Danny Hong
ECE-472: Deep Learning
HW 3 Writeup

Purpose:

The purpose of this assignment was to train a **convolutional neural network** for classifying the MNIST dataset. A convolutional neural network is a neural network that is able to extract features from an image by using a set of matrices known as filters. A dot product is then taken between the filter values and the image pixel values, which ultimately forms the convolution layer.

On the other hand, the MNIST dataset is a dataset commonly used in computer vision that consists of handwritten single digits ranging from 0-9. The dataset contains 60,000 training images and 10,000 test images. The trained convolutional neural network will ideally be able to correctly identify each image as being one of those ten digits.

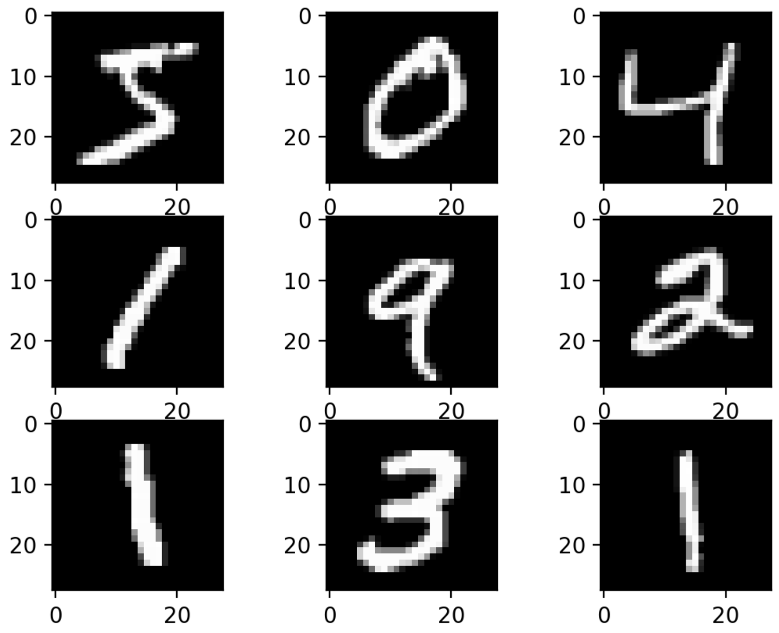

Figure 1: Sample images from the MNIST dataset

Data Preprocessing:

In regards to data preprocessing, the training and testing images/labels were first downloaded from this link: https://deepai.org/dataset/mnist and then read into the python notebook. I then performed **cross validation** on the training dataset, which involves splitting the training dataset into a smaller training set that contains 50,000 images and a validation set that contains the remaining 10,000 images. Cross validation is the process in which input data is divided into two smaller sets: one for training and the other for validation. The purpose of this procedure is to be able to use the validation set to evaluate the performance of our classifier at each training iteration so that possible overfitting of the training dataset may be detected and prevented. **Overfitting** occurs when a model fits too closely to the training data and would then generalize poorly to the test set. This would occur when the model reports wat better accuracy on the training set than it does on the validation set.

Model:

In terms of model specifics, I implemented a feedforward neural network known as a **multilayer perceptron**. The multilayer perceptron consists of two convolutional block layers, along with downsampling after each block through max pooling, which reduces the dimensionality of images by decreasing the number of pixels in the output from the previous convolutional layer. Using max pooling would help improve the computational efficiency of the neural network as it reduces the number of parameters to be learned. The cross entropy loss function is used to measure the performance of the classification algorithm per training iteration and the optimization function used to minimize the loss function is the Adam optimizer. Theoretically, the smaller the output of the loss function, the better the performance of the classification algorithm. The activation function "relu' was used for each convolutional block in order to add nonlinearity to the data so that the model would be able to pick up on more complex patterns. In addition, L2 regularization was done on the output layer, which adds the sum of the squared values of the predicted weights to the loss function in order to reduce the risk of underfitting/overfitting of the model. A Dropout function was called right before the output layer. The dropout command is used to randomly remove certain data points during each weight update in order to prevent overfitting from taking place. Finally, a dense output classification layer was used in the end to perform the actual classification.

Results and Conclusion:

Summarized below are the results of my classification algorithm after training the model for a total of five epochs:

	Train	**Test**	**Val**
Loss	0.1232	0.0877	0.0964
Accuracy	0.9790	0.9876	0.9855

Figure 2: Loss and Accuracy data after five epochs for the different datasets

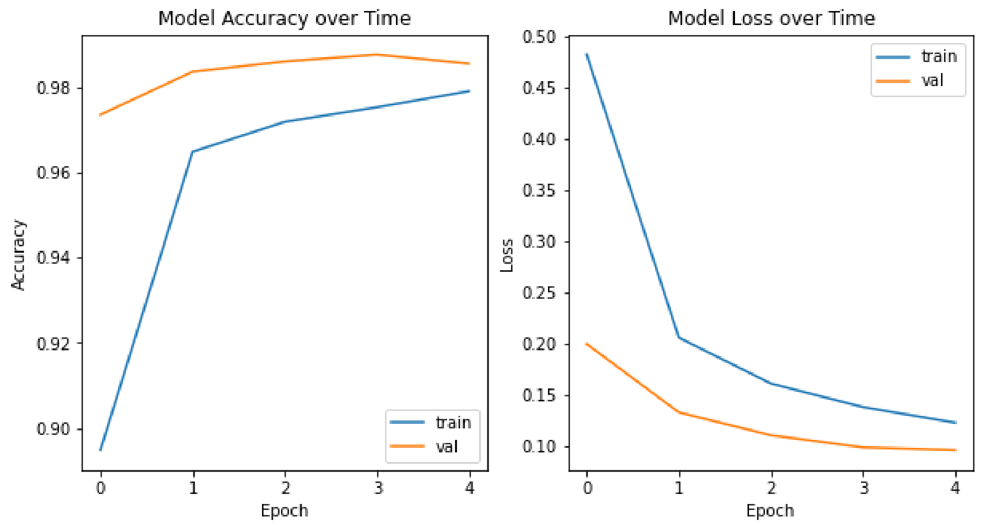

Figure 3: Graph depicting Model Accuracy and Loss over Time

The results show that the model seemed to perform well on all three datasets (training, validation, and testing). There's a number of models today that do a great job at classifying the MNIST dataset, mostly due to the fact that the MNIST images are simple enough for classification as they do not contain many features that need to be extracted. In fact, the state of the art accuracy on classifying MNIST is 99.91%. Further work that can be done include experimenting with other models to obtain even better results than the ones that were obtained here along with looking into ways to lower the number of parameters that are used in training (currently, the model has 225,034 parameters) in order to decrease computational efficiency.