Image classification using Convolution Neural Network on CIFAR-10 data set

Introduction

Image classification is the task of categorizing and assigning labels to groups of pixels or vectors within an image dependent on particular rules. The categorization law can be applied through one or multiple spectral or textural characterizations.

There are many techniques for image classification in machine learning and artificial intelligence. CNN (convolution neural network) has the demonstrated excellent method in image classification. Convolutional Neural Network (CNN) is a special type of multi-layer neural network inspired by the mechanism of the optical and neural systems of humans.

Aim & Objective

The aim is to use machine learning approaches to solve a computer vision task (Image Classification). Using the CIFAR-10 dataset, develop a deep neural network (Convolutional Neural Network) for the image classification problem. Then compare the CNN model performance with predefine model (we are using ResNet50) within keras.

Methodology

1. Choice of method

We have created the CNN architecture to build the model. CNN architecture in neural network is shown in Fig 1. The convolution neural network has 2,915,114 trainable parameters. CNN uses the dropout method to reduce the overfitting problem. The architecture contains thirteen layers, including six convolutional, three pooling, and two fully-connected layers with softmax classifier. The output layer provides a 10-way softmax, which recognizes 10 different class scores.

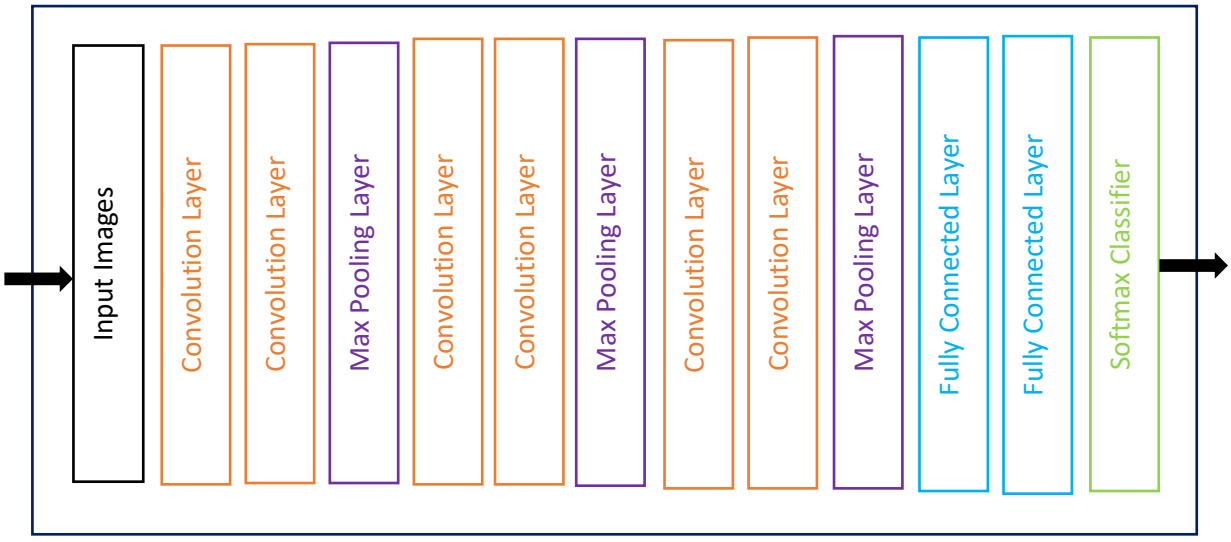


Figure 1

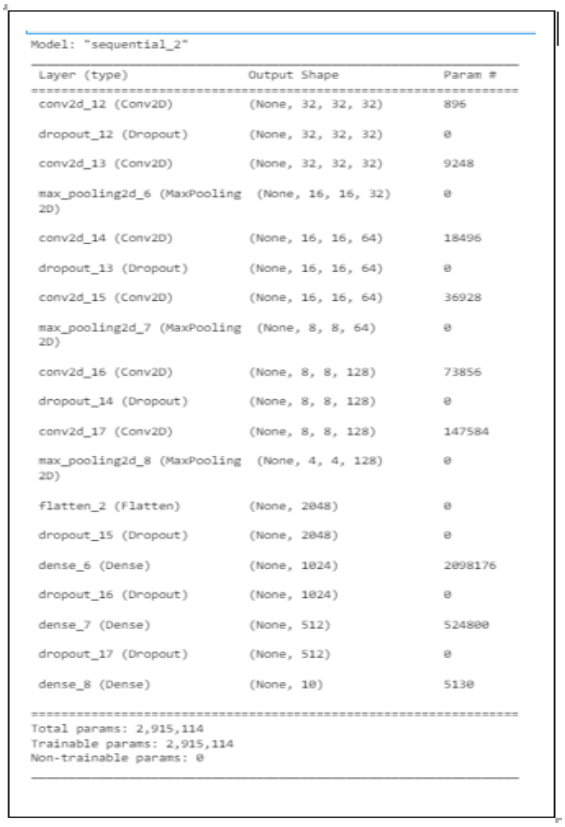
There are 32, is the number of filters needed. A filter is an array of numeric values. (3,3) is the size of the filter, which means 3 rows and 3 columns. The input image is 32*32*3 size, that is, 32 height, 32 widths, and 3 refer to RGB values. Each of the numbers in this array (32,32,3) is given values from 0 to 255, which describes the pixel intensity at that point. The output of this layer will be some feature maps. A feature map is a map that shows some specific features of the image.

We use the Dropout layer in our model to prevent overfitting. Overfitting is a modelling error that occurs to make an overly complex model. This layer drops out a random set of activations in that layer by setting them to zero as data flows through it.

MaxPooling layer is used for pooling. Pooling reduces the dimensionality of each feature map but retains the most important information. This helps to decrease the computational complexity of our network. Flatten is used to convert the feature map to 1-dimension. We use the Dense function to initialize a fully connected network.

An activation function of a neuron defines the output of that neuron, given some input. This output is then used as input for the next neuron and so on until the desired solution is obtained. We use ReLu and softmax activation functions in this model. ReLU replaces all the negative pixel values in the feature map with 0. Softmax takes as input a vector of K real numbers and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers.

This is how our CNN model looks like.



Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 32, 32, 32)	896
dropout_12 (Dropout)	(None, 32, 32, 32)	0
conv2d_13 (Conv2D)	(None, 32, 32, 32)	9248
max_pooling2d_6 (MaxPooling 2D)	(None, 16, 16, 32)	0
conv2d_14 (Conv2D)	(None, 16, 16, 64)	18496
dropout_13 (Dropout)	(None, 16, 16, 64)	0
conv2d_15 (Conv2D)	(None, 16, 16, 64)	36928
max_pooling2d_7 (MaxPooling 2D)	(None, 8, 8, 64)	0
conv2d_16 (Conv2D)	(None, 8, 8, 128)	73856
dropout_14 (Dropout)	(None, 8, 8, 128)	0
conv2d_17 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_8 (MaxPooling 2D)	(None, 4, 4, 128)	0
flatten_2 (Flatten)	(None, 2048)	0
dropout_15 (Dropout)	(None, 2048)	0
dense_6 (Dense)	(None, 1024)	2098176
dropout_16 (Dropout)	(None, 1024)	0
dense_7 (Dense)	(None, 512)	524800
dropout_17 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 10)	5130

Total params: 2,915,114
Trainable params: 2,915,114
Non-trainable params: 0

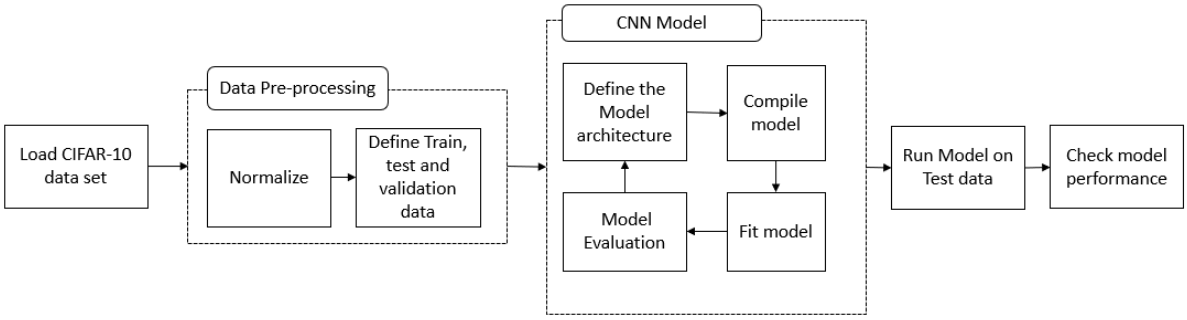
2. Data collection and load

We are provided the CIFAR-10 data set. Cifar-10 is a standard computer vision dataset used for image recognition. It is a subset of the 80 million tiny images dataset and consists of 60,000 32×32 colour images containing one of 10 object classes, with 6000 images per class. There are 50000 training images and 10000 test images. Data has been loaded using keras library as follow:

```
from keras.datasets import cifar10
```

3. Process flow

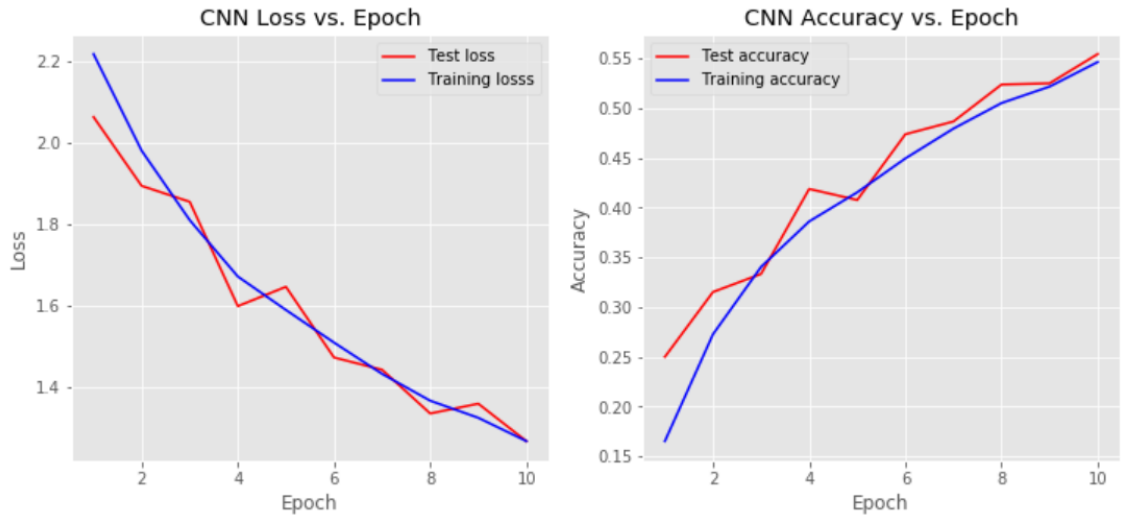
We have built the keras based CNN model for image classification, where first we have loaded the CIFAR-10 data set from keras library then we have normalized the image pixel. After that we have define our training, validation and test data set. Training data has been used to train the CNN model, validation data set has been used to validate the model performance at each epoch and test data set has been used to check the model performance. Below it the process flow diagram to implement the convolution neural network.



CNN model has been fine-tuned with different parameter like optimizer, filter size and learning rate to get best performance of the model. We have used the MaxPool and dropout in the network to reduced the model overfitting. After trained, model has been run on test data and checked that how model performs. Model performance has been checked by accuracy, precision, recall, f1 score, auc (area under curve), roc (receiver operating characteristic) and confusion matrix. Classification report has been generated too to summarized the model performance for each class.

4. Loss & accuracy

Due to system configuration constrained, we have used 10 epochs only with 128 batch size to train the model. Total 16250 images have been used to train model and 8750 images used for validating the performance at each epoch time. Below are two graphs of accuracy and loss learning curve of model at each epoch during the training.



Transfer Learning

Transfer Learning is a machine learning method where we reuse a pre-trained model as the starting point for a model on a new task. To put it simply, a model trained on one task is repurposed on a second, related task as an optimization that allows rapid progress when modelling the second task.

1. Choice of pre-trained model

We have used RestNet50 pre-trained model for image classification on CIFAR-10 data set and compared the model with CNN model.

2. Model implementation

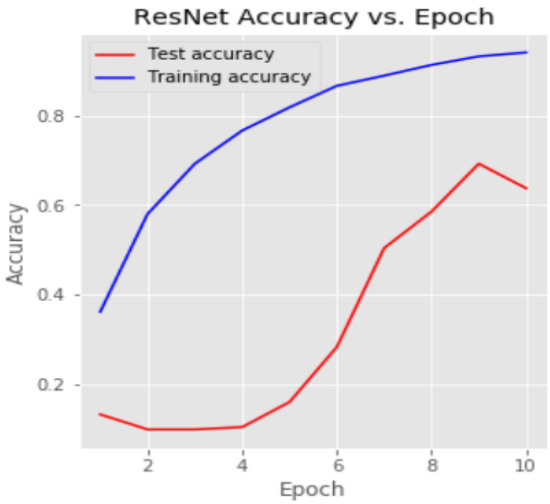
ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. We have used the keras library to load the ResNet pre-trained model and added fully connected layer at the last in the model. The last layer has been added as output layer. The output layer provides a 10-way softmax, which recognizes 10 different class scores. After adding output layer, model has been trained on train data and used the validation for validating the model performance at each epoch time.

This is how our ResNet model looks like after adding fully connected and output layers.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 1, 1, 2048)        23587712

 flatten_3 (Flatten)         (None, 2048)              0

 dense_9 (Dense)             (None, 1024)              2098176

 dropout_18 (Dropout)        (None, 1024)              0

 dense_10 (Dense)            (None, 512)               524800

 dropout_19 (Dropout)        (None, 512)               0

 dense_11 (Dense)            (None, 10)                5130

=================================================================
Total params: 26,215,818
Trainable params: 26,162,698
Non-trainable params: 53,120
_________________________________________________________________
```

3. Loss and accuracy

Due to system configuration constrained, we have used 10 epochs only with 128 batch size to train the model. Total 16250 images have been used to train model and 8750 images used for validating the performance at each epoch time. Below are two graphs of loss and accuracy learning curve of model at each epoch during the training.



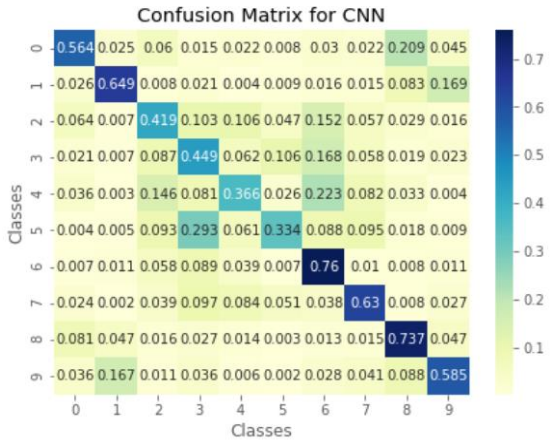
Model Performance

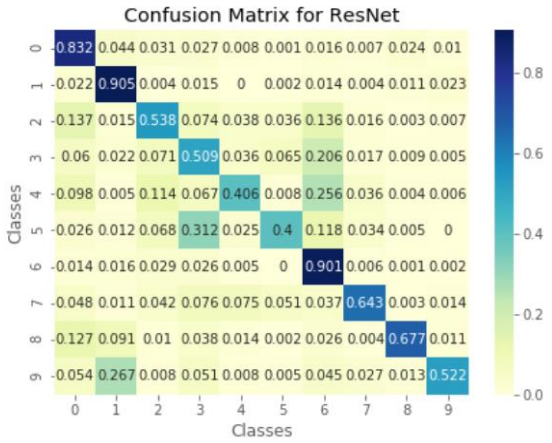
After implementing both CNN and ResNet model on CIFAR-10 data, we ran both the models on test data and compared both model performance with respect to score (accuracy, precision, recall & f1), confusion matrix, roc & auc of each class and classification report.

1. Compare performance score: Below table show all performance score for both models

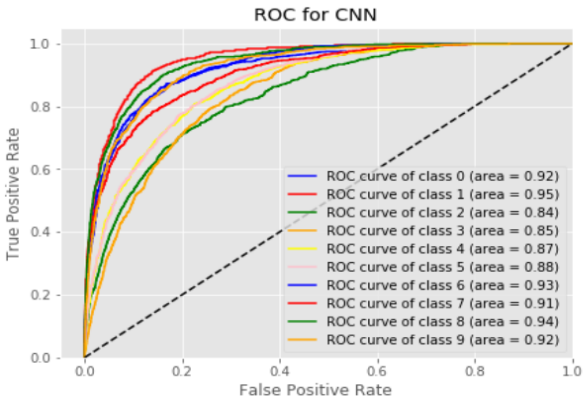
	Accuracy	Precision	Recall	F1
CNN	0.5493	0.555608	0.5493	0.544463
ResNet	0.6333	0.671048	0.6333	0.628126

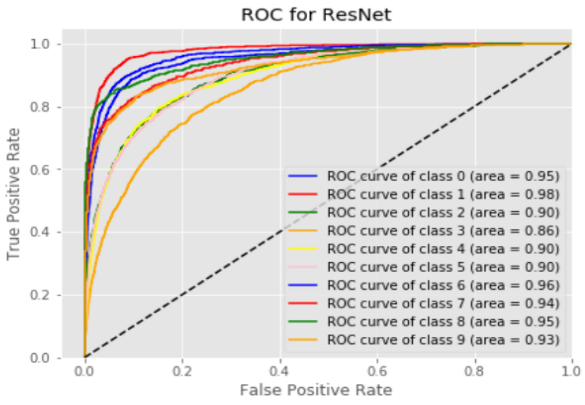
2. Compare confusion matrix: A confusion matrix visualizes and summarizes the performance of a classification model. Below is the confusion matrix comparison of models



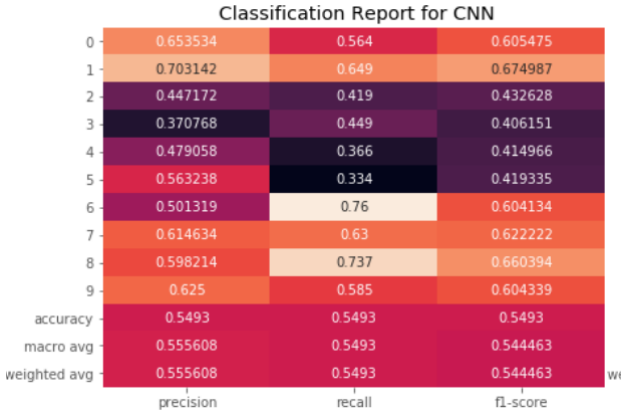


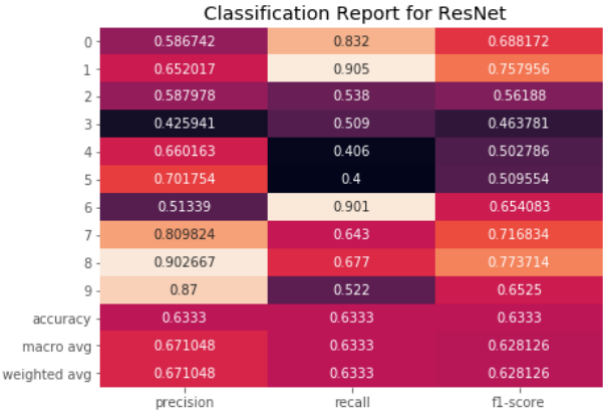
3. Compare ROC & AUC: An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds between False Positive Rate and True Positive Rate. Below is the ROC and AUC plots for both the model.





4. Compare classification report: A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report. Below is classification report for both the models.





Conclusion

We built the CNN model and ResNet (pre-trained model) image classification model on provided CIFAR-10 data set. We compared the model performance of both the model and found that ResNet model is performing better than CNN model. This was expected, since ResNet is very powerful pretrained model. Due to system configuration constrained we trained CNN model on limited images and less epoch time, so model performance can be improved to take more images and increase epoch time with low learning rate.