EE 569 HOMEWORK V

Royston Marian Mascarenhas

USC ID: 8286328166

Submission Date: April 7th, 2019

USC email: rmascare@usc.edu

**Convolutional Neural Networks**

**(a)**

1. **Abstract and Motivation:**
2. **Explanations:**
3. **Fully connected layer:**

The fully connected layer has connections which are fully connected to the all the inputs of the previous layer. For this reason, the previous layer needs to be **flattened** since the previous CONV layers does not have compatible one dimensional nodes. Therefore, the previous CONV layer dimensions have to be unravelled in order to facilitate the functionality of a dense layer.

The output from the CONV layer represents high – level features in the data. Flattening the layer ensures that **the non linear combinations are learned the easy way**.

The convolutional layer is used for feature extraction. The dense layers are used to **classify** the input images based on the extracted features. Instead of using a SVM to classify the input based on the features, we use dense layers to make it end to end trainable.

The **number of trainable parameters** in a dense layer is input x output + bias. The bias is the number of nodes in that densely connected layer. The input is the connections from the previous layer and the output is the number of connections in that layer. A layer with 120 connections from the previous layer and 84 connections in the present layer will have 10164 trainable parameters (120 x 84 + 84). Thus, their activations can be learnt by matrix multiplication followed by an offset.

*FC to Conv Layer:* The FC layers can be converted to CONV layers by reshaping them into even equally sized arrays. Each of these conversions involve reshaping the weight matrix in each FC layer to CONV layer. This allows easier transitions of the convnet over the spatial positions in one forward pass.

|  |  |
| --- | --- |
| **FC layer** | **CONV layer** |
| No parameter sharing | Parameter sharing among each slice of output volume depth |
| All nodes get all connections from the previous layer nodes | Each neuron is connected to a local region of the input |

**2. Convolutional Layer:**

The convolutional layer is a layer consisting of various interpretations of the previous layer. It is designed in such a way so as to allow for the input to be an image instead of just one dimensional nodes stacked together.

It takes after the name ‘convolution’ because it involves the operation of sliding a nx n window over the image and multiplying each input with the corresponding filter weight. When all the n x n entries are obtained, they are added together to create the entry for the center pixel of that window.

**Filter***: Every filter is a tool for interpreting the image in its own unique way.* A n x n filter consists of n x n **(kernel size)** learnable weights plus a bias of one unit. Every filter extends through the dimensions of the input volume. If the input was RGB, the filter would extend through 3 dimensions of the input. Each filter produces a 2 dimensional activation map which corresponds to a specific visual feature. These activation maps are stacked together to produce the output volume. Every entry in the output volume can be thought of a neuron gaining insight into a particular aspect of the input. To makes sure it scales to big input sizes, we will connect each neuron only to a specific region of the input. This is known as the receptive field of the neuron. For LeNet, each neuron will have weights 5x5x1 + 1 = 26 in number.

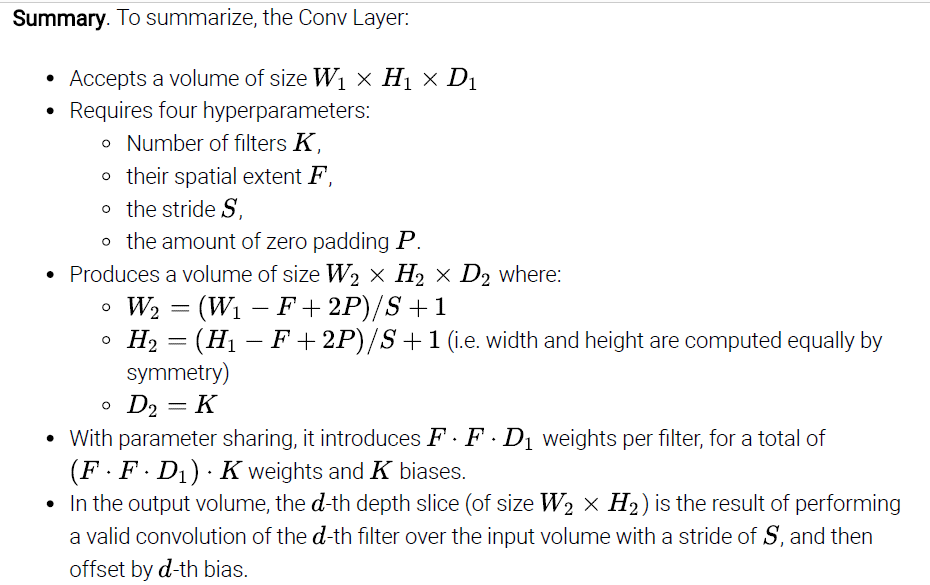
*Spatial properties:* Three hyperparameters determine the output size of the CONV layer. The **depth** of the output volume is the number of the filters employed in that CONV layer. The different neurons along the depth of the volume might activate based on different triggers such as edges, blobs, textures etc. The **stride** is used to slide the window of the filter by n pixels. By default, the stride is taken to be 1 so that the output volume is not reduced. The **extent of zero padding** is used to manage the spatial imbalance that occurs while choosing different parameters.

The correctness of the properties can be chosen by this formula:

Fit = (W – F + 2P) / (S + 1)

where W is the input volume size, F is the receptive field size of the conv layer neurons, S is the stride and P is the amount of zero padding. If fit is an integer, the parameters chosen are correct.

The CONV layer tends itself to **parameter sharing**. Each of the 28 x 28 inputs in the image use the slice of the depth (6 filters). This results in 6 x 5 x 5 + 6 = 156 filters instead of 28 x 28 = 784 connections.



Summary of the CONV layer dimensions and parameters *ref* [1]

Suppose we want to stack one CONV with kernel size of 7 x7 instead of 3 conv layers with kernel size of 3 x 3. The first method is not advisable as three layers would involve more non linearities making the features more expressive and it would result in loss of information as well. Although in the latter case, we might need more memory to hold all the weight during backpropagation.

**3. Max Pooling Layer:**

A pooling layer’s main purpose is to downsample the input to make computations easier and runtime faster. By extension, it also helps reduce overfitting since it allows only responses with the highest magnitude of energies.

The pooling layer works independently on each slice of the depth of the previous CONV layer to spatially downsize it. There are many **types** of pooling such as average pooling, median pooling but the most used is MAX pooling. With a window size of 2 x 2 and a stride of 2, it reduces the input activations by 75%.

A pooling layer usually has two settings: a 3x3 window with a stride of 2 or a 2x2 window with a stride of 2. Higher window sizes have a destructive effect and result in loss of information.

**(b)**

1. **Abstract and Motivation:**
2. **Approach and Procedures:**

**Setting 1 : Original LeNet Setting**

**Setting 2: Lesser filters in first CONV layer, more filters in second CONV layer**

**Setting 3: More filters in first CONV layer, lesser filters in second CONV layer**

**Setting 4: More filters in both CONV layers**

**Setting 5: Decrease kernel size**

**Setting 6: Increase dense units**

**Setting 7: Decrease pool size**

**Setting 8: Learning rate**

**Setting 9: Number of epochs**

**Setting 10: Batch Normalization and Dropout**

1. **Results:**

**Setting 1: Original LeNet Setting**

**Properties:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Filters** | **Kernel Size** | **Stride** | **Activation** |
| **Conv2D** | 6 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Conv2D** | 16 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Flatten** | output = 400 | - | - | - |
| **Dense Layer 1** | units = 120 | - | - | - |
| **Dense Layer 2** | units = 84 | - | - | - |
| **Dense Layer 3** | units = 10 | - | - | - |

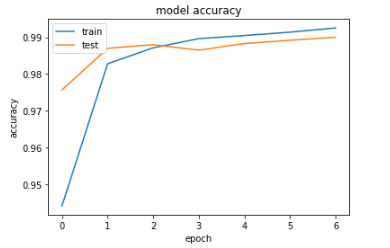
**Optimizer:** RMS prop with default learning rate (0.001)

**Loss:** Categorical Cross Entropy

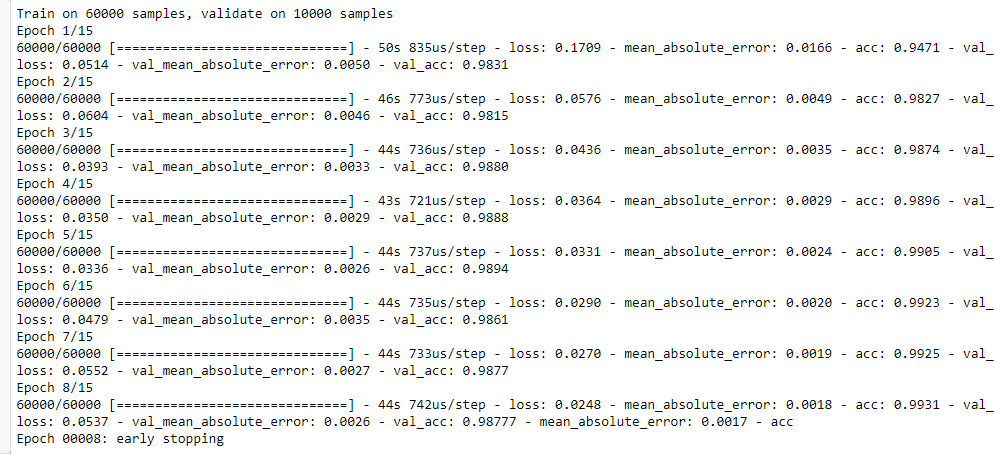
**Epochs:** 8

**Batch- size:** None

**Epoch – Accuracy Plot:**



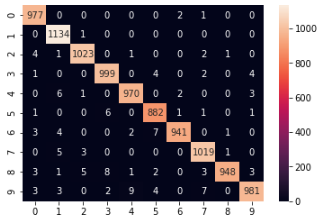
**Accuracy:**



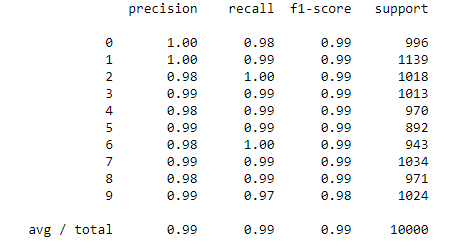
Training accuracy: Early stopping accuracy - 99.05% Stopped model accuracy – 99.31%

Test accuracy: Early stopping accuracy – 98.94% Stopped model accuracy – 98.77%

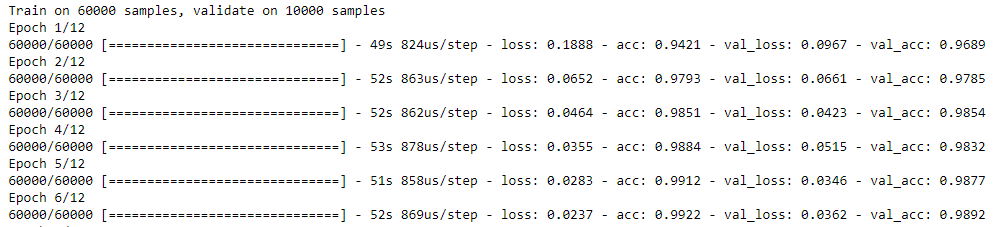
**Confusion Matrix :**



**Classification Report:**



**Setting 1A : Note:** Adam optimizer gives accuracy (98.92%)



**Setting 2: Filters:** Lesser filters in the first CONV layer, more filters in the second CONV layer. More dense units.

**Properties:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Filters** | **Kernel Size** | **Stride** | **Activation** |
| **Conv2D** | 6 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Conv2D** | 16 | - | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Flatten** | output = 400 | - | - | - |
| **Dense Layer 1** | units = 120 | - | - | - |
| **Dense Layer 2** | units = 84 | - | - | - |
| **Dense Layer 3** | units = 10 | - | - | - |

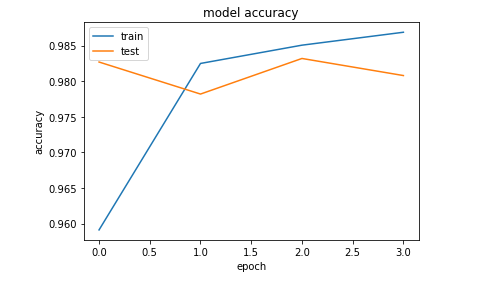
**Optimizer:** RMS prop with default learning rate (0.001)

**Loss:** Categorical Cross Entropy

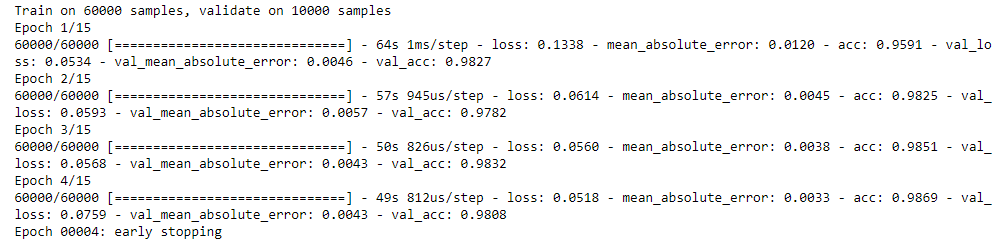
**Epochs:** 4

**Batch- size:** None

**Epoch – Accuracy Curve:**



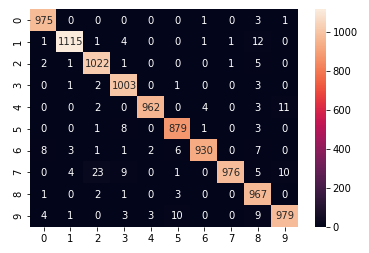
**Accuracy:**



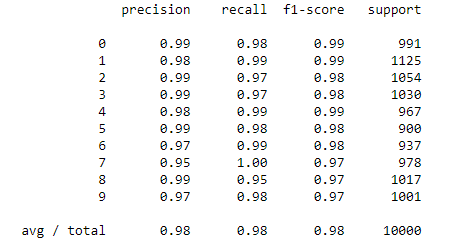
Training accuracy: Early stopping accuracy – 95.91 Stopped model accuracy – 98.69

Test accuracy: Early stopping accuracy – 98.27 Stopped model accuracy – 98.08

**Confusion matrix:**



**Classification Report:**



**Setting 3: Filters:** More filters in the first CONV layer, less filters in the second CONV layer

**Properties:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Filters** | **Kernel Size** | **Stride** | **Activation** |
| **Conv2D** | 18 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Conv2D** | 10 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Flatten** | output = 400 | - | - | - |
| **Dense Layer 1** | units = 120 | - | - | - |
| **Dense Layer 2** | units = 84 | - | - | - |
| **Dense Layer 3** | units = 10 | - | - | - |

**Optimizer:** RMS prop with default learning rate (0.001)

**Loss:** Categorical Cross Entropy

**Epochs:** 8

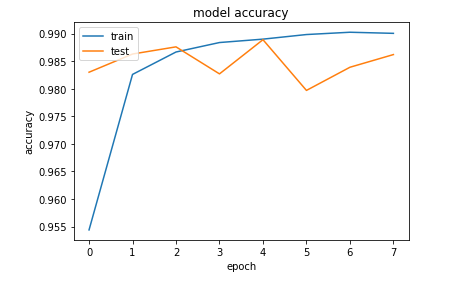
**Batch- size:** 16

Total params: 46,112

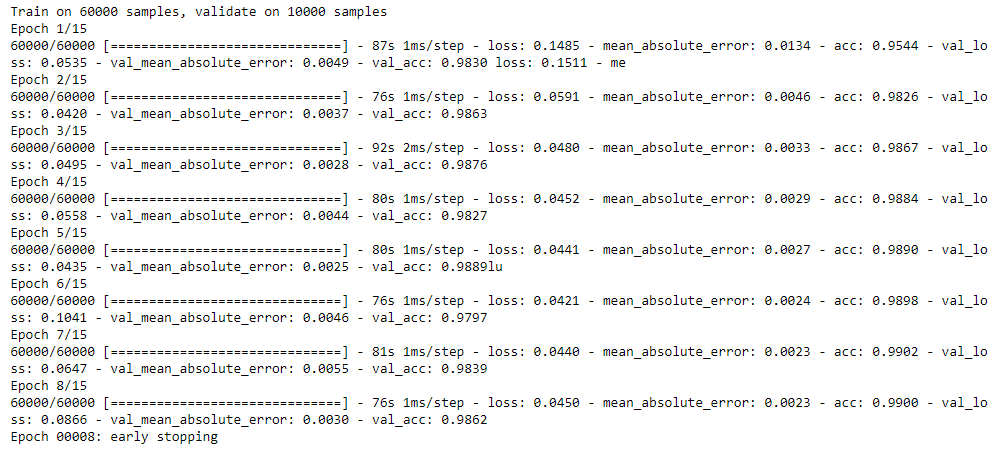
Trainable params: 46,112

Non-trainable params: 0

**Epoch – Accuracy Curve:**



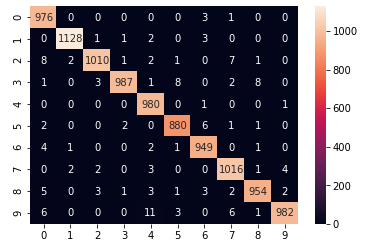
**Accuracy:**



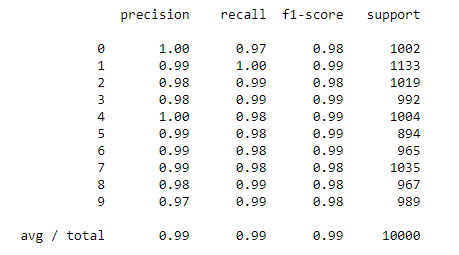
**Training accuracy: Early stopping accuracy – 98.9 Stopped model accuracy - 99**

**Test accuracy: Early stopping accuracy – 98.89 Stopped model accuracy – 98.62**

**Confusion Matrix:**



**Classification Report:**



**Setting 4: Filters:** Increase the number of filters in each CONV layer

**Properties:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Filters** | **Kernel Size** | **Stride** | **Activation** |
| **Conv2D** | 18 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Conv2D** | 25 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Flatten** | output = 400 | - | - | - |
| **Dense Layer 1** | units = 120 | - | - | - |
| **Dense Layer 2** | units = 84 | - | - | - |
| **Dense Layer 3** | units = 10 | - | - | - |

**Optimizer:** RMS prop with default learning rate (0.001)

**Loss:** Categorical Cross Entropy

**Epochs:** 7

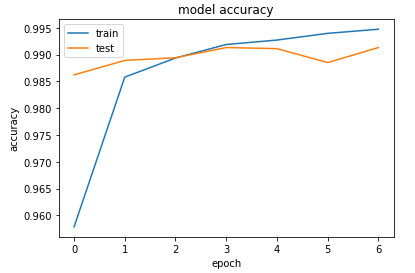
**Batch- size:** None

Total params: 97,877

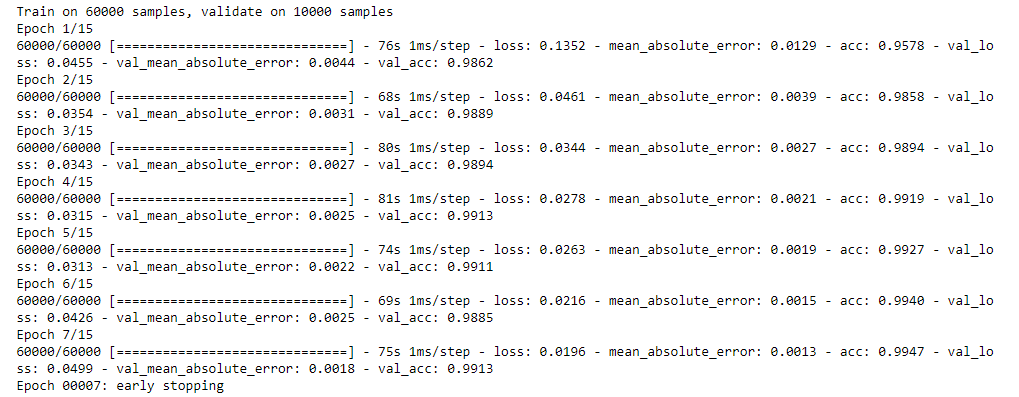
Trainable params: 97,877

Non-trainable params: 0

**Epoch – Accuracy Curve:**



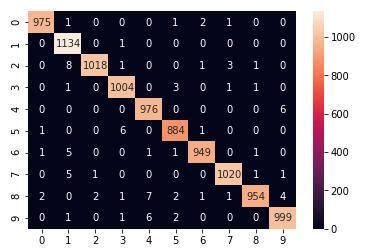
**Accuracy:**



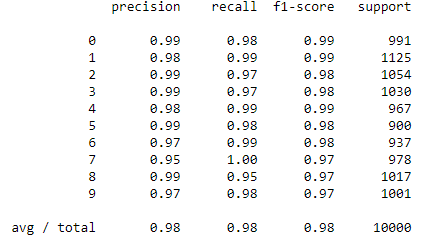
**Training accuracy: Early stopping accuracy – 99.19 Stopped model accuracy – 99.47**

**Test accuracy: Early stopping accuracy –99.13 Stopped model accuracy -99.13**

**Confusion Matrix:**



**Classification Report:**



**Setting 5: Kernel Size:** Decrease in kernel size

**Properties:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Filters** | **Kernel Size** | **Stride** | **Activation** |
| **Conv2D** | 18 | (3,3) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Conv2D** | 25 | (3,3) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Flatten** | output = 400 | - | - | - |
| **Dense Layer 1** | units = 120 | - | - | - |
| **Dense Layer 2** | units = 84 | - | - | - |
| **Dense Layer 3** | units = 10 | - | - | - |

**Optimizer:** RMS prop with default learning rate (0.001)

**Loss:** Categorical Cross Entropy

**Epochs:** 9

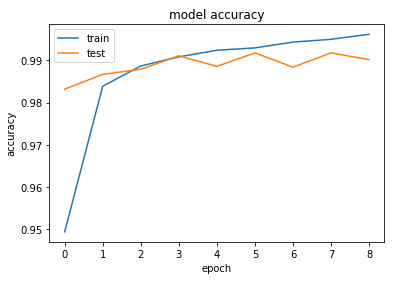
**Batch- size:** None

Total params: 123,389

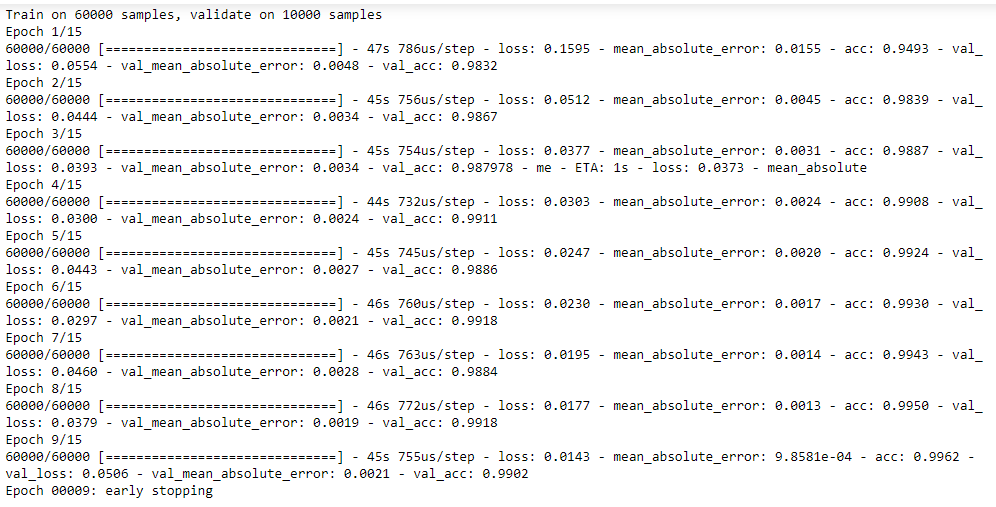
Trainable params: 123,389

Non-trainable params: 0

**Epoch – Accuracy Curve:**



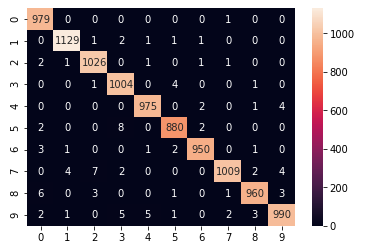
**Accuracy:**



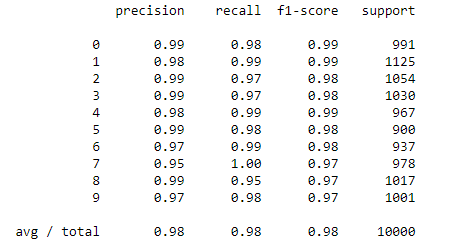
**Training accuracy: Early stopping accuracy – 99.3 Stopped model accuracy – 99.62**

**Test accuracy: Early stopping accuracy – 99.18 Stopped model accuracy – 99.02**

**Confusion Matrix:**



**Classification Report:**



**Setting 6: FC layer connections:** More number of fully connected units in dense layer

**Properties:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Filters** | **Kernel Size** | **Stride** | **Activation** |
| **Conv2D** | 18 | (3,3) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Conv2D** | 25 | (3,3) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Flatten** | output = 400 | - | - | - |
| **Dense Layer 1** | units = 250 | - | - | - |
| **Dense Layer 2** | units = 170 | - | - | - |
| **Dense Layer 3** | units = 10 | - | - | - |

**Optimizer:** RMS prop with default learning rate (0.001)

**Loss:** Categorical Cross Entropy

**Epochs:** 10

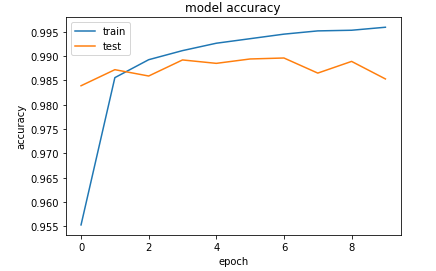
**Batch- size:** None

Total params: 273,885

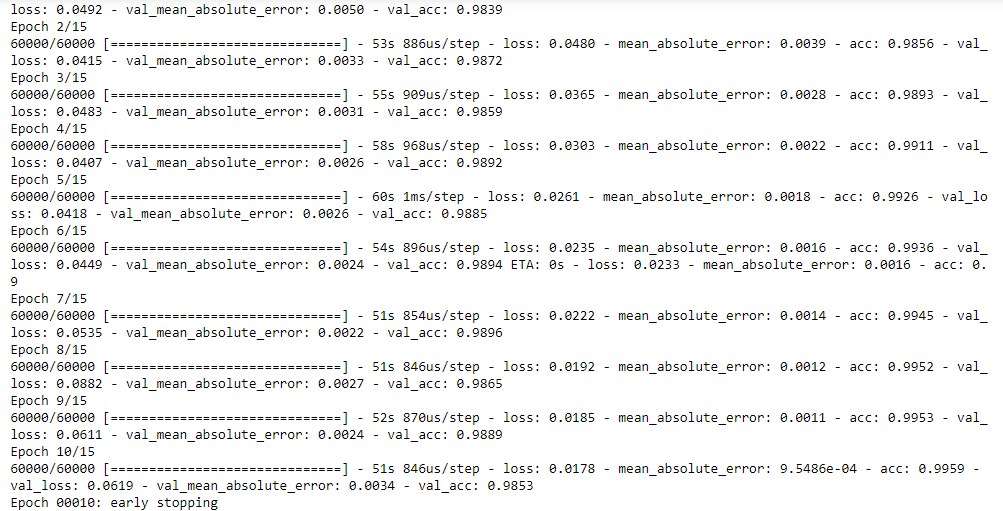
Trainable params: 273,885

Non-trainable params: 0

**Epoch – Accuracy Curve:**



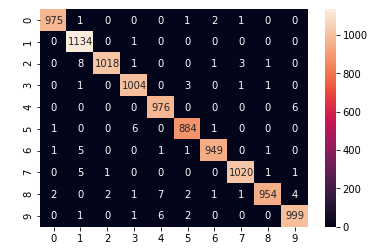
**Accuracy:**



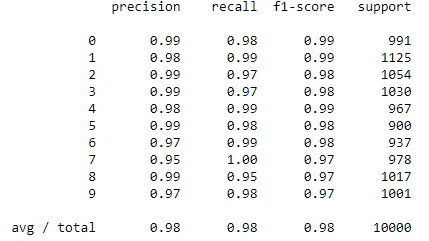
**Training accuracy: Early stopping accuracy – 99.45 Stopped model accuracy – 99.59**

**Test accuracy: Early stopping accuracy – 98.96 Stopped model accuracy – 98.53**

**Confusion Matrix:**



**Classification Report:**



**Setting 7: Max Pool Size:** Increase the kernel of the max pool layer

**Properties:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Filters** | **Kernel Size** | **Stride** | **Activation** |
| **Conv2D** | 6 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (3,3) | - | - |
| **Conv2D** | 16 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (3,3) | - | - |
| **Flatten** | output = 400 | - | - | - |
| **Dense Layer 1** | units = 120 | - | - | - |
| **Dense Layer 2** | units = 84 | - | - | - |
| **Dense Layer 3** | units = 10 | - | - | - |

**Optimizer:** RMS prop with default learning rate (0.001)

**Loss:** Categorical Cross Entropy

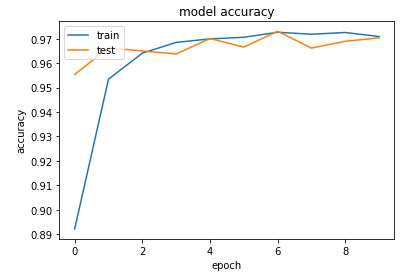
**Epochs:** 10

**Batch- size:** None

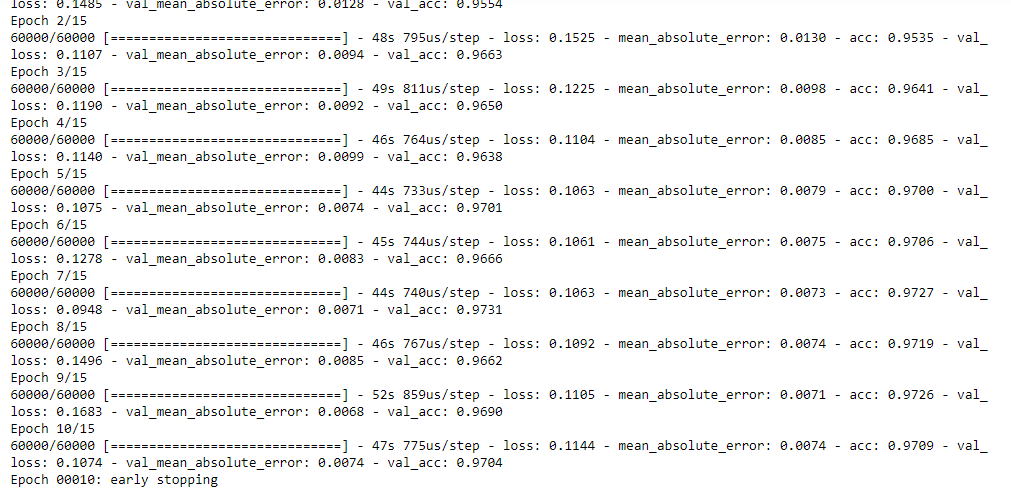
Total params: 15,626

Trainable params: 15,626

**Epoch – Accuracy Curve:**



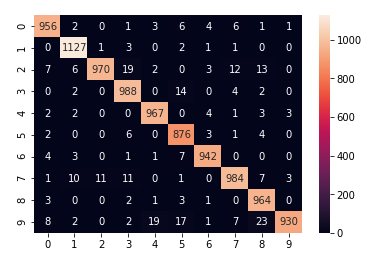
**Accuracy:**



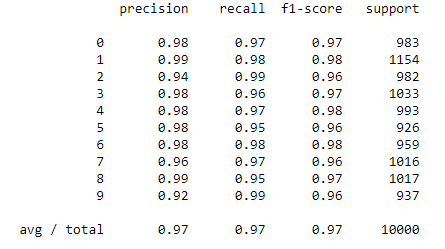
**Training accuracy: Early stopping accuracy – 97.27 Stopped model accuracy – 97.09**

**Test accuracy: Early stopping accuracy – 97.31 Stopped model accuracy – 97.04**

**Confusion Matrix:**



**Classification Report:**



**Setting 8: Learning rate:** High and low learning rate in optimizer

**Properties:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Filters** | **Kernel Size** | **Stride** | **Activation** |
| **Conv2D** | 6 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Conv2D** | 16 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Flatten** | output = 400 | - | - | - |
| **Dense Layer 1** | units = 120 | - | - | - |
| **Dense Layer 2** | units = 84 | - | - | - |
| **Dense Layer 3** | units = 10 | - | - | - |

**Optimizer:** RMS prop with default learning rate (0.001)

**Loss:** Categorical Cross Entropy

**Epochs:** 12 for lr=0.00001

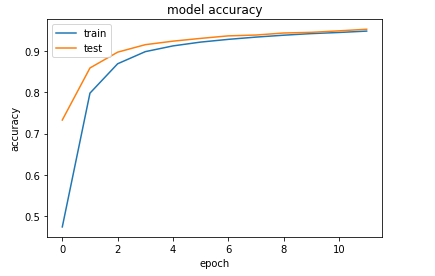
**Batch- size:** None

Total params: 61,706

Trainable params: 61,706

Non-trainable params: 0

**Epoch – Accuracy Curve:**

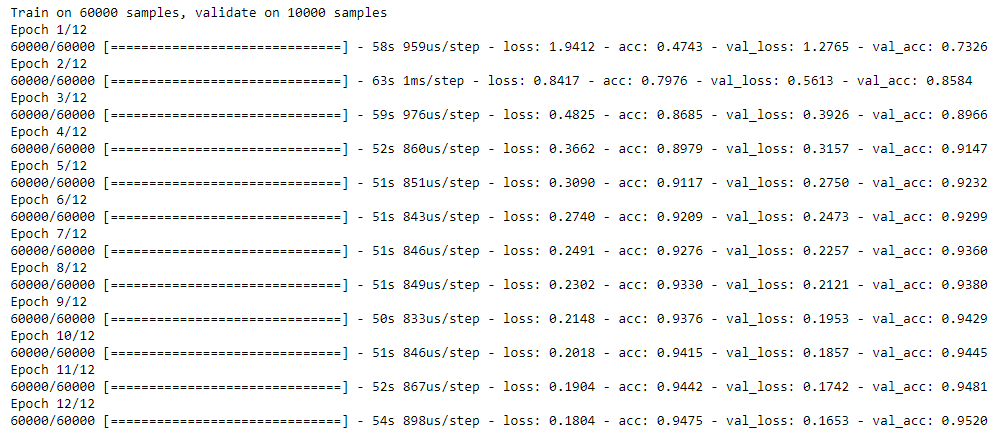


**Accuracy:**

Learning rate = 1



Learning rate = 0.00001



**Accuracy on training set: 94.75**

**Accuracy on test set: 95.2**

**Setting 9: Number of epochs:** Disable early stopping. Run for 30 epochs.

**Properties:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Filters** | **Kernel Size** | **Stride** | **Activation** |
| **Conv2D** | 6 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Conv2D** | 16 | (5,5) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Flatten** | output = 400 | - | - | - |
| **Dense Layer 1** | units = 120 | - | - | - |
| **Dense Layer 2** | units = 84 | - | - | - |
| **Dense Layer 3** | units = 10 | - | - | - |

**Optimizer:** RMS prop with default learning rate (0.001)

**Loss:** Categorical Cross Entropy

**Epochs:** 30

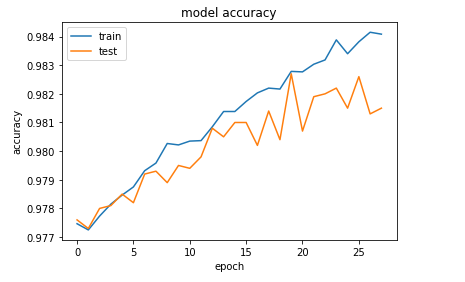
**Batch- size:** None

Total params: 61,706

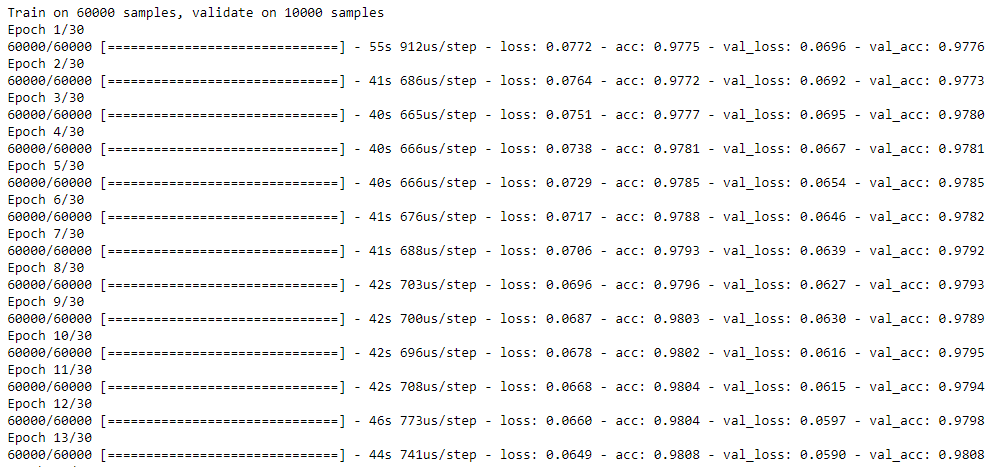
Trainable params: 61,706

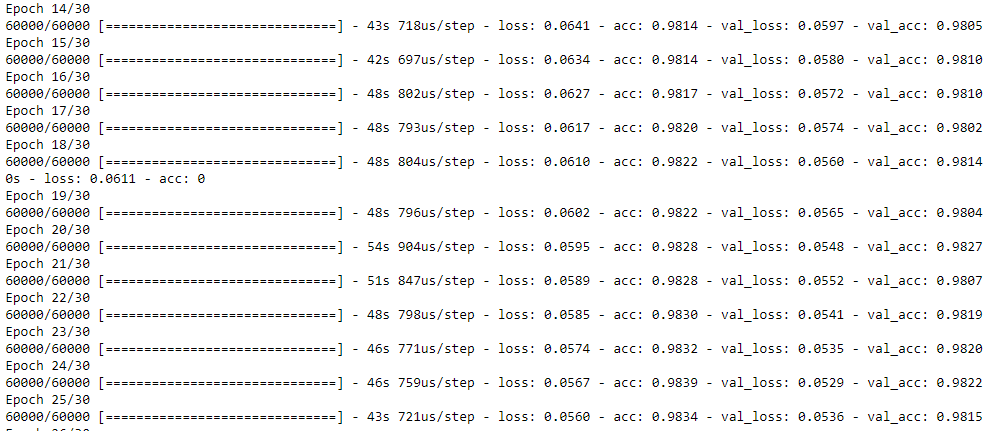
Non-trainable params: 0

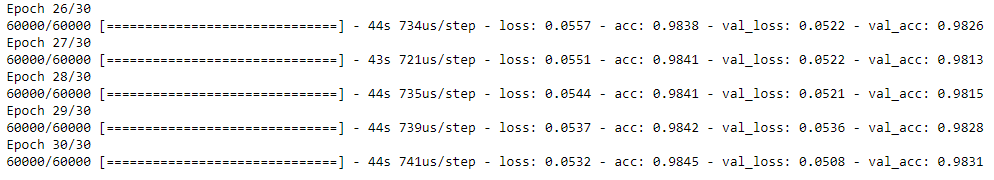
**Epoch – Accuracy Curve:**



**Accuracy:**







**Accuracy on the training set: 98.45**

**Accuracy on the test set: 98.31**

**Setting 10: Dropout and batch normalization:** Use the best settings from above with dropout layers and batch normalization

**Properties:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Filters** | **Kernel Size** | **Stride** | **Activation** |
| **Conv2D** | 18 | (3,3) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Conv2D** | 25 | (3,3) | 1 | relu |
| **MaxPool** | N/A | (2,2) | - | - |
| **Flatten** | output = 400 | - | - | - |
| **Dense Layer 1** | units = 120 | - | - | - |
| **Dense Layer 2** | units = 84 | - | - | - |
| **Dense Layer 3** | units = 10 | - | - | - |

**Optimizer:** RMS prop with default learning rate (0.001)

**Loss:** Categorical Cross Entropy

**Epochs:** 4

**Batch- size:** None

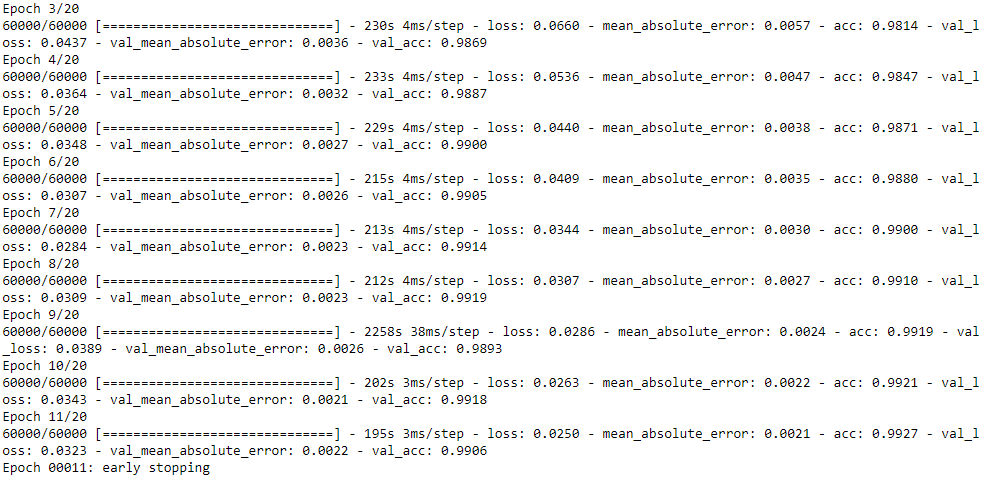
Total params: 123,561

Trainable params: 123,475

Non-trainable params: 86

**Epoch – Accuracy Curve:**

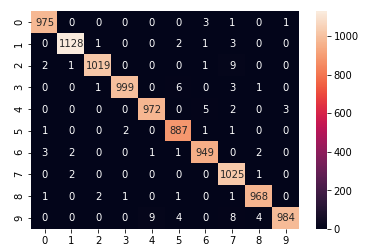
**Accuracy:**



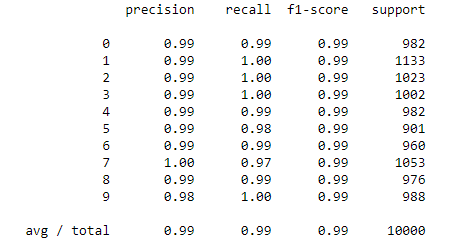
**Training accuracy: Early stopping accuracy – 99.1 Stopped model accuracy – 99.27**

**Test accuracy: Early stopping accuracy –99.19 Stopped model accuracy – 99.06**

**Confusion Matrix:**



**Classification Report:**



|  |  |  |
| --- | --- | --- |
|  | *Training accuracy* | *Test accuracy* |
| Setting 1 : Original LeNet Setting | 99.05 | 98.94 |
| Setting 2: Lesser filters in first CONV layer, more filters in second CONV layer | 95.91 | 98.27 |
| Setting 3: More filters in first CONV layer, lesser filters in second CONV layer | 98.9 | 98.89 |
| Setting 4: More filters in both CONV layers | 99.19 | 99.13 |
| Setting 5: Decrease kernel size | 99.3 | 99.18 |
| Setting 6: Increase dense units | 99.45 | 98.86 |
| Setting 7: Decrease pool size | 97.27 | 97.31 |
| Setting 8: Learning rate | 94.75 | 95.2 |
| Setting 9: Number of epochs | 98.45 | 98.31 |
| Setting 10: Batch Normalization and Dropout | 99.1 | 99.19 |

|  |  |  |
| --- | --- | --- |
|  | *Training accuracy* | *Test accuracy* |
| Setting 4: More filters in both CONV layers | 99.19 | 99.13 |
| Setting 5: Decrease kernel size with above setting | 99.3 | 99.18 |
| Setting 8: Learning rate | 94.75 | 95.2 |
| Setting 9: Number of epochs | 98.45 | 98.31 |
| Setting 10: Batch Normalization and Dropout | 99.1 | 99.19 |

*Mean of 5 training accuracies: 98.15799999999999*

*Mean of 5 testing accuracies: 98.202*

*Variance of training accuracies: 2.991255999999996*

*Variance of test accuracies: 2.363495999999997*

***Representative five settings:***

Setting 1 : Original LeNet Setting

*Part 1:* Setting 2, 3, 4: Filter size

*Part 2:* Setting 5: kernel size

Setting 6: dense units

Setting 7: pool size

*Part 3:* Setting 8: Learning rate

*Part 4:* Setting 9: Number of epochs

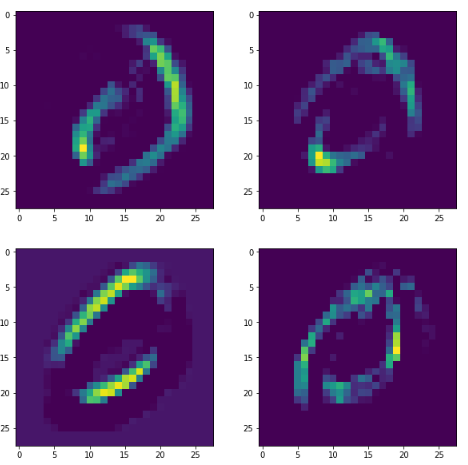
*Part 5:* Setting 10: Batch Normalization and Dropout

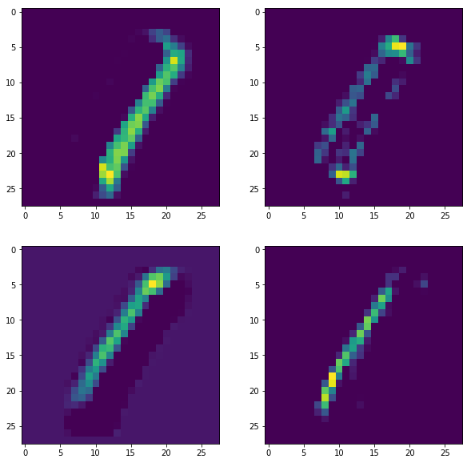
**4. Discussion:**

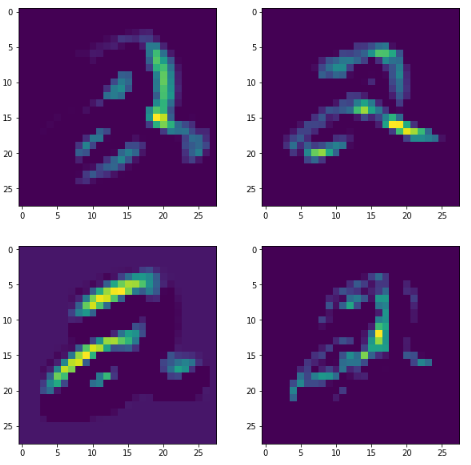
**How numbers 0, 1 and 2 look after each layer:**

**Setting 1 : Original LeNet Setting**

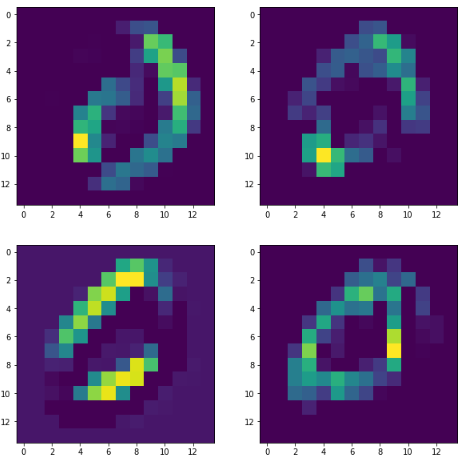
***Layer 1***

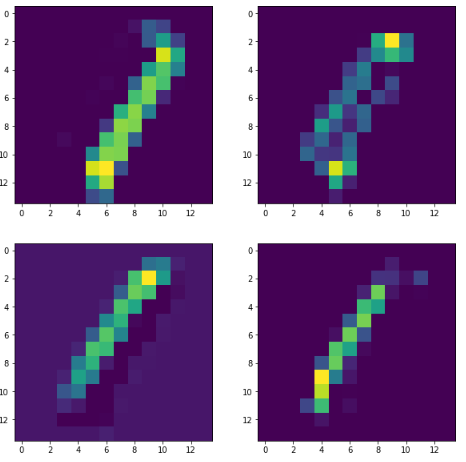


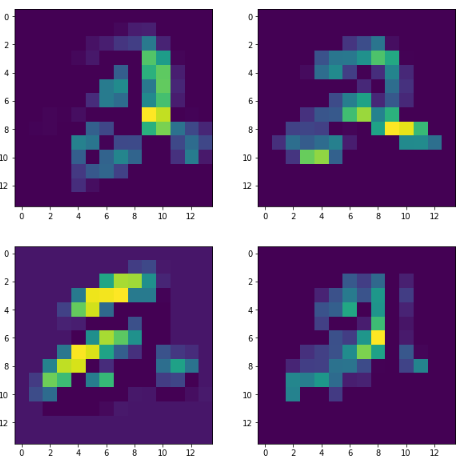




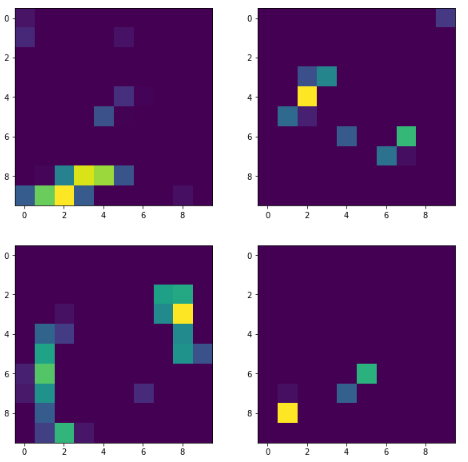
***Layer 2***

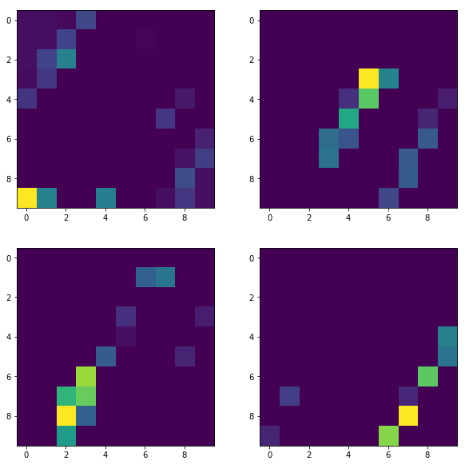


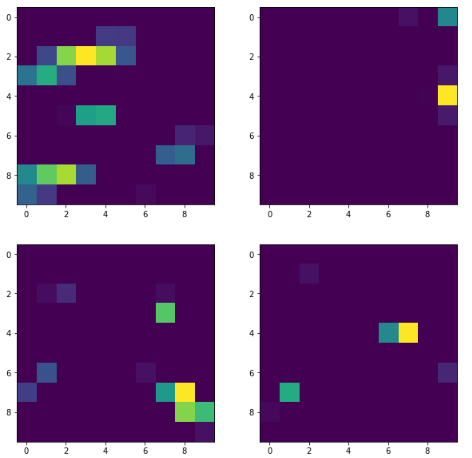




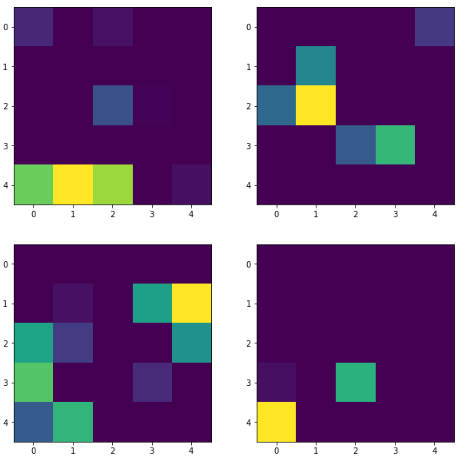
***Layer 3***

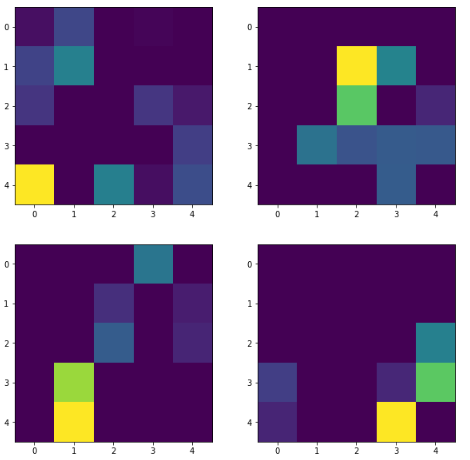


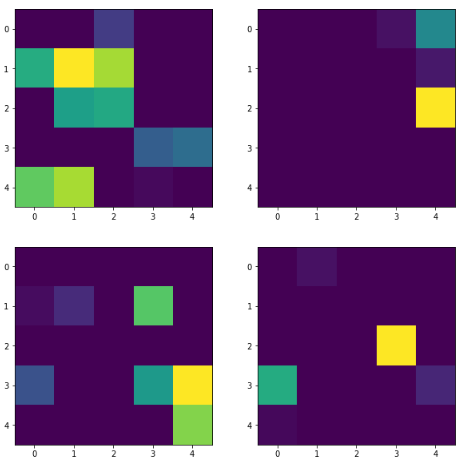




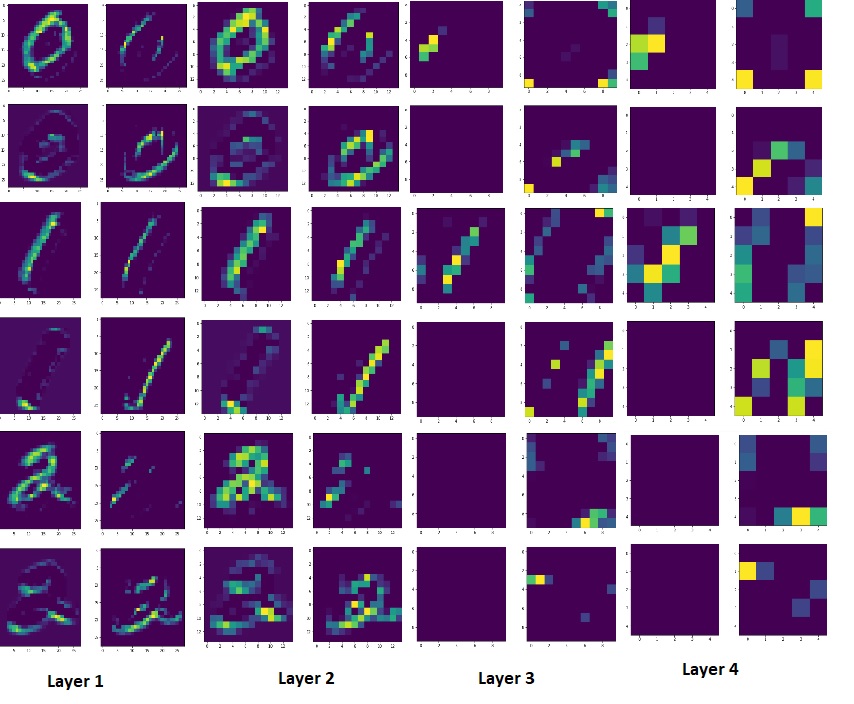
***Layer 4***







**Setting 2: Lesser filters in first CONV layer, more filters in second CONV layer**



**Setting 3: More filters in first CONV layer, lesser filters in second CONV layer**

**Setting 4: More filters in both CONV layers**

**Setting 5: Decrease kernel size**

**Setting 6: Increase dense units**

**Setting 7: Decrease pool size**

**Setting 10: Batch Normalization and Dropout**

Filtered images as per class : 0,1,2 in the first 4 filters of each layer

Effect of learning rate = 1



Low learning rate, more epochs to learn but gets better accuracy at the end of it all.

* **Batch size:**

Batch size does not affect the accuracy as much as the other factors. Batch size determines the number of samples that are considered in one forward and backward pass. Batch size determines the number of iterations needed to complete the epoch. A higher batch size would need a higher RAM in cases of large datasets since the system should have the computational resources to process multiple images in parallel. The larger the batch size, the quicker one epoch completes but quality degrades as the batch size increases beyond a certain point.

* **Number of filters:**

Each CONV layer will have a user defined number of filters to extract features. Each filter is initialized randomly from different distributions to stop two filters from detecting the same features. The following settings explored this parameter:

* **Setting 2:**

Here, we used less number of filters in the first layer and more number of filters in the second CONV layer. The initial CONV layers are used to learn abstract features. As we dive deeper into the network, the features will discern more specific features such as the eye, the nose and deeper would mean the whiskers of a cat. More number of filters in the first layer made the network learn more general features. It is now easy to make out that it is a number in the picture but what type of number is determined in the second CONV layer. Since lesser filters were included here, it gets harder for the network to differentiate between the numbers. Since the images were just numbers and the image size was only 28 x 28, this network did not suffer a huge blow in terms of accuracy but the decrease in accuracy can be seen as compared to the original LeNet setting.

* **Setting 3:**

Here, we used more number of filters in the second CONV layer and lesser filters in the first CONV layer. While specific features are learned better, the general features do not provide much basis for the second CONV layer filters to learn from. Therefore, this model will also undergo a dip in accuracy as can be observed.

* **Setting 4:**

So what happens if we increase the number of filters in both the CONV layers? More general features are learned and each filter in the second CONV layer will have as input each of the 18 filter feature maps in the first layer. This ensures that the feature maps in the network are quite thorough. As can be observed, the accuracy has improved to 99.13%.

* **Kernel size:**

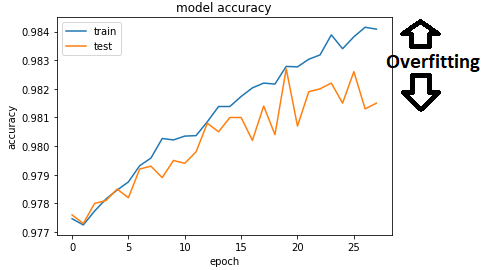
The kernel size corresponds to the size of the filter that is going to propagate/convolve through/alongside the image.

* **Setting 5:**

A smaller kernel size means the magnitude of the learned features in each entry of the filters are more accurate. However, a smaller kernel size also takes more time to run. An optimum kernel size is preferable if the required accuracy is achieved. A higher kernel size would result in loss of information and a faster training time.

* **Overfitting and epoch number:**
* **Setting 9:**

Overfitting and number of epochs are closely related. Although overfitting can be due to other factors, if the model is trained beyond a certain number of epochs, the training accuracy arcs upwards and the testing accuracy drops lower. This is because the model gets generalized over the training dataset. How fast the model generalized is a product of many factors including the size and variety of the dataset, the learning rate etc.



* **Early stopping:**

To prevent overfitting, regularization techniques are employed and one of those techniques are early stopping. In this case, patience is taken to be 3 (the number of epochs over which a drop in validation accuracy can be accepted before the model stops). This also removes the uncertainty of choosing the *right* number of epochs to run over. Instead, an upper limit is taken (in this case 15) and the model is run till validation accuracy is best and a few epochs after. The model with the best accuracy is saved and loaded later for testing.

* **Stride:**

The stride determines how many pixels the center of the pixel filter shifts over. A smaller stride parameter realized slower training and more accurate convolved entries. A larger stride may not significantly hurt the accuracy but it will reduce the time it takes to train the network. A stride larger than 3 will affect the accuracy visibly, especially for smaller images.

* **Max Pool Kernel**

Max pool kernel refers to the square clump of pixels considered to be reduced to one pixel, the one pixel being the highest magnitude pixel in that square clump.

* **Setting 7:**

A larger max pooling window means that more information is going to be lost. At the end of the day, pooling is just a downsampling operation and only entries with the highest energy are chosen. Therefore, max pooling window is kept as small as possible. i.e 2x2 unless of course, the image size is really big or the domain that the images are from are majorly consistent across their area. The main purpose of max pooling is to reduce computations while preserving information. If a higher kernel size is chosen, the time taken for execution will reduce drastically.

* **Dense units:**
* **Setting 6:**

Dense layer connections are used so that we can eventually arrive at the output probabilities of the ten classes. Increasing the number of dense unit connections causes the model to overfit and very few dense unit connections causes loss of accuracy. Therefore, in LeNet, on a trial and error basis, the number of dense layer connections are strategically chosen as 120 for the first dense layer and 84 for the second dense layer. Indeed, when it comes to choosing best settings options, these produce the best results.

* **Number of training variables:**

The number of training variables are calculated as follows:

For a conv layer, if the input is

1. an image in grayscale, input connections= 1
2. an image with RGB, input connections = 3
3. dense layer, input connection = number of dense units
4. conv layer, input connections = number of filters

The output connections of a conv layer is always the number of filters in that layer multiplied by the size of the size of each filter.

For a dense layer, the input is the number of connections is the number of connections from the previous layer. The output is the number of connections in that layer. If the previous layer is a CONV layer, then the **flattened** shape of the previous output is the input to the dense layer.

The number of trainable parameters in each layer are calculated by input x output + bias. The bias for CONV layer is the umber of filter in that layer and the bias for dense layers is the number of connections in that layer. The calculation in case of LeNet would be

*CONV layer 1* : (1 x 6x5x5) + 6 = 156

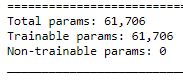
*CONV layer 2* : (6 x 16x5x5) + 16 = 2416

*Dense layer 1*: (400 x 120) + 120 = 48120

*Dense layer 2*: (120 x 84) + 84 = 10164

*Dense layer 3*: (84 x 10) + 10 = 850

Adding all the RHS terms, we get 61,702 learnable parameters. The answer is verified with the model summary from Keras.



In each case, the number of training variables vary. The more the filter size, the more number of training variables and the more the number of dense unit connections, the more the dense layer connections. If the kernel increases, the number of trainable parameters increase. If the max pool increases, the number of trainable parameters decrease as the flattened input for the dense connections decrease and the max pool layer does not have trainable parameters: it is just a downsampling layer.

* **Computational complexity:**

Computational complexity depends on a lot of factors:

1. number of epochs: The primary cause for runtime, the more the number of epochs, the more the runtime.
2. filter size: more the number of filters, more the computations
3. kernel size: lesser kernel size, lesser learning variables
4. max pool kernel size: larger the kernel, lesser computations
5. learning rate: more computations with lesser learning rate
6. training variables: more the number of training variables, more the time to train the network

* **Optimizer:**

Adam optimizer is nothing but rmsprop optimizer with momentum. Thus, we get better accuracies, since past gradients help smooth out the current gradient. It has a pre parameter learning rate that improves performance with sparse gradients. It combines the advantages of Adagad and rmsprop optimizers.

* **Loss:**

Categorical cross entropy is cross entropy functionality with softmax activation. It output a probability for each class. Since this is a multi class problem, categorical cross entropy is an obvious choice.

* **Learning rate:**
* **Setting 8:**

A very small learning rate infers slow convergence i.e. 0.001. As can be seen in the results, at 12 epochs, the accuracy is still 95%. This means that a few more epochs would be needed for absolute convergence. However, if the learning rate is really high i.e. 1. then it would mean that the global minimum might be skipped. This might mean that the algorithm might never converge. Therefore, a very high learning rate is strictly non advisable.

* **Choice of activation function:**

The advantages of ReLu are sparsity and reduced likelihood of vanishing gradient. While in sigmoid, the gradients of sigmoids become exceedingly small as the value of x increases, the constant gradient of ReLu results in faster learning. Sparsity is more for relu in CNNs which make it easier for computations. Therefore, relu is a direct choice.

* **Dropout and batch normalization:**

With dropout and batch normalization, we are generalizing the model. As a result, we can observe that the initial epochs have comparatively lower accuracy due to number of dense units converted to **non trainable parameters.**

* **Scope for improvement:**

1. Leaky relu can be used instead of relu to accommodate the range for truncated information since relu eliminates negative values which might result in loss of information.
2. Use more regularization methods

**(c)**

1. **Abstract and Motivation:**
2. **Approach and Procedures:**
3. **Results:**

**4.Discussion:**

**References:**

**[1]** <http://cs231n.github.io/convolutional-networks/>