Assignment 3

An Ni Xu, Jia Song , YiWen Zhang

1 Abstract

Neural networks models are useful in recognizing patterns and image classification, which belongs to deep machine learning. In this assignment, we will implement MLP (multilayer perceptron) from scratch to classify sign language alphabet dataset - in format of image - from Sign Language MNIST (Modified National Institute of Standars and Technology). Such model will be implemented with different numbers of hidden layers (none, one, and two), where the model with two hidden layers will be further tested using three different activation functions and L2 regularization. Our results indicated that MLP with one hidden layers, using ReLU activations performed the best with a testing accuracy of approximately 71.4% at 256 hidden units. We equally noticed that increasing the number of hidden units increases the accuracy. However, different activation functions (ReLU, Leaky ReLU, Sigmoid) all performs equally. To compare the performance of our model, we further create a CNN (Convolutional Neural Network) using built-in library Tensorflow and Keras to compare its test accuracy with our implemented MLP. As a result, CNN, with a test accuracy of 88.58%, performs better than our MLP since CNN takes tensors as input, which allows it to better understand spatial relationship among pixels within complex images such as sign language images we have in our assignment.

2 Introduction

This assignment consists of implementing MLP (with high accuracy) from scratch to classify image data as well as creating CNN and compare them. The image dataset we will be using is the original MNIST dataset, which is a collection of handwritten digits (0 to 9), which was used for imaged based machine learning. Scientists indeed wanted to develop a more complicated and challenging dataset based on this original dataset as to investigate machine learning behavior and to make it useful for daily life. For such purposes, the American Sign Language letter dataset was created. It consists of 24 classes where each data point/image in training and testing sets are injectively mapped to a letter. To evaluate the performance and quality of the models, we plot the CE loss and the accuracy for each model in 4 different amount of hidden units (32, 64, 128, 256) and 3 different layers (none, one, and two). We predict that MLP with two hidden layers using 256 units to be the best performing model among these all, as we have learnt that as the number of hidden layer and unit increases, the accuracy should increase as well. However, our results have shown that the optimal model is MLP with one hidden layer using 256 hidden units, with the highest testing accuracy of 71.4%. This was possibly due to overfit which will be discussed later in the conclusion. Furthermore, we will implement MLP with two hidden layers using 256 hidden units with different activation functions: ReLU (which is used by default by our control MLP model), Leaky-ReLU and sigmoid. Since the choice of activation function in the hidden layer will control how well MLP model learns the training dataset, we predict that there will be differences in accuracies (even if it is subtle), where a subtle difference was later noticed in results. Then, we were asked to implement MLP with L2 regularization. Such implementation has goal to reduce the chances of model overfitting and therefore keeping the same if not an increased testing accuracy. We will as well implement and compare the CNN model with our implemented MLP models. In order to have the best CNN model we proceed by hyperparameters tunning.

3 Datasets

1. Sign Language MNIST is a large database of sign language/hand gesture that deaf and dumb people used to communicate. It includes only letters of the alphabet in upper cases. This database contains 27455 cases of training data and 7172 cases of testing data.
2. The Sign language MNIST dataset consists of images of hand gesture digits belongs to 24 classes of letters. Each alphabetic letter A to Z are represented by a label between 0-25, excluding cases for 9=J and 25=Z as those are gesture motions. Each image is composed of a matrix of 28x28 pixel which gives us 784 pixels. The data is indeed a table which contains information on each of the pixels, as if the picture is laid into a flat line.
3. Before we proceed to classifying the pixels, we vectorize the data to have the appropriate dimensions and then normalized it for better evaluation - normalization can convert it to floating-point data in the range 0-1 which put them into the same scale and to have zero mean, which will help for better convergence.
4. Madhav Mathur has posted an implemented CNN on the same dataset. His model achieved 100% accuracy by using Keras. To avoid the problem of overfitting, he used data augmentation by applying small transformations such as directional flips.

4 Results

In this experiment, we investigated three different MLP models having ReLU activations: with no hidden layer, one hidden layer and two hidden layers. We experimented each model using 32, 46, 128, and 256 hidden units and choose the best hidden units for our model evaluation. From the results below, it is evident that the MLP with one hidden layer and 256 hidden units performed the best, achieving a training accuracy of 95.67%, a validation accuracy of 96.14%, and a testing accuracy of 71.4%.

4.1 MLP with Variate Hidden Layers

1. we first run an MLP with no hidden layer, as we could observe in the graphs below, the model has a testing accuracy of 69.3%, training and validation accuracy of approximately 92%. The high training and validation accuracy could be due to potential overfitting, which we will address on later.

MLP with No Hidden Layer

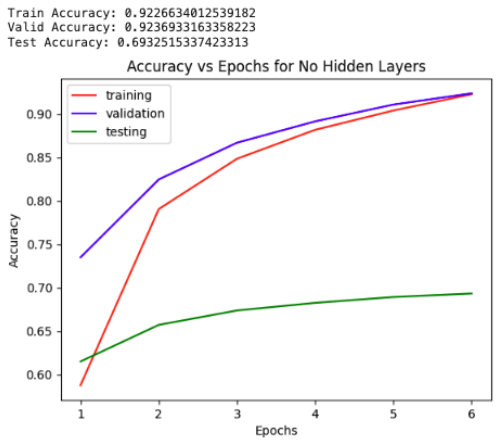

Figure 1: Accuracy

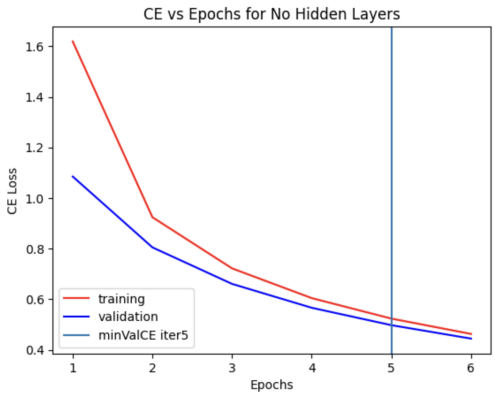

Figure 2: CE loss

2. We then ran an MLP with one hidden layer. As depicted in the summary table below and as expected, the model's performance improves as the number of hidden units increases. It achieves a testing accuracy of

71.4% and a training accuracy of 95.7% using 256 hidden units. Despite the disparity between testing and training accuracies, we did not observe overfitting in the CE vs Epochs graphs.

MLP with One Hidden Layer

hidden units	Training Accuracy (%)	validation Accuracy (%)	Test Accuracy (%)
32	84.8233	86.2502	63.9292
64	94.3115	94.7186	68.4746
128	95.5307	96.3213	69.4925
256	95.6720	96.1391	71.4027

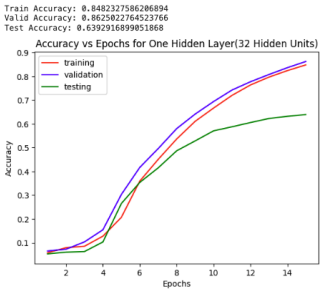

Figure 3: 32_Units_Accuracy

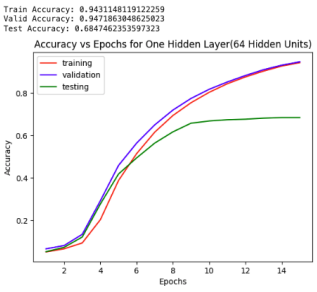

Figure 4: 64_Units_Accuracy

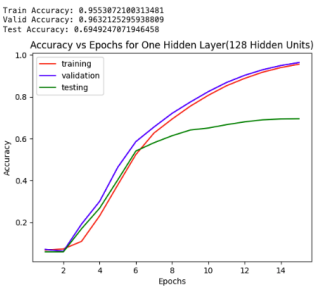

Figure 5: 128_Units_Accuracy

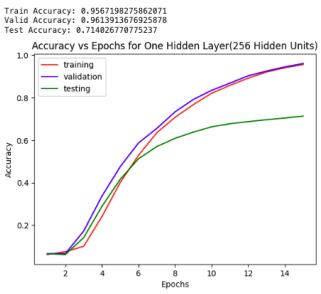

Figure 6: 256_Units_Accuracy

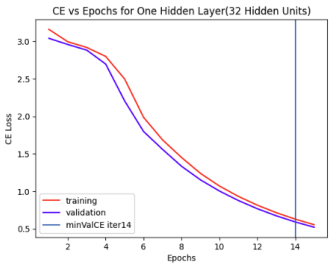

Figure 7: 32_Units_CE

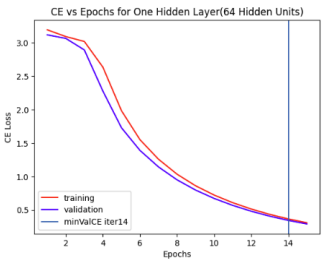

Figure 8: 64_Units_CE

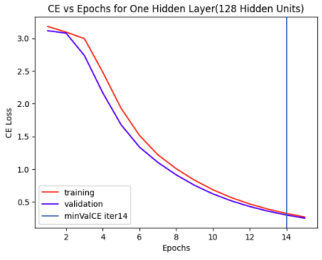

Figure 9: 128_Units_CE

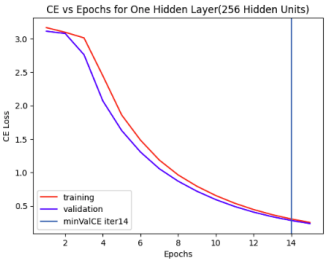

Figure 10: 256_Units_CE

3. Lastly, we ran an MLP with two hidden layers. Initially, we expected this model to be the best one, since performance should be improved with increasing hidden layers. However, test accuracies obtained are quite low as their best is around 66.57%, which are lower than the best test accuracies of MLP with one hidden layers of around 71.4% and the test accuracy of MLP with no hidden layers: 69.3%.

MLP with Two Hidden Layers

hidden units	Training Accuracy (%)	validation Accuracy (%)	Test Accuracy (%)
32	78.0454	76.4524	50.7808
64	98.0661	98.3974	61.7540
128	99.5496	99.6722	60.4155
256	99.8587	99.8179	66.5784

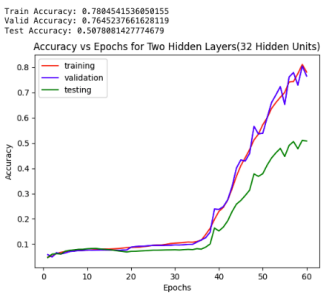

Figure 11: 32_Units_Accuracy

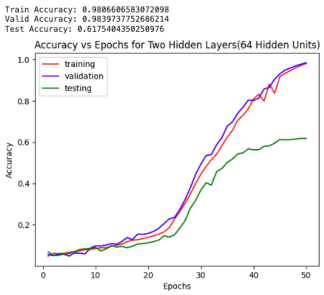

Figure 12: 64_Units_Accuracy

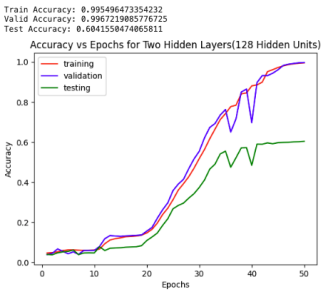

Figure 13: 128_Units_Accuracy

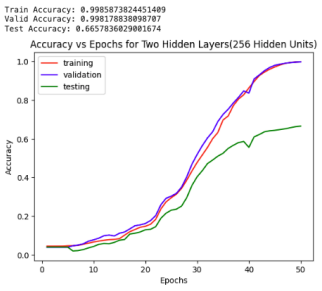

Figure 14: 256_Units_Accuracy

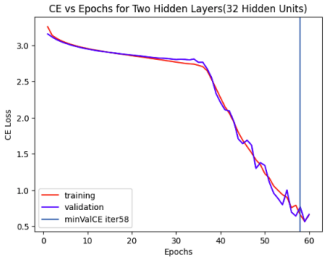

Figure 15: 32_Units_CE

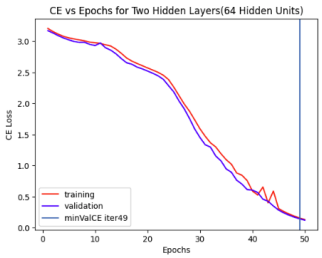

Figure 16: 64_Units_CE

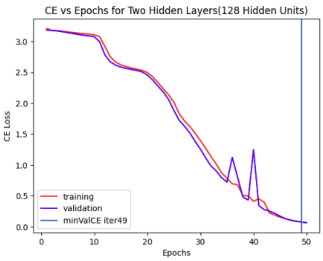

Figure 17: 128_Units_CE

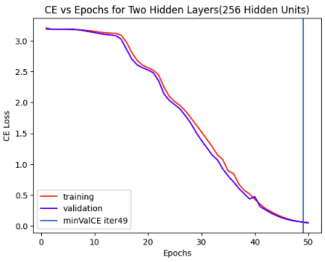

Figure 18: 256_Units_CE

4.2 MLP with Two Hidden Layers with Different Activation Functions

We take the MLP with two hidden layers using 256 hidden units with ReLu, Leaky ReLU, and Sigmoid activation functions to compare their test accuracies. The results shows that all three activation functions gives similar test accuracies of approximately 65%. However, in practice, ReLU and Leaky ReLU functions are prefered since Sigmoid function may have vanishing gradient problem as the derivative of the Sigmoid function in the positive and negative saturation regions will be close to 0. The results is the same as we expected.

Activation Function	Test Accuracy (%)
ReLU	66.5784
Leaky ReLU	64.6960
Sigmoid	63.1484

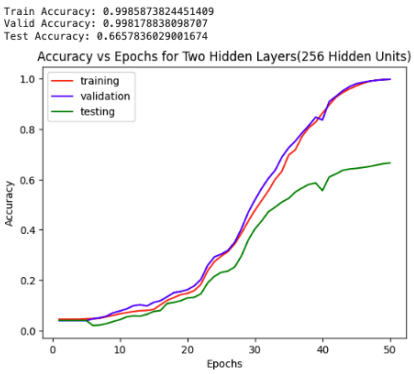

Figure 19: Accuracy_ReLU

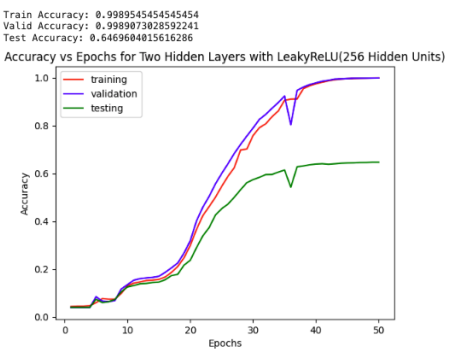

Figure 20: Accuracy_Leaky_ReLU

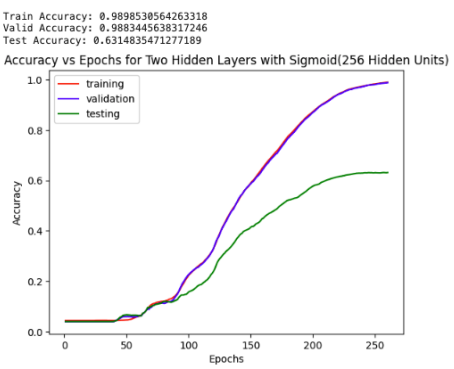

Figure 21: Accuracy_Sigmoid

4.3 MLP with L2 Regularization

L2 regularization is known to improve accuracy of training models. Therefore, we will investigate whether L2 regularization could improve the test accuracy of our implemented MLP model. The graph below shows how different values of λ affect test accuracy of our MLP with 2 hidden layers having 256 hidden units. In fact, different values of λ affect the test accuracy. As we can see below, λ=5 has the best test accuracy of approximately 65%, and λ=0.5 has the worst test accuracy of approximately 55%. However, L2 regularization in our case does not improve the test accuracy that much since it gives similar test accuracies as the 2 hidden layers MLP without L2 regularization which also has test accuracies of around 65%.

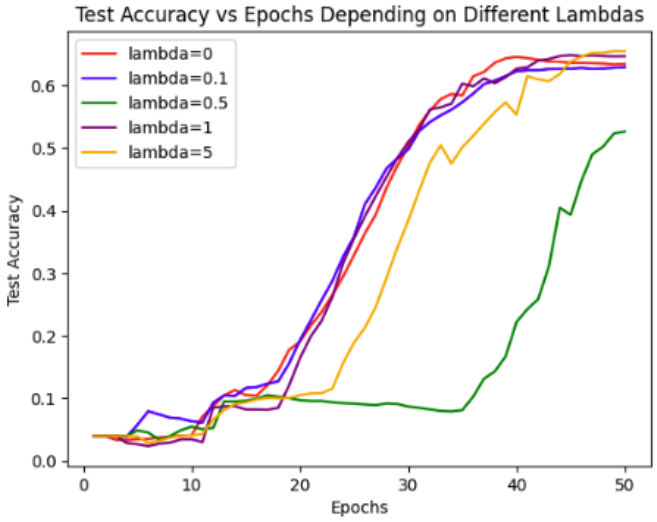

Figure 22: Accuracy_vs_Epochs_Lambdas

4.4 ConvNet

4.4.1 Hyperparameters Tunning

In order to have the best CNN model, hyperparameters tunning is an important step. To do this, we first define some default values for all hyperparameters and test different values on a chosen hyperparameter. The plots belows shows how different choices of kernel size, stride size, pooling size, and dropout rate affect validation/test accuracy of CNN model. In fact, based on these accuracy plots, the best units size is 256, the best kernel size is 6, the best stride size is 1, and the best dropout rate is 0.25. With these chosen hyperparameters, we will then be able to build the best CNN model.

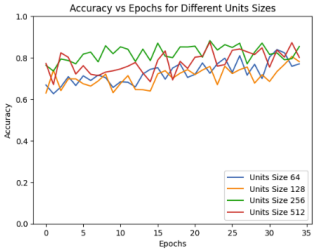

Figure 23: Accuracy_-Units

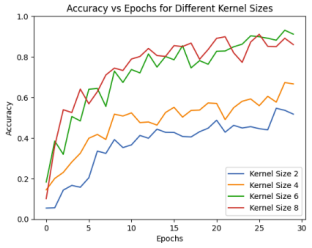

Figure 24: Accuracy_-Kernel

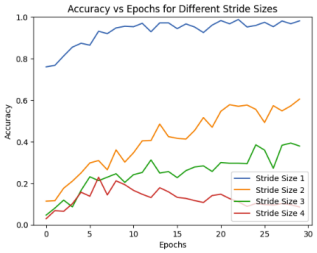

Figure 25: Accuracy_-Stride

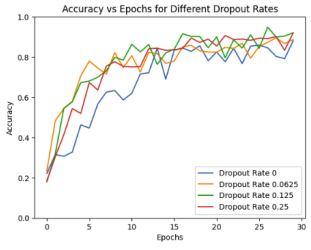

Figure 26: Accuracy_-Dropout

4.4.2 ConvNet with Best Hyperparameters

The plots below shows us the best ConvNet model with the best hyperparameters chosen previously using hyperparameters tunning. Fortunately, this ConvNet model is very efficient since it reaches a test accuracy of 88.58 %, which is also better than our best MLP model with one hidden layer which only has a test accuracy of 71.4%.

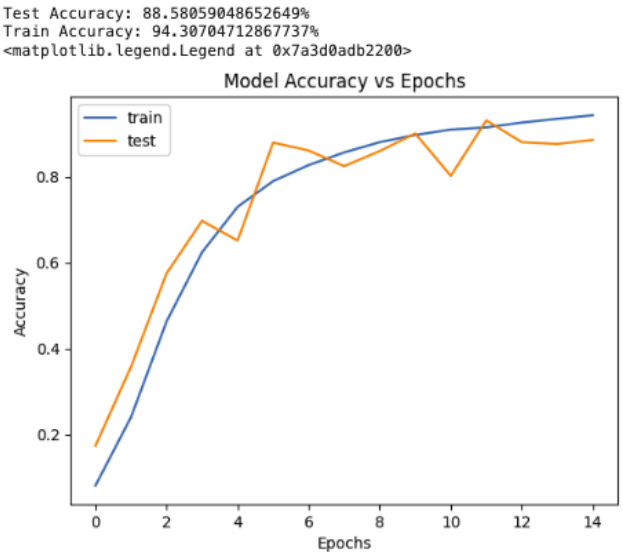

Figure 27: Accuracy_vs_Epochs_CNN

5 Discussion and Conclusion

5.1 Numbers of Hidden Layers in MLP

In our implemented MLP, the number of hidden layers plays crucial roles. As we could notice from the result obtained on the previous page, MLP with one hidden layer performs the best among the rest(no hidden layers and two hidden layers). This model could reach a test accuracy of 71.4%, which is the highest. MLP with no hidden layers with a test accuracy of 69.3% performs slightly worse than the MLP with one hidden layer but performs better than the MLP with two hidden layers having test accuracies of approximately 65%. Therefore, increasing number of hidden layers may or may not improve accuracy of a MLP model. In other words, increasing the number of hidden layers much more than the sufficient number of layers will cause test accuracy to decrease.

5.2 Numbers of Hidden Units in MLP

As we can see from plots of previous pages, increasing number of hidden units in the MLP could improve test accuracy of the MLP model. This improvement in test accuracy applies to MLP models with one hidden layer, and with two hidden layers. As we increase hidden units from 32 to 256, test accuracy of MLP with one hidden layer goes from 63.9% to 71.4%, which is approximately 7.5% improvement. Similarly, MLP with two hidden layers has test accuracy improved from 50.78% to 66.57%, as hidden units increases, which is approximately 16% improvement.

5.3 Activation Functions in MLP

Activation functions ReLU, Leaky ReLU, and Sigmoid, all gives similar results and test accuracies for MLP with two hidden layers. All three models has test accuracies that are approximately 65%, though ReLU(64.69%) and Leaky ReLU(66.58%) have slightly better accuracies than Sigmoid which has test accuracy of 63.15 %. ReLu

and Leaky ReLU activation function performs slightly better than Sigmoid thanks to its ability to avoid vanishing gradient problem. ReLU function performs almost equally as the Leaky ReLU function, which make sense since Leaky ReLU function is just an extension of Relu function. However, in general, ReLU function is the prefered function used to train models since it provides simpler structure and faster computational time.

5.4 MLP with L2 Regularization

L2 regularization in many cases is used to improve test accuracy of models. In this case, we use MLP with two hidden layers to evaluate our models effectiveness. However, from our plot result obtained on section 4.3, we see that there is no improvement in test accuracy with L2 regularization. In fact, two hidden layers MLP with or without L2 regularization have similar test accuracies of around 65%. So, L2 regularization is not necessary for training our MLP model dedicated to sign language images classfication.

5.5 ConvNet VS MLP

ConvNet model has a test accuracy of 88.58%, which is better than our best MLP model with one hidden layer that has a test accuracy of 71.4%. While both MLP and ConvNet model can be used for image data classification, CNN models operate on tensors, allowing them to better comprehend the spatial relationships among pixels within images. Therefore, for complex images such as signs language images used in our assignment, ConvNet model are expected to perform better than MLP due to their enhanced ability to capture intricate spatial patterns.

5.6 Summary

We do notice an increase in accuracy when the numbers of hidden layers and/or units were increased. However, increasing number of hidden layers can lead to overfit and thus decrease in test accuracy. Regarding changing of activation functions and L2 regularization, we did not see them having significant enough effect on model testing accuracies. Lastly, we notice that CNN performed better than our two hidden layer MLP with 256 hidden units.

5.7 Further Studies

5.7.1 Overfitting

In our MLP model, we did notice overfitting as we saw an decrease of test accuracy as we increased the number of hidden layers. Here are some possible ways to reduce the chances of overfitting:

- L2-Regularization and changing of activation functions: this was part of the assignment, however, we did not observe significant improvement of the overfit after using weight decay.
- Early stopping: this technique monitors the validation loss during training and stop the training process of the model right before the model begins to overfit. To implement this, we need to set aside some validation data from the training set, where the stopping criteria will be when the classification accuracy of validation set stopped improving (regardless of loss function performance).

5.7.2 Improving Performance of MLP

As the results have shown, there is still room for improvement for our mlp model. Here are some other possible adjustments we can experiment to increase model accuracy:

- Optimizing hyperparameters:
  - Learning Rate: we can try out different learnng rates to see which one converges faster. We did experiment with 2 other learning rates for some models, however, the runtimes were tremendous, which is why we did not continue experimenting with it.

  - Batch Size: we could try using smaller batch size, which can help with generalization, though may lead to more noise features.
  - Epochs: try training with different numbers of epochs and stop training when overfitting starts to occur. We did experiment a little with this hyperparameter, where we observed increase in test accuracy as the number of epochs increased.
- Data Augmentation: perform data alternation (rotation, flipping, translation, etc) to increase data diversity and therfore imprive generalization.

5.7.3 Originality: Use of CNN on Breast Cancer

Recalling that in our first assignment, we were asked to classify data points of a breast cancer dataset using KNN and DT. We think that it would be interesting to use CNN to classify such data. The biggest advantage of CNN over KNN or DT is its capability of classifying images of cancers better which can lead to more accurate classification. This implies that less data pre-processing/processing is needed, as extra help in feature identifications will not be needed and will be inclusively identified and analysed by CNN.

6 Statement of Contributions

- Jia Song: Note book sections (code organiser and editor), report reviser and editor.
- An Ni Xu: Main code writer, report editor and writer(content check).
- Yi Wen Zhang: Main report writer and reviser, code writer procedures ideas provider.

7 References

CS231N Convolutional Neural Networks for Visual Recognition, https://cs231n.github.io/neural-networks-2/#datapre. Accessed 27 Mar. 2024.

"Google Colaboratory." Google Colab, Google, https://colab.research.google.com/github/yueliyl/comp551-note
blob/master/AutoDiffMLP.ipynb#scrollTo=aPhdztjmaiDr. Accessed 13 Mar. 2024.

"Google Colaboratory." Google Colab, Google, https://colab.research.google.com/github/yueliyl/comp551-note
blob/master/NumpyDeepMLP.ipynb. Accessed 13 Mar. 2024.

"Google Colaboratory." Google Colab, Google, https://colab.research.google.com/github/yueliyl/comp551-note
blob/master/MLP.ipynb. Accessed 13 Mar. 2024.

Sayakdasgupta. "Sign-Language Classification CNN (99.40% Accuracy)." Kaggle, Kaggle, 24 Apr. 2020, https://www.kaggle.com/code/sayakdasgupta/sign-language-classification-cnn-99-40-accuracy. Accessed 13 Mar. 2024.

"Scipy.Optimize.Check_grad#." Scipy.Optimize.Check_grad - SciPy v1.12.0 Manual, https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.check_grad.html. Accessed 13 Mar. 2024.

Tecperson. "Sign Language Mnist." Kaggle, 20 Oct. 2017, https://www.kaggle.com/datasets/datamunge/sign-language-mnist/data. Accessed 13 Mar. 2024.

"3.2. Tuning the Hyper-Parameters of an Estimator." Scikit, https://scikit-learn.org/stable/modules/grid_search.html. Accessed 13 Mar. 2024.