International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 12 | December - 2022 Impact Factor: 7.185 ISSN: 2582-3930

BASIC IMAGE CLASSIFICATION WITH TENSOR FLOW

M. Yamini, K. Jemima Joy, T. Shiva Kumar

Under the Guidance of

Mr. V. Satheesh Kumar

Sreenidhi Institute of Science & Technology

ABSTRACT:

Basic image classification with TensorFlow aims to classify simple hand written numerical images. Using Artificial neural network, the model learns how to classify the images into 9 different classes that represent digits from 0-9. Implementation of our model yields an accuracy of 96% which is highly accurate than most of the existing models that deal with basic image classification.

Introduction:

Image Classification uses Machine Learning algorithms to analyse the presence of items in a picture and to categorize the picture. This task forms the basis of Computer Vision and Image Recognition. Machines do not analyse a picture as a whole. They only analyse a picture through pixel patterns or vectors.

Image classification is to identify and portray, as a unique gray level, it is the process of categorizing and labelling groups of pixels or vectors within an image based on specific rules. The

categorization law can be devised using one or more textural characteristics.

We will be using Keras which is a high-level python neural network API. Keras is tightly integrated with TensorFlow, a symbolic math library which is widely used for machine learning and neural network tasks. In our task we will be using mnist dataset which is preloaded with 60,000 hand written numerical images with each image dimension being 28*28 pixels.

Keras is a neural network Application Programming Interface. A neural network has different layers. The first layer is input layer, it picks up the input signals and passes them to the next layer. The next layer does calculations and feature extractions, often called as hidden layer, there might be more than one hidden layer. And finally, there is an output layer, which delivers the result.

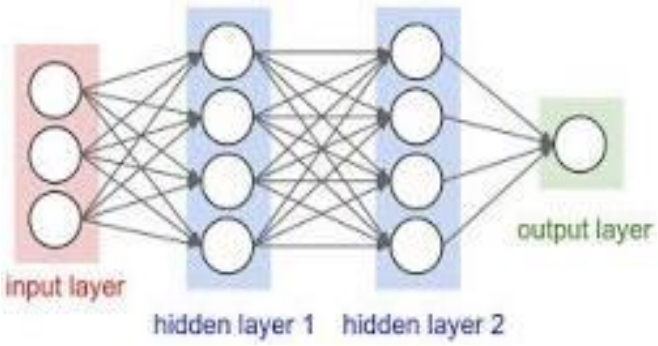


Fig: A simple Neural Network

Methods:

We need to import few important libraries and a model that required to build our model

- tensorflow
- keras
- dense
- sequential
- to_categorical
- mnist
- numpy

We split the mnist data set into train and test data sets. Train data set holds all 60,000 images and test data set holds 10,000 images.

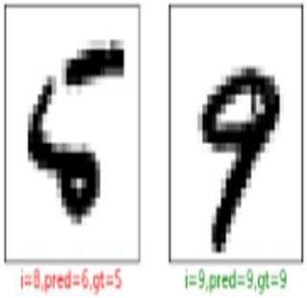
As the images are of two-dimensional size 28*28 pixels. Using NumPy library we store the pixel information in a single dimension array for easy and faster calculation of mean, standard-deviation, and normalization.

We use sequential model where we stack up multiple same or different layers where one's output ahead. Dense layer learns features from all the combinational features of the previous layer, then goes compiling the model, here we can get the model summary, next we train the model and evaluate the model and prepare the test data set to predict results from the newly developed model

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
y_train_encoded = to_categorical(y_train)
y_test_encoded = to_categorical(y_test)
x_train_reshaped = np.reshape(x_train, (60000, 784))
x_test_reshaped = np.reshape(x_test, (10000, 784))
x_mean = np.mean(x_train_reshaped)
x_std = np.std(x_train_reshaped)
epsilon = 1e-10
x_train_norm = (x_train_reshaped - x_mean) / (x_std + epsilon)
x_test_norm = (x_test_reshaped - x_mean) / (x_std + epsilon)
model = Sequential([
    Dense(128, activation='relu', input_shape=(784,)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
model = Sequential([
    Dense(128, activation='relu', input_shape=(784,)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
model.fit(x_train_norm, y_train_encoded, epochs=3)
preds = model.predict(x_test_norm)
plt.figure(figsize=(12, 12))
start_index = 0
for i in range(25):
    plt.subplot(5, 5, i+1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    pred = np.argmax(preds[start_index+i])
    gt = y_test[start_index+i]
    col = 'g'
    if pred != gt:
        col = 'r'
    plt.xlabel('i={},pred={},gt={}'.format(start_index+i, pred, gt), color=col)
    plt.imshow(x_test[start_index+i], cmap='binary')
plt.show()
```

Results:

As the test data is passed to the model, we are displaying a range of 25 sample outputs in the code above and showing a right and wrong prediction below



SOFTWARE AND HARDWARE REQUIREMENTS:

The TensorFlow has several requirements such as 64-bit Linux, Python 2.7 (or 3.3+ for Python 3), NVIDIA CUDA 7.5 (CUDA 8.0 required for Pascal GPUs) and NVIDIA, cuDNN v4. 0 (minimum) or v5. 1 (recommended).

CONCLUSION:

The objectives are linked directly with conclusions because it can determine whether all objectives are successfully achieved or not. It can be concluded that all results that have been obtained, showed quite impressive outcomes. The Artificial neural network (ANN) becomes the main agenda for this research, especially in image classification technology. ANN technique was studied in more details starting from assembling, training model and to classify images into categories. Implementation of sequential learning by using framework TensorFlow also gave good results as it can simulate, train and classified with up to 96% percent of accuracy. Lastly, Python have been used as the programming language throughout this research since it comes together with framework TensorFlow. Which leads to designing of the system involved Python from start until ends.

ACKNOWLEDGMENTS:

We would like to express our special thanks to our mentor

Mr. V. Satheesh Kumar who gave us a golden opportunity to do this project, which also helped us in doing a lot of research and knowing about various topics. We are thankful to them. Secondly, we would also like to thank my

co-authors who helped a lot in finalizing this project within the limited time frame.