CS6140 Assignment 4

Haotian Shen
Qiaozhi Liu

Abstract

In this project, we explore deep learning tools for both visual and non-visual tasks. We first built a convolutional neural network (CNN) for the MNIST digit recognition task, visualizing the network architecture and performance. We then experiment with different network variations to optimize performance and training time, potentially using the MNIST Fashion dataset for a more challenging task. We develop a plan to evaluate multiple network dimensions, formulate hypotheses on their behavior, and analyze the results.

We further apply transfer learning to a pre-trained MNIST network for recognizing Greek letters (alpha, beta, and gamma) by freezing the network weights and replacing the last layer with a new Linear layer with three nodes. We evaluate the training error and the required epochs for perfect identification using the provided Greek letter images.

Finally, we build two different networks to predict heart disease using the same dataset from a previous project, avoiding convolutional layers and focusing on the computational requirements and performance of the networks. We compare the performance of the new networks to the previous project and discuss the network architectures and their computational requirements.

Reflection

Throughout this project, I learned about various aspects of deep learning, specifically the design, implementation, and optimization of neural networks. I gained experience with convolutional neural networks (CNNs) and their application to visual tasks, such as digit and fashion item recognition. I also discovered the importance of experimenting with different network architectures and hyperparameters to improve performance and training efficiency.

Transfer learning proved to be a valuable technique, enabling me to leverage pre-trained networks for new tasks, such as recognizing Greek letters. I learned how to freeze network weights and modify the output layer to adapt the network to a new problem.

Additionally, I applied artificial neural networks (ANNs) to non-visual tasks, such as heart disease prediction. This experience helped me understand the importance of selecting appropriate network architectures based on the task at hand and considering the computational requirements of each network.

Task 1 MNIST Tutorial

Neural Network Depiction

```
Net(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
  (conv2_drop): Dropout2d(p=0.5, inplace=False)
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=320, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
)
```

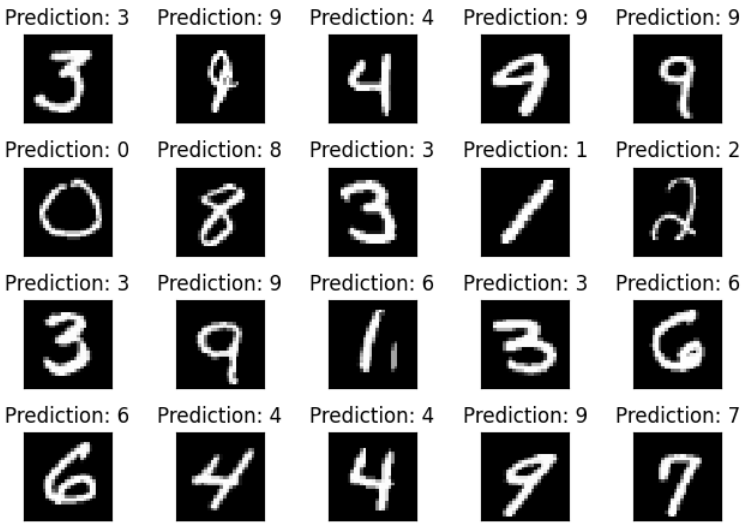

Figure 1: Digit picture and their predicted labels using 3 epochs

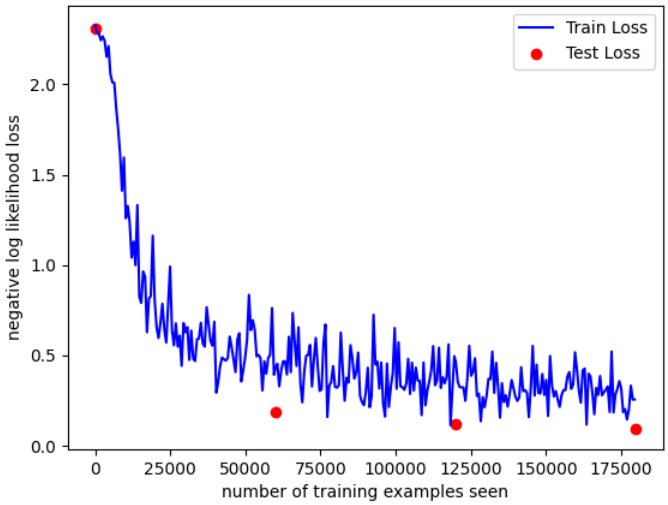

Figure 2: Testing and training error for 3 epochs

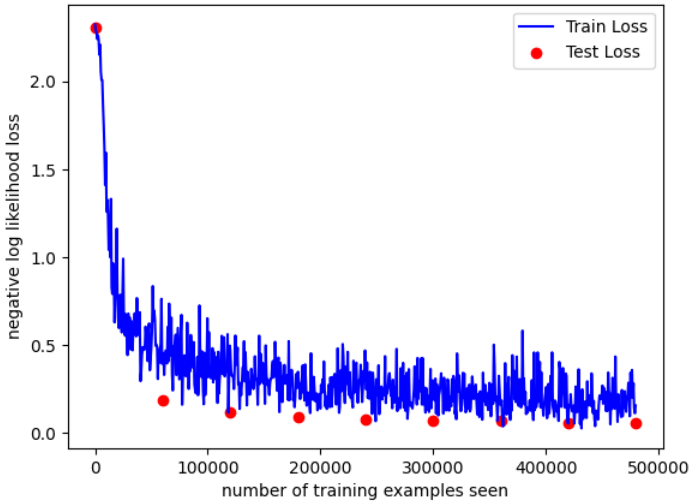

Figure 3: Testing and training error for 8 epochs

Task 2 Experiment with Network Variations

Initially, just train for 3 epochs:

The network setting is:

Net(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
  (conv2_drop): Dropout2d(p=0.5, inplace=False)
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=320, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=10, bias=True)
)

Test set: Avg. loss: 0.6507, Accuracy: 7473/10000 (75%)

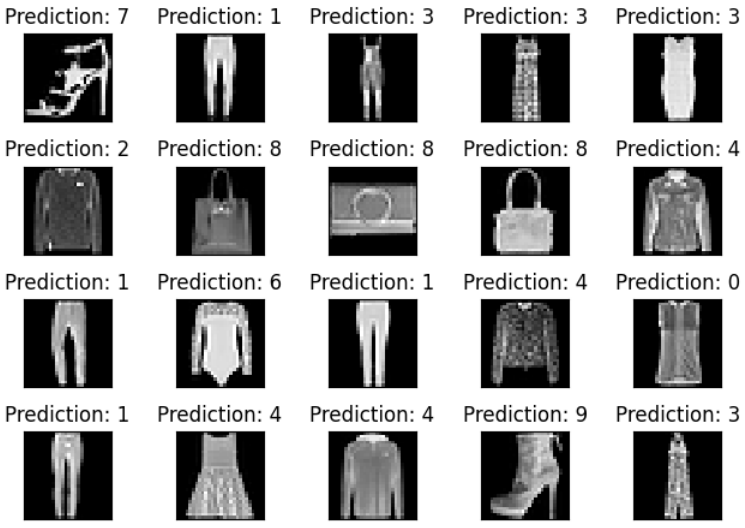

Figure 4: Fashion picture and their predicted labels using 3 epochs

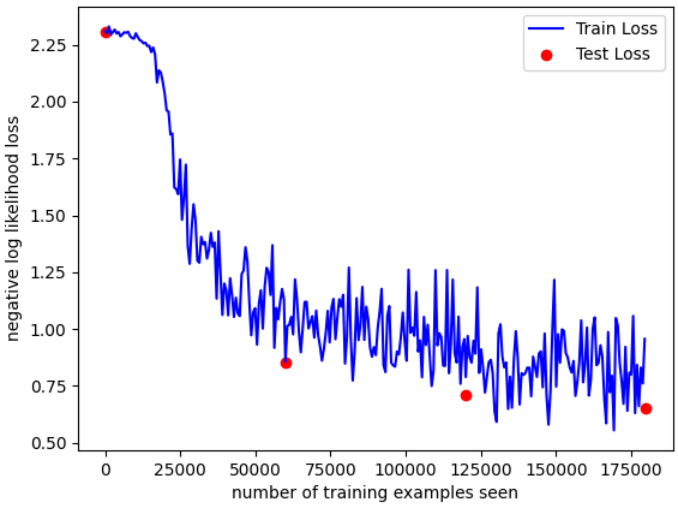

Figure 5: Testing and training error for 3 epochs

Now, we need to modify some hyperparameters in the network with the goal to improve performance.

Develop a plan

We will continue to use 2 convolution layers because after some proof of concept, we determine that using neither 1 nor 3 layers yields a better result compared to using 2 convolution layers.

The 3 aspects we are trying to explore are:

1. Size of the convolution filters
    a. Use kernel_size 2x2
    b. Use kernel_size 3x3
    c. Use kernal_size 4x4
    d. Use kernel_size 5x5
2. Number of convolution filters
    a. Use 1 to 8, and 8 to 16
    b. Use 1 to 16, and 16 to 32
    c. Use 1 to 32, and 32 to 64
3. Dropout rate
    a. 0.1
    b. 0.25
    c. 0.5

This will create a total of 36 variations to experiment.

After we determine the optimal combination of these 3 aspects, we will then variate the number of epochs of training to determine the best test accuracy we can achieve:

4. Number of epochs of training: 30

Predict the results

Size of the convolution filters:

The size of the convolution filter determines the receptive field of the network and can affect the types of features that the network can learn. Larger filter sizes can help capture more global features, while smaller filter sizes are better at capturing local features. Choosing the right filter size depends on the task and the characteristics of the input data. Using a filter size that is too small may not capture the desired patterns, while using a filter size that is too large can lead to overfitting. In our Fashion dataset, local details are important features to determine the specific category of the apparel, so

my prediction is that using a smaller kernel_size will give a better result. However, too small of kernel_size may focus too much on the local details and ignore the whole image.

Number of convolution filters:
The number of filters in each convolution layer determines the number of feature maps that are generated. Increasing the number of filters can help the network learn more complex representations of the input data. However, increasing the number of filters can also increase the computational cost of the network, and too many filters can lead to overfitting. In our Fashion dataset, my prediction is that the test accuracy may increase initially with the number of convolution filters but will then decrease.

Dropout rate:
The dropout rate affects the model's capacity to learn the underlying patterns in the data. A low dropout rate may lead to overfitting, where the model becomes too complex and is unable to generalize to new data. On the other hand, a high dropout rate may lead to underfitting. In our Fashion dataset, my prediction is that using a dropout rate lower than the default 0.5 may potentially yield a better result since the dataset is relatively small. Dropping out too many samples may lead to underfitting.

Execute my plan

Initially, I have automated the process and tested all the combination of these parameters using only 3 epochs, and the best combination which gives the highest test accuracy is below:
{'num_conv_layer': 2, 'kernel_size': 2, 'num_init_conv_filter': 32, 'dropout': 0.25}
The corresponding test accuracy is 81.55%.

Then, we will use this combination and execute the training process for 30 epochs to see if the test accuracy gets improved. After training for 30 epochs, we achieved 90% test accuracy, and the test loss did not show a sign of reaching a plateau, which means that the test accuracy can still get improved if training for more epochs.

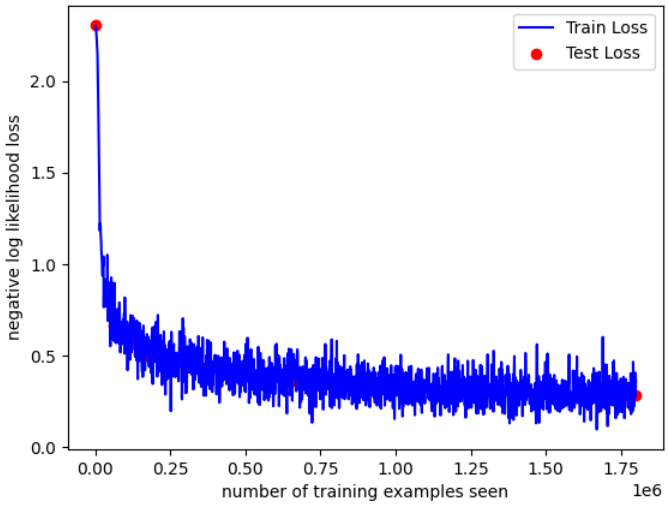

Figure 6: Testing and training error for 30 epochs on the Fashion MNIST dataset.

Compared to the initial test accuracy of 75%, the improvement to 90% test accuracy is very significant. This also indicated that manipulating the hyperparameter can significantly improve our model performance. Below is a picture with their predicted label using the updated model after 30 epochs of training and with the best tested hyperparameters.

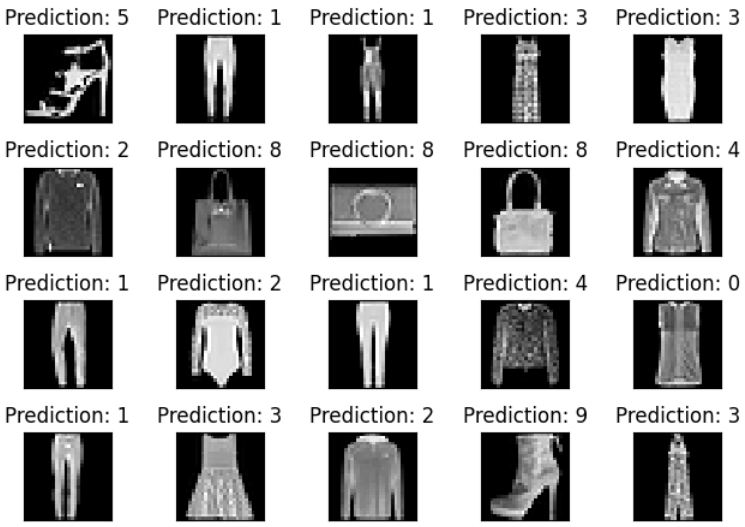

Figure 7: Fashion picture and their predicted labels using 30 epochs and best hyperparameters.

We can tell there are a lot of changes in the predicted label compared to what the initial image showed, and the result is more intuitive because our model learns more complex features of the clothing and generalizes them better.

The updated model setup was below:

```
CustomNet(
  (conv1): Conv2d(1, 32, kernel_size=(2, 2), stride=(1, 1), padding=(1, 1))
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(32, 64, kernel_size=(2, 2), stride=(1, 1), padding=(1, 1))
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2_drop): Dropout2d(p=0.25, inplace=False)
  (fc1): Linear(in_features=3136, out_features=522, bias=True)
  (fc2): Linear(in_features=522, out_features=10, bias=True)
)
Test set: Avg. loss: 0.2853, Accuracy: 8951/10000 (90%)
```

Analysis

The changes are increasing the number of convolution filters from 10 to 32, decreasing the dropout rate from the default 0.5 to 0.25, and decreasing the size of the convolution filters from 5 to 2.

Potential reasons:

1. Increasing the number of convolution filters from 10 to 32: By increasing the number of filters in the convolutional layers, the model can learn more complex and diverse features from the input images. This increased capacity allows the network to capture more fine-grained patterns in the input data, which can improve its ability to discriminate between different classes of images.
2. Decreasing the dropout rate from 0.5 to 0.25: Dropout is a regularization technique that can help to prevent overfitting in deep neural networks. However, too much dropout can also harm the performance of the network by suppressing important features during training. By decreasing the dropout rate, we allowed the model to retain more useful information during training, which can lead to better generalization and improved test accuracy.
3. Decreasing the size of the convolution filters from 5 to 2: Smaller filters can capture finer-grained features in the input images, allowing the model to learn more localized patterns. This can lead to more precise feature representations, which can improve the model's ability to distinguish between different classes of images.

Task 3 Transfer Learning on Greek Letters

Training results for 8 epochs:
Test set: Avg. loss: 1.0631, Accuracy: 12/27 (44%)
Train epoch: 1 [0/27 (0%)] Loss: 0.994387
Test set: Avg. loss: 0.2429, Accuracy: 23/27 (85%)
Train epoch: 2 [0/27 (0%)] Loss: 0.713438
Test set: Avg. loss: 0.0853, Accuracy: 27/27 (100%)
Train epoch: 3 [0/27 (0%)] Loss: 0.351910
Test set: Avg. loss: 0.0683, Accuracy: 27/27 (100%)
Train epoch: 4 [0/27 (0%)] Loss: 0.119621
Test set: Avg. loss: 0.0523, Accuracy: 27/27 (100%)
Train epoch: 5 [0/27 (0%)] Loss: 1.615111
Test set: Avg. loss: 0.0688, Accuracy: 27/27 (100%)
Train epoch: 6 [0/27 (0%)] Loss: 0.081540
Test set: Avg. loss: 0.0619, Accuracy: 27/27 (100%)
Train epoch: 7 [0/27 (0%)] Loss: 0.231174
Test set: Avg. loss: 0.0883, Accuracy: 27/27 (100%)
Train epoch: 8 [0/27 (0%)] Loss: 0.045315
Test set: Avg. loss: 0.0376, Accuracy: 27/27 (100%)

Starting from the 3rd epochs, the test accuracy achieved 100% and did not decrease with more epochs of training.

```
Net(
  (conv1): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
  (conv2_drop): Dropout2d(p=0.2, inplace=False)
  (fc1): Linear(in_features=320, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=3, bias=True)
)
```
Test set: Avg. loss: 0.0683, Accuracy: 27/27 (100%)

Figure 8: Greek letter pictures and their predicted label.

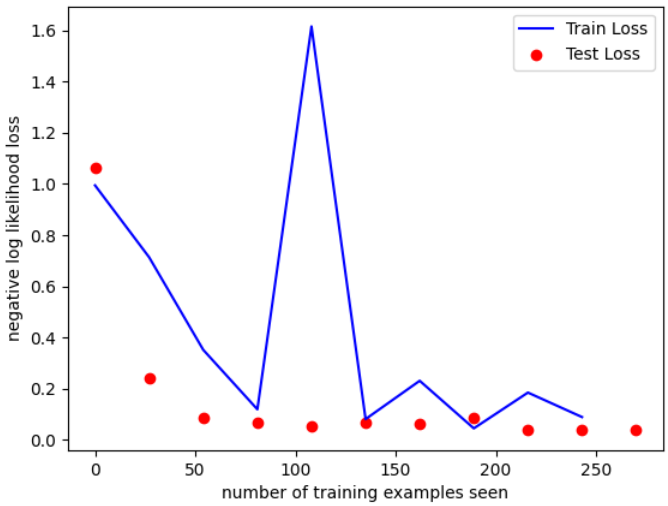

Figure 9: Testing and training error for 10 epochs using transfer learning on the greek letter dataset.

Task 4 Heart Disease Prediction Using an ANN

CNNs are specifically designed to handle spatial or temporal data, such as images, videos, or time-series data. Since the heart disease data does not have spatial or temporal structure and it is organized in grid structure, using a fully connected multilayer perceptron (MLP) is a more appropriate choice. The dataset is also very small and has low dimensionality, so a fully connected layer performs better than CNN or RNN. We will experiment on a few variations of the MLP to observe their performance.

Model Architecture and Computational Requirements

Model1:

Four fully connected layers with the first layer having an output dimension of 16. All layers use the activation function ReLU, and the last layer uses sigmoid.

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
Net1                                     [64, 1]                   --
├─Linear: 1-1                            [64, 16]                  144
├─Linear: 1-2                            [64, 8]                   136
├─Linear: 1-3                            [64, 4]                   36
├─Linear: 1-4                            [64, 1]                   5
==========================================================================
Total params: 321
Trainable params: 321
Non-trainable params: 0
Total mult-adds (M): 0.02
==========================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.01
Params size (MB): 0.00
Estimated Total Size (MB): 0.02
==========================================================================
Net1(
  (fc1): Linear(in_features=8, out_features=16, bias=True)
  (fc2): Linear(in_features=16, out_features=8, bias=True)
  (fc3): Linear(in_features=8, out_features=4, bias=True)
  (fc4): Linear(in_features=4, out_features=1, bias=True)
)
```

Model2:
Four fully connected layers with the first layer having an output dimension of 6. All layers use the activation function ReLU, and the last layer uses sigmoid.

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
Net2                                     [64, 1]                   --
├─Linear: 1-1                            [64, 6]                   54
├─Linear: 1-2                            [64, 4]                   28
├─Linear: 1-3                            [64, 2]                   10
├─Linear: 1-4                            [64, 1]                   3
==========================================================================
Total params: 95
Trainable params: 95
Non-trainable params: 0
Total mult-adds (M): 0.01
==========================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.01
Params size (MB): 0.00
Estimated Total Size (MB): 0.01
==========================================================================
Net2(
  (fc1): Linear(in_features=8, out_features=6, bias=True)
  (fc2): Linear(in_features=6, out_features=4, bias=True)
  (fc3): Linear(in_features=4, out_features=2, bias=True)
  (fc4): Linear(in_features=2, out_features=1, bias=True)
)
```

Task 4 Extension

Model3:
Four fully connected layers with the first layer having an output dimension of 16. All layers use the activation function ReLU, and each layer has a dropout rate of 0.05, and the last layer uses sigmoid.

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
Net3                                     [64, 1]                   --
├─Linear: 1-1                            [64, 6]                   54
├─Dropout: 1-2                           [64, 6]                   --
├─Linear: 1-3                            [64, 4]                   28
├─Dropout: 1-4                           [64, 4]                   --
├─Linear: 1-5                            [64, 2]                   10
├─Dropout: 1-6                           [64, 2]                   --
├─Linear: 1-7                            [64, 1]                   3
==========================================================================
Total params: 95
Trainable params: 95
Non-trainable params: 0
Total mult-adds (M): 0.01
==========================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.01
Params size (MB): 0.00
Estimated Total Size (MB): 0.01
==========================================================================
Net3(
  (fc1): Linear(in_features=8, out_features=6, bias=True)
  (dropout1): Dropout(p=0.05, inplace=False)
  (fc2): Linear(in_features=6, out_features=4, bias=True)
  (dropout2): Dropout(p=0.05, inplace=False)
  (fc3): Linear(in_features=4, out_features=2, bias=True)
  (dropout3): Dropout(p=0.05, inplace=False)
  (fc4): Linear(in_features=2, out_features=1, bias=True)
)
```

Model4:
Two fully connected layers with the first layer having an output dimension of 4. All layers use the activation function ReLU, and the last layer uses sigmoid.

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
Net4                                     [64, 1]                   --
├─Linear: 1-1                            [64, 4]                   36
├─Linear: 1-2                            [64, 1]                   5
==========================================================================
Total params: 41
Trainable params: 41
Non-trainable params: 0
Total mult-adds (M): 0.00
==========================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.00
Estimated Total Size (MB): 0.00
==========================================================================
Net4(
  (fc1): Linear(in_features=8, out_features=4, bias=True)
  (fc2): Linear(in_features=4, out_features=1, bias=True)
)
```

Model5:

Four fully connected layers with the first layer having an output dimension of 6. All layers use the activation function Tanh, and the last layer uses sigmoid.

```
==========================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
Net5                                     [64, 1]                   --
├─Linear: 1-1                            [64, 6]                   54
├─Linear: 1-2                            [64, 4]                   28
├─Linear: 1-3                            [64, 2]                   10
├─Linear: 1-4                            [64, 1]                   3
==========================================================================
Total params: 95
Trainable params: 95
Non-trainable params: 0
Total mult-adds (M): 0.01
==========================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.01
Params size (MB): 0.00
Estimated Total Size (MB): 0.01
==========================================================================
Net5(
  (fc1): Linear(in_features=8, out_features=6, bias=True)
  (fc2): Linear(in_features=6, out_features=4, bias=True)
  (fc3): Linear(in_features=4, out_features=2, bias=True)
  (fc4): Linear(in_features=2, out_features=1, bias=True)
)
```

Performance

Evaluate the performance on the test set.

Table1: Test set accuracy for all tested models.

Model1	85%
Model2	83.5%
Model3	52.5%
Model4	84%
Model5	83%

Based on the performance on the test set, we observed that Model 1 and 4 had the highest and similar performance, while Model 3 had the lowest performance. Usually, having an output dimension greater than the input dimension may cause the model to overfit, but from our observation, using a first-layer output dimension of 16 performed better than using an output dimension of 6. Using a dropout layer made the performance worse due to the small size of the training set. We need at least all the training data or potentially more to achieve better performance. Using Tanh and ReLU activation functions did not have much of a difference because we were predicting a binary classification problem, where the result labels were either 0 or 1. Using fewer fully connected layers in Model 4 did not lower the test accuracy too much, indicating that using four fully connected layers did not overfit our data.

Compared to the previous project result, using neural networks did not significantly increase the test accuracy. This could potentially be caused by the small training dataset.

Extensions

Extensions 1 and 2 were done using Google Colab to achieve faster training speed using GPU and TPU. I consulted with the professor, and he confirmed the use of this and allowed the notebook file as a submission.

Extension 1:

There is a nice ImageNet transfer learning tutorialLinks to an external site. on classifying Ants and Bees. Implement it and show the results. Refer to the notebook file extension1.ipynb for the run result.

Option 1:
Fine Tuning the convnet
Training complete in 1m 22s
Best val Acc: 0.960784

Option 2:
Convnet as fixed feature extractor
Training complete in 1m 0s
Best val Acc: 0.960784

The best validation accuracy of two options are the same. Refer to the pictures in the notebook file for the predicted label of the pictures.

Extension 2:
Create some additional test data for the greek letter task. Please include the examples as a zip file in your submission.

I have collected a dataset of all greek letters of training and test sets. The training set name is greek_train_extra, and the test set name is greek_test_extra. Please refer to the zipped data file and the notebook file extension2.ipynb for the run result.

Figure 10: Extra Greek letter pictures and their predicted label.

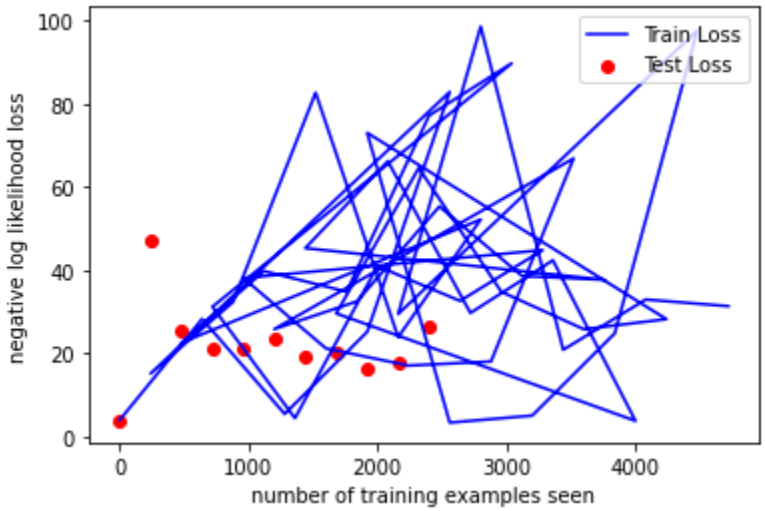

Figure 11: Testing and training error for 10 epochs using transfer learning on the extra greek letter dataset.

After 10 training epochs, the transfer learning model achieved a highest of 76% test accuracy on all greek letters.

Extension 3:
Please refer to the Task 4 Extension section for the other 3 model architectures we have tried and their analysis.

Acknowledgement

Consulted with the professor regarding the format of the extensions. No other consultation was made.