A3: Image Classification with Multilayer Perceptrons and Convolutional Neural Networks

Alexandre St-Aubin, Jonathan Campana, & Jake Gameroff

Comp 551: Applied Machine Learning

January 13, 2025

Abstract

The ability to automatically and accurately classify medical images using Neural Networks has been a significant innovation in medicine. In this project, we use Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) to classify 28×28 as well as 128×128 image data from the OrganAMNIST dataset [1]. We trained the MLP using backpropogation and mini-batch stochastic gradient descent; and to prevent over-fitting, we also applied $L1$ and $L2$ regularization to the MLP. We experimented with our MLP by varying its depth, width, and activation functions and assessing how these modifications affect the classification rate. To further enhance accuracy, we used a CNN and fine-tuned a pre-trained Resnet18 model by replacing its fully connected layers. After using data augmentation and implementing early stopping, our MLP was able to correctly classify 74.27% of images from our test dataset, while our best CNN correctly classified 93.66% of the images, demonstrating the benefits of spatial feature extraction. The fine-tuned ResNet achieved an accuracy of 90.25% on the test [1] set.

Introduction

The classification of medical images has become an essential task in modern medicine. However, medical images are often complex and have many pixels; this presents challenges to traditional ML methods, which heavily rely on manual feature extraction. Nonetheless, neural networks are powerful tools for addressing these challenges, particularly because they can extract features directly from raw data. However, due to computational limitations, neural networks are often restricted to classifying low-resolution images. Fortunately, the formalism of the *convolutional neural network* (CNN) changed this significantly, in particular because CNNs allow for the automatic extraction of spatial features directly from image data.

In this project, we use multilayer perceptrons (MLPs) and CNNs to classify 28×28 medical image data from the OrganAMNIST dataset [1]. Figure 3 shows a collage of 100 sample images from the dataset. Our main goal is to evaluate, compare, and contrast the performance of our MLP and CNN models on this dataset. We also explore how design choices such as the depth, width, and the types of activation functions used (e.g. ReLU, Leaky ReLU, sigmoid, tanh, etc.) impact accuracy and computational efficiency. We saw the improvement of test accuracy when increasing the number of layers, by training a model with no hidden layers, 1 hidden layer with 256 units and ReLU activation, and 2 hidden layers with 256 units each and ReLU activation, where the test accuracy were 62.16%, 62.16%, and 73.49% respectively. We also compared the performance of different activation functions like tanh, and leakyReLU with the 2 hidden layer ReLU, where ReLU had the best results with a test accuracy of 73.49%, followed by leakyReLU with 73.23%, and tanh with 68.19%. L1 and L2 regularization was performed on the 2 hidden layer ReLU MLP, which helped reduce overfitting and improved the test accuracy to 74.17% and 75.09% respectively. We tested the 2 hidden Layer ReLU MLP on unnormalized data, which had a significantly lower test accuracy of 54.57%. We also compared the models trained with 28×28 images with models trained with 128×128 which took significantly longer, without improving test accuracies, where the ReLU models had test accuracies of 73.49% and 70.39% respectively, and the models trained on the other activation functions saw even bigger drops in test accuracy.

We then trained a simple CNN on 28×28 grayscale images, achieving 80.51% test accuracy, which improves to 90.15% with Batch Normalization. An improved version with more layers and larger 128×128 images further increases accuracy to 93.66%, showcasing the benefits of higher resolution and a more expressive design. Similarly, we fine-tuned a pre-trained ResNet18 by adding fully connected layers and using techniques like Dropout and weight decay to balance accuracy and overfitting. This model achieves 90.25% accuracy but requires more computational time and underperforms compared to the improved CNN.

[1] Throughout, terms like Training, Validation, and Testing sets refer to the subsets of the OrganAMNIST dataset obtained by specifying the argument `split={train, val, test}` during data loading.

Related Work

The development of CNNs for image classification was significantly advanced by the introduction of AlexNet, which pioneered the use of ReLU activations for faster training, dropout for regularization, and GPU acceleration to help with computational efficiency [2]. Their model set a new standard for image classifiers, consequently leading to a lot more research in the area and the development of more complex, sophisticated models. Building on these advancements, machine learning researchers have streamlined various processes in the medical field by developing CNNs capable of accurately classifying medical images.

For example, in much the same way as we do in this paper, Doerrich et al. conducted an analysis using the MedMNIST+ dataset (which includes OrganAMNIST), experimenting with various CNN architectures to classify medical images [3]. Their work highlights the importance of systematically evaluating different CNN architectures to obtain optimal results.

Data Processing

We obtained our data from the MedMNIST dataset, specifically the OrganAMNIST subset of which. We use this dataset to train models to recognize which organ (a liver, kidney, bladder, etc.) is shown in a medical image. Each image depicts one of seven organ types, where the goal is to correctly determine the organ based on the image. To get the data ready, we applied one-hot encoding to the labels by converting each organ type into a binary vector. We also normalized the data to make sure the input ranges were consistent and to improve model performance.

The dataset is imbalanced, as shown in Figures 4 and 5. Some classes, like 6, have far more samples (18%), while others, like 3 and 2, have much fewer (around 4%). This could lead to the model performing better on larger classes and worse on smaller ones. The heatmap in Figure 6 shows the correlation between different classes in the dataset. Each class has perfect correlation with itself (1.00) and very low or negative correlation with other classes, meaning that each image belongs to only one class. This confirms that the labels are mutually exclusive and correctly one-hot encoded.

We can also understand the images in the dataset by analyzing the distributions of pixel intensities. Figure 7 shows the distribution of pixel values for 1000 randomly selected images from the dataset. We see that most pixel values are -1 (black) or 1 (white), with fewer in between. This shows that many organ images have strong contrasts, which might make it easier for the model to tell the organs apart based on their shapes and edges. Moreover, the bar plot in Figure 8 shows the average pixel intensity variance for each organ class. A class having high variance, such as class 7, may be due to differences in the organ's structure or how it is captured in the images. Classes with low variance, like class 0, may have simpler or more uniform images.

1 Multilayer Perceptron

The Multilayer Perceptron (MLP) is a feed-forward neural network with fully connected input, output, and hidden layers. We implement an MLP from scratch to classify medical image data, training it using backpropogation and mini-batch stochastic gradient descent.

We performed various experiments to test the performance of different MLP architectures. For example, we consider models with no hidden layers, one hidden layer, and two hidden layers; we try activation functions like ReLU, tanh, and Leaky ReLU; and we explore the effects of L1 and L2 regularization, unnormalized data, and larger image sizes on accuracy and training speed. We expand on these experiments as follows.

1.1. Number of Hidden Layers. We created three MLP models: one with no hidden layers, one with a single 256-unit hidden layer, and one with two 256-unit hidden layers. Both models with hidden layers use ReLU activations. We plot, for each of these MLPs, the train, validation, and test losses and accuracies as a function of the number of epochs. Figure 12 shows the performance of the MLP with no hidden layers. The model has trouble learning non-linear relationships, as expected, causing a low test accuracy of about 62%. Figure 13 and Figure 14 show the trends for the MLPs with one hidden layer and two hidden layers respectively. We observe a clear trend, as expected: more hidden layers lead to higher test accuracy because the model becomes better at fitting to non-linear relationships, reaching a test accuracy of around 73.5%.

1.2. Type of Activations. We make two new two-hidden-layer MLP, one copy will use only tanh activations and the other will use only Leaky-ReLU activations. Figures 14, 15, and 16 compare the performance of the two-hidden-layer neural network with ReLU, Tanh, and Leaky ReLU activations, respectively. ReLU and Leaky ReLU look similar, as expected, because they are similar functions. The ReLU functions seem to learn more slowly and accurately, with more stability. On the other hand tanh learns quickly, but not as accurately as the other two functions, with the test accuracies of 73.49%,

68.19%, and 73.23%, respectively. We also trained a model with sigmoid activation, but the model would learn more slowly compared to the other activation functions, yielding a test accuracy of around 60% after 20 epochs 25.

1.3. Regularization. Regularization is a common technique used to help reduce overfitting in neural networks. Regularization involves adding penalties to the model weights during training; this stops the weights from getting too large and forces the network to focus patters in the data (rather than memorizing it). We modify the two-hidden-layer MLP from (1.1) to adopt $L1$ and $L2$ regularization. Specifically, $L1$ regularization adds $\lambda \sum_j |w_j|$ to our loss function, while $L2$ regularization adds $\lambda \sum_j w_j^2$ to the loss function. Figure 17 shows the performance of the MLP using $L1$ regularization, and Figure 18 shows the same but with $L2$ regularization. Compared with Figure 14 (the same MLP with no regularization), we see a light improvement in the test accuracy, likely occurring due to a reduction in overfitting. It seems as though $L2$ regularization works slightly better than $L1$ regularization, possibly because the model looks to penalize large weights.

We also applied data augmentation to the OrganAMNIST dataset to further-prevent overfitting. We applied various transformations to the original images, such as random horizontal and vertical flips, slight rotations (up to 15 degrees), and random translations (up to 10% of the image size). Figure 19 shows examples of original and augmented images. Using this augmented data, our best MLP improved its test accuracy from around 73.3% to 74.05% compared to training on the original dataset. To further prevent overfitting, we implemented *early stopping*, which brought the test accuracy to a high of **74.27**% test accuracy (see Figures 18 and20 to compare both models trained on non-augmented vs. augmented data respectively). Hence, augmentation makes the model more robust by reducing overfitting.

1.4. Normalization. Rather than using normalized data, we now train the two-hidden-layer MLP from (1.1) using *unnormalized* images. Figures 21 and 14 show the performance of the MLP trained on unnormalized and normalized data, respectively. The training accuracy of the MLP using unnormalized data improves steadily but remains lower than the MLP with normalized data. Validation and test accuracies are also lower and improve more slowly. In contrast, with normalized data, training accuracy quickly reaches nearly 100% (as expected, since we don't regularize), while validation and test accuracies stabilize around 90% and 73%, respectively. Normalizing the data clearly leads to faster learning, better generalization, and higher overall performance.

1.5. Image Sizes. We train the same MLP from (1.3) (two hidden layers, ReLU activations, and with regularization) but with 128×128 normalized images instead of 28×28 data. We analyze the affect of this change on model accuracy, the number of epochs, and the running time. Figures 18 and 22 compare the performance of the MLP from (1.3) (with $L2$ regularization for both) trained on 28x28 and 128x128 images, respectively. As expected, the model trained on 128×128 images performed slightly worse than that trained on 28×28 images (the 128-image-size MLP had a test accuracy of around 70.39%, while the 28-image-size MLP had a test accuracy of around 73.3%. The tanh went from 68.19% to 64.86%, and leakyReLU saw a drop similar to ReLU). Interestingly, in the 28-image-size MLP, the train accuracy is above the validation accuracy, and vice versa for the 128-image-size MLP. This might happen because the 28×28 model may be overfitting to the training data in less epochs, while the 128×128 model took longer learn due to the larger input size providing richer features. On the same machine and under similar conditions, training the MLP on the 28×28 images took around 2 minutes and 8GB of RAM, while using 128×128 images took about 15 minutes and 16GB of RAM. We also trained MLPs as in (1.3) but with Leaky-ReLU and tanh activations (see Figures 23 and 24), though this did not yield superior results similarly to the two models with ReLU activation.

2 Convolutional Neural Network

2.1 Vanilla CNN

We first train a convolutional neural network (CNN) with two convolutional layers, each followed by max pooling (stride 2 and filter size 2×2) to reduce spatial dimensions. The network includes a fully connected layer with 256 units, using ReLU as the activation function for all hidden layers and 3×3 kernels for the convolutions. The architecture is illustrated in Figure 26.

The model is evaluated on the MedAMNIST dataset, consisting of 28×28 grayscale images. Training uses gradient descent with a learning rate of 10^{-4} and a batch size of 64. Higher learning rates slowed convergence beyond 50 epochs, while lower ones were unstable. A batch size of 64 balanced generalization performance and training efficiency, as smaller batch sizes improve generalization but increase training time [4].

Figure 9 shows the loss and accuracy over 70 training epochs. The model struggles to learn, achieving a test accuracy of only **80.51**%. Still, this is a considerable improvement over the MLP, which couldn't surpass 75% on the test set. To address this, we apply **Batch Normalization** after each convolutional layer, improving stability and convergence speed. Figure 10

illustrates the results of this modified model, which achieves a best test accuracy of **90.15**% at epoch 35—about a 10% improvement. Notably, training accuracy reaches 100% by epoch 30 without test set overfitting, indicating a well-chosen architecture.

2.2 Improved CNN

Next, we implement an improved version of the CNN presented above. The model is trained on larger 128×128 images, which are expected to provide more detailed information and improve performance. The architecture is expanded to include four convolutional layers (up from two) and two fully connected hidden layers (up from one), as illustrated in Figure 26. Additionally, we apply Batch Normalization after each convolutional layer to stabilize learning and accelerate convergence. To further optimize training efficiency, we increase the minibatch size from 64 to 128, reducing training time. The improved model achieves faster, more stable convergence and a higher test accuracy of **93.66**% at epoch 16, as shown in Figure 1. While more expressive to leverage the increased data, it also requires significantly longer training time due to the larger dataset and number of parameters.

We note that the old model achieved 90.15% accuracy at epoch 35, requiring a similar training time as 16 epochs of the improved model, which reached a significantly higher accuracy. This clearly establishes the superiority of the improved CNN. We also notice that the CNN handles large images considerably better than the MLP, which had lower accuracy with 128×128 images than with 28×28.

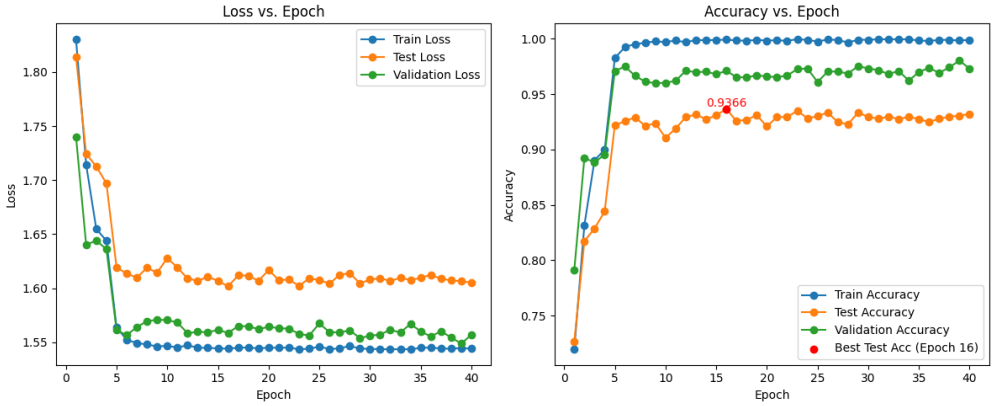

Figure 1: Loss and accuracy plots as a function of epochs for our improved CNN model.

3 Pre-trained Model

In this section, we fine-tune a pre-trained ResNet18 on OrganAMNIST 128×128 images by freezing its convolutional layers and appending a fully connected layer with 512 nodes and ReLU activation. Only the newly added fully connected layer is trained. Initially, learning is slow and the model shows subpar accuracy on the training set, indicating high bias, as seen in Figure 11. To address this, we increase the expressiveness of the network by adding more fully connected layers. We also add **Dropout** and weight decay to reduce overfitting.

The final network appended to ResNet's convolutional layers starts with a fully connected layer mapping 512 to 1024 features, followed by ReLU and 50% Dropout. Another layer reduces features to 512, with ReLU and Dropout repeated. Finally, a fully connected layer outputs 11 classes, with Softmax converting them to probabilities. This design balances model expressiveness with regularization, helping to prevent overfitting while effectively handling the feature-rich representations from ResNet.

Figure 2 shows the loss and accuracy of the fine-tuned ResNet as a function of the number of epochs. The fine-tuned model achieves a test accuracy of **90.25**%, comparable to the Vanilla CNN described in section 2.1, which was trained on 28×28 images. However, the fine-tuned ResNet underperforms relative to our improved CNN (see section 2.2), trained on 128×128 images, which achieved a higher test accuracy of 93.66%. This is most likely due to the fact that ResNet18 was trained on a set of images (CIFAR-10) very different to OrganMNIST. For starters, the images were RGB, and the set contained airplanes, automobiles and animals, rather than organs. Additionally, fine-tuning ResNet18 required significantly

more time compared to training the improved CNN from scratch. This result is expected, given that ResNet18's larger parameter count increases both computational cost and inference time.

Nonetheless, the ResNet performed way better than the MLP and even had shorter training times on the 128×128 images. This is also expected, as ResNet18 was built for performance on image classification.

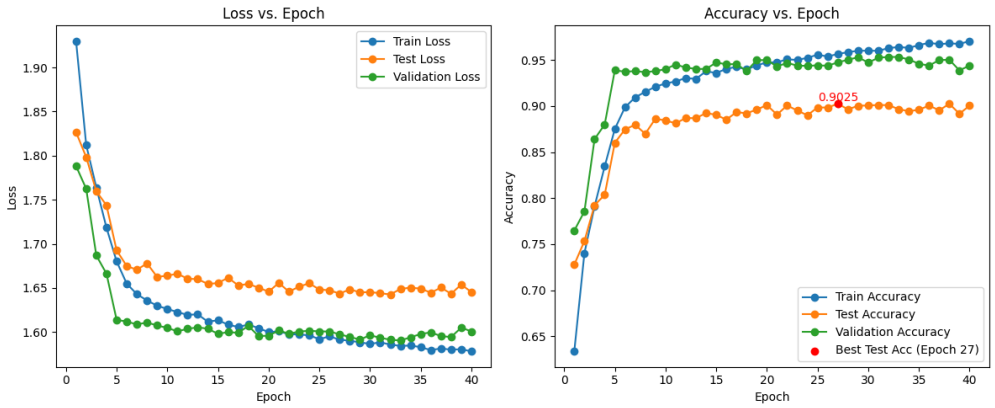

Figure 2: Our improved fine-tuned ResNet18 with 2 fully connected layers, dropout, and weight decay.

Conclusion

We performed various experiments to examine the performance of MLPs, CNNs, and a fine-tuned ResNet on the OrganAMNIST dataset. By trying different architectures and modifying our activation functions and our regularization and normalization techniques, we were able to explore how these design choices affect model accuracy and efficiency. Our results highlight the benefits of using more sophisticated neural networks like CNNs for image classification tasks. Add more specific main results that we can conclude here.

Creativity

1. Thorough statistical analysis of the OrganAMNIST dataset (please see the Introduction for more details), including (1) analyzing class distributions and identifying dataset imbalances (Figures 4 and 5), (2) calculating class correlations using a heatmap (Figure 6), (3) examining pixel intensity distributions across the dataset (Figure 7), and (4) measuring pixel variance for each class to explore structural and imaging differences (Figure 8).

2. Implementation of data augmentation to prevent over-fitting (please see Section 1 for more details). We applied transformations such as random horizontal and vertical flips, slight rotations (up to 15 degrees), and random translations (up to 10% of the image size). These augmentations helped increase the diversity of the training data without collecting new samples. See Figures 19 and 20 for examples of the augmented images compared to the original ones, and our results using an MLP trained on the augmented data.

3. Training additional MLPs using Leaky-ReLU and tanh activations to compare with the ReLU models from (1.3). While these experiments did not yield superior results, they provided insight into the effects of activation functions on model performance (Figures 23 and 24).

4. Implementation of early stopping to prevent overfitting and reduce unnecessary computation. Training stops if the validation loss does not improve for a set number of epochs (patience). This can be observed in the results shown in Figure 20.

5. Made a sigmoid activation layer class, and an MLP with sigmoid activation functions, to compare with the models trained with the other activation.

Contributions

- Alex: Implementation of the CNN and fine-tuning of the ResNet18, as well as part of the MLP and its training (my RAM go brrr).
- Jake: Worked on creating the MLP with Jonathan; writing of abstract, introduction, MLP section, and conclusion; implemented data augmentation and early stopping; ran various statistics on the image data.
- Jonathan: I built the MLP with all its components, hidden layers classes, loss function classes, and functions to convert the data into data frames. I made the main functions to train the models for all the parameters, and printing the plots of train, validation, and test accuracy and loss.

References

[1] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni, "Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification," *CoRR*, vol. abs/2110.14795, 2021.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, Curran Associates, Inc., 2012.

[3] S. Doerrich, F. D. Salvo, J. Brockmann, and C. Ledig, "Rethinking model prototyping through the medmnist+ dataset collection," 2024.

[4] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," 2017.

Appendices

A Appendix A

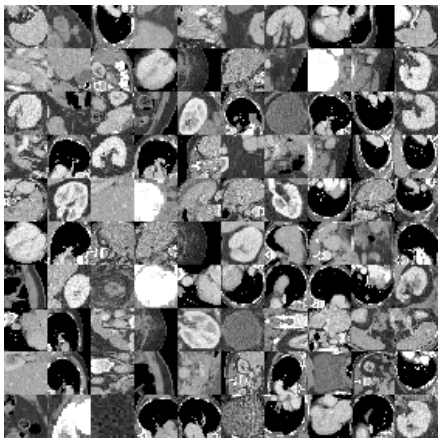

Figure 3: A collage of 100 sample images (each 28×28 pixels) from the dataset.

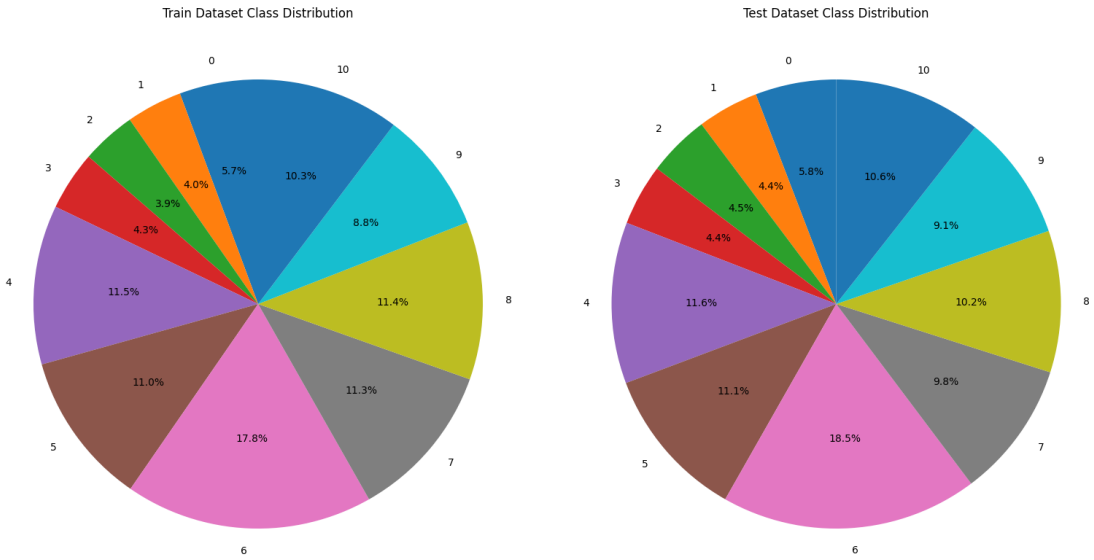

Figure 4: Class distribution of the training and test datasets. The pie charts show the relative proportion of each class; we see that the dataset is not perfectly balanced.

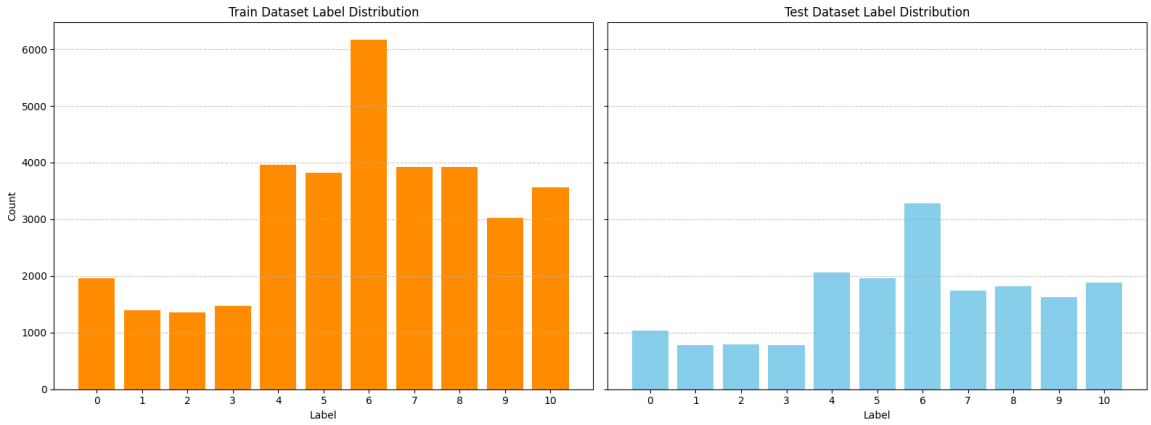

Figure 5: Label distribution of the training and test datasets. We observe that the dataset is not perfectly balanced.

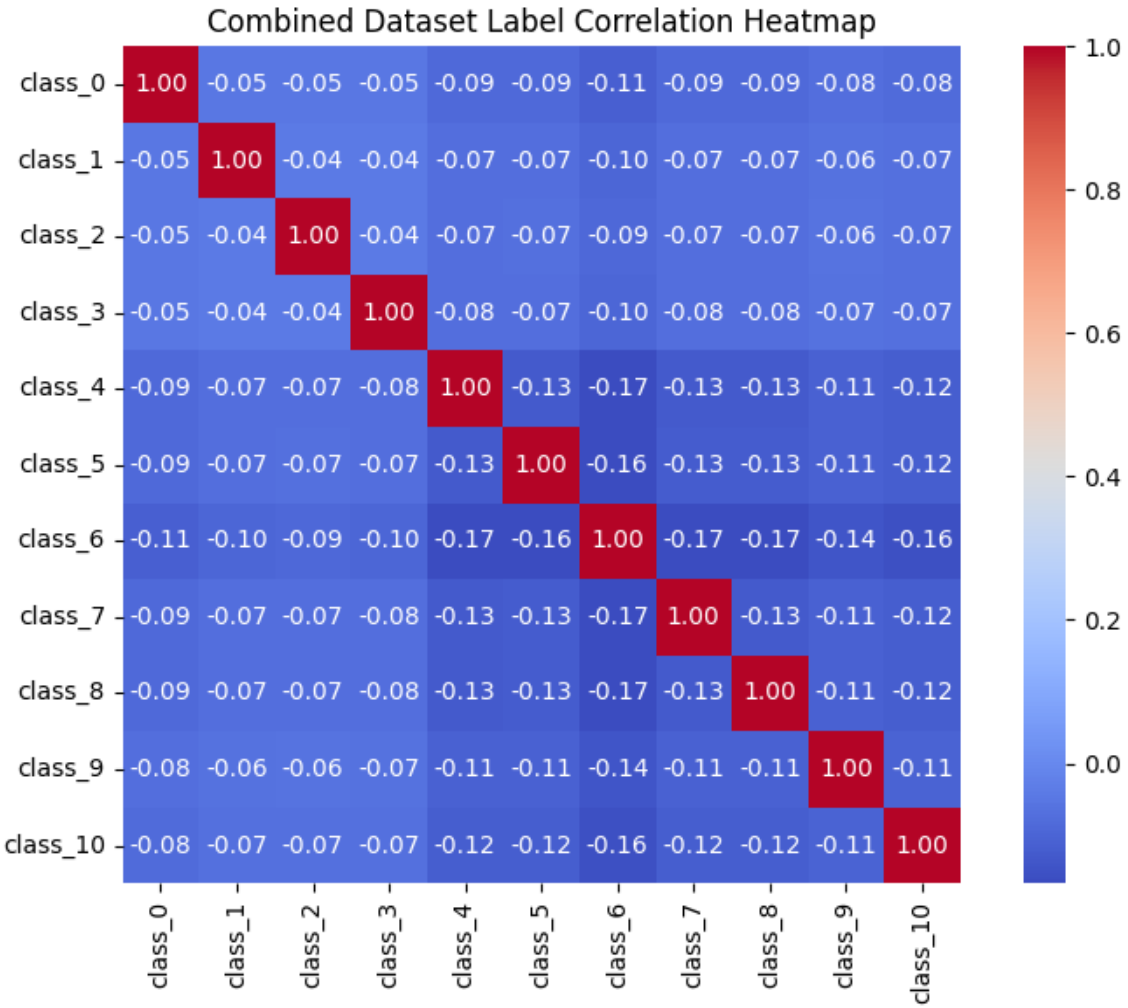

Figure 6: Label correlation heatmap for the dataset. Each class shows perfect self-correlation (1.00 on the diagonal) and near-zero with other classes. This confirms that labels are mutually exclusive and correctly one-hot encoded.

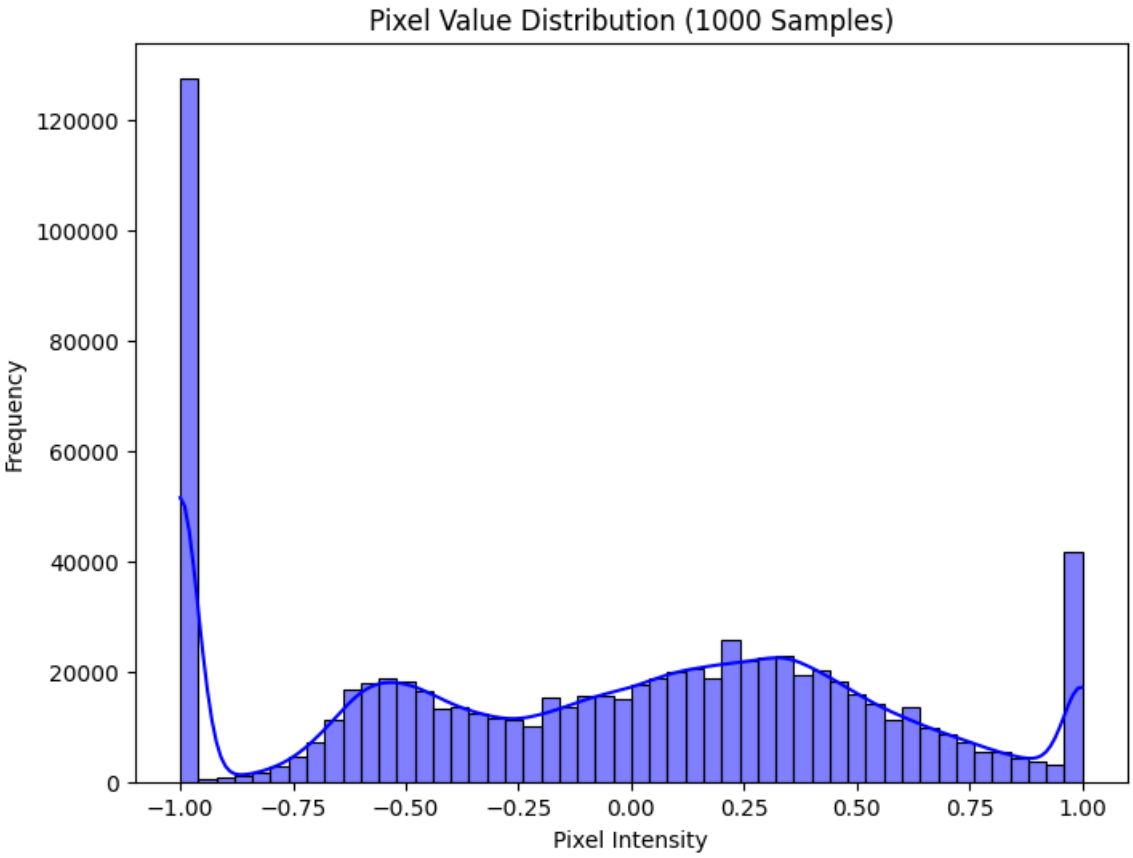

Figure 7: Pixel value distribution for 1000 randomly sampled images. The histogram shows the frequency of pixel intensities, ranging from -1 to 1, where -1 represents black pixels, 1 represents white pixels, and values in between represent varying shades of gray.

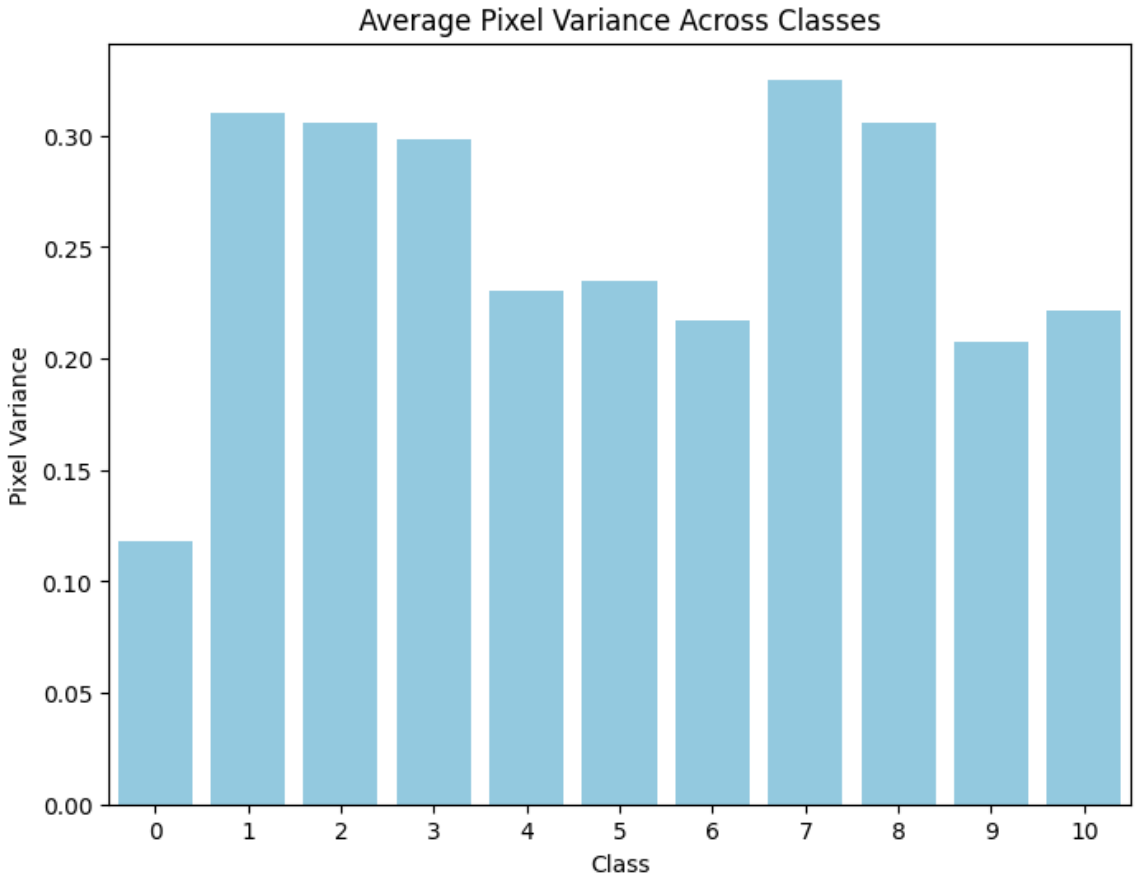

Figure 8: Average pixel variance across classes. The bar plot shows how much pixel intensities vary for each organ type. Higher variance means more diversity in the images. This may reflect differences in organ structure or imaging conditions.

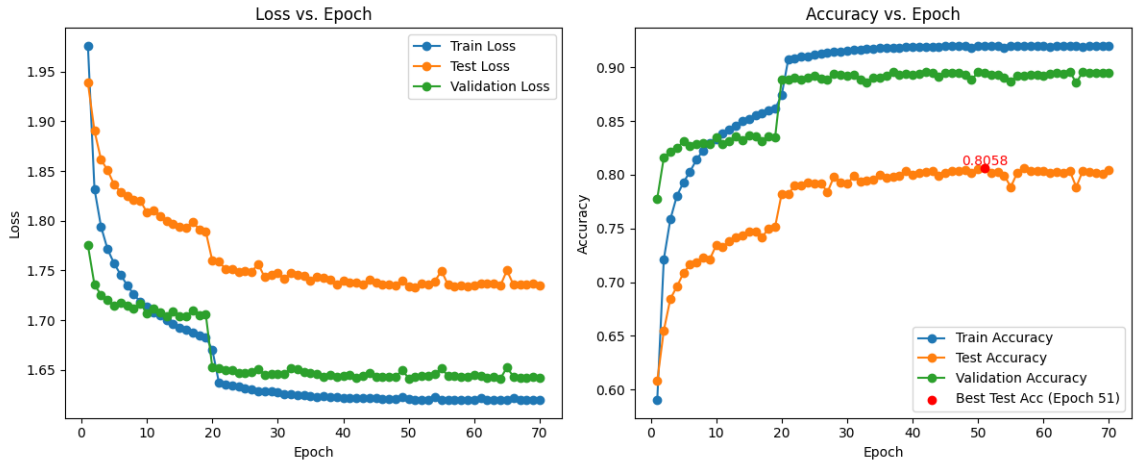

Figure 9: Vanilla CNN. Comparison of CNN models on 28×28 images.

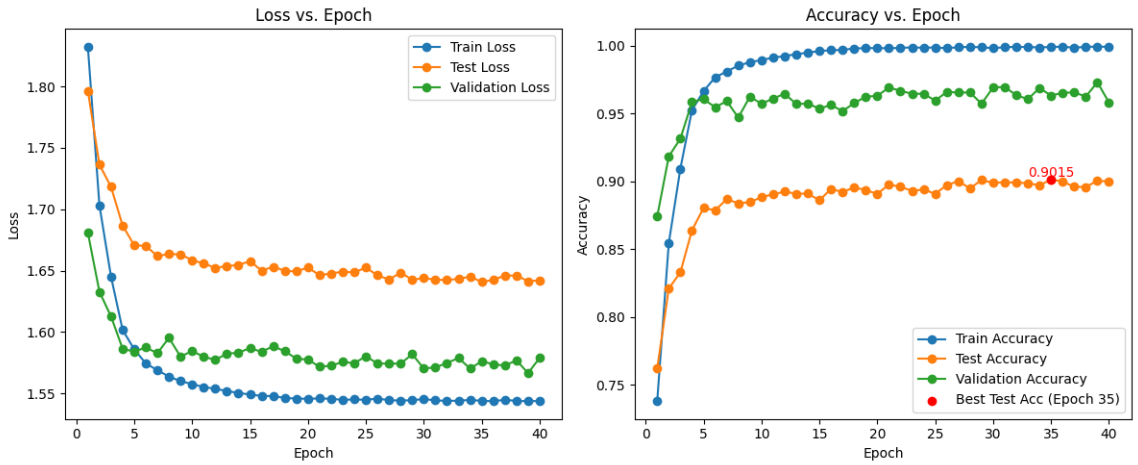

Figure 10: CNN with added batch normalization. Comparison of CNN models on 28×28 images.

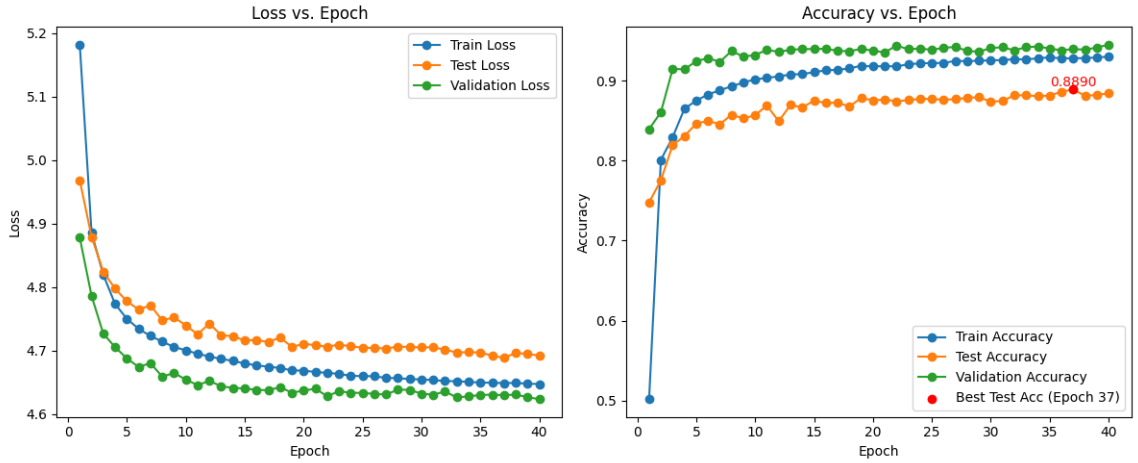

Figure 11: Our first fine-tuned ResNet18 with 1 added fully connected layer.

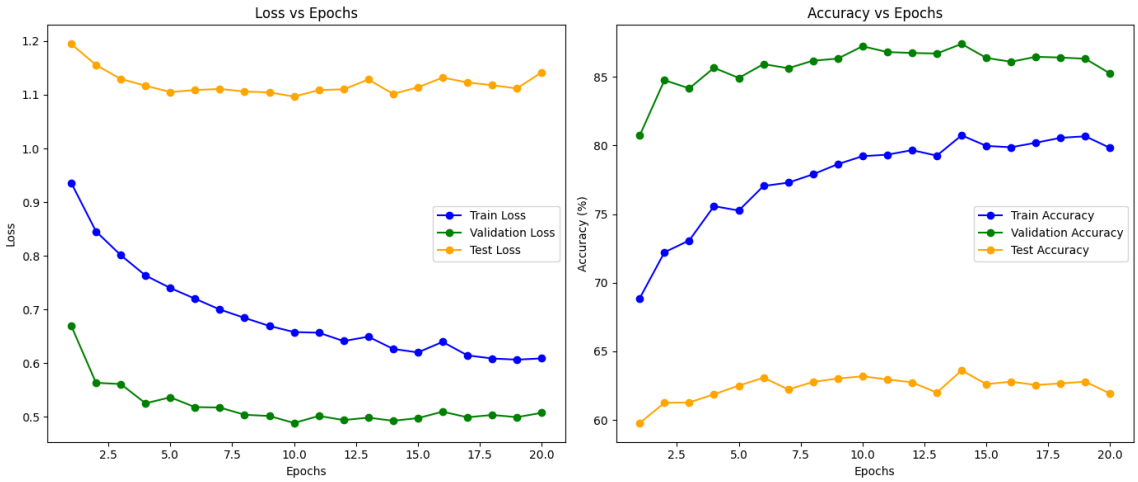

Figure 12: Performance of the neural network with no hidden layers.

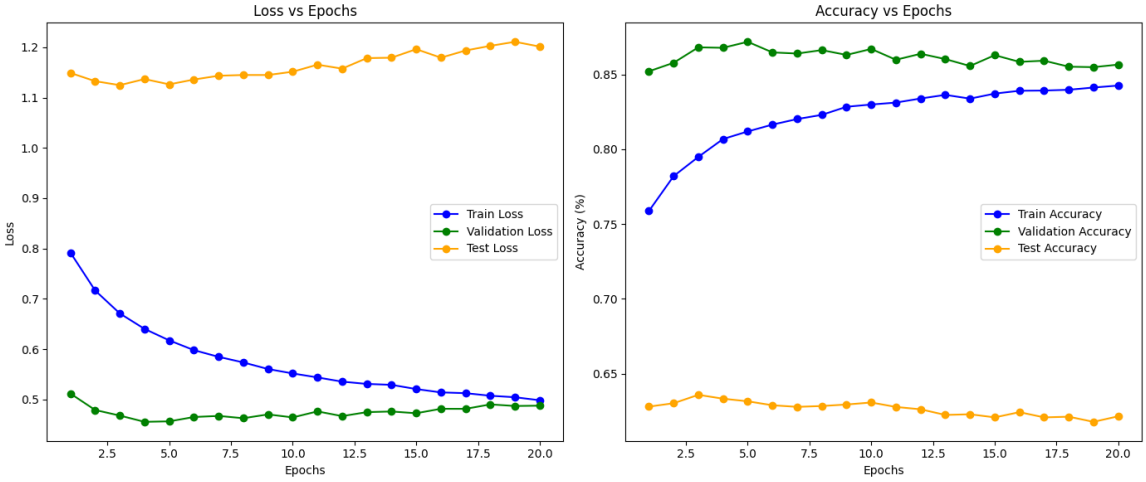

Figure 13: Performance of the neural network with one hidden layer and ReLU activation.

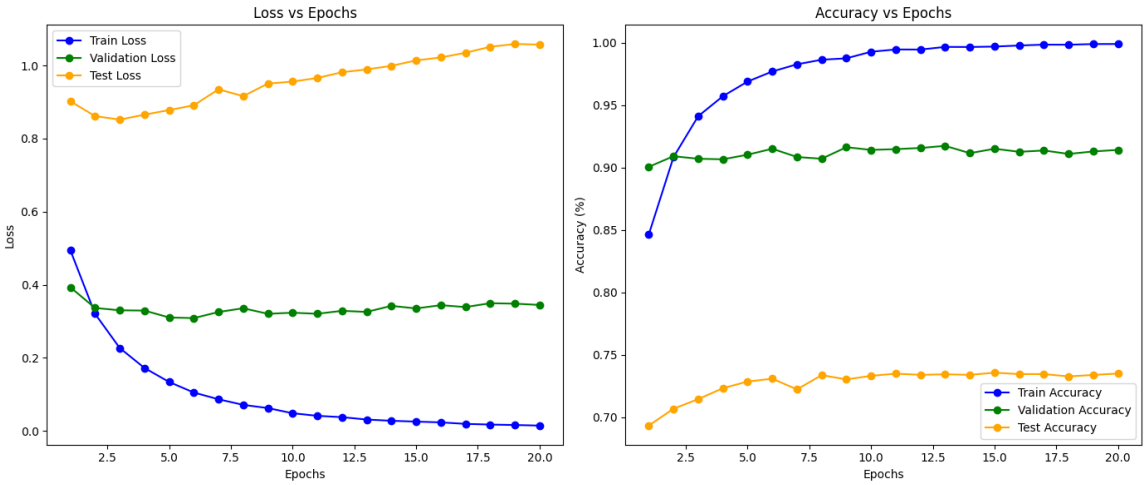

Figure 14: Performance of the neural network with two hidden layers and ReLU activation.

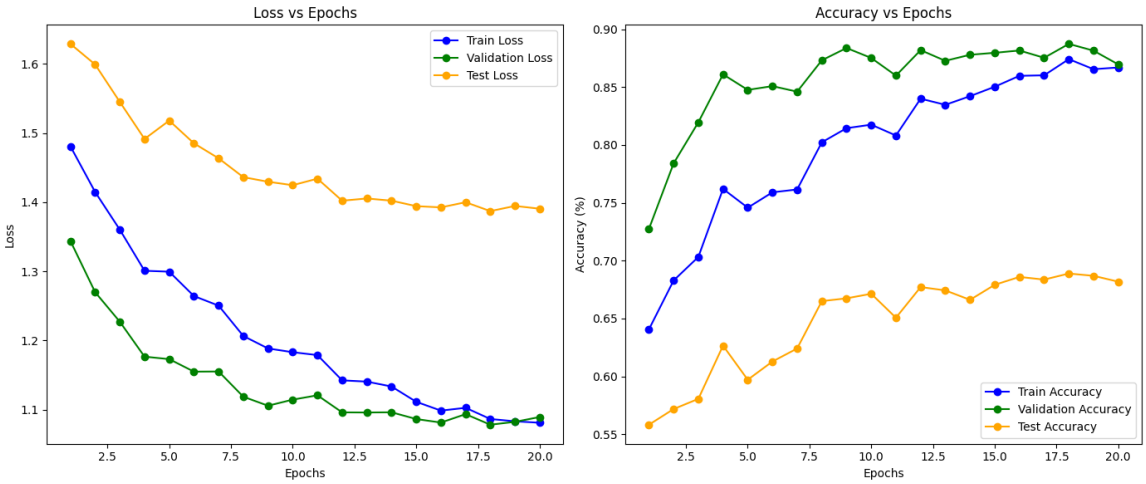

Figure 15: Performance of the neural network with two hidden layers and tanh activation.

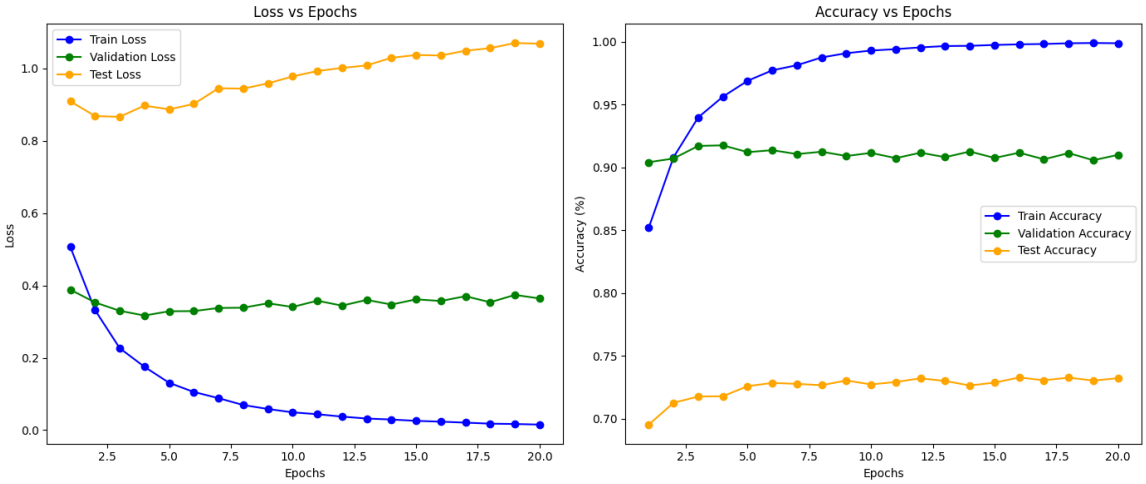

Figure 16: Performance of the neural network with two hidden layers and leaky ReLU activation.

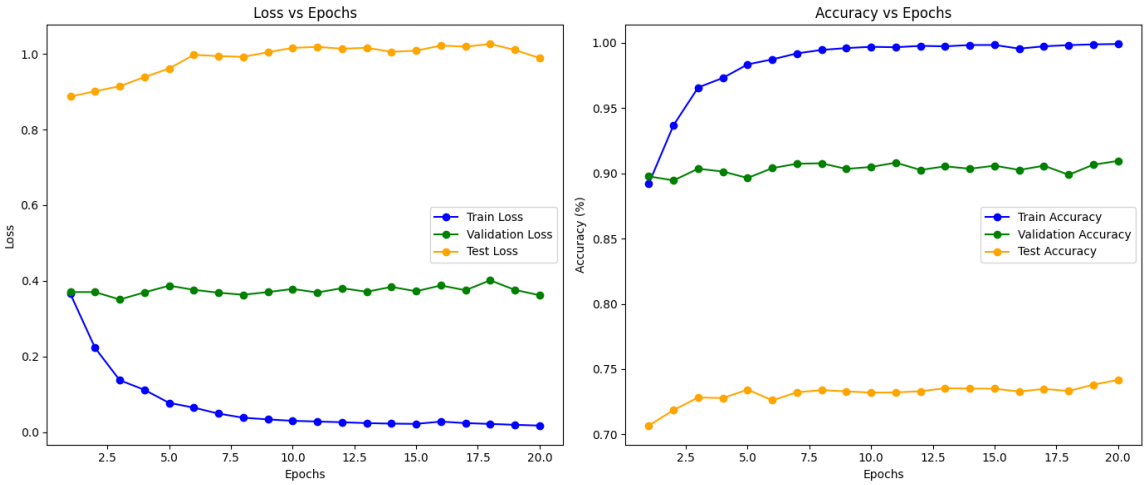

Figure 17: Performance of the neural network with two hidden layers and ReLU activation and L1 Regularization.

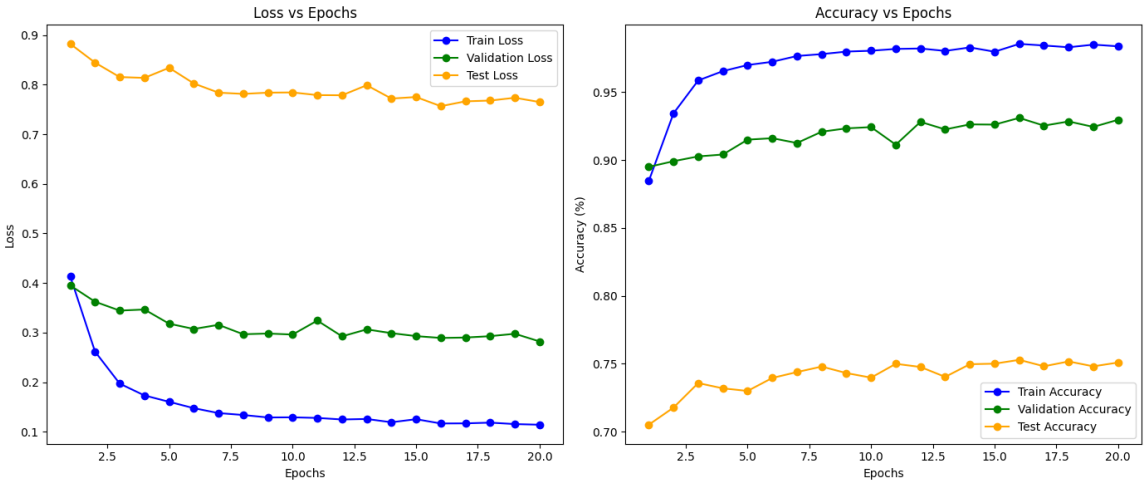

Figure 18: Performance of the neural network with two hidden layers and ReLU activation and L2 Regularization.

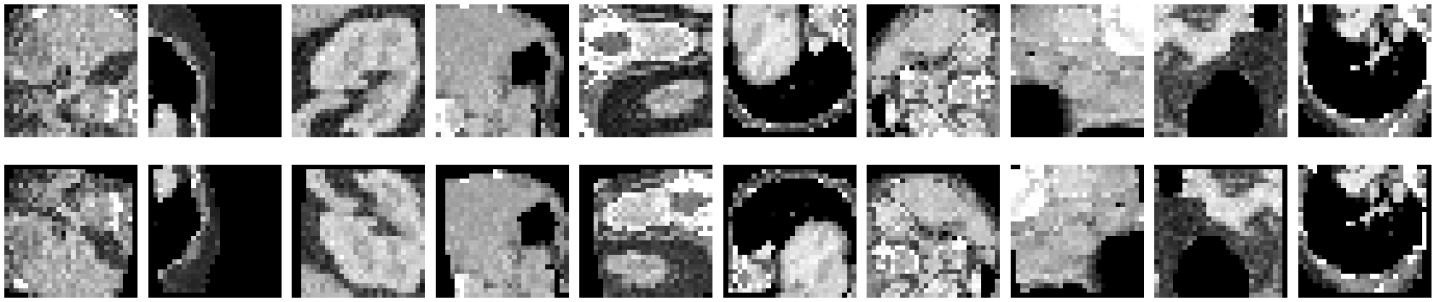

Figure 19: Comparison of original and augmented images from the OrganAMNIST dataset. The top row displays the original images, while the bottom row shows their corresponding augmented versions. Augmentations include random horizontal and vertical flips, small rotations, and translations.

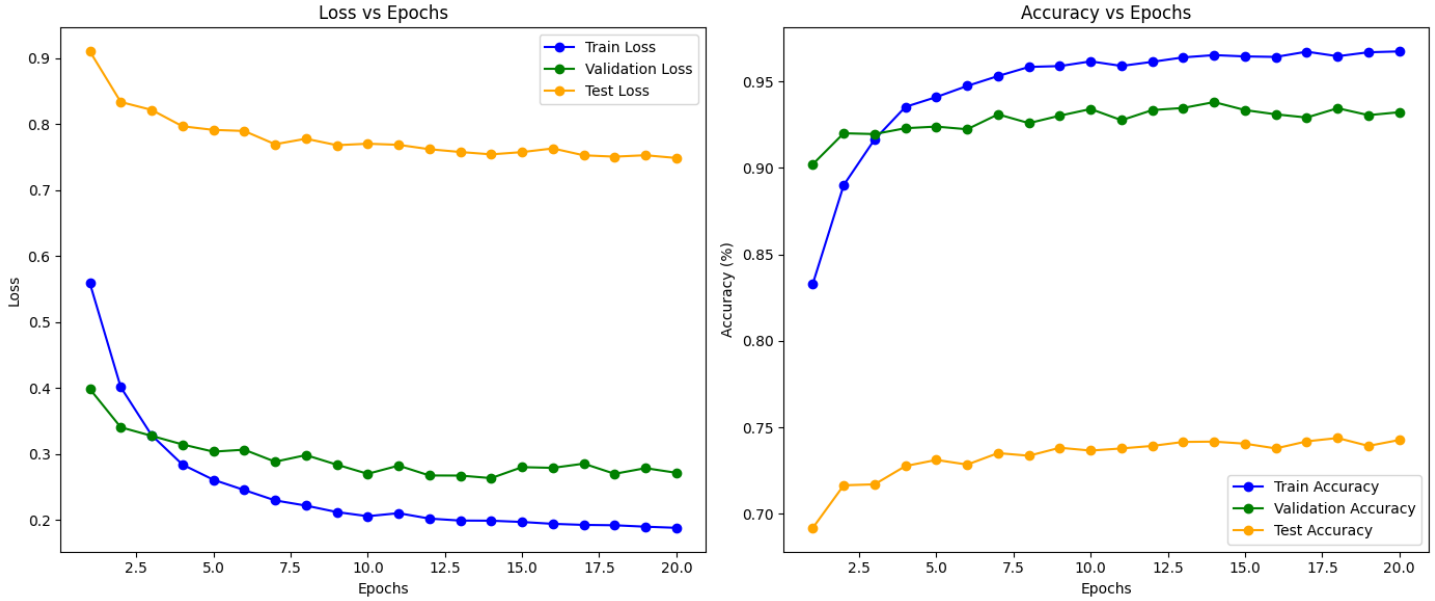

Figure 20: Our best MLP (two-hidden-layer MLP with 256 units on each hidden layer, ReLU activations, with $L2$ regularization) trained on augmented data and using early stopping.

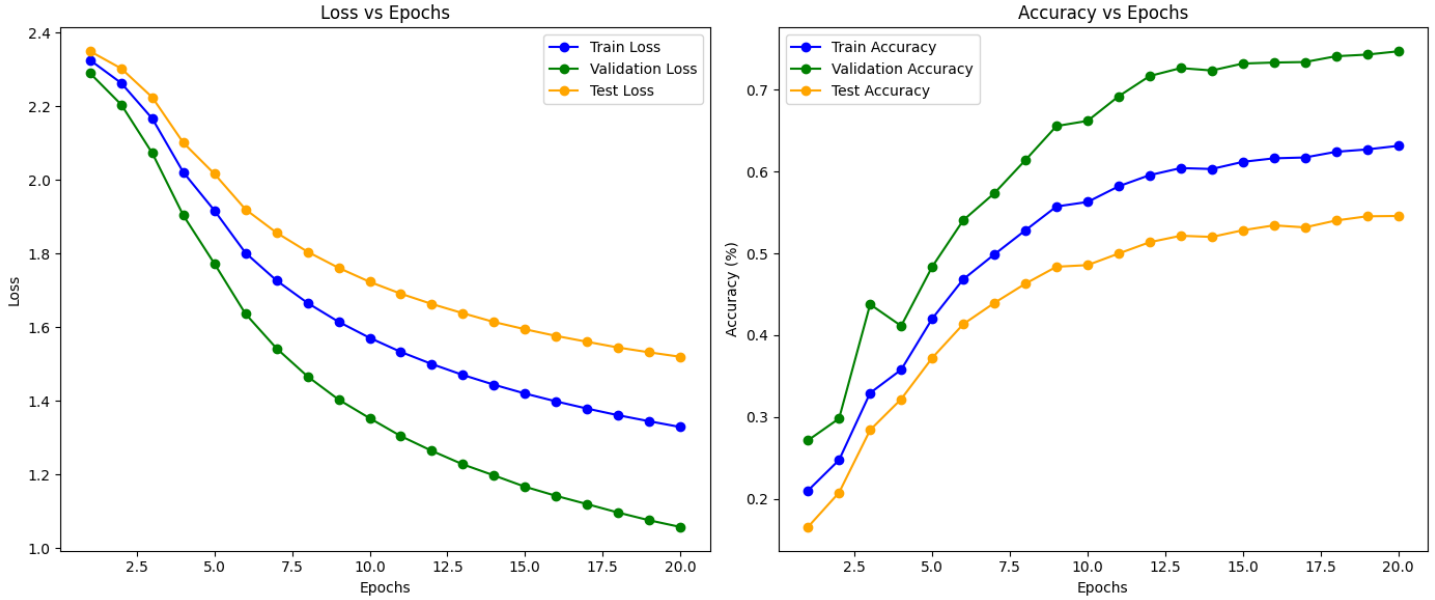

Figure 21: Training performance of a two-hidden-layer MLP with 256 units per hidden layer and ReLU activations. The plots show the accuracies and loss when training the MLP on unnormalized image data.

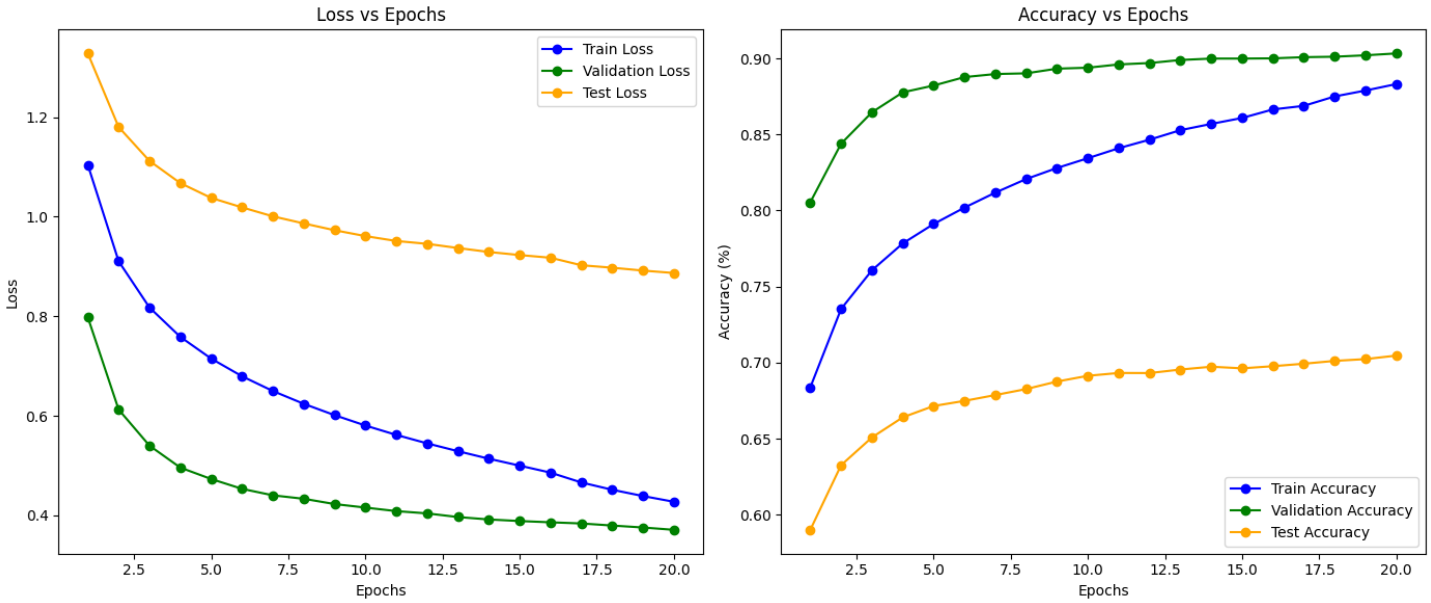

Figure 22: Performance of the MLP from (1.3) with 256 hidden units using ReLU activation. The plots show training, validation, and test accuracies and losses over 20 epochs.

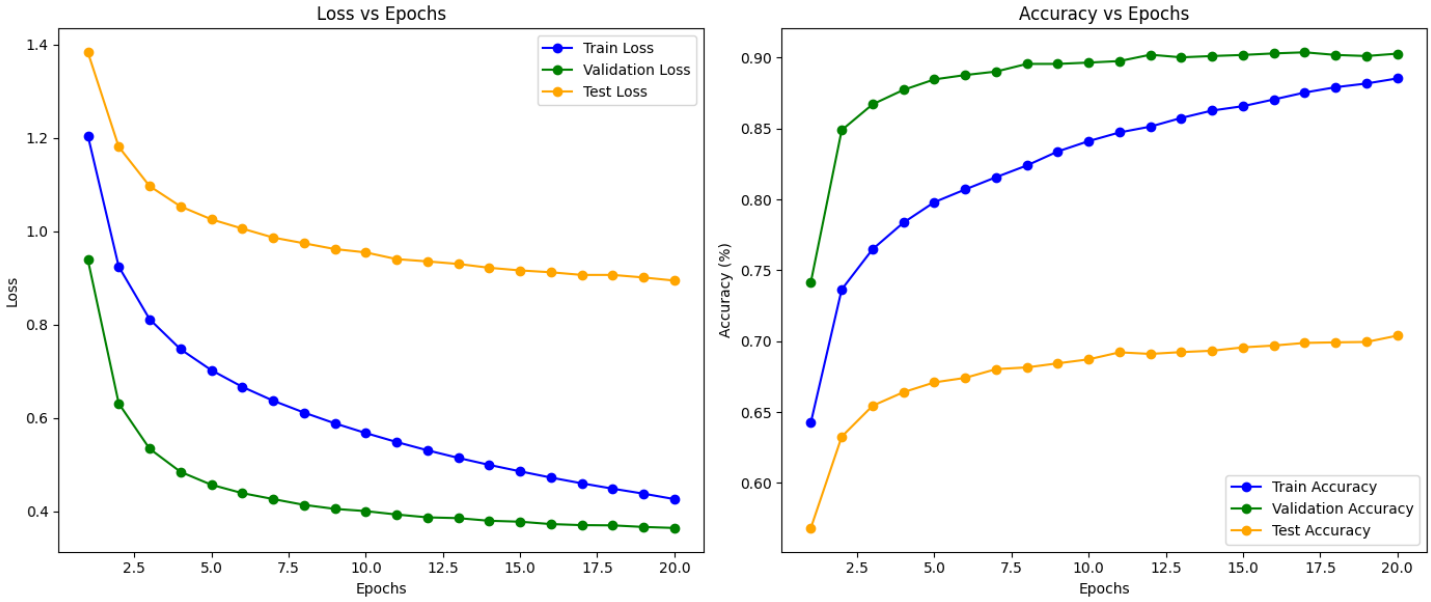

Figure 23: Performance of the MLP from (1.3) with 256 hidden units using Leaky ReLU activation. The plots show training, validation, and test accuracies and losses over 20 epochs.

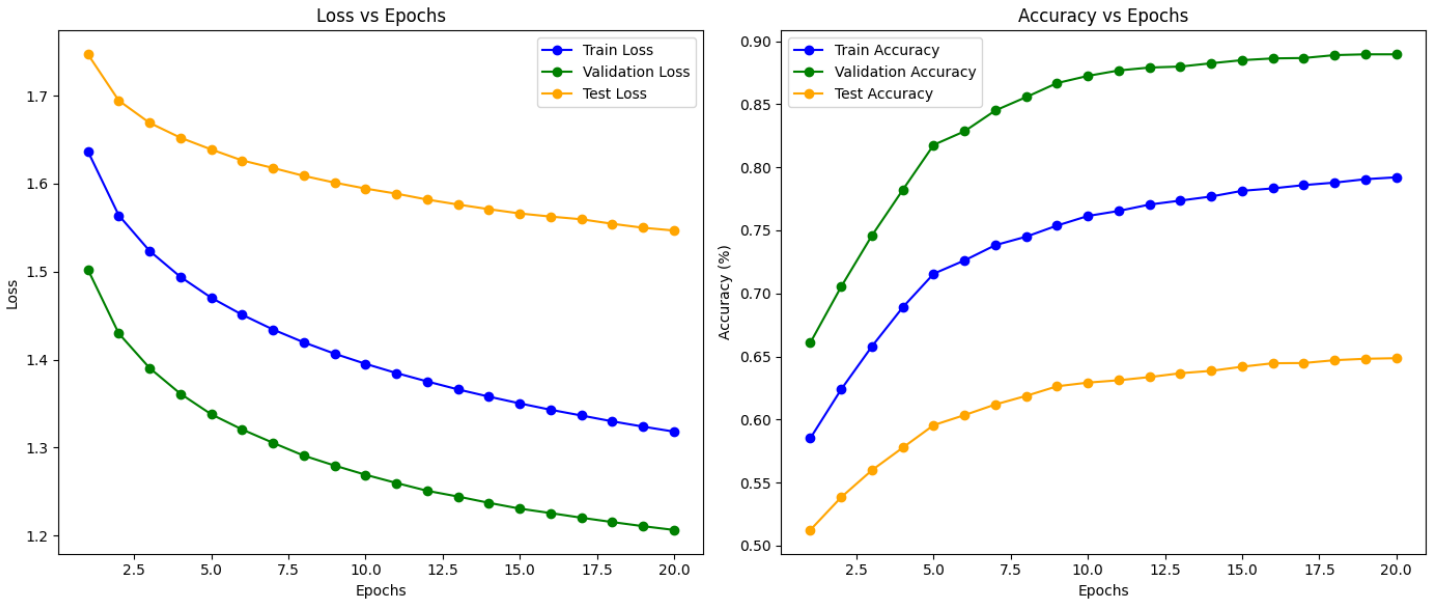

Figure 24: Performance of the MLP from (1.3) with 256 hidden units using Tanh activation. The plots show training, validation, and test accuracies and losses over 20 epochs.

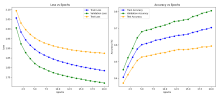

Figure 25: Performance of the MLP with 2 layers of 256 units with sigmoid activation function. The plots show training, validation, and test accuracies and losses over 20 epochs.

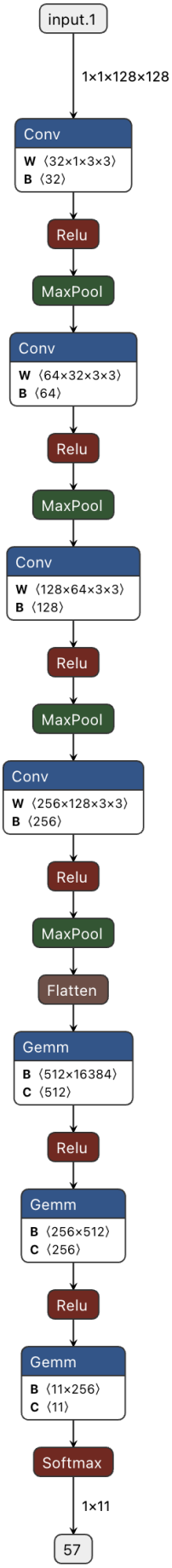

Figure 26: Graph representation of our improved CNN model. This model architecture was designed to optimize feature extraction and classification for the given dataset.