Introduction To Neural Network Class Classification

Casper Hsiao

2020-03-12

Abstract: The objective of this project is to implement artificial neural networks to classify the images of fashion items in terms of its category. A total of $60,000$ data samples were used for the model training and validation, and $10,000$ data samples used test the optimal model. Two types of network architectures, fully-connected and convolutional neural networks, were experimented to determine the optimal hyperparameters for highest prediction accuracy. The validation and test accuracy of the fully-connected neural network were 88.0% and 86.1%. The convolutional neural network performed better with a validation and test accuracy of 90.0% and 89.5%.

I. Introduction and Overview

Artificial neural network is a type of machine learning tool, which was inspired by the biological neural networks. The artificial neural network consists of input and output layers with hidden layers that transforms the input to output. Two different neural network, fully-connected and convolutional, were utilized to classify the images of fashion items. There are up to 10 possible categories, and each fashion item corresponds to one category. The data were divided into train, validation and test data where the number of data samples were $55,000$, $5,000$ and $10,000$ respectively. For each neural network model, different hyperparameters were tested to eventually determine the optimal hyperparameters for highest prediction accuracy. The variable hyperparameters were the number of layers, width of the layers, initial learning rate, $L2$ regularization, type of filter and activation function. The confusion matrix of the results were plotted to determine the acuuracy of the train, validation and test data.

II. Theoretical Background

The neural network was implemented as a supervised learning classification algorithm. The two different network architecture experimented were fully-connected and convolutional neural networks. The major difference between these two networks is their composition of hidden layers. Furthermore, the number of hidden layers determines the depth of an network. Therefore, neural networks with multiple hidden layers are often called “deep” network.

II.A. Fully-Connected Neural Network

The fully-connected neural network consists of a series of fully-connected layers, where each layer is a function from $\mathbb{R}^m$ to $\mathbb{R}^n$. Each layer generally consists of multiple neurons with weighted connections and a bias neuron. Eq. 1 showed the equation of the input and output of each layer, where y is the output of the layer with the neuron inputs x and weighted connections w. Furthermore, b is the value of the bias neuron and σ is the activation function. The number of neurons in a layer determined the width of the network. The width of the layers and type of activation are hyperparameters of the network.

$$y = \sigma(wx + b) \tag{1}$$

II.B. Convolutional Neural Network

The convolutional neural network is a class of deep learning neural networks that is commonly utilized for image classification. The convolution layers are able to assign weights to various aspects in the image, for instance an area of pixels. Therefore, differentiate each parts from the others. This unique modelling algorithm is inspired by the organization of visual cortex, where individual neurons responds to its receptive fields. Furthermore, the overlap information provides high-level features of the image such as gradients. The output of the convolution layer is determined by the kernel, stride length and padding.

III. Algorithm Implementation and Development

The data $fashion\_mnist.mat$ was loaded into the Matlab workspace. The first $5,000$ data samples of the train data were used to create the validation data set, which leaves $55,000$ data samples for training. Furthermore, $10,000$ were available to utilize as the test data for the final optimal model. Each input data X were provided with its corresponding output data y for the supervised learning. The data were processed and reshaped to the format required by the Matlab deep learning package.

The neural networks was built using the function $net(X\_train,\ y\_train,\ layers,\ options)$. The parameter X_train and y_train are the inputs and outputs of the training data. The $layers$ parameter is the network architecture of the input, hidden and output layers. Lastly, the *options* includes the hyperparameters of the network. Different $layers$ and *options* were tested for both fully-connected and convolutional networks, which includes the depth and width of the network, type of layer, activation function, initial learning rate, $L2$ regularization

and maximum epochs. After the optimal network architecture is determined, the confusion matrix of the train, validation and test data were plotted. The width of the layers varied from 700 to 10, and the depth of the network varied from 3 to 13. The $L2$ regularization value and initial learning rate varied from 10^{-3} to 10^{-4}. Furthermore, the activation function tested were $reluLayer$, $leakyReluLayer$, $tanhLayer$ and $softmaxLayer$.

IV. Computational Results

The fully-connected neural network model provided a relatively well accuracy. The training progress of the network is shown in Fig. 1. The hidden layers consists of two $fullyConnectedLayer$ with 500 and 250 neurons respectively.The activation functions of the layers was $reluLayer$. Furthermore, the maximum epochs is 5, initial learning rate is $7 * 10^{-3}$ and $L2$ regularization of $7 * 10^{-4}$. The validation and train accuracy were close to each other, which represents the non-overfitting feature of the model. Furthermore, the loss function also followed the trend of the accuracy where the train and validation values were close.

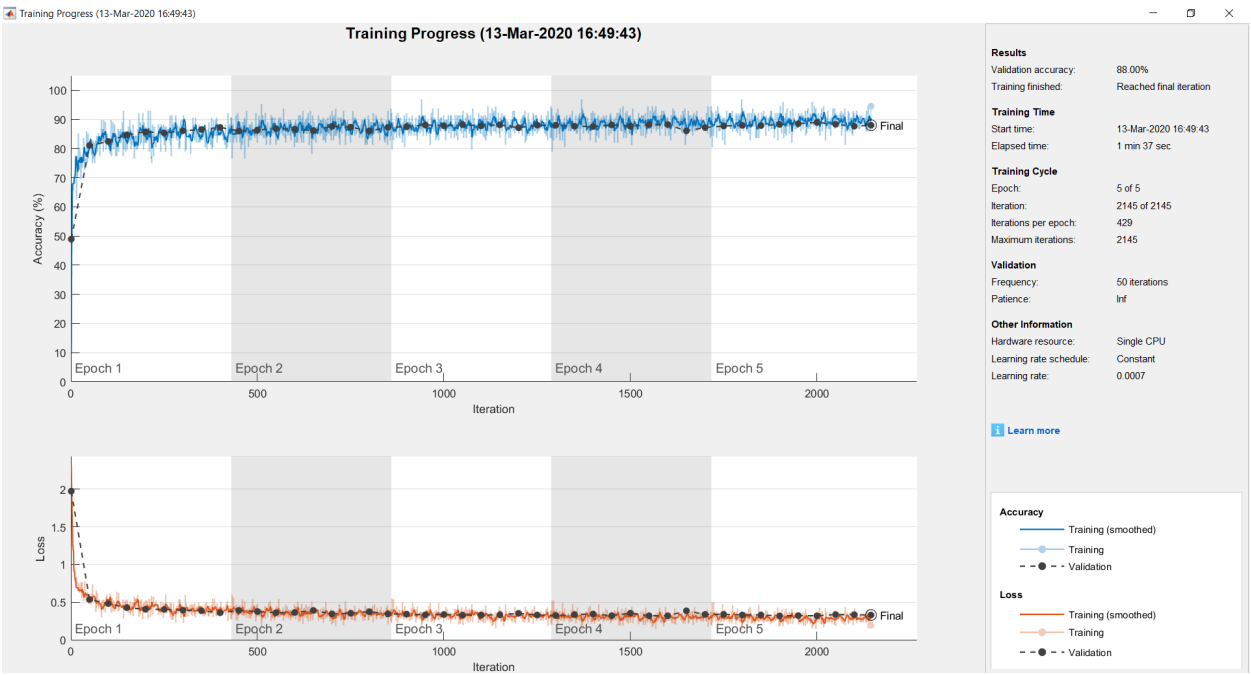
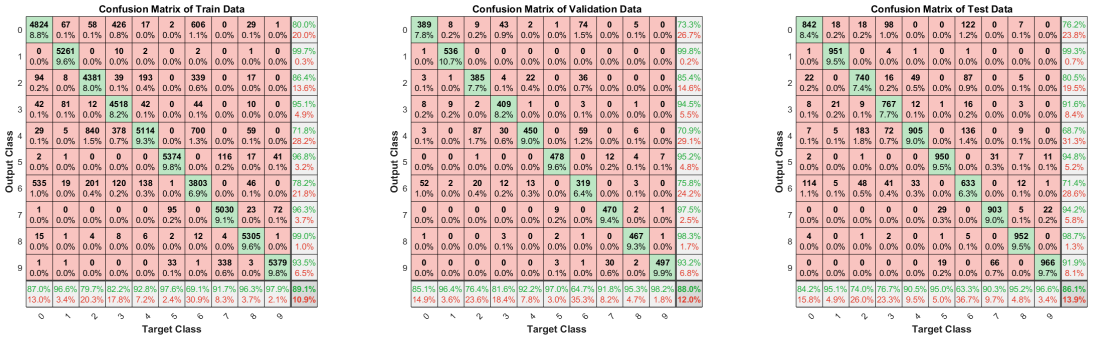


Fig. 1: The training progress of the fully-connected neural network.

As shown in Fig. 4, the accuracy of train, validation and test data were 89.1%, 88.0% and 86.1% respectively. The lowest accuracy of the model happens at category 6 where only 6.9% of the images were classified accurately. Furthermore, category 9 has up to 9.8% of classification accuracy, which is the highest amongst the categories.



(a) The confusion matrix of the train data. **(b)** The confusion matrix of the validation data. **(c)** The confusion matrix of the test data.

Fig. 2: The confusion matrices of the fully-connected neural network.

The convolutional neural network model provided a better accuracy of the predictions. The training progress of the network is shown in Fig. 3. The hidden layers consists of three *convolution2dLayer* and *tanhLayer*, one *averagePooling2dLayer*, and one *fullyConnectedLayer* and *leakyReluLayer* . Furthermore, the maximum epochs is 5, initial learning rate is 10^{-3} and $L2$ regularization of 10^{-4}. The validation and train accuracy were close to each other, which represents the non-overfitting feature of the model. Furthermore, the loss function also followed the trend of the accuracy where the train and validation values were close.

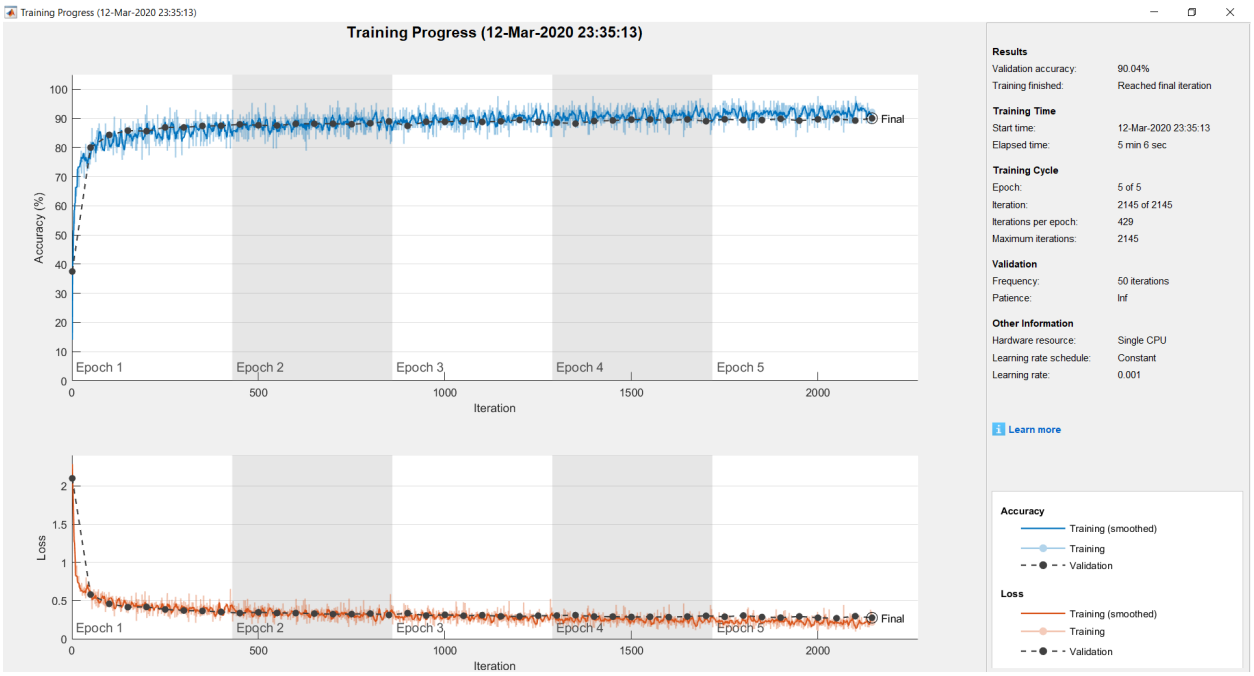
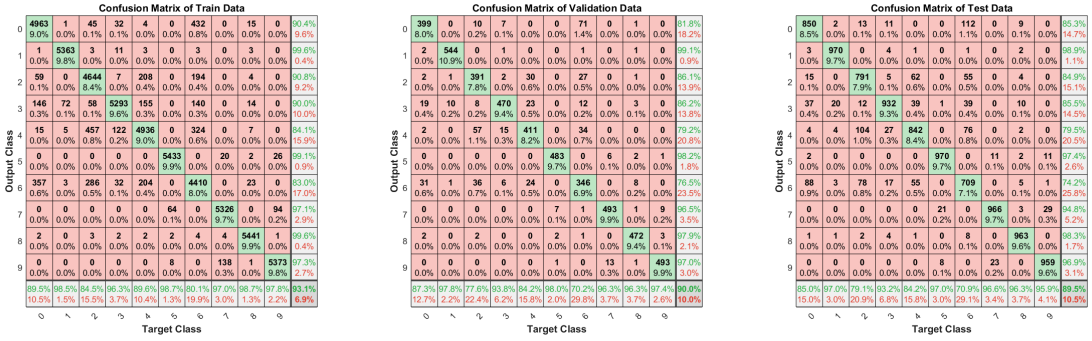


Fig. 3: The training progress of the convolutional neural network.

As shown in Fig. 4, the accuracy of train, validation and test data were 93.1%, 90.0% and 89.5% respectively. The lowest accuracy of the model occurs at category 6, which is same as the previous model, where only 8.0% of the images were classified accurately. This showed that the image of category 6 is particularly difficult to classify systematically. Furthermore, category 5 and 8 has up to 9.9% of classification accuracy, which are the highest amongst the categories. These accuracies are relatively higher than the results of the fully-connected network as the convolutional network is able of capturing high-level features of the images.



(a) The confusion matrix of the train data. **(b)** The confusion matrix of the validation data. **(c)** The confusion matrix of the test data.

Fig. 4: The confusion matrices of the convolutional neural network.

V. Summary and Conclusions

The implementation of artificial neural network on supervised image classification demonstrate the ability to classify the images into multiple classes. The fully-connected network provided a fair accuracy, 86.1%, of the test data classification. The covolutional network provided a better test data accuracy of 89.5%. This verified the ability of convolution layers in extracting high-dimensional featured of the images, which enhances the distinction between images. Both network were able to avoid overfitting in test data, which is verified by the close accuracy of test and validation data. The artificial neural network performed a well classification ability of multiple categories with the supply of sufficient data samples. Therefore, it is a extremely useful tool for classification in current data-driven globe.

Appendix A

$im2double(x)$: converts the image integer units of x to double units.
$reshape(x, sz)$: reshape matrix X to the specified size sz.
$permute(x, sq)$: rearranges the sequence of dimensions of x with sq.
$imageInputLayer(sz)$: computes a image input of dimensions sz.
$fullyConnectedLayer(n)$: computes a fully connected layer with n neurons.
$reluLayer$: computes a relu activation function layer.
$convolution2dLayer(sz, n)$: computes a convolutional 2d layer of kernel size sz and number of layers n.
$tahnLayer$: computes a $tanh$ activation function layer.
$leakyReluLayer$: computes a leaky relu activation function layer.
$softmaxLayer$: computes a softmax function layer.
$classificationLayer$: computes a classification layer.
$trainingOptions(...)$: set up the training options, which includes the hyperparameters, of the network.
$net(x, y, layers, options)$: train a network using x and y data with layers of the network, $layers$, and traning options, $options$.
$classify(net, x)$: computes the predictions of data x with network net.
$plotconfusion(y, y\_pred)$: plots the confusion matrix of data y and prediction y_pred.

Appendix B

Contents

- Load and process data
- Fully-connected neural network
- Plot confusion matrix of train and validation data
- Plot confusion matrix of test data
- Convolutional neural network
- Plot confusion matrix of train and validation data
- Plot confusion matrix of test data


```
clc; clear all; close all;
```

Load and process data

```
load('fashion_mnist');

X_valid = im2double(X_train(1:5000, :, :));
X_train = im2double(X_train(5001:end, :, :));
X_test = im2double(X_test);

X_valid = reshape(X_valid, [5000 28 28 1]);
X_train = reshape(X_train, [55000 28 28 1]);
X_test = reshape(X_test, [10000 28 28 1]);

X_valid = permute(X_valid, [2 3 4 1]);
X_train = permute(X_train, [2 3 4 1]);
X_test = permute(X_test, [2 3 4 1]);

y_valid = categorical(y_train(:, 1:5000))';
y_train = categorical(y_train(:, 5001:end))';
y_test = categorical(y_test)';
```

Fully-connected neural network

```
layers = [imageInputLayer([28, 28, 1])
    fullyConnectedLayer(500)
    reluLayer
    fullyConnectedLayer(250)
    reluLayer
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];

options = trainingOptions('adam',...
    'MaxEpochs', 5,...
    'InitialLearnRate', 7e-4,...
```

```
    'L2Regularization', 7e-4,...
    'ValidationData', {X_valid, y_valid},...
    'Verbose', false,...
    'Plots', 'training-progress');

net = trainNetwork(X_train, y_train, layers, options);
```

Plot confusion matrix of train and validation data

```
figure(1)
y_pred = classify(net, X_train);
plotconfusion(y_train, y_pred);
title('Confusion Matrix of Train Data');

figure(2)
y_pred = classify(net, X_valid);
plotconfusion(y_valid, y_pred);
title('Confusion Matrix of Validation Data');
```

Plot confusion matrix of test data

```
figure(3)
y_pred = classify(net, X_test);
plotconfusion(y_test, y_pred);
title('Confusion Matrix of Test Data');
```

Convolutional neural network

```
layers1 = [imageInputLayer([28, 28, 1])
    convolution2dLayer([5 5], 6, 'Padding', 'same')
    tanhLayer
    convolution2dLayer([5 5], 6, 'Padding', 'same')
    tanhLayer
    averagePooling2dLayer([2 2], 'Padding', 'same', 'Stride', [2 2])
    convolution2dLayer([5 5], 6, 'Padding', 'same')
    tanhLayer
    fullyConnectedLayer(250)
    leakyReluLayer
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];

% layers1 = [imageInputLayer([28, 28, 1])
%     convolution2dLayer([28 28], 3, 'Padding', 'same')
%      ([14 14], 3, 'Padding', 'same', 'Stride', [2 2])
%     tanhLayer
%     fullyConnectedLayer(20)
%     reluLayer
%     fullyConnectedLayer(10)
```

```
%     softmaxLayer
%     classificationLayer];

%     convolution2dLayer([5 5], 6, 'Padding', 'same')
%     tanhLayer
%     averagePooling2dLayer([2 2], 'Padding', 'same', 'Stride', [2 2])
%     convolution2dLayer([5 5], 6, 'Padding', 'same')
%     tanhLayer

options1 = trainingOptions('adam',...
    'MaxEpochs', 5,...
    'InitialLearnRate', 1e-3,...
    'L2Regularization', 1e-4,...
    'ValidationData', {X_valid, y_valid},...
    'Verbose', false,...
    'Plots', 'training-progress');

net1 = trainNetwork(X_train, y_train, layers1, options1);
```

Plot confusion matrix of train and validation data

```
figure(4)
y_pred1 = classify(net1, X_train);
plotconfusion(y_train, y_pred1);
title('Confusion Matrix of Train Data');

figure(5)
y_pred1 = classify(net1, X_valid);
plotconfusion(y_valid, y_pred1);
title('Confusion Matrix of Validation Data');
```

Plot confusion matrix of test data

```
figure(6)
y_pred1 = classify(net1, X_test);
plotconfusion(y_test, y_pred1);
title('Confusion Matrix of Test Data');
```

Published with MATLAB® R2019b