R}_{q,\nu}^{m,\varepsilon}f(z)$ is majorized by $\mathrm{R}_{q,\nu}^{m,\varepsilon}g(z)$ in $\overset{\circ}{\mathrm{U}}$, i.e.,

$$\mathrm{R}_{q,\nu}^{m,\varepsilon}g(z)\square\ \mathrm{R}_{q,\nu}^{m,\varepsilon}g(z).$$

Then, for $|z| \le r_2$,

$$\left|qzD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)\right)\right| \le \left|qzD_q\left(\mathrm{R}_{q,\nu}^{m,\varepsilon} g(z)\right)\right|,$$

where r_2 is the smallest positive root of the following equation $(1-r)(1-|2\gamma-1|r)-2rq=0.$

Conclusion

In this study, majorization results are obtained for a subclass of meromorphic functions defined by a $q-$ linear operator. The results generalize existing works and may lead to further studies through new subclasses or operators.

REFERENCE LIST

[1] Jackson, F. H. (1909). On $q-$ functions and a certain diference operator, Trans. R. Soc. Edinb., 46, 253–281.

[2] Jackson, F. H. (1910). On $q-$ definite integrals, Pure Appl. Math. Q., 41, 193–203.

[3] Ismail, M. E. H., Merkes, E. Styer, D. (1990). A generalization of starlike functions, Complex Variables, Theory and Appl.: An Int. J., 14, 77–84.

[4] Srivastava, H. M. (2020). Operators of basic (or q-) calculus and fractional q-calculus and their applications in geometric function theory of complex analysis, Iran J. Sci. Technol. Trans. A Sci., 44, 327–344.

[5] Srivastava, H. M. (1989). Univalent functions, fractional calculus, and associated generalized hypergeometric functions, in Univalent Functions, Fractional Calculus, and Their Applications, H. M. Srivastava and S. Owa, Eds., pp. 329–354, John Wiley, Sons, New York, NY, USA.

[6] Ahmad, B., Arif, M. (2018). New subfamily of meromorphic convex functions in circular domain involving q-operator, Int. J. Anal. Appl., 16, 75–82.

[7] Kota, W. Y., El-Ashwah, R. M., Breaz, N. (2025). Hadamard Product on Subclasses of Meromorphic Functions Involving q-Difference Operator. Journal of Function Spaces, 2025(1), 9959888.

[8] Arif, M., Haq, M. U., Liu, J. L. (2018). A subfamily of univalent functions associated with q-analogue of Noor integral operator, J. Funct. Spaces, 2018, 1–5.

[9] Srivastava, H. M., Arif, M., Raza, M. (2021). Convolution properties of meromorphically harmonic functions defined by a generalized convolution q-derivative operator, AIMS Mathematics, 6, 5869–5885.

[10] Abbas, M. I. (2020). Existence results and the Ulam stability for fractional differential equations with hybrid proportional Caputo derivatives, J. Nonlinear Funct. Anal. vol. 2020, Article ID 48.

[11] Libera, R. J. (1965). Some classes of regular univalent functions, A Proceedings of the American Mathematical Society, 16, 755–758.

[12] Ruscheweyh, S. (1975). New criteria for univalent functions, Proceedings of the American Mathematical Society, vol. 49, 109–115.

[13] Srivastava, H. M., Qureshi, M. I., Malik, S. H. (2021). Some hypergeometric transformations and reduction formulas for the Gauss function and their applications involving the Clausen function, J. Nonlinear Var. Anal.,5, 981–987.

[14] Abu Risha, M. H., Annaby, M. H., Ismail, Z. S., Mansour, Z. S. (2007). Linear q−difference equations, Zeitschrift fur Analysis und ihre Anwendungen, 26, 481–494.

[15] Gasper, G., Rahman, M. (1990). Basic Hypergeometric Series, Cambridge University Press, Cambridge.

[16] Jackson, F. H. (1905). The application of basic numbers to Bessel's and Legendre's functions, Proceedings of the London Mathematical Society, 3, 1–23.

[17] Ekram E. Ali, Rabha M. El-Ashwah, Abeer M. Albalahi. (2025). Geometric applications for meromorphic functions involving -linear operators. *AIMS Mathematics*, 10(12): 30990-31009.

[18] MacGregor, T. H. (1967). Majorization by univalent functions, Duke Math. J., 34, 95–102.

[19] Tang, H., Srivastava, H. M., Li, S. H., Deng, G. T. (2020). Majorization results for subclasses of starlike functions based on the sine and cosine functions, Bull. Iran. Math. Soc., 46, 381–388. 6 Proceedings of International Conference on Mathematical Advances and Applications.

[20] Arif, M., Ul-Haq, M., Barukab, O., Khan, Abullah, S. A. S. (2021). Majorization results for certain subfamilies of analytic functions, J. Funct. Spaces, 2021, 1–6.

[21] Cho, N. E., Oroujy, Z. E., Adegani, A., Ebadian, A. (2020). Majorization and coefcient problems for a general class of starlike functions, Symmetry, 12, 476.

[22] Tang, H., Deng, G. (2018). Majorization problems for two subclasses of analytic functions connected with the Liu–Owa integral operator and exponential function, J. Inequalities Appl., 2018, 277–311.

[23] Altıntaş, O., Srivastava, H. M. (2001). Some majorization problems associated with p-valently starlike and convex functions of complex order, East Asian Math. J., 17, 175–183.

[24] Altıntaş, O., Ozkan, O., Srivastava, H. M. (2001). Majorization by starlike functions of complex order, Complex Variables, Theory and Appl.: An Int. J., 46, 207–218.

[25] Dhuria, K., Mathur, R. (2013). Majorization for certain classes of meromorphic functions defined by integral operator, International Journal of Open Problems in Complex Analysis, 5, 50–56.

[26] Goyal, S. P., Goswami, P. (2012). Majorization for certain classes of meromorphic functions defined by integral operator, Annales UMCS, Math., 66, 57–62.

[27] Bulut, S., Adegani, E. A., Bulboaca, T. (2021). Majorization results for a general subclass of meromorphic multivalent functions, Oliteh. Univ. Buchar. Sci. Bull. Ser. A Appl. Math. Phys., 83, 121–128.

[28] Janani, T., Murugusundaramoorthy, G. (2014). Majorization problems for p-valently meromorphic functions of complex order involving certain integral operator, Glob. J. Math. Anal., 2, 146–151.

[29] Tang, H. Aouf, M. K., Deng, G. (2015). Majorization problems for certain subclasses of meromorphic multivalent functions associated with the Liu-Srivastava operator, Filomat, 29, 763–772.

[30] Rasheed, A., Hussain, S., Ghoos S., Shah, A., Darus, M., Lodhi, S. (2020). Majorization problem for two subclasses of meromorphic functions associated with a convolution operator, AIMS Mathematics, 5, 5157–5170.

[31] Panigrahi, T., El-Ashwah, R. (2017). Majorization for subclasses of multivalent meromorphic functions defined through iterations and combinations of the Liu-Srivastava operator and a meromorphic analogue of the Cho-Kwon-Srivastava operator, Filomat, 31, 6357–6365.

[32] Deniz, E., Kazımoglu, S., Kızıltepe, A. (2023). Majorization Properties for a Subclass of Analytic Function of Complex Order, Conference Proceeding Science and Technology, 1, 254–265.

[33] Janowski, W. (1973). Some extremal problems for certain families of analytic functions, Ann. Pol. Math., 28, 297–326.

[34] Nehari, Z. (1955). Conformal Mappings, MacGrow-Hill Book Company, New york, Torinto and London.

[35] Selvakumaran, K. A., Purohit, S. D., Secer, A. (2014). Majorization for a class of analytic functions defined by q-diferentiation, Math. Probl. Eng., Article ID 653917, 5 pages.

[36] Vijaya, K., Murugusundaramoorthy, G., Cho, N. E. (2021). Majorization Problems for Uniformly Starlike functions based on Ruscheweyh q-diferential operator defined with exponential function, Nonlinear Funct. Anal. Appl., 26, 71–81.

2025. Volume 9, Number 2 Pages 152-156

UDC:517.54
DOI: https://doi.org/10.30546/09090.2025.510.1029

SUFFICIENT CONDITIONS FOR A GENEREAL INTEGRAL OPERATOR RELATED WİTH BOUNDED BOUNDARY ROTATION

Erhan DENİZ*
Kafkas University,
ederiz36@gmail.com
Kars-Türkiye

ARTICLE INFO

Article history
Received:2026-01-09
Received in revised form:2026-01-12
Accepted:2026-01-16
Available online

Keywords:
Analytic function;
Univalent functions;
Starlike and convex;
Integral operator.
2010 Mathematics Subject Classifications: 30C45

ABSTRACT

In this paper, we define a new subclass $\mathrm{SP}_k^{\lambda}(\delta,\alpha,\omega,\beta,m)$ *of analytic functions by means of an appropriate differential operator. The introduced class is characterized through analytic conditions involving several real and complex parameters. For functions belonging to this subclass, we investigate and determine various structural and functional properties. In particular, attention is focused on the integral operator* $I_m(f_1,\ldots,f_r)$ *associated with this class of functions. Several results describing the behavior and preservation properties of this operator are established. The obtained findings contribute to the theory of analytic function classes defined via differential and integral operators and extend related results in the existing literature.*

1. Introduction

Let A denote the class of all analytic functions of the form

$$f(z)=z+\sum_{n=2}^{\infty}a_n z^n \qquad (33)$$

defined in the open unit disc $\mathrm{U}=\{z\in\mathbb{C}:|z|<1\}$. Let S be the subclass of A containing univalent functions defined in U. Let $\mathrm{P}_k^{\lambda}(\beta)$ denote the class of analytic functions $p(z)$ in U satisfying the following properties:

i. $p(0)=1$

ii. $\int_0^{2\pi}\left|\frac{\Re e^{i\lambda}p(z)-\beta\cos\lambda}{1-\beta}\right|d\theta\le k\pi\cos\lambda,$

where $k\ge 2$, λ is real, $|\lambda|<\frac{\pi}{2}$, $0\le\beta<1$, $z=re^{i\theta}$, $0\le r<1$.

Let $V_k^{\lambda}(\beta)$ (see [8]) denote the class of functions $f(z)$ analytic in U satisfies the normalization properties $f(0)=f'(0)-1=0$ and

$$1+\frac{z\,f''(z)}{f'(z)}\in P_k^{\lambda}(\beta),$$

where k, λ and β are as above.

For $\beta=0$ we get the class V_k^{λ} of functions with bounded boundary rotation studied by Moulis [7].

Any function $f(z)\in V_k^{\lambda}(\beta)$ if and only if

$$\Re\left\{e^{i\lambda}\left(1+\frac{z\,f''(z)}{f'(z)}\right)\right\}>\beta\cos\lambda,\qquad |z|<\frac{k-\sqrt{k^2-4}}{2}.$$

A function $f\in A$ with the normalization properties $f(0)=f'(0)-1=0$ is said to be in the class $U_k^{\lambda}(\beta)$ if $\frac{z\,f'(z)}{f(z)}\in P_k^{\lambda}(\beta)$.

In [6], Darus and Faisal defined the differential operator $D_\delta^m(\alpha,\omega)$ as follows:

$$\begin{aligned}
D_\delta^0(\alpha,\omega)f(z)&=f(z),\\
D_\delta^1(\alpha,\omega)f(z)&=Df(z)=\left(1-\delta\omega^{\alpha}\right)f(z)+\delta\omega^{\alpha}zf'(z), \qquad (2)\\
D_\delta^2(\alpha,\omega)f(z)&=D\left(D_\delta^1(\alpha,\omega)f(z)\right),\\
&\vdots\\
D_\delta^m(\alpha,\omega)f(z)&=D\left(D_\delta^{m-1}(\alpha,\omega)f(z)\right),
\end{aligned}$$

where $\delta,\alpha,\omega\ge 0$ and $m\in\mathbb{N}_0=\mathbb{N}\cup\{0\}$.

If f is given (1) then from the definition of the operator $D_\delta^m(\alpha,\omega)f(z)$ it is easy to see that

$$D_\delta^m(\alpha,\omega)f(z)=z+\sum_{n=2}^{\infty}\left[1+(n-1)\delta\omega^{\alpha}\right]^m a_n z^n. \qquad (3)$$

It should be remarked that the $D_\delta^m(\alpha,\omega)$ is a generalization of many other linear operators considered earlier. In particular, for $f\in A$ we have the following:

1. $D_1^m(1,1)f(z)=D^m f(z)$ the operator investigated by Salagean [9]

2. $D_{\delta}^{m}(1,1)f(z)=D_{\delta}^{m}f(z)$ the operator studied by Al-Oboudi [1]

3. $D_{1/2}^{m}(1,1)f(z)=I^{m}f(z)$ the operator studied by Uralegaddi and Somanatha [10].

Now, by making use of $D_{\delta}^{m}(\alpha,\omega)$, we define a new subclass of analytic functions.

Let $\mathrm{SP}_{k}^{\lambda}(\delta,\alpha,\omega,\beta,m)$ the class of functions $f\in\mathrm{A}$ and satisfying the condition

$$\frac{z\left(D_{\delta}^{m}(\alpha,\omega)f(z)\right)'}{D_{\delta}^{m}(\alpha,\omega)f(z)}\in \mathrm{P}_{k}^{\lambda}(\beta),$$

where β is real number with $0\le\beta<1$.

Let $r,m\in\mathbb{N}_{0}$ and $a_i>0$. We define the integral operator $I_m: A^r\to A$

$$I_m(f_1,\ldots,f_r)(z)=\int_0^z\left(\frac{D_{\delta}^{m}(\alpha,\omega)f_1(t)}{t}\right)^{a_1}\cdots\left(\frac{D_{\delta}^{m}(\alpha,\omega)f_r(t)}{t}\right)^{a_r}dt,\quad z\in\mathrm{U},\quad (4)$$

where $f_i\in\mathrm{A}$.

For special value of $m=0$ we have the integral operator

$$I_0(f_1,\ldots,f_r)(z)=\int_0^z\left(\frac{f_1(t)}{t}\right)^{a_1}\cdots\left(\frac{f_r(t)}{t}\right)^{a_r}dt \qquad (5)$$

introduced in [4].

2. Main Results

Our first result as follows.

Theorem 2.1. Let $a_i>0,$ and β_i be real numbers with the property $0\le\beta_i<1$ and $f_i\in\mathrm{SP}_{k}^{\lambda}(\delta,\alpha,\omega,\beta_i,m)$ for $i\in\mathbb{N}$. If $0<\sum_{i=1}^{r}a_i(1-\beta_i)\le 1,$ then $I_m(f_1,\cdots,f_r)\in\mathrm{V}_{k}^{\lambda}(\gamma)$, where $\gamma=1+\sum_{i=1}^{m}a_i(\beta_i-1).$

Proof. From (3), for $1\le i\le r,$ we have

$$\frac{D_{\delta}^{m}(\alpha,\omega)f_i(z)}{z}=1+\sum_{n=2}^{\infty}\left[1+(n-1)\delta\omega^{\alpha}\right]^{m}a_n z^{n-1},\quad m\in\mathbb{N}_{0}$$

and

$$\frac{D_{\delta}^{m}(\alpha,\omega)f_i(z)}{z}\neq 0,\quad \forall z\in\mathrm{U}.$$

We consider the operator

$$I_m(f_1,\ldots,f_r)(z)=\int_0^z\left(\frac{D_{\delta}^{m}(\alpha,\omega)f_1(t)}{t}\right)^{a_1}\cdots\left(\frac{D_{\delta}^{m}(\alpha,\omega)f_r(t)}{t}\right)^{a_r}dt.$$

On successive differentiation of $I_m(f_1,\cdots,f_r)$ we get

$$I_m(f_1,\cdots,f_r)'(z)=\left(\frac{D_\delta^m(\alpha,\omega)f_1(z)}{z}\right)^{a_1}\cdots\left(\frac{D_\delta^m(\alpha,\omega)f_r(z)}{z}\right)^{a_r},$$

$$I_m(f_1,\cdots,f_r)''(z)=\sum_{i=1}^{r}a_i\left(\frac{D_\delta^m(\alpha,\omega)f_i(z)}{z}\right)^{a_i-1}\frac{z\left(D_\delta^m(\alpha,\omega)f_i(z)\right)'-D_\delta^m(\alpha,\omega)f_i(z)}{z^2}$$
$$\times\prod_{j=1}^{r}\left(\frac{D_\delta^m(\alpha,\omega)f_j(z)}{z}\right)^{a_j}$$

and so

$$\frac{I_m(f_1,\cdots,f_r)''(z)}{I_m(f_1,\cdots f_r)'(z)}=\sum_{i=1}^{r}a_i\left[\frac{z\left(D_\delta^m(\alpha,\omega)f_i(z)\right)'}{D_\delta^m(\alpha,\omega)f_i(z)}-\frac{1}{z}\right].$$

Thus we obtain that

$$e^{i\lambda}\left(1+\frac{z\,I_m(f_1,\cdots,f_r)''(z)}{I_m(f_1,\cdots,f_r)'(z)}\right)=\sum_{i=1}^{n}a_ie^{i\lambda}\left[\frac{z\left(D_\delta^m(\alpha,\omega)f_i(z)\right)'}{D_\delta^m(\alpha,\omega)f_i(z)}\right]+e^{i\lambda}\left(1-\sum_{i=1}^{r}a_i\right).$$

From the last equality, we have

$$\Re\left\{e^{i\lambda}\left(1+\frac{z\,I_m(f_1,\cdots,f_r)''(z)}{I_m(f_1,\cdots,f_r)'(z)}\right)\right\}=\sum_{i=1}^{r}a_i\Re\left\{e^{i\lambda}\left[\frac{z\left(D_\delta^m(\alpha,\omega)f_i(z)\right)'}{D_\delta^m(\alpha,\omega)f_i(z)}\right]\right\}+\cos\lambda\left(1-\sum_{i=1}^{r}a_i\right).$$

Since $f_i\in \mathrm{SP}_k^{\lambda}(\delta,\alpha,\omega,\beta_i,m)$ from the hypothesis of theorem we get

$$\begin{aligned}\Re\left\{e^{i\lambda}\left(1+\frac{z\,I_m(f_1,\cdots,f_r)''(z)}{I_m(f_1,\cdots,f_r)'(z)}\right)\right\}&=\sum_{i=1}^{r}a_i\Re\left\{e^{i\lambda}\left[\frac{z\left(D_\delta^m(\alpha,\omega)f_i(z)\right)'}{D_\delta^m(\alpha,\omega)f_i(z)}\right]\right\}+\cos\lambda\left(1-\sum_{i=1}^{r}a_i\right)\\&>\sum_{i=1}^{r}\beta_ia_i\cos\lambda+\cos\lambda\left(1-\sum_{i=1}^{r}a_i\right)\\&=\cos\lambda\left(1+\sum_{i=1}^{r}a_i(\beta_i-1)\right).\end{aligned}$$

Hence $I_m(f_1,\cdots,f_r)(z)\in V_k^{\lambda}(\gamma)$, where $\gamma=1+\sum_{i=1}^{r}a_i(\beta_i-1)$.

For special values $m=0$, $k=2$ and $\lambda=0$ we get the following results.

Corollary 2.2. Let $a_i, i\in\{1,2,\cdots,r\}$ be real numbers with the properties $a_i>0$ for $i\in\{1,2,\cdots,r\}$ and $r<\sum_{i=1}^{r}a_i\le r+1$. We suppose that the functions f_i are the starlike functions of order $\frac{1}{a_i}, i\in\{1,2,\cdots,r\}$, that is, $f_i\in S^*\left(\frac{1}{a_i}\right)$. Then the integral operator defined in (4) is convex of order $\tilde{\gamma}=r+1-\sum_{i=1}^{r}a_i$ (see [2]).

Corollary 2.3. Let $a_i, i \in \{1,2,\cdots,r\}$ be real numbers with the properties $a_i > 0$ for $i \in \{1,2,\cdots,r\}$ and $0 < \sum_{i=1}^{r} a_i \leq 1$. We consider that the functions $f_i \in S^*$ for $i \in \{1,2,\cdots,r\}$. In these conditions, the integral operator defined in (4) is convex of order $1 - \sum_{i=1}^{r} a_i$ (see [5]).

For $\beta_1 = \beta_2 = \cdots = \beta_r = \beta$ and $m = 0,$ similarly we prove the following theorem.

Theorem 2.4. Let a_i be real numbers with the properties $a_i > 0$ and $f_i \in SP_k^{\lambda}(\delta, \alpha, \omega, \beta, 0)$ for $i \in$. If $0 < \sum_{i=1}^{r} a_i \leq \frac{1}{1-\beta},$ then the integral operator (4) is in the class $V_k^{\lambda}(\gamma),$ where $\gamma = (\beta - 1)\sum_{i=1}^{r} a_i + 1.$

For special values $m = 0,$ $k = 2$ and $\lambda = 0$ in Theorem 2.4, we get the following result [3].

Corollary 2.5. Let $a_i > 0$ for $i \in \{1,2,\cdots,r\}$ and $f_i \in SP_2^0(\delta, \alpha, \omega, \beta, 0)(0 \leq \beta < 1)$. If $0 < \sum_{i=1}^{r} a_i \leq \frac{1}{1-\beta},$ then the integral operator $I_0(f_1, \cdots, f_r)$ is convex of order $(\beta - 1)\sum_{i=1}^{r} a_i + 1.$

3. Conclusion

In the current study, a new subclass of analytic functions associated with a differential operator has been introduced and investigated. Also we introduced a new general integral operator $I_m(f_1, \cdots, f_r)$. Main results related with sufficient conditons of this integral operator to belonging bounded boundary rotation. For special cases of parameters we obtained the earlier results.

REFERENCE LIST

[1] Al-Oboudi, F. M. (2004). On univalent functions defined by a generalized Salagean operator, *Int. J. Math. Math. Sci.*, 27, 1429-1436.

[2] Breaz, D. (2008). A convexity property for an integral operator on the class $S_p(\beta)$, *J. Inequal. Appl.*, 2008, Article ID 143869, 4 pages.

[3] Breaz, D., Breaz., N. (2006). Some convexity properties for a general integral operator, *J. Inequal. Pure Appl. Math.*, 7(5).

[4] Breaz, D., Breaz., N. (2002). Two integral operators, *Studia Universitatis Babeş -Bolyai, Mathematica*, Cluj Napoca, 47(3), 13-21.

[5] Breaz, D., Owa, S., Breaz, N. (2008). A new integral univalent operator, *Acta Univ. Apulensis Math. Inform.*, 16, 11-16.

[6] Darus, M., Faisal, I. (2012). Some subclasses of analytic functions of complex order defined by new differential operator. *Tamkang J. Math.*, 43(2), 223-242.

[7] Moulis, E. J. (1972). A generalization of univalent functions with bounded boundary rotation, *Trans. Am. Math. Soc.*, 174, 369 -381.

[8] Moulis, E. J. (1979). Generalization of the Robertson functions, *Pacific J. Math.*, 81, 169-174.

[9] Salagean, G.S. (1983). Subclasses of univalent functions, *Lect. Notes Math.*, Springer-Verlag, 1013, 362-372.

[10] Uralegaddi, B.A., Somanatha, C. (1992). Certain classes of univalent functions, *In: Current Topics in AnalyticFunction Theory. Eds. H.M. Srivastava and S. Owa. World Scientific Publishing Company, Singapore*, 3

JOURNAL OF BAKU ENGINEERING UNIVERSITY- MATHEMATICS AND COMPUTER SCIENCE
2025. Volume 9, Number 2 *Pages 157-167*

UDC: 517.54
DOI:

GEOMETRIC PROPERTIES OF MULTIVALENT FUNCTIONS ASSOCIATED WITH BOREL DISTRIBUTION FUNCTIONS

Sercan KAZIMOĞLU[1*]
Kafkas University
sercan.kazimoglu@kafkas.edu.tr
Kars-Türkiye

ARTICLE INFO

Article history:
Received:2025-12-18
Received in revised form:2025-12-18
Accepted:2025-12-24
Available online

Keywords:
Analytic function;
Borel distribution;
Strongly starlike and convex;
Coefficient estimates;
Integral representation.
2010 Mathematics Subject Classifications: 30C45

ABSTRACT

In this paper, we determine the necessary and sufficient conditions for the power series $f(z)$ whose coefficients are probabilities of the Borel distribution and the differential operator to be in the family $J_{\delta}(p,\lambda,\alpha,\beta,\gamma)$ of analytic functions which defined in the open unit disk. We derive a number of important geometric properties, such as, coefficient estimates, integral representation, radii of starlikeness and convexity. Also we discuss the extreme points and neighborhood property for functions belongs to this family.

1. Introduction

Indicate by A_p the family of all functions f of the form

$$f(z)=z^p+\sum_{n=p+1}^{\infty}a_n z^n \quad (34)$$

which are analytic and multivalent in the open unit disk $U=\{z\in\mathbb{C}:|z|<1\}$.

Also, let W_p denote the subfamily of A_p consisting of functions of the form:

$$f(z)=z^p-\sum_{n=p+1}^{\infty}a_n z^n \ (a_n\geq 0, p\in\mathbb{N}). \quad (35)$$

The function $f\in W_p$ is said to be starlike of order $\rho(0\leq\rho<p)$ if it satisfies the condition:

$$\operatorname{Re}\left\{\frac{zf'(z)}{f(z)}\right\}>\rho \quad (z\in U).$$

A function $f \in \mathrm{W}_p$ is said to be convex of order $\rho\left(0 \le \rho < p\right)$ if it satisfies the condition

$$\mathrm{Re}\left\{1+\frac{zf''(z)}{f'(z)}\right\} > \rho \quad (z \in \mathrm{U}).$$

A function $f \in \mathrm{W}_p$ is said to be close-to-convex of order $\rho\left(0 \le \rho < p\right)$ if it satisfies the condition

$$\mathrm{Re}\left\{\frac{zf'(z)}{z^{p-1}}\right\} > \rho \ \ (z \in \mathrm{U}).$$

Let the function f and g be analytic in U. We say that the function f is subordinate to g, if there exists a Schwarz function w analytic in U with $w(0)=0$ and $|w(z)|<1$ $(z \in \mathrm{U})$ such that $f(z)=g\left(w(z)\right)$. This subordinate is denoted by $f \prec g$ or $f(z) \prec g(z)$ $(z \in \mathrm{U})$. It is well known that (see [7]), if the function g is univalent in U then $f \prec g$ if and only if $f(0)=g(0)$ and $f(\mathrm{U}) \subset g(\mathrm{U})$.

Denote by $\mathrm{S}^*(\alpha)$ and $\mathrm{C}(\alpha)$ the families of starlike and convex functions of order ρ, respectively. These families were introduced and studied by Silverman [10].

The elementary distributions such as the Poisson, the Pascal, the Logarithmic, the Binomial have been partially studied in the Geometric Function Theory from a theoretical point of view (see [1,4,5,8,9,11,12]).

A discrete random variable x is said to have a Borel distribution if it takes the values $1,2,3,\ldots$ with the probabilities $\frac{e^{-\lambda}}{1!}, \frac{2\lambda e^{-2\lambda}}{2!}, \frac{9\lambda^2 e^{-3\lambda}}{3!}, \cdots$ respectively, where λ is called the parameter.

Very recently, Wanas and Khuttar [13] introduced the Borel distribution (BD) whose probability mass function is

$$P(x=r)=\frac{(\lambda r)^{r-1}e^{-\lambda r}}{r!}, \quad r=1,2,3,\ldots.$$

Wanas and Khuttar [13] introduced a series whose coefficients are probabilities of the Borel distribution (BD)

$$M_p(\lambda;z)=z^p-\sum_{n=p+1}^{\infty}\frac{\left[\lambda(n-p)^{n-p-1}e^{-\lambda(n-p)}\right]}{(n-p)!}z^n, \quad 0<\lambda\le 1, \ p\in\square.$$

For $\delta \ge 0$, Amini et. al. [2] defined the linear operator $\mathrm{B}^m_{\lambda,\delta}f(z):\mathrm{W}_p \to \mathrm{W}_p$ as

$$M_p(\lambda;z)*f(z) := \mathrm{B}^0_{\lambda,\delta} f(z) = z^p - \sum_{n=p+1}^{\infty} \frac{\left[\lambda(n-p)^{n-p-1} e^{-\lambda(n-p)}\right]}{(n-p)!} a_n z^n$$

$$\mathrm{B}^1_{\lambda,\delta} f(z) = (1-\delta)\mathrm{B}^0_{\lambda,\delta} f(z) + \frac{\delta}{p} z\left(\mathrm{B}^0_{\lambda,\delta} f(z)\right)'$$

$$= z^p - \sum_{n=p+1}^{\infty} \frac{\left[\lambda(n-p)^{n-p-1} e^{-\lambda(n-p)}\right]}{(n-p)!}\left[1+\delta\left(\frac{n}{p}-1\right)\right] a_n z^n$$

$$\mathrm{B}^2_{\lambda,\delta} f(z) = (1-\delta)\mathrm{B}^1_{\lambda,\delta} f(z) + \frac{\delta}{p} z\left(\mathrm{B}^1_{\lambda,\delta} f(z)\right)'$$

$$= z^p - \sum_{n=p+1}^{\infty} \frac{\left[\lambda(n-p)^{n-p-1} e^{-\lambda(n-p)}\right]}{(n-p)!}\left[1+\delta\left(\frac{n}{p}-1\right)\right]^2 a_n z^n$$

$$\vdots$$

$$\mathrm{B}^m_{\lambda,\delta} f(z) = (1-\delta)\mathrm{B}^{m-1}_{\lambda,\delta} f(z) + \frac{\delta}{p} z\left(\mathrm{B}^{m-1}_{\lambda,\delta} f(z)\right)'$$

$$= z^p - \sum_{n=p+1}^{\infty} \frac{\left[\lambda(n-p)^{n-p-1} e^{-\lambda(n-p)}\right]}{(n-p)!}\left[1+\delta\left(\frac{n}{p}-1\right)\right]^m a_n z^n$$

$$= z^p - \sum_{n=p+1}^{\infty} \Phi^m_{n,p}(\lambda,\delta) a_n z^n, \quad m \in \mathbb{N} \cup \{0\},$$

where $a_n \geq 0,\ 0 < \lambda \leq 1,\ z \in \mathrm{U}$ and

$$\Phi^m_{n,p}(\lambda,\delta) := \frac{\left[\lambda(n-p)^{n-p-1} e^{-\lambda(n-p)}\right]}{(n-p)!}\left[1+\delta\left(\frac{n}{p}-1\right)\right]^m.$$

We now recall the following Lemmas that will be used to prove our main results.

Lemma 1: [6] If f and g are analytic in U with $f \prec g$, then

$$\int_0^{2\pi} \left| f\left(re^{i\theta}\right)\right|^{\mu} d\theta \leq \int_0^{2\pi} \left| g\left(re^{i\theta}\right)\right|^{\mu} d\theta$$

where $\mu > 0,\ z = re^{i\theta}$ and $0 < r < 1$.

Lemma 2: [3] Let $\alpha \geq 0$. Then $\mathrm{Re}(w) > \alpha$ if and only if $|w-(p+\alpha)| < |w+(p-\alpha)|$, where w be any complex number.

2. Main Results

We begin this section by defining the family $\mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$ as follows:

Definition 1: A function f of the form (35) is said to be in the class $\mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$ if satisfies the following condition:

$$\operatorname{Re}\left\{\frac{z\left(\mathrm{B}_{\lambda,\delta}^{m} f(z)\right)' + \gamma z^{2}\left(\mathrm{B}_{\lambda,\delta}^{m} f(z)\right)''}{\left[\begin{array}{l}\gamma z\left[\left(\mathrm{B}_{\lambda,\delta}^{m} f(z)\right)' + \beta z\left(\mathrm{B}_{\lambda,\delta}^{m} f(z)\right)''\right] \\ +(1-\gamma)\left[\beta z\left(\mathrm{B}_{\lambda,\delta}^{m} f(z)\right)' + (1-\beta)\mathrm{B}_{\lambda,\delta}^{m} f(z)\right]\end{array}\right]}\right\} > \alpha, \quad (36)$$

where $0 \le \alpha < 1,\ 0 \le \beta < 1,\ 0 \le \gamma < 1,\ \delta \ge 0$ and $0 < \lambda \le 1.$

Theorem 1: Let $f \in \mathrm{W}_{p}$ then $f \in \mathrm{J}_{\delta}(p, \lambda, \alpha, \beta, \gamma)$ if and only if

$$\sum_{n=\rho+1}^{\infty} \left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\right)\Phi_{n,p}^{m}(\lambda,\delta)\, a_{n} \le \left(\gamma(p-1)+1\right)\left(\rho-\alpha-(p-1)\alpha\beta\right) \quad (37)$$

where $0 \le \alpha < 1,\ 0 \le \beta < 1,\ 0 \le \gamma < 1,\ \delta \ge 0,\ 0 < \lambda \le 1$ and $p \in \mathbb{N}$.

The result is sharp for the function

$$f(z) = z^{p} - \frac{\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right)}{\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\right)\Phi_{n,p}^{m}(\lambda,\delta)} z^{n}, \quad n \ge p+1,\ p \in \mathbb{N}. \quad (38)$$

Proof. Assume that $f \in \mathrm{J}_{\delta}(p, \lambda, \alpha, \beta, \gamma)$, so we have

$$\operatorname{Re}\left\{\frac{z\left(\mathrm{B}_{\lambda,\delta}^{m} f(z)\right)' + \gamma z^{2}\left(\mathrm{B}_{\lambda,\delta}^{m} f(z)\right)''}{\left[\begin{array}{l}\gamma z\left[\left(\mathrm{B}_{\lambda,\delta}^{m} f(z)\right)' + \beta z\left(\mathrm{B}_{\lambda,\delta}^{m} f(z)\right)''\right] \\ +(1-\gamma)\left[\beta z\left(\mathrm{B}_{\lambda,\delta}^{m} f(z)\right)' + (1-\beta)\mathrm{B}_{\lambda,\delta}^{m} f(z)\right]\end{array}\right]}\right\} > \alpha.$$

Then

$$\operatorname{Re}\left\{\frac{\begin{array}{l} pz^{p} - \sum_{n=p+1}^{\infty} \Phi_{n,p}^{m}(\lambda,\delta)\, n a_{n} z^{n} + \gamma p(p-1) z^{p} - \sum_{n=p+1}^{\infty} \Phi_{n,p}^{m}(\lambda,\delta)\gamma n(n-1) a_{n} z^{n} \end{array}}{\left[\begin{array}{l}\gamma p z^{p} - \sum_{n=p+1}^{\infty} \Phi_{n,p}^{m}(\lambda,\delta)\gamma n a_{n} z^{n} + \gamma\beta p(p-1) z^{p} - \sum_{n=p+1}^{\infty} \Phi_{n,p}^{m}(\lambda,\delta)\gamma\beta n(n-1) a_{n} z^{n} \\ +(1-\gamma)\beta p z^{p} - \sum_{n=p+1}^{\infty} \Phi_{n,p}^{m}(\lambda,\delta)(1-\gamma)\beta n a_{n} z^{n} + (1-\gamma)(1-\beta)\left(z^{p} - \sum_{n=p+1}^{\infty} \Phi_{n,p}^{m}(\lambda,\delta) a_{n} z^{n}\right)\end{array}\right]}\right\} > \alpha.$$

Or equivalently

$$\operatorname{Re}\left\{\frac{\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right)z^p-\sum_{n=p+1}^{\infty}\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\right)\Phi_{n,p}^m(\lambda,\delta)a_n z^n}{\alpha\left(1+\gamma(p-1)\right)\left(1+\beta(p-1)\right)z^p-\sum_{n=p+1}^{\infty}\Phi_{n,p}^m(\lambda,\delta)\alpha\left(1+\gamma(n-1)\right)\left(1+\beta(n-1)\right)a_n z^n}\right\}>0.$$

This inequality is correct for a $z\in \mathrm{U}$ Letting $z\to 1^-$ yields

$$\operatorname{Re}\left\{\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right)z^p-\sum_{n=p+1}^{\infty}\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\Phi_{n,p}^m(\lambda,\delta)a_n\right)\right\}>0.$$

Therefore

$$\sum_{n=p+1}^{\infty}\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\right)\Phi_{n,p}^m(\lambda,\delta)a_n\le\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right).$$

Conversely, let (37) hold. We will prove that (36) is correct and then $f\in \mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$. By Lemma 2, we put

$$w=\frac{z\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)'+\gamma z^2\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)''}{\left[\begin{array}{l}\gamma z\left[\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)'+\beta z\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)''\right]\\+(1-\gamma)\left[\beta z\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)'+(1-\beta)\mathrm{B}_{\lambda,\delta}^m f(z)\right]\end{array}\right]}.$$

Or show that

$$T=\frac{1}{\left|\mathrm{N}(z)\right|}\left|z\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)'+\gamma z^2\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)''-(p+\alpha)z\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)'-(p+\alpha)\gamma\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)''\right|<Q,$$

where

$$\mathrm{N}(z)=\gamma z\left[\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)'+\beta z\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)''\right]+(1-\gamma)\left[\beta z\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)'+(1-\beta)\left(\mathrm{B}_{\lambda,\delta}^m f(z)\right)\right]$$

and it is easy to verify that $Q-T>0$.

And so the proof is complete.

Corollary 1: If $f\in \mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$ then

$$a_n\le\frac{\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right)}{\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\Phi_{n,p}^m(\lambda,\delta)\right)},\quad n\ge p+1,\ p\in\square.$$

Theorem 2: Let a function $f\in \mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$. Then f is p-valently close-to-convex of order $\rho(0\le\rho<p)$ in the disk $|z|<R_1$, where

$$R_1=\inf_n\left\{\frac{(p-\rho)\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\right)\Phi_{n,p}^m(\lambda,\delta)}{\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right)}\right\}^{\frac{1}{n-p}},\quad n\ge p+1;\ p\in\square.$$

The result is sharp, with the extermal function f given by (38).

Proof. It is sufficient to show that

$$\left|\frac{f'(z)}{z^{p-1}}-p\right|\le p-\rho,\quad 0\le\rho<p,$$

for $|z|<R_1$, we have that

$$\left|\frac{f'(z)}{z^{p-1}}-p\right|\le\sum_{n=p+1}^{\infty}\Phi_{n,p}^{m}(\lambda,\delta)na_n|z|^{n-p}.$$

Thus

$$\left|\frac{f'(z)}{z^{p-1}}-p\right|\le p-\rho,$$

if

$$\sum_{n=p+1}^{\infty}\frac{\Phi_{n,p}^{m}(\lambda,\delta)a_n|z|^{n-p}}{p-\rho}\le 1. \qquad (39)$$

Hence, by Theorem 1, (38) will be true if

$$\frac{1}{p-\rho}|z|^{n-p}\le\frac{(\gamma(n-1)+1)(n-\alpha-(n-1)\alpha\beta)\Phi_{n,p}^{m}(\lambda,\delta)}{(\gamma(p-1)+1)(p-\alpha-(p-1)\alpha\beta)},$$

and hence

$$|z|\le\left\{\frac{(p-\rho)(\gamma(n-1)+1)(n-\alpha-(n-1)\alpha\beta)\Phi_{n,p}^{m}(\lambda,\delta)}{(\gamma(p-1)+1)(p-\alpha-(p-1)\alpha\beta)}\right\}^{\frac{1}{n-p}},\quad n\ge p+1;\ p\in\mathbb{N}.$$

The result is sharp for the function f given by (38).

Theorem 3: Let $f\in J_{\delta}(p,\lambda,\alpha,\beta,\gamma)$. Then f is p-valently starlike of order $\rho(0\le\rho<p)$ in the disk $|z|<R_2$ where

$$R_2=\inf_n\left\{\frac{(p-\rho)(\gamma(n-1)+1)(n-\alpha-(n-1)\alpha\beta)\Phi_{n,p}^{m}(\lambda,\delta)}{(n-p)(\gamma(p-1)+1)(p-\alpha-(p-1)\alpha\beta)}\right\}^{\frac{1}{n-p}},\quad n\ge p+1;\ p\in\mathbb{N}.$$

The result is sharp for the function f given by (38).

Proof. It is sufficient to show that

$$\left|\frac{zf'(z)}{f(z)}-p\right|\le p-\rho,\quad 0\le\rho<p,$$

for $|z|<R_2$, we have

$$\left|\frac{zf'(z)}{f(z)}-p\right|\le\frac{\sum_{n=p+1}^{\infty}\Phi_{n,p}^{m}(\lambda,\delta)(n-p)a_n|z|^{n-p}}{1-\sum_{n=p+1}^{\infty}a_k|z|^{n-p}}.$$

Thus

$$\left|\frac{zf'(z)}{f(z)}-p\right|\le p-\rho,$$

if

$$\sum_{n=p+1}^{\infty}\frac{n-\rho}{p-\rho}\Phi_{n,p}^{m}(\lambda,\delta)a_n|z|^{n-p}\le 1. \qquad (40)$$

Hence, by Theorem 1, (40) will be true if

$$\frac{(n-p)}{(p-\rho)}|z|^{n-p}\le\frac{\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\right)}{\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right)},$$

and hence

$$|z|\le\left\{\frac{(p-\rho)\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\right)}{(n-p)\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right)}\right\}^{\frac{1}{n-p}},\quad n\ge p+1;\ p\in\mathbb{N}.$$

Setting $|z|=R_2$, we get the desired result.

Theorem 4: Let $f\in \mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$. Then f is p-valently convex of order $\left(\rho(0)\le\rho<p\right)$ in the disk $|z|<R_3$, where

$$R_3=\inf_n\left\{\frac{(p-\rho)\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\right)}{(n-p)\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right)}\right\}^{\frac{1}{n-p}},\quad n\ge p+1;\ p\in\mathbb{N}.$$

The result is sharp with the extermal function f given by (38).

Proof. It is sufficient to show that

$$\left|1+\frac{zf''(z)}{f'(z)}-p\right|\le p-\rho,\ 0\le\rho<p,$$

for $|z|<R_3$, we have

$$\left|1+\frac{zf''(z)}{f'(z)}-p\right|\le\frac{\sum_{n=p+1}^{\infty}\Phi_{n,p}^{m}(\lambda,\delta)n(n-p)a_n|z|^{n-p}}{p-\sum_{n=p+1}^{\infty}\Phi_{n,p}^{m}(\lambda,\delta)a_n|z|^{n-p}}.$$

Thus

$$\left|1+\frac{zf''(z)}{f'(z)}-p\right|\le p-\rho,$$

if

$$\sum_{n=p+1}^{\infty} \frac{n(n-\rho)}{p(p-\rho)} \Phi_{n,p}^{m}(\lambda,\delta) a_n |z|^{n-p} \leq 1. \qquad (41)$$

Hence, by Theorem 1, (41) will be true if

$$\frac{n(n-p)}{p(p-\rho)}|z|^{n-p} \leq \frac{(\gamma(n-1)+1)(n-\alpha-(p-1)\alpha\beta)}{(\gamma(p-1)+1)(p-\alpha-(p-1)\alpha\beta)'}$$

and hence

$$|z| \leq \left\{ \frac{p(p-\rho)(\gamma(n-1)+1)(n-\alpha-(n-1)\alpha\beta)}{n(n-p)(\gamma(p-1)+1)(p-\alpha-(p-1)\alpha\beta)} \right\}^{\frac{1}{n-p}}, \quad n \geq p+1;\ p \in \mathbb{N}.$$

Setting $|z| = R_3$, we get the desired result.

Theorem 5: Let the functions f_r defined by

$$f_r(z) = z^p - \sum_{n=p+1}^{\infty} a_{n,r} z^n, \quad a_{n,r} \geq 0,\ n \geq p+1,\ p \in \mathbb{N},\ r = 1,2,3,4,..., \qquad (42)$$

be in the class $\mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$ for every $r = 1,2,...,l.$

Then the function h_1 defined by

$$h_1(z) = z^p - \sum_{n=p+1}^{\infty} e_n z^n, \quad e_n \geq 0,\ n \geq p+1,\ p \in \mathbb{N},$$

also belongs to the class $\mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$ where

$$e_n = \frac{1}{l}\sum_{r=1}^{l} a_{n,r}, \quad n \geq p+1,\ p \in \mathbb{N}.$$

Proof. Since $f_r \in \mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$, it follows from Theorem 1, that

$$\sum_{n=p+1}^{\infty} (\gamma(n-1)+1)(n-\alpha-(n-1)\alpha\beta)\Phi_{n,p}^{m}(\lambda,\delta) a_{n,r} \leq (\gamma(p-1)+1)(p-\alpha-(p-1)\alpha\beta),$$

for every $r = 1,2,...,l.$ Hence

$$\begin{aligned}
&\sum_{n=p+1}^{\infty} (\gamma(n-1)+1)(n-\alpha-(n-1)\alpha\beta)\Phi_{n,p}^{m}(\lambda,\delta) e_n \\
&= \sum_{n=p+1}^{\infty} (\gamma(n-1)+1)(n-\alpha-(n-1)\alpha\beta)\Phi_{n,p}^{m}(\lambda,\delta)\left(\frac{1}{l}\sum_{r=1}^{l} a_{n,r}\right) \\
&= \frac{1}{l}\sum_{r=1}^{l}\left(\sum_{n=p+1}^{\infty} (\gamma(n-1)+1)(n-\alpha-(n-1)\alpha\beta)\Phi_{n,p}^{m}(\lambda,\delta) a_{n,r}\right) \\
&\leq (\gamma(p-1)+1)(p-\alpha-(p-1)\alpha\beta).
\end{aligned}$$

By Theorem 1, it follows that $h_1 \in \mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$.

Theorem 6: Let the functions f_r defined by (42) be in the class $\mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$ for every $r=1,2,...,l.$ Then the function h_2 defined by

$$h_2(z)=\sum_{r=1}^{l}c_r f_r(z)$$

is also in the class $\mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma),$ where

$$\sum_{r=1}^{l}c_r=1,\quad c_r\geq 0.$$

Proof. By Theorem 1, for every $r=1,2,...,l,$ we have

$$\sum_{n=p+1}^{\infty}\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\right)\Phi_{n,p}^{m}(\lambda,\delta)a_{n,r}\leq\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right).$$

But

$$h_2(z)=\sum_{r=1}^{l}c_r f_r(z)=\sum_{r=1}^{l}c_r\left(z^p-\sum_{n=p+1}^{\infty}a_{n,r}z^n\right)=z^p-\sum_{n=p+1}^{\infty}\left(\sum_{r=1}^{l}c_r a_{n,r}\right)z^n.$$

Therefore

$$\begin{aligned}
&\sum_{n=p+1}^{\infty}\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\right)\Phi_{n,p}^{m}(\lambda,\delta)\left(\sum_{r=1}^{l}c_r a_{n,r}\right)\\
&=\sum_{r=1}^{l}c_r\left(\sum_{n=p+1}^{\infty}\left(\gamma(n-1)+1\right)\left(n-\alpha-(n-1)\alpha\beta\right)\Phi_{n,p}^{m}(\lambda,\delta)a_{n,r}\right)\\
&\leq\sum_{r=1}^{l}c_r\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right)\\
&=\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right)
\end{aligned}$$

and the proof is complete.

Theorem 7: Let $\tau>0.$ If $f\in\mathrm{J}_{\delta}(p,\lambda,\alpha,\beta,\gamma)$ and suppose that f_s is defined by

$$f_s(z)=z^p-\frac{\left(\gamma(p-1)+1\right)\left(\rho-\alpha-(p-1)\alpha\beta\right)}{\left(\gamma(s-1)+1\right)\left(s-\alpha-(s-1)\alpha\beta\right)\Phi_{s,p}^{m}(\lambda,\delta)}z^s,\quad s\geq p+1,\ p\in\mathbb{N}.$$

If there exists an analytic function w defined by

$$\left(w(z)\right)^{s-p}=\frac{\left(\gamma(s-1)+1\right)\left(s-\alpha-(s-1)\alpha\beta\right)\Phi_{s,p}^{m}(\lambda,\delta)}{\left(\gamma(p-1)+1\right)\left(p-\alpha-(p-1)\alpha\beta\right)}\sum_{n=p+1}^{\infty}a_n z^{n-p}.$$

Then for $z=re^{i\theta}$ and $(0<r<1),$

$$\int_0^{2\pi}\left|f(z)\right|^{\tau}d\theta\leq\int_0^{2\pi}\left|f_s(z)\right|^{\tau}d\theta,\quad\tau>0.\quad(43)$$

Proof. We show that

$$\int_0^{2\pi}\left|1-\sum_{n=p+1}^{\infty}a_n z^{n-p}\right|^{\tau}d\theta\le\int_0^{2\pi}1-\left|\frac{\left(\gamma\left(p-1\right)+1\right)\left(p-\alpha-\left(p-1\right)\alpha\beta\right)}{\left(\gamma\left(s-1\right)+1\right)\left(s-\alpha-\left(s-1\right)\alpha\beta\right)\Phi_{s,p}^{m}\left(\lambda,\delta\right)}z^{s-p}\right|^{\tau}d\theta.$$

By applying Lemma 1, it suffices to show that

$$1-\sum_{n=p+1}^{\infty}a_n z^{n-p}\prec 1-\frac{\left(\gamma\left(p-1\right)+1\right)\left(p-\alpha-\left(p-1\right)\alpha\beta\right)}{\left(\gamma\left(s-1\right)+1\right)\left(s-\alpha-\left(s-1\right)\alpha\beta\right)\Phi_{s,p}^{m}\left(\lambda,\delta\right)}z^{s-p}.$$

Set

$$1-\sum_{n=p+1}^{\infty}a_n z^{n-p}=1-\frac{\left(\gamma\left(p-1\right)+1\right)\left(p-\alpha-\left(p-1\right)\alpha\beta\right)}{\left(\gamma\left(s-1\right)+1\right)\left(s-\alpha-\left(s-1\right)\alpha\beta\right)\Phi_{s,p}^{m}\left(\lambda,\delta\right)}\left(w\left(z\right)\right)^{s-p}.$$

We find that

$$\left(w\left(z\right)\right)^{s-p}=\frac{\left(\gamma\left(s-1\right)+1\right)\left(s-\alpha-\left(s-1\right)\alpha\beta\right)\Phi_{s,p}^{m}\left(\lambda,\delta\right)}{\left(\gamma\left(p-1\right)+1\right)\left(p-\alpha-\left(p-1\right)\alpha\beta\right)}\sum_{n=p+1}^{\infty}a_n z^{n-p},$$

which readily yield $w\left(0\right)=0.$

Furthermore using (37), we obtain

$$\begin{aligned}\left|w\left(z\right)\right|^{s-p}&=\left|\frac{\left(\gamma\left(s-1\right)+1\right)\left(s-\alpha-\left(s-1\right)\alpha\beta\right)\Phi_{s,p}^{m}\left(\lambda,\delta\right)}{\left(\gamma\left(p-1\right)+1\right)\left(p-\alpha-\left(p-1\right)\alpha\beta\right)}\sum_{n=p+1}^{\infty}a_n z^{n-p}\right|\\&\le\left|z\right|\left|\sum_{n=p+1}^{\infty}\frac{\left(\gamma\left(n-1\right)+1\right)\left(n-\alpha-\left(n-1\right)\alpha\beta\right)\Phi_{n,p}^{m}\left(\lambda,\delta\right)}{\left(\gamma\left(p-1\right)+1\right)\left(p-\alpha-\left(p-1\right)\alpha\beta\right)}a_n\right|\le\left|z\right|<1.\end{aligned}$$

Next, the proof for the first derivative.

Theorem 8: Let $\tau>0.$ If $f\in \mathrm{J}_{\delta}\left(p,\lambda,\alpha,\beta,\gamma\right)$ and

$$f_s\left(z\right)=z^p-\frac{\left(\gamma\left(p-1\right)+1\right)\left(p-\alpha-\left(p-1\right)\alpha\beta\right)}{\left(\gamma\left(s-1\right)+1\right)\left(s-\alpha-\left(s-1\right)\alpha\beta\right)\Phi_{s,p}^{m}\left(\lambda,\delta\right)}z^s,\ \ s\ge p+1,\ p\in\square.$$

Then for $z=re^{i\theta}$ and $\left(0<r<1\right),$

$$\int_0^{2\pi}\left|f'\left(z\right)\right|^{\tau}d\theta\le\int_0^{2\pi}\left|f_s'\left(z\right)\right|^{\tau}d\theta,\quad\tau>0.\ (44)$$

Proof. It is sufficient to show that

$$1-\sum_{n=p+1}^{\infty}\frac{n}{p}a_n z^{n-p}\prec 1-\frac{s\left(\gamma\left(p-1\right)+1\right)\left(p-\alpha-\left(p-1\right)\alpha\beta\right)}{p\left(\gamma\left(s-1\right)+1\right)\left(s-\alpha-\left(s-1\right)\alpha\beta\right)\Phi_{s,p}^{m}\left(\lambda,\delta\right)}z^{s-p}.$$

This follows because

$$\begin{aligned}\left|w\left(z\right)\right|^{s-p}&=\left|\frac{p\left(\gamma\left(s-1\right)+1\right)\left(s-\alpha-\left(s-1\right)\alpha\beta\right)\Phi_{s,p}^{m}\left(\lambda,\delta\right)}{s\left(\gamma\left(p-1\right)+1\right)\left(p-\alpha-\left(p-1\right)\alpha\beta\right)}\sum_{n=p+1}^{\infty}\frac{n}{p}a_n z^{n-p}\right|\\&\le\left|z\right|\left|\sum_{n=p+1}^{\infty}\frac{\left(\gamma\left(n-1\right)+1\right)\left(n-\alpha-\left(n-1\right)\alpha\beta\right)\Phi_{s,p}^{m}\left(\lambda,\delta\right)}{\left(\gamma\left(p-1\right)+1\right)\left(p-\alpha-\left(p-1\right)\alpha\beta\right)}a_n\right|\le\left|z\right|<1.\end{aligned}$$

3. Conclusion

In the current study, a new subclass of multivalent analytic functions associated with the Borel distribution series and the differential operator has been introduced and investigated. By employing a linear operator defined via the Hadamard product, necessary and sufficient coefficient conditions for functions belonging to the proposed class have been established. Several significant geometric properties of this class have been derived, including sharp coefficient estimates, radii of close-to-convexity, starlikeness, and convexity of a given order. Furthermore, results concerning extreme points, neighborhood properties, and closure under convex linear combinations and integral transforms have been obtained. The sharpness of the main results has been demonstrated by means of appropriate extremal functions.

REFERENCE LIST

[1] Altınkaya, S., Yalçın, S. (2017). Poisson distribution series for certain subclasses of starlike functions with negative coefficients. *An. Univ. Oradea Fasc. Mat.,* 24(2), 5-8.

[2] Amini, E., Fardi, M., Zaky, M. A., Lopes, A. M. & Hendy, A. S. (2023). Inclusion properties of p -valent functions associated with Borel distribution functions. *Mathematics*, *11*(16), 3511.

[3] Aqlan, E. S. (2004). Some problems connected with geometric. Function Theory, Ph.D. Thesis, Pune University, Pune.

[4] El-Deeb, S. M., Bulboacă, T. & Dziok, J. (2019). Pascal distribution series connected with certain subclasses of univalent functions. *Kyungpook Mathematical Journal*, *59*(2), 301-314.

[5] Frasin, B.A., Porwal, S. & Yousef, F. (2021). Subclasses of starlike and convex functions associated with Mittag-Leffler-type Poisson distribution series. *Montes Taurus J. Pure Appl. Math.,* 3(3) 147-154.

[6] Littlewood, J. E. (1925). On inequalities in the theory of functions, *Proc. London Math. Soc.* 23(2) ,481-519.

[7] Miller, S. S. & Mocanu, P. T. (2000). Differential subordinations: theory and applications. CRC Press.

[8] Nazeer, W., Mehmood, Q., Kang, S. M. & Haq, A. U. (2019). An application of Binomial distribution series on certain analytic functions. *J. Comput. Anal. Appl*., 26(1), 11-17.

[9] Porwal, S. & Kumar, M. (2016). A unified study on starlike and convex functions associated with Poisson distribution series. *Afr. Mat.* 27, 10-21.

[10] Silverman, H. (1975). Univalent functions with negative coefficients. Proc. Amer. Math. Soc., 51(1), 109-116.

[11] Themangani, R., Porwal, S. & Magesh, N. (2020). Generalized hypergeometric distribution and its applications on univalent functions. *J. Inequal. Appl.,* Art. ID: 249, 1-10.

[12] Wanas, A. K. & Al-Ziadi, N. A. (2021). Applications of Beta negative binomial distribution series on holomorphic function. *Earthline J. Math. Sci.,* 6(2), 271-292.

[13] Wanas, A. K. & Khuttar, J. A. (2020). Applications of Borel distribution series on analytic functions. *Earthline Journal of Mathematical Sciences*, *4*(1), 71-82.

2025. Volume 9, Number 2 *Pages 168-175*

UDC: 004.89:004.942
DOI: https://doi.org/10.30546/09090.2025.1010.2055

DEVELOPMENT OF THE ARCHITECTURAL SOLUTION OF AN INTELLIGENT DECISION SUPPORT SYSTEM AND SELECTION OF THE OPTIMAL SOLUTION

Vugar SALAHLI[*], Zaur AGHAMALIYEV[2]

[1] *Odlar Yurdu University, Baku, Azerbaijan*

[2] *Academician Y.H. Mammadaliyev Institute of Petrochemical Processes of the Ministry of Science and Education of the Republic of Azerbaijan, Baku, Azerbaijan;*
Azerbaijan State Oil and Industry University, Baku, Azerbaijan

ARTICLE INFO

Article history:
Received:2025-12-23
Received in revised form:2026-01-12
Accepted
Available online

Keywords:
Intelligent decision support system;
Decision support system;
Soft computing;
System architecture;
Business models
2010 Mathematics Subject Classification: 68T01; 68T37; 90B50

ABSTRACT

This study comprehensively investigates theoretical and practical issues related to the development and implementation of intelligent decision support systems used in the management processes of commercial enterprises. Functional and technical requirements for the architectural structure of these systems are identified, and an optimal architectural solution is scientifically selected through comparative analysis of various alternatives. The effectiveness of decision support systems in managerial decision-making is analyzed in terms of efficiency, responsiveness, and accuracy. The impact of implementing intelligent decision support systems on business process effectiveness and rational use of resources in the modern business environment is evaluated. It is substantiated that soft computing methods – such as artificial neural networks, fuzzy logic, and genetic algorithms – significantly expand decision-making capabilities under conditions of uncertainty. It is demonstrated that the proposed modular architectural approach for commercial enterprises enhances management quality and competitiveness

1. Introduction

In the modern era, both public and private sector organizations operating in complex and rapidly changing economic environments require timely, well-grounded, and optimal decision-making. Globalization, intensified competition, market volatility, differentiation of customer demands, and the rapid growth of information flows have significantly increased the complexity of management processes. Under these conditions, the development and implementation of decision support systems (DSS) based on modern information technologies become particularly relevant for improving managerial efficiency, ensuring rational resource utilization, and achieving strategic objectives [1].

Decision support systems are interactive information systems that assist decision-makers by processing, analyzing, and interpreting data collected from various sources. Traditional DSS are primarily aimed at solving structured and semi-structured problems and are based on deterministic and statistical models. Such systems have been successfully applied in various fields. For example, in the aviation sector, analytical decision support systems are used to analyze flight and cargo flows, plan resources, and optimize operational decisions. DSS built on

geographic information systems support spatial decision-making and are widely applied in urban planning, transportation, forestry, and security. In the telecommunications industry, DSS are used for pricing strategy formulation, marketing campaign planning, and analysis of customer behavior [2,3].

In the banking and insurance sectors, decision support systems are applied to solve critical tasks such as credit risk assessment, fraud detection, customer segmentation, and financial risk management [4,5]. These examples demonstrate the significant role of DSS in managerial decision-making across various domains. However, in recent years, the rapid increase in the volume of processed information, data heterogeneity, and uncertainty have limited the capabilities of traditional decision support systems.

In commercial enterprises in particular, the decision-making process is influenced by numerous internal and external factors. Issues such as inventory management, sales planning, pricing strategy selection, customer behavior forecasting, and evaluation of marketing and advertising campaign effectiveness are multi-criteria, nonlinear, and inherently uncertain. Such problems are often difficult to fully describe using classical mathematical models and rigid formal methods. Uncertainty, incomplete information, subjective assessments, and expert knowledge constitute integral components of the decision-making process.

For this reason, at the current stage, the intellectualization of decision support systems—namely, the integration of artificial intelligence and soft computing methods into these systems—is considered essential. Intelligent decision support systems (IDSS) offer broader capabilities compared to traditional systems in handling uncertain and poorly formalized data, modeling expert knowledge, and substantiating alternative decision options. IDSS function as systems that complement human cognitive capabilities by providing analytical and intelligent support to decision-makers in solving complex problems [6–8].

Soft computing methods play a crucial role in the development of intelligent decision support systems. These methods include fuzzy logic, artificial neural networks, genetic and other evolutionary algorithms, probabilistic approaches, and their hybrid combinations [9,10]. Fuzzy logic enables the modeling of imprecise, linguistic, and expert assessments. Artificial neural networks are considered effective tools for learning complex nonlinear relationships and solving forecasting problems. Genetic algorithms are widely applied to identify optimal or near-optimal solutions among multiple alternatives. The combined use of these methods enhances flexibility and adaptability in the decision-making process.

Studies indicate that intelligent decision support systems based on soft computing methods demonstrate high effectiveness in solving problems such as inventory optimization, sales forecasting, customer behavior analysis, and pricing strategy formulation in commercial enterprises [11–16]. Artificial intelligence-based decision mechanisms are widely used in modern e-commerce platforms, CRM systems, and analytical marketing tools, further confirming the practical significance of IDSS in the business environment.

It should be noted that the development of intelligent decision support systems depends not only on the methods employed but also directly on the system architecture. Architectural solutions determine system flexibility, scalability, modularity, integration capabilities with various tools, and security levels. In particular, IDSS used in commercial enterprises must have an open architecture that enables the integration of modules developed in different program-

ming environments within a unified system. This makes the selection of an optimal architectural approach a key scientific and practical problem.

In the author's previous studies, issues related to the application of fuzzy logic and artificial intelligence methods in the analysis of commercial and business processes were investigated, and effective results were obtained in modeling customer behavior and analyzing shopping baskets [17,18]. These studies demonstrated that fuzzy and intelligent approaches more adequately reflect the uncertainty and subjectivity inherent in real business environments. The present article continues this line of research by focusing on the architectural aspects of intelligent decision support systems and aims to develop an optimal architectural solution for commercial enterprises.

Thus, this article formulates requirements for intelligent decision support systems by considering the characteristics of business models used in commercial enterprises, analyzes existing approaches, and proposes a modular architectural solution that ensures the integration of soft computing methods. This approach contributes to improving managerial efficiency, scientifically substantiating decision-making processes, and enhancing the competitiveness of commercial enterprises.

2. Systematization of Business Models in Commercial Enterprises and Requirements for Intelligent Decision-Making Systems

The activities of modern commercial enterprises are carried out in a dynamic market environment characterized by a high level of competition and rapidly changing consumer demands. For this reason, managing business processes based on the capabilities of information technologies has become one of the key factors in ensuring sustainable development and competitive advantage for enterprises. As a result of the integration of computer science, data analysis methods, and artificial intelligence technologies into the business environment, business models applied in commercial enterprises have acquired a more flexible, adaptive, and analytically oriented nature.

The formation of business models in commercial enterprises is directly related to the current level of development and application of information technologies. These models aim to structure the core business processes performed within the enterprise and to ensure their interrelationships. According to modern approaches, the main business processes carried out in commercial enterprises include procurement of goods and materials, inventory management, organization of sales and customer service, marketing and advertising activities, financial management, human resource management, as well as analytical analysis and managerial decision-making [19]. The effective management of these processes has a direct impact on the overall performance indicators of the enterprise.

Figure 1 presents a generalized classification of business models typical for commercial enterprises. As shown in the figure, business models are grouped according to functional directions and encompass operational, tactical, and strategic management levels. This approach enables the hierarchical classification of business processes and facilitates a more systematic construction of decision-making mechanisms.

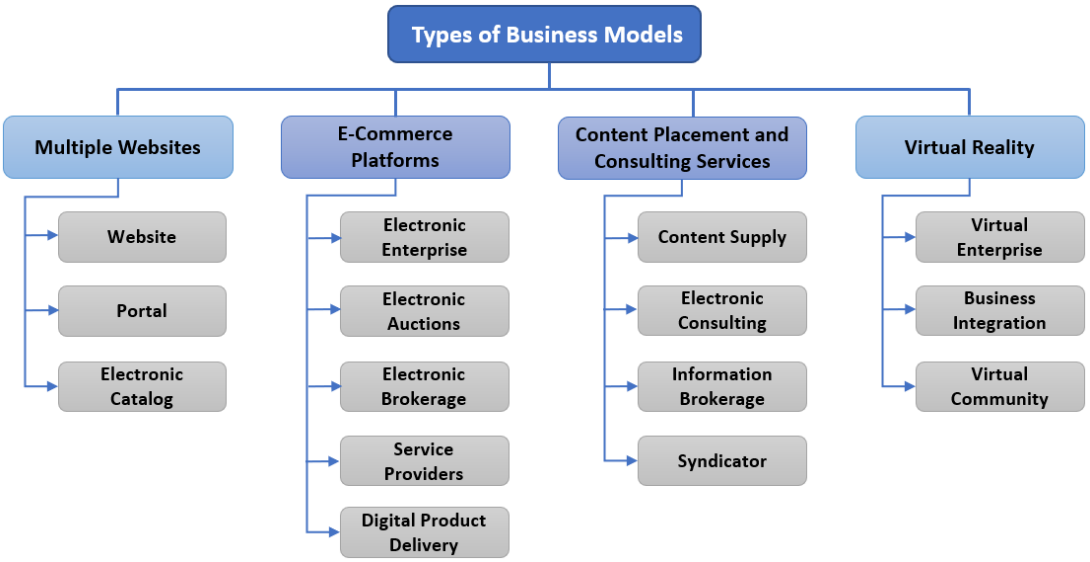


Fig. 1 Classification of business models of commercial enterprises (by operational processes – tactical management – strategic decision-making levels)

The classification shown in Figure 1 clearly demonstrates the integration of business processes with information technologies and the role of analytical tools in decision-making at different management levels. At the operational level, the primary focus is on the execution of day-to-day activities; at the tactical level, on the optimal allocation of resources; and at the strategic level, on making long-term development decisions.

The conducted studies indicate that the effectiveness of methods and tools applied in business process management depends on the extent to which quantitative and qualitative factors influencing these processes are taken into account. The activities of commercial enterprises are shaped by both internal and external factors. Internal factors are related to the organizational structure, resource potential, and management strategy of the enterprise, while external factors are associated with market conditions, the competitive environment, and consumer behavior. A significant portion of these factors is characterized by uncertainty and difficulty in formalization.

Uncertainty, incomplete information, subjective assessments, and ambiguity of objectives complicate the decision-making process and increase the risk of incorrect decisions. Therefore, classical deterministic models have limited capabilities in managing business processes in commercial enterprises. Under such conditions, the application of modern soft computing methods is considered necessary to ensure more substantiated and flexible decision-making.

Soft computing methods represent a set of approaches that include fuzzy logic, artificial neural networks, genetic and other evolutionary algorithms, machine learning, and natural language processing techniques [20]. These methods enable working with imprecise data, modeling expert knowledge, and adapting to changing environments. The use of fuzzy logic in modeling business processes allows decision formation based on linguistic variables, while the application of artificial neural networks facilitates learning complex nonlinear relationships.

The conducted analyses show that intelligent decision-making systems developed for commercial enterprises should be tailored to the enterprise's operational profile and support the daily decision-making processes of management personnel. The methods used in these systems and the results obtained should be understandable and interpretable for users. Solving the same

decision problem using different methods is of significant importance for comparing alternatives and substantiating decisions, which necessitates the modular design of such systems.

Taking the above characteristics into account, the following key requirements are proposed for intelligent decision-making systems in commercial enterprises: an open and scalable architecture, modularity, the ability to integrate modules developed in different instrumental environments, result visualization, data processing and storage, report generation, as well as the presence of security, authentication, and authorization mechanisms. These requirements ensure that intelligent decision-making systems are designed in accordance with the real needs of commercial enterprises.

3. General Architectural Structure of the Intelligent Decision-Making System for Commercial Enterprises

Based on the functional and technical requirements formulated in the second section, a modular and scalable general architectural approach is proposed for the development of intelligent decision-making systems (IDMS) in commercial enterprises. This architectural approach enables the integration of methods and models developed in various instrumental environments within a unified software system, thereby allowing decision-making processes to be carried out in a flexible and adaptive manner.

The general architecture of the proposed intelligent decision-making system is presented in Figure 2. The figure illustrates the main functional modules of the system and their interactions. The architectural structure covers all stages of the decision-making process, starting from data collection and processing, through intelligent analysis and decision generation, and ending with the presentation of results to users.

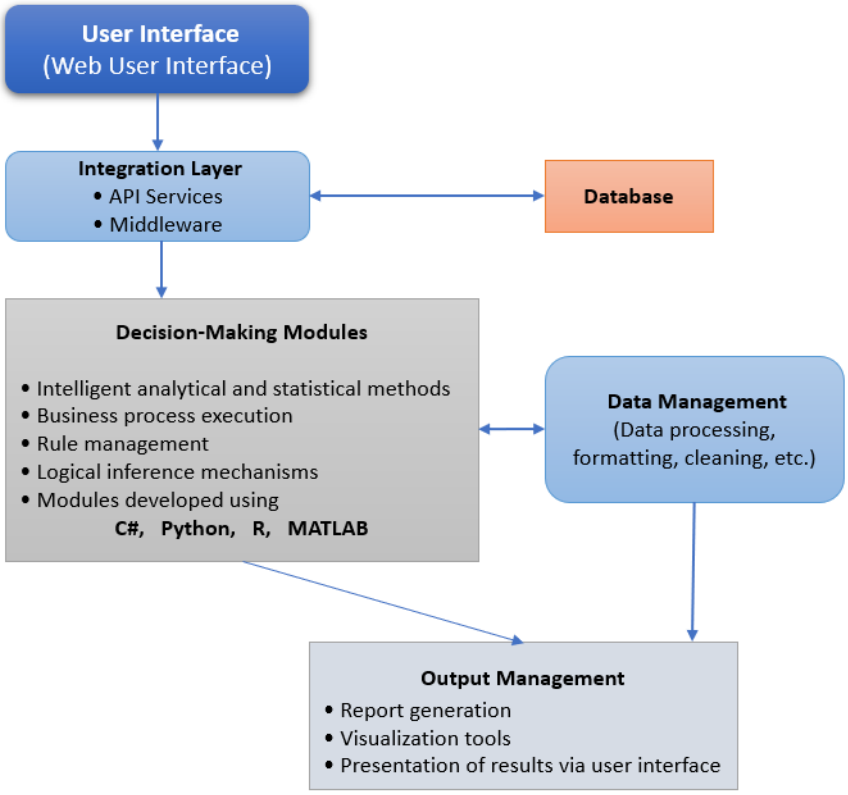


Fig. 2 General architecture of the intelligent decision-making system for commercial enterprises

As shown in Figure 2, the system consists of several main layers. The first layer is the user interface. Since the system is designed based on the web application principle, the user interface should provide remote access capabilities, a user-friendly graphical environment, and functionality tailored to different categories of users. This layer enables interactive communication between managers, decision-makers, and the system.

The second layer is the integration layer, which includes the database, application programming interfaces (APIs), and middleware components. The primary function of this layer is to collect data from various sources, structure the data, and transfer it to the decision-making subsystem. API services ensure flexible and secure data exchange between the database and application modules.

The third and core layer is the decision-making subsystem. This subsystem consists of analytical and intelligent modules developed in different instrumental environments. The main components of the decision-making system are implemented using programming and computational environments such as C#, Java, Python, R, and MATLAB. It should be noted that the core control modules of the system were developed in the Microsoft Visual Studio environment using the C# programming language.

The intelligent analysis and logical inference modules are primarily implemented using fuzzy logic–based Fuzzy Inference System (FIS) tools developed in the MATLAB environment. Integration of these modules into the main software system is achieved through the MATLAB API, which allows dynamic invocation and utilization of various intelligent models. This approach expands the functional capabilities of the system and ensures compliance with the modularity principle.

Example based on MATLAB API:

```
import matlab.engine

eng = matlab.engine.start_matlab()

result = eng.fuzzyInferenceFunction(input_data)

eng.quit()
```

Instead of fuzzyInferenceFunction, any developed FIS module or MATLAB function can be invoked.

One of the key components of the architectural structure is the security module. This module provides authentication and authorization mechanisms, as well as secure communication protocols, to ensure the confidentiality, integrity, and availability of data.

The final layer is the output management module. This module is responsible for the visual presentation of generated decisions and analysis results to users in the form of graphs, charts, and reports. Presenting results in a clear and comprehensible manner facilitates faster and more well-grounded decision-making.

Thus, the proposed architecture of the intelligent decision-making system is characterized by an open, modular, and scalable structure and provides a flexible and functional platform for the effective management of various business processes in commercial enterprises

4. Conclusion

Within the framework of the conducted research, issues related to the development and implementation of intelligent decision support systems (IDSS) aimed at improving the efficiency of managerial decision-making in commercial enterprises were comprehensively investigated. For this purpose, the functional capabilities of existing traditional decision support systems were analyzed, and the methods and approaches used in developing intelligent decision support systems based on artificial intelligence and soft computing techniques were evaluated using various practical examples. As a result of the analysis, functional and technical requirements that take into account the operational characteristics of commercial enterprises were identified.

One of the main outcomes of the study is the proposal of an open, modular, and scalable architectural structure for intelligent decision support systems. This architectural approach enables the integration of analytical and intelligent modules developed in different instrumental environments within a unified software system, thereby expanding the possibilities for efficient utilization of existing internal resources of commercial enterprises. The flexibility of the proposed architecture allows the system to be adapted to the changing needs of enterprises and facilitates the gradual incorporation of new methods and technologies.

For future research, it is considered appropriate to develop a prototype of a real intelligent decision support software system based on the proposed architecture, improve integration mechanisms, and conduct experimental testing across commercial enterprises with different operational profiles. The results obtained can serve as a scientific and methodological foundation for the development and practical implementation of intelligent decision support systems in commercial enterprises of various types.

REFERENCE LIST

1. Power, D. (2008). *Decision Support Systems: A Historical Overview*. In: **Handbook on Decision Support Systems 1.** International Handbooks on Information Systems. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-48713-5_7
2. McRoberts, R. E., Miles, P. D., Barbour, R. J., Gebert, K. M., & Liknes, G. C. (2004). Using forest inventory data and geographic information systems to support forest management decision-making. *Proceedings of the 15th International Workshop on Database and Expert Systems Applications*, Zaragoza, Spain, 576–580. https://doi.org/10.1109/DEXA.2004.1333536
3. Chen, N., Ribeiro, B., & Chen, A. (2016). Financial credit risk assessment: A recent review. *Artificial Intelligence Review*, 45, 1–23. https://doi.org/10.1007/s10462-015-9434-x
4. Lin, C. (2009). Using neural networks as a support tool in decision making for the insurance industry. *Expert Systems with Applications*, 36(3), 6914–6917.
5. Brockett, P. L., Cooper, W. W., Golden, L. L., & Pitaktong, U. (1994). A neural network method for obtaining an early warning of insurer insolvency. *The Journal of Risk and Insurance*, 61(3), 402–424.
6. Abbasov, V. M., Aydinsoy, E. A., Aghamalieva, D. B., & Aghamaliyev, Z. Z. (2024). Assessment of bactericidal efficacy of seaside water samples and automated detection of sulfate-reducing bacteria using computer vision models. *Journal of Engineering Sciences & Modern Technologies*, 1(1), 80–84.
7. Yusibov, F. V., Aghamaliyev, Z. Z., Namazova, S. R., & Aydinsoy, E. A. (2025). Mathematical modelling and optimisation of delayed coking unit operation. *Processes of Petrochemistry and Oil-Refining*, 26(1), 160–177. https://doi.org/10.62972/1726-4685.2025.1.160
8. Abbasov, V. M., Aghamaliyev, Z. Z., Aydinsoy, E. A., & Alimadatli, N. Y. (2023). Modelling an image detection algorithm to evaluate the degree of corrosion. *Processes of Petrochemistry and Oil Refining*, 24(3), 589–596. https://doi.org/10.36719/1726-4685/95/589-596

9. Aliev, R. A., & Aliev, R. R. (2001). *Soft Computing and Its Applications*. World Scientific, Singapore.
10. Zadeh, L. A. (1994). Fuzzy logic, neural networks, and soft computing. *Communications of the ACM*, 37, 77–84.
11. Wankhade, M., Rao, A. C. S., & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55, 5731–5780. https://doi.org/10.1007/s10462-022-10144-1
12. Trunfio, M., & Rossi, S. (2021). Conceptualising and measuring social media engagement: A systematic literature review. *Italian Journal of Marketing*, 267–292. https://doi.org/10.1007/s43039-021-00035-8
13. Rosário, A., & Raimundo, R. (2021). Consumer marketing strategy and e-commerce in the last decade: A literature review. *Journal of Theoretical and Applied Electronic Commerce Research*, 16, 3003–3024. https://doi.org/10.3390/jtaer16070164
14. Adam, M., Wessel, M., & Benlian, A. (2021). AI-based chatbots in customer service and their effects on user compliance. *Electronic Markets*, 31, 427–445. https://doi.org/10.1007/s12525-020-00414-7
15. Kumar, A., Adlakha, A., & Mukherjee, K. (2016). Modeling of product sales promotion and price discounting strategy using fuzzy logic in a retail organization. *Industrial Management & Data Systems*, 116(8), 1418–1444. https://doi.org/10.1108/IMDS-10-2015-0438
16. Wang, L., & Song, H. (2022). E-commerce credit risk assessment based on fuzzy neural network. *Computational Intelligence and Neuroscience*, Article ID 3088915. https://doi.org/10.1155/2022/3088915
17. Veliyev, A., Abdullayev, T., Alekperov, R., & Salahli, V. (2021). Solution of the retail marketing problem of rational choice of the location of trade enterprises using the method of hierarchy analysis and fuzzy set theory. *Advances in Intelligent Systems and Computing*, 1306. https://doi.org/10.1007/978-3-030-64058-3_9
18. Salahli, V. (2020). Analysis of food shopping baskets in a supermarket in terms of different parameters using fuzzy logic. *Open Journal of Business and Management*, 8, 2195–2204. https://doi.org/10.4236/ojbm.2020.85134
19. Ramiz, A., Kahraman, C., Tolga, A. C., Cevik Onar, S., Cebi, S., Oztaysi, B., & Sari, I. U. (2022). The method of ranking business processes on weaknesses based on the theory of fuzzy sets. *Lecture Notes in Networks and Systems*, 504. Springer, Cham. https://doi.org/10.1007/978-3-031-09173-5_17
20. Salahli, V., & Suleymanov, A. (2022). Financial performance analysis with intuitive fuzzy logic and entropy-based multi-criteria decision making method. *Lecture Notes in Networks and Systems*, 362, 711–718. Springer, Cham. https://doi.org/10.1007/978-3-030-92127-9_94

2025. Volume 9, Number 2 *Pages 176-185*

UOT: 004.67:004.8:631.6:633
DOI: https://doi.org/10.30546/09090.2025.210.005

CROPINTEL DATASET: A COMPREHENSIVE AGRO-ENVIRONMENTAL RESOURCE FOR CROP CLASSIFICATION, IRRIGATION PLANNING, AND YIELD ANALYSIS

Artughrul GAYIBOV

agayibov@beu.edu.az
https://orcid.org/0009-0009-7349-0286
Baku Engineering University
Baku, Azerbaijan

ARTICLE INFO

Article history:
Received: 2025-06-17
Received in revised form:2025-06-17
Accepted:2025-07-02
Available online

Keywords:
Precision Agriculture;
Crop Recommendation;
Machine Learning;
Irrigation Prediction;
Yield Forecasting;
Data-Driven Agriculture;
Agro-Environmental Modeling

2010 Mathematics Subject Classifications: 62H30 → 62J02; 68T05; 68T10; 62P12.

ABSTRACT

The growing demand for sustainable agricultural intensification necessitates advanced tools for optimizing crop selection and resource management. This study presents an integrated analytical framework using machine learning to derive agro-environmental intelligence from the comprehensive CropIntel dataset. We evaluated a suite of over fifteen machine learning algorithms for three critical precision agriculture tasks: crop classification, seasonal irrigation requirement prediction, and yield potential forecasting. The results demonstrate the exceptional capability of tree-based models in this domain. Notably, Decision Tree, Bagging, and Gaussian Naïve Bayes classifiers achieved near-perfect accuracy (Acc≈99.7%) in identifying suitable crops based on climatic and edaphic conditions. For regression tasks, LightGBM and Histogram Gradient Boosting models proved most effective, explaining approximately 84% of the variance in irrigation needs (R^2≈0.84) and about 70% of the variance in yield potential (R^2≈0.696). These findings underscore the potential of machine learning to create powerful decision support systems that can guide crop selection, optimize water allocation, and provide reliable yield forecasts. This research contributes to a holistic, data-driven methodology that can enhance the efficiency and sustainability of agricultural practices.

1. INTRODUCTION

With a growing population and increasing environmental pressures like climate change, water scarcity, and land degradation, the world's agricultural sector is at a turning point in its history and must boost productivity to meet these demands (FAO, 2021). According to Chlingaryan, Sukkarieh, and Whelan (2018), precision agriculture has become a paradigm shift that moves away from uniform field management and towards a data-intensive approach that optimizes and manages agricultural production at a granular level. Growing the "right crop for the right land," which optimizes yield potential while reducing resource inputs and environmental impact, is a fundamental component of this paradigm (Rani, Mishra, Kataria, Mallik, & Qin, 2023).

Machine learning (ML) has emerged as a game-changing tool in agriculture thanks to the growth of data from sources like satellites, weather stations, and soil sensors as well as improvements in processing power. Numerous agricultural problems, such as yield forecasting, weed control, crop disease detection, and intelligent irrigation, have been effectively addressed by researchers using machine learning techniques (Navarro-Hellín et al., 2016; van Klompenburg, Kassahun, & Catal, 2020). These uses show how machine learning algorithms can interpret intricate, non-linear relationships in large agro-environmental datasets that are frequently too big for conventional statistical techniques.

Despite these developments, a large portion of current research concentrates on resolving discrete issues. Many studies, for example, create models especially for crop recommendation, while others only concentrate on yield prediction or irrigation optimization. A major research gap is represented by this fragmentation: there aren't enough integrated frameworks that can offer farmers and agricultural planners comprehensive, end-to-end decision support. To enable a thorough cost-benefit analysis prior to the sowing of a single seed, a truly intelligent system should not only recommend which crop to plant but also estimate its resource requirements and potential productivity simultaneously.

Using the new CropIntel dataset, this study attempts to close this gap by creating and assessing a thorough, machine learning-based analytical framework. The three main goals of our research are to: (1) accurately identify the crop that is best suited for a particular set of agro-climatic conditions; (2) forecast the crop's seasonal irrigation water requirements; and (3) estimate the crop's potential yield under those environmental conditions. This study adds to a more comprehensive approach to data-driven agricultural intelligence by addressing these related tasks using a single methodology. The results are meant to offer a strong basis for creating decision-support tools that improve farming operations' profitability and sustainability.

2. LITERATURE REVIEW

The use of machine learning in agriculture has expanded significantly, as evidenced by the abundance of research showing its applicability in a variety of fields. Key findings in the three main areas that are pertinent to our study—crop recommendation, irrigation management, and yield prediction—are summarized in this review.

2.1. Crop Recommendation

Based on soil properties and climate, crop recommendation systems seek to pair crops with the best available land. Rule-based logic was frequently used in early systems, but more recently, ML classifiers have been used to produce predictions that are more accurate and dynamic. For instance, Pudumalar et al. (2017) suggested crops using algorithms such as K-Nearest Neighbors (KNN) and Naïve Bayes. More sophisticated research, like that conducted by Rani et al. (2023), used a Random Forest classifier on a mix of soil and weather data and achieved an accuracy of more than 97%. These studies demonstrate that crop suitability can be accurately predicted by environmental characteristics. They might not always incorporate projections of future resource requirements, though, and frequently function in regional contexts.

2.2. Irrigation Management

In contemporary agriculture, water efficiency is crucial. ML has played a key role in creating "smart" irrigation systems that maximize water use. Navarro-Hellín et al. (2016) created a

decision support system that automated and managed irrigation scheduling using sensor data and predictive models, showing a great deal of promise for water savings. Like this, scientists have forecasted crop water requirements based on meteorological data using methods ranging from adaptive neuro-fuzzy inference systems to support vector regression (SVR) (Goap et al., 2018). Although useful, these studies frequently treat irrigation as a stand-alone issue, unrelated to the initial crop selection procedure or the implications for final yield.

2.3. Crop Yield Prediction

Because of its significant economic ramifications, crop yield prediction is one of the most researched applications of machine learning in agriculture. Numerous algorithms, such as Random Forest, SVR, and deep neural networks, have been used to forecast yields for different crops, according to a systematic review by van Klompenburg et al. (2020). Chlingaryan et al. (2018) pointed out that by more accurately representing the intricate interactions between variables influencing growth, data-driven models usually perform better than conventional process-based or statistical models. Paudel et al. (2022) demonstrated the scalability of machine learning by successfully implementing it for regional crop yield forecasting in Europe. However, yield prediction remains a complex challenge, as performance is highly dependent on the quality and diversity of input data, and models often do not concurrently predict the resource inputs required to achieve that yield.

The literature currently in publication attests to machine learning's effectiveness for agricultural tasks. However, it also highlights a recurring weakness in the integration of these tasks. Using a single, unified dataset, our study contributes to the field by explicitly connecting crop selection (classification) with predictions of its primary resource need (water; regression) and its output (yield; regression).

2. THEORETICAL FRAMEWORK

Agroecology and land evaluation science, which hold that agricultural productivity is a direct result of the interplay between a crop's genetic potential and its surroundings, serve as the foundation for this study. We use machine learning as a tool to model these intricate, well-established relationships empirically.

3.1. Agro-Climatic Suitability

Agro-climatic suitability is the fundamental idea driving our crop classification task. For optimum growth, each crop species has a different set of environmental requirements, such as ranges for soil pH, moisture, temperature, and sunlight (FAO, 1976). The ecological niche of a crop is defined by these conditions. Our framework assumes that an ML model can learn the decision boundaries that define the niche for each crop if it is given a rich set of features that describe these environmental parameters. As a result, the classification task is an empirical approximation of the agroecological principle of matching a crop to its ideal environment rather than just a pattern recognition exercise.

3.2. Land Capability and Productivity

The idea of Land Capability Classification (LCC) serves as the theoretical foundation for the regression tasks of yield and irrigation prediction. According to its innate ability to support agricultural production without degrading, the LCC is a methodical framework for evaluating land (Helms, 1992). Classes I through VIII are used to grade land, with lower classes having

greater productivity potential and fewer restrictions. LCC ratings for both irrigated and non-irrigated conditions are included in the CropIntel dataset. This gives our models a solid theoretical foundation. We predict that while irrigation requirement is a function of the difference between a crop's water requirements and the environment's capacity to supply them (a function of rainfall, soil water holding capacity, etc.), yield potential is highly connected with land capability. Our regression models are thus designed to learn these functional relationships, predicting how land quality and climate translate into specific resource needs and production outcomes.

3. METHODOLOGY

This study employs a quantitative, data-driven approach to build and evaluate predictive models for crop selection, irrigation needs, and yield potential.

4.1. The CropIntel Dataset

The empirical foundation of this research is the CropIntel dataset, a large and comprehensive collection of agro-environmental data. It contains 10,000 unique records, each representing a specific crop scenario defined by a combination of geographic, climatic, topographic, and edaphic (soil) factors. The dataset's strength lies in its breadth, covering 29 countries across North America, Europe, and Asia, encompassing a wide range of Köppen climate zones (e.g., Cfa, Cfb, Dfb, Bsk, Csa) (Arnfield, 2019). This diversity ensures that the models are trained in a wide spectrum of agricultural conditions.

The dataset includes over 35 features for each record, which can be categorized as follows:

- **Locational and Climatic Features:** Country, Subregion, Köppen Climate Zone, Season, Seasonal Average Temperature (°C), Seasonal Total Rainfall (mm), Seasonal Average Humidity (%), and Seasonal Average Daily Sunlight (hours).
- **Topographic Features:** Elevation (m), Slope (degrees), and Aspect (e.g., North-facing, South-facing).
- **Soil Properties:** A detailed soil profile including Soil pH, Soil Organic Carbon (%), texture (Sand, Silt, and Clay %), Bulk Density (g/cm³), Cation Exchange Capacity (meq/100g), Water Holding Capacity (mm/m), Soil Depth (cm), and Drainage Class.
- **Nutrient Status:** Levels of primary macronutrients, including Nitrogen (N), Phosphorus (P), and Potassium (K), measured in kg/ha.
- **Land Capability Indices:** Land Capability Class for both non-irrigated and irrigated scenarios, providing a synthesized measure of land quality.
- **Target Variables:**

1. **Crop:** The specific crop type (over 120 types, such as 'Maize', 'Wheat', 'Apples'), which serves as the target for our classification task.
2. **Seasonal_Avg_Irrigation_Needed_mm:** The estimated volume of irrigation water (in mm) required during the season to supplement rainfall for optimal growth. This is a target for the first regression task.
3. **Yield_Potential_Score:** A normalized index (0-100) that represents the potential productivity of the crop in the given environment. This is the target for the second regression task.

4.2. Data Preprocessing and Feature Selection

The data was preprocessed in several ways prior to model training. First, label encoding was used to transform categorical features (such as Country, Subregion, and Crop) into numerical representations. Second, the median and mode were used to impute numerical and categorical columns, respectively, to account for any missing values. For algorithms that are sensitive to feature magnitudes (such as SVM and MLP) to function at their best, all numerical features were finally standardized using StandardScaler (scikit-learn, StandardScaler, 2019) to have a mean of zero and a standard deviation of one.

We used SelectKBest (sklearn, SelectKBest, 2021) for a feature selection step to simplify the model and concentrate on the most important variables. We determined the ten features with the greatest variation across crop classes for the classification task using the F-statistic from ANOVA (f_classif). We chose the ten features that were most correlated with the corresponding target variables (yield and irrigation) for the regression tasks using the F-statistic from univariate linear regression (f_regression).

4.3. Machine Learning Models and Evaluation

We evaluated a comprehensive suite of over 15 ML models for both classification and regression to ensure a thorough comparison of different algorithmic approaches.

- **Classification Models:** The list included Decision Tree, Support Vector Machine (SVC), K-Nearest Neighbors, Random Forest, Extra Trees, AdaBoost, Histogram Gradient Boosting, Bagging, Logistic Regression, Ridge Classifier, SGD Classifier, Gaussian Naïve Bayes, and a Multi-Layer Perceptron (MLP) Classifier, as well as XGBoost and LightGBM classifiers.
- **Regression Models:** A parallel set of regressors was used, including Decision Tree, SVR, K-Nearest Neighbors, Random Forest, Extra Trees, AdaBoost, Histogram Gradient Boosting, Bagging, Ridge, Lasso, ElasticNet, SGD Regressor, Huber Regressor, MLP Regressor, XGBoost, and LightGBM.

A rigorous three-fold cross-validation process was used to evaluate the model's performance. The dataset was divided into three folds, two of which were used for training and one of which was used as a validation set. After that, the three folds' performance metrics were averaged.

- **Evaluation Metrics:**
  - For **classification**, we measured **Accuracy**, and weighted **Precision**, **Recall**, and **F1-Score** to account for the multi-class nature of the problem.
  - For **regression**, we measured the **Coefficient of Determination (**R^2**)**, **Mean Absolute Error (MAE)**, and **Mean Squared Error (MSE)**.

4. RESULTS

Below are the empirical findings from our three analytical tasks: yield prediction, irrigation prediction, and crop classification. Tables and figures provide a summary and visual representation of the performance of the different models.

5.1. Crop Classification Performance

The crop classification task yielded exceptionally clear results, with a distinct separation between high-performing and low-performing models. As shown in Table 1 and Figure 1, several algorithms were able to predict the correct crop type with near-perfect accuracy.

Table 1. Performance Metrics for Crop Classification Models

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree Classifier	0.997	0.997	0.997	0.997
Bagging Classifier	0.997	0.997	0.997	0.997
Gaussian Naive Bayes	0.997	0.994	0.997	0.995
Extra Trees Classifier	0.956	0.954	0.956	0.950
Random Forest Classifier	0.896	0.890	0.896	0.886
MLP Classifier	0.803	0.791	0.803	0.788
Logistic Regression	0.524	0.474	0.524	0.467
K-Nearest Neighbors	0.362	0.334	0.362	0.341
LightGBM Classifier	0.224	0.238	0.224	0.195

The most remarkable outcome is the Acc≈99.7% accuracy attained by Gaussian Naïve Bayes classifiers, Bagging (which starts with decision trees), and the standard Decision Tree classifier. This suggests that the CropIntel dataset's environmental and soil characteristics include distinct, distinct signals that specify each crop's suitability. With accuracies of 95.6% and 89.6%, respectively, ensemble techniques like Random Forest and Extra Trees also demonstrated strong performance. The non-linear character of the decision boundaries was highlighted by the difficulties faced by distance-based models (KNN) and linear models (such as logistic regression).

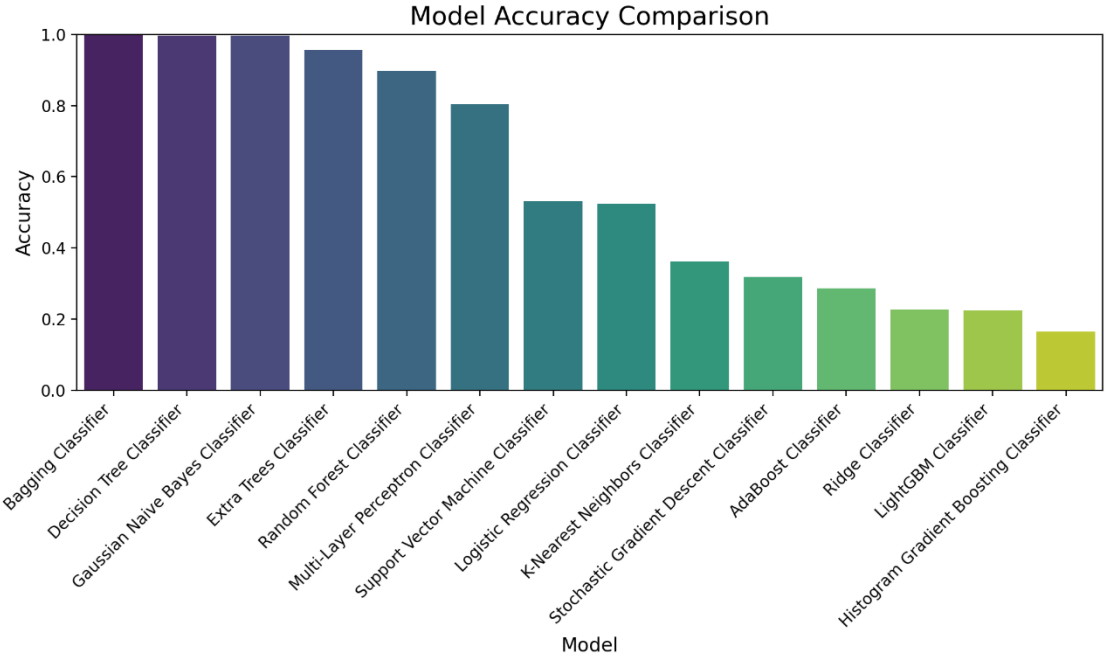


Figure 1. Crop Classification Accuracy Across Models

5.2. Irrigation Requirement Prediction Performance

Ensemble regression models showed good predictive power for the task of forecasting the seasonal average irrigation required. The performance is displayed in Table 2 and Figure 2, where each model's R^2 score represents the percentage of variance it can account for.

Table 2. Performance Metrics for Irrigation Requirement Regression Models

Model	R^2-Score	MAE (mm)	MSE (mm^2)
LightGBM Regressor	0.842	5.95	215.90
Hist. Gradient Boosting	0.841	5.82	217.61
Random Forest Regressor	0.781	6.30	300.37
Extra Trees Regressor	0.771	6.90	314.07
Bagging Regressor	0.754	6.54	337.13
Decision Tree Regressor	0.620	6.60	521.24
MLP Regressor	0.536	12.37	635.82
K-Nearest Neighbors	0.450	12.40	754.28
Support Vector Regressor	0.234	13.10	1049.63

With an R^2 of roughly 0.84, the LightGBM and Histogram Gradient Boosting regressors were the best performing. This means that 84% of the variation in irrigation needs based on input features can be explained by these models. A high level of precision was indicated by these models' extremely low Mean Absolute Error, which was between 5.8 and 5.9 mm. Linear and distance-based models were much less successful than other tree-based ensembles, such as Random Forest and Extra Trees.

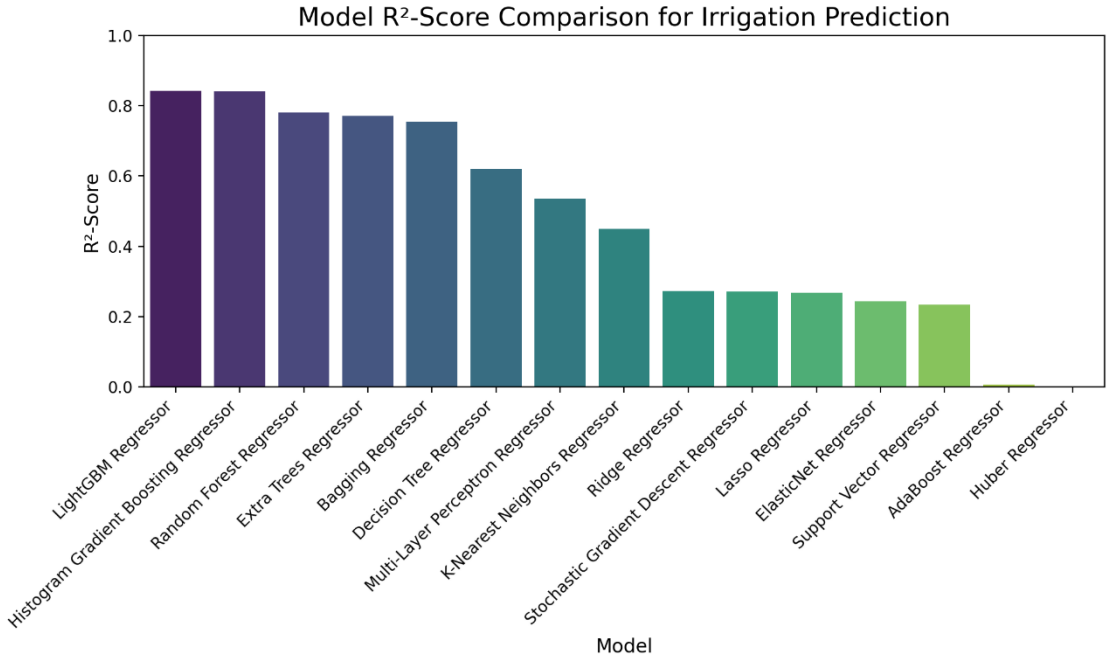


Figure 2. R^2 Scores for Irrigation Prediction Models

5.3. Yield Potential Prediction Performance

Of the three tasks, predicting the Yield Potential Score was the most difficult, but the models still performed admirably. Table 3 and Figure 3 provide a summary of the findings.

Table 3. Performance Metrics for Yield Potential Regression Models

Model	R^2-Score	MAE	MSE
LightGBM Regressor	0.696	5.48	49.84
Hist. Gradient Boosting	0.695	5.47	50.04
Extra Trees Regressor	0.630	6.09	60.69
Random Forest Regressor	0.620	6.15	62.24
Bagging Regressor	0.582	6.45	68.52
MLP Regressor	0.518	7.05	78.94
Support Vector Regressor	0.456	7.57	89.06
AdaBoost Regressor	0.429	7.79	93.55
Decision Tree Regressor	0.228	8.64	126.40

With an R^2 of almost 0.70, LightGBM and Histogram Gradient Boosting once again demonstrated their superior performance. This indicates that they were able to account for roughly 70% of the Yield Potential Score's variation. At about 5.5 points on the 0-100 scale, the MAE was remarkably low. Although this is an impressive result, the lower R^2 when compared to the irrigation task implies that more intricate interactions or variables not fully represented in the dataset affect yield.

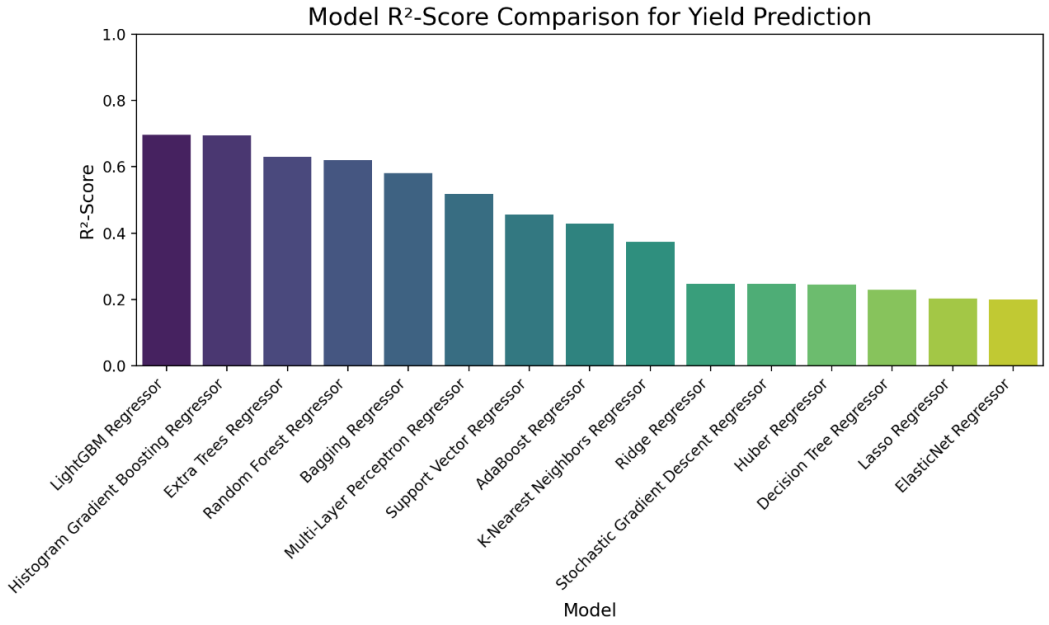


Figure 3. R^2 Scores for Yield Potential Prediction Models

5. DISCUSSION

The study's findings provide important new information about how machine learning can be used to achieve integrated agro-environmental intelligence. Agro-climatic niches for crops are clearly defined and learnable, as evidenced by the exceptional performance in the classification task, with nearly perfect accuracy from multiple models. The fundamental idea of data-driven crop recommendation is validated by the features in the CropIntel dataset, which are adequate to define these niches. Because of the dataset's thoroughness and cleanliness, this result is consistent with—and even surpasses—the high accuracies documented in the body of existing literature (e.g., Rani et al., 2023). Practically speaking, automated systems can offer farmers extremely trustworthy crop recommendations, which could increase land use efficiency and avoid crop

failures brought on by environmental mismatches.

Ensemble tree-based models—especially LightGBM and Histogram Gradient Boosting—perform better in the regression tasks. When modelling agricultural systems, their capacity to capture intricate, non-linear relationships and threshold effects is essential. With an R^2 of 0.84 for irrigation, it is possible to predict water requirements with high confidence. This ability serves as the basis for developing tools that assist farmers in scheduling irrigation events, allocating water resources, and ultimately conserving water—a crucial objective in sustainable agriculture (Navarro-Hellín et al., 2016).

With a top R^2 of about 0.70, the yield prediction task demonstrates the inherent difficulty of predicting agricultural output. Although most of the variance was explained by the models, the remaining 30% points to the impact of variables not included in the dataset, such as the pressure from pests and diseases, severe weather conditions (like hail or frost), or fine-grained management choices. This outcome is in line with the larger body of research on yield prediction, which recognizes that crop growth is multifactorial (Chlingaryan et al., 2018; van Klompenburg et al., 2020). However, a model that accounts for 70% of yield variance is a useful tool for risk assessment, policymaking, and planning.

This work's integrated approach is its main contribution. We show the potential for a comprehensive decision support system by tackling classification, irrigation, and yield prediction in a single framework and dataset. A more thorough and informed planning process could be possible with such a system, which could, for example, not only suggest planting wheat but also estimate its probable water requirements (e.g., 150 mm) and potential yield (e.g., a score of 85/100).

6. CONCLUSION

This study successfully demonstrated the power of an integrated machine learning framework for enhancing agro-environmental intelligence. Using the comprehensive CropIntel dataset, we have shown that it is possible to:

1. **Classify suitable crops** with exceptional accuracy (Acc≈99.7%), providing a reliable basis for crop recommendation systems.
2. **Predict seasonal irrigation needs** with high fidelity (R^2≈0.84), offering a valuable tool for water resource management.
3. **Forecast crop yield potential** with strong performance (R^2≈0.70), enabling better planning and risk assessment.

Our results demonstrate that models based on ensemble trees are especially effective at representing the intricate, non-linear dynamics present in agricultural systems. This study offers a comprehensive approach that goes beyond discrete prediction tasks, opening the door for increasingly complex, comprehensive precision agriculture decision support systems.

There are significant practical ramifications for farmers, agronomists, and legislators. The models created here can optimize the distribution of limited water resources, guide strategic crop selection decisions, and offer useful projections for production scheduling. Although the dataset used in this study is well-structured, a critical next step is to validate and modify these models using actual, in-field data. Future research might also concentrate on integrating economic

factors to optimize profitability in addition to agronomic suitability, as well as dynamic variables like real-time weather forecasts and remote sensing data. We can open new possibilities for creating a food system that is more resilient, productive, and sustainable if we keep bridge the gap between data science and agricultural science.

REFERENCES

1. Chlingaryan, A., Sukkarieh, S., & Whelan, B. (2018). Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review. *Computers and Electronics in Agriculture*, *151*, 61–69.
2. FAO. (1976). *A framework for land evaluation*. Food and Agriculture Organization of the United Nations.
3. FAO. (2021). *The State of the World's Land and Water Resources for Food and Agriculture – Systems at breaking point (SOLAW 2021)*. Rome: FAO.
4. Goap, A., Sharma, D., Shukla, A. K., & Krishna, C. R. (2018). An IoT based smart irrigation management system using machine learning and open-source technologies. *Computers and Electronics in Agriculture*, *155*, 41-49.
5. Helms, D. (1992). The tonnage of the land: A history of the Soil Conservation Service. US Department of Agriculture, Soil Conservation Service.
6. Arnfield, A. J. (2019). Koppen climate classification | Description, Map, & Chart. In Encyclopædia Britannica. https://www.britannica.com/science/Koppen-climate-classification
7. Navarro-Hellín, H., Martínez-del-Rincon, J., Domingo-Miguel, R., Soto-Valles, F., & Torres-Sánchez, R. (2016). A decision support system for managing irrigation in agriculture. *Computers and Electronics in Agriculture*, *124*, 121–131.
8. Paudel, D., Boogaard, H., de Wit, A., van der Velde, M., Claverie, M., Nisini, L., Janssen, S., Osinga, S., & Athanasiadis, I. N. (2022). Machine learning for regional crop yield forecasting in Europe. *Field Crops Research*, *276*, 108377.
9. Pudumalar, S., Ramanujam, E., Rajashree, R., Kavya, C., Kaviya, T., & Monisha, B. (2017). Crop recommendation system for precision agriculture. *In 2016 Eighth International Conference on Advanced Computing (ICoAC)* (pp. 32-36). IEEE.
10. Rani, S., Mishra, A. K., Kataria, A., Mallik, S., & Qin, H. (2023). Machine learning-based optimal crop selection system in smart agriculture. *Scientific Reports*, *13*(1), 15997.
11. Van Klompenburg, T., Kassahun, A., & Catal, C. (2020). Crop yield prediction using machine learning: A systematic literature review. *Computers and Electronics in Agriculture*, *177*, 105709.
12. scikit-learn, StandardScaler. (2019). *StandardScaler*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
13. sklearn, SelectKBest — *scikit-learn 0.23.0 documentation*. (2021.). Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

2025. Volume 9, Number 2 Pages 186-195

UDC: 004.932.2
DOI: https://doi.org/10.30546/09090.2025.1010.2015

BENCHMARKING 2D AND 3D OBJECT DETECTION MODELS ON DIFFERENT EDGE COMPUTING PLATFORMS

Shahla URALOVA[1][0009-0004-0146-5173], Rasim MAHMUDOV[2][0009-0006-1385-1412], Amil BABAYEV[1][0009-0005-4823-6201] and Qurban YUSUFOV[1][0009-0004-0530-9709]

[1] *Cyber Security and Computer Engineering, Baku Engineering University, Baku, Azerbaijan*
suralova@std.beu.edu.az, amilb@beu.edu.az, quyusufov@beu.edu.az

[2] *Information Technology and Programming, Baku Engineering University, Baku, Azerbaijan rkmahmudov@gmail.com*

ARTICLE INFO

Article history:
Received: 2025-09-01
Received in revised form:2025-09-11
Accepted:2026-01-12
Available online

Keywords:
Edge computing, Convolutional neural networks, Benchmarking.

2010 Mathematics Subject Classifications: 68T07, 68T45

ABSTRACT

The deployment of artificial intelligence models on edge computing platforms is imperative for applications necessitating real-time perception, including object detection and gesture recognition. This paper presents a thoroughgoing benchmarking study of 2D and 3D convolutional neural network models across a range of representative edge devices, including the NVIDIA Jetson AGX Orin, Jetson Nano, and Raspberry Pi 3. The study utilizes gesture recognition and mask detection as the primary use cases for evaluating the performance of these models. We evaluated the inference time, memory footprint, and power consumption under varying input resolutions and sequence lengths, thereby capturing the latency–accuracy–energy trade-offs that govern practical deployment. The results show that AGX Orin consistently handles high-demand scenarios with low latency and stable throughput, whereas Jetson Nano offers competitive efficiency in energy-constrained settings. Raspberry Pi 3, while limited, establishes a useful baseline for CPU-only operation and highlights optimization needs. We also examined the effects of quantization and runtime optimizations and reported configurations that deliver robust performance without substantial accuracy loss. This study synthesizes these findings into actionable guidelines for selecting models, precisions, and device classes according to operational constraints. By providing a reproducible evaluation framework and device-aware recommendations, this study supports developers in balancing speed, accuracy, and energy efficiency, thereby enabling reliable real-time AI on resource-limited edge platforms. Our methodology uses warm-up iterations, repeated trials per configuration, synchronized power sampling, and pre-processing to ensure fair comparisons.

1 Introduction

In the evolving landscape of edge computing, the deployment of artificial intelligence (AI) models for real-time applications, such as object detection and gesture recognition, has become increasingly crucial. Unlike cloud-based approaches, where reliance on centralized servers often introduces delays, edge platforms offer the opportunity to process data locally, thereby enabling faster and more reliable inference. This study presents a comprehensive benchmarking of both 2D and 3D convolutional neural network (CNN) models across multiple edge devices, including NVIDIA's Jetson AGX Orin, Jetson Nano, and Raspberry Pi, to evaluate their performance under varying levels of computational complexity and input demand.

Traditional cloud-based solutions for AI inference often struggle with latency, bandwidth limitations, and privacy concerns, particularly in time-sensitive applications such as health monitoring, autonomous driving, and surveillance. Edge computing directly addresses these challenges by enabling on-device processing, reducing the dependency on external servers, and supporting real-time decision-making. The widespread deployment of IoT devices, smart cameras, and mobile robots further emphasizes the need for efficient and robust AI inference at the edge, making the selection of appropriate models and hardware platforms a critical research topic.

In this context, the present study investigates the performance of 3D CNN models for gesture recognition using the 20BN and Jester datasets, as well as 2D CNN and ResNet50 models for real-time mask detection using the MaskedFace-Net and RMFD datasets. The experiments are conducted on three representative edge devices: Jetson AGX Orin, Jetson Nano, and Raspberry Pi 3. The key specifications of these devices are summarized in Table 1. The differences in CPU/GPU architecture, memory capacity, and power requirements serve as a foundation for understanding how hardware constraints affect model performance.

Table 1. Specification of used edge devices.

Specification	**AGX Orin**	**Jetson Nano**	**Raspberry Pi 3**
CPU	8-core ARM Cortex-A78AE @ 2.188 GHz	4-core ARM Cortex-A57 @ 1.43 GHz	Quad-core ARM Cortex-A53 @ 1.2 GHz
GPU	2048 CUDA cores	128-core Maxwell GPU	Broadcom VideoCore IV
Memory	32 GB LPDDR5	4 GB LPDDR4	1 GB LPDDR2
Storage	64 GB eMMC	microSD	MicroSD
Power	15-60 W	5-10 W	5 W
JetPack / OS	JetPack 6.0	JetPack 4.2.1	N/A (Uses Raspbian)
Framework	TensorFlow, PyTorch	TensorFlow, PyTorch	TensorFlow, PyTorch
AI Performance	Up to 275 TOPS	0.5 TOPS	Limited AI capabilities

Although several benchmarking studies have explored deep neural network (DNN) deployment on embedded systems, most have focused either on single-model scenarios (e.g., only 2D CNNs) or single-device evaluations (e.g., Jetson Nano or TX2). Moreover, much of the prior work primarily emphasizes accuracy and inference speed while neglecting equally important considerations, such as memory footprint and energy efficiency. This gap underscores the need for a more comprehensive evaluation framework that captures the trade-offs between computational power, resource utilization, and energy consumption, thereby enabling informed decisions for practical edge AI deployments.

Key performance metrics, such as inference time, memory usage, and CPU and GPU power consumption, were systematically analyzed across these models and devices. This research highlights the trade-offs between computational power and energy efficiency, which are crucial for optimizing the deployment of AI models in resource-constrained environments. This benchmarking provides valuable insights into selecting appropriate hardware and model configurations for specific edge computing applications, focusing on achieving a balance between speed, accuracy, and power consumption of the model.

Here are the main contributions of this paper:

- We present a comprehensive benchmarking of 3D CNN models for gesture recognition using the 20BN and Jester datasets, evaluating their scalability across different frame input sizes.
- We evaluate 2D CNN and ResNet50 models for face mask detection using the Masked Face-Net and RMFD datasets, providing insights into their robustness under varying image resolutions.
- We analyze and compare edge devices, including Jetson AGX Orin, Jetson Nano, and Raspberry Pi 3, focusing on inference time, memory usage, and power consumption.
- We provide practical guidelines for deploying AI models on edge platforms, highlighting trade-offs between speed, accuracy, and energy efficiency that are critical for real-world applications.

This comprehensive set of contributions provides a robust benchmarking for understanding AI model deployments on edge computing platforms.

2 Related work

In recent years, many studies have benchmarked the deployment of deep neural networks (DNNs) on embedded/edge hardware, especially the NVIDIA Jetson family (Nano, TX2, Xavier/AGX, Orin), for image classification, object detection, face/gesture recognition, and scene understanding. Prior studies typically report end-to-end performance under strict compute and power budgets and contrast 2D vs. 3D CNN architectures, runtime stacks (TensorRT, ONNX Runtime, PyTorch/TensorFlow), and model-level optimizations such as quantization/pruning and kernel fusion [1–5], [7–9]. However, evaluation protocols vary widely across studies terms of metrics (latency/throughput, accuracy, memory footprint, CPU/GPU power draw) and experimental knobs (input resolution, sequence length, batch size, device power/thermal modes), which complicates cross-paper comparison despite valuable insights [1–5], [15]. Therefore, we motivate a unified benchmark contrasting 3D CNNs for gesture recognition and 2D CNN/ResNet-50 for mask detection on common edge devices, with consistent measurements of inference time, memory usage, and CPU/GPU power. Techniques such as approximations [6], lightweight architectures like MobileNetV2 [8] and MobileNetV3 [9], and quantization/pruning have been proposed to reduce compute and memory overhead for embedded AI.

2.1 DNN Deployment on Embedded Boards

A substantial line of work evaluates DNN workloads on resource-constrained embedded boards, with particular attention to Jetson devices. Süzen et al. compare Jetson Nano, Jetson TX2, and Raspberry Pi 4 using a CNN for fashion-product classification, reporting power consumption, GPU/CPU/RAM utilization, accuracy, and total cost across datasets from 5k to 45k images [1]. Complementing device-level comparisons, EdgeFace targets face recognition under edge constraints and shows that careful architectural choices can retain accuracy while improving computational efficiency for small form-factor boards [2]. Broader edge-class benchmarking on phone-grade SoCs likewise links memory bandwidth and on-device acceleration to end-to-end throughput [15], while direct Nano vs. Raspberry Pi studies emphasize the role of CUDA-capable GPUs for real-time inference at moderate resolutions. Concurrently, model-side techniques such as parameter sharing, approximations, and quantification/pruning are employed to minimize computing and memory expenditures. [6], [8], [9].

Relation to our study. These works establish feasibility and highlight cost/power trade-offs, but typically examine single tasks or device classes. We extend this line by using a unified protocol over heterogeneous devices (AGX Orin, Nano, RPi 3) and two application families (gesture, mask detection), while jointly reporting latency, memory, and CPU/GPU power.

2.2 Learning Spatiotemporal Features with 3D CNNs for Gesture Recognition

Beyond 2D image pipelines, several studies explore 3D inference and spatiotemporal or geometric modeling on embedded hardware. Ullah and Kim [3] benchmark the Jetson platform for 3D point-cloud and hyperspectral classification, emphasizing that memory bandwidth and on-chip acceleration critically impact throughput when input dimensionality grows. Similarly, Choe et al. [4] analyze 3D object detectors on Jetson devices, showing that detector capacity, input resolution, and kernel-level optimizations (e.g., inference runtimes and deployment stacks) jointly determine real-time viability. Collectively, these results reinforce a common observation: 3D workloads impose heavier compute and memory pressure than comparable 2D pipelines, tightening power and thermal margins on embedded GPUs. Early works such as C3D [10] and I3D [11] demonstrated the effectiveness of 3D CNNs for spatiotemporal feature learning, though their computational cost remains a challenge on edge devices.

Relation to our study. Our benchmark brings this perspective to video-based gesture recognition with 3D CNNs, contrasting it against 2D CNN pipelines under matched measurement settings to quantify the cost of temporal modeling.

2.3 AI for Health Safety Measures in Edge Computing

Edge-resident perception has also been explored for public-health and safety applications. Jovović et al. demonstrate face-mask detection on Jetson-class devices, integrating ML with IoT to enable responsive, on-device screening in pandemic scenarios [5]. Widely used mask datasets include MaskedFace-Net, which distinguishes correctly and incorrectly worn masks [13], and the real-world RMFD collection [14], both enabling reproducible assessment of lightweight 2D CNNs and higher-capacity backbones such as ResNet-50 [7].

Relation to our study. We evaluate ResNet-50 on MaskedFace-Net and a lighter 2D CNN on RMFD, directly measuring the latency–memory–power envelope that governs deployability of mask-detection models on edge hardware.

2.4 Summary and Research Gap

Across [1–5],[10–15], prior work often focuses on single-model families or single devices and primarily reports accuracy/latency, while memory footprint and CPU/GPU power draw are covered less systematically. In addition, protocol heterogeneity—datasets, input sizes, runtimes—complicates cross-paper comparison. Unlike ad-hoc reports, we hold preprocessing, runtime stack, and power settings constant across devices [15], enabling fair attribution of performance differences to model capacity and input complexity. Our study addresses these limitations with a task-balanced, cross-device benchmark that (i) spans 3D gesture and 2D mask detection pipelines, (ii) enforces consistent measurement (batch size, warm-up, metrics), and (iii) reports inference time, peak memory, and CPU/GPU power draw together to surface actionable deployment trade-offs.

3 Benchmark analysis

3.1 Experimental Setup

We conduct a comprehensive evaluation of two representative application families—specifically gesture recognition and face-mask detection—across three distinct embedded edge computing devices: the NVIDIA Jetson AGX Orin, Jetson Nano, and Raspberry Pi 3. Our benchmark methodology encompasses both 3D and 2D inference regimes and implements a carefully standardized protocol to guarantee fair, reproducible, and meaningful comparisons across these diverse hardware platforms. Figure 1 provides a detailed overview of the experimental setup, including the tasks being evaluated, the model architectures employed (specifically 3D CNNs for temporal analysis and both lightweight and ResNet-based 2D CNNs for spatial analysis), the datasets utilized for training and validation (including 20BN-Jester, Jester, MaskedFace-Net, and RMFD), and the comprehensive range of input dimensions strategically selected to simulate both computationally light and heavy workloads. In the subsequent sections of this paper, we elaborate on the precise resolutions, sequence lengths, runtime environments, and measurement procedures implemented throughout our testing process. The following text is intended to provide a comprehensive overview of the subject matter. The methodological elements outlined above collectively establish a thorough and systematic blueprint for accurately evaluating critical performance metrics. These metrics include processing latency, memory utilization footprint, and power consumption characteristics. The blueprint effectively isolates the intrinsic hardware capabilities of each device from potentially confounding software implementation differences. For gesture recognition, we used the 20BN and Jester datasets [12], which are widely adopted for benchmarking hand-gesture classification.

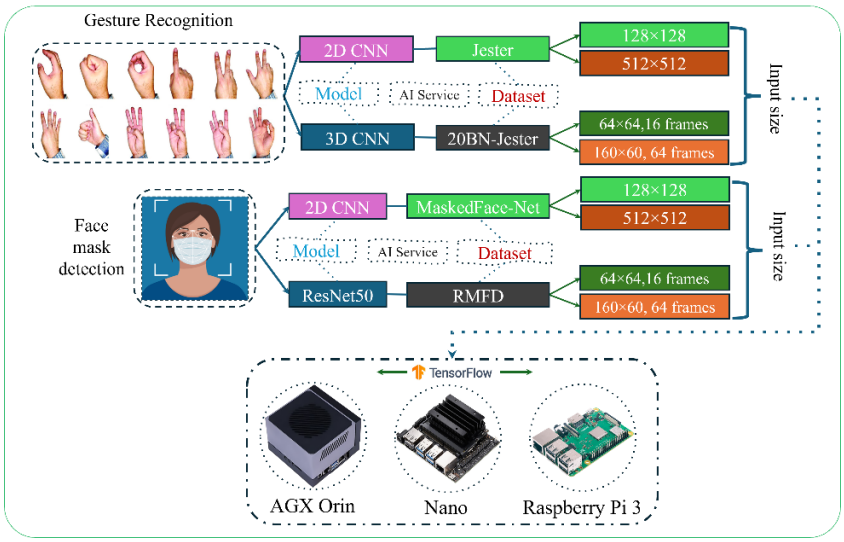


Fig. 1. Architecture of benchmarking system

For mask detection, we evaluate two model families: a lightweight 2D CNN on the RMFD dataset (inputs at 300 × 300 and 516 × 516), and a ResNet-50 on the MaskedFace-Net dataset (224 × 224 and 960 × 544). This combination reflects both constrained and computationally demanding pipelines, and allows us to capture how model capacity interacts with device limitations.

All devices were configured in performance-oriented modes to reduce throttling and stabilize results. On Jetson boards, experiments were run under JetPack environments (v6.0 for AGX Orin and v4.2.1 for Nano), while the Raspberry Pi used Raspbian with TensorFlow/PyTorch backends.

We locked GPU clocks where possible to minimize DVFS-related variance. Unless otherwise specified, batch size = 1 and single-stream inference were used.

Measurements were taken under consistent conditions. Latency is reported as the median inference time over repeated runs after discarding warm-up iterations. Memory usage reflects peak resident set size (RSS) during inference. CPU and GPU power draw were recorded through on-device telemetry (tegrastats for Jetson boards and vcgencmd for Raspberry Pi) at 1 Hz sampling, then averaged. To account for runtime variability, each experiment was repeated three times, and the most stable run was reported.

This setup provides a transparent basis for comparing devices with widely different compute capabilities, memory budgets, and power envelopes, and lays the foundation for the benchmarking results reported in Section 3.2.

3.2 Benchmarking results

The performance evaluations summarized in Table 2 reveal consistent though device-dependent patterns across models, datasets, and input sizes. Under a unified protocol (batch size = 1; post-warm-up medians), we report inference time (ms), peak memory (GB), and CPU/GPU power (mW) for both 3D gesture and 2D mask-detection pipelines on AGX Orin, Jetson Nano, and RPi 3. Latency scales predictably with spatiotemporal complexity and resolution, while memory and power track model capacity; consequently, heavier inputs and larger backbones widen the gap between devices. Notably, feasibility limits appear for high-capacity models on low-end boards (e.g., N/F on RPi 3 with ResNet-50), underscoring the role of memory headroom and acceleration. For readability, Fig. 2 visualizes latency differences and Fig. 3 consolidates memory footprints; detailed numeric results remain in Table 2 for exact cross-device comparisons.

Under a unified protocol, latency is reported as the median over repeated runs after warm-up (batch size = 1, single stream). Peak memory refers to the maximum resident memory observed during inference. CPU/GPU power values are collected from on-device telemetry and averaged over runs. We additionally estimate energy per inference as

$$E \approx (P_{CPU} + P_{GPU}) \times t (1)$$

where P is power (W) and t is latency (s). This approximation enables a consistent comparison of efficiency across devices and input regimes.

Table 2. Benchmarking results of AI models across different hardware platforms. Column "Alg." indicates the algorithm type, where "GR" represents **gesture recognition** and "MD" represents **mask detection** models. "N/F" denotes cases where execution was not feasible.

Alg.	Model	Data set	Input Size	Inference Time (ms)			Memory (G)			CPU (Power/mW)			GPU (Power/mW)		
				AGX Orin	*Nano*	*RPi 3*	*AGX Orin*	*Nano*	*RPi 3*	*AGX Orin*	*Nano*	*RPi 3*	*AGX Orin*	*Nano*	*RPi 3*
GR	3D CNN	20BN Jester	64x64, 16F	15	50	60	2.5	1.8	0.7	360	750	850	950	370	400
			160x160, 64F	40	122	183	4.0	3.0	0.83	420	820	920	1200	550	600
	2D CNN	Jester	128x128	10	35	35	1.8	1.2	0.62	320	680	750	800	290	330
			512x512	30	102	144	3.5	2.5	0.84	380	760	840	1050	450	500
MD	ResNet50	Masked Face-Net	224x224	35	105	N/F	4.5	3.29	N/F	450	850	N/F	1150	600	N/F
			960x544	50	150	N/F	5.5	3.81	N/F	500	900	N/F	1250	650	N/F
	2D CNN	RMFD	300x300	20	60	70	2.0	1.52	0.8	340	700	780	900	350	380
			516x516	25	80	90	2.8	2.15	0.83	370	770	840	1000	420	460

3.3 Gesture recognition with 3D CNNs.

For the lighter clip (64×64, 16 F), AGX Orin achieves 15 ms processing time compared to 50 ms on Jetson Nano and 60 ms on RPi 3—approximately 3.3× and 4.0× faster than Nano and RPi 3, respectively (Table 2). When the clip size increases to 160×160, 64 F, latency scales predictably: 40 ms (Orin), 122 ms (Nano), and 183 ms (RPi 3), representing about 3.0× and 4.6× performance gains for Orin. Peak memory usage follows a similar pattern (Orin: 2.5→4.0 GB, Nano: 1.8→3.0 GB), while GPU power consumption increases with input complexity (Orin: 950→1200 mW, Nano: 370→550 mW; Table 2, Fig. 2–3).

Despite Orin's higher instantaneous GPU power draw, its energy per inference remains more efficient due to shorter processing times (approximately 14 mJ at 64×64, 16 F versus 18.5 mJ on Nano; 48 mJ at 160×160, 64 F versus 97 mJ on Nano). This demonstrates that in temporal models, superior performance translates to better energy efficiency.

3.4 Gesture recognition with 2D CNNs (Jester).

For the 128×128 resolution, the AGX Orin device demonstrates exceptional performance with an execution time of just 10 ms, while both the Jetson Nano and Raspberry Pi 3 require significantly longer processing times of 35 ms (approximately 3.5 times slower than Orin). When the resolution is increased to 512×512, the processing latency grows proportionally across all devices, reaching 30 ms for Orin, 102 ms for Nano, and 144 ms for RPi 3. As the spatial resolution is increased, there is an attendant increase in memory consumption. The range of memory consumption is from 1.8 GB (Orin), 1.2 GB (Nano), and 0.62 GB (RPi 3) at lower resolutions to 3.5 GB (Orin), 2.5 GB (Nano), and 0.84 GB (RPi 3) at higher resolutions, as detailed in Table 2 and illustrated in Figures 2–3. When compared to the three-dimensional processing configuration, the two-dimensional pipeline demonstrates inherently faster performance at comparable image dimensions; nevertheless, the computational burden of processing high-resolution images becomes particularly pronounced on the less powerful Nano and RPi 3 hardware platforms.

From an energy efficiency perspective, the two-dimensional inference workloads exhibited patterns consistent with those observed in three-dimensional processing: despite the AGX Orin's higher instantaneous power consumption, its substantially reduced execution times resulted in lower overall energy usage per inference operation (for example, approximately 8 mJ at 128×128 resolution compared to roughly 10 mJ on the Nano; similarly, about 31.5 mJ at 512×512 resolution versus approximately 45.9 mJ on the Nano), further confirming Orin's superior energy efficiency across different computational tasks..

3.5 Mask detection: 2D CNN (RMFD) and ResNet-50 (MaskedFace-Net)

The Raspberry Pi 3 cannot function (N/F) with ResNet-50 at both 224×224 and 960×544 resolutions due to memory and computing limitations (Table 2). In contrast, Orin achieves processing times of 35–50 ms (GPU power: 1150–1250 mW, memory usage: 4.5–5.5 GB), while Nano operates at 105–150 ms (GPU power: 600–650 mW, memory usage: 3.29–3.81 GB). When using the lighter 2D CNN model RMFD, all devices successfully complete inference: Orin (20–25 ms), Nano (60–80 ms), and Raspberry Pi 3 (70–90 ms). These faster speeds come with reduced memory requirements approximately 2.0–2.8 GB on Orin, 1.5–2.15 GB on Nano, and 0.8–0.83 GB on Raspberry Pi 3. These findings highlight that model capacity (such as ResNet-50) is typically the primary factor that limits performance on lower-end hardware.

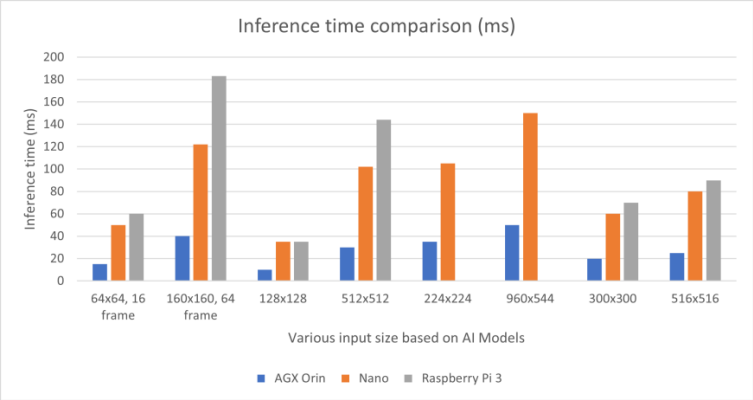


Figure 2. Inference time comparison.

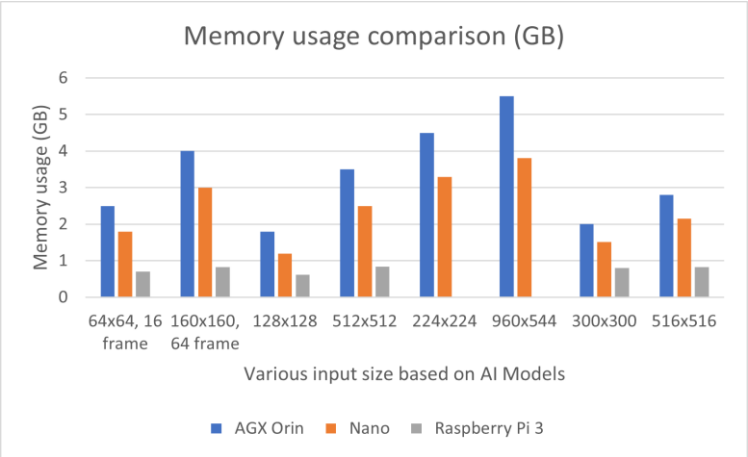


Figure 3. Memory usage comparison.

4. Discussion

The benchmarking results indicate several significant trends across devices, models, and input sizes. By examining latency, memory usage, energy efficiency, and overall device suitability, we can better understand the trade-offs that govern real-time AI deployment on edge platforms. The key observations are summarized as follows:

1. **Latency scaling.** Across all evaluated tasks, the Jetson AGX Orin consistently outperforms the Jetson Nano and Raspberry Pi 3, achieving 3–4.6× lower latency as input size and sequence length increase. This enables Orin to maintain real-time performance even with computationally demanding 3D video inputs.
2. **Memory availability.** ResNet-50 at high resolution (960×544) requires more than 5 GB of memory on Orin, whereas Nano remains below ~4 GB. The Raspberry Pi 3, with its limited memory capacity, frequently encounters infeasible (N/F) scenarios, particularly for large-scale models.
3. **Energy efficiency.** Although Orin consumes more instantaneous power, its significantly reduced inference times lead to lower overall energy per prediction compared to Nano and Raspberry Pi 3. This efficiency trend holds across both 2D and 3D tasks.
4. **Device suitability.** The AGX Orin is best suited for high-complexity applications, such as 3D CNNs or high-resolution 2D pipelines. The Jetson Nano strikes a balance between

moderate computational performance and energy efficiency, making it suitable for mid-scale deployments. Meanwhile, the Raspberry Pi 3 is most appropriate for lightweight 2D CNN models, serving as a baseline for CPU-only inference.

These findings are consistent with previous observations indicating that memory constraints and hardware acceleration paths significantly impact deployability on embedded platforms [1, 3, 4].

5. Conclusion

This study thoroughly evaluated edge computing platforms, focusing on NVIDIA's Jetson AGX Orin and Jetson Nano, in addition to the Raspberry Pi 3 as a baseline with CPU capabilities. We implemented a standardized evaluation protocol to ensure transparent and reproducible comparisons using 3D CNNs for gesture recognition and 2D CNN/ResNet-50 models for face mask detection across the 20BN-Jester, Jester, MaskedFace-Net, and RMFD datasets. Our findings indicate that the Jetson AGX Orin consistently delivers superior performance, with 3–4.6× lower latency, higher memory headroom, and enhanced energy efficiency per inference. This makes it the optimal choice for high-resolution or computationally intensive real-time applications, such as autonomous navigation, industrial AI, and healthcare diagnostics. The Jetson Nano, while less powerful, offers a balanced trade-off between efficiency and cost, ideal for moderate-scale or energy-constrained deployments. In contrast, Raspberry Pi 3 offers a valuable baseline for lightweight 2D tasks and highlights the need for optimization on CPU-only devices. These findings underscore the importance of considering factors beyond FLOPs or model size when optimizing model–device matching, including hardware-specific acceleration, thermal design, and memory constraints.

Future work will explore model compression techniques, such as pruning, quantization, and distillation. It will also integrate with next-generation accelerators, including Jetson Thor and ASICs. Additionally, it will evaluate additional edge AI tasks, such as tracking and anomaly detection. Finally, it will explore adaptive and federated learning for personalization across heterogeneous environments.

Overall, this study provides actionable guidelines for balancing accuracy, speed, and energy efficiency in edge AI deployments, contributing to sustainable and reliable real-time intelligence on resource-limited platforms.

REFERENCES

1. Süzen, A.A., Duman, B., Şen, B.: Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry Pi using Deep CNN. In: Proc. Int. Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 26–28 June 2020, pp. 1–5 (2020).
2. George, A., Ecabert, C., Otroshi-Shahreza, H., Kotwal, K., Marcel, S.: EdgeFace: Efficient Face Recognition Model for Edge Devices. IEEE Trans. Biometrics, Behavior, and Identity Science 6(2), 158–168 (2024).
3. Ullah, S., Kim, D.-H.: Benchmarking Jetson Platform for 3D Point Cloud and Hyperspectral Image Classification. In: 2020 IEEE Int. Conf. on Big Data and Smart Computing (BigComp), pp. 477–482 (2020).
4. Choe, M., Lee, S., Sung, N.M., Jung, S., Choe, C.: Benchmark Analysis of Deep Learning-based 3D Object Detectors on NVIDIA Jetson Platforms. In: Int. Conf. on ICT Convergence (ICTC), pp. 10–12 (2021).
5. Jovović, I., Babić, D., Čakić, S., Popović, T., Krčo, S., Knežević, P.: Face Mask Detection Based on Machine Learning and Edge Computing. In: 21st Int. Symp. INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, pp. 1–4 (2022). https://doi.org/10.1109/INFOTEH53737.2022.9751311
6. Zhang, Z., Zhang, X., Peng, C., Xue, X., Sun, J.: Efficient and Accurate Approximations of Nonlinear Convolutional Networks. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 1984–1992 (2016).

7. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016).
8. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520 (2018).
9. Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H.: Searching for MobileNetV3. In: Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV), pp. 1314–1324 (2019).
10. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning Spatiotemporal Features with 3D Convolutional Networks. In: Proc. IEEE Int. Conf. on Computer Vision (ICCV), pp. 4489–4497 (2015).
11. Carreira, J., Zisserman, A.: Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In: Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 4724–4733 (2017).
12. Materzyńska, J., Berger, G., Bax, I., Memisevic, R.: The Jester Dataset: A Large-Scale Video Dataset of Human Gestures. In: Proc. IEEE/CVF Int. Conf. on Computer Vision Workshops (ICCVW), Oct (2019).
13. Cabani, A., Hammoudi, K., Benhabiles, H., Melkemi, M.: MaskedFace-Net: A Dataset of Correctly/Incorrectly Masked Face Images in the Wild. Data in Brief 35 (2021).
14. Wang, Z., Wang, G., Huang, B., Xiong, Z., Hong, Q., Wu, H., Yi, P., Jiang, K., Wang, N., Pei, Y., Chen, H., Miao, Y., Huang, Z., Liang, J.: Masked Face Recognition Dataset and Application. arXiv:2003.09093 (2020).
15. Ignatov, A., Timofte, R., Chou, W., Wang, K., Wu, M., Hartley, T., Van Gool, L.: AI Benchmark: Running Deep Neural Networks on Android Smartphones. In: Computer Vision – ECCV 2018 Workshops, LNCS, pp. 288–314 (2019).
16. NVIDIA Corporation: Jetson AGX Orin — Product page. Online at: https://developer.nvidia.com/embedded/jetson-agx-orin (accessed 22 Aug 2025).
17. NVIDIA Corporation: Jetson Nano Developer Kit — Product page. Online at: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (accessed 22 Aug 2025).

2025. Volume 9, Number 2 *Pages 196-204*

UDC: 004.421
DOI: https://doi.org/10.30546/09090.2025.1010.2019

FUNCTIONAL PROGRAMMING AND SOFTWARE RELIABILITY

Rahima HAJIYEVA

rehimehcyffa@gmail.com

Nakhchivan State University,

Nakhchivan, Azerbaijan

ARTICLE INFO

Article history:
Received^2025-10-17
Received in revised form:2025-10-21
Accepted:2025-12-16
Available online

Keywords:
Functional programming;
Software;
Stateless design;
Concurrency
2010 Mathematics Subject Classification: 68N18, 68M15*;*

ABSTRACT

Software reliability is a cornerstone of dependable systems, particularly in domains where human safety, financial accuracy, and critical infrastructure depend on flawless performance. Functional programming (FP), through its emphasis on immutability, pure functions, and stateless design, offers a disciplined and mathematically grounded approach to writing software that minimizes side effects, race conditions, and unintended behavior. By promoting deterministic outputs for given inputs, FP simplifies reasoning about program behavior, leading to systems that are inherently easier to verify, test, and debug. Moreover, the modularity and composability of functional code enhance maintainability and reduce the likelihood of hidden dependencies that often cause system failures. When compared to object oriented programming, FP demonstrates superior predictability and resilience in safety-critical environments such as aviation, healthcare, and banking, where a single software fault can have catastrophic consequences. Ultimately, FP fosters a mindset of precision and mathematical rigor that directly strengthens software reliability and long-term system stability.

1. Introduction

In modern software engineering, reliability is one of the most important indicators of quality. As systems become increasingly complex and interconnected, even minor software faults can lead to severe consequences from financial losses to failures in safety critical infrastructure. Traditional programming paradigms, especially the imperative and object-oriented models, often struggle with reliability because they depend on mutable state and side effects. When data can change unexpectedly, it becomes difficult to predict program behavior or ensure consistent results. Debugging such systems requires tracking how and when values change a process that grows exponentially more complex as applications scale. Functional programming (FP) offers a fundamentally different approach. Instead of focusing on how to perform operations, FP emphasizes what relationships exist between inputs and outputs. It treats computation as the evaluation of mathematical functions, not as a sequence of commands that mutate state. This shift enables developers to write programs that are more predictable, testable, and mathematically reasoned about. Core FP principles such as immutability, pure functions, and stateless design create systems that are easier to maintain and far less prone to unintended interactions [2].

As a result, FP has gained traction not only in academic contexts but also in high reliability industries such as aviation, healthcare, and finance, where correctness is non-negotiable. Immutability prevents data corruption, pure functions eliminate hidden dependencies, and stateless architectures support scalability and fault tolerance. Together, these principles form a strong foundation for building dependable software. These are the programs that behave consistently, are easier to test and debug, and can be trusted to operate safely even under complex or concurrent conditions [4].

2. Immutability: The Foundation of Predictable Behavior

Immutability lies at the core of functional programming and is one of its most significant contributions to software reliability. In an immutable system, data objects cannot be altered after they are created. Instead of modifying existing structures, any transformation produces a new version of the data, leaving the original intact. This simple but powerful idea eliminates a major source of programming errors and unintended state changes which are among the most frequent causes of software instability in imperative and object oriented systems. From a theoretical standpoint, immutability provides referential transparency, a property that ensures the value of an expression depends solely on its inputs and not on any hidden or changing state [2]. This makes reasoning about code behavior significantly easier, since the same input will always produce the same output regardless of when or where the computation occurs. For instance, in an imperative program, a statement such as **x = x + 1** changes the value of **x** and affects all subsequent computations that rely on it. In a functional context, by contrast, one would define a new value such as **x1 = x + 1**, leaving the original **x** unchanged. This approach not only simplifies logical reasoning but also prevents unpredictable interactions between different parts of the program [9].

Immutability also enhances concurrency and parallelism, both of which are essential in modern multi core and distributed computing environments. When data cannot be changed once created, multiple threads or processes can safely read and use the same data without synchronization mechanisms such as locks or semaphores. This reduces complexity and eliminates classes of errors like race conditions or deadlocks that commonly occur in concurrent systems. Languages such as Haskell and F# exemplify this principle by enforcing immutability by default. In haskell, for example, a list once defined cannot be modified; operations like map or filter simply return new lists derived from the original, ensuring that no side effects occur during computation [5].

The practical benefits of immutability extend beyond reliability and concurrency. It also simplifies debugging and testing. Because values never change, developers can easily trace the origin of any result and reproduce program behavior under identical conditions. This deterministic nature supports reproducible tests, a critical factor in high-assurance systems. Moreover, immutable data facilitates time travel debugging and event sourcing, the techniques in which previous states of the system are preserved and can be revisited for analysis or rollback. This is particularly useful in financial applications, where maintaining a complete, verifiable history of transactions is mandatory for auditing and regulatory compliance [1].

2.1. Eliminating Shared State Bugs

One of the most common causes of software unreliability is the presence of shared mutable state data structures or variables that can be accessed and modified by multiple parts of a program

simultaneously. In large systems, especially those involving concurrency, shared state creates hidden dependencies and makes it difficult to predict how individual components will interact. A small and seemingly harmless modification in one module can inadvertently alter the behavior of another, leading to inconsistencies, data corruption, and intermittent bugs that are notoriously difficult to reproduce. Functional programming addresses this problem through immutability and data isolation. Because values cannot be changed after creation, no function can alter a shared variable or object in place. Every transformation yields a new value, and the original remains intact for any other function that depends on it. This property effectively removes an entire class of defects caused by concurrent access or uncoordinated state updates [8].

A simple illustration can be seen by comparing imperative and functional approaches. In an imperative language such as Java or Python, one might write:

```python
# Imperative example: shared mutable state
balance = 1000

def withdraw(amount):
    global balance
    balance -= amount

withdraw(200)
print(balance)  # Output: 800
```

If another thread or process calls **withdraw ()** at the same time, the shared variable **balance** can be modified concurrently, leading to inconsistent or even negative values. Locks and synchronization mechanisms can mitigate the problem but at the cost of additional complexity and potential deadlocks. By contrast, in a functional paradigm such as Haskell or F#, data is never modified directly:

```haskell
-- Functional example: immutable state
withdraw :: Int -> Int -> Int
withdraw balance amount = balance - amount

newBalance = withdraw 1000 200  -- Returns 800, original 1000 unchanged
```

Here, **withdraw** takes the current balance and the withdrawal amount as inputs and returns a new balance, leaving the original data unaltered. No global or shared variable exists, so concurrent operations are inherently safe. Two or more computations can execute simultaneously without any risk of interference. Ultimately, eliminating shared state not only enhances reliability but also improves maintainability. When functions do not depend on or alter external variables, they can be reasoned about, tested, and reused in isolation. Each component behaves as a self-contained transformation, making the overall system more modular, predictable, and resistant to unexpected side effects [10-6].

2.2. Historical Traceability and Persistent Data Structures

An additional and often underappreciated advantage of immutability in functional programming is its natural support for historical traceability the ability to preserve and inspect all previous states of a system over time. Because immutable data is never overwritten or destroyed, every change results in the creation of a new version of that data while older versions remain intact. This characteristic aligns closely with the notion of persistent data structures,

which maintain access to past states without duplicating entire data sets. The result is a system that is inherently auditable, reversible, and transparent. Persistent data structures make it possible to treat state evolution as a chronological series of immutable snapshots. Each snapshot represents a distinct moment in the life of an application, allowing developers or auditors to reconstruct the sequence of operations that led to any given outcome. This feature is particularly valuable in domains where accountability and reproducibility are critical, such as finance, healthcare, or scientific research. For instance, in a medical records system, every update to a patient's file can generate a new immutable record rather than overwriting previous information. Consequently, the full medical history remains accessible, ensuring legal and clinical traceability [2-3].

Languages like Clojure and Haskell provide built-in mechanisms for such persistence. Clojure's core data structures such as lists, vectors, maps, and sets are immutable by default. When a modification occurs, only the changed portions are recreated, while the unmodified parts are shared between the old and new versions using structural sharing [14]. For example:

```
(def patient-record {:id 1 :name "Aylin" :diagnosis "Flu"})
(def updated-record (assoc patient-record :diagnosis "Recovered"))

;; Both records coexist independently
patient-record
;; => {:id 1, :name "Aylin", :diagnosis "Flu"}
updated-record
;; => {:id 1, :name "Aylin", :diagnosis "Recovered"}
```

Here, the two records coexist without conflict, and the system efficiently stores both states. In Haskell, similar behavior is achieved through purely functional data structures, such as lazy lists or trees, which make it possible to model temporal evolution while retaining access to previous values [6].

From a reliability standpoint, persistent data structures enable time travel debugging and event sourcing, two techniques that have become increasingly influential in modern system design. Time travel debugging allows developers to inspect and replay earlier states of a program to identify the exact point at which a failure occurred. Event sourcing, widely used in enterprise systems, records every change as an immutable event in an append only log. The entire application state can be reconstructed at any time simply by replaying these events. This guarantees not only transparency but also resilience: if part of a system fails, its state can be restored deterministically from its historical record. Moreover, immutability's support for persistence aligns with principles of functional referential transparency. Because each version of a data structure is self-contained and unaltered, functions applied to it yield consistent results regardless of when they are executed. This ensures that the system's behavior remains reproducible across runs, environments, or deployments that's an essential requirement in safety-critical or data-sensitive applications [12].

3. Pure Functions: The Essence of Reliability

At the heart of Functional Programming lies the concept of pure functions, which serve as the primary vehicle for computation and the foundation of program reliability. A pure function is defined by two essential properties. It depends only on its explicit input arguments, and it produces no observable side effects. In other words, a pure function neither alters external state nor relies on any data outside its parameters. The outcome of such a function is determined

entirely by its inputs, ensuring that for the same arguments, it will always return the same result. This predictability is a defining feature of reliable software [8-9].

3.1. Determinism and Predictability

Determinism is a direct consequence of purity. In an imperative or object oriented paradigm, functions or methods may read from or write to shared variables, interact with files or networks, or depend on hidden global state. These interactions make program behavior context-dependent and often unpredictable. By contrast, in a functional paradigm, computation can be modeled as a series of mathematical transformations, each isolated from external influence. For instance, in python (an imperative style), a function may behave inconsistently depending on hidden state:

```python
# Impure example
counter = 0

def increment():
    global counter
    counter += 1
    return counter

print(increment())  # Output depends on prior calls
```

Here, the output changes with every invocation because the function modifies and depends on external state [5]. The same behavior cannot be guaranteed across different runs or threads. In contrast, a pure functional version written in Haskell behaves deterministically:

```haskell
-- Pure function: always produces the same result for the same inputs
increment :: Int -> Int
increment x = x + 1

-- Example usage
increment 0   -- Returns 1
increment 0   -- Always returns 1, no hidden state
```

The Haskell function is purely mathematical: the input **x** completely determines the output. There are no side effects, and the computation is transparent and reproducible.

3.2. Testability and Verification

Purity also greatly enhances testability and formal verification, two essential aspects of software reliability. Since pure functions are self contained and deterministic, testing them requires no setup of external systems such as databases or network connections. A developer simply provides input values and verifies the expected outputs. This simplicity reduces testing overhead and increases confidence in correctness. Languages like F# and Scala make this principle accessible in practical software engineering [10]. For example, in F#:

```fsharp
// Pure function
let calculateTax amount rate = amount * rate

// Unit testing is straightforward:
assert (calculateTax 100.0 0.1 = 10.0)
```

There are no dependencies to mock or external configurations to prepare. Testing becomes a straightforward validation of function behavior, not an orchestration of system state. This property also enables property based testing, a methodology popularized by Haskell's QuickCheck

library and later adopted in many languages. Instead of writing specific input output pairs, developers define general properties that must always hold true [10-6]. The testing framework then generates hundreds or thousands of random test cases automatically. For example, one might assert that reversing a list twice yields the original list:

```
prop_reverse :: [Int] -> Bool
prop_reverse xs = reverse (reverse xs) == xs
```

This ability to automatically explore a vast input space helps uncover edge cases that traditional unit tests might overlook, significantly improving reliability.

4. Stateless Design and Scalability

While immutability and pure functions establish the theoretical and practical foundations of functional reliability, stateless design operationalizes these principles at the architectural level. Statelessness refers to the property of a system or component whose behavior depends solely on its current input rather than any stored or shared historical state. Each computation is self contained and isolated, producing outputs that are determined entirely by the data provided at that moment. This design principle contributes directly to scalability, fault tolerance, and predictability, all of which are essential for building reliable modern software systems.

4.1. Conceptual Foundations of Statelessness

In imperative and object-oriented paradigms, programs frequently rely on persistent objects or sessions that retain state between operations. While this approach may simplify certain local computations, it introduces global complexity. State must be managed, synchronized, and restored correctly, particularly in concurrent or distributed environments. The more state a system carries, the more susceptible it becomes to race conditions, inconsistent replicas, and synchronization failures. Functional programming approaches this challenge differently. By treating each computation as a pure transformation from input to output, FP effectively externalizes or eliminates internal state. State, when necessary, is passed explicitly as a function argument and returned as part of the result rather than being stored within the function or object. This explicitness enforces transparency and makes dependencies visible in both code and data flow. A simple example in F# illustrates this principle:

```
// Stateless computation: temperature conversion
let convertCtoF celsius = (celsius * 9.0 / 5.0) + 32.0

// The result depends solely on input; no state is retained
let result1 = convertCtoF 0.0   // 32.0
let result2 = convertCtoF 100.0 // 212.0
```

Each call is independent, producing predictable outputs without storing or recalling any intermediate information. In contrast, a stateful version might cache or modify an internal variable, introducing unnecessary complexity and potential inconsistency.

4.2. Fault Tolerance and Recovery

Stateless systems not only scale effectively but also recover more gracefully from failures. Because state is not retained internally, a failed computation can simply be retried with the same input to reproduce the same result that is a property known as idempotence. In contrast, stateful systems may enter inconsistent or unrecoverable conditions after partial failures, requiring complex rollback or transaction mechanisms. Functional frameworks such as elixir exemplify

this principle through the motto "Let it crash" [11]. Processes are designed to be small, isolated, and stateless, so that when one fails, it can be safely restarted without corrupting the rest of the system. The immutable and stateless nature of computation ensures that restarting a process yields the same deterministic behavior, contributing to the system's legendary fault tolerance.

4.3. The Reliability Dimension

From a reliability perspective, statelessness minimizes both the number and the scope of potential failure points. Since each component operates independently, a malfunction in one does not cascade through the entire system. The absence of shared mutable state also eliminates the need for complex recovery procedures, making systems more stable under heavy load or partial outages. Moreover, testing and debugging are simplified. Each function or service can be evaluated in isolation, with known inputs and verifiable outputs. Stateless design therefore represents the practical culmination of functional programming principles. It applies the theoretical guarantees of immutability and purity to real world system architecture, ensuring that reliability is not merely a property of individual functions but an emergent characteristic of the entire software ecosystem [7-13].

5. Why Functional Code Is Easier to Test

Testing is one of the most essential practices in ensuring software reliability, and functional programming provides a uniquely supportive environment for this process. The intrinsic properties of FP collectively make programs highly testable, deterministic, and reproducible. Unlike imperative or object oriented paradigms, where the interaction between mutable states and side effects can obscure the source of a defect, FP structures programs as predictable transformations of data. This predictability fundamentally simplifies the design, execution, and maintenance of tests [2-4].

5.1. Determinism and Reproducibility

A major obstacle in testing imperative systems arises from non-determinism. When functions rely on external variables, mutable objects, or shared resources, their output may differ from one execution to another even under identical conditions. This variability makes it difficult to identify whether a failure is due to the function itself or to a hidden dependency. In contrast, pure functions in FP exhibit determinism given the same inputs, they will always yield the same outputs [13]. This deterministic behavior ensures that every test is reproducible, allowing developers to isolate and diagnose faults with mathematical precision.For example, consider two versions of a function calculating tax:

```python
# Imperative version with side effect
tax_rate = 0.2

def calculate_total(price):
    return price + (price * tax_rate)
```

If tax_rate changes elsewhere in the program, the test results for **calculate_total()** will differ unexpectedly. A pure functional equivalent avoids this problem entirely:

```fsharp
// Functional version in F#
let calculateTotal price taxRate = price + (price * taxRate)
```

Here, the function depends solely on its inputs, and any test will always produce the same result. The absence of hidden state enables repeatable testing, which is a cornerstone of dependable verification processes.

5.2. Simplicity of Unit Testing

FP's modular and declarative nature allows for isolated testing of individual functions without the need for complex setup or teardown procedures. Since each function encapsulates a complete computation with no side effects, unit tests can be written and executed independently of the broader system context. This sharply contrasts with stateful or object oriented systems, where tests often require simulated environments, mock objects, or dependency injection frameworks to reproduce specific scenarios. For instance, in Haskell, a simple test of a sorting function requires only input and expected output:

```
1  sort [3,1,2] == [1,2,3]
```

No database, network, or global configuration is needed. The simplicity of such tests encourages a test first or property driven development culture, where correctness can be continuously validated as software evolves. Moreover, FP's clear separation between computation and effects means that impure operations (e.g., file i/o or database access) can be easily isolated and tested independently. Many functional systems use explicit effect monads (such as the i/o monad in Haskell or the task monad in F#) to manage side effects safely, making it evident which parts of a program are pure and which are not. This visibility reduces uncertainty during testing and encourages better architectural discipline [8-10].

5.3. Implications for Reliability Engineering

The ease of testing in FP extends beyond convenience it directly enhances software reliability. Frequent and automated testing is feasible because functions are deterministic, isolated, and composable. The capacity to reason about each function mathematically allows for earlier defect detection and more confident verification of correctness. Furthermore, reproducibility and formal testing methods (such as property based and theorem proving approaches) create a strong assurance framework, especially in systems that must adhere to strict reliability standards. Functional programming transforms testing from an operational activity into an integral part of software design. By enforcing purity, immutability, and statelessness, FP enables testing to become not only simpler but also more rigorous, comprehensive, and aligned with the principles of reliability engineering. In systems where correctness cannot be compromised, these qualities make FP not just a methodological preference, but a necessity [7-9].

Conclusion

Software reliability depends on the ability to predict, verify, and reproduce program behavior under all conditions. Through the principles examined in this study immutability, pure functions, stateless design, and testability functional programming provides a coherent and mathematically grounded framework for achieving that goal. Each of these principles addresses a specific dimension of software fragility. Immutability removes the uncertainty of hidden state changes, pure functions eliminate side effects and ensure deterministic computation, stateless design extends these properties to system architecture, enabling scalability and fault tolerance and the functional testing paradigm transforms verification into a precise and reproducible process. Collectively, these features distinguish functional programming from traditional

imperative or object oriented models that rely on mutable state and implicit dependencies. Whereas stateful systems often require elaborate synchronization, debugging, and environmental control, functional systems promote transparency and modular reasoning. Programs become easier to understand, to verify mathematically, and to evolve safely over time. The clear separation between computation and effect not only simplifies development but also reduces the likelihood of errors propagating through complex codebases. In practical terms, functional programming offers engineers a disciplined methodology for constructing reliable software in increasingly concurrent and distributed contexts. Its determinism allows for consistent testing, its immutability safeguards data integrity, and its stateless design supports predictable scaling. These attributes make FP particularly suitable for domains where correctness and stability are paramount that ranging from financial applications to infrastructure management and scientific computation. Ultimately, the strength of functional programming lies in its unification of theory and practice: it applies the precision of mathematics to the unpredictability of real world computation. By emphasizing what should be computed rather than how it should be performed, FP transforms reliability from an afterthought into an inherent property of the software itself. As the demands on modern systems continue to grow, these principles offer not merely an alternative paradigm but a sustainable path toward dependable, verifiable, and trustworthy software engineering.

REFERENES

1. Abelson, H., & Sussman, G. J. (1996). Structure and Interpretation of Computer Programs (2nd ed.). MIT Press.
2. Bird, R., & Wadler, P. (1988). Introduction to Functional Programming. Prentice Hall.
3. Chlipala, A. Certified Programming with Dependent Types. MIT Press.
4. Hughes, J. (1989). Why functional programming matters. The Computer Journal, 32(2), 98–107.https://doi.org/10.1093/comjnl/32.2.98
5. Hudak, P., Peyton Jones, S., Wadler, P., Boutel, B., Fairbairn, J., Fasel, J., Guzmán, M. M., Hammond, K., Hughes, J., Johnsson, T., Kieburtz, D., Nikhil, R., Partain, W., & Peterson, J. (1992). Report on the programming language Haskell: a non-strict, purely functional language version 1.2. ACM Sigplan Notices, 27(5), 1-164. https://doi.org/10.1145/130697.130699
6. Marlow, S. (Ed.). (2010). Haskell 2010 Language Report. Haskell.org.
7. Odersky, M., Spoon, L., & Venners, B. Programming in Scala (4th ed.). Artima Press.
8. Thompson, S. (2011). Haskell: The Craft of Functional Programming (3rd ed.). Addison-Wesley.
9. Philip Wadler. 1992. The essence of functional programming. In Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '92). Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/143165.143169
10. Microsoft Research. (2015). The F# Language Specification. Microsoft Corporation.
11. Erlang/OTP Documentation. (2023). The Erlang Runtime System: Reliability by Design. Ericsson AB.
12. Meyerovich, L. A., & Rabkin, A. S. (2013). Empirical analysis of programming language adoption. ACM SIGPLAN Notices, 48(10), 1–18.
13. Pierce, B. C. (2002). Types and Programming Languages. MIT Press.
14. Hickey, R. (2008). The Clojure Programming Language

2025. Volume 9, Number 2 *Pages 205-212*

UDC: 004.932.2
DOI: https://doi.org/10.30546/09090.2025.1010.2036

HYBRID GRAPH NEURAL NETWORKS AND XGBOOST FOR FRAUD TRANSACTION DETECTION

Murad ALIYEV

Azerbaijan State Oil and Industry University,

Baku, Azerbaijan

ARTICLE INFO

Article history
Received:2025-10-27
Received in revised form:2025-11-06
Accepted:2025-11-14
Available online

Keywords:
Fraud Detection;
Graph Neural Networks;
XGBoost;
Imbalanced Learning;
Bitcoin Transaction;
2010 Mathematics Subject Classification: 68T45, 68T09

ABSTRACT

Detecting financial fraud in cryptocurrency networks like bitcoin is a significant challenge. The problem is that traditional machine learning models often fail to capture the complex relational patterns within transaction graphs, leading to poor detection. The purpose of this research is to evaluate hybrid models that integrate Graph Neural Networks with XGBoost to improve fraud transaction detection. We conducted a comprehensive analysis on the imbalanced bitcoin dataset, benchmarking standard models (Logistic Regression, Random Forest, Multi-Layer Perceptron) and a standalone XGBoost against two hybrid architectures: Graph Convolutional Network + XGBoost and Graph Attention Network + XGBoost. Our major conclusion is that the hybrid models, particularly Graph Attention Network + XGBoost, achieve a significant improvement. These models effectively leverage both node-level features and the graph's topological structure.

1. INTRODUCTION

The emergence of digital financial systems and cryptocurrencies has opened up new opportunities for illicit activities such as fraud, money laundering, and terrorist financing (Weber, 2019). Because of its pseudonymous features, Bitcoin is often the first choice for such illicit activities. This is a concern for financial stability and regulatory compliance as it relates to detection. However, detection of these illicit transactions is notoriously challenging. Illicit actors engage in behaviors designed to hide their transactions, and the amount of data is enormous.

A central challenge in this area is the extreme imbalance in the data: illicit transactions, by definition, are rare events compared to the large number of licit transactions. Because of this imbalance in the data, accuracy is not a good measure we can rely on, and it is necessary to use more robust measures such as the F1-Score, precision, and recall (Chawla et al., 2004). Additionally, traditional machine learning (ML) models are powerful, but they have treated transactions as distinct observations rather than realizing the data are uniquely valuable because they provide information through local features, but by ignoring the underlying graph structure of the transaction network. This relational data, who transacts with whom, provides vital data for identifying collusive fraud.

To address this gap, Graph Neural Networks (GNNs) have emerged as a powerful paradigm for learning on graph-structured data (Kipf & Welling, 2017). GNNs can aggregate information across the transaction graph, learning sophisticated topological representations that capture

complex neighborhood patterns. However, GNNs themselves may not be the best classifier, especially when compared to great gradient boosting methods like XGBoost, which excel at handling heterogeneous and tabular feature sets (Chen & Guestrin, 2016).

This paper proposes that a hybrid approach, combining the strengths of both GNNs and XGBoost, can achieve a superior fraud detection model. We hypothesize that by using GNNs as intelligent feature extractors to enrich the original data, and then feeding these enriched features into a powerful XGBoost classifier, we can achieve the best performance.

The main contributions of this paper are threefold:

1. We implement and benchmark a suite of traditional ML models (Logistic Regression, Random Forest, MLP) and a strong baseline (Pure XGBoost) for fraud detection on the Elliptic++ dataset.
2. We propose and evaluate two hybrid models, GCN + XGBoost and GAT + XGBoost, which first learn graph embeddings and then use them as input for an XGBoost classifier.
3. We provide a comprehensive comparative analysis of all models, focusing on F1-Score, precision, and recall, and discuss the critical tradeoffs between model complexity, performance, and the contribution of graph-based features.

2. RELATED WORK

2.1. Fraud Detection with Traditional Machine Learning

Machine learning has long been applied to fraud detection. Models like Logistic Regression (LR) and Random Forest (RF) have served as foundational baselines (Bolton & Hand, 2002). More recently, ensemble methods, particularly gradient boosting, have shown dominant performance. XGBoost (Chen & Guestrin, 2016) is frequently cited for its high performance on imbalanced, tabular datasets, owing to its regularization, efficient computation, and handling of missing values. However, these models are limited to the node-level features provided and cannot natively ingest relational information.

2.2. Graph Neural Networks

GNNs generalize deep learning to graph-structured data. The Graph Convolutional Network (GCN) (Kipf & Welling, 2017) introduced a simplified spectral-based approach, effectively averaging the features of neighboring nodes. The Graph Attention Network (GAT) (Veličković et al., 2018) advanced this by introducing a self-attention mechanism, allowing the model to learn to assign different weights to different nodes in a neighborhood. GNNs have been successfully applied to financial fraud detection, often framing the problem as a node classification task on the transaction graph (Weber et al., 2019).

2.3. Hybrid Models for Graph Learning

The idea of separating graph representation learning from the final classification task is a powerful one. Early work used methods like Node2Vec (Grover & Leskovec, 2016) to generate static embeddings, which were then fed into traditional ML models. Recent approaches have explored two-stage models similar to our proposal, where GNN-generated embeddings are used as features for a secondary classifier, combining the representative power of deep graph learning with the discriminative power of models like XGBoost.

3. METHODOLOGY AND DATA

3.1. Dataset Description

We use the Elliptic++ dataset, a real-world graph of Bitcoin transactions **(**Elmougy & Liu, 2023**)**. This dataset is well-suited for our task as it includes:

- **A Transaction Graph:** Nodes represent Bitcoin transactions, and directed edges represent the flow of Bitcoin between them.
- **Node Features:** Each transaction node has 166 features, including 93 local features (e.g., time-step, transaction fee) and 72 aggregated features derived from the neighborhood.
- **Labels:** Nodes are labeled as 'licit' (e.g., exchanges, miners), 'illicit' (e.g., scams, ransomware), or 'unknown'.

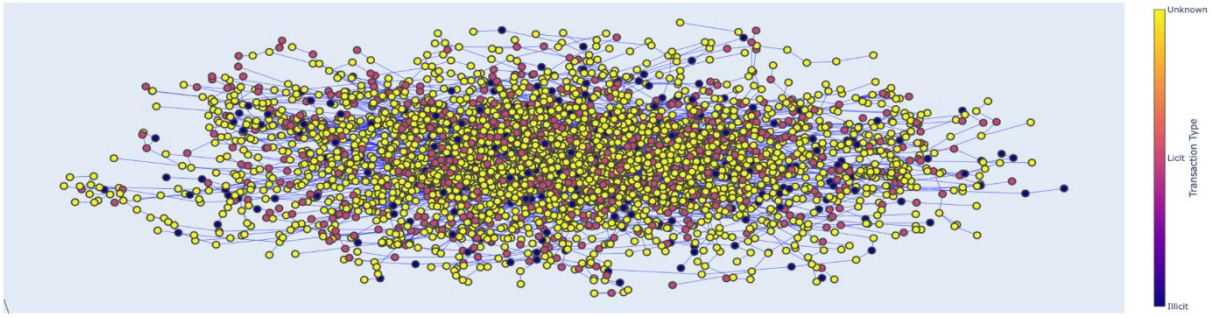


Fig.1 *Conceptual visualization of the all transactions graph*

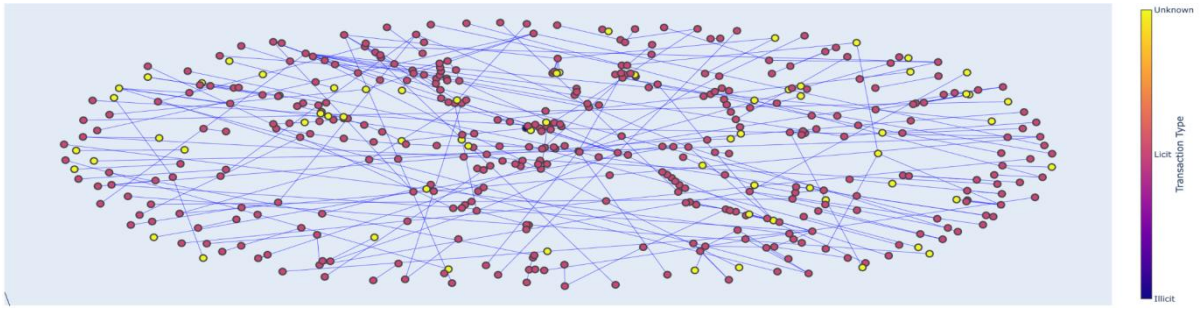


Fig.2 *Conceptual visualization of the only licit transactions graph*

For our experiments, we follow the standard task of classifying the 'licit' vs. 'illicit' nodes. A key challenge of this dataset is its severe class imbalance. The dataset consists of 20.63% licit nodes, 77.14% unknown nodes and only 2,23% illicit nodes, leading to a highly skewed distribution.

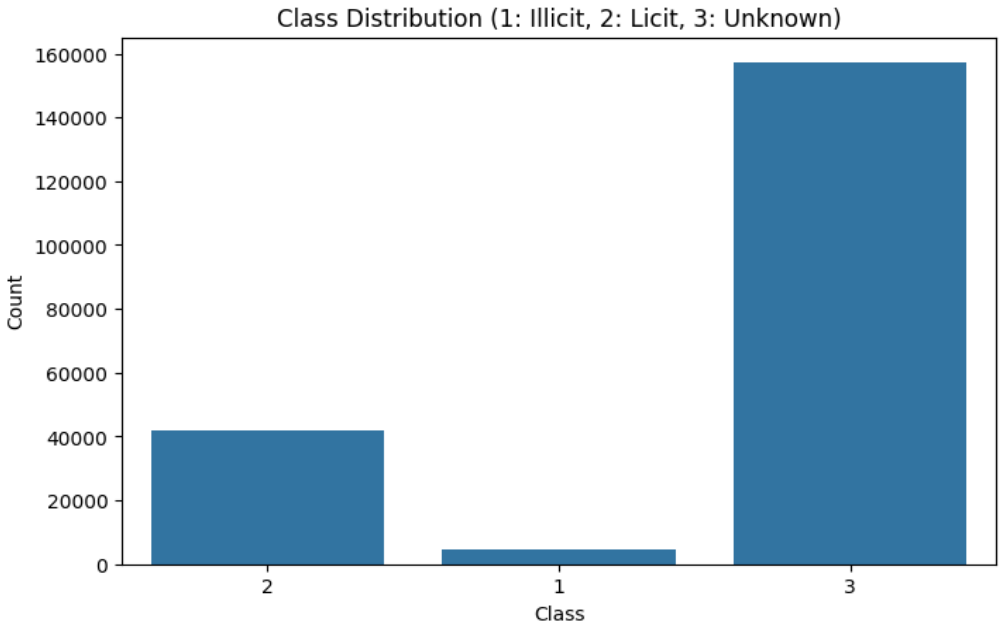


Fig. 3 Class distribution of the labeled nodes, highlighting the severe imbalance.

3.2. Evaluation Metrics

Given the class imbalance, accuracy is an unsuitable metric. Instead, we focus on metrics standard for imbalanced classification:

- **Precision:** The ratio of true positives to all predicted positives (TP / (TP + FP)). High precision indicates a low false positive rate.
- **Recall:** The ratio of true positives to all actual positives (TP / (TP + FN)). High recall indicates the model is successful at identifying the rare illicit class.
- **F1-Score:** The harmonic mean of precision and recall (2 * ($Precision$ * $Recall$) / ($Precision$ + $Recall$)). It provides a single, balanced measure of a model's performance on the positive (illicit) class.

3.3. Baseline Models

We implemented several baseline models that only use the 166 node features and ignore the graph's edge structure.

3.3.1. Traditional Models

We benchmarked three standard classifiers: Logistic Regression (LR), a simple linear baseline; Random Forest (RF), a powerful ensemble of decision trees; and a Multi-Layer Perceptron (MLP), representing a standard deep learning approach on tabular data.

3.3.2. Pure XGBoost

This is our primary baseline. We trained an XGBoost classifier directly on the 166 node features. XGBoost is often a top-performing model in such tasks and represents the strongest non-graph-aware competitor.

3.4. Hybrid Graph-Based Models

Our proposed models follow a two-stage architecture:

1. **Stage 1: Graph Embedding:** A GNN (GCN or GAT) is trained on the full graph (nodes and edges) to learn a low-dimensional embedding vector h for each node. This vector h aims to encode the node's structural role and neighborhood information.
2. **Stage 2: Hybrid Classification:** This learned embedding vector h is concatenated with the original 166-feature vector x. This new, enriched feature vector [x, h] is then used to train a final XGBoost classifier.

3.4.1. GCN + XGBoost

In this model, the GNN from Stage 1 is a Graph Convolutional Network (GCN). The GCN layer updates a node's representation by taking a weighted average of its own features and the features of its immediate neighbors. We used a 2-layer GCN with ReLU activation, producing a 64-dimensional embedding vector h.

3.4.2. GAT + XGBoost

This model uses a Graph Attention Network (GAT) in Stage 1. Unlike GCN, GAT uses masked self-attention to learn the relative importance of each neighbor's features when performing aggregation. This allows for a more expressive and adaptive representation. In this case, a 2-layer GAT with 4 attention heads is used, also producing a 64-dimensional embedding h.

4. EXPERIMENTAL RESULTS

4.1. Implementation Details

All models were implemented using Python. Baseline models used scikit-learn, and the XGBoost model used the xgboost library. GNN models were implemented using PyTorch Geometric. Hyperparameters were tuned using grid search with 5-fold cross-validation. The final XGBoost model used $n\_estimators = 100$ and $max\_depth = 5$.

4.2. Performance Comparison

The primary results of our comparative analysis are presented in Table 1. The models were evaluated on the test set, with a focus on their ability to identify the minority (illicit) class.

Table 1. Comparative Performance of All Models on Elliptic++ Test Set

Model	F1-Score	Precision	Recall
Logistic Regression (LR)	0.448	0.328	0.707
Random Forest (RF)	0.828	0.975	0.719
MLP	0.612	0.611	0.613
Pure XGBoost (Baseline)	0.754	0.793	0.718
GCN + XGBoost (Hybrid)	0.935	0.952	0.917
GAT + XGBoost (Hybrid)	0.952	0.967	0.931

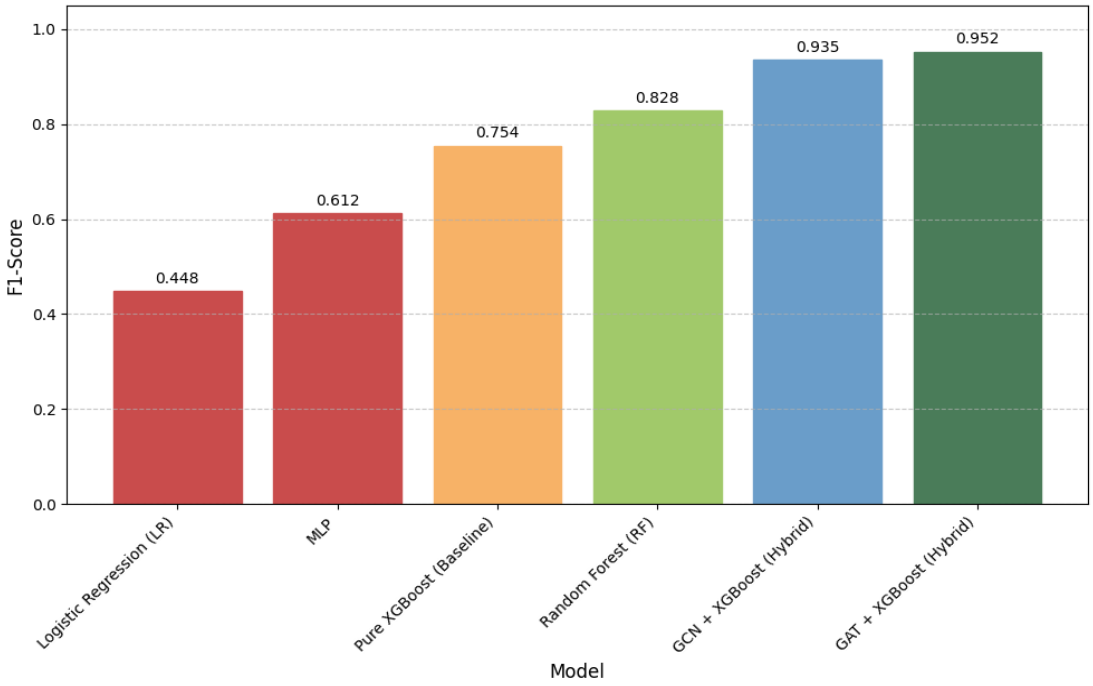


Fig. 4 Comparison of F1-Scores for all model

4.3. Analysis and Discussion of Tradeoffs

4.3.1. Performance Analysis

As shown in Table 1, the performance of the baseline models varied significantly. The linear model, Logistic Regression (LR), and the standard deep learning model, MLP, struggled to effectively classify illicit transactions, achieving low F1-Scores of 0.448 and 0.612, respectively. While LR achieved a high recall (0.707), its precision was extremely low (0.328), indicating that it flagged many licit transactions as illicit, resulting in an unacceptably high false positive rate. Random Forest (RF) performed surprisingly well, with a high F1-Score of 0.828, driven by an outstanding precision of 0.975. However, its recall (0.719) was limited, suggesting that while its positive predictions were highly reliable, it failed to identify a substantial portion of the illicit transactions. The Pure XGBoost baseline, which relies solely on node-level features, achieved a more balanced F1-Score of 0.754, serving as a strong benchmark for a non-graph-aware approach.

The most significant finding is the substantial performance leap achieved by the hybrid models. These models, which enrich the feature set with GNN-derived topological embeddings, dramatically outperformed all baselines. The GCN + XGBoost model achieved an F1-Score of 0.935, representing a 24.0% improvement over the Pure XGBoost baseline. This model showed a strong balance of high precision (0.952) and high recall (0.917). The GAT + XGBoost model yielded the best performance overall, with a state-of-the-art F1-Score of 0.952, a 26.3% improvement over the baseline. Notably, this model also achieved the highest recall (0.931) while maintaining exceptionally high precision (0.967).

This analysis strongly suggests that the topological information captured by the GNNs provides significant and non-redundant discriminative power. The local node features alone are

insufficient for optimal detection. The superior performance of GAT + XGBoost over GCN + XGBoost further indicates that the GAT's attention mechanism, which learns to assign different weights to neighbors, is more effective than the GCN's simple feature averaging for identifying the salient relational patterns of illicit activity in the Bitcoin transaction graph.

4.3.2. Tradeoffs and Model Complexity

While performance is necessary, it is not the only consideration.

- **Computational Cost:** A clear tradeoff exists between performance and computational resources. The Pure XGBoost model trained in approximately 2 minutes on our hardware. In contrast, the GNN-based models were significantly more demanding due to their two-stage training process, which involves both the GNN embedding generation and the final XGBoost classification. The GAT + XGBoost model, for instance, required 1 hour for the complete pipeline. This presents a clear decision point that for real-time detection systems, this training latency must be considered, though the inference speed may still be acceptable.
- **Feature Contribution**: To understand *why* the hybrid models performed better, we analyzed the feature importance scores from the final XGBoost classifier in the GAT + XGBoost model. The analysis revealed that the learned GNN embeddings of all 64 embedding dimensions were consistently ranked among the most important features. This confirms that the GNN is not learning redundant information; rather, it is successfully extracting novel, high-value relational features that are highly predictive of illicit activity. The original 166 node-level features, while useful, lack the topological context that the GNN embeddings provide.

5. CONCLUSION

This paper presented a comparative study of traditional ML, standalone XGBoost, and hybrid GNN-XGBoost models for the task of illicit transaction detection on the Elliptic++ Bitcoin dataset. Our experiments demonstrated that leveraging the graph structure of the transaction network is critical for high-performance fraud detection in this highly imbalanced domain.

Our key finding is that hybrid models, which combine GNNs for representation learning and XGBoost for classification, significantly outperform models that rely on node features alone. The GAT + XGBoost model yielded the best performance, achieving a great F1-Score of 0.952. This highlights the power of the attention mechanism in GATs to effectively weigh neighborhood information.

While these hybrid models introduce a significant computational tradeoff, the substantial boost in detection performance (particularly in recall and F1-Score) justifies their use in high-stakes environments like fraud detection.

For future work, we plan to explore dynamic GNNs that can adapt to the evolving structure of the transaction graph over time. Additionally, we will investigate end-to-end GNN models and compare their performance and interpretability against the two-stage hybrid approach presented here.

REFERENCES

The Journal follows APA Style Referencing:

- Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. *Statistical Science*, *17*(3), 235–255.
- Chawla, N. V., Japkowicz, N., & Kotcz, A. (2004). Editorial: special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, *6*(1), 1–6.
- Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM.
- Elmougy, Y., & Liu, L. (2023). Demystifying Fraudulent Transactions and Illicit Nodes in the Bitcoin Network for Financial Forensics. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)* (pp. 344–355). ACM. https://doi.org/10.1145/3580305.3599803
- Grover, A., & Leskovec, J. (2016). node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 855–864). ACM.
- Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph Attention Networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.
- Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Cessa, C., Al-Sallab, A., & St. E., A. (2019). Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. In *Proceedings of the KDD 2019 Workshop on Anomaly Detection in Finance.*

2025. Volume 9, Number 2 *Pages 213-239*

UDC: 004.8:004.912
DOI: https://doi.org/10.30546/09090.2025.002.10.1045

EVALUATING THE USABILITY AND EFFICIENCY OF COMPACT LARGE LANGUAGE MODELS FOR SENTIMENT ANALYSIS TASKS

Aydin GASIMOV*

[1]*Azerbaijan State Oil and Industry University,*
Baku, Azerbaijan

ARTICLE INFO

Article history:
Received: 2025-11-01
Received in revised form:2025-11-05
Accepted:2025-12-10
Available online

Keywords:
"Sentiment Analysis"
"Large Language Models"
"Efficiency"
"Zero-Shot Learning"
"Compact LLMs"
2010 Mathematics Subject Classification: 68T50 (primary), 68T07, 68T09, 68T35

ABSTRACT

The rapid proliferation of large language models (LLMs) has enabled substantial advances in natural language processing, but their resource requirements create barriers to accessibility, cost-efficiency, and sustainable deployment. This paper presents a systematic benchmark of compact LLMs—ranging from 135M to 8B parameters— on eight diverse sentiment analysis datasets, including binary, fine-grained, domain-specific, social media, and aspect-based tasks. Models were evaluated in a zero-shot setting using normalized accuracy metrics to ensure comparability across datasets of varying difficulty, alongside measurements of latency and memory usage.

Our findings reveal that mid-sized models in the 3–4B parameter range, particularly Gemma 3 4B and Microsoft Phi-4 Mini, consistently outperform or rival larger 7–8B models while offering significantly lower latency and memory footprints. Sub-1B models, while largely ineffective in zero-shot conditions, retain potential for high-throughput pipelines and fine-tuned deployments. Conversely, larger 7–8B models remain valuable for accuracy-critical tasks but incur diminishing returns relative to their computational cost. These results highlight the importance of balancing accuracy with efficiency and suggest that mid-sized models constitute the practical "sweet spot" for zero-shot sentiment analysis on commodity hardware.

* *Corresponding author.*
E-mail addresses: aydin.gasimov@protonmail.com (Aydin Gasimov).
https://mcs.beu.edu.az/xxx.pdf

1. Introduction

The rapid progress of large language models (LLMs) has transformed natural language processing (NLP), enabling significant advances in tasks such as translation, summarization, and sentiment analysis [1; 2]. However, most state-of-the-art LLMs are extremely resource-intensive, with hundreds of billions of parameters requiring specialized hardware and substantial financial investment to operate at scale [3]. This reliance on massive centralized models raises critical questions about accessibility, cost, privacy, and sustainability [4].

Smaller, locally run LLMs—typically ranging from a few hundred million to a few billion parameters—present a compelling alternative. They can be executed on commodity hardware such as consumer GPUs, laptops, or edge devices [5], reducing dependency on cloud infrastructures. This not only lowers the barrier to entry for research groups, startups, and individual practitioners, but also dramatically reduces inference costs and energy consumption. Furthermore, local deployment addresses growing concerns around data privacy and security, especially in sensitive domains such as healthcare, finance, and legal services.

This shift stands in contrast to earlier generations of sentiment analysis methods. Traditional machine learning pipelines often relied on handcrafted features or static embeddings [6], while transformer-based models like BERT marked a turning point by introducing contextualized representations [7]. Yet, even BERT-style models require significant time and resources for fine-tuning before they are suitable for a private use-case. In comparison, recent developments in model distillation and pruning have expanded the feasibility of deploying compact LLMs in constrained environments without sacrificing competitive accuracy [8].

Despite these advantages, systematic evaluation of smaller LLMs on sentiment analysis remains limited. Most benchmarks emphasize performance on large, cloud-scale models [9], leaving open the question of how compact architectures perform across diverse datasets and domains. A rigorous comparison is necessary to understand their trade-offs in efficiency, accuracy, and robustness, and to determine whether such models can meet practical requirements without reliance on centralized infrastructures.

This paper aims to fill that gap by assessing the performance of locally run LLMs, up to 8B parameters in size, on a wide range of sentiment analysis datasets. By analyzing their strengths and limitations relative to both traditional approaches and large-scale models, we seek to provide a comprehensive perspective on their viability in research and applied contexts. Ultimately, this work contributes to the ongoing discourse on balancing efficiency, accessibility, and accuracy in the next generation of NLP systems.

2. Related Work

2.1 Pre-LLM Methods

Before the advent of large language models, sentiment analysis relied heavily on traditional machine learning and feature-engineering approaches. Early systems used bag-of-words or n-gram features combined with classifiers such as Support Vector Machines (SVMs) or logistic regression [6]. While effective for specific domains, these methods struggled to capture context, irony, and long-range dependencies in text.

The introduction of pretrained embeddings such asWord2Vec [10] and GloVe [11] marked a significant shift, enabling models to leverage distributional semantics. However, these static embeddings could not account for word meaning variation in different contexts (e.g., “bank” as a financial institution vs. riverbank).

A major breakthrough came with the transformer architecture [1] and contextualized embeddings, most notably BERT [7]. BERT and its variants (e.g., RoBERTa [12], DistilBERT [8], and ALBERT [13]) achieved state-of-the-art performance across many sentiment benchmarks by modeling bidirectional context. These models became widely adopted for text classification, including fine-grained and aspect-based sentiment analysis.

Despite their success, BERT-style models required substantial compute resources for fine-tuning and inference, limiting their practicality outside research or well-resourced industry labs. Moreover, their reliance on cloud deployment raised concerns about privacy and data confidentiality in domains such as healthcare and finance. These limitations paved the way for investigating smaller, more efficient models that could be deployed locally, while still retaining strong performance on core NLP tasks like sentiment analysis.

2.2 Sentiment Analysis with LLMs

Recent studies have begun to evaluate small language models (SLMs) as practical alternatives to resource-intensive LLMs. Pham et al. introduced SLM-Bench, the first large-scale benchmark dedicated to small language models, evaluating 15 models across 23 datasets and nine task categories [14]. Unlike earlier benchmarks that focused narrowly on accuracy, SLM-Bench integrates additional dimensions such as runtime, energy consumption, and CO2 emissions. This work demonstrates that some small models achieve competitive accuracy while offering markedly lower energy costs, highlighting their potential for sustainable deployment. However, the benchmark remains broad in scope: while it includes classification tasks such as sentiment analysis, it does not provide a detailed exploration of how SLMs handle the nuanced challenges of emotional tone, sarcasm, or aspect-based opinion mining.

Tan et al. proposed a learnware framework for organizing and deploying specialized SLMs across domains such as finance, healthcare, and mathematics [15]. This system outperformed individual SLMs and even surpassed large 70B models in domain-specific tasks, highlighting the potential of decentralized, privacy-preserving model reuse. However, its application to sentiment analysis remains largely unexplored.

Lepagnol et al. conducted a large-scale study on zero-shot classification, comparing models from 77M to 40B parameters across 15 datasets [16]. They showed that small models, when instruction-tuned and paired with suitable prompting strategies, can rival or surpass larger models. This finding reinforces the argument that model size is not the sole determinant of classification performance, particularly for sentiment-oriented datasets. Couto et al. examined compact multilingual LLMs on Iberian languages, introducing benchmarks that stress-test models in low-resource and end-user deployment scenarios [17]. Their results reveal persistent gaps for subtle linguistic phenomena like irony and sarcasm, suggesting that sentiment analysis in underrepresented languages remains an open challenge for compact models.

Koto et al. proposed a multilingual lexicon-based pretraining method for zero-shot sentiment analysis across 34 languages, including 25 low-resource ones [18]. Their approach outperformed GPT–3.5, BLOOMZ, and XGLM in many settings, underscoring that lexicon-augmented SLMs can generalize better across diverse linguistic contexts without relying on sentence-level annotations.

Liu et al. examined whether large language models actually possess sentiment sensitivity and are capable of consistently detecting emotional tone [19]. Their experiments revealed that while LLMs show basic sensitivity, their performance varies widely across datasets and model families. Misclassifications were especially common in cases involving subtle emotional cues such as irony or strongly contextual expressions. Importantly, the paper underscores that different architectures produce divergent outcomes even under identical testing conditions, suggesting that sentiment performance is far from a solved problem.

Together, these studies highlight efficiency, modularity, multilingual generalization, and sentiment awareness as key themes, yet none provide a systematic evaluation of small, locally run models on diverse sentiment datasets. Our work addresses this gap by benchmarking ≤8B models specifically for sentiment analysis.

3. Datasets Used

To ensure a comprehensive evaluation of compact large language models on sentiment analysis, we selected a diverse range of benchmark datasets capturing different challenges, from large-scale and fine-grained movie review corpora (IMDB, SST-2, SST-5) to domain-specific texts (FinancialPhraseBank), social media data with informal language and offensive content (Twitter Airline, OLID), emotion-oriented resources (Emotions), and aspect-based sentiment tasks (SentiHood). This diversity allows us to test not only polarity detection but also subtle affective cues and multi-entity reasoning, thereby reflecting both established benchmarks and realistic application scenarios. To establish a baseline of worst-case performance, we also calculated the probability of correct classification under random guessing. For an imbalanced dataset, this probability is not uniform across classes but instead depends on their relative frequencies. Specifically, if a dataset has K classes with class proportions $p_1, p_2, \ldots, p_k$, then the expected accuracy

of random guessing is given by [20]

$$P_{random} = \sum_{i=1}^{K} \quad p^2{}_i \ (1)$$

This formula captures the intuition that randomly guessing according to the class distribution yields higher expected accuracy when one or a few classes dominate. Finally, to contextualize model performance, we reported the strongest available State of the Art (SOTA) scores for each dataset (Table 1), which provide a practical upper bound on model capability.

- **IMDB Reviews Dataset** The IMDB Movie Reviews dataset is a widely used benchmark for sentiment analysis, containing 50,000 movie reviews labeled as either positive or negative. The dataset has an exact 50/50 split of positive and negative reviews and it was first introduced by Maas et al. in 2011. [21] Since this dataset is much larger than other datasets in our testing, we took a stratified sample of 10,000 reviews, preserving the 50/50 distribution. In our testing, we found that models perform slightly better on longer sentiments and slightly worse on shorter sentiments. We measured that the sample we took had near-identical mean and median values for review length to the original dataset.
- **Stanford Sentiment Treebank SST-5** The Stanford Sentiment Treebank SST-5 is a fine-grained sentiment analysis benchmark introduced by Socher et al. (2013) [22], constructed from the Rotten Tomatoes movie review corpus. The dataset was originally annotated across 25 distinct sentiment levels that were obtained using Amazon Mechanical Turk crowdworkers. For evaluation, the 25-way labels are typically collapsed into five categories (very negative, negative, neutral, positive, very positive), defining the widely used SST-5 task for sentence-level sentiment classification.
- **Stanford Sentiment Treebank SST-2** The Stanford Sentiment Treebank SST-2 is a binary sentiment classification benchmark introduced by Socher et al. (2013) [22] and popularized by the GLUE Benchmark [23]. The dataset was constructed by discarding the neutral class from the fine-grained annotations, leaving only positive and negative labels. It should be

noted that, while the SOTA scores reported in the table are based on the test set, our experiments were conducted on the validation set, since the gold test labels are withheld by the GLUE benchmark organizers.

- **FinancialPhraseBank** The FinancialPhraseBank is a domain-specific sentiment analysis dataset consisting of financial news sentences annotated as positive, negative, or neutral. It was introduced by Malo et al. in 2014 [24]. The sentences were labeled by a group of finance experts to ensure high-quality domain-relevant sentiment annotations.
- **Twitter Airline Sentiment Dataset** This dataset, introduced by Crowdflower in 2015 [25], contains tweets directed at major U.S. airlines, labeled as positive, neutral, or negative. Due to its social media origin, it introduces challenges such as informal language, sarcasm, and domain-specific vocabulary. It has become a benchmark for evaluating sentiment models in noisy, short-form text typical of Twitter data.
- **Emotions Dataset** The Emotions dataset is a fine-grained sentiment analysis benchmark containing English sentences annotated with six emotion categories (sadness, joy, love, anger, fear, surprise). It was introduced by Saravia et al. in 2018 [26], where it was proposed as a resource for evaluating emotion classification models beyond simple polarity detection
- **OLID (Offensive Language Identification Dataset)** The Offensive Language Identification Dataset (OLID) is a benchmark for offensive language detection in social media, annotated with a hierarchical three-level labeling scheme (offensive vs. not offensive; targeted vs. untargeted; and target category). It was introduced by Zampieri et al. (2019) [27] for the SemEval-2019 Task 6 (OffensEval) shared task, making it a widely used resource for studying abusive language classification. OLID is made up of 3 levels and we are using the first level which is the largest and most commonly used subset.
- **SentiHood** The SentiHood dataset is an aspect-based sentiment analysis (ABSA) benchmark focused on urban neighborhood discussions extracted from Yahoo! Answers. Each instance consists of a sentence mentioning one or more neighborhoods, annotated with aspect categories (e.g., price, safety, live, quiet) and their associated sentiment polarity (positive, negative, none). It was introduced by Saeidi et al. in 2016 [28].

Table 1. Random guessing probabilities and reported SOTA scores for benchmark sentiment analysis datasets.

Dataset	Random Guessing (%)	SOTA Score (%)
IMDB	50.00	96.21 [29]
SST-5	21.48	60.48 [30]
SST-2	50.02	97.9 [31]
FinancialPhraseBank	44.76	90.9 [32]
Twitter Airline	46.39	92.36 [33]
Emotions	24.40	88.5 [34]
OLID (Level A)	55.62	85.12 [35]
SentiHood	60.08	93.8 [36]

4. Large Language Models used

Large Language Models (LLMs) were organized into four categories according to their parameter size: large-scale models in the 7B-8B range, mid-sized models in the 3B–4B range,

smaller models slightly above or around 1B parameters and sub-1B models, the largest of which contains 0.5B parameters. Model size generally determines the overall performance potential of LLMs across diverse tasks, but larger models require greater memory capacity and typically operate more slowly, even on capable hardware. An additional aim of this study was to evaluate the practicality of newer, compact LLMs for sentiment analysis. To this end, several sub-1B models were selected and assessed for their performance characteristics. All tested models were quantized to Q8 (8-bit), except for Gemma3 1B QAT, which applies a specialized quantization-aware training method to achieve improved performance at Q4_0 (4-bit) quantization [37]. All models considered were non-reasoning variants. Although reasoning-augmented LLMs have demonstrated substantial gains on complex tasks, they also introduce significant latency and computational overhead. While such capabilities may be beneficial in sentiment analysis for scenarios without strict time constraints, they fall outside the scope of this paper.

One notable exception concerns the Qwen3 family, which achieves state-of-the-art results within its respective categories. The 1.7B, 4B, and 8B Qwen3 models are hybrid reasoning architectures, allowing reasoning to be toggled at inference. However, when tested in non-reasoning mode, their performance was unsatisfactory, and they were excluded from comparison to avoid misrepresenting typical use. Qwen3 0.6B was tested across all eight datasets but consistently failed to yield meaningful results, leading to its removal from tables and evaluations [38].

All models employed were instruction-tuned and evaluated in a zero-shot setting for sentiment analysis. No additional fine-tuning was applied beyond their publicly released versions.

4.1 Models in 7B to 8B Range

- **Llama 3 8B** Part of Meta's Llama 3 family, this 8B parameter model provides strong general-purpose performance across reasoning, comprehension, and generation tasks. It is widely adopted as a standard mid-sized baseline in research and practice. Llama 3 8B is one of the most widely used LLMs in the 8B range. While it does have a newer version that scores slightly higher on intelligence benchmarks, we found that Llama 3 8B is more reliable when it comes to sentiment analysis tasks than Llama 3.1 8B. [39]
- **Granite 3.2 8B** Developed by IBM, Granite 3.2 8B is an open foundation model optimized for enterprise and domain-specific workloads. It emphasizes trustworthy outputs and efficiency, making it well-suited for applied research settings. [40]
- **MiniCPM 4 8B** MiniCPM-4 is a lightweight, high-performance model developed by OpenBMB. The 8B variant balances efficiency with strong multilingual and reasoning capabilities. [41]
- **OLMoE-1B-7B-0125** Released by the Allen Institute for AI, this model follows a Mixture-of-Experts (MoE) design with 64 experts and 1B active parameters per forward pass from a 7B parameter pool [42]. Sparse expert routing reduces compute compared to dense 7B–8B models [43]. While MoE architectures are more common in >20B parameter models, AllenAI's OLMoE is one of the few implementations scaled down to the 7B range. The version we tested is an updated variant released in January of 2025.

4.2 Models in 3B to 4B Range

- **Gemma 3 4B** Released by Google DeepMind, Gemma 3 4B is designed for efficient deployment with high-quality text generation and alignment. It achieves strong performance relative to its size. This model is multi-modal and can work with visual inputs. [44]

- **Phi-4 Mini** A member of Microsoft's Phi series, Phi-4 Mini emphasizes compactness and reasoning efficiency. It is optimized for instructional tasks and educational applications. The model is 3 billion parameters in size. [45]
- **Llama 3.2 3B** A smaller variant in the Llama 3.2 family, the 3B model targets lower memory footprints while retaining competitive performance on common NLP. [39]

4.3 Models around 1B parameters

- **Liquid LFM-2 1.2B** A compact model optimized for research on quantization and low-resource inference. It is frequently employed in efficiency studies and resource-constrained deployment scenarios. [46]
- **Llama 3.2 1B** The smallest official member of the Llama 3.2 family, this 1B parameter variant is designed for experimentation on edge devices and constrained environments. [39]
- **Gemma 3 1B QAT** A quantization-aware trained (QAT) variant of Gemma 3, designed to preserve accuracy under aggressive compression, making it highly efficient for deployment on resource-limited systems. [44] [37]

4.4 Sub-1B parameter models

- **MiniCPM 4 0.5B** A 0.5B variant of MiniCPM, this model is tuned for speed and compact deployment while maintaining acceptable performance on standard NLP tasks. [41]
- **SmolLM 2 360M** Part of the SmolLM family, this 360M parameter model is designed as an extremely lightweight transformer for quick experimentation and edge-level deployment. [47]
- **Ernie 4.5 0.3B** Released by Baidu, Ernie 4.5 0.3B is a compact multilingual model integrating knowledge-enhanced pretraining with efficient inference performance. It is the smallest model in the family and the only dense (non-MoE) model. [48]
- **SmolLM 2 135M** The smallest model in the SmolLM 2 family, with 135M parameters, designed primarily for testing scalability and micro-deployment scenarios. This is the smallest model that we tested and it pushes the limits of usability at very little compute cost. [47]

5. Methodology

5.1 Running automation script

We automated evaluation with lightweight Python scripts that handle both dataset loading and model prompting. Each dataset was normalized to a minimal (text, label) format, with adaptations for tasks such as aspect-based sentiment analysis or multi-class emotion classification. The scripts support multiple file formats (CSV, JSONL, Parquet), ensuring consistency across heterogeneous datasets.

5.1.1 Prompts and Decoding

Each prompt was deliberately minimal, restricting the model to a closed label set. For example, binary sentiment tasks required exactly positive or negative, fine-grained SST-5 required a digit in {0, 1, 2, 3, 4}, and ABSA tasks required one of {positive, negative, none}. To enforce validity, outputs were post-processed with simple regex and keyword matching rules that coerce stray variants into the canonical label set.

5.1.2 Deterministic Decoding

All runs were performed with temperature fixed at T = 0, corresponding to greedy decoding. Given logits z_t at step t, the probability distribution is defined as

$$p_t(i) = \frac{exp(z_{t,i}/T)}{\sum_j exp(z_{t,j}/T)} \quad (2)$$

As T → 0, the distribution collapses to a point mass on the maximum logit, yielding

$$y_t = argmax_i z_{t,i} \quad (3)$$

This setting is equivalent to top-k sampling with k = 1 or nucleus sampling with p → 0. It guarantees reproducibility by eliminating randomness and ensures that model comparisons are strictly deterministic.

5.2 Evaluation Metrics

The models were evaluated using standard classification metrics [49]:

- **Accuracy:** $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$ (3)
- **Macro-F1 score:** $MacroF1 = \frac{1}{C}\sum_{i=1}^{C} \frac{2 \cdot Precision_i \cdot Recall_i}{Precision_i + Recall_i}$ (4)
- **Matthews Correlation Coefficient (MCC):** MCC is a correlation coefficient between the observed and predicted binary classifications. It takes into account true and false positives and negatives, producing a value between −1 and +1. A value of +1 indicates perfect prediction, 0 corresponds to random prediction, and −1 indicates total disagreement between prediction and ground truth. [50] Unlike accuracy or F1-score, MCC is considered a balanced measure because it remains informative even when the dataset is imbalanced. This makes it particularly suitable for sentiment classification tasks where some classes may be underrepresented.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (5)$$

- **Cohen's Kappa (κ):** Cohen's Kappa measures the agreement between predicted and true labels, adjusting for the possibility of agreement occurring by chance. It ranges from −1 (complete disagreement) to +1 (perfect agreement), with 0 representing chance-level performance. [51] The advantage of κ is that it penalizes models that achieve seemingly high accuracy simply by exploiting class imbalances. In multi-class sentiment analysis, this is especially valuable because it reflects how reliably the model performs across all categories, rather than being dominated by the most frequent class.

$$\kappa = \frac{p_0 - p_e}{1 - p_e} \quad (6)$$

For the **SST-5 dataset**, the sentiment labels form an ordinal scale ranging from very negative to very positive. While the task is evaluated as classification, the ordered nature of the labels allows prediction errors to be meaningfully interpreted in terms of distance between categories. To capture the magnitude of these deviations, we additionally report standard regression-style error metrics that quantify absolute and squared differences between predicted and true labels [52]

- **Mean Absolute Error:** $MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$ (7)
- **Mean Squared Error:** $MAE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$ (8)
- **Root Mean Squared Error:** $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$ (9)

5.3 Speed and Latency testing

It is widely recognized that larger language models (LLMs), which require greater memory capacity, also demand more computational resources during inference. However, architectural differences can further influence performance, allowing certain models to achieve advantages in speed. To account for these variations, we conducted measurements of speed and latency for several LLMs executed locally.

For sentiment analysis tasks, prompt processing speed is a more meaningful performance indicator than raw token throughput (tokens per second). This is because models must process long input sequences while often producing only one or two output tokens. Such processing is primarily constrained by the memory bandwidth of the RAM/VRAM allocated to the processor. To evaluate this dimension of performance, we constructed three dedicated datasets designed specifically for controlled prompt ingestion, rather than for accuracy evaluation.

Our experiments showed that response latency scales significantly with input length: for instance, when classifying long IMDB reviews, models often required more than ten times the processing time compared to short reviews. Accordingly, we defined three test sets with average sentence lengths of approximately 30, 150, and 1200 tokens. Sentences were primarily sampled from the IMDB dataset, as it provides the widest variation in input length, including reviews exceeding 1000 tokens.

All tests were performed using llama.cpp (Windows Vulkan backend, version 1.50.2). The hardware consisted of an AMD Radeon 890M integrated GPU with 8 GB of dedicated GPU memory (BIOS allocation) and 4 GB of shared GPU memory (32 GB total) with a rated bandwidth of 120 GB/s. The GPU driver version was 25.10.25.10-250825a-418637C, and the Vulkan API version was 1.4.315. Given that the tests were run on a laptop platform, fan profiles influenced power delivery. Stable performance was observed at the lowest fan setting, where power draw remained at approximately 10 W with minimal spikes. All benchmarks were therefore conducted under these conditions to avoid variability from power fluctuations.

It is important to note that these results are specific to GPUs operating under Vulkan. Models optimized for other backends such as MLX, CUDA, or OpenVINO (e.g., MiniCPM-4) may exhibit different performance characteristics. Furthermore, while CPU inference on the same test system (Ryzen AI 9 HX 370) achieved comparable token throughput, prompt processing was three to four times slower, rendering CPU execution impractical for high-volume sentiment analysis workloads.

6. Accuracy Results

The evaluation results are presented across ten tables, with additional breakdowns for SST-5 and the Emotions dataset to account for their fine-grained label structures. Each table reports five complementary metrics-accuracy, macro-F1, micro-F1, Matthews Correlation Coefficient (MCC), and Cohen's κ, chosen to capture both overall correctness and robustness under class imbalance.

The highest score within each metric group is highlighted in bold. In some cases, particularly with the smallest models, the "best" result within a group may reflect the least inadequate performance rather than genuine effectiveness, but these values are retained for completeness and comparability.

6.1 IMDB Dataset

On binary sentiment classification of long-form reviews, larger models consistently outperformed compact ones. OLMoE-1B-7B achieved the strongest performance (accuracy 95.3%), closely followed by MiniCPM-4 8B and IBM Granite 3.2 8B. Among smaller models, Liquid LFM-2 1.2B and Gemma 3 1B QAT demonstrated competitive accuracy (≥87%), indicating that carefully optimized 1B–1.2B models can approach near-state-of-the-art results on balanced datasets. Among the smallest models, Ernie 4.5 0.3B greatly outperformed its peers and came close to much larger models, suggesting that this dataset could've been part of its training data. (Table 2)

Table 2. IMDB Dataset results

Model	Accuracy	Macro-F1	MCC	Cohen's κ
SmolLM2 135M	0.617	0.585	0.281	0.234
Ernie 4.5 0.3B	**0.872**	**0.872**	**0.749**	**0.744**
SmolLM2 360M	0.796	0.791	0.620	0.592
MiniCPM4 0.5B	0.635	0.583	0.381	0.270
Gemma 3 1B QAT	0.874	0.874	0.751	0.748
Llama 3.2 1B Instruct	0.826	0.824	0.664	0.652
Liquid LFM2 1.2B	**0.921**	**0.921**	**0.842**	**0.842**
Llama 3.2 3B Instruct	0.694	0.688	0.403	0.388
Phi-4 Mini	**0.925**	**0.925**	**0.850**	**0.850**
Gemma 3 4B	0.896	0.895	0.803	0.792
OLMoE-1B-7B-0125	**0.953**	**0.953**	**0.907**	**0.906**
Llama 3 8B Instruct	0.917	0.917	0.837	0.834
IBM Granite 3.2 8B	0.934	0.934	0.869	0.868
MiniCPM4 8B	0.945	0.945	0.891	0.890

6.2 Stanford Sentiment Treebank SST-5

SST-5. Fine-grained sentiment classification proved especially challenging. Even larger models struggled, with the best result (Gemma 3 4B, accuracy 52.1%) falling short of the reported SOTA (Table 3). Smaller models frequently collapsed predictions into a subset of classes, yielding poor macro-F1 and inflated per-class sparsity (Table 4). This highlights the difficulty of fine-grained polarity detection in zero-shot settings, where subtle contextual cues are often missed.

Table 3. SST-5 Overall results

Model	Accuracy	MAE	MSE	RMSE	Cohen's κ
SmolLM2 135M	0.165	**1.28**	**2.37**	**1.54**	-0.00869
Ernie 4.5 0.3B	0.127	2.05	5.96	2.44	0.000438
SmolLM2 360M	**0.176**	1.95	5.5	2.35	-0.0118
MiniCPM4 0.5B	0.167	1.80	4.88	2.21	**0.0132**
Gemma 3 1B QAT	0.255	1.23	2.51	1.58	**0.0626**
Llama 3.2 1B Instruct	0.187	1.78	4.82	2.20	0.0110
Liquid LFM2 1.2B	**0.296**	**1.12**	**2.16**	**1.47**	0.0568
Llama 3.2 3B Instruct	0.330	0.865	1.33	1.15	0.164
Phi-4 Mini	0.455	0.623	0.799	0.894	0.295
Gemma 3 4B	**0.521**	**0.543**	**0.689**	**0.830**	**0.366**
OLMoE-1B-7B-0125	0.316	1.065	2.015	1.419	0.108
Llama 3 8B Instruct	0.426	0.640	**0.781**	**0.884**	0.244
IBM Granite 3.2 8B	**0.496**	**0.588**	**0.781**	**0.884**	**0.377**
MiniCPM4 8B	0.437	0.656	0.854	0.924	0.306

Table 4. SST-5 per-class F1 scores

Model	V. Negative	Negative	Neutral	Positive	V. Positive
SmolLM2 135M	0.126	0.0453	**0.283**	0	0
Ernie 4.5 0.3B	0.224	0	0.00513	0	0
SmolLM2 360M	0.0488	0.0673	0	0	**0.294**
MiniCPM4 0.5B	**0.246**	**0.101**	0	**0.175**	0.0816
Gemma 3 1B QAT	**0.286**	0.0529	0	**0.405**	0
Llama 3.2 1B Instruct	0.235	0.296	0	0.0372	0
Liquid LFM2 1.2B	0.238	**0.456**	**0.152**	0.175	**0.0242**
Llama 3.2 3B Instruct	0.397	0.109	**0.357**	0.462	0.0933
Phi-4 Mini	**0.44**	0.543	0.323	0.571	0
Gemma 3 4B	0.345	**0.647**	0.265	**0.596**	**0.326**
OLMoE-1B-7B-0125	0.007	0.327	0.099	0.406	0.311
Llama 3 8B Instruct	0.0685	**0.64**	0.345	0.415	0.0675
IBM Granite 3.2 8B	0.514	0.319	0.375	**0.592**	**0.656**
MiniCPM4 8B	**0.536**	0.0455	**0.412**	0.581	0.516

6.3 Stanford Sentiment Treebank SST-2

In contrast, binary sentiment classification on SST-2 yielded strong results across model families. Phi-4 Mini (93.2%), Gemma 3 4B (93.0%), and IBM Granite 3.2 8B (93.3%) performed comparably, with OLMoE-1B-7B narrowly leading at 94.0%. Even lightweight 1B models such as Liquid LFM-2 (90.4%) showed reliable performance (Table 5), reflecting the relative simplicity of binary sentiment tasks when compared to SST-5.

Table 5. SST-2 results

Model	Accuracy	Macro-F1	MCC	Kohen's κ
SmolLM2 135M	0.766	0.764	0.547	0.534
Ernie 4.5 0.3B	**0.818**	**0.814**	**0.671**	**0.637**
SmolLM2 360M	0.620	0.566	0.370	0.251
MiniCPM4 0.5B	0.795	0.790	0.630	0.592
Gemma 3 1B QAT	0.882	0.882	0.764	0.764
Llama 3.2 1B Instruct	0.878	0.878	0.772	0.758
Liquid LFM2 1.2B	**0.904**	**0.904**	**0.809**	**0.808**
Llama 3.2 3B Instruct	0.686	0.659	0.470	0.378
Phi-4 Mini	**0.932**	**0.932**	**0.867**	**0.865**

Gemma 3 4B	0.930	0.930	0.866	0.860
OLMoE-1B-7B-0125	**0.940**	**0.940**	**0.883**	**0.881**
Llama 3 8B Instruct	0.497	0.342	0.0746	0.0111
IBM Granite 3.2 8B	0.933	0.933	0.874	0.867
MiniCPM4 8B	0.921	0.921	0.847	0.842

6.4 FinancialPhraseBank

Domain-specific sentiment analysis in finance revealed a clear stratification. MiniCPM-4 8B achieved the highest accuracy (79.1%), followed closely by Phi-4 Mini and Gemma 3 4B. While sub-1B models struggled, Liquid LFM-2 (63.0%) and Llama 3.2 1B (63.0%) provided mid-tier results (Table 6), suggesting limited domain transfer without fine-tuning.

Table 6. FinancialPhraseBank results

Model	Accuracy	Macro-F1	MCC	Cohen's κ
SmolLM2 135M	0.295	0.213	0.119	0.0226
Ernie 4.5 0.3B	0.309	0.291	**0.217**	0.141
SmolLM2 360M	**0.589**	**0.447**	0.205	**0.201**
MiniCPM4 0.5B	0.328	0.323	0.19	0.115
Gemma 3 1B QAT	0.471	0.491	0.352	0.259
Llama 3.2 1B Instruct	**0.63**	0.421	0.224	0.159
Liquid LFM2 1.2B	**0.63**	**0.595**	**0.397**	**0.383**
Llama 3.2 3B Instruct	0.641	0.448	0.263	0.233
Phi-4 Mini	0.735	**0.738**	**0.586**	**0.566**
Gemma 3 4B	**0.744**	0.712	0.555	0.546
OLMoE-1B-7B-0125	0.738	0.727	0.526	0.526
Llama 3 8B Instruct	0.659	0.492	0.312	0.269
IBM Granite 3.2 8B	0.647	0.448	0.277	0.236
MiniCPM4 8B	**0.791**	**0.758**	**0.608**	**0.597**

6.5 Twitter Airline Sentiment

Social media sentiment classification exposed substantial model variance. While compact models such as Ernie 4.5 (72.6%) achieved respectable results, stronger performance was seen in Phi-4 Mini (80.9%) and Gemma 3 4B (81.1%). Performance degradations in some larger models (e.g., Llama 3 8B, 37.7%) underscore the instability of zero-shot inference on noisy, sarcasm-rich text, as well as, the models' ability to follow instructions without defaulting to standard responses. (Table 7)

Table 7. Twitter Airline Sentiment results

Model	Accuracy	Macro-F1	MCC	Cohen's κ
SmolLM2 135M	0.202	0.138	0.0766	0.0213
Ernie 4.5 0.3B	**0.726**	**0.515**	**0.464**	**0.413**
SmolLM2 360M	0.237	0.236	0.0771	0.0437
MiniCPM4 0.5B	0.672	0.485	0.307	0.282
Gemma 3 1B QAT	0.715	0.496	0.496	0.459
Llama 3.2 1B Instruct	0.706	**0.598**	**0.510**	**0.491**
Liquid LFM2 1.2B	**0.742**	0.542	0.500	0.447
Llama 3.2 3B Instruct	0.212	0.118	0.0103	0.000372
Phi-4 Mini	0.809	**0.751**	**0.656**	**0.651**
Gemma 3 4B	**0.811**	0.726	0.640	0.632
OLMoE-1B-7B-0125	0.753	0.576	0.0199	0.00526
Llama 3 8B Instruct	0.377	0.415	0.222	0.145
IBM Granite 3.2 8B	**0.791**	**0.747**	**0.639**	**0.631**
MiniCPM4 8B	0.329	0.374	0.270	0.128

6.6 Emotions Dataset

Multi-class emotion detection presented another challenging benchmark. IBM Granite 3.2 8B emerged as the strongest performer (accuracy 56.6%), significantly outperforming most smaller models. Among mid-sized candidates, Phi-4 Mini and Gemma 3 4B performed competitively (Table 8). Sub-1B models largely failed to capture the diversity of emotional categories, frequently defaulting to a small subset of labels (Table 9) .

Table 8. Emotions Dataset Overall results

Model	Accuracy	Macro-F1	MCC	Cohen's κ
SmolLM2 135M	0.238	0.109	0.0182	0.0149
Ernie 4.5 0.3B	0.137	0.136	0.0189	0.0156
SmolLM2 360M	**0.289**	0.0748	-0.0145	-0.00138
MiniCPM4 0.5B	0.168	0.116	0.0181	0.0133
Gemma 3 1B QAT	**0.198**	0.123	**0.107**	**0.0695**
Llama 3.2 1B Instruct	0.183	**0.153**	0.0734	0.0491
Liquid LFM2 1.2B	0.113	0.107	0.0112	0.00884
Llama 3.2 3B Instruct	0.354	0.354	0.277	0.246
Phi-4 Mini	0.464	0.442	0.349	0.334
Gemma 3 4B	**0.465**	**0.447**	**0.375**	**0.351**
OLMoE-1B-7B-0125	0.165	0.114	0.043	0.03
Llama 3 8B Instruct	0.395	0.380	0.298	0.275
IBM Granite 3.2 8B	**0.566**	**0.477**	**0.434**	**0.428**
MiniCPM4 8B	0.403	0.354	0.269	0.252

Table 9. Emotions Dataset per-class F1 scores

Model	Sadness	Joy	Love	Anger	Fear	Surprise
SmolLM2 135M	0.41	0.188	0	0	0	0.0588
Ernie 4.5 0.3B	0.211	0.135	0.108	**0.102**	**0.192**	0.0696
SmolLM2 360M	**0.449**	0	0	0	0	0
MiniCPM4 0.5B	0.251	**0.216**	**0.146**	0	0.00844	0.0769
Gemma 3 1B QAT	0	**0.204**	**0.19**	**0.301**	0.0129	0.0274
Llama 3.2 1B Instruct	0.0734	0.0992	0.178	0.271	0.161	**0.138**
Liquid LFM2 1.2B	**0.157**	0.0497	0.0209	0.137	**0.209**	0.0661
Llama 3.2 3B Instruct	0.424	0.236	0.28	0.523	0.348	0.311
Phi-4 Mini	0.521	0.443	**0.344**	0.467	**0.548**	0.33
Gemma 3 4B	**0.538**	**0.467**	0.323	**0.544**	0.414	**0.396**
OLMoE-1B-7B-0125	0.0656	0.0586	0.0495	0.253	0.219	0.0351
Llama 3 8B Instruct	0.23	0.496	0.336	0.433	**0.446**	0.343
IBM Granite 3.2 8B	**0.617**	**0.674**	**0.343**	**0.525**	0.336	**0.365**
MiniCPM4 8B	0.459	0.379	0.325	0.415	0.4	0.146

6.7 OLID

Offensive language detection (Level A) was highly demanding for compact models. All sub-1.2B models failed to produce meaningful outputs, often defaulting to majority-class predictions that inflated accuracy but yielded near-zero κ and MCC. For example, SmolLM2 135M and 360M appeared to reach 72% accuracy, but with $\kappa \approx 0$, showing no real agreement beyond chance. Reliable performance only emerged with larger architectures: IBM Granite 3.2 8B (76.8%) led the group, followed by MiniCPM-4 8B (73.9%) and Llama 3 8B (70.4%). Mid-sized models such as Gemma 3 4B (68.2%) and Phi-4 Mini (71.2%) showed competitive results with balanced agreement, but anything below 1B parameters was essentially unusable (Table 10).

Table 10. OLID results

Model	Accuracy	Macro-F1	MCC	Cohen's κ
SmolLM2 135M	0.719	0.430	**0.0209**	**0.00666**
Ernie 4.5 0.3B	0.448	**0.443**	0.00853	**0.00666**
SmolLM2 360M	**0.722**	0.419	0	0
MiniCPM4 0.5B	0.282	0.224	0.0134	0.00127
Gemma 3 1B QAT	**0.559**	**0.549**	**0.208**	**0.173**
Llama 3.2 1B Instruct	0.308	0.281	-0.024	-0.00857
Liquid LFM2 1.2B	0.278	0.218	0	0
Llama 3.2 3B Instruct	0.539	0.539	0.304	0.214
Phi-4 Mini	**0.712**	0.558	0.161	0.144
Gemma 3 4B	0.682	**0.671**	**0.445**	**0.385**
OLMoE-1B-7B-0125	0.598	0.588	0.285	0.24
Llama 3 8B Instruct	0.704	0.683	0.418	0.387
IBM Granite 3.2 8B	**0.768**	**0.715**	0.430	0.430
MiniCPM4 8B	0.739	0.713	**0.459**	**0.438**

6.8 SentiHood

Aspect-based sentiment analysis proved highly sensitive to model capacity. Gemma 3 4B attained the strongest result (84.4%), with Phi-4 Mini close behind (78.8%). Sub-1B models displayed frequent misclassifications, and even some larger models underperformed (e.g., IBM Granite 3.2 8B at 60.0%), suggesting that ABSA remains a non-trivial challenge without explicit aspect-level supervision (Table 11).

Table 11. Sentihood dataset results

Model	Accuracy	Macro-F1	MCC	Cohen's κ
SmolLM2 135M	0.389	0.257	0.129	0.082
Ernie 4.5 0.3B	0.609	0.399	0.203	0.257
SmolLM2 360M	0.621	0.410	0.260	0.309
MiniCPM4 0.5B	**0.782**	**0.492**	**0.319**	**0.476**
Gemma 3 1B QAT	0.072	0.060	0.056	0.017
Llama 3.2 1B Instruct	**0.463**	**0.370**	**0.251**	**0.207**
Liquid LFM2 1.2B	0.354	0.261	0.179	0.112
Llama 3.2 3B Instruct	0.000	0.000	0.000	0.000
Phi-4 Mini	0.788	0.550	0.446	0.571
Gemma 3 4B	**0.844**	**0.559**	**0.466**	**0.659**
OLMoE-1B-7B-0125	**0.637**	**0.476**	0.33	**0.364**
Llama 3 8B Instruct	0.086	0.074	0.056	0.016
IBM Granite 3.2 8B	0.600	0.435	**0.340**	0.318
MiniCPM4 8B	0.495	0.337	0.267	0.210

Summary: Across datasets, the accuracy results reveal that the strongest overall performance did not come from the largest models, but rather from the mid-sized 3–4B range. In particular, Gemma 3 4B and Phi-4 Mini consistently achieved the highest or near-highest scores, often surpassing the 7–8B models on tasks such as SST-2, Twitter Airline, and Sentihood, while remaining highly competitive even on more demanding datasets like Emotions and FinancialPhraseBank. Larger models such as MiniCPM-4 8B and Granite 3.2 8B retained an advantage on specific benchmarks—notably IMDB and certain fine-grained emotion classification tasks—but their lead was neither consistent nor decisive. By contrast, models below 1B parameters generally failed to generalize, with their performance frequently collapsing to

trivial label distributions despite occasional isolated successes. Taken together, these results indicate that the most reliable balance of accuracy across diverse sentiment analysis settings is achieved by compact mid-sized models, with 7–8B systems offering marginal improvements only in select cases and at substantially higher computational cost.

7. Analyzing Model Failures

During evaluation, several lightweight instruction-tuned models systematically failed to comply with the classification directive (e.g., Table 11, Llama 3.2 3B Instruct). Instead of generating the expected single-token output such as positive, negative, or none, these models frequently reproduced fragments of the instruction prompt itself (e.g., "Please respond with one of the following words..."). Extending the decoding budget from a single token to 8–25 tokens did not resolve the issue; rather, it allowed the model to continue paraphrasing or elaborating on the instructions. In these cases, the parser could not extract a valid label, causing predictions to default to none. Similar behaviors have been reported in prior work, where instruction-tuned chat models echo or re-interpret prompts when evaluated outside of their intended template format [53; 54].

This failure mode reflects a broader mismatch between completion-style evaluation and the way many instruction-tuned models have been optimized. Chat-oriented models are typically trained and deployed under specific templates, where reinforcement from human feedback rewards polite, verbose responses. When these templates are not used, models may revert to their safer defaults of restating or expanding the given instructions, rather than emitting a constrained label [55]. Prior studies on prompt sensitivity have shown that small variations in formatting, role definitions, or system instructions can lead to large shifts in accuracy, even with identical underlying models [56].

A second, distinct failure pattern emerged in fine-grained sentiment classification tasks such as SST-5 and the Emotions dataset. Here, smaller models often ignored parts of the label space and collapsed predictions into only a few categories. Despite repeated adjustments to the prompt format, the models consistently defaulted to high-frequency or semantically safe labels such as positive or neutral. This tendency aligns with documented issues of label bias and surface form competition, where language models exhibit strong priors toward particular verbalizers and over-rely on frequent tokens at the expense of rare ones [57; 58]. The result was inflated sparsity across classes, leading to poor macro-F1 scores even when raw accuracy appeared less affected.

These observations highlight that failures were not arbitrary but arose from identifiable mechanisms: (i) instruction-following models misinterpreting prompts when stripped from their native templates, and (ii) smaller models lacking the capacity to fully represent fine-grained distinctions, leading to class collapse. Importantly, the use of a consistent prompt across models and datasets in this study ensured comparability, even though prompt design itself is a recognized source of variance in LLM evaluation [57]. Future work may explore mitigations such as contextual calibration, constrained decoding, or template alignment, but the uniform prompting strategy adopted here provides a fair and controlled baseline for cross-model analysis.

8. Normalized Accuracy Metric

A challenge in evaluating the performance of large language models (LLMs) across heterogeneous sentiment analysis datasets lies in the fact that raw accuracy is not directly

comparable. This stems from two issues. First, the baseline accuracy achievable by random guessing varies widely depending on the number of classes and their distribution. For example, in the IMDB dataset, a binary classification task, random guessing achieves 50% accuracy, whereas in the five-class SST-5 dataset the baseline is only 20% (Figure 1). Second, the attainable performance upper bound also differs between datasets: while state-of-the-art (SOTA) models can exceed 95% accuracy on IMDB, the best reported results on SST-5 remain around 60%. Although it is theoretically possible for other models to surpass these scores, they are typically obtained through task-specific fine-tuning. By contrast, the smaller LLMs evaluated here are tested in a zero-shot setting, where such levels of performance are not realistically anticipated.

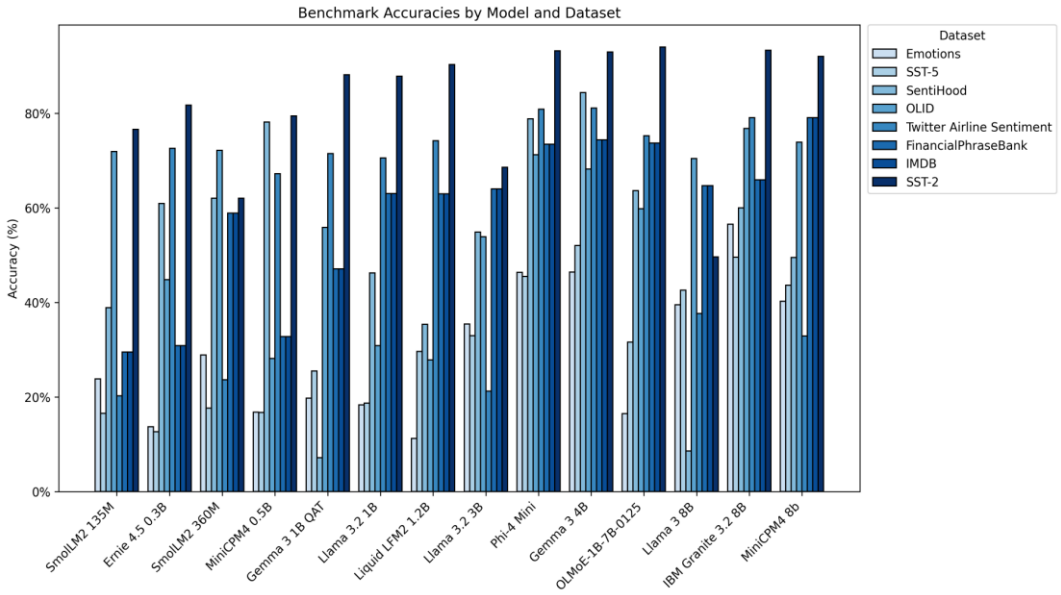


Fig. 1. Model accuracies across dataset before normalization

Existing metrics such as Matthews Correlation Coefficient (MCC) or Cohen's Kappa attempt to account for chance agreement, but they still suffer from comparability issues. For instance, obtaining an MCC of 0.4 on a complex five-class dataset may indicate stronger relative performance than achieving 0.8 on a balanced binary dataset. Thus, these measures do not provide a unified scale across tasks with different inherent difficulties. To address this, we introduce a **normalized accuracy metric**. For each dataset, we define a performance range between two reference points:

- **Lower bound:** the probability of correct classification by random guessing (A_{rand}), computed as the weighted sum of class proportions.
- **Upper bound:** the accuracy of the strongest available SOTA model (A_{SOTA}) reported in the literature.

The observed accuracy of a model on the dataset (A_{model}) is then linearly rescaled into a 0–100 scale:

$$NormAcc = \frac{A_{model} - A_{rand}}{A_{SOTA} - A_{rand}} \times 100\% \quad (10)$$

In this formulation, a model matching random guessing receives a score of 0, while a model achieving the SOTA accuracy receives 100. Scores below random guessing yield negative values, penalizing cases where the model fails to comply with the evaluation format under the given

prompt template (Figure 2). For example, in some instances models produced long descriptive outputs instead of the required single-word classification. This treatment ensures that poor performance on a single dataset meaningfully reduces the model's overall average (Figure 3).

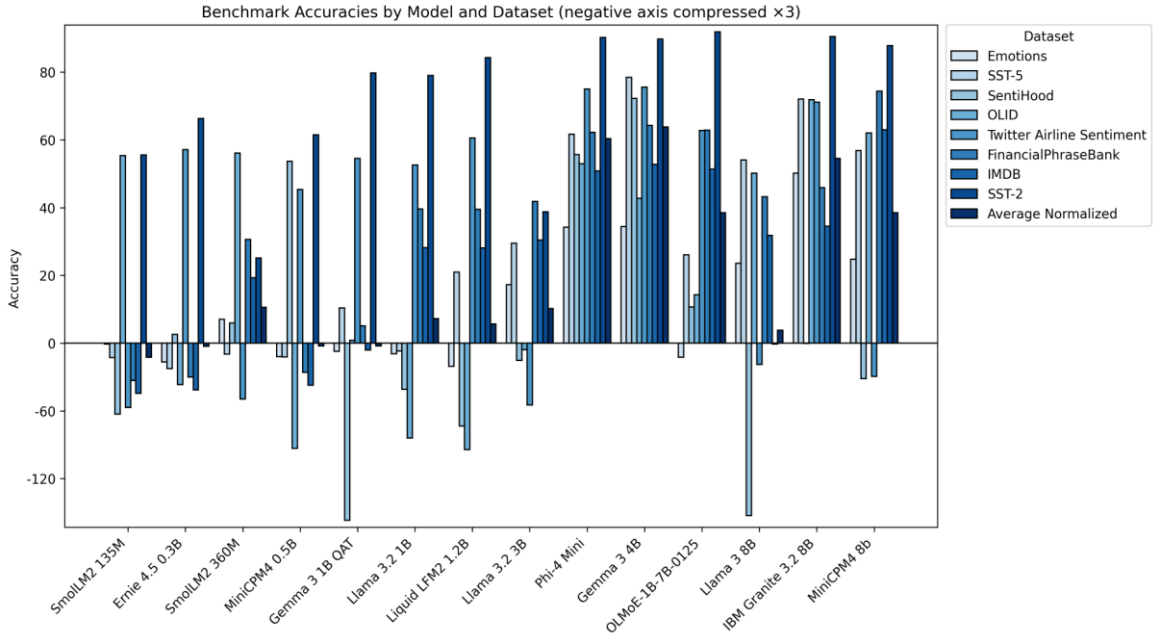


Fig. 2. Model accuracies across dataset after normalization

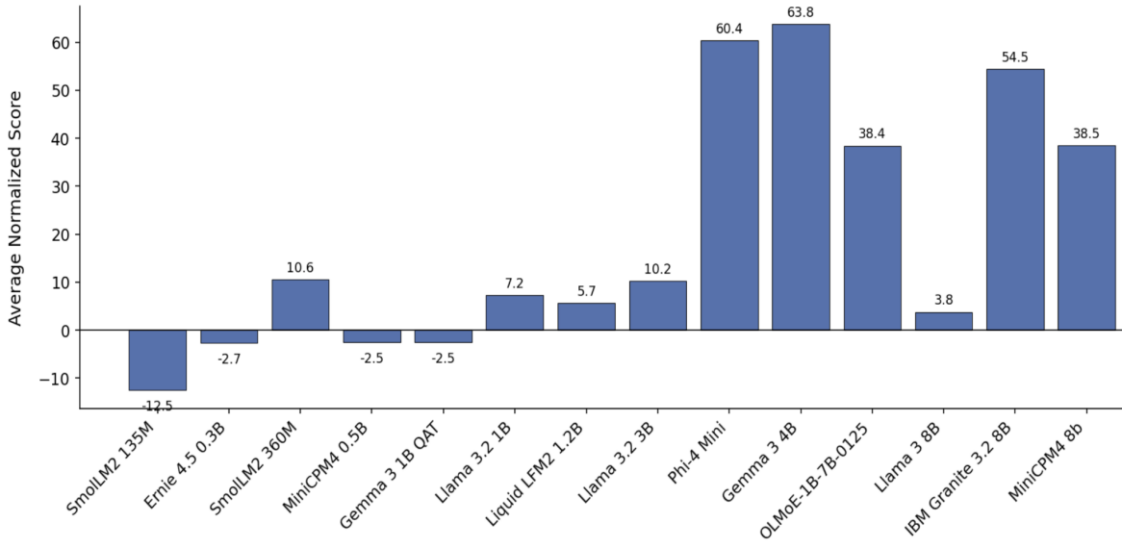


Fig. 3. Average normalized accuracy scores with penalization

At the same time, to avoid overly harsh penalization, we also consider an alternative version of the metric where scores below random guessing are clipped to zero (Figure 4). This second view treats severe underperformance as contributing no positive value, rather than further lowering the average, and can be more appropriate when the goal is to prevent outliers from disproportionately affecting results (Figure 5).

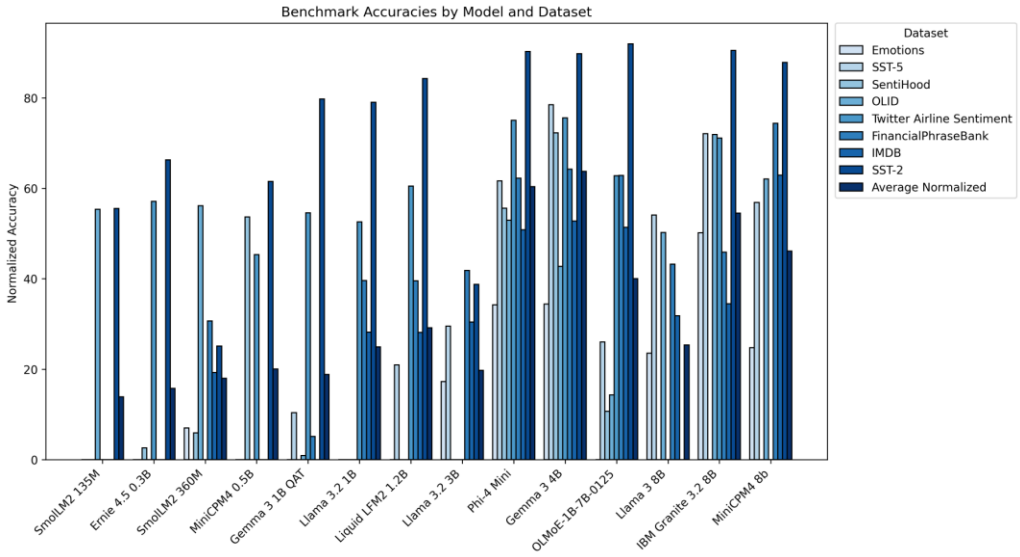


Fig. 4. Model accuracies across dataset after normalization with no penalty

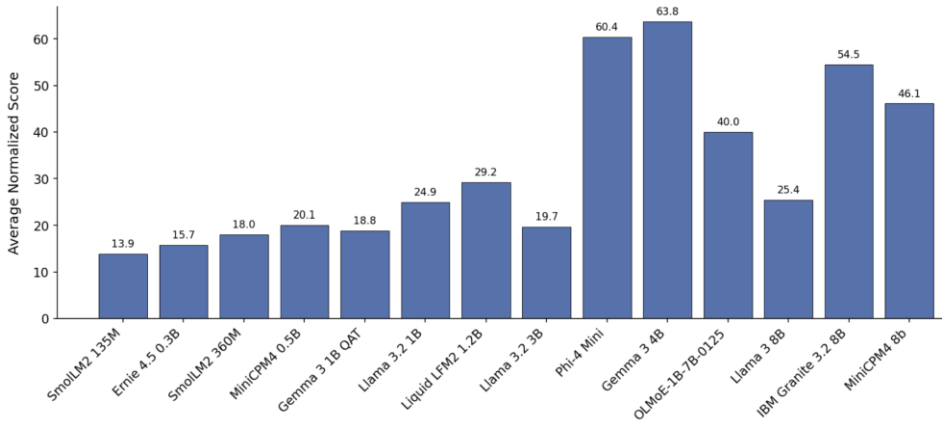


Fig. 5. Average normalized accuracy scores with no penalty

By normalizing in this way, we produce a comparable metric across datasets regardless of class imbalance or intrinsic task difficulty. For example, performance on IMDB (50% baseline, 95% SOTA) and SST-5 (20% baseline, 60% SOTA) are mapped to the same 0– 100 scale, despite the raw accuracies being on very different ranges. Crucially, this shared scale enables us to calculate the **average normalized score across all eight datasets** for each model (Figures 3 and 5), yielding a single interpretable performance figure that reflects its relative success across tasks of varying complexity. This averaging step is the central motivation for introducing the normalization, as it allows systematic comparison of models in a way that raw accuracy cannot.

9. Speed and Latency

We measured the average time per review (in seconds) and reviews per minute (RPM) after running the models through each of the 3 datasets and recorded the results (Tables 12 and 13). We also added the memory usage at a standard 4096 token context window for each of the models (Table 13). The best result in each category is highlighted in bold, though unsurprisingly the smallest model in each group is the fastest.

Table 12. Time per review (seconds) and Reviews per Minute for reviews at 30-token and 150-token ranges.

Model	Time(30t)	RPM(30t)	Time(150t)	RPM(150t)
Llama 3 8B Instruct	0.564	106.34	1.515	39.61
MiniCPM4 8B	0.674	89.00	1.829	32.80
IBM Granite 3.2 8B	0.584	102.67	1.866	32.16
OLMoE-1B-7B-0125	**0.233**	**257.38**	**0.382**	**157.08**
Gemma 3 4B	0.238	252.41	0.442	135.73
Phi-4 Mini Instruct	0.223	268.84	0.421	142.49
Llama 3.2 3B Instruct	**0.194**	**309.37**	**0.361**	**166.35**
Liquid LFM2 1.2B	0.158	380.74	0.231	260.08
Llama 3.2 1B Instruct	0.097	618.23	0.154	388.83
Gemma 3 1B QAT	**0.092**	**653.20**	**0.130**	**462.58**
MiniCPM4 0.5B	0.060	999.75	0.087	685.95
SmolLM2 360M Instruct	0.055	1091.38	0.076	788.47
ERNIE 4.5 0.3B	0.050	1192.06	0.073	827.57
SmolLM2 135M Instruct	**0.044**	**1370.73**	**0.055**	**1098.64**

Table 13. Time per review (seconds) and Reviews per Minute for reviews at 1200-token range and the memory footprint of the models (with 4096 token context window)

Model	Time(1200t)	RPM(1200t)	Memory footprint
Llama 3 8B Instruct	11.886	5.05	8.54 GB
MiniCPM4 8B	15.202	3.95	8.7 GB
IBM Granite 3.2 8B	15.383	3.90	8.68 GB
OLMoE-1B-7B-0125	**2.318**	**25.89**	**7.36 GB**
Gemma 3 4B	2.279	26.33	4.98 GB
Phi-4 Mini Instruct	2.293	26.17	4.08 GB
Llama 3.2 3B Instruct	**2.033**	**29.51**	**3.42 GB**
Liquid LFM2 1.2B	1.090	55.03	1.25 GB
Llama 3.2 1B Instruct	0.790	75.84	1.32 GB
Gemma 3 1B QAT	**0.569**	**105.42**	**720.43 MB**
MiniCPM4 0.5B	0.460	130.45	542.74 MB
SmolLM2 360M Instruct	0.449	133.55	386.4 MB
ERNIE 4.5 0.3B	0.359	167.21	385.79 MB
SmolLM2 135M Instruct	**0.244**	**246.12**	**144.81 MB**

We also construct graphs that visualize how gains in prompt processing speeds diminish beyond a certain threshold as model sizes are reduced (Figures 6 and 7). These visualizations make it clear that while smaller models initially provide substantial improvements in throughput, the rate of return gradually decreases, revealing a point at which further downsizing yields only marginal benefits.

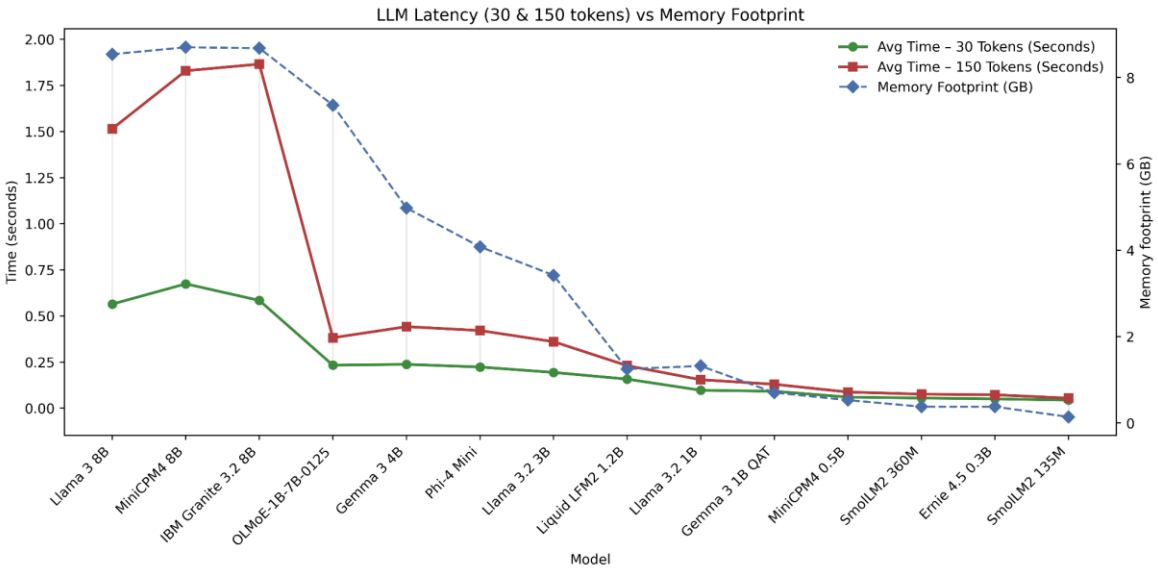


Fig. 6. Latency and Memory usage (30 tokens and 150 tokens)

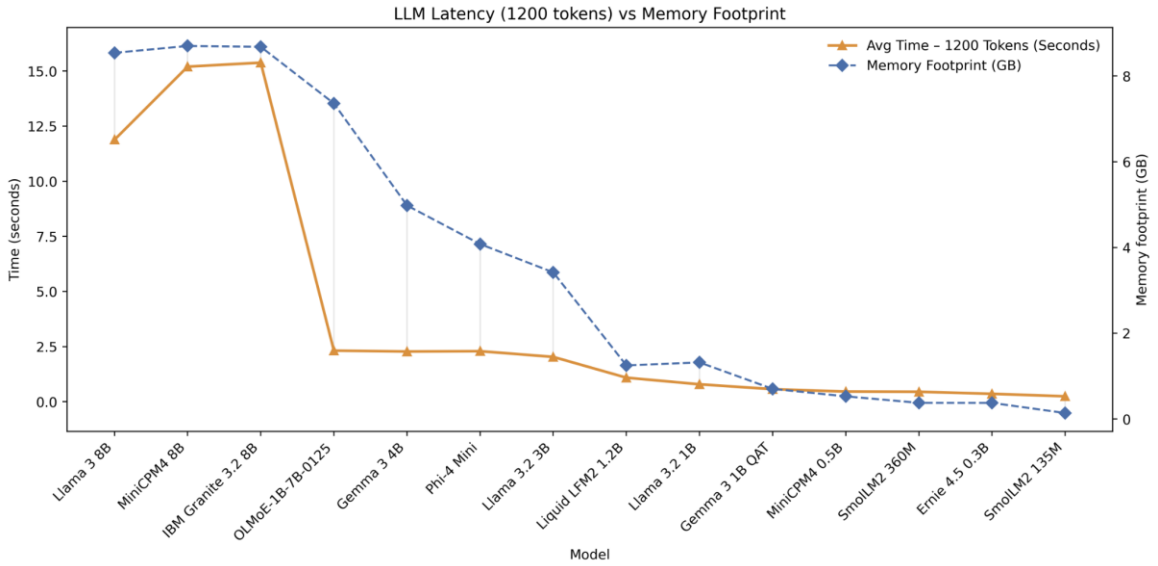


Fig. 7. Latency and Memory usage (1200 tokens)

The experimental results clearly demonstrate that inference latency is strongly determined by input length across all model sizes. For instance, Gemma-3 4B requires 0.238 s, 0.442 s, and 2.279 s to process inputs of approximately 30, 150, and 1200 tokens, respectively, confirming the near-linear growth of processing time with sequence length. At the same time, the relationship between latency and model size is less straightforward. While smaller models achieve faster processing speeds, the relative advantages diminish considerably once models fall below the 1B parameter range. As an illustration, SmolLM2-135M reaches 246 reviews per minute (RPM) at 1200 tokens, compared with 26.33 RPM for Gemma-3 4B, corresponding to a ≈ 9.3× improvement despite a ≈ 35× reduction in parameters. At shorter sequence lengths, this gap narrows further, with the speedup dropping to ≈ 5.4× at 30 tokens. These findings indicate that the practical gains from extremely small models are partially offset by their reduced accuracy, as shown in earlier evaluation sections.

The largest models tested (7–8B parameters) exhibit a pronounced increase in latency at long sequence lengths. At 1200 tokens, these models require approximately 12–15 s per review, compared with 2–2.3 s for 3–4B models and less than one second for 1B models. This disproportionate slowdown is explained by memory pressure: the observed memory footprints of ~8.5–8.7 GB exceed the 8 GB of dedicated VRAM available on the test hardware, leading to reliance on shared system memory and reduced throughput. Consequently, caution is warranted when selecting models that approach the memory capacity of the system, since longer prompts can easily push them over the threshold and result in severe performance degradation. Thus, performance in this regime arises not solely from parameter count, but also from hardware limitations.

An additional observation is that the Llama family exhibits slightly faster performance relative to other models in their respective parameter ranges. For example, Llama 3 8B Instruct processes 1200-token inputs more efficiently than Granite 3.2 8B or MiniCPM4 8B. This advantage likely arises from two factors: first, the llama.cpp framework is highly optimized for the Llama architecture, with kernels tuned to its tensor shapes and memory layouts; and second, the architecture itself is comparatively lean, employing rotary embeddings and streamlined transformer blocks that reduce computational overhead. Thus, the observed speedup reflects both implementation bias and architectural efficiency.

Architectural choices further modulate runtime behavior. The mixture-of-experts (MoE) model OLMoE-1B-7B constitutes a notable outlier, achieving inference times comparable to 3–4B dense models despite its nominal parameter scale. This result suggests that even for smaller models, MoE can greatly boost inference speeds without sacrificing accuracy.

In summary, the findings demonstrate four main points: (i) inference time scales predictably with input length across all models, (ii) speed advantages of smaller models diminish below the 1B range, (iii) hardware constraints and architectural design substantially affect throughput beyond what parameter counts alone would predict, and (iv) framework-level and architecture-specific optimizations, as seen with the Llama models, can produce consistent gains even within the same parameter class. These considerations are critical when selecting models for tasks dominated by either short or long documents, as they emphasize the need to balance model size, accuracy, and hardware efficiency in practical deployments.

10. Accuracy and Computational Cost Balance

We present a visualization that captures the trade-off between models' normalized accuracy on sentiment analysis tasks and their throughput when applied at scale (Figure 8). This representation allows for a direct comparison of how well models balance predictive performance against computational efficiency. To further clarify these trade-offs, we construct a Pareto Front (Figure 9), which highlights the most efficient model at each performance tier and makes explicit the frontier beyond which improvements in one dimension necessarily entail sacrifices in the other. In doing so, we illustrate how different architectures occupy distinct regions of the accuracy–throughput spectrum, offering insights into the scenarios where each model is most appropriately deployed.

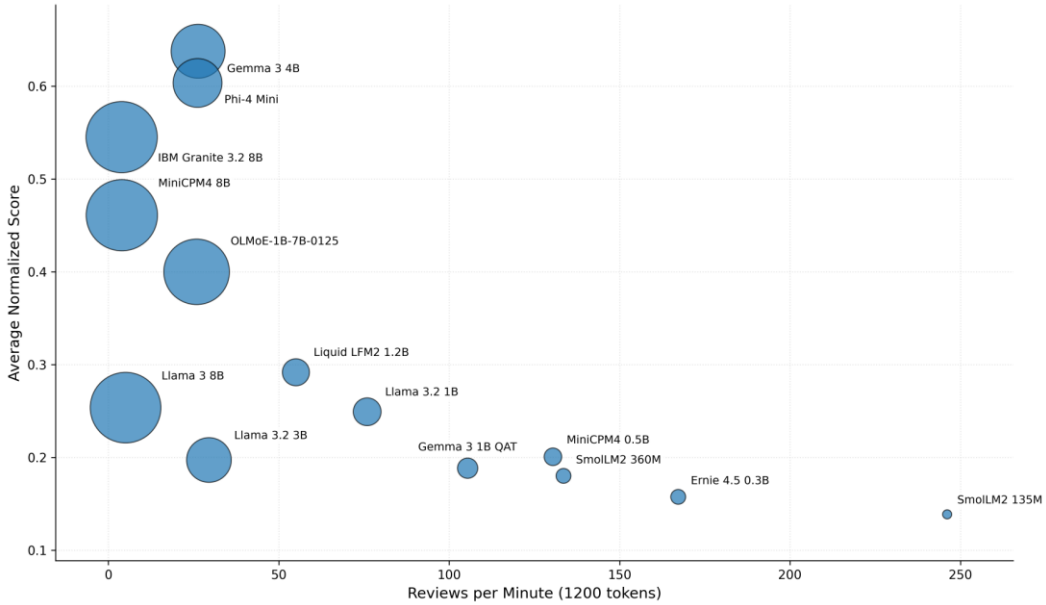


Fig. 8. Normalized Accuracy Score with no penalization vs Reviews per Minute for reviews at 1200 token length. Size of the bubbles correspond to the memory footprints of the models

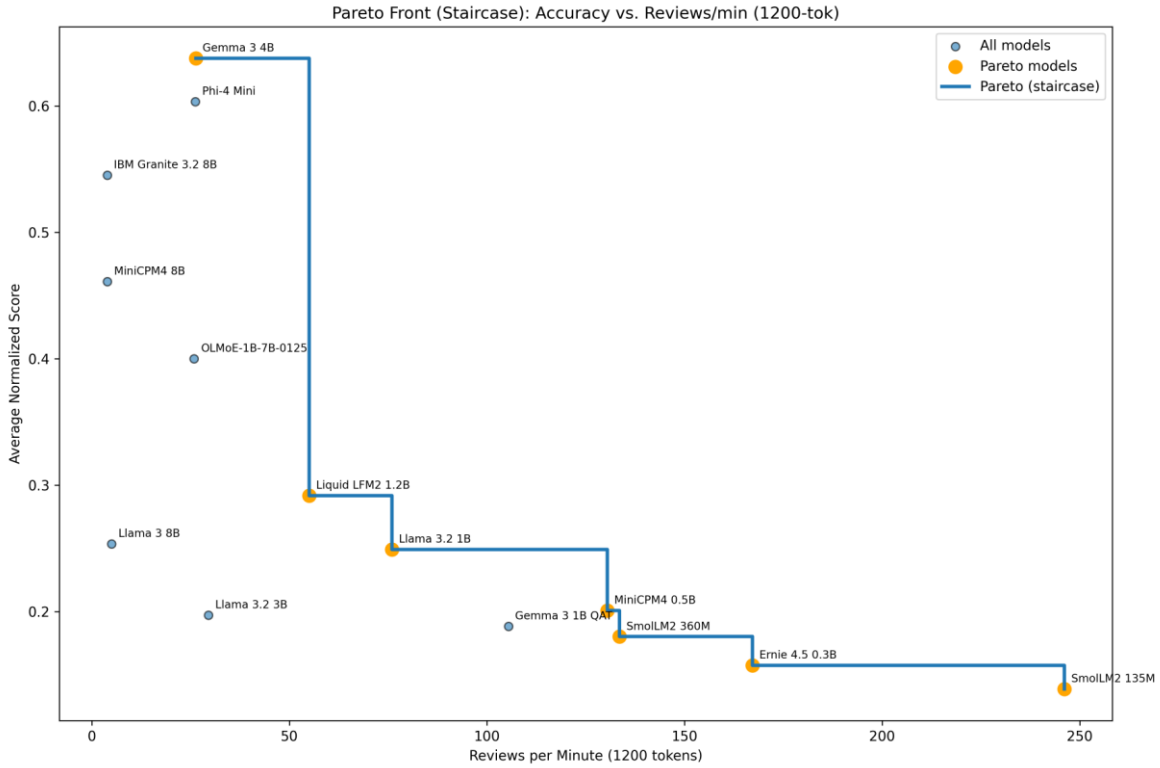


Fig. 9. Pareto Front of the best model at every latency target for zero-shot Sentiment Analysis

Across our zero-shot evaluations, mid-sized 3–4B models: especially Gemma 3 4B and Phi-4 Mini outperform the 7–8B group on our normalized accuracy and in the overall speed–accuracy trade-off. In practice, they deliver accuracy higher than their larger counterparts, while maintaining substantially lower latency and memory pressure, which makes them the most practical default choices for zero-shot sentiment analysis deployments with limited memory budget.

For applications where throughput and latency dominate and some loss in accuracy is acceptable, Liquid LFM2 (1.2B) and Llama 3.2 1B provide markedly faster inference and remain viable in pipelines tolerant of occasional misclassifications. These models are well suited to high-volume screening or interactive settings that must respond within tight time budgets.

When the workflow includes fine-tuning, extremely compact models such as MiniCPM4 0.5B and ERNIE 4.5 0.3B become compelling due to their speed and low memory footprint, enabling rapid iteration and frequent domain refreshes. In scenarios where a model is already fine-tuned for its deployment domain and only needs to execute prompt-based classification at scale, SmolLM2 135M offers the fastest prompt processing among the models we tested.

We also constructed Pareto Fronts for input lengths of 30 and 150 tokens (omitted from the main text). At 150 tokens, Microsoft Phi-4 Mini and OLMoE-1B-7B-0125 join the Pareto frontier; however, their end-to-end prompt ingestion speeds are not materially faster than Gemma 3 4B, so the 3–4B "sweet spot" remains intact unless the deployment context specifically favors those architectures.

11. Discussion

The results of this study highlight the complex balance between accuracy, efficiency, and usability when deploying compact large language models for sentiment analysis. While it is often assumed that larger models will invariably yield better performance, our findings demonstrate that this relationship is not straightforward. In particular, mid-sized models in the 3–4B parameter range, most notably Gemma 3 4B and Microsoft Phi-4 Mini emerged as the most practical defaults. They consistently outperformed some 7–8B models in normalized accuracy while offering substantially lower latency and memory consumption. This finding challenges the prevailing tendency to treat 7–8B models as the minimum viable baseline for serious sentiment analysis tasks.

By contrast, sub-1B models were unable to deliver reliable results in a zero-shot setting. Although they offered impressive speed and memory efficiency, their predictions frequently collapsed to a subset of labels or defaulted to majority classes. This behavior was especially pronounced on fine-grained datasets such as SST-5 and Emotions, where smaller models struggled to capture subtle distinctions between closely related categories. Nevertheless, these compact models retain potential in scenarios where fine-tuning is possible or where the deployment context prioritizes throughput over classification quality. In such cases, their lightweight nature enables frequent retraining, rapid iteration, and deployment on resource-limited hardware.

At the high end of the parameter spectrum, 7–8B models did achieve strong results on binary and domain-specific sentiment tasks, but their relative advantage was diminished once normalized against dataset-specific baselines. Furthermore, their high memory footprint and latency at longer input lengths raise practical concerns for real-world use. The observed slowdown was not solely a function of parameter count but also reflected hardware bottlenecks, particularly when GPU memory capacity was exceeded and inference spilled into shared system RAM. This underscores the importance of considering system-level constraints alongside raw model capabilities when selecting architectures for deployment.

Our evaluation also revealed instances where framework-level optimizations strongly influenced performance. For example, the Llama family consistently processed long inputs faster than

comparably sized alternatives, an advantage likely attributable to llama.cpp's architecture-specific tuning. Similarly, the OLMoE-1B-7B model benefited from its mixture-of-experts design, delivering inference speeds more typical of mid-sized dense models. These cases illustrate how runtime behavior cannot be attributed solely to model size or architecture but also depends on the surrounding software and inference stack.

From a practical standpoint, the trade-offs identified here point toward differentiated deployment strategies. For accuracy-sensitive applications that must still operate efficiently, mid-sized models appear to offer the best balance. For extremely high-throughput or edge-constrained scenarios, compact sub-1B models may still be viable if paired with task-specific fine-tuning. Conversely, when absolute accuracy is paramount and hardware resources are not a limiting factor, 7–8B models remain a robust, albeit less efficient, choice.

Finally, several limitations of this work must be acknowledged. First, all experiments were conducted in a zero-shot setting on English-language datasets; the extent to which these results generalize to multilingual contexts or fine-tuned workflows remains an open question. Second, the experiments were performed on a single hardware platform (Radeon 890M via Vulkan), and different backends or GPU architectures may yield different relative results. Third, only instruction-tuned non-reasoning variants were considered, leaving open the possibility that reasoning-augmented models may alter the trade-offs observed here. Addressing these limitations presents opportunities for future research, particularly in exploring cross-lingual sentiment tasks, incorporating fine-tuning strategies, and extending the evaluation to diverse hardware environments.

12. Conclusion

This paper presented a systematic benchmark of compact large language models, ranging from 135M to 8B parameters, on diverse sentiment analysis datasets. By normalizing results across tasks with different levels of difficulty and class imbalance, we demonstrated that performance does not scale linearly with model size. Instead, models in the 3–4B parameter range, particularly Gemma 3 4B and Microsoft Phi-4 Mini delivered the most favorable balance of accuracy, latency, and memory usage. These mid-sized architectures consistently outperformed or matched the results of larger 7–8B models while avoiding the prohibitive computational overhead that hinders deployment in practical settings.

At the same time, our analysis showed that sub-1B models remain limited in zero-shot conditions, often defaulting to trivial predictions. Nevertheless, their speed and minimal memory footprint make them promising candidates for scenarios involving fine-tuning or extremely resource-constrained deployment. Conversely, 7–8B models retain their value in cases where absolute accuracy is prioritized above efficiency, though their heavy computational requirements limit their accessibility for many real-world applications.

Taken together, these findings suggest that the "sweet spot" for zero-shot sentiment analysis lies in the mid-sized model category, offering a pragmatic compromise between usabil-ity and reliability. Future work should extend this benchmark to multilingual datasets, incorporate fine-tuning and reasoning-augmented variants, and evaluate performance across diverse hardware backends. Such efforts will be essential for understanding how compact models can be most effectively integrated into sentiment analysis pipelines, particularly in domains where efficiency, accessibility, and privacy are critical.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, pp. 5998–6008, 2017.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., "Language models are few-shot learners," Advances in neural information processing systems, vol. 33, pp. 1877–1901, 2020.

[3] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al., "Gpt-4 technical report," arXiv preprint arXiv:2303.08774, 2023.

[4] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for modern deep learning research," in Proceedings of the AAAI conference on artificial intelligence, vol. 34, pp. 13693–13696, 2020.

[5] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient fine-tuning of quantized LLMs," Advances in neural information processing systems, vol. 36, pp. 10088–10115, 2023.

[6] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in Proceedings of the ACL-02 conference on Empirical methods in natural language processing, pp. 79–86, Association for Computational Linguistics, 2002.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), pp. 4171–4186, 2019.

[8] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," arXiv preprint arXiv:1910.01108, 2019.

[9] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W.and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," Journal of machine learning research, vol. 21, no. 140, pp. 1–67, 2020.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in Proceedings of Workshop at ICLR, 2013.

[11] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543, 2014.

[12] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," arXiv preprint arXiv:1907.11692, 2019.

[13] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in International Conference on Learning Representations (ICLR), 2020.

[14] N. Thanh Pham, T. Kieu, D.-M. Nguyen, S. H. Xuan, N. Duong-Trung, and D. Le- Phuoc, "Slm-bench: A comprehensive benchmark of small language models on environmental impacts–extended version," arXiv e-prints, pp. arXiv–2508, 2025.

[15] Z.-H. Tan, Z.-C. Zhao, H.-Y. Shi, X.-Y. Zhang, P. Tan, Y. Yu, and Z.-H. Zhou, "Learnware of language models: Specialized small language models can do big," arXiv preprint arXiv:2505.13425, 2025.

[16] P. Lepagnol, T. Gerald, S. Ghannay, C. Servan, and S. Rosset, "Small language models are good too: An empirical study of zero-shot classification," arXiv preprint arXiv:2404.11122, 2024.

[17] L. Couto Seller, Í. Sanz Torres, A. Vogel-Fernández, C. González Carballo, P. M. Sánchez Sánchez, A. Carruana Martín, and E. d. M. Ambite, "Evaluating compact llms for zero-shot iberian language tasks on end-user devices," arXiv e-prints, pp. arXiv–2504, 2025.

[18] F. Koto, T. Beck, Z. Talat, I. Gurevych, and T. Baldwin, "Zero-shot sentiment analysis in low-resource languages using a multilingual sentiment lexicon," arXiv preprint arXiv:2402.02113, 2024.

[19] Y. Liu, X. Zhu, Z. Shen, Y. Liu, M. Li, Y. Chen, B. John, Z. Ma, T. Hu, Z. Li, et al., "Do large language models possess sensitive to sentiment?," arXiv preprint arXiv:2409.02370, 2024.

[20] T. Fawcett, "An introduction to ROC analysis," Pattern recognition letters, vol. 27, no. 8, pp. 861–874, 2006.

[21] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies, pp. 142– 150, 2011.

[22] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in Proceedings of the 2013 conference on empirical methods in natural language processing, pp. 1631–1642, 2013.

[23] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multitask benchmark and analysis platform for natural language understanding," arXiv preprint arXiv:1804.07461, 2018.

[24] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala, "Good debt or bad debt: Detecting semantic orientations in economic texts," Journal of the Association for Information Science and Technology, vol. 65, no. 4, pp. 782–796, 2014.

[25] Crowdflower, "Twitter us airline sentiment." https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment , 2015.

[26] E. Saravia, H.-C. T. Liu, Y.-H. Huang, J. Wu, and Y.-S. Chen, "Carer: Contextualized affect representations for emotion recognition," in Proceedings of the 2018 conference on empirical methods in natural language processing, pp. 3687–3697, 2018.

[27] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Predicting the type and target of offensive posts in social media," arXiv preprint arXiv:1902.09666, 2019.

[28] M. Saeidi, G. Bouchard, M. Liakata, and S. Riedel, "Sentihood: Targeted aspect based sentiment analysis dataset for urban neighbourhoods," arXiv preprint arXiv:1610.03771, 2016.

[29] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," Advances in neural information processing systems, vol. 32, 2019.

[30] G. Nkhata, S. Gauch, U. Anjum, and J. Zhan, "Fine-tuning bert with bidirectional LSTM for fine-grained movie reviews sentiment analysis," arXiv preprint arXiv:2502.20682, 2025.

[31] "Glue benchmark leaderboard." https://gluebenchmark.com/leaderboard/

[32] V. M., "Text-classification-financial-phrase-bank." https://github.com/vrunm/Text-Classification-Financial-Phrase-Bank , 2023.

[33] M. M. Rahman, A. I. Shiplu, Y. Watanobe, and M. A. Alam, "Roberta-bilstm: A context-aware hybrid model for sentiment analysis," IEEE Transactions on Emerging Topics in Computational Intelligence, 2025.

[34] J. Yan, P. Pu, and L. Jiang, "Emotion-rgc net: A novel approach for emotion recognition in social media using roberta and graph neural networks," Plos one, vol. 20, no. 3, p. e0318524, 2025.

[35] Z.Wu, H. Zheng, J.Wang, W. Su, and J. Fong, "Bnu-hkbu uic nlp team 2 at semeval- 2019 task 6: Detecting offensive language using bert model," in Proceedings of the 13th international workshop on semantic evaluation, pp. 551–555, 2019.

[36] Z. Wu and D. C. Ong, "Context-guided BERT for targeted aspect-based sentiment analysis," in Proceedings of the AAAI conference on artificial intelligence, vol. 35, pp. 14094–14102, 2021.

[37] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integerarithmetic- only inference," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2704–2713, 2018.

[38] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al., "Qwen3 technical report," arXiv preprint arXiv:2505.09388, 2025.

[39] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al., "The llama 3 herd of models," arXiv e-prints, pp. arXiv–2407, 2024.

[40] I. Granite Team, "Granite 3.0 language models," URL: https://github. com/ibmgranite/ granite-3.0-language-models, 2024.

[41] M. Team, C. Xiao, Y. Li, X. Han, Y. Bai, J. Cai, H. Chen, W. Chen, X. Cong, G. Cui, et al., "Minicpm4: Ultra-efficient LLMs on end devices," arXiv preprint arXiv:2506.07900, 2025.

[42] N. Muennighoff, L. Soldaini, D. Groeneveld, K. Lo, J. Morrison, S. Min, W. Shi, P. Walsh, O. Tafjord, N. Lambert, et al., "Olmoe: Open mixture-of-experts language models," arXiv preprint arXiv:2409.02060, 2024.

[43] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," Neural computation, vol. 3, no. 1, pp. 79–87, 1991.

[44] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin,T. Matejovicova, A. Ramé, M. Rivière, et al., "Gemma 3 technical report," arXiv preprint arXiv:2503.19786, 2025.

[45] A. Abouelenin, A. Ashfaq, A. Atkinson, H. Awadalla, N. Bach, J. Bao, A. Benhaim, M. Cai, V. Chaudhary, C. Chen, et al., "Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras," arXiv preprint arXiv:2503.01743, 2025.

[46] Liquid AI, "Introducing lfm2: The fastest on-device foundation models on the market,"

https://www.liquid.ai/blog/liquid-foundation-models-v2-our-second-series-of-generative-ai-models

[47] L. B. Allal, A. Lozhkov, E. Bakouch, G. M. Blázquez, G. Penedo, L. Tunstall, A. Marafioti, H. Kydlíček, A. P. Lajarín, V. Srivastav, et al., "Smollm2: When smol goes big–data-centric training of a small language model," arXiv preprint arXiv:2502.02737, 2025.

[48] B. .-E. Team, "Ernie 4.5 technical report." https://ernie.baidu.com/blog/publication/ERNIE_Technical_Report.pdf , 2025.

[49] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," arXiv preprint arXiv:2010.16061, 2020.

[50] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," Biochimica et Biophysica Acta (BBA)-Protein Structure, vol. 405, no. 2, pp. 442–451, 1975.

[51] J. Cohen, "A coefficient of agreement for nominal scales," Educational and psychological measurement, vol. 20, no. 1, pp. 37–46, 1960.

[52] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (RMSE) in assessing average model performance," Climate research, vol. 30, no. 1, pp. 79–82, 2005.

[53] A. Webson and E. Pavlick, "Do prompt-based models really understand the meaning of their prompts?," in Proceedings of the 2022 conference of the North American chapter of the association for computational linguistics: Human language technologies, pp. 2300–2344, 2022.

[54] F. Jiang, Z. Xu, L. Niu, B. Y. Lin, and R. Poovendran, "Chatbug: A common vulnerability of aligned LLMs induced by chat templates," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, pp. 27347–27355, 2025.

[55] K. Lyu, H. Zhao, X. Gu, D. Yu, A. Goyal, and S. Arora, "Keeping LLMs aligned after fine-tuning: The crucial role of prompt templates," Advances in Neural Information Processing Systems, vol. 37, pp. 118603–118631, 2024.

[56] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, et al., "Holistic evaluation of language models," arXiv preprint arXiv:2211.09110, 2022.

[57] Z. Zhao, E.Wallace, S. Feng, D. Klein, and S. Singh, "Calibrate before use: Improving few-shot performance of language models," in International conference on machine learning, pp. 12697–12706, PMLR, 2021.

[58] Y. Fei, Y. Hou, Z. Chen, and A. Bosselut, "Mitigating label biases for in-context learning," arXiv preprint arXiv:2305.19148, 2023.

INSTRUCTIONS FOR AUTHORS

1. "The Baku Engineering University Mathematics and Computer Science" accepts original unpublished articles and reviews in the research field of the author.
2. Articles are accepted in English.
3. File format should be compatible with **Microsoft Word** and must be sent to the electronic mail (**journal@beu.edu.az**) of the Journal. The submitted article should follow the following format:

- Article title, author's name and surname
- The name of workplace
- Mail address
- Abstract and key words

4. The title of the article should be in each of the three languages of the abstract and should be centred on the page and in bold capitals before each summary.
5. **The abstract** should be written in **9 point** type size, between **100** and **150** words. The abstract should be written in the language of the text and in two more languages given above. The abstracts of the article written in each of the three languages should correspond to one another. The keywords should be written in two more languages besides the language of the article and should be at least three words.
6. .**UDC** and **PACS** index should be used in the article.
7. The article must consist of the followings:

- Introduction
- Research method and research
- Discussion of research method and its results
- In case the reference is in Russian it must be given in the Latin alphabet with the original language shown in brackets.

8. **Figures, pictures, graphics and tables** must be of publishing quality and inside the text. Figures, pictures and graphics should be captioned underneath, tables should be captioned above.
9. **References** should be given in square brackets in the text and listed according to the order inside the text at the end of the article. In order to cite the same reference twice or more, the appropriate pages should be given while keeping the numerical order. For example: [7, p.15].

Information about each of the given references should be full, clear and accurate. The bibliographic description of the reference should be cited according to its type (monograph, textbook, scientific research paper and etc.) While citing to scientific research articles, materials of symposiums, conferences and other popular scientific events, the name of the article, lecture or paper should be given.

Samples:

a) ***Article:*** Demukhamedova S.D., Aliyeva İ.N., Godjayev N.M.. *Spatial and electronic structure af monomerrik and dimeric conapeetes of carnosine üith zinc,* Journal of structural Chemistry, Vol.51, No.5, p.824-832, 2010

b) ***Book:*** Christie ohn Geankoplis. *Transport Processes and Separation Process Principles*. Fourth Edition, Prentice Hall, p.386-398, 2002

c) ***Conference paper:*** Sadychov F.S., Aydın C., Ahmedov A.İ.. Appligation of Information – Commu-nication Technologies in Science and education. II International Conference."*Higher Twist Effects In Photon- Proton Collisions", Bakı, 01-03 Noyabr, 2007, ss 384-391*

References should be in 9-point type size.

10. The margins sizes of the page: - Top 2.8 cm. bottom 2.8 cm. left 2.5 cm, right 2.5 cm. The article main text should be written in Palatino Linotype 11 point type size single-spaced. Paragraph spacing should be 6 point.
11. The maximum number of pages for an article should not exceed 15 pages
12. The decision to publish a given article is made through the following procedures:

- The article is sent to at least to experts.
- The article is sent back to the author to make amendments upon the recommendations of referees.
- After author makes amendments upon the recommendations of referees the article can be sent for the publication by the Editorial Board of the journal.