ALBANY STATE UNIVERSITY, GA

WEB DESIGN AND DEVELOPMENT USING AI

CHAPTER FIVE – TRAIN MODEL TWO

BY

Wanjun Hu

Department Of Math, Cs & Physics

Albany State University

504 College Dr, Albany Ga

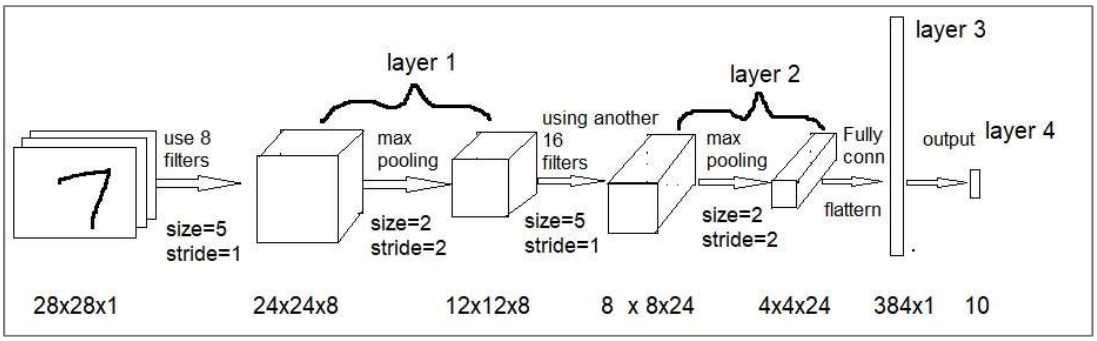
Wanjun.Hu@asurams.edu

https://www.backupspirit.com/camp

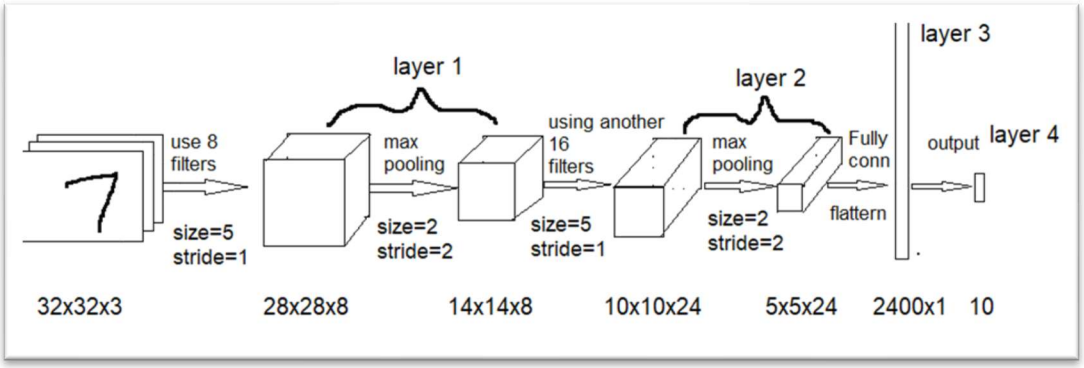
Table of Contents

Section 5.1 What is a convolutional neural network?

In chapter 3, section 3.7, we discuss the following CNN model.



In this chapter, we will train a very much similar model as shown below.



The Model

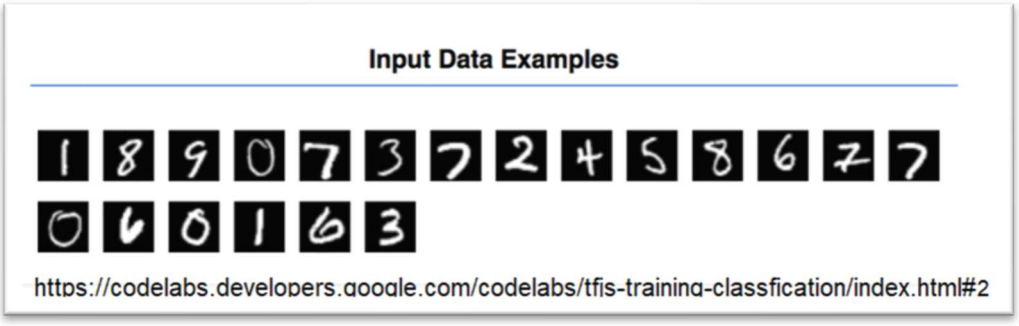
It is a 4-layer CNN model, with two convolution and pooling layers, one FC layer and one output layer. From left to right, we have the following layers (including input)

1. Input: we will provide a dataset of images of size 28 x 28 x 1, which are grayscale images of 0, 1,2,3,4,5,6,7,8,9, the ten digits. They are hand written.

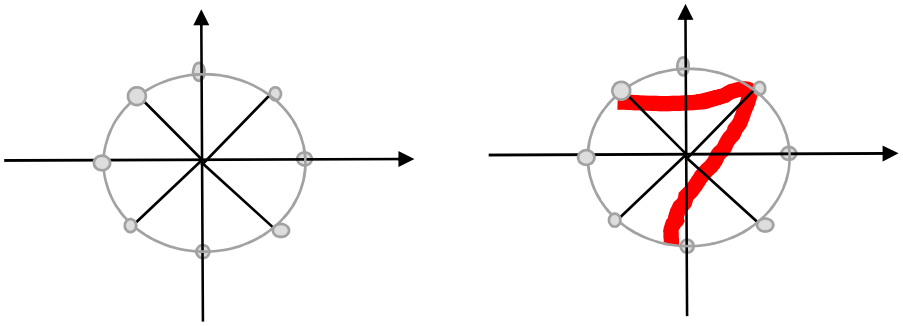
   The dataset is from Google, https://storage.googleapis.com/learnjs-data/model-builder/mnist_images.png.

   Its label is https://storage.googleapis.com/learnjs-data/model-builder/mnist_labels_uint8.

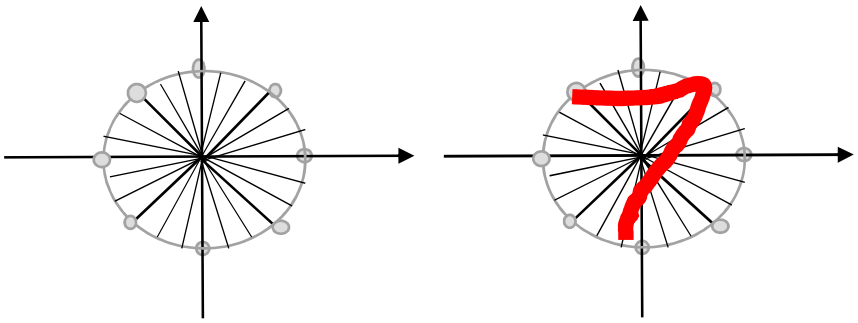
   The following are some examples from the dataset.



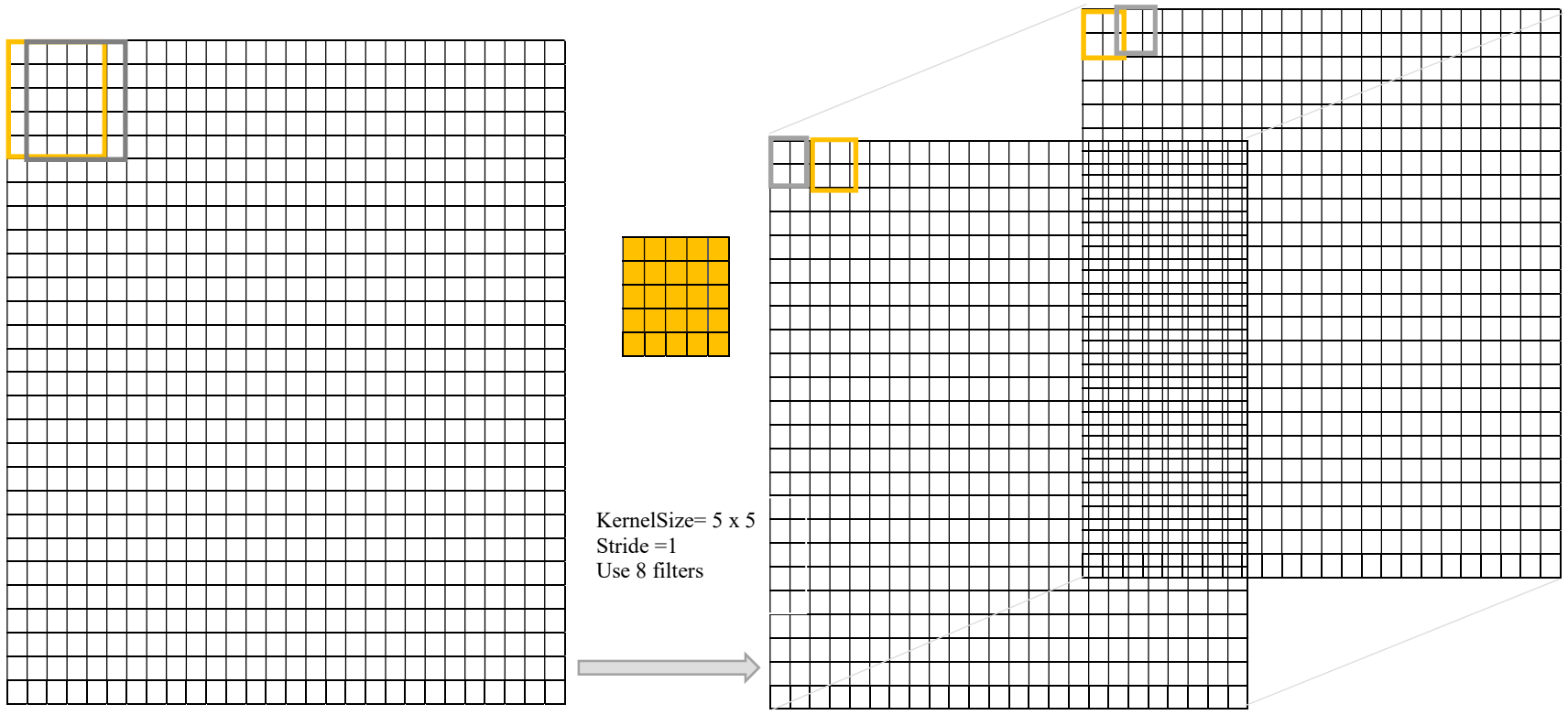
2. When go to layer 1, we first apply 8 convolution filters, each of kernel size=5 x 5 and stride=1. The 8 convolution filters are for edge/pattern along 8 angles,



3. For layer 2, we do that again. We use the same kernel size of the convolution. But, this time, we apply 16 filters for 16 angles.



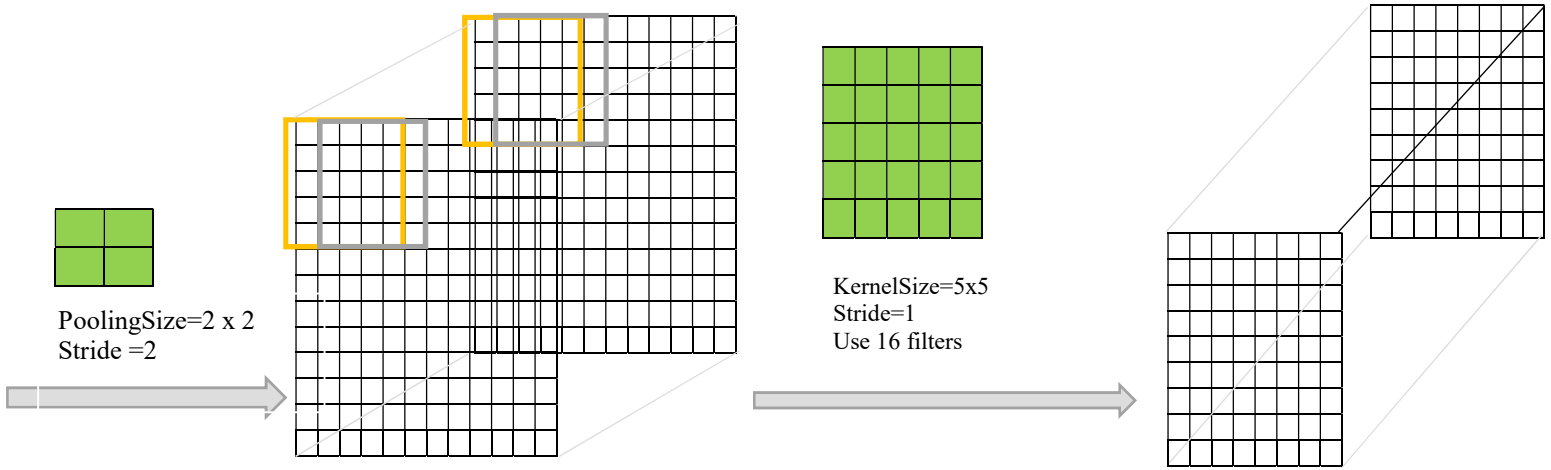
4. For layer 3, we flat the dataset obtained in layer 2, so that it can fully connected to the output layer, where predictions or classifications work.



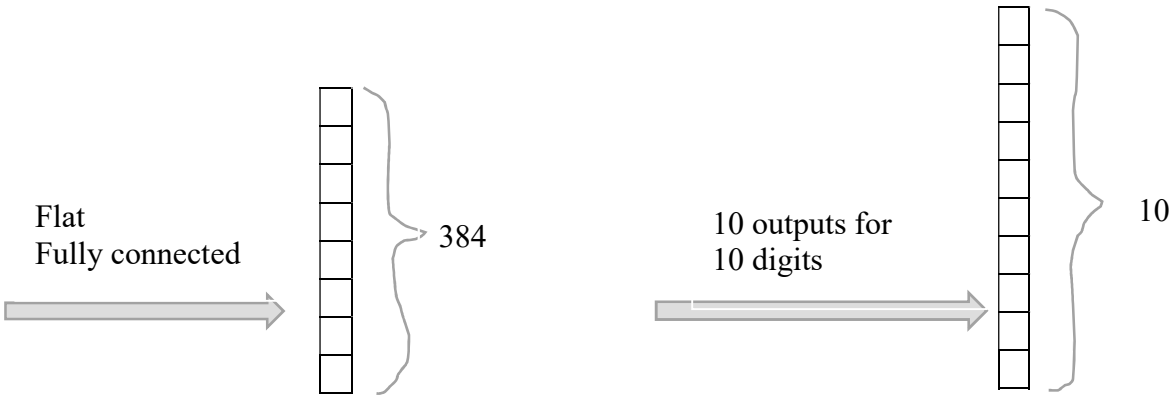
28 x 28 x 1

24 x 24 x8 (8 filters for different angles)

Layer 1: Convolution+Pooling



Layer 2: Another Convolution+Pooling



Layer 3: FC **Layer 4: Output**

Section 5.2 The training process of model two

Step 1: load the javascript libraries

We will need two javascript libraries from TensorFlow, one is the "tf.min.js" for defining the machine learning model, and the other is "tfjs-vis.umd.min.js" for browser visualization of training process. The links are

https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js, and

https://cdn.jsdelivr.net/npm/@tensorflow/tfjs-vis@1.0.2/dist/tfjs-vis.umd.min.js.

Step 2: Fetch the dataset and extract the targeting key:value pairs

We read the dataset from google, then extract above mentioned two key:value pairs from the dataset. The dataset is from Google, i.e., https://storage.googleapis.com/learnjs-data/model-builder/mnist_images.png. The dataset is arranged inside one picture files. There are 65,000 images in that file. Each image has size 28 x 28 x 1. The last one means it is a grayscale image with no color information. Each image is labeled as 0,1,2,…9. It is a supervised learning. The labels are in the same order as the images. The labels are in the file https://storage.googleapis.com/learnjs-data/model-builder/mnist_labels_uint8.

1. The dataset "window.dataset" contains 65000 images. Each image has size of 28 * 28 *1. So the image size = 28*28*1=784. Those images are classified into 10 classes, each representing one digit of 0,1,2,3,4,5,6,7,8,9. We will use 55000 images for training and the remaining 10000 for testing. Add a button in . When clicking, extract the first 5000 images from window.dataset and the first 5000 labels from window.labels. Save them to global variables too.

Step 3: Extract 55,000 images for training and save the remaining 10,000 for testing

We plan to randomly select 55,000 images and their corresponding labels for training and use the remaining 10,000 images and their labels for testing.

After that, we will do a shuffling and also normalization.

Step 4: Visualize the extracted dataset

Now, visualize the first 20 images.

Step 5: Define the machine learning model

We will use a sequential model. However, it has two layers of convolution+polling, one layer of fully connected, and the output layer.

For the input, the shape is 28x28x1.

For the first layer, we add a convolution with a filter size = 5 x5, stride=1, and a total of 8 filters. We use "*RELU*" as the activation function. Then, we add a pooling with pool size=2x2, stride=[2, 2]. We use the max pooling method, i.e., choosing the maximum element in the pool of 2x2. To improve the result, we also use a so-called *kernelInitializer* parameter and set it to *VarianceScaling*.

For layer 2, we add another convolution with filter size=5x5, stride=1, activation="*relu*", and a total of 16 filters. Following that is the pooling with pool size =[2,2], stride=[2,2]. For the third layer, we add a Fully connected, or simply use the flatten.

The last layer is the output layer, which has 10 outputs, with activation function "*softmax*". Again, we use the "*kernelInitializer*" parameter and set it to "*VarianceScaling*".

When train the model, we use the cost function "*Categorical Cross Entropy*" that corresponding to the *softmax* activation function. Both *softmax* activation function and the *Categorical Cross Entropy* cost function are rather complicated, which require college level mathematics. We will not go into details about them.

Step 6: Show a summary of the model

After create the model, use the tfjs.vis library to show a summary of the model to the web browser.

Step 7: Prepare the data to conver to TensorFlow tensor

We will need to convert each image (sample) to a TensorFlow tensor.

Step 7: train the model

Now, we can train the model using the selected images. We will need to setup two *hyperparameters* batch size and epochs.

Training data size: The total number of images, here 55,000. However, to train it using web browser, we will only use the first 5,000 images. Therefore, the training data size = 5,000.

Sample: Each image with its label is considered as one sample.

Batch: Each batch contains several samples. The cost function is a sum of individual loss of each sample in a batch.

Batch size: That number of samples in a batch is called *batch size*. The batch size can be any number between 1 and total number of images.

a. When batch size=1, we will calculate cost function and apply gradient descent 5,000 times and on each image. That is called ***Stochastic Gradient Descent***.

b. When batch size=5,000 (the total number of training samples), we will calculate cost function only once and apply gradient descent once. That is called ***Batch Gradient Descent***.

c. When batch size is a number between 2 and 5,000-1, say 512, we will calculate cost function and apply gradient descent 5,000/512 = 9 times. That is called ***Mini-Batch Gradient Descent***.

Epoch: By one epoch, we mean the training go over the whole training dataset once. Sometimes, we need to go over the whole training dataset more than once in order to reach a better result.

For this model, we set

Batch size = 512

Epochs = 10

When running the model, also visualize the training performance to web browser.

Step 8: Use the next 500 images to evaluate the trained model

We will use the next 500 images after the first 5,000 images for evaluating the trained model. We first calculate the predictions on those 500 images. Since the model output the probabilities of an image to be a digit, we will choose the highest probability as our prediction.

Then, we will show a plot that shows for each digit, how many samples are related and what is the accuracy of the prediction against the label of each image.

Finally, we will show a plot that shows for each digit, how many related samples are predicted accurately. If there are incorrect predictions, then how many samples are predicted to be 0, 1,2,3,4,5,6,7,8,9.

Section 5.3 AI prompts for creating webpage for model two

Use the following sequence of prompts.

2. Create a webpage that contains a banner showing “Summer CAM WDD Using AI”.
3. Make the background color to silver.
4. Set the webpage to use bootstrap framework.
5. inside <body>, add a <main> block.
6. inside the main block, add 20 's with id to be "div1", "div2", ...,"div20".
7. inside the , add headline 2, with content "Section 1. What is a convolutional neural network?". Under the headline 2, add an unordered list. The first list item shows "It tries to find an edge or pattern in an image". The second list item shows "It usually consists of one or more convolution plus pooling layer, and one fully connected (FC) layer”. The third list item shows "When doing convolution on an image, we use a 5x5 filter. Further, we may apply several such filters to detect edge/pattern along several lines" The fourth item shows "The cost function will be “. The fifth list item says "The last layer is output lay where we use a so-called Sigmoid activation, so that several outputs can be used." Inside the fifth list item, shows the picture "convolutionExample3.jpg" on a new line.
8. Resize the image "convolutionExample3.jpg" inside to 600 by 300.
9. add border bottom to all 's inside the main block.

10. inside , add a headline 2 “Section 2. Load TensorFlow for Java Script libraries” and then add a paragraph "We will use TensorFlow.js to train the model”. Then, add another paragraph, "Click the 'Run' button below to import two javaScript libraries". Then add a paragraph shows "import https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.0/dist/tf.min.js for defining and training models". Add another paragraph shows "import https://cdn.jsdelivr.net/npm/@tensorflow/tfjs-vis@1.0.2/dist/tfjs-vis.umd.min.js for web browser visualization".

11. Inside , add a button with content "Run". When clicking on the button, it will load the two libraries "https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.0/dist/tf.min.js", and "https://cdn.jsdelivr.net/npm/@tensorflow/tfjs-vis@1.0.2/dist/tfjs-vis.umd.min.js". Display the console output in

12. Inside , add a headline 2 “Section 3. The dataset of images”, and add a paragraph that says "We will use a dataset from Google, https://storage.googleapis.com/learnjs-data/model-builder/mnist_images.png, and their labels, https://storage.googleapis.com/learnjs-data/model-builder/mnist_labels_uint8". Add another paragraph, “The dataset is a big image file. It contains 65,000 images, each of size 28 x 28 x 1. The label file contains 65,000 labels. Each image use 10 numbers to indicate which digit the image represent.”

13. Inside , add a headline 2 “Section 4. The JavaScript module to process the dataset”. Then, add a paragraph “We will use a JavaScript class from ‘./data.js’ to process the dataset.” Then add a button "Load the dataset". When clicking the "Load the dataset" button, use the javascript class from data.js to process the dataset. Save an object of that class to a global variable called “mnistData”. Display the lengths of trainImages and trainLabels of mnistData to .

14. Inside , add a headline 2 “Section 5 We will show 20 images from the dataset”. Then add a button "Show the first 20 images". When clicking, use the TFJS_VIS 1.0.2 to display the first 20 images from mnistData inside . In addition, avoid using "tfvis.render.image" function.

15. Add the following code to the function that handles the button “Show the first 20 images”.

```
const label=examples.labels.slice([i,0], [1, 10]);
const labelDiv = document.createElement('div');
```

```
labelDiv.textContent = `Label: ${label}`;
surfaceElement.appendChild(labelDiv);
```

16. Inside , add a headline 2, “Section 6. Define a CNN.”. Then add a paragraph “”Our CNN has input of shape 28 x 28 x 1”. Then, add a paragraph, “The first layer is a convolution + pooling. We will use 8 filters, each filter has kernel size=5 x 5 and stride =1. The activation function is ‘relu’. We use the max pooling of size 2 x 2 with stride=[2,2]”. Then add a paragraph, “The second layer is also a convolution + pooling. This time, we use 16 filters each of kernel size=5x5 and stride =1. Th activation function is also ‘relu’. We use the max pooling of size 2 x 2 with stride=[2,2]”. Then add a paragraph, “The third layer is a fully connected layer with shape * x 1”. Finally, add a paragraph, “The last layer is the output layer, with 10 outputs representing 10 digits. The activation function is now ‘softmax’”.

17. Inside , add a button “Define the model”. When clicking, use TFJS to define a sequential model. Then add a conv2d layer with input shape = 28 by 28 by 1. There are 8 filters. Each filter has kernel size= 5 x 5. Strides=1. Activation=’relu’. Also set kernelInitializer=’varianceScaling’. Then add a max pooling 2d layer with pool size= 2 by 2 and stride= 2 by 2. Then add another 2d convolution layer with kernel size = 5, number of filters= 16, stride=1, activation=’relu’, kernelInitializer=’varianceScaling’. Then add a 2d max pooling with pool size=2 by 2 and strides=2 by 2. Then, add a 1d fully connected layer. Finally add the output layer of 10 outputs, with kernelInitializer=’varianceScaling’ and activation=’softmax’. Display the console output inside .

18. Inside the function handling the button “Define the model”, use tfjs_vis to display the model summary inside

19. inside the function defineModel, set the optimizer of the model to "adam", and use "categoricalCrossentropy" as cost function. The metrics to use are "accuracy". Then compile the model.

20. In the function handling the button "Define the model" inside , save the defined model to a global variable.

21. Inside , add a headline 2 “Section 7. Train the model using 5000 images”. Then add a paragraph “The batch size = 512, and epochs = 10”. Then, add a paragraph “The test size = 1000”. Add a paragraph “The callbacks have

name='Model Training', tab='Model', styles='{height: 1000px'}, metrics={'loss', 'val_loss', 'acc", 'val_acc'".

22. Inside , add a button "Train the model". When clicking, the model globalModel is trained using the first 5000 images and first 1000 images for testing from mnistData. Set batch size=512, epochs=10. Set the metrics to {"loss", "val_loss", "acc", "val_acc"}, and set name="Model Training", tab="Model" and styles="{height: 400px;}".

23. When clicking the button "Train the model", first compile the globalModel using optimizer="adam", cost="categoricalCrossentropy" and metrics=['accuracy']. Then use mnistData.nextTrainBatch to obtain 5000 train Examples, and mnistData.nextTestBatch to obtain 1000 test examples. Reshape the first parameters of both train examples and test examples to the right size. To train the model, use the train examples and test examples, Set batch size=512, epochs=10. Set the metrics to {"loss", "acc"}, and styles="{height: 400px;}".

24. inside the trainModel function, change trainData,x to trainData,xs, and trainData.y to trainData,labels. Also change the testData.x to testData.xs, and testData.y to testData.labels.

25. Inside , add a headline 2 "Section 8. Evaluate the model". Then, add a paragraph "We will obtain 500 images from the dataset mnistData". Add another paragraph, "We use the trained model to calculate the predictions and then compare the predictions with the provided labels". Add another paragraph, "We then draw a confusion matrix to show for each digit, how many images are labeled for that digit and how many images that predicted to that digit. Further, how many difference".

26. Inside , add a button "Show a confusion matrix". When clicking, use the mnistData.nextTestBatch to obtain 500 examples. Reshape the first parameter of the examples to the right size. Save the second parameter. Then, use the trained model and the images from the examples to calculate the predictions. Use the tfjs_vis library to show per class accuracy between the predictions and labels. Then use tfjv_vis to show the confusion matrix.

27. Inside the function showConfusionMatrix, the variable model should be globalModel.

Section 5.4 Link to the final product

Too see the final product using AI, please visit https://www.backupspirit.com/camp/model2.html.

Question: What is a convolution?

Answer: It contains two layers

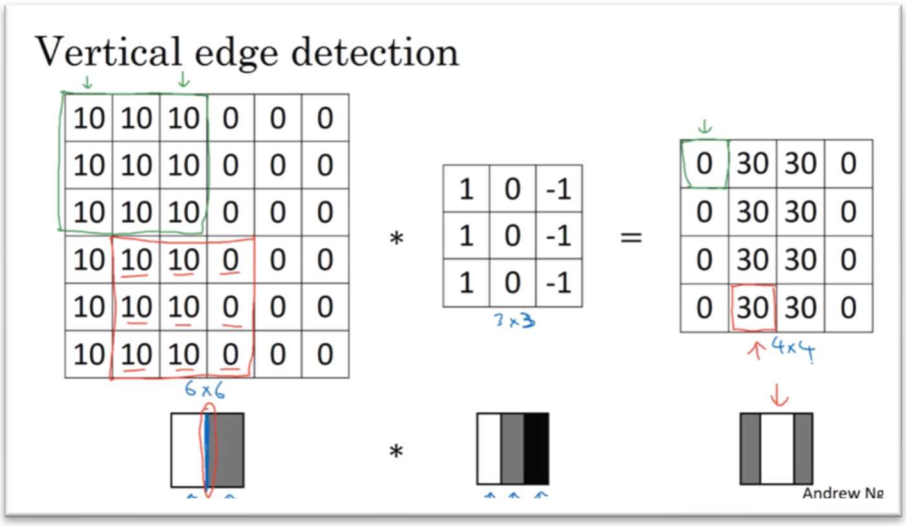
a. Convolution and pooling layer

1. Convolution to detect edge, patterns
2. Pooling to down sample the size of data for fast computations

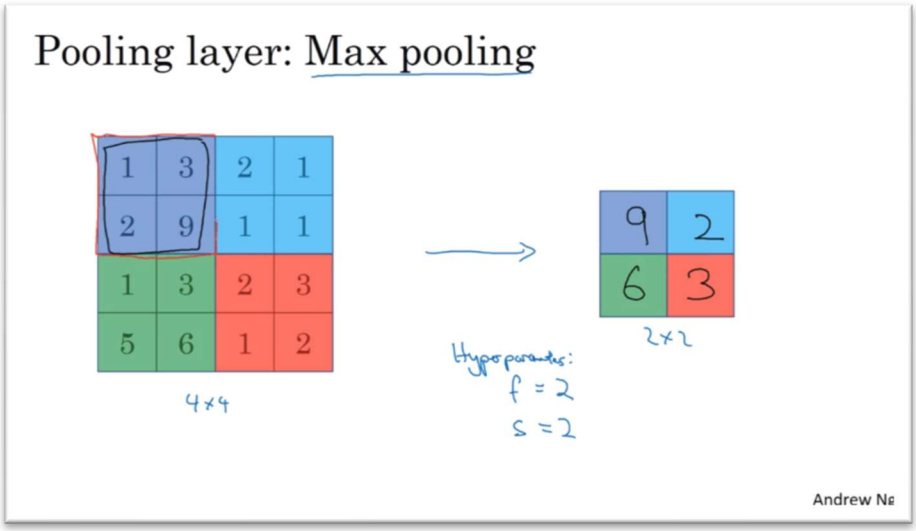
b. Fully connected layers that flats multi-dimensional dataset to one dimensional so that it can connect fully to the last layer, i.e., the output layer.

Convolution and pooling layer -- part 1

– convolution to do edge detection using filters (also called kernels)



Usually, we apply other filters to detect edges at different angles.



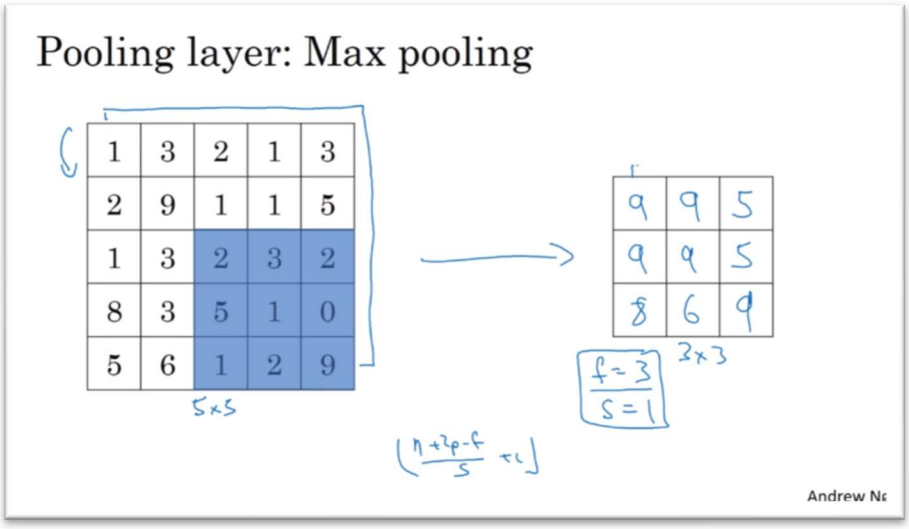
Convolution and pooling layer -- part 2

--Pooling to reduce the size of dataset for fast computation

Max pooling with filter=2, stride=2

Max pooling with filter=3, stride=1

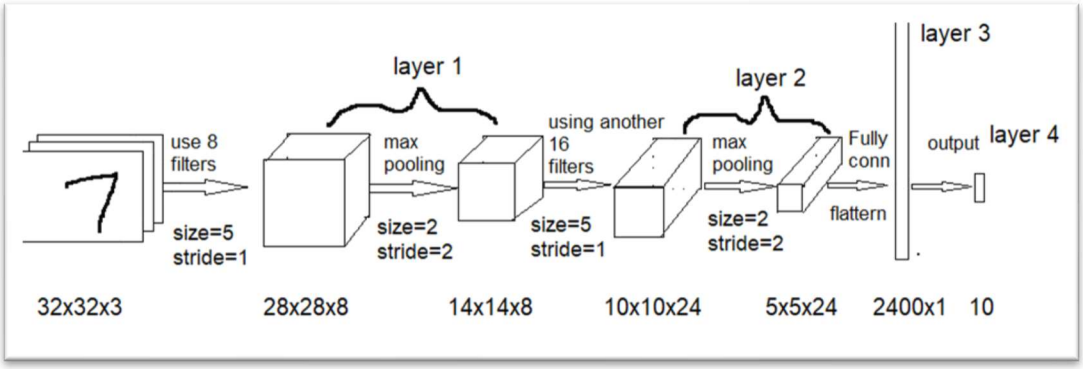
Fully connected Layer



Flat a multi-dimensional dataset to 1 dimensional dataset, so that it can be fully connected to the last layer, the output layer, where predictions or classifications are made. In the graph below, it is the layer 3.

Putting together

The follow graph shows a simple convolutional neural network (CNN) that can recognize hand written digits. It is a 4-layer CNN.



We will train above CNN model using a dataset from Google.