Big data and Machine Learning Coursework : Object Recognition
By Tanmay Mittal - 2218439

Introduction

Due in part to practical and affordable camera technology, the volume of visual data is rapidly increasing. With applications ranging from medical diagnostics to Snapchat effects, teaching computers to recognize things in a scene offers a wide range of potential uses. Despite years of study in the disciplines of computer vision and machine learning, the topic of object recognition remains difficult and unsolved for both academic and industrial researchers. The CIFAR-10 dataset consists of 6000 pictures per class in 10 classes totaling 60000 32x32 color images. 10000 test photos and 50,000 training images are available. Five training batches and one test batch, each containing 10,000 photos, make up the dataset. Exact 1000 randomly chosen photos from each class make up the test batch. The remaining photos are distributed across the training batches in random order, however certain training batches can have a disproportionate number of images from a particular class. The training batches consist of exactly 5000 photos from each class combined. These are the classes in the dataset, Airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. The classes are completely mutually exclusive. i.e. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks. Two of the machine learning algorithms were used Neural Networks and Convolutional Neural Network for processing the image and then confusion matrix was used to check the accuracy. The result which came out to be wasn't satisfactory, as the maximum accuracy which was 40%. This is near the desired output which was asked to bring in the project.

Methods

As mentioned in the introduction, Neural Networks and CNN were used. Firstly NN, Computer programmes called neural networks are made to spot patterns. They take their name from the structure of the human brain in terms of architecture. There are three different kinds of layers in them: input, hidden layers, and output. A signal is received by the input layer, processed by the hidden layer, and then decided upon or forecasted by the output layer based on the input data. Artificial neurons that are linked form the building blocks of each network layer. The activation function used in the convolution layers and hidden layers is ReLU which works well for those layers. The output layers need to classify images into 10 different classes and uses Softmax activation function. Softmax function forces output of the neural network to sum to 1 so that they can represent a probability distribution across a bunch of discrete mutually exclusive alternatives. SGD, Similar to gradient descent, except with only one example instead of the full set receiving weight updates. However, this approach results in jerky updates and has uneven convergence. The mini-batch gradient approach, which updates weights on a mini-batch that is a subset of the total set, is a superior substitute for the SGD method. Instead than using 1 point as in SGD, we use m points to estimate gradient. Gradient approximations are used by stochastic and mini-batch descents to speed up calculations, but choosing the right learning rate can be challenging. Gradient descent oscillates and requires several iterations to reach the optimal weights for complex functions. A neural network is trained for one cycle during an epoch using all of the training data. We only utilise each piece of information once within an era. A pass is considered to be one pass if it is both forward and backward: Each epoch consists of one or more batches in which the neural network is trained using a portion of the dataset. Thereby, for this model we used epochs for 100 times. It helped us getting the stability in the loss function.

The second method used was the Convolution Neural Networks, Similar to gradient descent, except with only one example getting weight changes as opposed to the entire collection. This method, however, causes jerky updates and uneven convergence. A better alternative to the SGD technique is the Adam, The Adam optimizer produces results that are typically superior to those of conventional optimization methods, takes less time to compute, and needs fewer tuning parameters. Adam is suggested as the default optimizer for the majority of applications as a result of all of that. Use sparse categorical cross entropy when your classes are mutually exclusive (e.g. when each sample belongs exactly to one class) and categorical cross entropy when one sample can have multiple classes or labels are soft probabilities (like [0.5, 0.3, 0.2]). For case when classes are exclusive, you don't need to sum over them - for each sample only non-zero value is just $-logp(s\in c)$−logp(s∈c) for true class c. This allows to conserve time and memory. Consider case of 10000 classes when they are mutually exclusive - just 1 log instead of summing up 10000 for each sample, just one integer instead of 10000 floats.

Result

For Neural Networks, The accuracy which turned to appear was 21%. This is way less than what the expectation was.

```
from sklearn.metrics import confusion_matrix, classification_report
y_pred = nn.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print('Classification Report: \n', classification_report(y_test, y_pred_classes))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.24      0.54      0.33       100
           1       0.23      0.14      0.17       100
           2       0.11      0.04      0.06       100
           3       0.00      0.00      0.00       100
           4       0.00      0.00      0.00       100
           5       0.12      0.35      0.18       100
           6       0.22      0.51      0.31       100
           7       0.00      0.00      0.00       100
           8       0.33      0.41      0.37       100
           9       0.25      0.10      0.14       100

    accuracy                           0.21      1000
   macro avg       0.15      0.21      0.16      1000
weighted avg       0.15      0.21      0.16      1000
```

Fig 1 Confusion Matrix, after training the model in Neural Networks.

Despite having achieved respectable performance in image classification, traditional neural networks have been characterized by feature engineering, a time-consuming process that leads to poor generalization to test data. I provide a convolutional neural network (CNN) method for categorizing CIFAR-10 datasets in this study. It has been demonstrated in earlier publications that this strategy can increase performances without feature engineering. The underlying picture characteristics were extracted using pooling layers and learnable filters.

```
print('Classification Report: \n', classification_report(y_test, y_classes))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.41      0.59      0.48       100
           1       0.53      0.31      0.39       100
           2       0.31      0.30      0.31       100
           3       0.38      0.13      0.19       100
           4       0.35      0.39      0.37       100
           5       0.45      0.33      0.38       100
           6       0.38      0.62      0.47       100
           7       0.35      0.33      0.34       100
           8       0.43      0.60      0.50       100
           9       0.51      0.45      0.48       100

    accuracy                           0.41      1000
   macro avg       0.41      0.40      0.39      1000
weighted avg       0.41      0.41      0.39      1000
```

Fig 2 Confusion Matrix, after training the model in CNN.

The test accuracy for ANN for 100 epochs was 21% whereas, the test accuracy of CNN for same number of epochs was 41%. Although, the accuracy for CNN was much better than that of ANN, but it is still very low. The reason for such low accuracy is that the dataset is not very simple. Plus, the available dataset is also very little

Also, the computation speed of CNN is also very low compared to the ANN

Conclusion

We used a convolutional neural network to categorise photos into 10 categories in this article. The network was trained using the CIFAR-10 dataset as a baseline. For updating weights, the Adam optimization method provides the most accurate picture categorization. I would need to work more on the Neural Networks as after running epochs 100 times, my accuracy was still 21%. And maybe by hyper tuning the parameters more often might increase the accuracy. Whereas in CNN, The accuracy touched 41%, which is more than fair. Now, working on Cifar-10, we can jump to Cifar-100 and deal with bigger data-sets to increase our expertise.

Referencing

- Raveen Doon, T. K. (2018, November 30). Cifar-10 Classification using Deep Convolutional Neural Network.
- Ba, D. K. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.