Multiclass Classification of Skin Lesions using Neural Networks on DermaMNIST

José Miguel Norte de Matos

Department of Informatics Engineering of the University of Coimbra, Portugal
https://www.uc.pt/en/fctuc/dei/

Abstract. This work targets multiclass classification of 7 types of skin lesions using the low-resolution DermaMNIST dataset through exploration and optimization of Multi-Layer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Vision Transformers (ViT). Models were tuned using validation-based selection, focusing on Macro-F1 (MAF1) to handle class imbalance. Data augmentation was employed to mitigate overfitting. The optimized CNN achieved the best performance, with a test MAF1 of 0.5309 $\pm$ 0.0166 and Accuracy (ACC) of 0.7664 $\pm$ 0.0071. The ViT yielded the highest AUC (0.9514 $\pm$ 0.0017), while the MLP performed poorly (MAF1 0.3593 $\pm$ 0.0458). The custom CNN outperformed MLP and ViT on MAF1/ACC and proved competitive with benchmark models on this specific dataset, demonstrating the effectiveness of tailored CNNs even for low-resolution medical images.

Keywords: Deep Learning · Skin Lesion Classification · CNN · ViT · MLP.

1 Introduction

Automated classification of skin lesions using deep learning presents a significant opportunity for aiding clinical diagnosis, yet challenges remain, particularly concerning the visual similarity between different diseases, which can make the diagnosis difficult, even for medical experts.

This study investigates the comparative performance of three neural network architectures for multiclass skin lesion classification. Exploration and optimization of each architecture was done through hyperparameter tuning, employing techniques such as data augmentation to mitigate overfitting and validation-based selection focused on the Macro-averaged F1 Score (MAF1) to address inherent class imbalance. Further evaluation utilized Accuracy (ACC) for overall correctness and the Area Under the ROC Curve (AUC) to assess discriminative ability.

The aim is to determine the most suitable architecture for this specific medical image classification task and benchmark performance against established results [1] [4].

2 Problem description

This work takes advantage of the DermaMNIST dataset [1], a component of the MedMNIST v2 collection designed for the benchmark of lightweight image classification models. The data consists of 10,015 dermatostopic images derived from the HAM100000 dataset. Images were resized to 3x28x28 pixels, which facilitates experimentation and iteration, but on the other hand loses out on the detail of higher-resolution images.

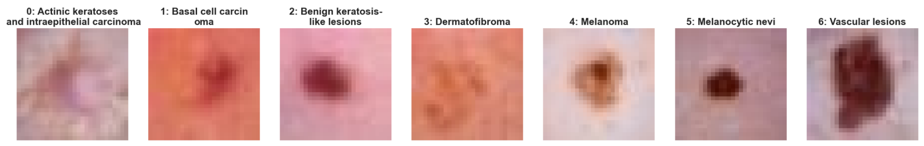

Fig. 1: Samples from each class of the DermaMNIST dataset in 28x28 resolution

The dataset is split into standardized training, validation, and test sets following a 7:1:2 ratio, respectively. The low image resolution necessitates models capable of learning effectively from limited spatial information. Furthermore, inherent class imbalance within medical datasets like this, requires careful metric selection for meaningful evaluation.

3 Methodology

3.1 Data Preprocessing & Handling

The pre-split DermaMNIST dataset is fed into the models using PyTorch DataLoader instances using a batch size of 128. Pixel values were standardized to the 0 to 1 range by dividing the values by 255. This addresses the issue of highly varying scales, allows for faster training and has been shown to improve generalization capabilities [2].

Data Augmentation (DA) On some trained models, augmentation transformations were done to the training split, with the final objective of reducing overfitting by increasing the diversity of the data. The chosen transformations were not very destructive and consisted of random vertical and horizontal flips, and small rotations (± 15 degrees) of the images. Non-destructive transformations retain the nature of medical imaging.

3.2 Models

Multi-Layer Perceptron (MLP) Feedforward network with fully connected layers. For image input and this task, it requires flattening the 3x28x28 pixel data into a single 2352 element vector. This process eliminates the inherent spatial structure of the image, potentially limiting performance in computer vision tasks.

Convolutional Neural Network (CNN) They use convolutional layers with learnable filters to automatically extract hierarchical features like edges and textures, combined with pooling layers often used to reduce dimensionality and increase robustness helping the network learn that the presence of a feature in a general area is more important than its exact pixel-perfect location. The features extracted from the raw image data are then passed through fully connected layers for classification.

Vision Transformer (ViT) ViTs apply a Transformer-like architecture to patches of the image, leveraging self-attention mechanisms to capture both local and global relationships within images. The outputs are then passed through a feedforward network.

3.3 Evaluation Metrics

Macro-averaged F1 Score (MAF1) Primary evaluation metric, especially during the hyperparameter tuning phase. Balances precision and recall for each class, not being influenced by the imbalance of the dataset. Has been used in related work with the original HAM10000 dataset [3].

Accuracy (ACC) Direct measure of overall correctness. Useful for comparison with the original MedMNIST v2 paper [1].

Area under ROC curve (AUC) In the final evaluation, ROC plots were analyzed and the AUC was computed. Micro-average AUC was used, focusing more on correct predictions rather than proper distinguishing between classes, contrary to MAF1. Allows for comparison with the dataset's paper [1].

4 Experiments

The first stage was to perform hyperparameter tuning on train and validation splits. Models were assessed mainly using MAF1. Every training had early stopping with 10 epoch patience. The best performing model was then chosen for the second stage of experimentation where they were then trained again on the training split, but this time evaluated on the unseen test split to check their generalization capability.

This choice to only use the test split at a final stage, comes from the fact that in a real-world setting, when performing hyperparameter tuning on a model, there is no information on future data that might be fed into it. This setup replicates that and avoids any information leakage between sets.

Experiments were repeated five times to account for stochasticity in the training process, such as that of weight initialization. Results reported are the mean of those five runs and the 95% confidence interval for each one.

4.1 MLP Experiments

The first step of experimentation was aimed at the layer configuration that better balanced performance and computational cost.

Experimentation was done with hidden layer configurations of varying sizes relative to the input (2352 elements), testing depths from 1 to 3 layers (Table 1), ensuring a smooth and progressive reduction of layer size up until the output layer. Each run had a maximum of 50 training epochs.

For the experiments regarding loss functions and optimizers, the maximum number of epochs was raised to 100 as they can behave differently over longer training durations. Testing these under a higher number of epochs ensures that their performance is evaluated after sufficient convergence.

Table 1: Results of layer configuration experiments with the MLP over 5 repetitions with CrossEntropyLoss and ADAM@0.005

Hidden Layers	**MAF1**	**ACC**	**TimeToTrain**
Exp.1 Layer Size			
(256,64)	0.2848 ± 0.0366	0.7099 ± 0.0050	$\mathbf{85.7314 \pm 2.4939}$
(1024,256)	$\mathbf{0.3557 \pm 0.0238}$	$\mathbf{0.7240 \pm 0.0069}$	154.6892 ± 2.3927
(4096,1024)	0.3528 ± 0.0390	0.7204 ± 0.0092	574.7653 ± 41.5736
Exp.2 Layer Depth			
(1024)	0.3003 ± 0.0269	0.7111 ± 0.0049	185.5934 ± 30.8837
(1024,256)*	$\mathbf{0.3557 \pm 0.0238}$	$\mathbf{0.7240 \pm 0.0069}$	$\mathbf{154.6892 \pm 2.3927}$
(1024,512,256)	0.3001 ± 0.0139	0.7129 ± 0.0057	178.3227 ± 31.3523

*Note: Chosen configuration taken from experiment 1.

Table 2: Results of loss function and optimizer experiments with the MLP over 5 repetitions.

Configuration	**MAF1**	**ACC**	**TimeToTrain**
Exp 1. Loss Function (Optimizer: ADAM@0.005)			
CrossEntropyLoss	$\mathbf{0.4028 \pm 0.0136}$	$\mathbf{0.7384 \pm 0.0042}$	325.5116 ± 28.1469
MultiMarginLoss	0.3472 ± 0.0300	0.7167 ± 0.0063	$\mathbf{288.4557 \pm 45.5772}$
CW CrossEntropyLoss	0.2729 ± 0.1858	0.4000 ± 0.2576	291.3144 ± 163.4039
CW MultiMarginLoss	0.2996 ± 0.1018	0.4173 ± 0.1823	331.4701 ± 19.6480
Exp 2. Optimizer (Loss: CrossEntropyLoss)			
ADAM (0.01)	0.3935 ± 0.0207	0.7306 ± 0.0089	313.4196 ± 52.7056
ADAM (0.005)*	$\mathbf{0.4028 \pm 0.0136}$	$\mathbf{0.7384 \pm 0.0042}$	325.5116 ± 28.1469
ADAM (0.001)	0.3667 ± 0.0172	0.7240 ± 0.0024	413.1132 ± 26.6667
RMSProp (0.01)	0.2149 ± 0.0842	0.6931 ± 0.0207	$\mathbf{282.1688 \pm 54.6735}$
RMSProp (0.005)	0.3063 ± 0.0343	0.7147 ± 0.0042	306.9326 ± 21.8260
RMSProp (0.001)	0.3237 ± 0.0331	0.7190 ± 0.0084	303.3869 ± 50.9419

*Note: Chosen configuration taken from experiment 1.

After analyzing the results on Table 2, a MLP with two hidden layers with 1024 and 256 neurons using CrossEntropyLoss and ADAM optimizer with a learning rate of 0.005 was picked for the next stage.

4.2 CNN Experiments

For experimentation with CNNs, the objective was to try and optimize a simple CNN architecture from scratch and see how it then performs against bigger and established architectures tested in other works, like ResNet-18 and ResNet-50 [1]. The simple baseline is described in Table 3. The fully-connected layers from the baseline CNN follow the best MLP layer configuration found in the MLP experiments (Table 1).

Table 3: Baseline CNN Architecture

Layer	Configuration (Output Shape)
Block 1	Conv(3, 64, 3×3, p=1) $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ MaxPool(2×2) ($32 \times 14 \times 14$)
Block 2	Conv(64, 128, 3×3, p=1) $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ MaxPool(2×2) ($64 \times 7 \times 7$)
Classifier	Flatten $\rightarrow$ FC(3136, 1024) $\rightarrow$ ReLU $\rightarrow$ D(0.5) $\rightarrow$ FC(1024, 256) $\rightarrow$ ReLU $\rightarrow$ D(0.5) $\rightarrow$ FC(256, 7)

After analyzing the results from the baseline experimentation, which are reported in Table 4 and Figure 2, the regularization provided by the data augmentation (DA) is clear by looking at the loss curves. After preventing overfitting with the use of DA, performance took a hit, so the dropout rate was reduced as 0.5 might have been too aggressive when combined with data augmentation. Layer configuration, loss function and optimizer experiments followed and are reported in Table 8 and Table 6.

Table 4: Effects of data augmentation and dropout on baseline CNN

Configuration	MAF1	ACC	TimeToTrain (s)
Baseline	0.4513 ± 0.0247	0.7577 ± 0.0056	52.8717 ± 12.6322
Exp.1 Effect of Data Augmentation on Baseline CNN			
Baseline + DA	0.3921 ± 0.0454	0.7420 ± 0.0160	223.2340 ± 89.0770
Exp.2 Dropout Rate (using Baseline + DA)			
Dropout=0.0	$\mathbf{0.4788 \pm 0.0423}$	0.7627 ± 0.0111	$\mathbf{222.8341 \pm 86.7411}$
Dropout=0.2	0.4718 ± 0.0461	$\mathbf{0.7673 \pm 0.0122}$	301.8048 ± 84.7763

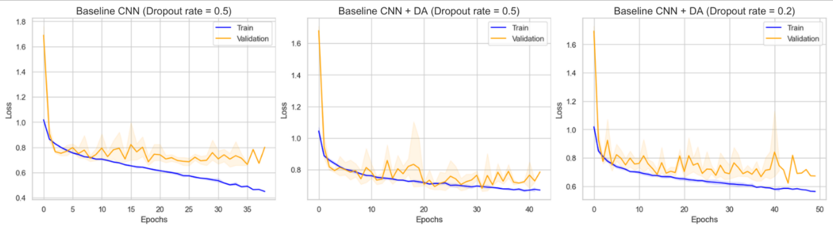

Fig. 2: Effect of DA and different dropout rates on the loss curves of the Baseline CNN. Average loss values at each epoch after 5 repetitions.

Table 5: CNN Network Architecture Experiments (Baseline: DA + Dropout=0.2)

Configuration	MAF1	ACC	TimeToTrain (s)
Baseline	0.4718 ± 0.0461	0.7673 ± 0.0122	301.8048 ± 84.7763
Deeper (3 Conv Layers)	0.4769 ± 0.0559	0.7653 ± 0.0188	$\mathbf{223.0539 \pm 95.3618}$
Wider (Double Filters)	0.4528 ± 0.0209	0.7641 ± 0.0088	261.2127 ± 48.1556
Shallow Classifier (1024,)	$\mathbf{0.4847 \pm 0.0296}$	$\mathbf{0.7711 \pm 0.0094}$	227.1772 ± 65.5171

The chosen configuration for the next stage was the baseline CNN layer configuration, having only one hidden layer in the classifier, a 0.2 dropout rate

Table 6: CNN Loss Function and Optimizer Experiments (Baseline: Aug + D=0.2 + 2 Conv Layers)

Configuration	**MAF1**	**ACC**	**TimeToTrain (s)**
Loss Function Experiments (Optimizer: ADAM@0.001)			
CrossEntropyLoss*	$\mathbf{0.4847 \pm 0.0296}$	$\mathbf{0.7711 \pm 0.0094}$	227.1772 ± 65.5171
MultiMarginLoss	0.4356 ± 0.0241	0.7601 ± 0.0091	244.6889 ± 73.7341
CW CrossEntropyLoss	0.3775 ± 0.0175	0.5557 ± 0.0364	$\mathbf{183.6612 \pm 91.6026}$
Optimizer & Learning Rate Experiments (Loss: CrossEntropyLoss)			
ADAM (LR=0.001)*	0.4847 ± 0.0296	0.7673 ± 0.0122	$\mathbf{227.1772 \pm 65.5171}$
RMSProp (LR=0.001)	0.4523 ± 0.0371	0.7603 ± 0.0130	321.5652 ± 97.6051
ADAM (LR=0.0005)	$\mathbf{0.4994 \pm 0.0347}$	$\mathbf{0.7745 \pm 0.0125}$	260.5072 ± 77.8942

and data augmentation, unweighted CrossEntropyLoss, and ADAM with 0.0005 learning rate.

4.3 ViT Experiments

The ViT architecture explored is derivated from the Base-ViT [5], with reduced size and depth due to computational constraints. The specific configuration used as a starting point was patch size 4, embeddings with 128 dimensions, 8 transformer layers, 8 attention heads, MLP with dimension 1024, and no dropout. For this model, fewer experiments were performed and are reported in Table 7.

Table 7: ViT Hyperparameter Tuning Experiments over 5 repetitions.

Configuration	**MAF1**	**ACC**	**TimeToTrain (s)**
Exp.1 Regularization (ADAM@0.0005, CE Loss)			
Baseline*	0.4521 ± 0.0470	0.7460 ± 0.0077	$\mathbf{475.2340 \pm 147.9349}$
Drop=0.2	$\mathbf{0.4774 \pm 0.0500}$	$\mathbf{0.7587 \pm 0.0107}$	580.8495 ± 160.9610
Drop=0.2+Weight Decay	0.4518 ± 0.0188	0.7480 ± 0.0071	638.9166 ± 131.4393
Exp.2 Architecture (Dropout=0.2, ADAM@0.0005, CE Loss)			
Baseline (dim=128)*	$\mathbf{0.4774 \pm 0.0500}$	$\mathbf{0.7587 \pm 0.0107}$	$\mathbf{580.8495 \pm 160.9610}$
Bigger Emb. (dim=256)	0.4692 ± 0.0394	0.7517 ± 0.0056	805.8573 ± 152.0638
Exp.3 Loss Function & Optimizer (Dropout=0.2, ADAM@0.0005, CE Loss)			
Baseline	$\mathbf{0.4774 \pm 0.0500}$	$\mathbf{0.7587 \pm 0.0107}$	580.8495 ± 160.9610
MultiMarginLoss	0.4145 ± 0.0362	0.7380 ± 0.0105	637.0704 ± 153.6314
RMSProp (LR=0.0005)	0.4350 ± 0.0190	0.7402 ± 0.0083	724.8945 ± 68.5015
ADAM (LR=0.001)	0.4291 ± 0.0242	0.7400 ± 0.0086	$\mathbf{577.7388 \pm 154.9098}$

*Note: Was overfitting, so regularization was introduced.

5 Final Results & Discussion

The final performance of the optimized MLP, CNN, and ViT models chosen from the experiments was evaluated on the unseen DermaMNIST test set. All models were trained for 50 epochs on the training set, and the model corresponding to the epoch with the lowest validation loss during the training phase was selected for final evaluation. The average results over 5 independent runs, along with 95% confidence intervals, are presented in Table 8 and visualized partially in the ROC curves on Figure 3.

Table 8: Final model's evaluation on the test set.

Model	**MAF1**	**AUC**	**ACC**
MLP	0.3593 ± 0.0458	0.9438 ± 0.0016	0.7165 ± 0.0046
CNN	$\mathbf{0.5309 \pm 0.0166}$	0.9477 ± 0.0046	$\mathbf{0.7664 \pm 0.0071}$
ViT	0.4804 ± 0.0396	$\mathbf{0.9514 \pm 0.0017}$	0.7500 ± 0.0052

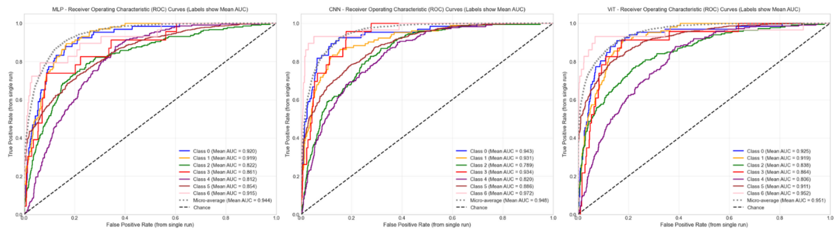

Fig. 3: ROC curves generated by the proposed models on the last repetition of evaluation on the DermaMNIST test split. Values in the label are the average of the 5 repetitions.(*Left to right:* MLP, CNN, ViT)

The CNN achieved the highest MAF1 at 0.5309 and the highest accuracy at 0.7664, making it the best performing model according to the primary evaluation metric chosen to handle class imbalance, and overall classification correctness. A superior MAF1 suggests it may have a better balance of precision and recall across all classes, crucial given this dataset. The ViT's slightly lower MAF1 compared to the CNN could indicate it struggled slightly more with certain minority classes despite its excellent overall AUC. The tighter confidence interval for the CNN's MAF1 score also suggests more stable performance across runs.

The ViT achieved the highest AUC at 0.9514, marginally surpassing the CNN's 0.9477, but didn't perform so well in other metrics, and could not beat the CNN. When compared to an article where the author employed the original ViT-Base architecture [5] pre-trained on 14 million images, his solution achieves 0.942 AUC and 0.8 accuracy on DermaMNIST [4], and while the custom ViT is able to beat it in AUC, it falls short on accuracy, which suggests pre-training and deeper architectures have an important role it's performance.

As expected, the MLP struggled significantly, with an MAF1 of 0.3593. Although its AUC appears relatively high, this can be misleading in imbalanced datasets when not considering class-specific performance. the low MAF1 confirms its weakness in handling the different classes effectively compared to the spatially-aware models.

Comparing results to the benchmarks in the original MedMNIST v2 paper [1] provides valuable context in the task. Taking a look at the results in this work, every model surpasses the highest AUC obtained in the original paper: 0.920 using ResNet-18 with 224x224 resized images. The highest accuracy reported in the paper was 0.768, obtained by Google AutoML Vision. The CNN almost beats it at 0.766, while obtaining a significantly higher AUC.

Interestingly, the custom CNN results beat the competing CNN architectures from the original benchmark: ResNet-18 and ResNet-50, both with 28x28 and 224x224 images. This suggests the relatively simple CNN architecture achieved better performance than much deeper and standardized architectures, on this specific low-resolution dataset.

6 Conclusion

This study evaluated the performance of optimized MLP, CNN, and ViT architectures for multiclass skin lesion classification on the low-resolution DermaMNIST dataset. The evaluation framework was designed to ensure a correct experimental analysis, rigorous hyperparameter tuning confined to the validation set, and reporting averaged results over five independent runs to account for stochasticity.

By looking at the results, a clear performance hierarchy is revealed. The hyperparameter-tuned CNN appears as the best alternative and demonstrates significant advantages over the others, mainly achieving the highest balanced classification capability (MAF1) and overall correctness (ACC), surpassing results reported for standard ResNet benchmarks on the same 28x28 dataset [1], providing strong evidence that an optimized, relatively simpler architecture can outperform deeper, generic models in constrained data scenarios like low-resolution images.

The findings validate the CNN's effectiveness for this specific problem, highlighting the relationship between model architecture, data characteristics, metric selection, and hyperparameter optimization in achieving robust and reliable classification performance in medical image analysis.

Acknowledgements

The code for the plots was developed with the help of Claude Sonnet 3.7.

References

1. Yang, J. et al: MedMNIST v2 - A large-scale lightweight benchmark for 2D and 3D biomedical image classification. Sci Data. 10, 41 (2023). https://doi.org/10.1038/s41597-022-01721-8.
2. The European Information Technologies Academy, E.: Why is it necessary to normalize the pixel values before training the model?, https://tinyurl.com/4vuab2cz, last accessed 2025/03/31.
3. Mietkiewicz, L., Ciechanowski, L., Jemielniak, D.: The Skin Game: Revolutionizing Standards for AI Dermatology Model Comparison (2025). https://doi.org/10.48550/arXiv.2502.02500
4. Beierle, F.: Medical Image Classification with Vision Transformers, https://beierle.de/2024/01/medical-image-classification-with-vision-transformers/, last accessed 2025/04/06.
5. Dosovitskiy, A. et al: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, http://arxiv.org/abs/2010.11929, (2021). https://doi.org/10.48550/arXiv.2010.11929.