Image Classification using CIFAR-10

Saurav Patel, Karan Patel *Department of Computer Science & Engineering, Institute of Technology Nirma University*
Ahmedabdad, India

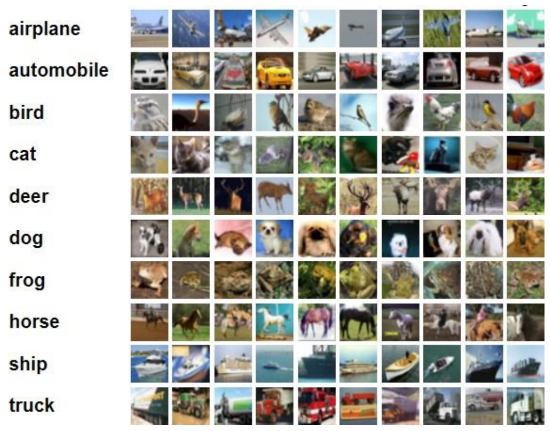
***Abstract*—In this place paper, we study the efficiency of various classifiers on the CIFAR-10 dataset, and build an ensemble of classifiers to reach a better accomplishment. We have improved accuracy using different classifiers like K Nearest Neighbors (KNN), Decision Tree, Random Forest, Boosting, Artificial Neural Network (ANN), and Convolutional Neural Network (CNN) on CIFAR- 10 Dataset and ensemble it with a CNN to increase its accuracy. Our approach corrects our best CNN model with an accuracy of 98.69%.**

***Index Terms*—KNN, Decision Tree, Random Forest, Boosting, ANN, CNN, Ensamble of Classifiers**

I. Introduction

The **CIFAR-10** dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.



We will begin with training simple classifiers and incrementally move towards more complex alternatives: Decision Trees, Random Forests, Boosting and K-Nearest Neighbors (KNN), Artificial Neural Networks (ANN). Eventually, we will train a Convolutional Neural Network (CNN). In an attempt to improve the state-of-the-art accuracy, we will devise an ensemble of classifiers. We will compare the performance of the previously mentioned classifiers, feature extractors, and their effect on the final ensemble.

Identify applicable funding agency here. If none, delete this.

CIFAR-10 presents a challenging classification problem. 32 × 32 images don't contain enough information for most classifiers to draw clear decision boundaries. A clear example of this is the confusion between "cat" and "dog" classes. The images objects are different in scale, rotation, position, and background. Some of the images are very unclear and hard to classify (even for human being).

II. Feature Reduction

PCA is an unsupervised linear transformation technique. It helps us to identify patterns in data based on the correlation betweeen features. In a nutshell , PCA aims at finding the directions of maximum variance in high-dimensional data and projects it onto a new subspace of lower or equal number of dimensions than original feature space.

Suppose we have a (n,d) dimensional data-set with d being the number of dimensions using PCA we create a (d,k) dimensional trnasformation matrix W that allows us to map a sample vector x onto a new k-dimensional feature subspace that has fewer dimensions than the original d-dimensional feature space. As a result the (n,d) dimension dataset get transformed into a (n,k) dimension data i.e. with k features. These k-dimensions are automatically sorted in decreasing order of their importance i.e. the first principal component will have the largest possible variance.
PCA doesn't delete any features to reduce dimensionality instead it kind of takes in all the features and outputs entirely new features by transforming them linearly.

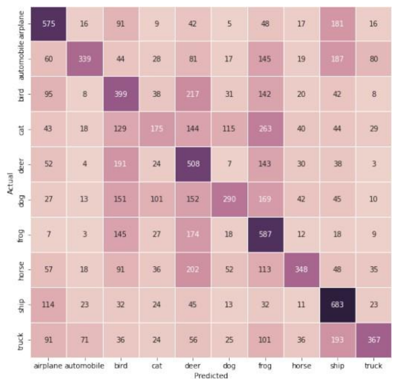
III. Classification Techniques

A. K-Nearest Neighbors

K-Nearest Neighbours Classifier is the one of the most simple and basic non-parametric classification method in Machine Learning, which consists of the geometric distance between k closest training samples in the feature space, which is robust to treat the noisy and unbalance dataset. The CIFAR-10 dataset contains enough sample size and reasonable samples distribution per class so that KNN suppose to work fine on object recognition for our dataset.

It requires three elements: The set of labelled samples, distance metric to compute distance between samples, and the number of k nearest neighbors to retrieve. Given the images dataset, sends a query image sample to the algorithm server, while the server returns k neighbors of query image sample from within dataset where each distance between the k neighbors and query image sample is minimal in dataset, that is, the k neighbors are the k nearest neighbors to query image. The algorithm approach collect the k neighbors samples (X1,X2,. ,Xk) of query image sample (Y) and classify the Y according to the closest distance within X1,X2,. ,Xk:

$$C(Y) = MinDistance_{i=1}^{k}(X_i, Y)$$



Accuracy : 42.71%

B. *Decision Tree*

Decision tree-based algorithms are an important part of the classification methodology. Their main advantage is that there is no assumption about data distribution, and they are usually very fast to compute. In image classification, the decision trees are mostly reliable and easy to interpret, as their structure consists of a tree with leaves which represent class labels, and branches that use logical conjunction to produce a value based on an ”if-then” rule. These values produce a set of rules that can be used to interpret the instances in a given class.

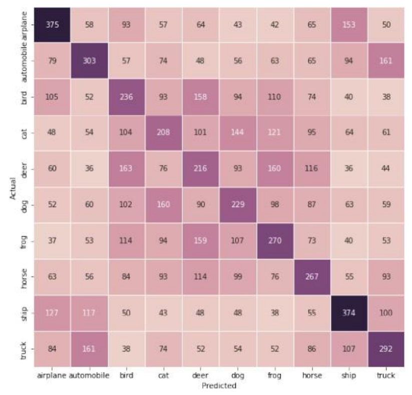
Image classification can be done by mainly four decision tree techniques:

- J48 (C4.5)
- Hoeffding Tree
- Random Tree

The J48 algorithm builds a decision tree which classifies the class attribute based on the input attributes. The algorithm is based on the C4.5 algorithm. The algorithm uses a greedy search method to create decision trees, and allows changing different parameters in order to obtain a better classification accuracy.

The Hoeffding Tree algorithm is a decision tree algorithm that can learn from massive data streams. The assumption presumed by this algorithm is that the distribution does not change over time. This algorithm works well with small samples, as it uses the Hoeffding bound which computes the number of observations that are necessary to estimate statistics values within a prescribed precision.

A Random Tree algorithm draws a random tree from a set of possible trees. The distribution of trees is considered uniform, as all trees from the set have the same chance of being sampled.

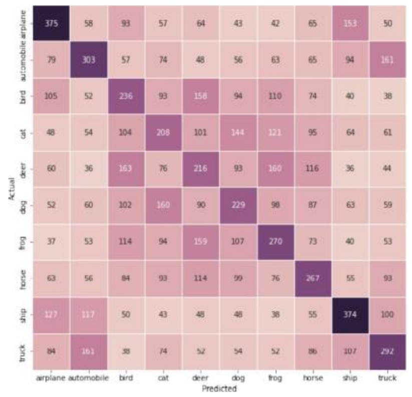


Accuracy : 27.7%

C. *Random Forest*

A Random Forest algorithm is an ensemble classifier that draws multiple decision trees using a bagging approach, hence the same sample can be selected multiple times, while some other sample may not be selected at all. In-bag samples are used to train the trees, while out-of-the-bag samples are used for internal cross-validation. The algorithm usually yields a very good classification accuracy, and it can handle multicolinearity well.

Random forests are good classifiers because they are highly accurate, robust to outliers, and does not suffer from overfitting. The main drawback is that they are slow in generating predictions due to the multiple decision trees.



Accuracy : 27.7%

Using Ensemble Techniques

Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

- Bagging (Boostrap Agregation) : Random Forest
- Boosting : AdaBoost, GradientBoosting, XgBoost (Extreme Gradient Boosting)

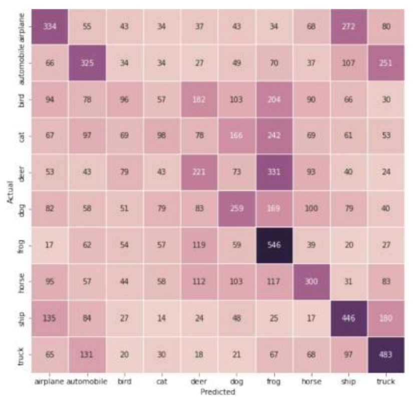
Decision Tree suffers Overfitting (since decision made will work fine for train set but poorly on test set) because of Low bias, High variance Random Forest Combines multiple DT which reduces the variance, hence gives better solution than DT.

D. Boosting

Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. Firstly, a model is built from the training data.Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

- ADABoost (ADAPTIVE BOOSTING):

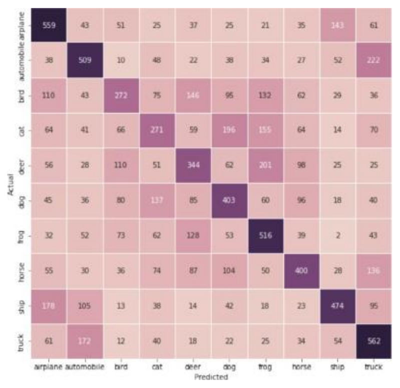
  The proposed AdaBoost is to build a classifier based on the classification results of each classifier. AdaBoost generally uses the decision tree as the weak classifier. In each decision tree, different thresholds can be selected as decision points. The advantage of this framework is that it can be applied to almost all current machine learning algorithms and has improved the original accuracy. At the same time, no prior conditions are needed. The decision of different classifiers can reduce the lower limit of classifiers.



Accuracy : 31.08%

- Gradient Boosting:
  In gradient boosting machines, or simply, GBMs, the learning procedure consecutively fits new models to provide a more accurate estimate of the response variable. The principle idea behind this algorithm is to construct the new base-learners to be maximally correlated with the negative gradient of the loss function, associated with the whole ensemble. The loss functions applied can be arbitrary, but to give a better intuition, if the error function is the classic squared-error loss, the learning procedure would result in consecutive error-fitting.
  This high flexibility makes the GBMs highly customizable to any particular data-driven task. It introduces a lot of freedom into the model design thus making the choice of the most appropriate loss function a matter of trial and error. However, boosting algorithms are relatively simple to implement, which allows one to experiment with different model designs. Moreover the GBMs have shown considerable success in not only practical applications, but also in various machine-learning and data-mining challenges.

- XGBoost (Extreme Gradient Boosting):
  XGBoost is a popular and efficient open-source implementation of the gradient boosted trees algorithm. It has been applied to solve many classification problems in diverse fields. The most important factor behind the success of XGBoost is its scalability in all scenarios. The system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings. The scalability of XGBoost is due to several important systems and algorithmic optimizations. These innovations include a novel tree learning algorithm is for handling sparse data; a theoretically justified weighted quantile sketch procedure enables handling instance weights in approximate tree learning. Parallel and distributed computing makes learning faster which enables quicker model exploration. More importantly, XGBoost exploits out-of-core.

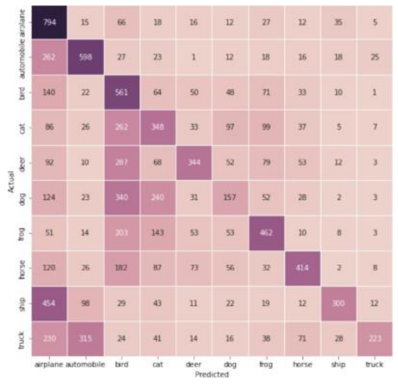


Accuracy : 43.1%

E. ANN

Artificial Neural Networks (ANN) are algorithms based on brain function and are used to model complicated patterns and forecast issues. The Artificial Neural Network (ANN) is a deep learning method that arose from the concept of the human brain Biological Neural Networks. The development of ANN was the result of an attempt to replicate the workings of the human brain. The workings of ANN are extremely similar to those of biological neural networks, although they are not identical. ANN algorithm accepts only numeric and structured data.

ANNs are efficient data-driven modelling tools widely used for nonlinear systems dynamic modelling and identification, due to their universal approximation capabilities and flexible structure that allow to capture complex nonlinear behaviors.



Accuracy：42.71%

F. CNN

Convolutional Neural Networks are the Neural Networks with at least one, at most all convolutional layers, and this class of Neural Networks is particularly useful for image recognition

To demonstrate how standard neural networks do not scale well as image size increases, let's again consider the CIFAR-10 dataset. Each image in CIFAR-10 is 32×32 with a Red, Green, and Blue channel, yielding a total of 32×32×3 = 3,072 total inputs to our network.

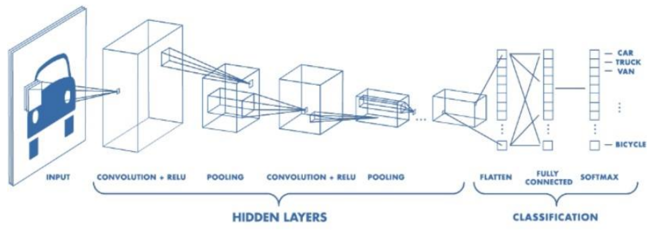


Figure 2: Architecture of a CNN (Source)

A total of 3,072 inputs does not seem to amount to much, but consider if we were using 250×250 pixel images — the total number of inputs and weights would jump to 250×250×3 = 187,500 — and this number is only for the input layer alone! Surely, we would want to add multiple hidden layers with a varying number of nodes per layer — these parameters can quickly add up, and given the poor performance of standard neural networks on raw pixel intensities, this bloat is hardly worth it.

Instead, we can use Convolutional Neural Networks (CNNs) that take advantage of the input image structure and define a network architecture in a more sensible way. Unlike a standard neural network, layers of a CNN are arranged in a 3D volume in three dimensions: width, height, and depth (where depth refers to the third dimension of the volume, such as the number of channels in an image or the number of filters in a layer).

Convolution Operation

The name of Convolutional Neural Network is because they used multiple convolutional filters to conduct the convolution operations to the inputs to each convolutional layer. Suppose a 3 × 3 convolutional filter, it performs the dot product to a same size area on the inputs and get a scalar output, and then do the same operation step by step goes from left to right, top to bottom according to the stride size s. Simply speaking, each filter can be thinks as containing some particular features such as a color dot or a shape. By adding more convolutional filters, the model can have the ability to abstract more higher-level features, and later reconstruct these features to make the prediction.

Pooling Layer

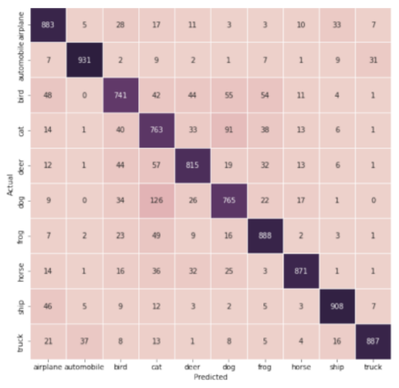
The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

Fully connected Layers

Neurons in FC layers are fully connected to all activations in the previous layer, as is the standard for feedforward neural networks. FC layers are always placed at the end of the network (i.e., we don't apply a CONV layer, then an FC layer, followed by another CONV) layer.

```
INPUT => CONV => RELU => POOL => CONV => RELU => POOL => FC
```

To make this example more concrete, again consider the CIFAR-10 dataset: the input volume will have dimensions 32×32×3 (width, height, and depth, respectively). Neurons in subsequent layers will only be connected to a small region of the layer before it (rather than the fully connected structure of a standard neural network) — we call this local connectivity, which enables us to save a huge amount of parameters in our network. Finally, the output layer will be a 1×1×N volume, which represents the image distilled into a single vector of class scores. In the case of CIFAR-10, given ten classes, N = 10, yielding a 1×1×10 volume.



Accuracy : 98.69%

Convolutional Neural Networks are very similar to the Neural Networks; they all constructed by multiple layers with bunch of neurons in each layer. The neurons are actually the weights, and generally they are tiny random numbers at the beginning. The network then performs the forward propagation and back propagation to train these weights, intend to the latent non-linear function. The forward propagation performs the dot product of the inputs and weights, then typically follows with some activation functions. The back propagation updates the weights in each layers by taking their derivatives respect to the defned loss function, and the magnitude of each update are controlled by the learning rate.

$$E(C_1, C_2) = \underset{w_i, w_j \in R}{\text{argmax}} \; w_i \times C_1 + w_j \times C_2$$

We estimate all parameters recursively. For example, we search for weights for C1 and C2 to produce C1+2. Then, we search for weights for C1+2 and C3 to produce C1+2+3 (i.e. E(E(C1, C2), C3)), and so on.

IV. Experiments & Discussion

A. *K Nearest Neighbors*

Test Accuracy : 42.71%

B. *Decision Tree*

Test Accuracy : 27.70%

C. *Random Forest*

Test Accuracy : 27.70%

D. *ADA*

Test Accuracy : 31.08%

E. *XGBoost*

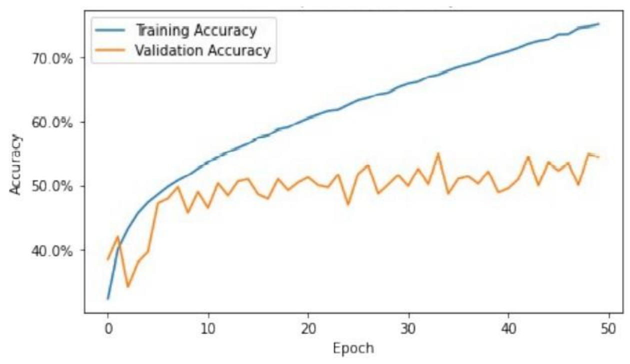
Test Accuracy : 43.10%

F. *ANN*

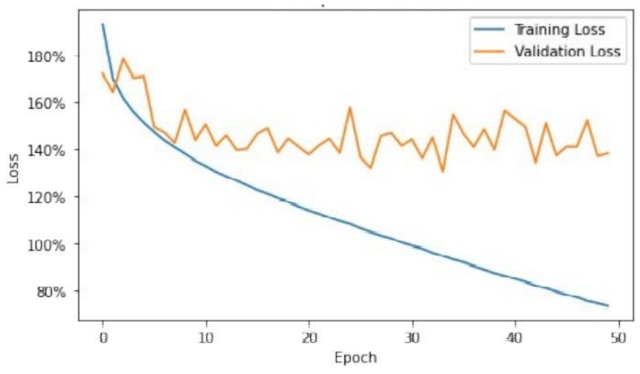
Training Data
10 Epoch : 52.67%
20 Epoch : 59.84%
30 Epoch : 65.27%
40 Epoch : 70.40%
50 Epoch : 75.06%

Test Accuracy : 42.01%

ANN Epoch and Accuracy



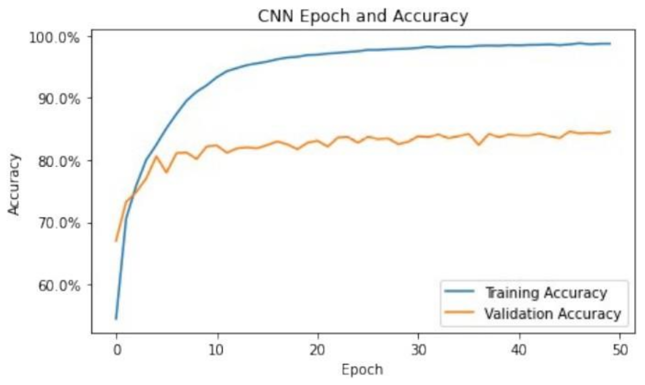
ANN Epoch and Loss

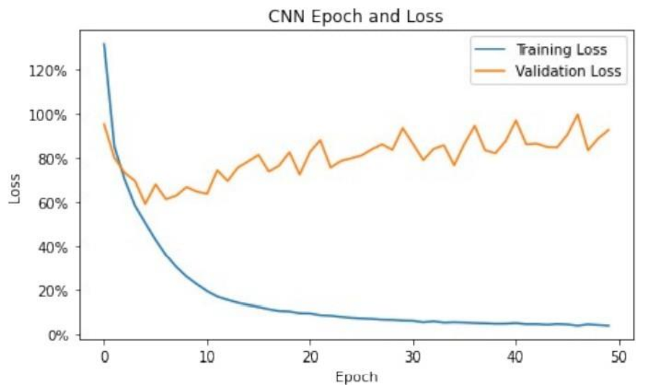


G. *CNN*

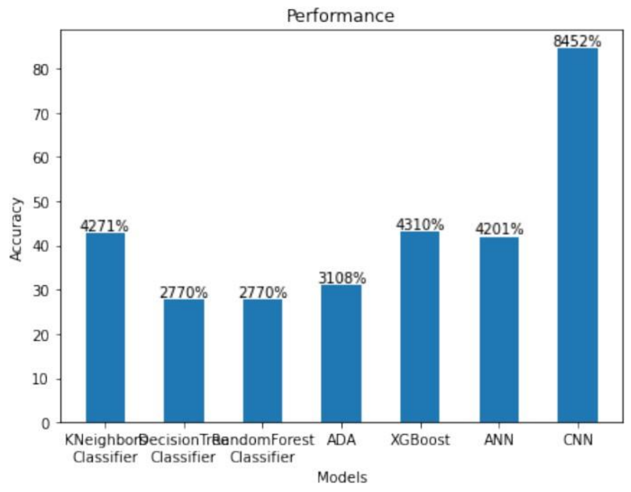
Training Data
10 Epoch : 92.00%
20 Epoch : 96.86%
30 Epoch : 97.90%
40 Epoch : 98.46%
50 Epoch : 98.69%

Test Accuracy : 84.52%





We propose an ensemble-based approach to improve the CIFAR-10 test accuracy. We experiment with possible learning models, and try to find the best classifier to reduce classifier confusion and improve accuracy. We found that Artificial Neural Network (ANN) consistently improves the accuracy of Convolutional Neural Networks (CNN).In ANN accuracy percentage after 50 epochs is 75.63%, while in CNN We have implemented hidden layers, convolution and ’ pooling techniques in such way that in 50 epochs we reached at accuracy of 98.69%.



References

[1] Bianca Antonio, Kevin Nguyen, Christine Phan, Cindy Vu, David Yu Image Classification

[2] Adrian Rosebrock Convolutional Neural Networks (CNNs) and Layer Types.