Object Recognition in Images

Xinyue Wang, Youjia Li, Jinghan Chen

October 4, 2018

Abstract

The objectives of this project is to develop an algorithm to classify the object in images into ten classes. We used CIFAR-10 data set provided in a Kaggle competition which consists of 60,000 32 $\times$ 32 color image in 10 classes to train classifier. There are four techniques that used in this project, including K-nearest neighbors (KNN), Decision Tree, Random Forest and deep convolution neural networks (CNNs). We compare the performance of these classifiers, and it is shown that CNNs has the best performance, which achieves the accuracy of 70.4% .

1 Introduction

With the ever-increasing amount and abilities of modern cameras and monitoring systems, images and photos produced accumulates exponentially. The ability in humans to recognize thousands of object categories cannot match with the great demanding for image processing in terms of efficiency and accuracy. To tackle this task, traditional computer vision techniques are introduced to achieve image recognition and object recognition at first. However, with the development of Deep Learning algorithms, a new idea comes along over the past decades. Deep learning is so powerful and effective that it becomes the mainstream in computer vision. At the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2012, the champion program - an algorithm based on Deep Learning shook the world with astounding accuracy of 85% . Techniques like CNN, KNN and SVM have been applied in Deep Learning based object recognition. For this project, we aim to achieve something similar.

Object Recognition has recently become one of the most exciting fields in computer vision and AI. Object Recognition in Images competition is launched Kaggle that is both practically and theoretically interesting. it is trying to recognize and classify the object in an image into ten classes, the class includes airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.

In our project, we aim to solve this problem and would like to use different strategies to achieve higher performance. The machine learning techniques we used including k-nearest neighbors (KNN), Decision Tree, Random Forest and deep convolutional neural networks (CNN).

2 Data Preprocessing

2.1 Description of data sets

The dataset we used is the CIFAR-10 dataset provided by a Kaggle competition to develop an algorithm for object recognition in images. The CIFAR-10 dataset consists of 50,000 32 $\times$ 32 color images in 10 class, with 5000 images per class. Our training set contains 40,000 images and test set contains 10,000 images. The label classes in the dataset are shown in the following table.

label	Class Name	label	Class Name
1	airplane	6	dog
2	automobile	7	frog
3	bird	8	horse
4	cat	9	ship
5	deer	10	truck

There are some example images from training set.

2.2 Data Acquiring, wrangling and Cleaning

Data is loaded by using Keras API, and train and test images (32 px $\times$ 32 px) are stock into numpy.ndarray as matrices of shape (32, 32, 2), since given image is RGB images, which has 3 channels (red, green and blue). The label of each training image is acquired from trainLabels.csv, for K-nn, Decision tree, and random forest algorithm, we encode these string labels to the digits in the range 1-10, and for C-NN, we encode the labels to on hot vectors.

For some algorithms, the matrices of shape (32, 32, 2) is reshaped into 3072 dimensions, the first 1024 entries are for red, next 1024 for green and the last 1024 for blue. We also normalize the data so that all the individual samples to have unit norm and the same treatment in the model, the normalization also reduces the effect of illumination's differences for different images.

3 Implementation and Analysis

3.1 machine learning techniques

3.1.1 KNN

To correctly classify the test images based on the information of training images, it is intuitive to firstly compare the new images with old ones with respect to appropriate criteria and then make decisions. This idea is extended to k-NN(K nearest neighbors) classification, which stores all labeled (classified) cases and classify new cases decided by a similarity measure. The classification result is derived from the majority of k nearest neighbors of the testing case. k-NN algorithm has been applied statistical estimation and pattern recognition. In this project, k-NN is applied to object recognition by developing a k-NN classifier.

The procedure of k-NN is relatively simple compared to other classification algorithms since it barely uses the training data to do any generalization during the training phase. After the data preprocessing mentioned above, each image is converted to a normalized feature vector, which will be stored by k-NN algorithm. For each iteration in the test stage, regarding each test image, the algorithm computes the Euclidian distance between test image feature vector and stored image feature vectors as a form of similarity calculation. Next the algorithm takes k feature vectors into consideration and classified the test image to the class shared by the majority of neighbors. In the coding, k-NN algorithm is imported from sk-learn module in Python and all other parameters (e.g. leaf size, power parameter for distance) except k values has been optimized to achieve maximum accuracy.

3.1.2 Decision Tree classifier

Decision Tree is a non-parametric supervised learning method. A decision tree classifier is trained to solve the project problem and it is implemented by Scikit-Learn python library. The training dataset and the corresponding label is learned by the classifier. Once the tree is built, it can predict the class of the object in the image in test dataset. In particular, construed a decision tree can be illustrated as solve a classification problem by asking a series of questions, each question seems as the node of the tree, each node must select a feature test condition to split the training data into the smaller dataset. There are many algorithms provide the different strategy for specifying the test condition for the different attribute[1]. In this project,CART (Classification and Regression Tree) algorithm which constructed binary trees using the feature and threshold that leads to the largest information gain at each node is used. To predict the class of the test data, starting from the root node, once meet the node, split the data according to the test condition, and recursive do this step until the data point reaches the terminal nodes and assign the label.

3.1.3 Random Forest classifier

In this project, the random forest classifier is used. Random forest is a supervised learning algorithm, the 'forest' is an ensemble of decision trees, and it trained with the bagging algorithm which is a combination of learning models improve the overall prediction performance. In contrast to the decision tree, the condition for splitting a node is chosen is the best split among a random subset of the features, which means each tree is grown using a different subset due to the different split condition [2]. In particular, an image is classified by sending it from root to leaf through every tree and averaging the reached leaf distributions to predict the label. Due to the randomness and averaging, random forest may result in slight increases in bias and decreases in variance. Compare to the decision tree, it may have better performance in over-fitting. To implement the random forest, we used the python library Scikit-Learn.

3.1.4 CNN

Convolutional Neural Networks(CNN) are like ordinary Neural Networks which is made up of neurons that have learnable weights and biases. Except the layers of a CNN have neurons arranged in 3 dimensions: width, height, depth. So the input images in CIFAR-10 are an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively). Moreover, the final output layer would for CIFAR-10 have dimensions 1x1x10, because by the end of the CNN architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension.

As we know, a simple CNN is a sequence of layers and three main types of layers to build CNN architectures are: Convolutional Layer, Pooling Layer, and Fully-Connected Layer. We also use activation layer like Relu or Softmax and regularization functinon to prevent overfitting like Dropout layer. We will stack these layers to form a full CNN architecture.

In this project, we will use a high-level neural networks API - Keras to implement different architecture of layers. Keras is written in Python and capable of running on top of TensorFlow, CNTK, or Theano.

3.2 Models Training and Tuning

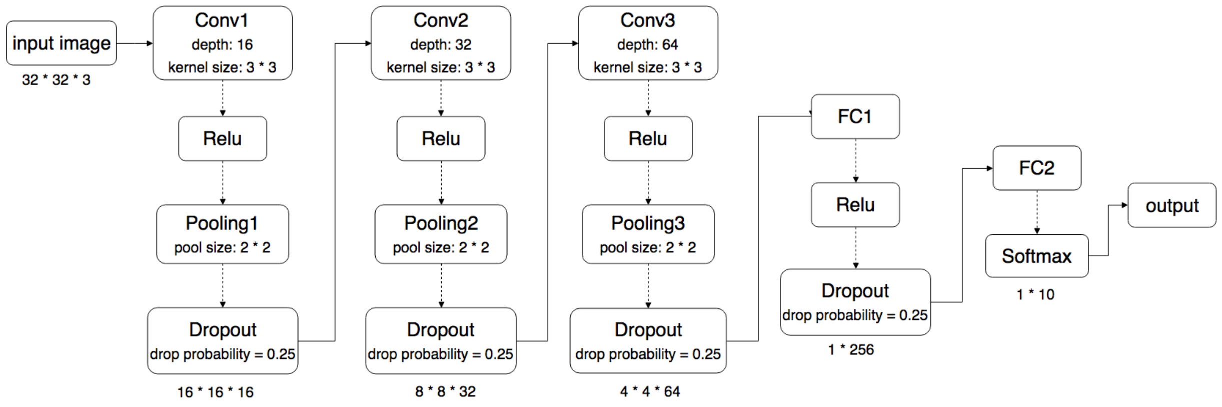
To select appropriate hyperparameters that would maximize performance for knn, random fores, and decision tree model, we use 10-fold cross-validation technique. The original training set is randomly partitioned into 10 equal size sub-dataset. For each time, 1 sub-dataset is selected as test data to evaluate the model, and the remaining 9 sub-dataset are used as training data. This process would be repeated 10 times, and averaged these 10 results to obtain the cross-validation score. In this project, we tuned the parameter to reach the lowest cross-validation error for each model.

For k-NN, the algorithm simply stores the feature vectors and does not establish a specific model from the training data. The training phase in this case requires no tuning process. For the testing phase, we tried to use different k values to train the classifier, and found the best k value is 5.

For random forest, when we trained the classifier with 50 estimators, 100 maximum features, the cross-validation error is the lowest.

For Decision Tree, when the min samples split is 2 and the max depth is none (the nodes are expanded until all leaves are pure or until all leaves contain less than min samples split), the model has the best performance.

For CNN, we started with a simple CNN model with the basic architecture [INPUT - [CONV - RELU - POOL] * 3 - FC - RELU - FC]. Then we tried to modified the simple model in two ways: changing parameters and builing a deeper CNN. After training and testing several models, we found the one with satisfying performance. We will elaborate more on the best one and the architecture of this model is shown below.



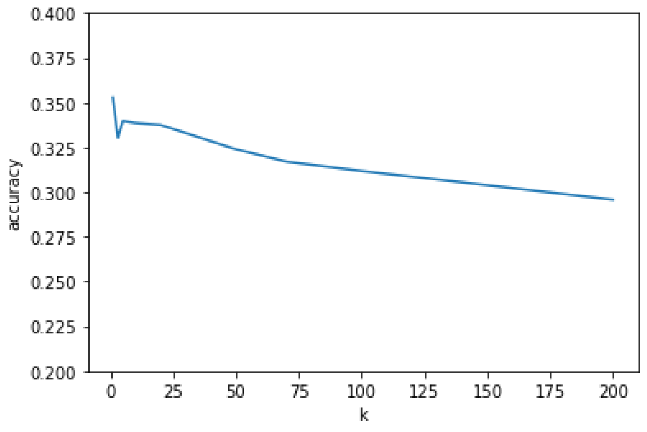
In each iteration, we considered 32 training examples at once and iterated 150 times over the entire training set. All the convoluational layers used 3 * 3 kernels and all the pooling layers used 2 * 2 pooling. We initially had 16 kernels in first convolutional layer. Then we switched to 32 after the first pooling layer, to 64 after the second pooling layer and switched to 128 after the third pooling layer. In order to prevent overfitting in a deeper CNN model, we used dropout layer with probability 0.25 after each pooling layer as well as after the first FC layer, and another dropout layer in the FC layer with probability 0.5. In the end, there is one softmax readout layer.

The model was trained with 3.1 GHz Intel Core i5 processor and 8 GB 2133 MHz LPDDR3 RAM.

3.3 Models Performance Evaluation

3.3.1 KNN

The performance evaluation and analysis of k-NN algorithm is mainly focused on the influence of different k values, which decides number of selected entries to as neighbors. With different values of k, the corresponding accuracy is tabulated below. It seems that the accuracy can achieve its maximum (35.39%) at k = 1. And It is clear that with the increasing of k, accuracy is generally decreasing but stabilized at around 30% . Intuitively, k should be picked in order to get the best possible fit for the dataset. Considering the performance of accuracy, k =1 seems to be the ideal value. However, a small value for K provides the most flexible fit, which will have low bias but high variance, which will lead to over-fitting. A more appropriate choice could be k=5 with an accuracy of 33.98% .



k	Accuracy
1	35.39%
3	33.03%
5	33.98%
10	33.86%
20	33.75%
50	32.39%
70	31.70%
100	31.18%
200	29.58%

3.3.2 Decision Tree and Random Forest classifier

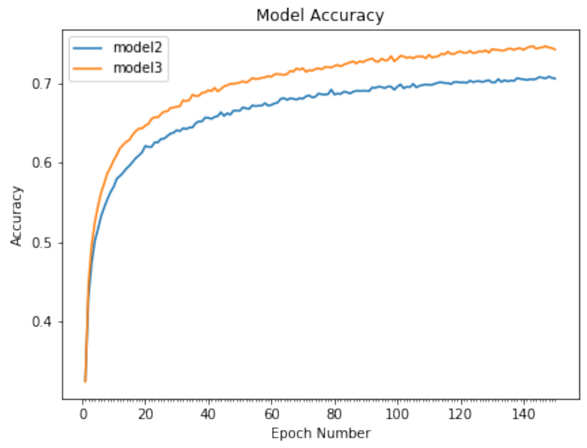
The decision tree classifier with max-depth 10 can get 25.4% accuracy, it is better than the random guess which has around 10% accuracy. But when we increase the max-depth of the tree from 10 to infinite, which means the nodes are expanded until all leaves are pure, the accuracy increase to 37.2% .But when a decision tree is grown to full depth, it is more likely to fit the noise in the data and results in overfiting the training data. The Random Forest classifier with 100 estimators and 50 maximum features achieves the 44.3% accuracy. Apparently, Random Forest has better performance than decision tree for this problem.

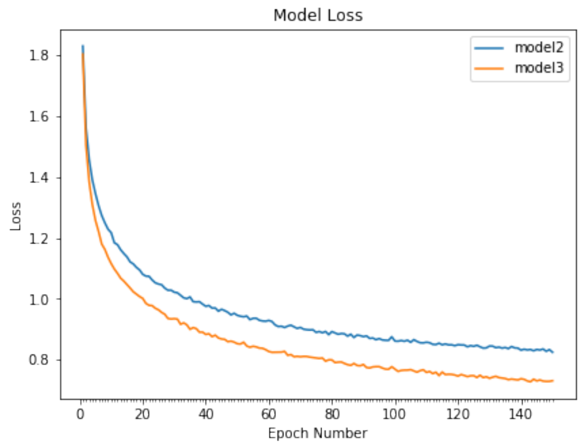
3.3.3 CNN

	CNN Architecture	Filter size	Feature number	Epoch number
model 1	INPUT - [CONV - RELU - POOL] * 3 - FC - RELU - FC	3 * 3	8 - 16 - 32	50
model 2	INPUT - [CONV - RELU - POOL] * 3 - FC - RELU - FC	3 * 3	8 - 16 - 32	150
model 3	INPUT - [CONV - RELU - POOL] * 3 - FC - RELU - FC	3 * 3	16 - 32 - 64 - 128	150

	Prediction accuracy	Runtime / second
model 1	68.19%	1699.02
model 2	70.39%	4891.00
model 3	73.38%	8054.41

As shown in the table above, model2 and model3 are modification of model 1 which is a simple CNN. Comparison between the prediction accuracy and loss of two modified models are show in the graph below.





It can be easy seen in the graph that model3 has slightly higher prediction accuracy as well as lower loss, while model3 cost almost twice runtime as model2. So we choose model2 as our CNN model.

3.4 Models Comparison

The classification accuracy of each model with the best parameter is shown int the following table.

Model	Accuracy
K-NN	33.9%
Decision Tree	37.2%
Random Forest	44.3%
CNN	70.4%

As shown in the above table, CNN model has distinct advantages over others in the tasks, which achieves 70.4% accuracy. There are several possible reasons.

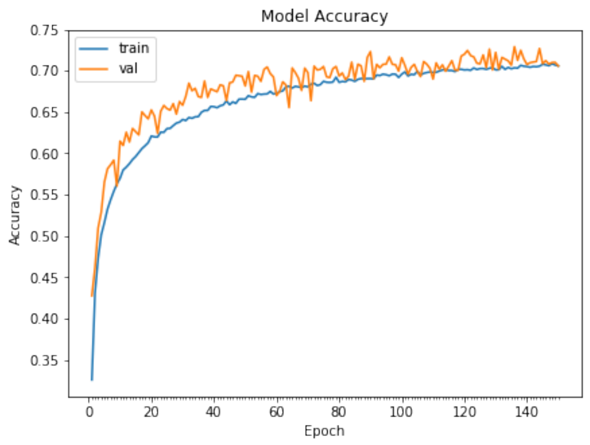
1. The trained data is 32 × 32 image, for K-nn, Decision Tree and Random Forest model, we reshaped the data into matrices of 3072 dimensions before training, which might loss some information. In contrast, CNN model uses ”convolution” to extract feature, which is an very effective feature extractor for image recognition compare to the above method.

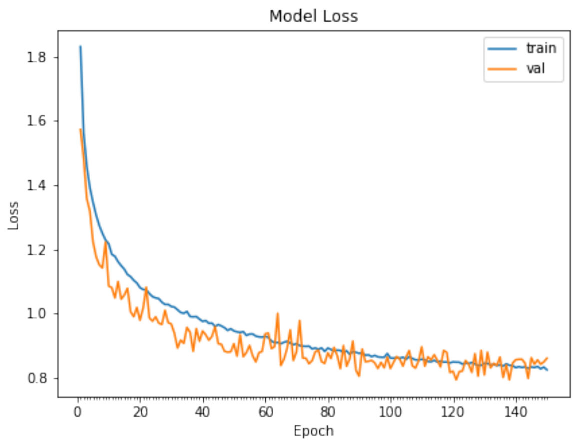
2. With a effective CNN design(like prefer a stack of small filter CONV layer to one large receptive field CONV layer), we can emphasize the advantage of using CNN: make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

The result also indicates that the Random Forest model has better performance than the Decision Tree. The reason is the Decision Tree is weak learner, it might be too much biased to the training set, and makes misclassification for a test set. Random forest combines a number of weak learner which can be seem as a strong learner and reduces the prediction variance.

4 Conclusion

In order to tackle the task of object recognition, we have developed and tested different algorithms including k-NN, Decision Tree, Random Forest and Convolutional Neural Networks (CNN). Considering the overall performances in terms of efficiency and accuracy, we agree on the conclusion that CNN model 2 is the optimal solution amongst other algorithms specifically for this problem. After training and tuning, the model can achieve a test accuracy of 70.4% .





During the process of data analysis and programming model construction, there are still several limitations and concerns for further improvements in the future research. Firstly, although k-NN algorithm does not perform well for this problem, it has the advantages of simplicity and low dependency on training and tuning model. Possible improvements over the basic k-NN in terms of accuracy could be weighted feature vector (e.g. the similarities between center pixel values are clearly more significant than edge pixels) or using artificially generated training set called bootstrapped training set, etc. Secondly, as we can see in the graph, if we run more iteration, there could be some further improvement. Also build a deeper CNN can help but will cost a longer runtime. So we choose CNN model 2 which has satisfying prediction accuracy and relatively acceptable runtime. Furthermore, in this project we are provided with 50,000 training images with equally distributed objects from 10 different class. However, in practice it highly possible to have limited number of training samples and unbalanced labeled training samples from different classes. Theses technical challenges should be considered and investigated in the future research.

References

[1] Kun-Che Lu, Don-Lin Yang, and Ming-Chuan Hung , *Decision Trees Based Image Data Mining and Its Application on Image Segmentation.*

[2] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, *Classification: Basic Concepts, Decision Trees, and Model Evaluation*, 2005.

[3] Benjamin Graham, *Fractional Max-Pooling*,2014.

[4] Stanford CS class CS231n, *Convolutional Neural Networks for Visual Recognition notes*