CS-233 MS2 Report

362578 Clea Maisonnier | 375535 Sam Lee | 379094 Yahan Zhang

1. Introduction

In this project, our goal is to classify skin lesion images into one of seven diagnostic categories using the DermaMNIST dataset. The dataset consists of 9,012 dermoscopic images resized to 28x28 RGB format, with 7,007 images for training and 2,005 for testing. We evaluate two deep learning models, MLP and CNN implemented in PyTorch. To account for class imbalance, we report both Accuracy and Macro F1-Score as evaluation metrics.

2. Method

a. Data Preprocessing

- **Normalization**: All pixel values were scaled to the range [0, 1] by dividing by 255. This step helps neural networks converge more efficiently during training.

- **Channel reordering**: Since PyTorch expects input images in the (C, H, W) format, we transposed the input arrays from (N, 28, 28, 3) to (N, 3, 28, 28).

- **Train/Validation split**: Without the --test flag, we randomly split the training set into 80% training and 20% validation, strictly excluding validation data from training. Given the low resolution of the 28x28 images, we avoided applying aggressive data augmentations, as they could introduce noise or remove informative patterns.

b. MLP (Multilayer Perceptron)

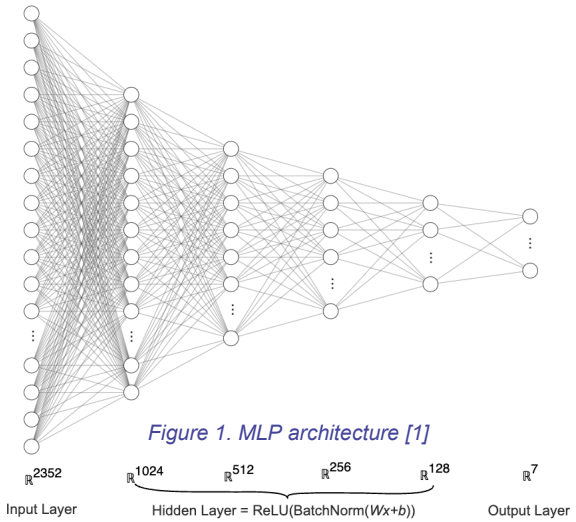

Figure 1. MLP architecture [1]

As seen in *Figure 1*, our architecture consists of 4 hidden layers, each followed by a ReLU activation and Batch Normalization to stabilize training. Given the dataset's size, we selected a 4-layer configuration (among 3–5 tested, their validation accuracy were: ~64-70% for 3, ~68-72% for 4, ~66-70% for 5) to minimize underfitting and overfitting. The default hidden layer size array [1024, 512, 256, 128] was chosen to gradually reduce dimensionality and compress features progressively.

b. MLP-Mixer

We investigated improved architectures for MLP-based image classification and implemented a simplified version of MLP-Mixer, a purely MLP-based model introduced by Google Research in 2021. [2] Unlike traditional convolutional networks, MLP-Mixer separates spatial (token) and channel-wise mixing using stacked mixer blocks composed only of linear layers and non-linear activations.

Our implementation divides images into non-overlapping patches and applies alternating token-mixing and channel-mixing MLPs after linear embedding, followed by global average pooling and classification. This design allows the model to capture both local and global dependencies without convolutional operations.

c. CNN (Convolutional Neural Network)

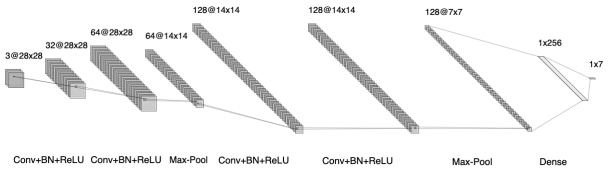

Figure 2. CNN architecture [3]

The model consists of four convolutional layers, each followed by Batch Normalization and a ReLU activation. The second and fourth convolutional layers are followed by MaxPool2d operations, effectively forming spatial downsampling. Initially, we experimented with a simpler architecture with fewer layers, but the resulting confusion matrix showed poor separability for minority classes and low F1-scores. So we added deeper convolutional layers and Batch Normalization. After the convolutional feature extractor, the network includes a fully connected classifier with a Flatten layer, one hidden dense layer, and an output layer. This final architecture is illustrated in *Figure 2*.

d. Training Procedure

We implemented early stopping based on validation macro F1-score to prevent overfitting. Training terminates when no improvement is observed for the 10 consecutive epochs (early_stop_patience).

To address the class imbalance inherent in the dataset, we used weighted CrossEntropyLoss with class weights inversely proportional to class frequencies. This ensures that minority classes contribute more to the total loss, helping the model to learn from underrepresented categories.

e. Implementation Details

To support faster training, we implemented device-agnostic model execution. Our code supports CPU, CUDA (GPU), and Apple MPS backends via the --device flag. All tensors and model components are explicitly transferred to the selected device. Experiments were run using the MPS backend on macOS, with fixed random seeds for reproducibility.

3. Experiment/Results

We selected the final models based on their validation performance during a sweep over learning rates. To avoid overfitting, we excluded configurations where training accuracy exceeded 99% but validation performance showed only marginal improvements.

	lr	Training Acc	Training F1	Validation Acc	Validation F1
MLP	1e−3	94.648 %	0.917603	68.830 %	0.461519
	1e−4	92.079 %	0.911708	69.330%	0.483847
	1e−5	97.288 %	0.966881	72.040 %	0.512035
MLP-Mixer	**1e−3**	95.772%	0.938857	69.472%	0.444795
	1e−4	91.472%	0.902656	68.902%	0.440765
	1e−5	86.994%	0.887439	65.549%	0.409798
CNN	1e−3	96.735%	0.967595	72.397%	0.554126
	1e−4	98.162%	0.967314	75.606%	0.550969
	1e−5	100.000%	1.000000	76.106%	0.56198

Table 1. Performance across different learning rates

While the MLP-Mixer did not demonstrate a significant performance improvement on this dataset, likely due to the relatively low input resolution and limited sample size, it was nevertheless interesting to explore this architecture. Its ability to model both spatial and channel-wise interactions without convolution or attention remains a promising direction for future work. However, due to its inferior generalization and higher overfitting tendency during testing, it was excluded from further discussion.

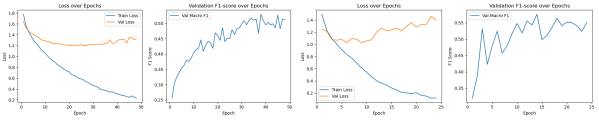

Figure 3. Loss and Validation over Epochs (MLP - CNN)

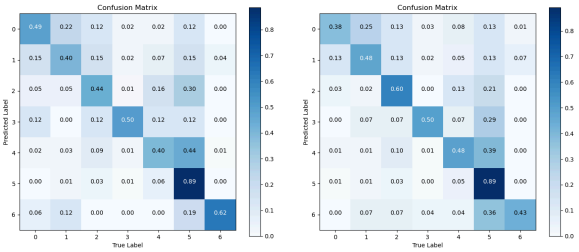

Figure 4. Confusion Matrix of CNN and MLP

Note that in both models, validation loss begins to rise after a certain point, triggering early stopping at epoch 47 for MLP and 23 for CNN as illustrated in *Figure 3*.

4. Discussion/Conclusion

Method	MLP	CNN
Training Accuracy	99.129%	99.872%
Training F1	0.987842	0.994540
Test Set Accuracy	71.771%	74.663%
Test Set F1	0.486196	0.545667
Training Time	146.29 s	177.71 s
Inference Time	0.14 s	0.19 s
Total Parameters	3.1M	1.8M

Table 2. Evaluation results on the test set

- **Accuracy :** CNN achieved superior performance by effectively capturing spatial features through convolutional operations. Unlike MLP-based models that treat pixels independently, this was particularly advantageous for the DermaMNIST dataset, where local texture, lesion boundaries, and color patterns play a crucial role in differentiating between diagnostic classes.

- **Time Efficiency** : Despite its higher accuracy, the CNN took longer to train. Interestingly, the MLP—despite having more parameters—completed training more quickly. This suggests that while convolutional layers improve representational power, they also introduce computational overhead per epoch. In this case, the MLP was more efficient, although at the cost of lower generalization performance.

Reference

[1] Figure generated using NN-SVG by Alex Lenail, with minor manual edits to the SVG.
[2] Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., ... & Dosovitskiy, A. (2021). *MLP-Mixer: An all-MLP Architecture for Vision*. Retrieved from https://research.google/pubs/mlp-mixer-an-all-mlp-architecture-for-vision/
[3] Figure generated using NN-SVG by Alex Lenail, with minor manual edits to the SVG.