Object Classification using Convolution Neural Networks

Mohan Burugupalli

Adviser: Dr. Simone Ludwig

Computer Science Department

North Dakota State University

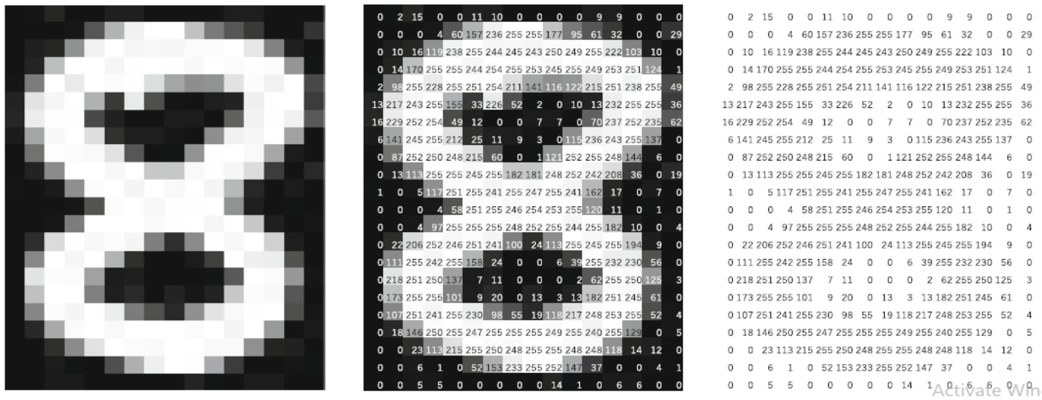
mohan.burugupalli@ndsu.edu

Introduction

Image classification is of increasing importance recently. Prediction of outcome from the given sample data is very complex process as it needs large amount of data to be trained. To overcome this, in this project I have used multi complex sample data that has various types of unique data collection. Firstly, accurate data collection is the primary goal. For that I have used the 'keras' library datasets namely Fashion-MNIST and CIFAR10 using Python programming. The datasets are loaded as train data and test data. The network is trained using the train data and then validated using test data. This helps in retrieving and analyzing the test data using various graph and plots as and when required. Also, this helps to better classify and visualize the image recognition. As the sampling size include multi data that has around 60,000 images in Fashion-MNIST and 50,000 images in CIFAR10 datasets respectively, we can predict much closer to the real occurrence.

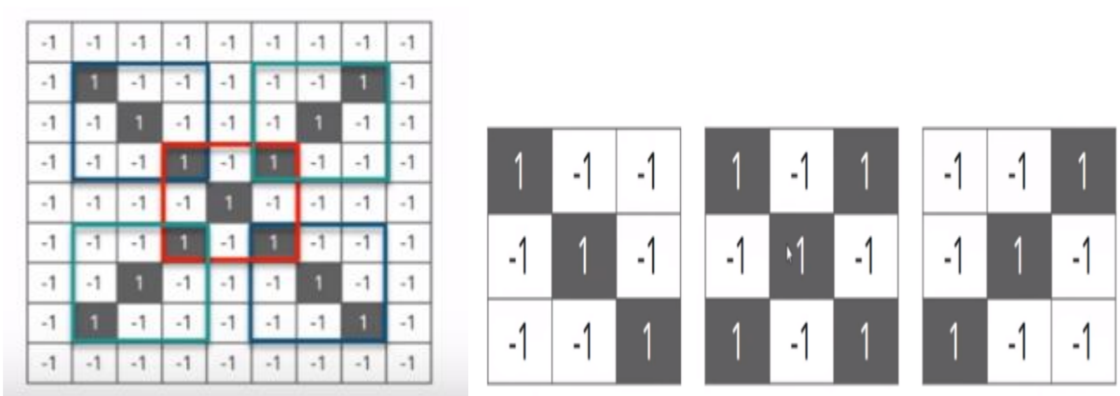
Background

Convolution Neural Networks is a concept that has multiple layers for training the data and when a test data is fed to it, the image is classified. Below is the image how computer sees an image.

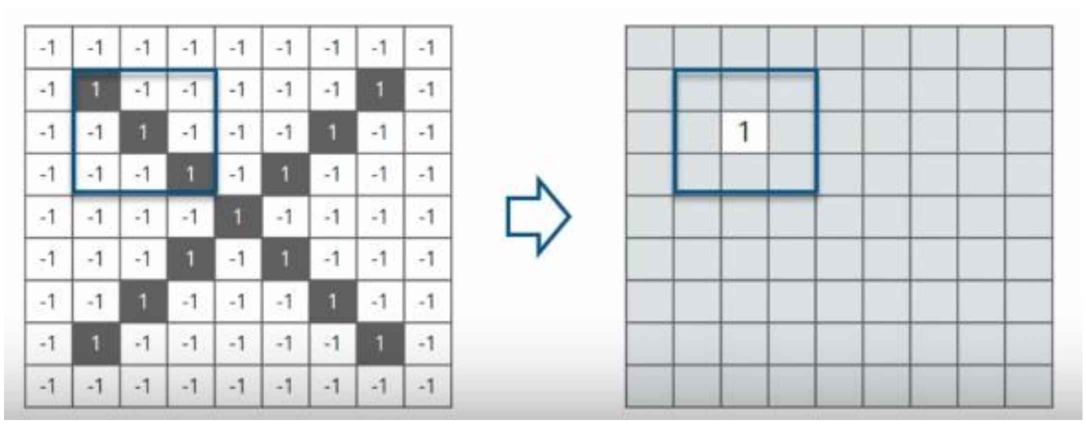


Convolution Neural Networks has the layers as Convolution layer, ReLU activation layer, Pooling layer and Fully connected layer.

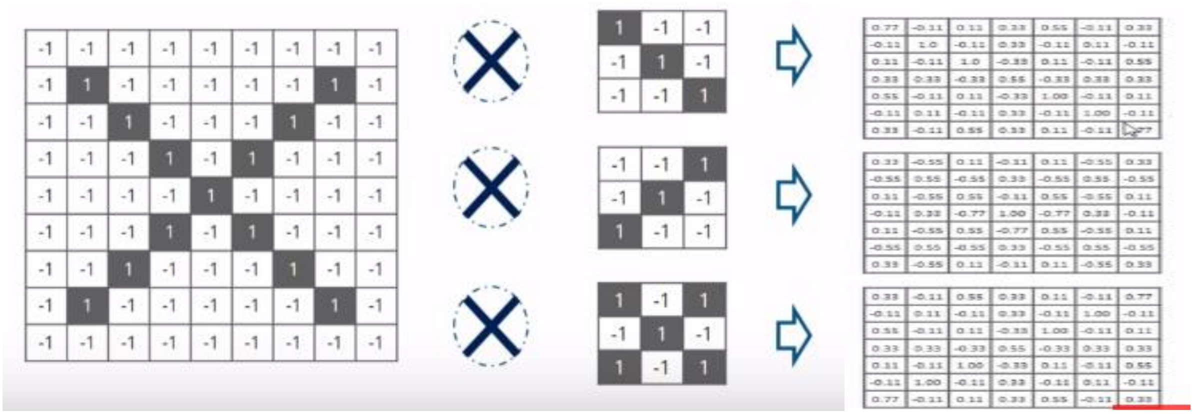
Smallest unique features of the input image are taken for training. Below is an example where small pieces/features of bigger image are taken for convolution.



Move each feature on the big image and perform matrix calculation of the values in the feature and the position places in big image. Average all the values in the resulting 3*3 matrix and the result is taken as the final convolution value of the image for the given feature



The result of all three features convoluted over the given image is as below



Then pass through ReLU activation layer and max pooling layer. The resulting values are as below

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	1.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

After performing one more instance of convolution, ReLU and max pooling on the above result we get the numerical notation of the input image as below.

For testing, we will generate the numerical notation of the image that is being tested and compare its values with the existing values and classify the image accordingly.

Experiments on Datasets

The existing datasets in python 'keras' library namely Fashion-MNIST and CIFAR10 are used for the experiments.

Step-1: Import the necessary libraries

Step-2: Load the '*fashion_mnist*' dataset from the keras library as below. There are 60,000 images in training dataset and 10,000 images in testing dataset with the pixel size of 28*28 respectively.

```
#Load the data from the keras fashion_mnist dataset
# Reference : https://keras.io/datasets/
(x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()
#(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()
```

Step-3: y_train and y_test has the labels from 0 to 9 in which each number denotes a class label from the below class_labels list

```
#Class labels do not come with dataset and hence we code them here
# Reference : https://keras.io/datasets/
class_labels = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
#class_labels = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```

Step-4: All the images are read with the values ranging from 0-255 in each pixel. For better calculations for the convolution we scale the values to the range between 0 and 1.

```
# From the result of above print statement we can see that an image is read with the values ranging from 0-255
# Scaling all testing and training images to range of 0-1
x_train = x_train/255.0
x_test = x_test/255.0
```

Step-5: Sequential model is chosen to perform the calculations layer by layer in sequence. The first layer is Flatten layer that converts the image data from 2-d to 1-d. The second layer converts data to output to 128 nodes using 'relu' activation function and the third layer to 10 nodes using 'softmax' function.

```
# Number of layers in neural network are designed
network = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)), #transforms data from 2-d to 1-d
    keras.layers.Dense(128, activation=tf.nn.relu), #This layer has 128 nodes with relu as activation function
    keras.layers.Dense(10, activation=tf.nn.softmax) #converts all values to within range 0-1
])
```

Step-6: Compile the network using Adamoptimizer that updates the model according to the input fed data and given loss function.

```
# Compiling the network
network.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Step-7: Train the model using the python inbuilt 'fit' method.

```
#Training the network
network.fit(x_train, y_train, epochs=10)
```

Step-8: Select a random image from the testing dataset

```
img1 = randint(0, len(x_test))
img = x_test[img1]
print(img.shape)
```

```
(28, 28)
```

Step-9: Convert the image to three dimension before feeding to predict function as it takes three dimensional inputs only.

```
#Reference https://docs.scipy.org/doc/numpy/reference/generated/numpy.expand_dims.html
img = (np.expand_dims(img,axis=0))
print(img.shape)
```

Step-10: Fed the expanded image to predict function.

```
pred1 = network.predict(img)
pred2 = pred1[0]
print(pred2)
```

```
[3.21443717e-04 2.36775214e-03 4.66466248e-02 3.75291129e-04
 9.12996590e-01 4.43485249e-10 3.72055322e-02 3.39686721e-12
 8.68199786e-05 1.15545976e-11]
```

Step-11: Check the maximum value in the above result.

```
ma = np.argmax(pred1[0])
print(ma)
```

```
4
```

Step-12: Verify that all probabilities of the predicted result sum approximately to 1.

```python
##Verification to check whether all probabilities sum to 1
val1 = 0
for val in pred2:
    val1 = val1 + val
print("Sum of all predictions = "+str(val1))
```

```
Sum of all predictions = 1.000000059030556
```

Step-13: Calculate accuracy and print the output.

```python
# Output
na = y_test[img1]
print("Testing image fed to the network is: "+class_labels[na])
print("Image predicted is: "+class_labels[ma])
print("Accuracy of prediction is: "+str(accuracy)+"%")
```

```
Testing image fed to the network is: Coat
Image predicted is: Coat
Accuracy of prediction is: 91.29965901374817%
```

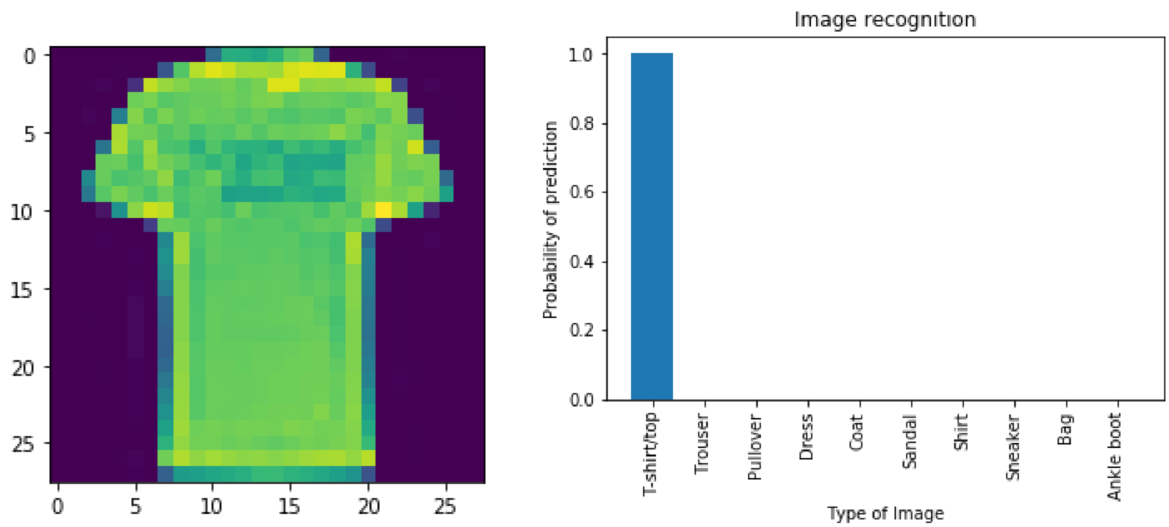
Step-14: Repeat the above steps for CIFAR10 dataset with the below changes as the data pixels and dimensions differ

```python
# Number of layers in neural network are designed
network = keras.Sequential([
    keras.layers.Flatten(input_shape=(32, 32,3)), #transforms data from 2-d to 1-d
    keras.layers.Dense(128, activation=tf.nn.relu), #This layer has 128 nodes with relu as activation function
    keras.layers.Dense(10, activation=tf.nn.softmax) #converts all values to within range 0-1
])
```

Results

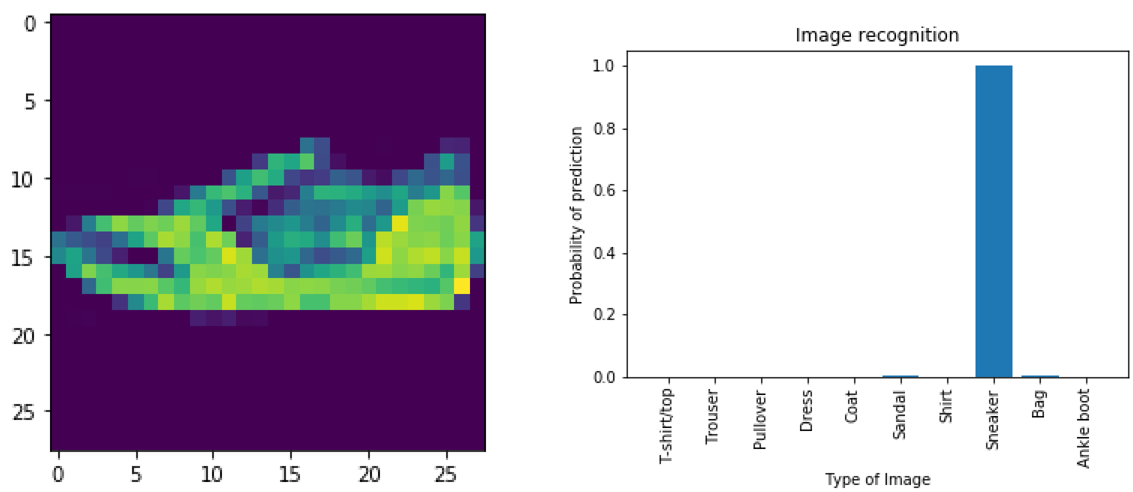
Fashion-MNIST Dataset

1)



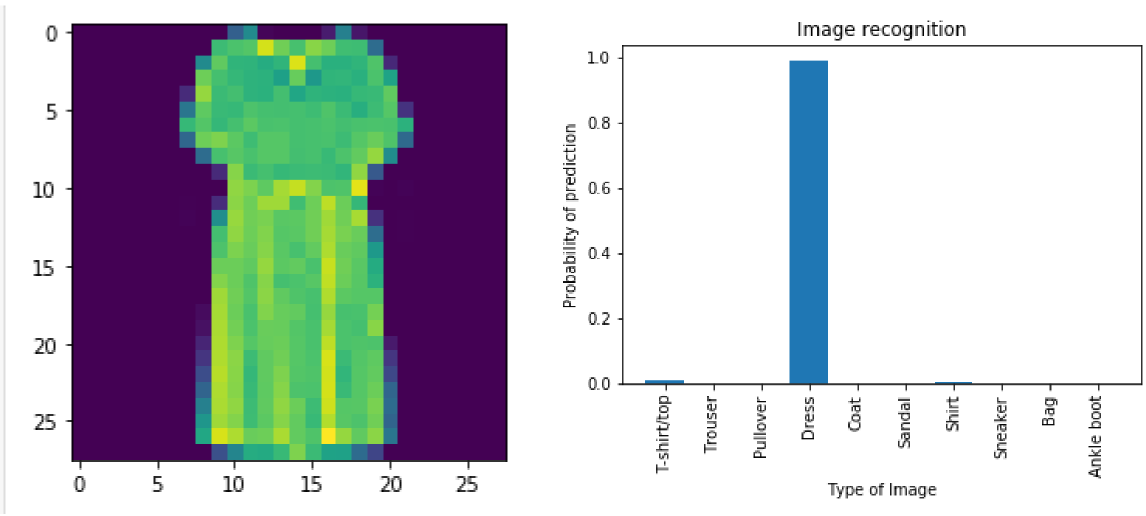

```
Testing image fed to the network is: T-shirt/top
Image predicted is: T-shirt/top
Accuracy of prediction is: 99.9794065952301%
```

2)



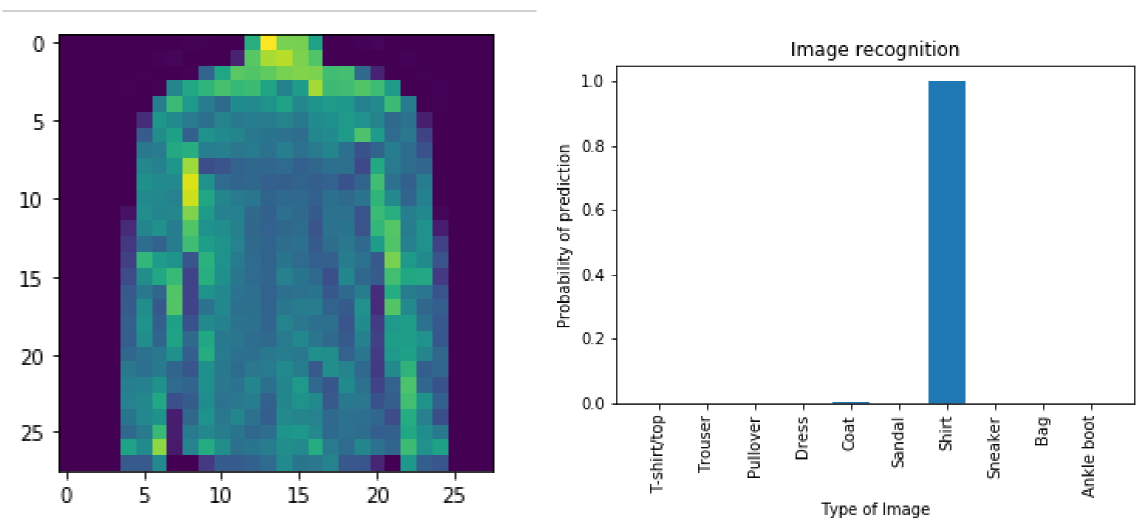

```
Testing image fed to the network is: Sneaker
Image predicted is: Sneaker
Accuracy of prediction is: 99.91387128829956%
```

3)



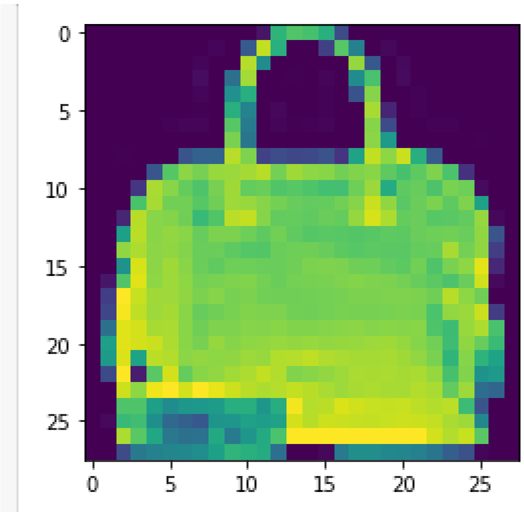

```
Testing image fed to the network is: Dress
Image predicted is: Dress
Accuracy of prediction is: 98.88255000114441%
```

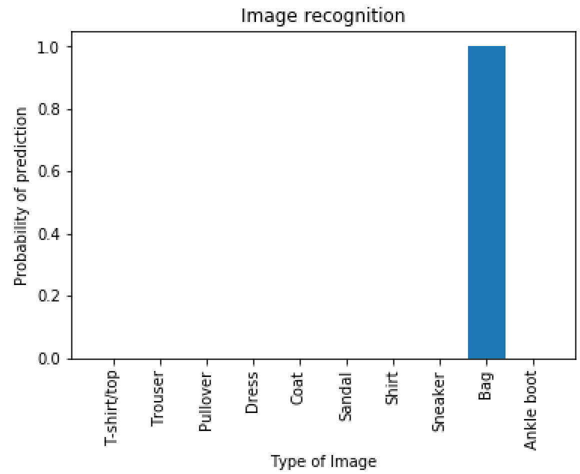
4)




```
Testing image fed to the network is: Shirt
Image predicted is: Shirt
Accuracy of prediction is: 99.86782670021057%
```

5)

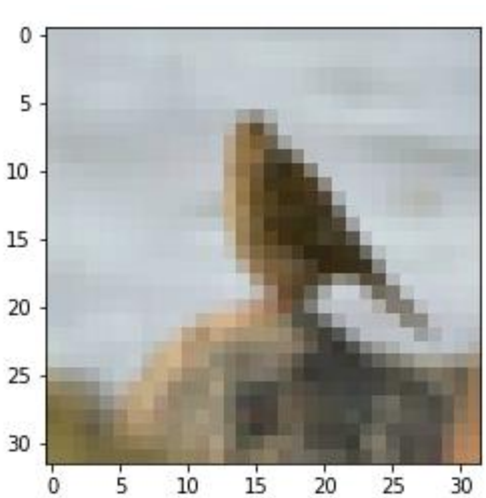


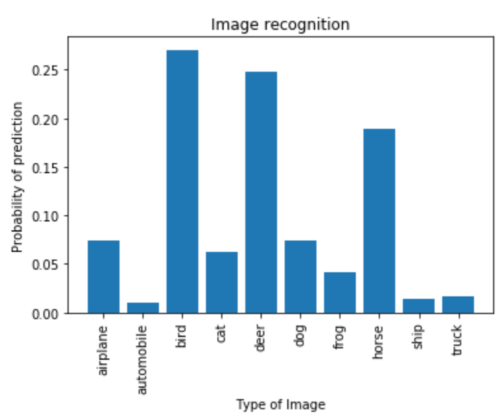



```
Testing image fed to the network is: Bag
Image predicted is: Bag
Accuracy of prediction is: 99.99918937683105%
```

CIFAR10 Dataset

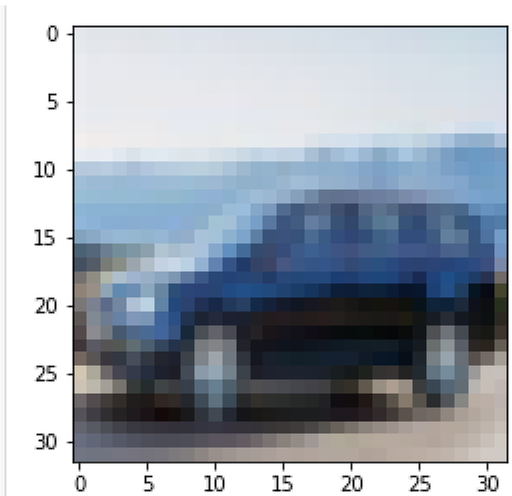
1) Due to less picture quality some images identification accuracy is low as below.

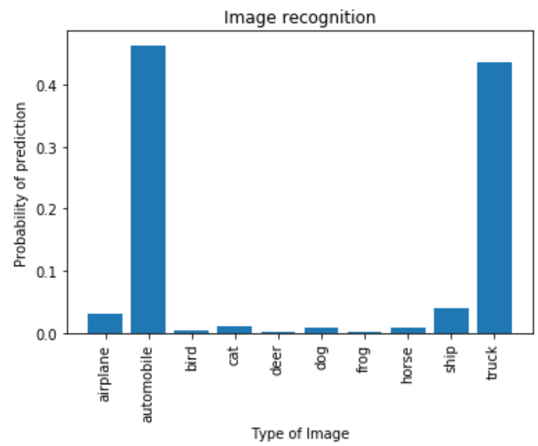





```
Testing image fed to the network is: bird
Image predicted is: bird
Accuracy of prediction is: 27.054086327552795%
```

2)






```
Testing image fed to the network is: automobile
Image predicted is: automobile
Accuracy of prediction is: 46.35330140590668%
```

Conclusion

Two types of datasets namely Fashion-MNIST and CIFAR10 are taken from the keras library in python. The datasets are split into training data and testing data. Using the training data convolution neural network is trained to classify similar images. Once the model is ready, testing data is fed and the results are captured. Fashion-MNIST dataset gave accuracy near to 99% in most cases while CIFAR10 dataset accuracy is quite less due to unclear images.

Future Work for next Semester

As discussed, I shall focus on 'Transfer Learning' that uses existing Convolution Neural networks such as AlexNet, GoogLeNet etc. to classify images. Most probably medical datasets are recommended for image classification.

References

1. https://jovianlin.io/keras-models-sequential-vs-functional/
2. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Flatten
3. https://docs.w3cub.com/tensorflow~python/tf/keras/layers/flatten/
4. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

5. https://keras.io/layers/core/
6. https://keras.io/datasets/
7. https://elitedatascience.com/keras-tutorial-deep-learning-in-python
8. https://www.edureka.co/ai-deep-learning-with-tensorflow?qId=18117901ec9b65fd72d2577b0910cf84&index_name=prod_search_results_courses&objId=544&objPos=1
9. https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
10. https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/
11. https://pythonprogramming.net/convolutional-neural-network-cnn-machine-learning-tutorial/
12. https://campus.datacamp.com/courses/convolutional-neural-networks-for-image-processing/image-processing-with-neural-networks?ex=6
13. https://www.coursera.org/learn/neural-networks-deep-learning?specialization=deep-learning
14. https://pythonprogramming.net/convolutional-neural-network-deep-learning-python-tensorflow-keras/
15. https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-fashion-mnist-clothing-classification/