Image Classification of CIFAR-10 Dataset Images Using Deep Neural Networks

Basil Varghese
DS5220: Supervised Machine Learning (Spring 2022)
Khoury College of Computer Sciences, Northeastern University
varghese.b@northeastern.edu

Abstract

In this project we aim to compare the performance of two deep learning models in the classification of images in the CIFAR-10 dataset. Deep Neural Network models with convolution layers and without convolution layers were trained and performance metrics such as accuracy, time to convergence and loss were analyzed. After training the models for 50 epochs, Convolutional Neural Network model showed the best performance with an accuracy of 82%.

Introduction

Deep learning[1] has witnessed a monumental growth in the past decade with the advent of big data, availability of large datasets and high-performance GPUs. Numerous research and has gone into creating new deep learning models for various applications and the introduction of deep learning frameworks such as TensorFlow[2], Keras[3] and PyTorch[4] has made it easier for researchers to implement complex deep learning models. Convolutional neural networks[5] have been developed to learn the spatial and temporal features of an image and can hence be used for applications related to image and video data. In this project we analyze the performance of convolutional neural networks against deep neural networks without convolutional layers on the CIFAR-10[6] dataset. This dataset contains images of size 32x32x3 equally balanced across 10 categories. The dataset has 50,000 images in the training set and 10,000 images in the test set. The 10 categories are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. After training the two models on the training dataset, we used the trained model to predict the labels for images in the test dataset and analyzed the results.

Methods

Deep Neural Networks (Deep NN)

Neural Network models with multiple hidden layers are called Deep Neural Network Models. The input data is fed into the input layer which performs some weighted additions using specified weights and the output is passed into the subsequent layers where similar computations are performed. The computation in the final output layer returns the probability that the given input belongs to each class (for a classification problem). The output in each iteration is compared to the ground truth value and the weights in each layer are updated using the gradients of each variable. Various optimization techniques such as stochastic gradient descent[7], adam optimization[8] etc. are used to find the optimum set of weights for a given training set. Such models aim to mimic the behavior of a human brain and are extensively used in problems such as Image recognition, Objection detection, language translation etc.

When all the nodes of one layer is connected to every other node in the next layer, it is called a Dense Layer. A typical deep neural network consists of multiple dense layers of various sizes (number of nodes) stacked one after the other and an output layer in the end. Such models typically perform well for common classification problems such as bank loan prediction systems, credit card fraud detection, customer churn prediction, etc. However, for applications related to images, videos, audio and textual data, these models find it difficult to generalize well on the training dataset.

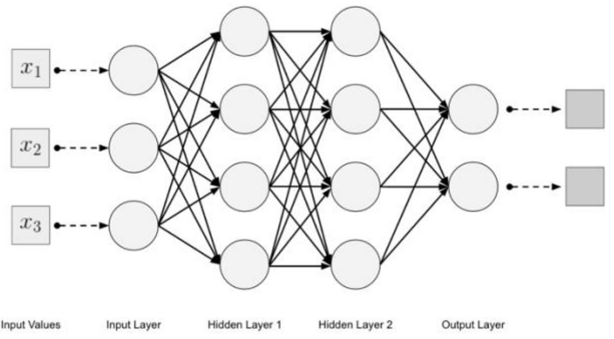

Figure 1: A typical Deep NN architecture

Convolutional Neural Networks (Conv NN)

Images are generally represented using a 3D matrix of various sizes based on the quality of the image. For an image of size 32x32, the total number of values needed to represent the image is 32x32x3 = 3072 per image. Training a Deep Learning model on image data would be computationally expensive and would not be able to capture the spatial and temporal features in an image. Convolutional Neural networks solve this problem using various filters, and pooling layers followed by dense layers at the end. The use of convolutional layers reduces the dimensions of the image while maintaining the important features in the image. This would result in faster training time along with better prediction performance. The main components of a convolutional neural network are:

1) **Convolutional Layer(filters)**: A set of 3D matrices which is multiplied by the input layer to produces a new layer of smaller dimension. Several filters are used in each layer based on the application. Filters do the job of extracting high level features such as edges and corners from an image.

2) Pooling Layer: Similar to filters, a pooling layer reduces the dimensions of the input layer which would decrease the computational complexity of the model while maintain the rotational and positional invariant features in an image. A pooling layer also helps remove noise from the image.

3) **Dense Layer:** The output from the convolutional layers is flattened and fed into multiple dense layers to learn any inherent features that might be present in the convolutional layers. A typical convolutional NN architecture contains multiple stacked convolutional + pooling layers and followed by a dense layer towards the end.

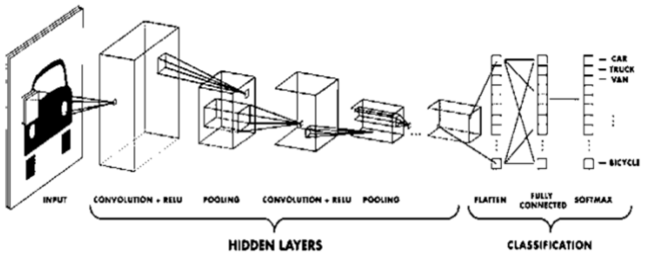

Figure 2: A typical Convolutional NN architecture[5]

Approach

In this project, the performance of convolutional neural networks is compared against a deep neural network model without convolutional layers. Data is imported from keras library in python and split into training and testing datasets. The training data was preprocessed and then used to train the neural network using tensorflow with 20% of the training data used for validation on which accuracy was calculated in each iteration. Once the model had converged, the test dataset was used to calculate model accuracy. Below are the architectures used on both models.

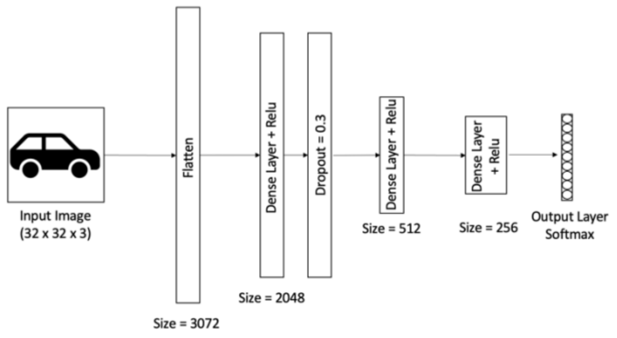

Figure 3: Architecture of DNN model used

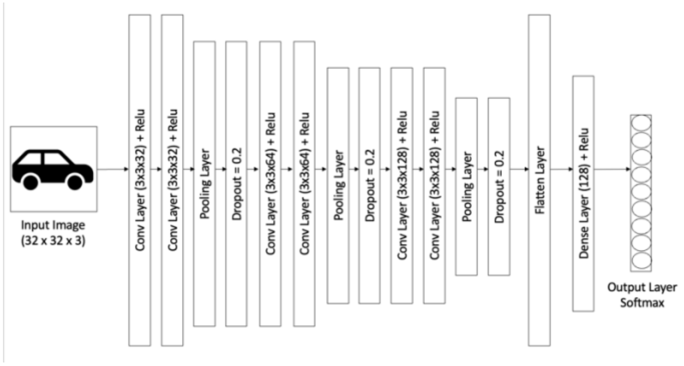
Input Image
(32 x 32 x 3)
Conv Layer (3x3x32) + Relu
Conv Layer (3x3x32) + Relu
Pooling Layer
Dropout = 0.2
Conv Layer (3x3x64) + Relu
Conv Layer (3x3x64) + Relu
Pooling Layer
Dropout = 0.2
Conv Layer (3x3x128) + Relu
Conv Layer (3x3x128) + Relu
Pooling Layer
Dropout = 0.2
Flatten Layer
Dense Layer (128) + Relu
Output Layer
Softmax

Figure 4: Architecture of CNN model used

The models were trained for 50 epochs and the accuracy, loss values across the epochs were tracked. Additionally, the time for training was also calculated. Adam optimizer was used for optimization along with categorical cross entropy as the loss function for both the models. Categorical cross entropy loss[9] is given by:

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i * \log \hat{y}$$

Where $\hat{y}$ is the i-th model output value, y_i is the target value. In each epoch, a batch size of 128 was used for batch training.

Results

Below are the results from both the models.

Model	Accuracy	Loss	Time to convergence
Deep NN	51%	1.39	20.49 minutes
Conv NN	83.4%	0.739	4.8 hours

Table 1: Comparison of model performance on test data between Deep NN model and Convolutional NN

The loss value along with the accuracy of the model on validation data in each epoch was tracked and plotted as shown below. After several epochs, we can see the loss values of accuracy values have plateaued. This indicates that the model has converged and cannot improve its performance.

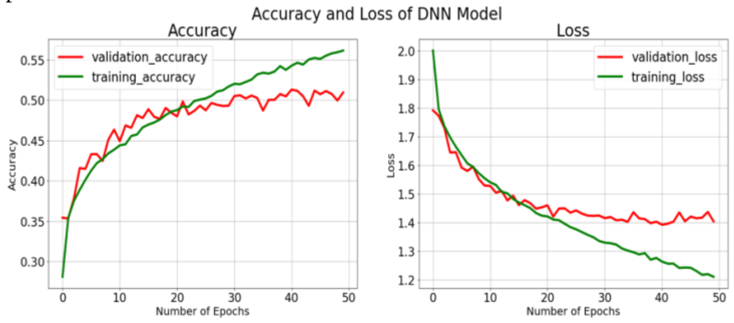

Figure 5: Accuracy and Loss of Deep NN model

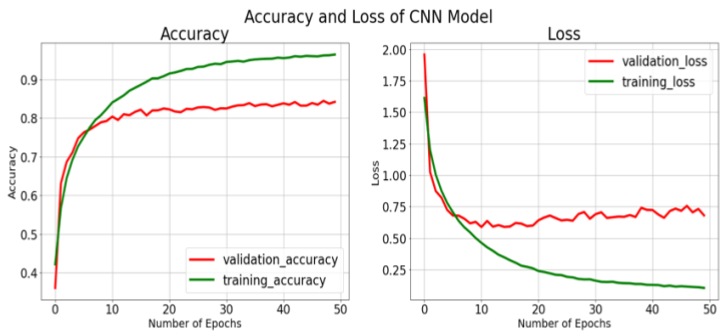

Figure 6: Accuracy and Loss of Convolutional NN model

Conclusion

From the results above, we can conclude that Convolutional Neural Networks have better performance in classifying image data. The DNN model converged faster as it was a much simpler model compared to the more complex CNN model. However, the accuracy and loss value of the CNN model is significantly better than DNN model.

References

[1] Deep Learning - *https://machinelearningmastery.com/what-is-deep-learning/*

[2] TensorFlow- *https://www.tensorflow.org/*

[3] Keras – *https://keras.io/*

[4] PyTorch – *https://pytorch.org/*

[5] Convolutional NN – *https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53*

[6] CIFAR-10 – *https://www.cs.toronto.edu/~kriz/cifar.html*

[7] Stochastic Gradient Descent – *https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31*

[8] Adam optimizer – *https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/*

[9] Github Link - *https://github.com/BasilKVarghese/CIFAR-10_CNN*