Homework Five Report

Zifan Wang

April 1, 2019

1 Problem 1: CNN Architecture and Training

1.1 CNN Architecture and Training

1.1.1 Abstract and Motivation

Object recognition and classification are two main topics in computer vision field. The previous studies did not provide high accuracy results in these two areas. Nowadays, with the help of big data and high speed Graphic Processing Unit (GPU), people utilize the idea of neural network to recognize and classify objects. Neural networks, inspired by the biological neural networks that constitute animal brains, are large computing systems to model a data set. Convolutional neural network (CNN) is one of neural network, commonly applied to object recognition and classification.

1.1.2 Approach and Procedures

With over twenty years developments, CNN has lots of different architectures. However, they all share the same main idea. The LeNet-5 proposed by Yan leCun in 1998 is the simplest and classical CNN architectures to be used to introduce CNN architecture. The following figure shows the LetNet-5 architecture [1].

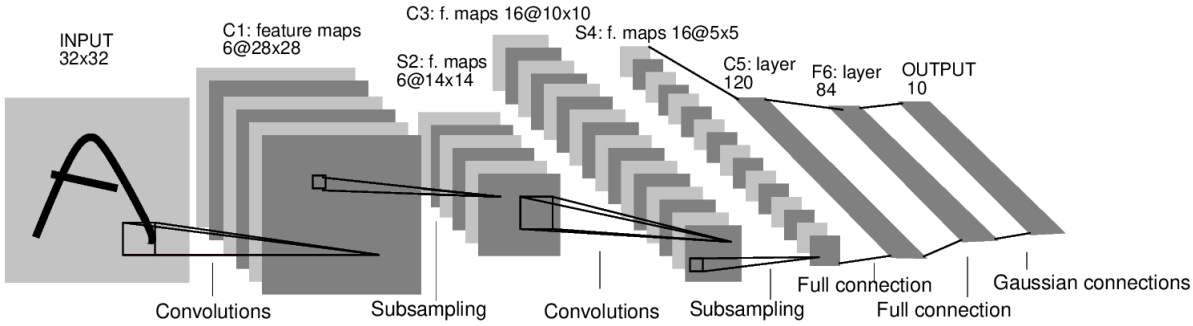


Figure 1: LeNet-5 Architecture

From figure 1,there are three different layers in the LeNet-5: convolutional layer, subsampling layer, and fully connected layer.

- **Convolutional Layer**
  The convolutional layer is the most important stage in the convolutional neural network. This layer consists of different feature maps.The following figure 2 illustrates how convolutional layer extracts the important features.

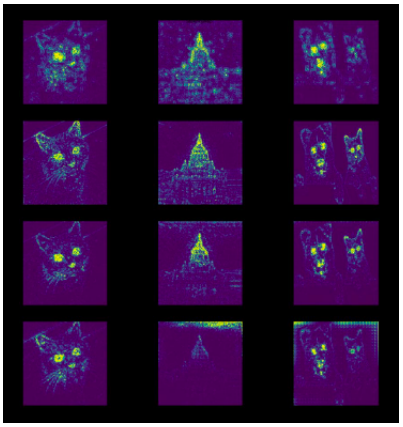


Figure 2: Convolutional Layer

In the LeNet-5 architectures, there are two convolutional layers. The first layer uses 6 filters to extract 6 feature maps and the second layer uses 16 filters to extract 16 feature maps. The process used to extract features is called convolution operation. Before stepping into the convolution operation, the hyperparameters of convolutional layers have to be introduced. There are three hyperparameters: the depth, stride, and padding to control the size of output volume after convolutional process:

- **Depth**: The depth of output volume corresponds to the number of filters are applied in the convolution. For example, in the LeNet-5, the first convolutional layer uses 6 filters to generate 6 different feature maps from input images and the second convlutional layer uses 16 filters to extract 16 different feature maps.
- **Stride**: The stride is another hyperparameter to control the slide step of filters. The stride 1 means that filters is moved one pixel at a time. The stride 2 moves filters 2 pixels at a time. The stride 3 or more is uncommonly used in CNN, because they will result a lossing important features.
- **Padding**: The padding helps keep features near the boundary. Zero-padding is the most commonly used padding in CNN. The padding size is a hyperparameter which controls the spatial size of the output volumes. During the zero-padding, we pad the input volume with zeros around the border.

The convolution operation is accomplished by doing a convolution between a kernel and an image or a feature map. The following expression shows the convolution operation:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

where $S(i,j)$ is the filtered image, I is the original image, and K is the filter kernel. The following figure 3 shows the convolution operation with two $3 * 3$ filters, one zero-padding, and stride 2.

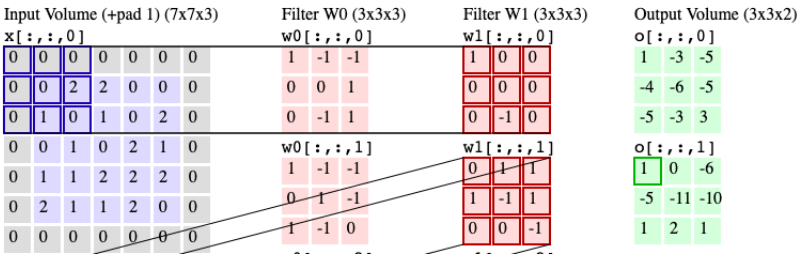


Figure 3: Convolution Operation

- **Pooling Layer**
  The pooling layer usually appears after convolutional layer. It down sample the convolutional layer and aggregates important features. There are two types of pooling: max pooling and average pooling. The following figure 4 shows how max and average pooling work with a 2×2 window and stride 2.

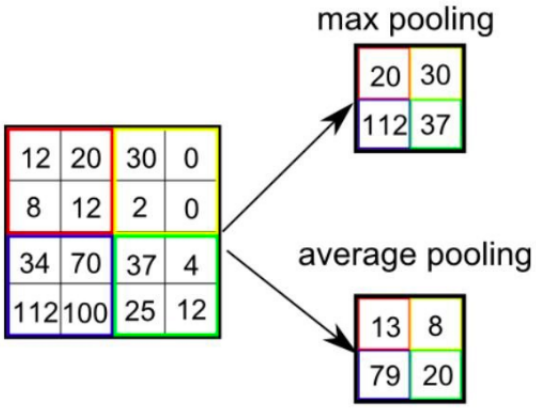


Figure 4: Pooling Layer

  For max-pooling, it acts like a maximum filter which replaces the central pixel with the largest pixel value in the running window. For average-pooling, it acts like a mean filter with replaces the central pixel with the average pixel value in the running window. The max-pooling with 2×2 window and stride 2 is usually used in CNN. The LeNet-5 network uses two max-pooling layers to sub-sample and aggregate important features.

- **Fully-connected layer** In the fully-connected layer, neurons have full connections to all activations in the previous layer. This layer can be expressed as $z = \sum_i w_i x_i + b_i$, where z is the output of fully-connected layer, w_i is the corresponding weight of each input, x_i is the input, and b_i is the bias of corresponding input.

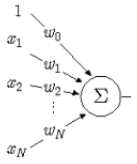


Figure 5: Fully-Connected Layer

Usually, one convolutional layer and one max-pooling layer are called one convolutional layer. Inspired by human brains, the **activation function** introduces a non-linearity to CNN system, which makes CNN truly functional. After each layer, an activation is applied. There are three commonly used activation functions in deep learning: **sigmoid**, **tanh**, and **relu**.

- **sigmoid**:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

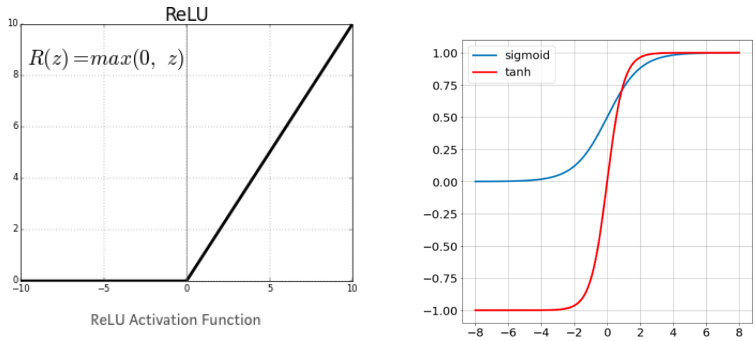
- **tanh**:

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- **relu**:

$$relu(x) = max(0, x)$$

The following figures 6 shows the sigmoid, tanh, and relu functions:



(a) Relu (b) Sigmoid Tanh

Figure 6: Activation Functions

The Relu activation function is the most commonly used because it reduced likelihiood of the fradient to vanish. This arises when $x > 0$. In this regime the gradient has a constant value. However, the gradient of sigmoids becomes increasingly small as the absolute value of x increases.
The **softmax** is another activation function which is used in the multi-category classification at the output layer. The softmax can be expressed as $a = [\frac{e^{s1}}{\sum_{i=1}^{N} e^{s_i}} \quad \cdots \quad \frac{e^{sN}}{\sum_{i=1}^{N} e^{s_i}}]$. This will output a probability distribution. The one with the highest probability will be chosen.
Convolutional Neural network is a supervised machine learning technique which approximate a target function f that maps input variables (x) to an output variable or a ground truth (y).

$$y = f(x)$$

An important consideration in learning the target function from the training data is how well the model generalizes to new data. The goal of a good CNN is to generalize well from the training data to any data from the problem domain. This allow people to make predictions in the future on data the model has never seen. Under-fitting, good-fitting, and over-fitting are three terminologies used to describe the goodness of fitting. The following figure 7 shows three types of fitting:

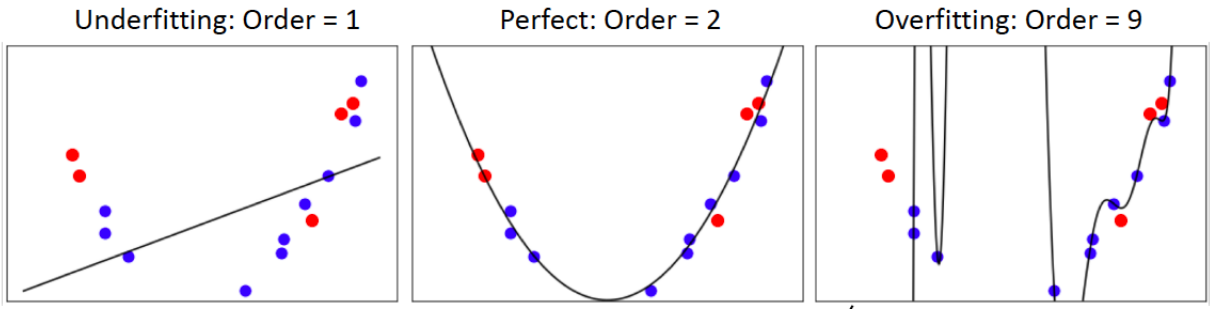


Figure 7: Fitting

- **Overfitting**
  Overfitting refers to a model that models the training data too well. When a model learns the detail and noise of training data in the training process, the overfitting happens and this will cause low accuracy in the new data. There are four possible ways to reduce overfitting. First, we can add more training data. If we have limited training data, the data augmentation is a way for people to have more data. The more training data, the more CNN will be learned, which will cause to reduce overfitting. Second, regularization after cost function will help to reduce overfitting. There are two commonly used regularization: L1 and L2 weight penalty:

  - **L1**

    $$C(w) = C_0(w) + \lambda||w||_1$$

    where $C(w)$ is regularized cost, $c_0(w)$ refers to original cost, and $\lambda||w||_1$ is regularization term.

  - **L2**

    $$C(w) = C_0(w) + \lambda||w||_2^2$$

    where $C(w)$ is regularized cost, $c_0(w)$ refers to original cost, and $\lambda||w||_2^2$ is regularization term.

  Third, applying dropout layer which refers to delete a random selection of input and hidden nodes for every minibatch during training will prevent CNN from overfitting. Usually, the dropout value is less than 0.5. The following figure 8 illustrate the idea of dropout.

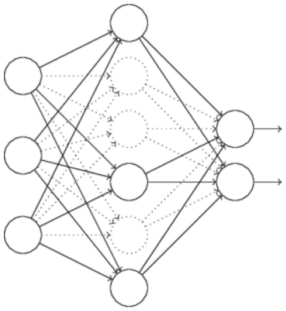


Figure 8: Dropout

Fourth, early stopping is the last method to prevent overfitting. This method refers to stop before overfitting happening.

- **Underfitting**
  Underfitting refers to a model that can neither model the training data nor generalize to new data. Adding more data and re-design the model are possible ways to solve underfitting issue.

Explain the loss function and the classical backpropagation (BP) optimization procedure to train such a convolutional neural network.

- **Loss Function**
  Loss function is commonly used to evaluate the performance of a neural network. The less loss function returns, the better CNN is. There are loss function cross-entropy loss and quadratic loss (MSE).

  - Cross-entropy loss: The cross-entropy loss is commonly used in classify multi-objects and corresponding to the softmax function. The following function shows the cross-entropy loss:

    $$C = -\sum_{i=1}^{N^{(L)}} y_i^{(L)} ln\, a_i^{(L)}$$

    where C is the loss, $y_i^{(L)}$ is the ground truth labels, and a_i^L is network outputs. For binary labels, the cross-entropy loss reduce to:

    $$C = -[y^{(L)} ln\,(a^{(L)}) + (1 - y^{(L)}) ln\,(1 - a^{(L)})]$$

  - Quadratic loss (MSE)
    The quadratic loss is often used in classifying single object. For example, people want to know the image shows cat or not. They will use quadratic loss function. The following equation gives quadratic loss:

    $$C = \sum_{i=1}^{N^{(L)}} (y_i - a_i)^2$$

- **Back Propagation**
  The back propagation is the method that make CNN learn filters. The idea behind the back propagation is updating the kernel weights of filters and weights in the fully-connected layers to fit CNN. The back propagation contains two parts: derivatives calculation and optimization. There are two parts of derivatives calculation in CNN: convolutional layers and fully connected layers. The following figure 9 shows the example of derivative calculation in convolutional layers:

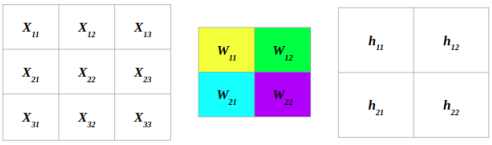


Figure 9: Back Propagation in Convolutional Layer

The following equations shows the derivatives of filter weights:

$$\partial W_{11} = x_{11}\partial h_{11} + x_{12}\partial h_{12} + x_{21}\partial h_{21} + x_{22}\partial h_{22}$$

$$\partial W_{12} = x_{12}\partial h_{11} + x_{13}\partial h_{12} + x_{22}\partial h_{21} + x_{23}\partial h_{22}$$

$$\partial W_{21} = x_{21}\partial h_{11} + x_{22}\partial h_{12} + x_{31}\partial h_{21} + x_{32}\partial h_{22}$$

$$\partial W_{22} = x_{22}\partial h_{11} + x_{23}\partial h_{12} + x_{32}\partial h_{21} + x_{33}\partial h_{22}$$

The derivative calculation in the fully-connected layer is:

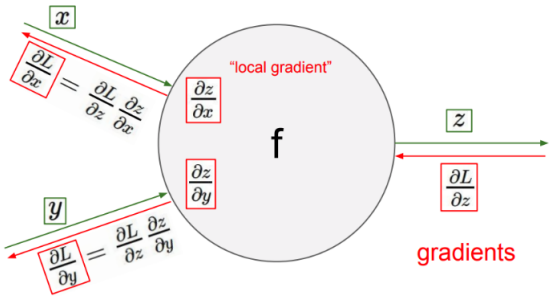


Figure 10: Back Propagation in Fully-Connected Layer

Optimization is a strategy or algorithm to reduce the loss function and make the network learn. The stochastic gradient descent (SGD) is the simplest optimizer.

$$p = p - \eta \nabla_p C$$

where p represents weights of filters in convolutional layers and weights in the fully-connected layer. The SGD is not a good optimizer in large CNN, because the large CNN will result that weights are hard to reach global minima. The following figure 12 shows this idea:

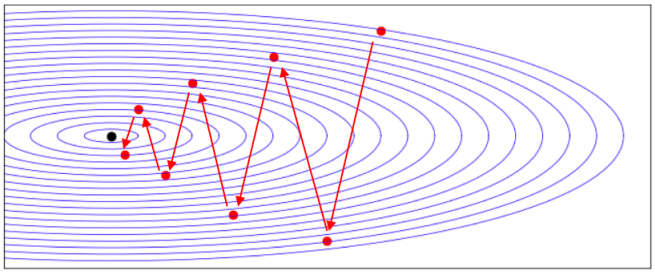


Figure 11: Ill-conditioned Gradient descent

Momentum is another idea of optimizer.

$$v = \alpha v - \eta \nabla_p C$$

$$p = p + v$$

where α which shows how much of past history should be applied is between zero and one, typically 0.9.

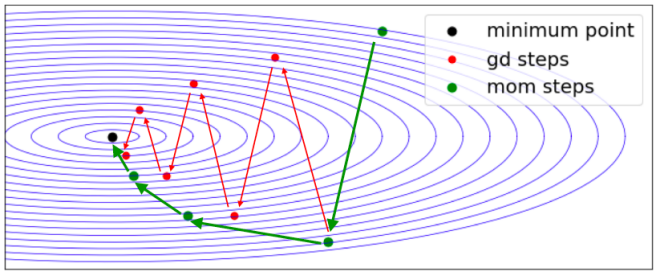


Figure 12: Momentum and SDG Comparison

1.1.3 Discussion

Why CNNs work much better than other traditional methods in many computer vision problems?
In the past, people use SIFT and SURF method to extract keypoints in the graphics and using SVM or k-means clustering to classify objects. On the contrary, the CNN not only extracts key features but also it optimize the features. Features used to detect objects or classify objects are learned during the training process. This is different from the traditional methods which key features are keep the same and use key features' energy to classify objects. For example, in the homework 4, after training zero and one digits, the eight digit still has 0.7 probability to be zero and 0.3 probability to be one which means that the digit eight has similarly feature to both digit zero and digit one. However, the CNN will return the different result. The convolutional layer will learned key features of digit zero and key features of digit one. The testing digit eight will neither be classified into digit zero nor digit one. In conclusion, the CNN is more data driven and learns features better during the training. As a result, it will give better results than traditional computer vision method.

1.2 B. Train LeNet-5 on MNIST Dataset

1.2.1 Abstract and Motivation

LeNet-5 is the classical convolutional neural network and MNIST dataset are commonly used to classify digits. We will use both to train a CNN to classify digits.

1.2.2 Approach and Procedures

The following figure 13 is the required CNN in the homework 5:

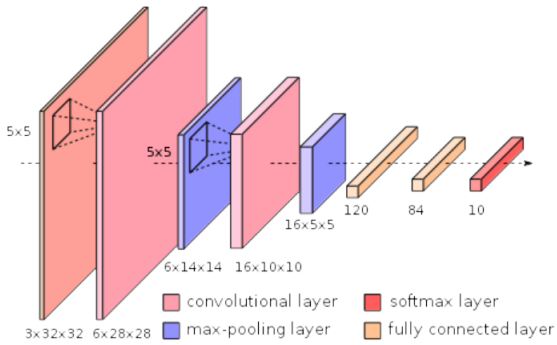


Figure 13: Required leNet

1.2.3 Result

Five experiments are done to test impacts of hyperparameters in the CNN. The following table 1 shows five different hyperparameters chosen during the training.

Configuration	1	2	3	4	5
Learning Rate	0.001	0.001	0.01	0.0001	0.001
Momentum	false	true	true	true	true
KernelRegularizer	true	true	true	true	false

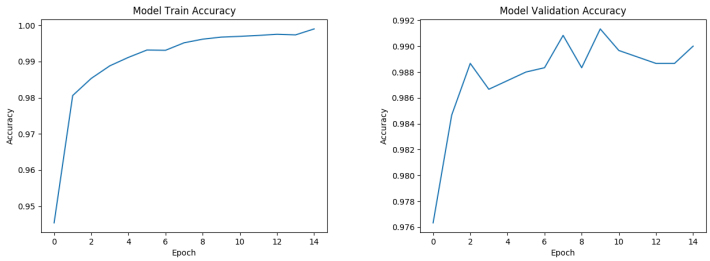
Table 1: CNN Configurations

The following table 2 shows the results of five different setting of CNN.

Configuration	1	2	3	4	5
Mean Training Accuracy	0.99027	0.98958	0.19100	0.97384	0.98953
Var Training accuracy	0.00016	0.00019	0.04633	0.00077	0.98594
Mean Validation Accuracy	0.98778	0.98754	0.16215	0.98081	0.00015
Var Validation accuracy	1.1e-05	6.5e-06	0.04501	5.7e-05	1.2e-05
Final training accuracy	0.99900	0.99740	0.11200	0.99070	0.99740
Final validation accuracy	0.99000	0.99100	0.10500	0.98370	0.98770
Test accuracy	0.99100	0.98750	0.1135	0.9843	0.99030

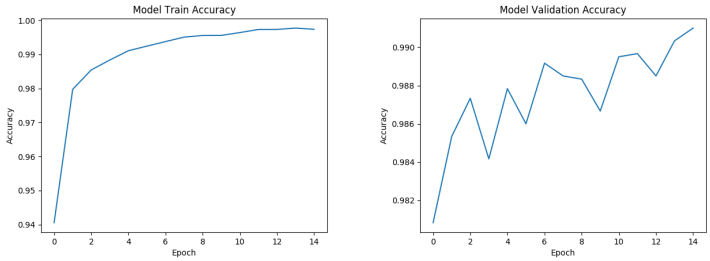
Table 2: Digit Classification Result

The following figures are the plots of training accuracy and validation accuracy during the training.



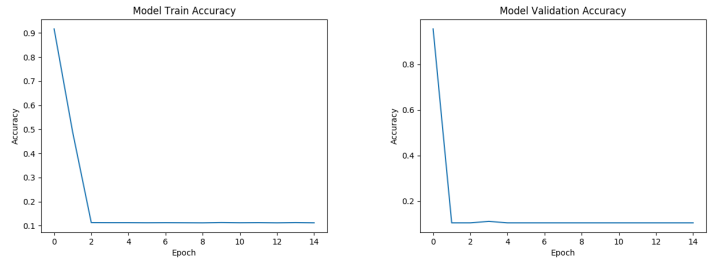
(a) Training Accuracy (b) Validation Accuracy

Figure 14: Configuration One



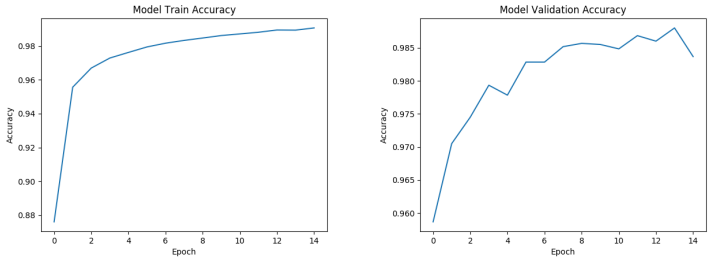
(a) Training Accuracy (b) Validation Accuracy

Figure 15: Configuration Two



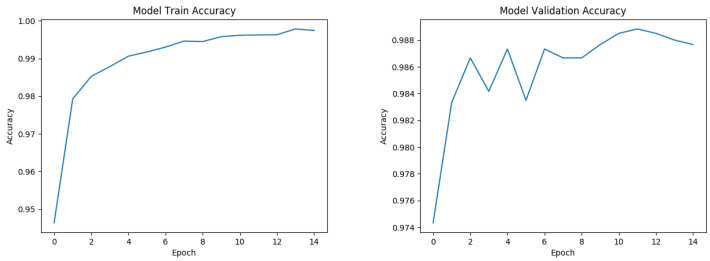
(a) Training Accuracy (b) Validation Accuracy

Figure 16: Configuration Three



(a) Training Accuracy (b) Validation Accuracy

Figure 17: Configuration Four



(a) Training Accuracy (b) Validation Accuracy

Figure 18: Configuration Five

1.2.4 Discussion

From the table 2, the overfitting and underfitting does not happen in all five configurations. Comparing different configurations is important to know the effects of different hyperparameters:

- Configuration 2 vs. Configuration 1
  The only difference between configuration 1 and 2 is that the configuration 2 uses momentum method in the optimization process. The small dataset and small dimensional image do not make momentum method have higher accuracy than the the method without momentum method. Instead, the one without this method has slightly higher accuracy than the one with. However, when training the large CNN, the momentum method will help us improve accuracy.

- Configuration 2 vs. Configuration 3
  The difference between configuration 2 and 3 is that the configuration 3 uses high learning rate. It is obvious that configuration 2 with low learning rate has better performance than the configuration 3 with high learning rate. This is because the high learning rate causes CNN misses the optimization value (global minima) and finally result a low accuracy.

- Configuration 2 vs. Configuration 4
  The difference between configuration 2 and 4 is that the configuration 4 uses low learning rate. The performances of two configuration is similarly. However, the configuration 2 with higher learning rate trains CNN faster and the configuration 3 with lower learning rate trains CNN slower. In the small neural network, people do not have to choose a very lower learning rate. The low learning rate indeed will cause people to use more time to train the network.

- Configuration 2 vs. Configuration 5
  The only difference between configuration 2 and 5 is that the configuration 2 uses kernel regularizer during the training. Two results are similarly to each other. However, the configuration 2 with kernel regularizer has the slightly better performance than the one without. In the large CNN, kernel regularizer will have to reduce overfitting and improve the result.

The best CNN I achieved is the first configuration with 0.001 learning rate, no momentum, and kernel regularzier. The testing accuracy is 0.99100. The figure 14 shows the performance curves of validation set and training set.

1.3 C. Apply trained network to negative images

1.3.1 Abstract and Motivation

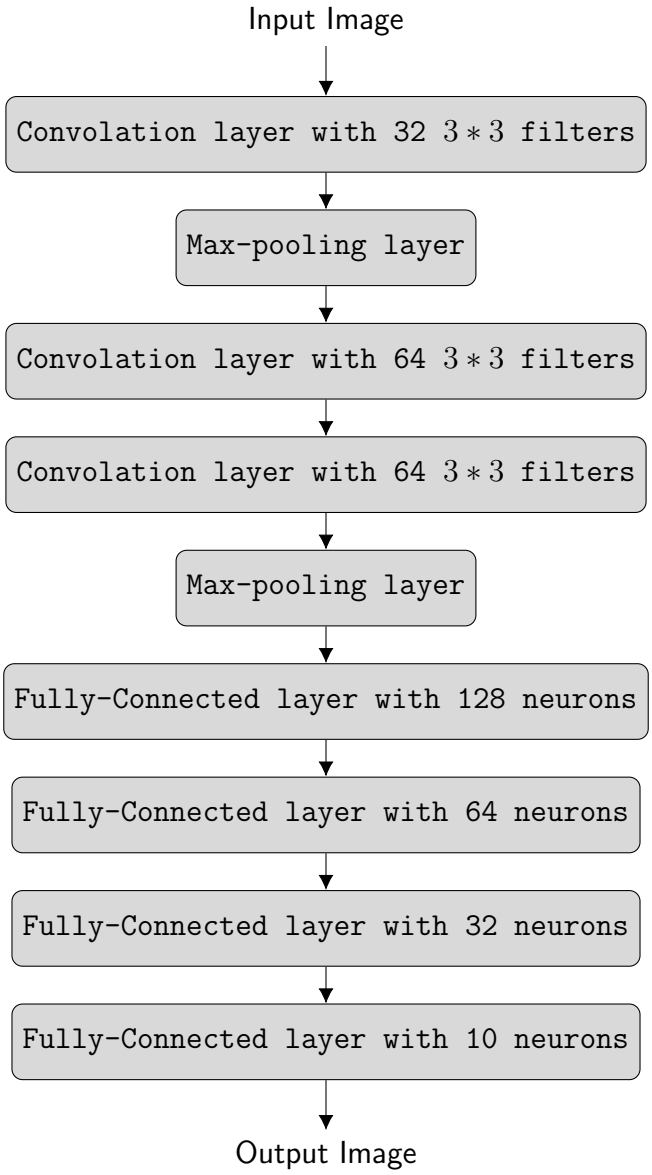
CNN is lazy which means that it cannot classify or recognize objects when people changing the photo background. This small changing will make neural network miss label objects.

1.3.2 Approach and Procedures

To classify the negative images' digits, it is important to use negative images as training data. The first step is to create negative data from the given dataset. The following equation does this conversion:

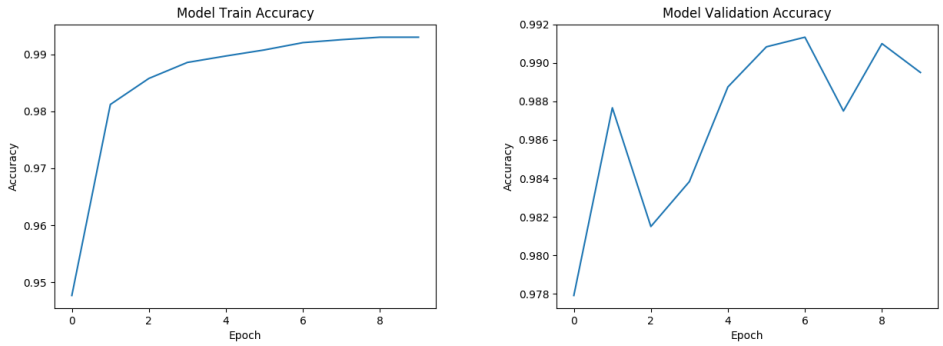
$$r(x, y) = 255 - p(x, y)$$

where $p(x, y)$ is the value of the original image at location (x, y). The second step is to set up my CNN. The following flow chart is my nerual network setting.



1.3.3 Results

Using the best configuration from the previous problem, the test accuracy is 0.2274 and test loss is 9.106 when using negative images as testing images. Due to this low accuracy, I design a new CNN and using negative images as training data. The following figure shows the training accuracy and validation accuracy during the training.



(a) Train Accuracy (b) Validation Accuracy

Figure 19: CNN used to Classify Negative Images

The following table gives the training result:

Mean Training Accuracy	0.9854
Var Training accuracy	0.00017
Mean Validation Accuracy	0.9869
Var Validation accuracy	1.8e-05
Final training accuracy	0.9930
Final validation accuracy	0.9895
Test accuracy	0.98945

Table 3: Digit Classification Result

1.3.4 Discussion

The reason that the configuration in the previous problem returns low accuracy is that we use negative images as testing data. According to the training data inputs which were positive images, the neural network only learns how to classify digit in the positive images. However, when giving negative testing data, the CNN cannot correctly classify these digits. The only way to make CNN learn to classify both types of images is that give CNN both types of images. The neural network developed in this homework has more layers than the traditional leNet-5 and uses both type of images in the training process. The result is given in the table 3 and accuracy plots are in the figure 19. The test accuracy is 0.98945, so it is a good CNN to classify both types of digits.

References

[1] Y. LeCun *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 86(11):2278-2324, November 1998.