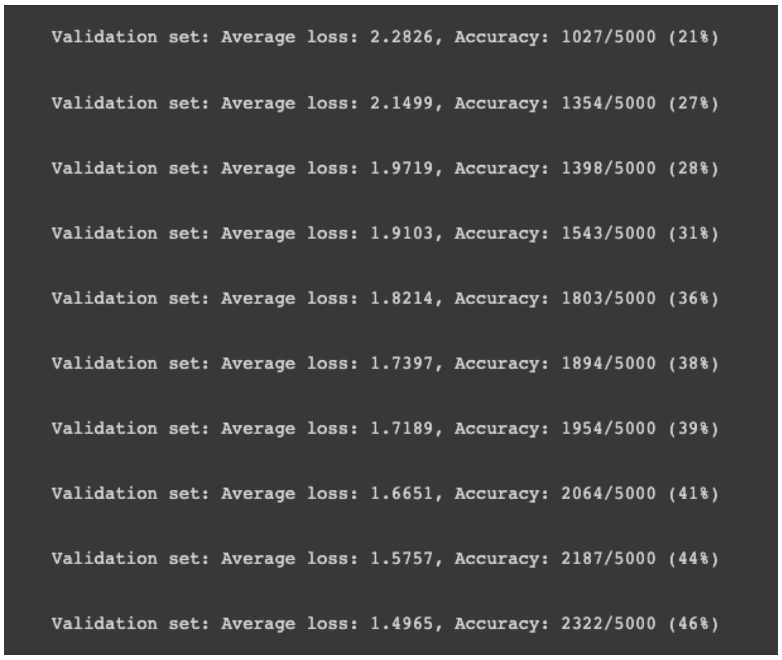
Daniel Choy
Spring 2023
CSE 144

Assignment 2 CNN for Image Processing (Report)

Loss/Accuracy Results:

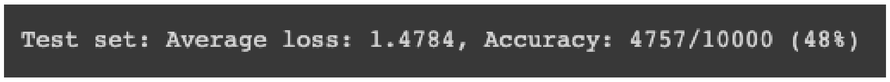
CNN Model 3 Validation Set Results:



```
Validation set: Average loss: 2.2826, Accuracy: 1027/5000 (21%)

Validation set: Average loss: 2.1499, Accuracy: 1354/5000 (27%)

Validation set: Average loss: 1.9719, Accuracy: 1398/5000 (28%)

Validation set: Average loss: 1.9103, Accuracy: 1543/5000 (31%)

Validation set: Average loss: 1.8214, Accuracy: 1803/5000 (36%)

Validation set: Average loss: 1.7397, Accuracy: 1894/5000 (38%)

Validation set: Average loss: 1.7189, Accuracy: 1954/5000 (39%)

Validation set: Average loss: 1.6651, Accuracy: 2064/5000 (41%)

Validation set: Average loss: 1.5757, Accuracy: 2187/5000 (44%)

Validation set: Average loss: 1.4965, Accuracy: 2322/5000 (46%)
```

—>Best Average Loss: 1.4965
—>Best Accuracy: 46%

CNN Model 3 Test Set Result:

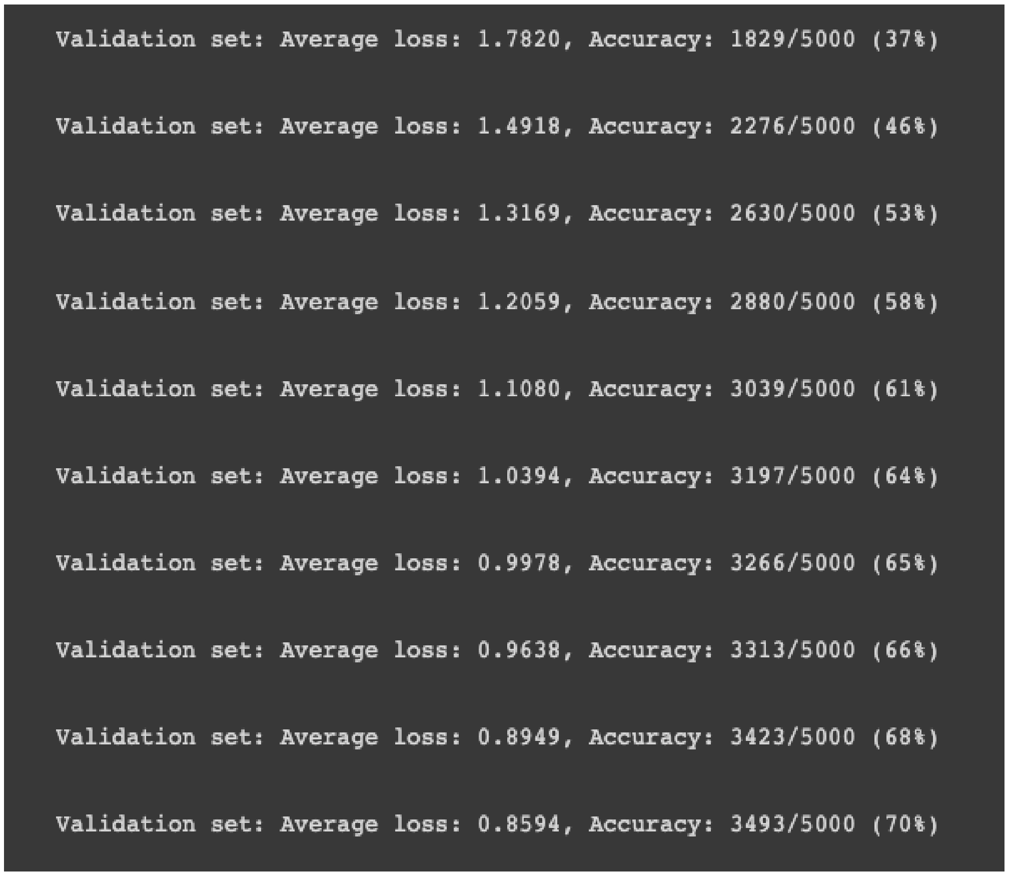


```
Test set: Average loss: 1.4784, Accuracy: 4757/10000 (48%)
```

CNN Model 3 Architecture:

- -Convolution Layer 1
- -ReLU Activation Function
- -Max Pooling
- -Convolution Layer 2
- -ReLU Activation Function
- -Max Pooling
- -Flatten
- -Fully Connected Layer 1
- -ReLU Activation Function
- -Fully Connected Layer 2
- -Log Softmax Activation Function

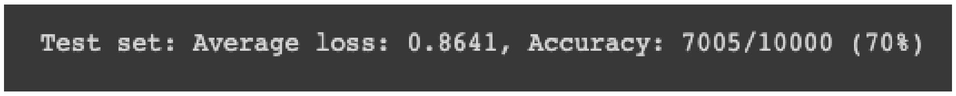
CNN Model 5 Validation Set Results:



```
Validation set: Average loss: 1.7820, Accuracy: 1829/5000 (37%)

Validation set: Average loss: 1.4918, Accuracy: 2276/5000 (46%)

Validation set: Average loss: 1.3169, Accuracy: 2630/5000 (53%)

Validation set: Average loss: 1.2059, Accuracy: 2880/5000 (58%)

Validation set: Average loss: 1.1080, Accuracy: 3039/5000 (61%)

Validation set: Average loss: 1.0394, Accuracy: 3197/5000 (64%)

Validation set: Average loss: 0.9978, Accuracy: 3266/5000 (65%)

Validation set: Average loss: 0.9638, Accuracy: 3313/5000 (66%)

Validation set: Average loss: 0.8949, Accuracy: 3423/5000 (68%)

Validation set: Average loss: 0.8594, Accuracy: 3493/5000 (70%)
```

—>Best Average Loss: 0.8594
—>Best Accuracy: 70%

CNN Model 5 Test Set Result:

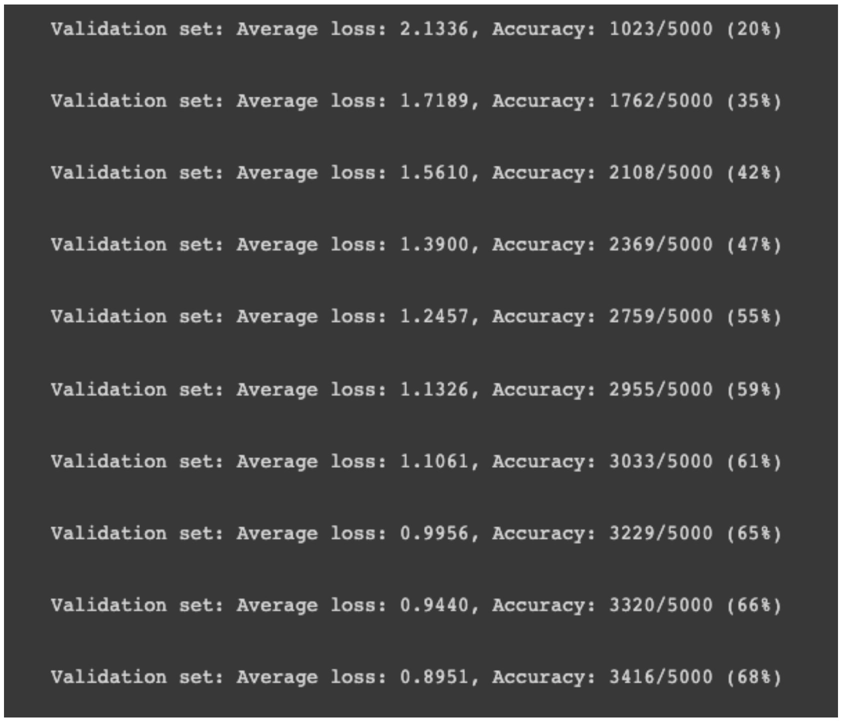


```
Test set: Average loss: 0.8641, Accuracy: 7005/10000 (70%)
```

CNN Model 5 Architectue:

- -Convolution Layer 1
- -ReLU Activation Function
- -Max Pooling
- -Convolution Layer 2
- -ReLU Activation Function
- -Max Pooling
- -Flatten
- -Fully Connected Layer 1
- -ReLU Activation Function
- -Dropout Layer w/ 0.5 Drop Rate
- -Fully Connected Layer 2
- -Log Softmax Activation Function

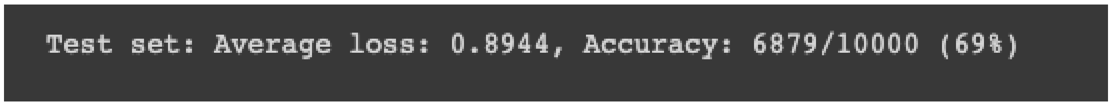
CNN Model 6 Validation Set Results:



```
Validation set: Average loss: 2.1336, Accuracy: 1023/5000 (20%)

Validation set: Average loss: 1.7189, Accuracy: 1762/5000 (35%)

Validation set: Average loss: 1.5610, Accuracy: 2108/5000 (42%)

Validation set: Average loss: 1.3900, Accuracy: 2369/5000 (47%)

Validation set: Average loss: 1.2457, Accuracy: 2759/5000 (55%)

Validation set: Average loss: 1.1326, Accuracy: 2955/5000 (59%)

Validation set: Average loss: 1.1061, Accuracy: 3033/5000 (61%)

Validation set: Average loss: 0.9956, Accuracy: 3229/5000 (65%)

Validation set: Average loss: 0.9440, Accuracy: 3320/5000 (66%)

Validation set: Average loss: 0.8951, Accuracy: 3416/5000 (68%)
```

—>Best Average Loss: 0.8951
—>Best Accuracy: 68%

CNN Model 6 Test Set Result:

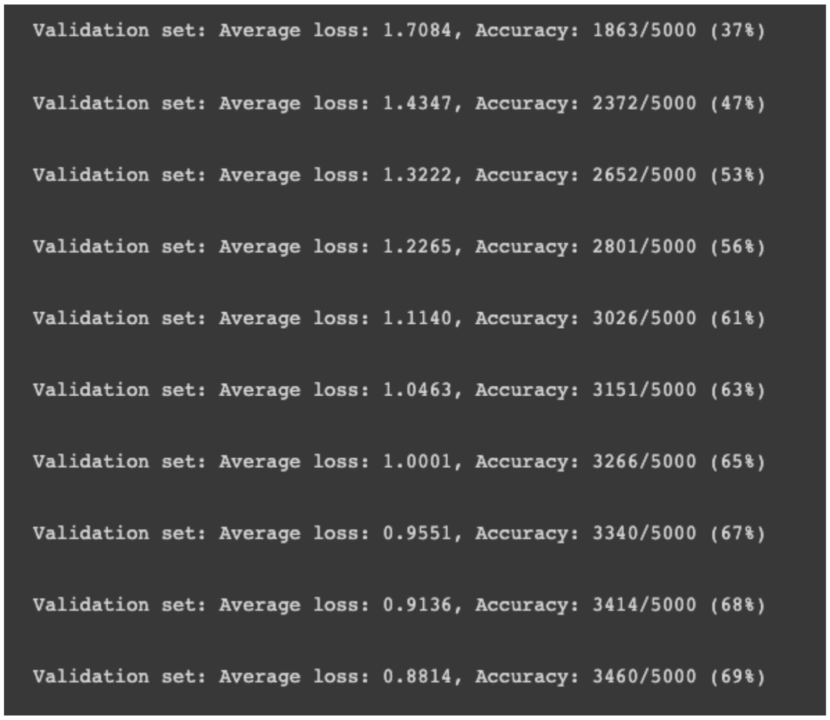


```
Test set: Average loss: 0.8944, Accuracy: 6879/10000 (69%)
```

CNN Model 6 Architectue:

-Convolution Layer 1
-ReLU Activation Function
-Max Pooling
-Convolution Layer 2
-ReLU Activation Function
-Max Pooling
-Convolution Layer 3
-ReLU Activation Function
-Convolution Layer 4
-ReLU Activation Function
-Flatten
-Fully Connected Layer 1
-ReLU Activation Function
-Dropout Layer w/ 0.5 Drop Rate
-Fully Connected Layer 2
-Log Softmax Activation Function

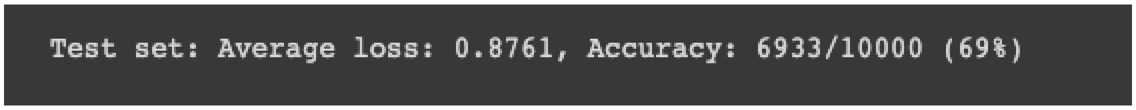
CNN Model 7 Validation Set Results:



```
Validation set: Average loss: 1.7084, Accuracy: 1863/5000 (37%)

Validation set: Average loss: 1.4347, Accuracy: 2372/5000 (47%)

Validation set: Average loss: 1.3222, Accuracy: 2652/5000 (53%)

Validation set: Average loss: 1.2265, Accuracy: 2801/5000 (56%)

Validation set: Average loss: 1.1140, Accuracy: 3026/5000 (61%)

Validation set: Average loss: 1.0463, Accuracy: 3151/5000 (63%)

Validation set: Average loss: 1.0001, Accuracy: 3266/5000 (65%)

Validation set: Average loss: 0.9551, Accuracy: 3340/5000 (67%)

Validation set: Average loss: 0.9136, Accuracy: 3414/5000 (68%)

Validation set: Average loss: 0.8814, Accuracy: 3460/5000 (69%)
```

—>Best Average Loss: 0.8814
—>Best Accuracy: 69%

CNN Model 7 Test Set Result:



```
Test set: Average loss: 0.8761, Accuracy: 6933/10000 (69%)
```

CNN Model 7 Architectue:

- -Convolution Layer 1
- -elu Activation Function
- -Max Pooling
- -Convolution Layer 2
- -elu Activation Function
- -Max Pooling
- -Convolution Layer 3
- -elu Activation Function
- -Convolution Layer 4
- -elu Activation Function
- -Flatten
- -Fully Connected Layer 1
- -elu Activation Function
- -Dropout Layer w/ 0.5 Drop Rate
- -Fully Connected Layer 2
- -Log Softmax Activation Function

What I learned from this Assignment 2?

Preprocessing the Data is Important

Before using the data to train a machine learning model, we have to preprocess it in order for the model to be able to understand and use it.

I **split the dataset into training, validation, and test sets**. I made the validation set be 10% of the training set. By appropriately splitting the data into train, validation, and test sets, I can ensure that my model is trained and evaluated robustly and unbiasedly, enabling me to make more reliable conclusions about its performance and generalization capabilities.

I also implemented **data augmentation** in order to increase the diversity and variability of the training data by applying random transformations to the input samples. This improved my model's ability to generalize and handle different variations in the input data. Specifically, I used transforms.RandomHorizontalFlip() which randomly flips the input image horizontally with a probability of 0.5.

I had a **batch size of 128** in order to load my data in batches rather than as a whole which improves my training efficiency. I also **shuffled my training data** in order to randomize the order in which the data is loaded to help my model learn from a more diverse set of examples.

Tricks to Improve Model Accuracy

In all my models, I added **max pooling** after the first and second convolution layers which takes the max element/weight from all the elements in the current filter window in the feature map. I did this to reduce the spatial dimensions of the feature maps to make the computation of the model faster and to extract the most important features from the feature map. I only used two max pooling operations for each of my models with a kernel size of 2x2 as too much max pooling can lead to loss of information and prevent the model from learning some more important features.

I also used **activation functions** to introduce non-linearity to my model in order to help my model learn/represent complex patterns. Specifically, I used ReLu for models 3, 5, and 6 as the activation function and ELU for model 7. Using the ELU activation function resulted in better accuracy for my model 7 in comparison to using other activation functions like ReLU or RReLU.

I also implemented a **dropout layer with a drop rate of 0.5** to my models 5, 6, and 7 as my regularization technique to prevent overfitting. This dropout layer randomly drops some neurons which forces the remaining neurons to become more robust and independent, since they cannot rely on the presence of the dropped-out neurons.

I used **log softmax as my last activation function** after the second fully connected layer instead of using just softmax. The reason is that I used Cross Entropy Loss which log softmax is more compatible with in comparison to softmax as it provides numerical stability.

I used **stochastic gradient descent** in order to reduce memory use, introduce more randomness into the optimization process, help speed up convergence, and improve the generalization performance of the model.

Hyperparameter and Architecture Tunning are Important

Hyperparameter and architecture tuning of your model is important as they both are big factors in determining the accuracy of your model.

Some of the architecture decisions I decided that resulted in better performance are using ELU instead of other activation functions for my model 7, using log softmax instead of softmax, using SGD rather than regular gradient descent, and using random horizontal flip for data augmentation for my training set.

Some of the hyperparameters that I tuned were the number of epochs and learning rate. I left my model to have 10 epochs as required by the assignment and because it resulted in a stable/decent accuracy for all my models. But, while testing with different numbers of epochs, I noticed that having a bit more epochs caused some of my models to improve in accuracy because they learned more and some to reduce in accuracy due to overfitting. I left my model to have a 0.01 learning rate as required by the assignment and because if I made it bigger it resulted in some of my models to overshoot the optimal solution and even diverge.

Lastly, while building these models I learned a very important architectural fact which is that more complex doesn't necessarily mean better. Specifically, more convolution layers doesn’t necessarily mean better performance. We see this in my assignment when model 5 having only 2 convolution layers obtained a 70% accuracy while model 6 and 7 having 4 convolution layers both obtained 69% which is slightly lower.