COMP 551 Assignment 3: Classification of Image Data with Multilayer Perceptrons and Convolutional Neural Networks

Group 44: Ayaz Ciplak - 261013785, Christopher Smith - 261013926, Jacqueline Silver - 261012002

Abstract

This project explores the implementation and analysis of multilayer perceptrons (MLPs) for image classification, focusing on the OrganAMNIST dataset from MedMNIST. The project aims to provide hands-on experience with building a neural network from scratch, implementing backpropagation for gradient computation, and optimizing using mini-batch gradient descent. Experiments compared the performance of different MLP architectures, including variations in activation functions, network depth, and regularization methods. Additionally, convolutional neural networks (CNNs) and a pre-trained ResNet18 were implemented, comparing their performances with MLPs and evaluating the effects of increased image resolution. These experiments highlighted the impact of architectural choices and preprocessing techniques on model accuracy and training efficiency. Through this, we gained insights into how neural networks can be tuned for optimal performance in classification tasks.

Introduction

Multilayer perceptrons (MLPs) are fundamental tools in machine learning, particularly for tasks involving modelling complex, non-linear relationships. (Jerga) MLPs consist of an input layer, one or more hidden layers, and an output layer, with activation functions applied between layers to enable non-linear transformations. This project involves implementing a MLP from scratch, focusing on the OrganAMNIST dataset, which contains abdominal CT scan images categorized into 11 classes. The MLP was trained using backpropagation and mini-batch gradient descent, and was optimized using techniques like regularization and data normalization.

In addition to exploring MLPs, our project investigates the performance of convolutional neural networks (CNNs) and pre-trained ResNet models on the same dataset. Key experiments included comparing MLPs with varying depths and activation functions, examining the effects of regularization, and analysing the impact of image resolution (i.e. input data complexity) on CNNs. These experiments provided a deeper understanding of how design decisions affect neural network performance. By systematically comparing MLPs, CNNs and pre-trained models, we aimed to uncover insights into optimizing accuracy and efficiency in medical image classification.

Dataset

OrganAMNIST

The dataset used in this project is OrganAMNIST, a multi-class subset of the MedMNIST dataset consisting of abdominal CT scan images categorized into 11 distinct organ classes. It comprises 58,830 images with a predefined split of 34,561 training samples, 6,491 validation samples, and 17,778 test samples (Yang). Most experiments utilize the 28×28-pixel version of the dataset; however, we also examine the effects of higher image resolution using the 128×128-pixel variant, which introduces additional computational complexity and parameters to the models.

Preprocessing

Before training, the images and labels undergo several preprocessing steps to prepare them for the models. The raw image data, structured as [num-samples x channels x rows x columns], is flattened to [num-samles x pixels]. Since the dataset only includes black and white images and therefore a singular channel, the flattening is done by simply flattening the [row x column] pixel data to a [pixels] vector for each data point. Labels are then converted into a one-hot encoded format, transforming each class label into a binary array where only the index corresponding to the correct class is marked as 1. The images are normalized to ensure consistent input ranges (and a more stable training process), and the data is split into the provided training, validation and test sets. To facilitate efficient training, a DataLoader is initialized to batch the data and manage input-output subsets during experiments.

Exploratory Analysis

A brief exploratory analysis of the dataset revealed that the class distribution across the training set is relatively balanced, with all 11 organ categories represented. This characteristic reduces the risk of class imbalance issues

during training. Additionally, visualizing samples confirmed the variability in image clarity and organ appearance, which poses a realistic challenge for model classification tasks.

Ethical Considerations

Since OrganAMNIST images originate from real medical records, ethical concerns must be addressed. These include safeguarding patient privacy to ensure anonymity and minimizing biases that may arise from underrepresentation of certain demographic groups within the dataset. The dataset documentation notes compliance with ethical guidelines for medical data usage (Yang). However, researchers must remain cautious of potential biases and limitations when interpreting results.

Results

Task 2 - MLP Model Classes

In task 2, we implemented two MLP classes, "MLP" and "ClassificationMLP", as well as a GradientDescent optimizer class. Both MLP classes contain the functions fit(), fit.gradient(), predict() and evaluate-accuracy(). They are both initialized with parameters such as num-hidden-layers, num-hidden-units (per layer), hidden-activation-functions, and verbose-mode. The ClassificationMLP model is optimized for classification tasks and hardcodes a softmax output activation function, while the regular MLP model takes in this parameter as an additional variable. The GradientDescent class configures hyperparameters like the learning rate, max-iters, and epsilon.

Task 3 - Experiments:

3.1: Nonlinearity and Network Depth Comparison

In experiment 3.1, we observed how MLP depth affects classification accuracy. We created three models: one with no hidden layers, with one hidden layer, and with two hidden layers; all using ReLU activations and 256 hidden units per layer. On all models, an increase in iterations led to an increase in train performance. However, we also see that in all models, increasing the amount of iterations past a certain threshold leads to overfitting and therefore a decline in validation error. We can also observe the effects of nonlinearity – The linear model struggles to completely fit all classes properly, and reaches a maximal training error of 85 percent. However, the nonlinear models are able to completely fit all of the training data.

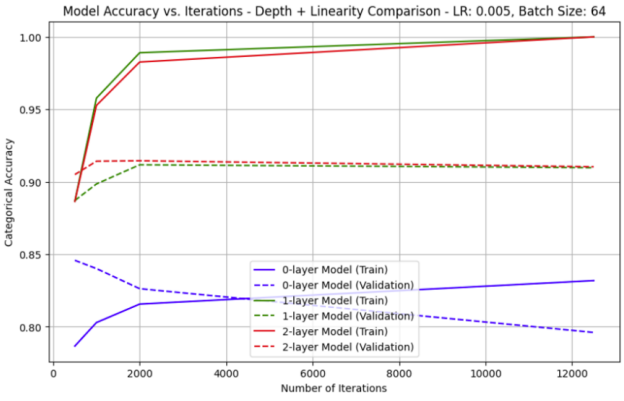

Figure 1: Nonlinearity and Depth Comparison

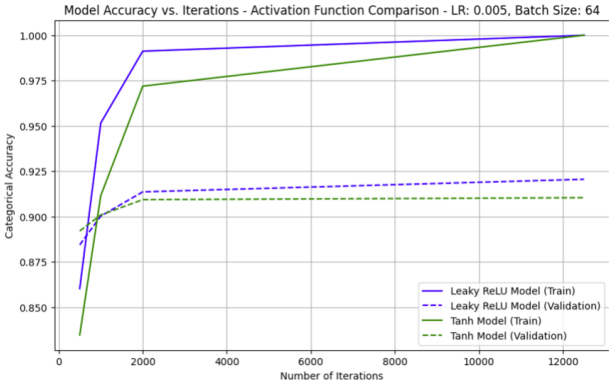

Figure 2: Activation Function Comparison

3.2: Activation Function Comparison: Tanh vs. Leaky-ReLU

In task 3.2, we explored the impact of different activation functions on our MLP models. We observe two trends. Firstly, Leaky ReLU has better train and validation accuracy with the same number of iterations (faster convergence). This can be explained by the fact that tanh only outputs values between -1 and 1, which slows down the learning process since the model cannot make significant updates to the weights. Second of all, Tanh's validation accuracy is slightly worse even when the train performance is the same. This can be explained by the fact that leaky ReLU can produce sparser representations (i.e. many neurons output 0), which can act as regularization (improving generalization).

3.3: Regularization Implementation and Analysis (L1 & L2)

In 3.3, we implemented L1 and L2 regularization within our MLP model. We observe very similar performance on both train and test sets, even with varying L1 and L2 regularization values. However, there is some evidence of a slight performance increase on the validation set, proportional to the amount of regularization. These similar performances can be explained by two things: Firstly, the underlying structure of our model: Our MLP model already has a large amount of trainable parameters, given the large amounts of hidden units. This means the

model can already fit the data's underlying distribution easily, resulting in low effects from regularization. Second of all, the Dataset Size: We are already training on a relatively large dataset – The optimizer we are using makes 1000 iterations on batches of size 64, meaning that it is already influenced by a large part of our train set (of size ¿ 30 000). Since a higher dataset size already leads to better generalization capabilities, the model is already able to generalize well on the validation set.

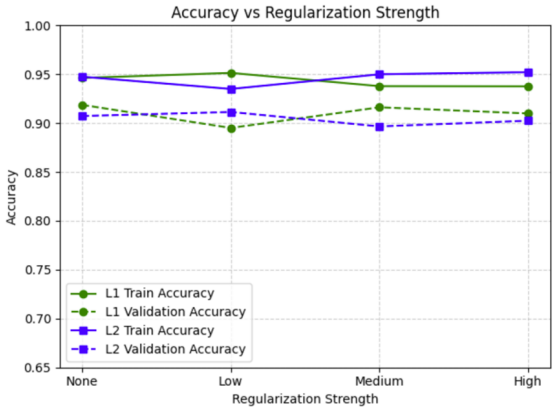

Figure 3: Regularization Implementation and Analysis

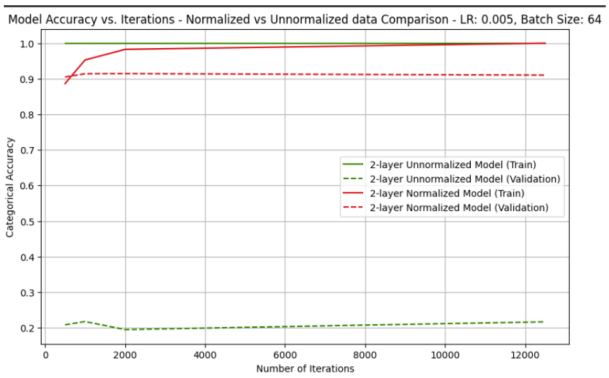

Figure 4: Data Normalization Comparison

3.4: Data Normalization Comparison

In experiment 3.4, we trained the model on unnormalized input data and analysed the effects. We notice two things: 1. Normalized data performs significanly better on validation set: This is because normalization leads to gradient stability, ensuring gradient updates are more stable, as well as the unnormalized model overfitting as per the point below. 2. Unnormalized data has consistently high train accuracy: This is due to the fact that unnormalized features could exxagerate the separability of classes in the training dataset and exploit dominant features in the data.

3.5: High Resolution Data Comparison

In task 3.5, we observed how image resolution plays a role in the training of an MLP model and evaluated its effects on accuracy and increased training time. The two layer MLP model used was trained on the 128 pixel version (128x128) of the dataset, instead of the 28 pixel version (28x28) we had previously trained the MLP models on.

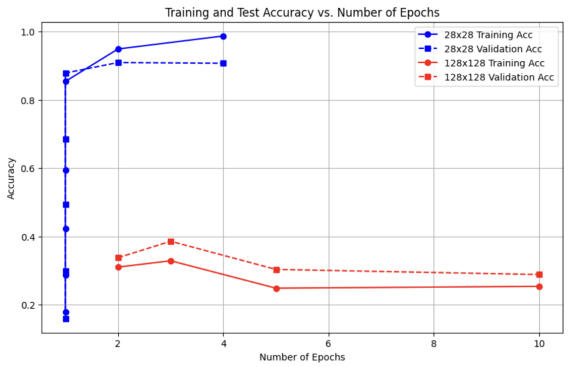

Figure 5: Resolution Accuracy Comparison

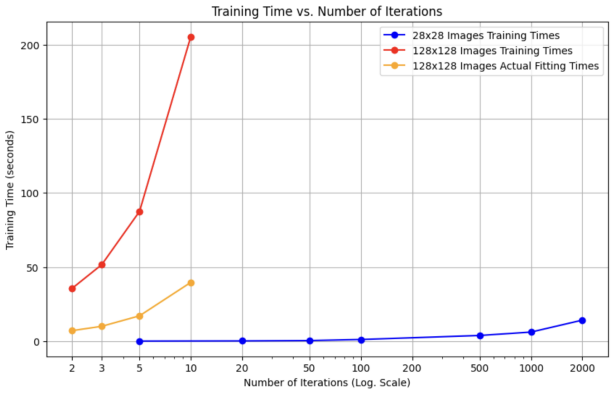

Figure 6: Resolution Training Time Comparison

By training on 128x128 images, our model is exposed to a dramatically larger set of parameters (16,384 compared to 784) that must be trained, leading to an increased training time. As a result, the model cannot reach a high level of accuracy even as the number of epochs increases due to numerical instability. Its significantly increased training time due to the fact that it must be fit in batches using PyTorch's DataLoader reduces the amount of epochs the model can be trained on. This all results in lower training and validation accuracies compared to those of the same model trained on the 28x28 image size data, and serious underpeformance.

From the orange line in Figure 6, we observe that the model's fitting time is similar to that of the MLP on 28×28 data, indicating that the increased training time likely results from loading data in batches. To address this, we could employ early dropout regularization with a high dropout rate, which would enhance model generality and reduce the number of learnable parameters, thereby reducing training time.

3.6: CNN Model

In task 3.6, we implemented a convolutional neural network (CNN) consisting of one fully connected hidden layer and one fully connected output layer. This model was then trained on the same OrganAMNIST 28x28 dataset,

and its accuracy was compared to the accuracies of the MLPs we had trained previously.

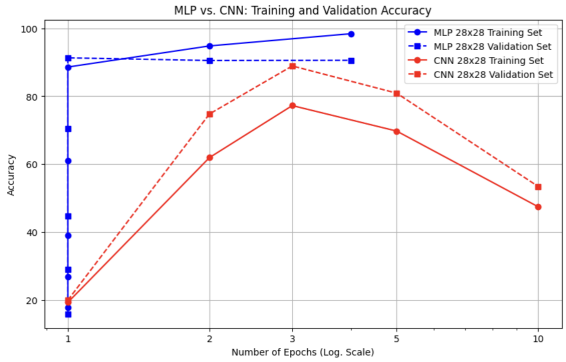

Figure 7: MLP vs CNN Accuracy Comparison

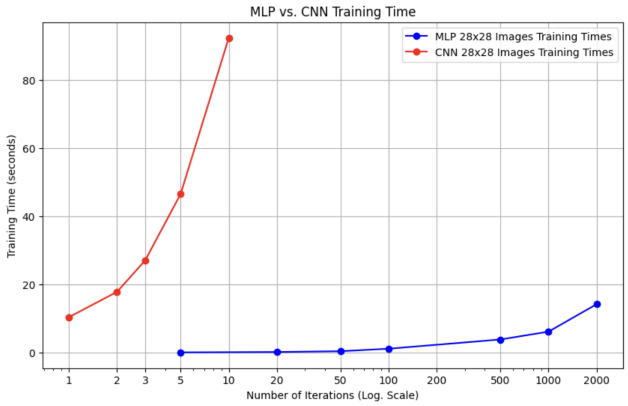

Figure 8: MLP vs CNN Time Comparison

After finetuning our hyperparameters (a 3x3 kernel and 6 and 16 out channels chosen for the first and second layer respectively as according to the LeNet-5 Architecture), we found that the CNN model achieved higher training and validation accuracies compared to the MLP at a more consistent rate, however, this came at the expense of a higher number of required epochs. This is to be expected, as the simpler MLP model reaches over 90% accuracy within its first 2 epochs, while it takes the CNN at least 5 epochs to reach similar levels. The CNN showed a steep increase specifically after the third epoch highlighting its efficiency in learning meaningful features. Additionally, the CNN required considerably more time training time to reach these levels of accuracies (as seen in figure 8), growing exponentially with each increase in number of epochs. This emphasizes the suitability of the MLP model for simple image (28x28) classification tasks, and suggests that the more complex CNN might be more suited to the more complex higher resolution dataset.

3.7: CNN Model with 128 Pixel Images

After some minor adjustments, we were able to implement the CNN model from 3.6 to train on the higher resolution dataset. To achieve this, we added two pooling layers to reduce the dimensions of the feature map, without increasing the width of the network.

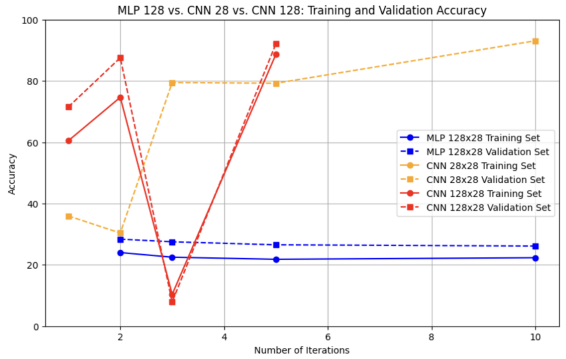

Figure 9: MLP 128 vs CNN 28 & 128 Accuracy Comparison

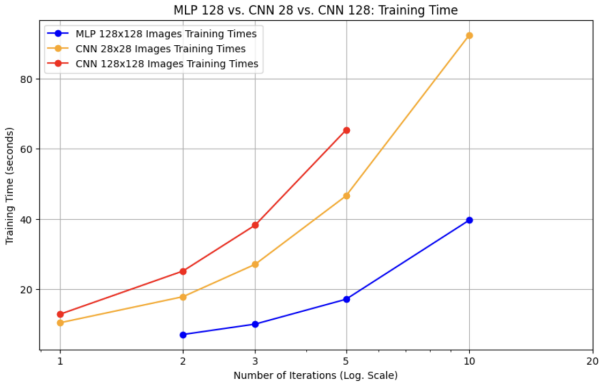

Figure 10: MLP 128 vs CNN 28 & 128 Training Time Comparison

As expected, we found that the CNN performed significantly better than the MLP model on the higher resolution dataset, even when the model is trained on just one epoch. However, it is important to note that this comes at the expense of an increased training time for the same number of epochs. Additionally, compared to the 28 pixel data, the CNN trained on 128 image size data requires a lower number of epochs, but is also less stable and consistent in its accuracy.

3.8: PreTrained Model: ResNet18

In this next task, we implemented a pre-trained ResNet18 model, freezing its convolutional layers and removing its fully connected layers, so that we could add our own fully connected layers to be trained on both of our datasets.

We found that while the pretrained ResNet18 model performs well regarding its accuracy (a steady increase in accuracy with each epoch), it requires a higher amount of epochs to reach that level than the MLP, as well as the CNN model. This again comes as no surprise as the model is more complex, however we have seen that on the 128x128 dataset, the ResNet18 outperforms the MLP and CNN models due to the fact that it has been pretrained. We do note however, that the CNN and ResNet have similar training time, both increase steeply around the third epoch, but overall the CNN still demands the highest training time on the 28x28 dataset.

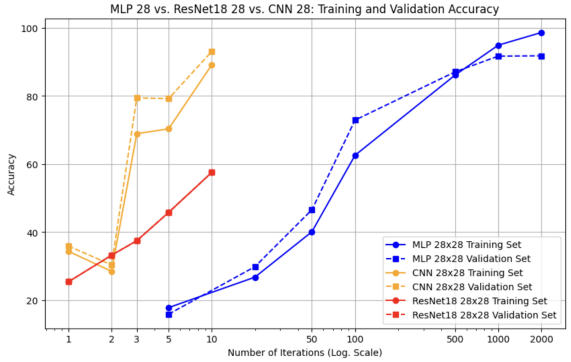

Figure 11: MLP vs. CNN vs. ResNet18 on 28x28 Data: Accuracy Comparison

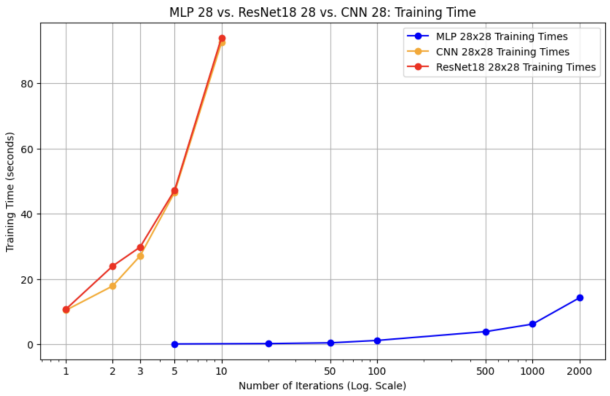

Figure 12: MLP vs. CNN vs. ResNet18 on 28x28 Data: Training Time Comparison

3.9: Testing

In this final part of Task 3, we decided to test our best performing models given the number of epochs required and most consistent high levels of accuracy on the same test dataset (the 28x28 image size data), serving as a baseline. As seen in figure 13, the MLP achieved the highest test accuracy at 74.10%, slightly outperforming the CNN. However, the pre-trained ResNet18, despite being efficient in early training stages, achieved only 60.46% accuracy.

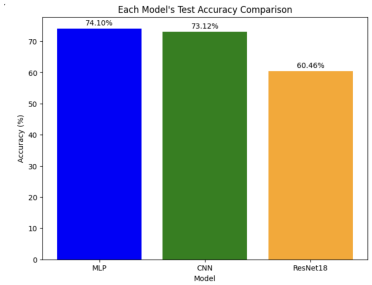

Figure 13: Test Results of the best models on the 28x28 Image Size Dataset

The training plots illustrate that while the MLP rapidaly improves within the first few epochs, the CNN required more training time to achieve high accuracy, but was ultimately more stable. ResNet18, while faster per epoch due to frozen layers, underperformed in overall accuracy compared to our custom models. Given enough compute power, we would find that the slightly more complex CNN and ResNet18 models would significantly outperform the MLP on the 128x128 image size dataset, hence the need to adapt your models to your datasets.

Additional Experiments

To extend our project further, we explored the effect varying the number of units within each hidden layer of an MLP model with two hidden layers. We initialized five MLP models with 50, 150, 256, 350, and 450 units respectively in each of the two layers. The results were then used to observe how the width of a hidden layer impacts accuracy.

Each of the models, with different hidden layer widths, performed very similarly on the image classification task. Although the 50 unit model was slightly less accurate on train and test sets than the rest, they all displayed the same shape and had very similar accuracy values. This aligns with what we know about how changing the depth in an MLP model is more effective than changing the width.

Conclusion

Overall, this assignment emphasized the importance of key factors in image classification tasks. We explored how varying parameters such as depth and activation functions can significantly influence the performance of an MLP model, particularly in terms of accuracy. We also observed the important role that dataset details, including normalization and resolution, play in model performance. By comparing different model types—MLP, CNN, and the pre-trained ResNet—we gained insights into the strengths of each approach and the conditions in which they excel. For further investigation, future experiments could explore hyperparameter optimization techniques, the effects of data augmentation strategies, such as rotation or scaling, to improve generalization, and fine-tuning pre-trained models like ResNet with deeper layers or alternative architectures to enhance accuracy.

Statement of Contribution

The assignment was worked on together collaboratively; Ayaz completed tasks 1 and 2, then Ayaz, Jacqueline, and Christopher all worked on task 3, and the report was completed by Jacqueline and Christopher.

Sources

Jerga, Filip. "Mastering the Multi-Layer Perceptron (MLP) for Image Classification." Medium, Eincode, 12 Sept. 2024.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, Bingbing Ni. Yang, Jiancheng, et al. "MedMNIST v2-A large-scale lightweight benchmark for 2D and 3D biomedical image classification." Scientific Data, 2023.

Jiancheng Yang, Rui Shi, Bingbing Ni. "MedMNIST Classification Decathlon: A Lightweight AutoML Benchmark for Medical Image Analysis". IEEE 18th International Symposium on Biomedical Imaging (ISBI), 2021.

Appendix

```
Model trained on 2 epochs, with SGD optimizer:
        Training Time: 502.55 seconds
        Training Accuracy: 78.69%
        Validation Accuracy: 89.11%

Model trained on 2 epochs, with Adam optimizer:
        Training Time: 522.77 seconds
        Training Accuracy: 17.84%
        Validation Accuracy: 15.91%
```

Figure 14: CNN model: Adam vs. SGD optimizer

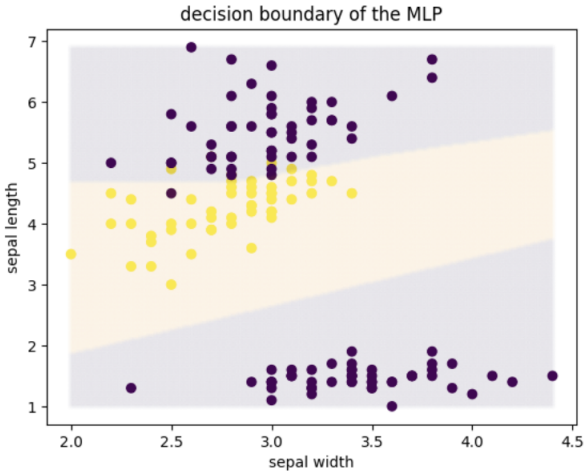

Figure 15: MLP on Iris Dataset