Implement neural network and convolutional neural network on Fashion-MNIST clothing images

---

Pang Yen Wu
Department of Computer Science
University at Buffalo
Buffalo,NY 14214
wupangye@buffalo.edu

Abstract

This is the report for CSE474 Project 2. In this report will present how three different type of neural network perform on the Fashion-MNIST clothing images. Neural Network with one hidden layer second one is build multi-layer Neural Network, third one is build Convolutional Neural Network(CNN)

Introduction

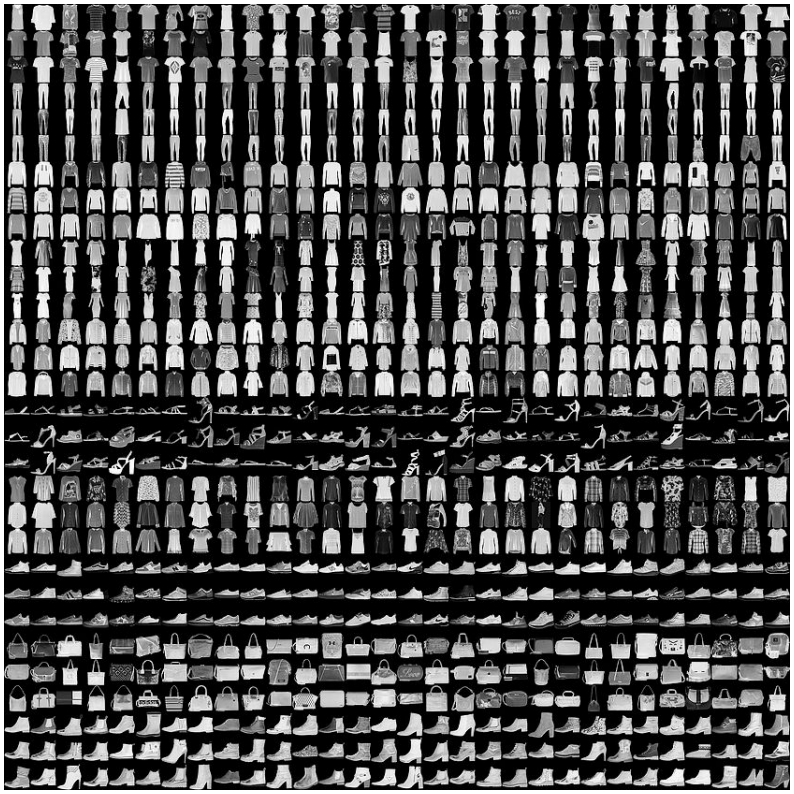
Neural Network, more properly referred to as an 'artificial' neural network(ANN), the inventor of one of the first neurocomputers, Dr.RObert Hecht-Nielsen. He defines a neural network as " a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs." In other way you can think that Artificial Neural Network as computational model that is inspired by the way biological neural networks in the human brain process information

Dataset

For this project we are using the data set from Fashion MNIST dataset it includes
1.60000 training examples
2.10000 testing examples
3.10 classes
4. 28x28 grayscale/single channel images

The ten fashion class labels:

1.T-shirt/top
2.Trouser/pants
3.Pullover shirt
4.Dress
5.Coat
6.Sandal
7.Shirt
8.Sneaker
9.Bag
10.Ankle boot



Pre-Processing

Initializing weights and biases

In the given dataset y_train is like(60000,) y_test is like(10000,) it didn't specify how many class is there so we use OneHotEncoder to convert your label data to one-hot encoded form.

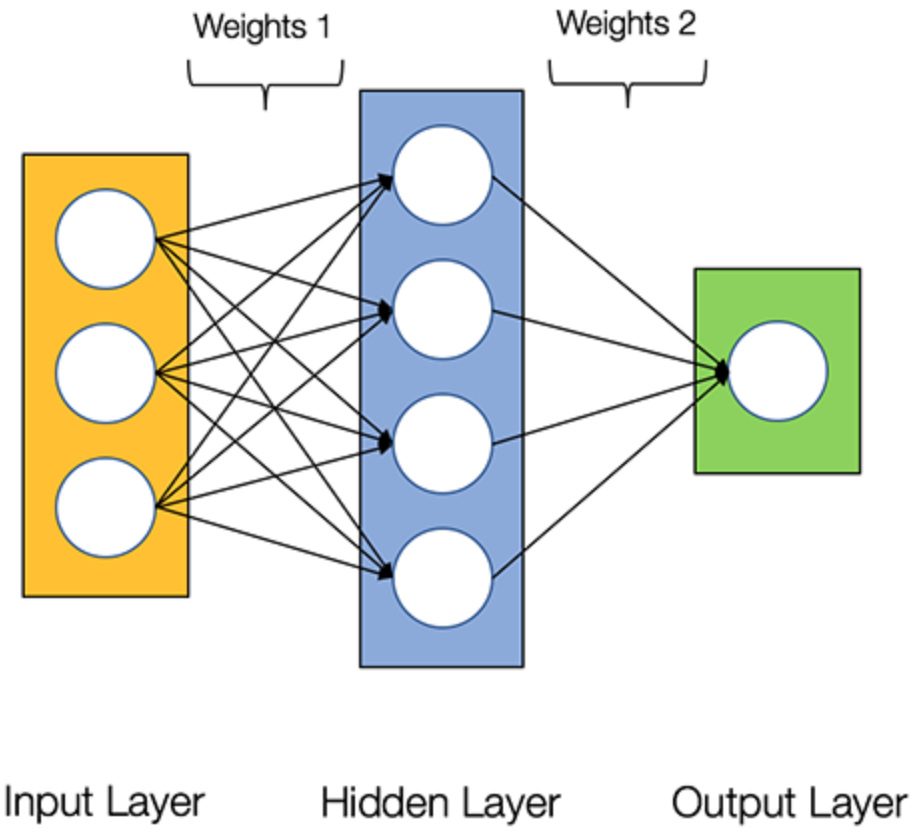
We normalize the data dimensions so that they are approximately the same scale

We use scikit-learn's scale to normalize our data.

For build multi-layer neural network, we try to reshape data from 2D to 3D, For building Convolutional Neural Network , after loading and splitting the data, I preprocess them by reshaping them into the shape network expects and scaling them so that all values are in the [0,1] interval. The training data were stored in an array of shape(60000,28,28) of type uint8 with values in the [0,255] interval. I transform it into a float32 array of shape(60000,28*28) with values between 0 and 1

Architecture

Task one, neural network with one hidden layer



Neural Network consist of the following components

1. An input layer, x
2. An arbitrary amount of hidden layers
3. An output layer, $\hat{y}$
4. A set of weights and biases between each layer, W and b
5. A choice of activation function for each hidden layer

The output $\hat{y}$ of a simple 2-layer Neural Network is:

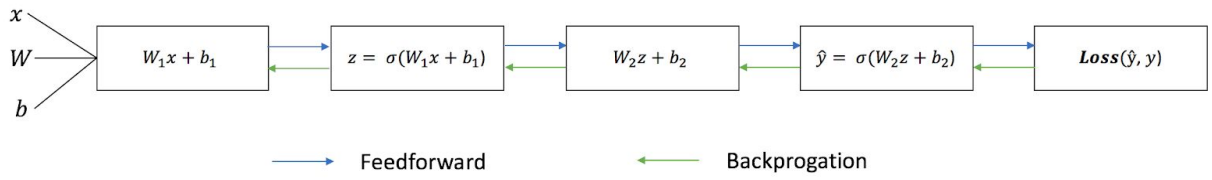
$$\hat{y} = \sigma(W_2 \sigma(W_1 x + b_1) + b_2)$$

weights W and the biases b are the only variables that affects the output $\hat{y}$
The right values for the weights and biases determines the strength of the predictions. The process of fine-tuning the weights and biases from the input data is known as training the neural network.

Each iteration of the training process consists of the following steps
1.Calculating the predicted output $\hat{y}$, known as feedforward

2.Updating the weights and biases, known as backpropagation

The graph below show the process:



For the multi-layer neural network, which is just the neural network that has more than one hidden layer, is also called the Deep learning neural network.

Training the first Neural Network Model
1.Build the architecture
2.Compile the model
3.Train the model
4.Evaluate the model

Build the architecture:
Build the architecture deciding the number of layers and activation functions. We'll start with a 3-layer Neural Network. In the first layer we 'flatten' the data, so that a(28*28)shape flattens to 784. The second layer is a dense layer with a ReLu activation function and has 128 neurons. The last layer is a dense layer with a softmax activation function that classifies the 10 categories of the data and has 10 neurons.

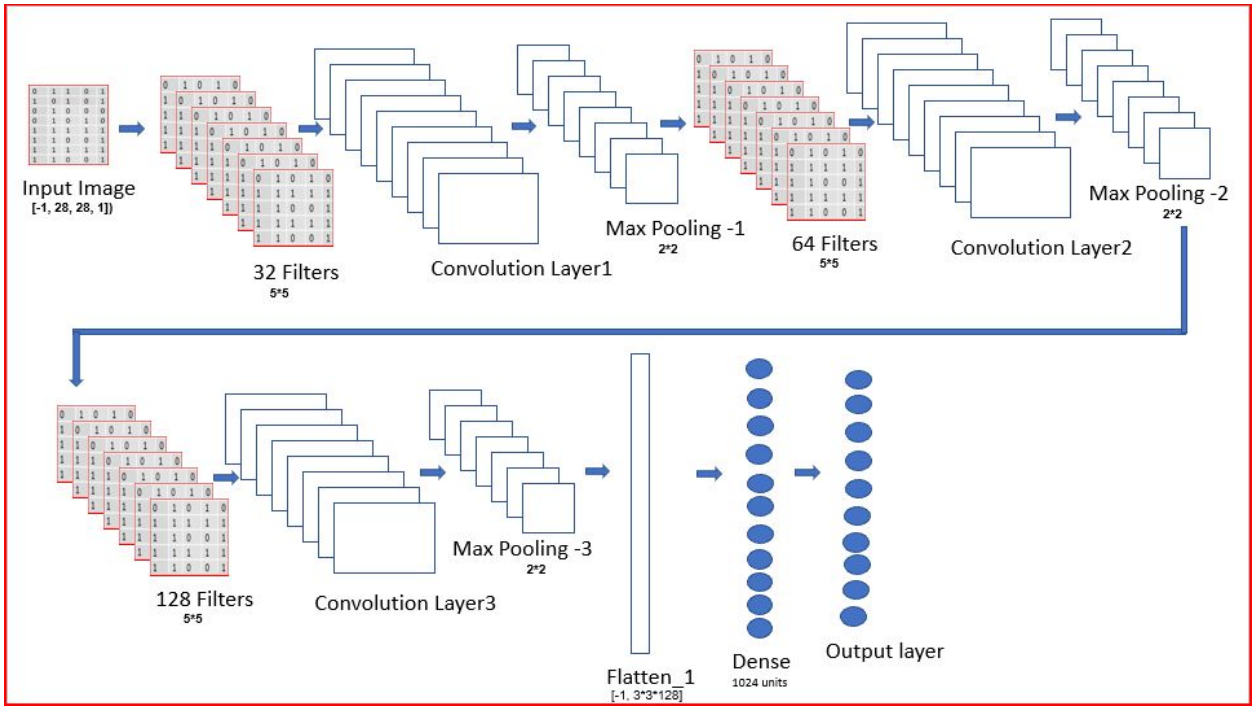
Compile the model:

1. Lost function: calculates the difference between the output and the target variables. It measures the accuracy of the model during training and we want to minimize this function

   Cross-entropy is the default loss function to use for a multi-class classification problem and it's sparse because our targets are not one-hot encodings but are integers

2. Optimizer: how the model is updated and is based on the data and the loss function. Adam is an extension to the classic stochastic gradient descent.
3. Metrics-monitors the training and testing steps. Accuracy is a common metric and it measures the fraction of images that are correctly classified.

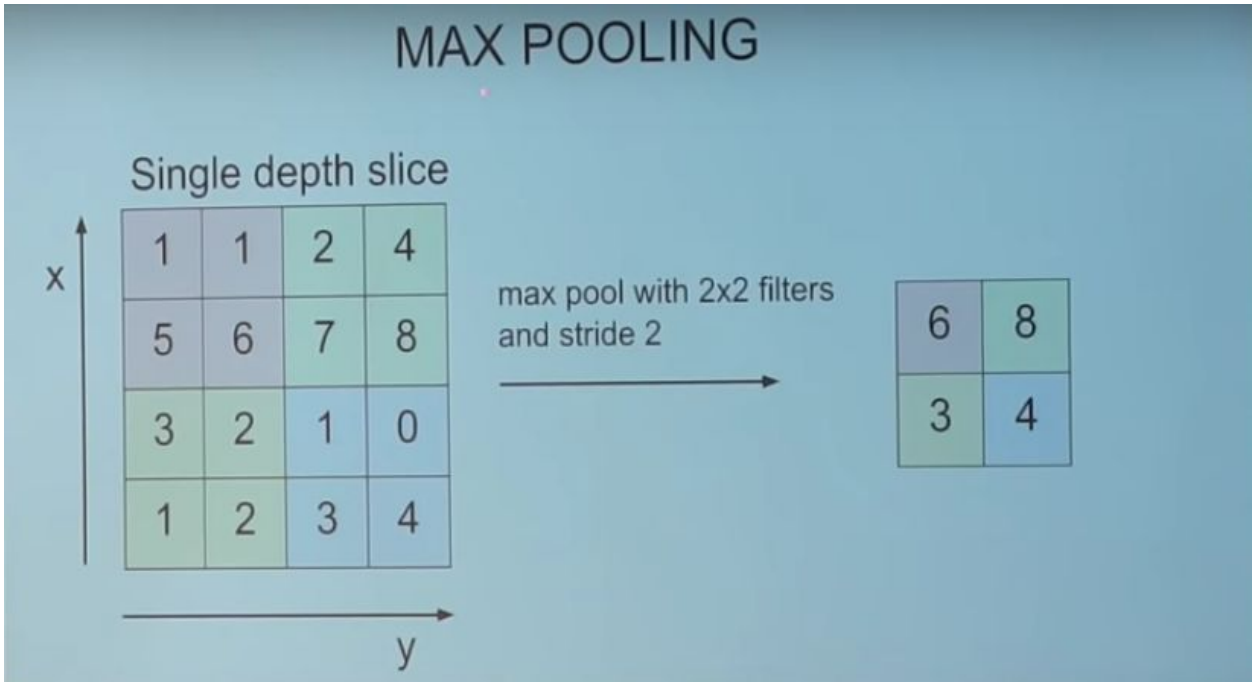
The third task Convolutional neural network



This CNN takes as input tensors of shape(image_height,image_width,image_channels). In this case, I configure the CNN to process inputs of size(28,28,1) which is the format of the FashionMNIST images. I do this by passing the argument input_shape=(28,28,1) to the first layer

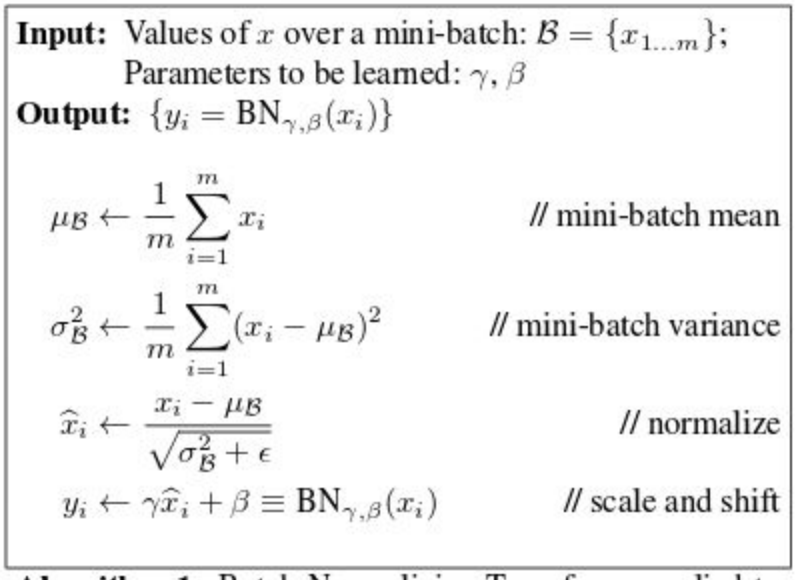
The Conv2D layers are used for the convolution operation that extracts features from the input images by sliding a convolution filter over the input to produce a feature map. Here I choose feature map with size 3x3 The MaxPooling2D layers are used for the max-pooling operation that reduces the dimensionality of each feature, which helps shorten training

time and reduce number of parameters. Here I choose the pooling window with size 2x2



Normalize the input layers, I use BatchNormalization layers to adjust and scale the activations.Batch Normalization reduces the amount by what the hidden unit values shift around(covariance shift). Also, it allows each layer of a network to learn by itself a little bit more

independently of other layers.



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

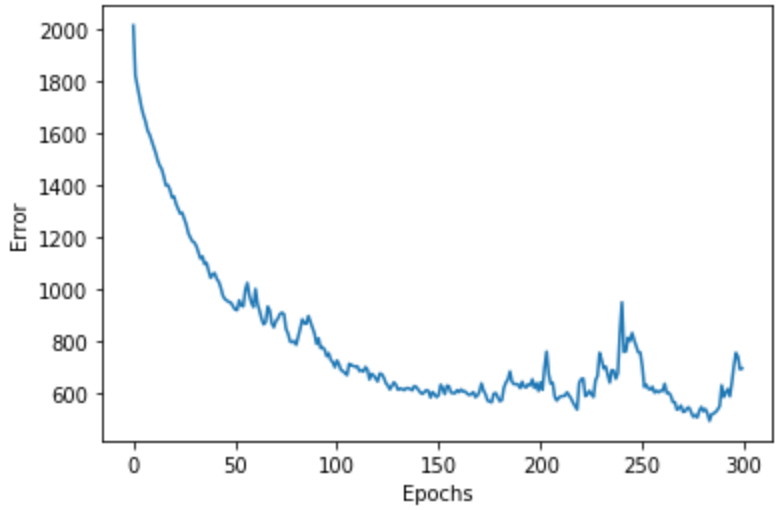
To combat overfitting, I add a *Dropout* layer as the 3rd layer, a powerful regularization technique. **Dropout** is the method used to reduce overfitting. It forces the model to learn multiple independent representations of the same data by randomly disabling neurons in the learning phase. In this model, dropout will randomly disable 20% of the neurons.

The next step is to feed the last output tensor into a stack of *Dense* layers, otherwise known as **fully-connected** layers. These densely connected classifiers process vectors, which are 1D, whereas the current output is a 3D tensor. Thus, I need to **flatten** the 3D outputs to 1D, and then add 2 *Dense* layers on top.

I do a 10-way classification (as there are 10 classes of fashion images), using a final layer with 10 outputs and a softmax activation. **Softmax** activation enables me to calculate the output based on the probabilities. Each class is assigned a probability and the class with the maximum probability is the model's output for the input.

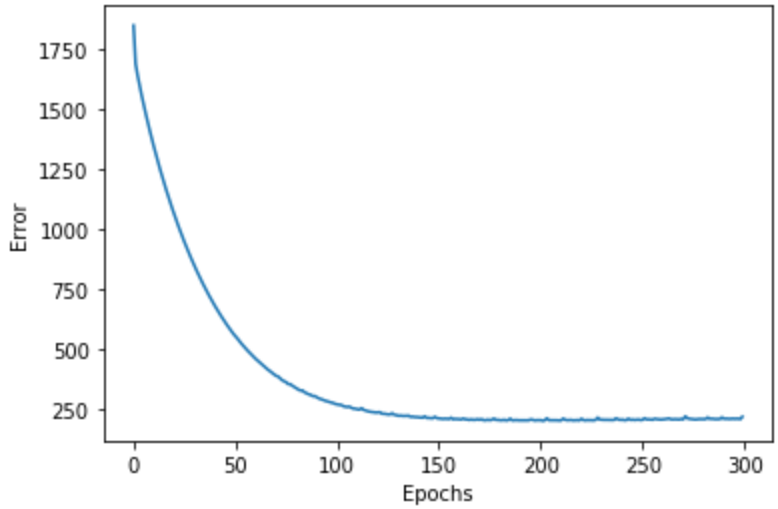
Results

One Hidden layer
300 epochs
learning rate 0.001
Number of batches 25
Number of hidden units = 50

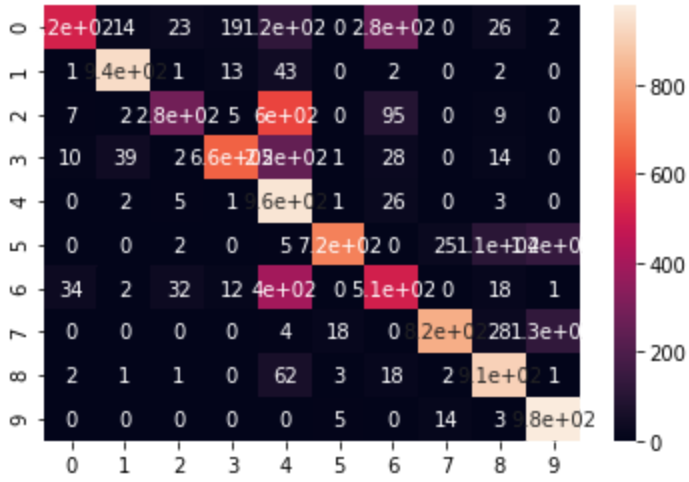


When the data is normalized, ude scikit-learn's scale to normalize our data, center to the mean and component wise scale to unit variance

One Hidden layer
300 epochs
learning rate 0.001
Number of batches 25
Number of hidden units = 50



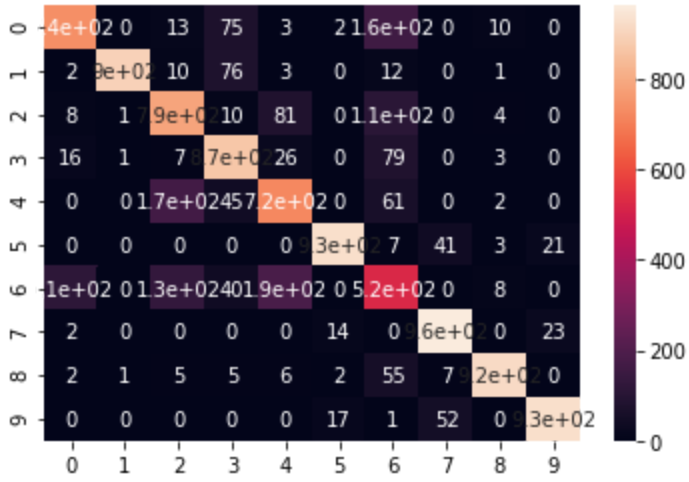
Confusion Matrix



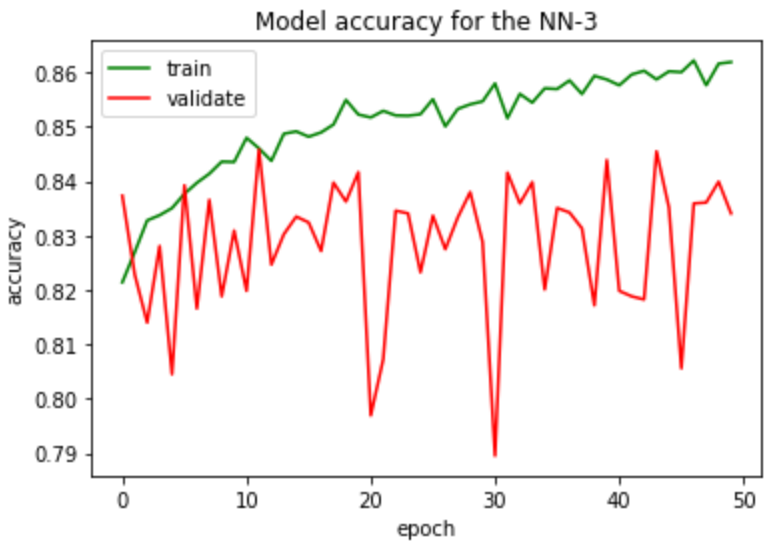
Multi-layer Neural Network

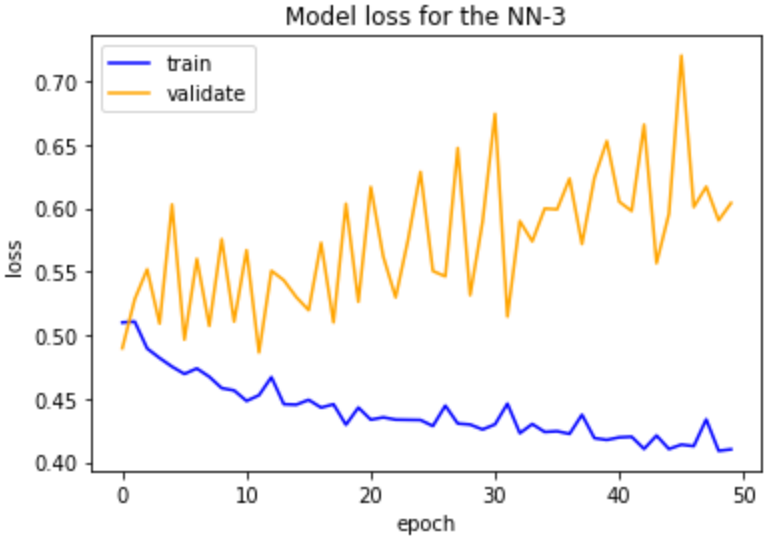
3 Hidden layer
5 epochs

Confusion Matrix



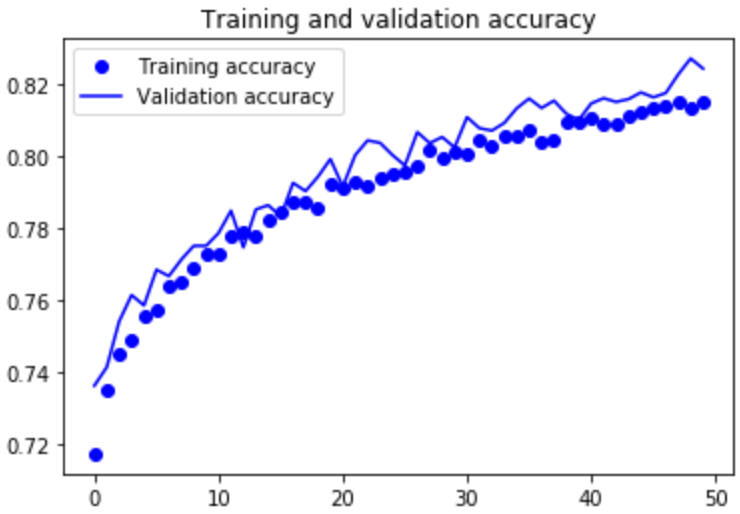
3 Hidden Layer with 50 epochs

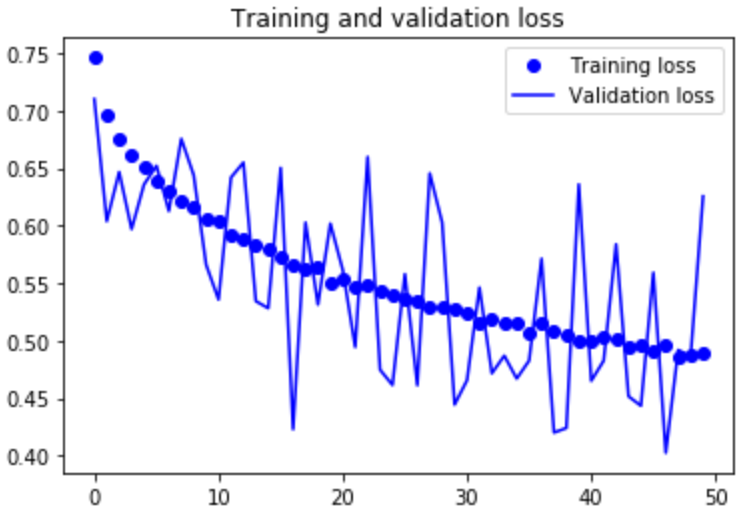




Convolutional Neural Network
With 1 Convolutional Layer

50 epochs





Confusion Matrix

Conclusion

3 Different way of using neural network is introduce in this report, using the Fashion-MNIST dataset

Reference:

https://medium.com/datadriveninvestor/implementing-convolutional-neural-network-using-tensorflow-for-fashion-mnist-caa99e423371

https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c

https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8

https://towardsdatascience.com/the-4-convolutional-neural-network-models-that-can-classify-your-fashion-images-9fe7f3e5399d