Image Classification on Caltech-256 Dataset

Springboard Data Science Capstone Report

Florian Sestak

April 2017

Table of contents

1 Introduction

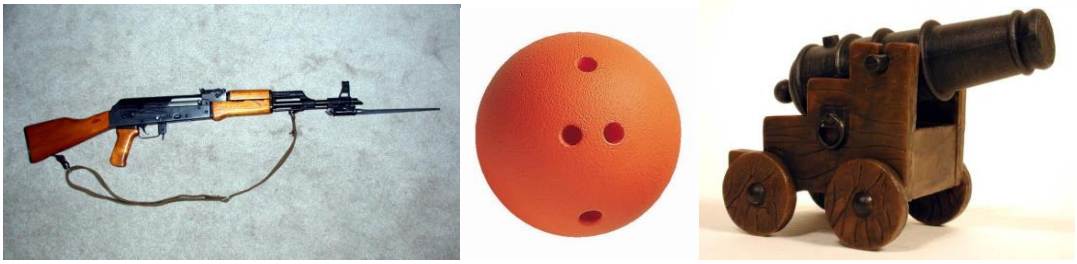
The aim of this document is to describe my Capstone Project of the Springboard Data Science Intensive Course. On the following pages I will go through the steps I took.

1.1 Dataset

The dataset I took was the Caltech-256 image dataset. It contains categories of 257 different images. The following table show a short overview about the data. The data is available at: http://www.vision.caltech.edu/Image_Datasets/Caltech256/.

Released	Categories	Pictures Total
2006	257	30608

The following pictures give an overview how the images in the dataset look.



1.2 Objective

The objective of this project is to build a model for image classification.

1.3 Approach

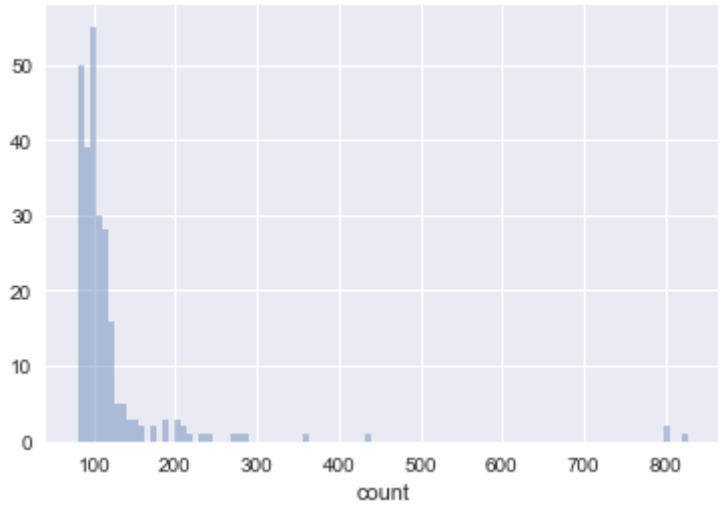
This section describes the approach I used for this dataset.

- Exploring the data
- Try different Machine Learning Models
- Compare the Machine Learning Models
- Conclusion

2 Exploratory Data Analysis

2.1 Distribution of different categories

The following graphic shows how the different samples are distributed. Below is a table with the exact statistics.



Mean	**Min**	**Max**
119	80	827

It shows that we have at least 80 images per category, but also that some categories contain, a very huge number of images.

2.2 Distribution of different image resolutions

Different Image Resolutions
15423

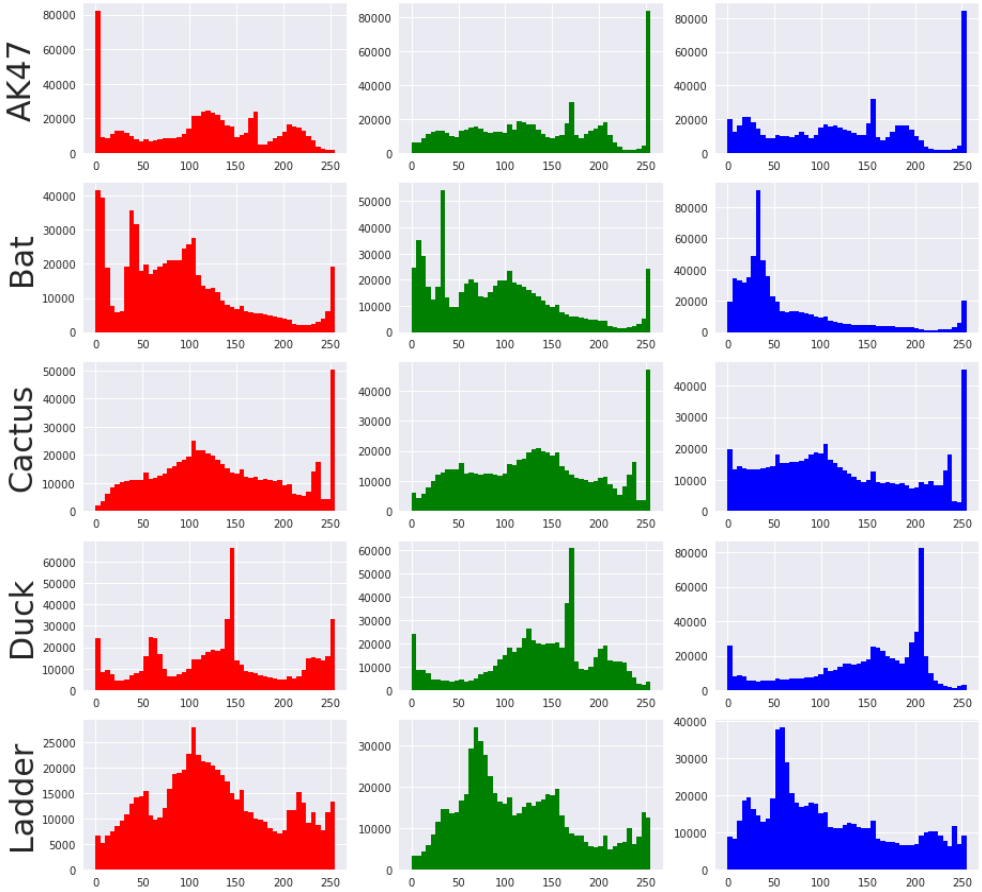
The dataset contains 15423 different resolutions for the images.

2.3 Distribution of RGB-Values over different categories

Due the fact that we have so many different image resolutions I did a resizing to the mean width and the mean height over all images.

Mean - Width	Mean - Height
382	345

After resizing and averaging over the rgb-values of 5 images for each category the distributions looks like in the following graphic.



An ideas was to classify the images over the distribution over the RBG values, but we will see in the problem section why this idea won't be very successful.

3 Problems

3.1 Images contain more objects than the object to classify

The following table shows to different categories. We can see people on both images but only one categories is the category people, this is really hard to classify, because also as

human you don’t know what you are looking for. A solution for this problem would be object detection and then classification but this is out of the scope of this project.

On the images below we can also see that there would be too much noise for classifying the images over the distribution of the RGB-values.

AK47	People
	

3.2 Objects are not centred in the images

Another big problem is that the objects we want to classify, are not centred in the image all the time. The following image contains the category billiards, but it contains the object twice, and they are not centred in anyway.

Billiards


3.3 Training Time

A big problem in Machine Learning is the training time of some classifiers. To tackle this problem, we setup an Ubuntu Instance in the Google Cloud with a Nvidia Geforce K80 GPU. This won’t help us for all classifiers but for some of them.

4 Machine Learning

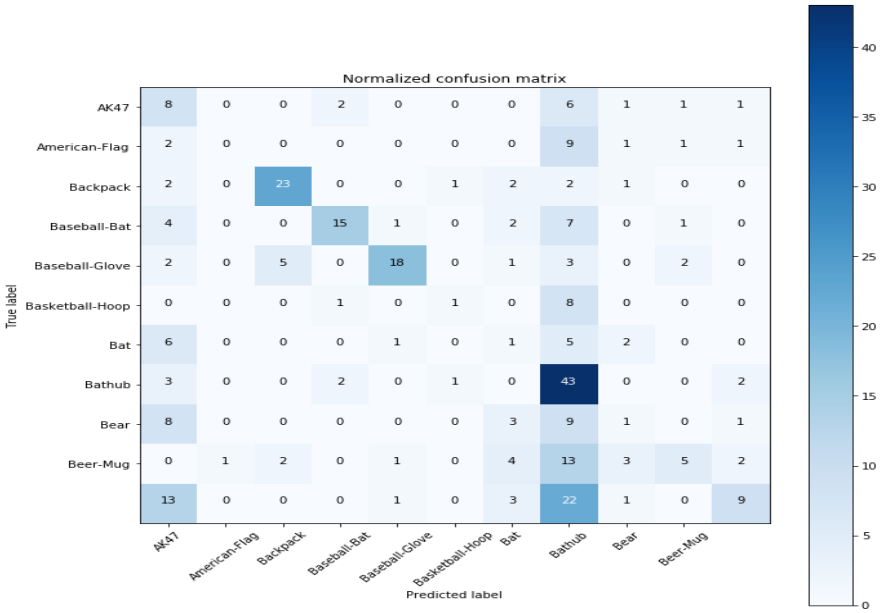
To reduce the time, we spend on trying different models the amount of training data is reduced to 10 classes. We will try different models on these 10 classes, later we will take the model with the best accuracy on the test data and train the model on the whole data. Due the fact that there are so many different image resolutions we will resize these 10 images to 150x150 pixels.

4.1 K-Nearest

The code for this model is provided in the "capstone-project/notebooks/3.2 K-Nearest.ipynb" notebook. To find the parameters we used GridSearch from sklearn. We come up with the following parameters:

Parameters	Accuracy Training Data	Accuracy Test Data
N-Neighbours: 10 Metric: Euclidean	0.40	0.42

We can see that the accuracy on test data is a little bit higher, if we would increase the amount data, I think the training accuracy will go over the test accuracy. It is not common but it can happen, a reason might be that we are only using 10 classes.
Below is a confusion matrix of the results.



4.2 Support Vector Machine

The code for this model is provided in the "capstone-project/notebooks/3.1 Support Vector Machine (SVM).ipynb" notebook.
To find the parameters I used GridSearch from sklearn. We come up with the following parameters:

Parameters		Accuracy Training Data	Accuracy Test Data
C: Gamma: Kernel:	0.01 Euclidean linear	0.19	0.16

The accuracy is near random guessing, and compared to K-Nearest worse.

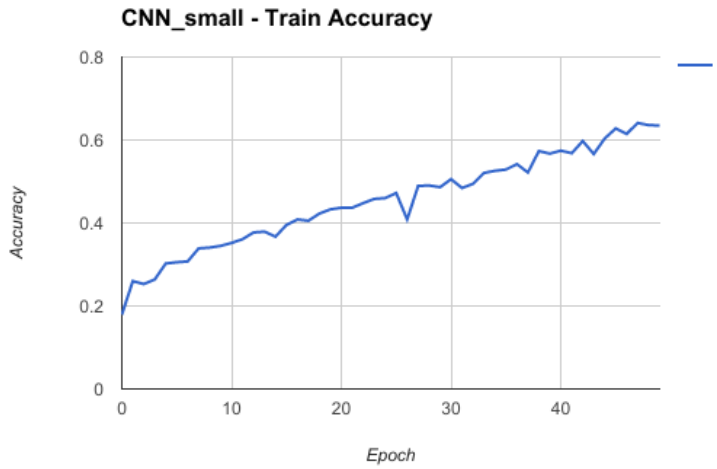
4.3 Convolutional Neural Network

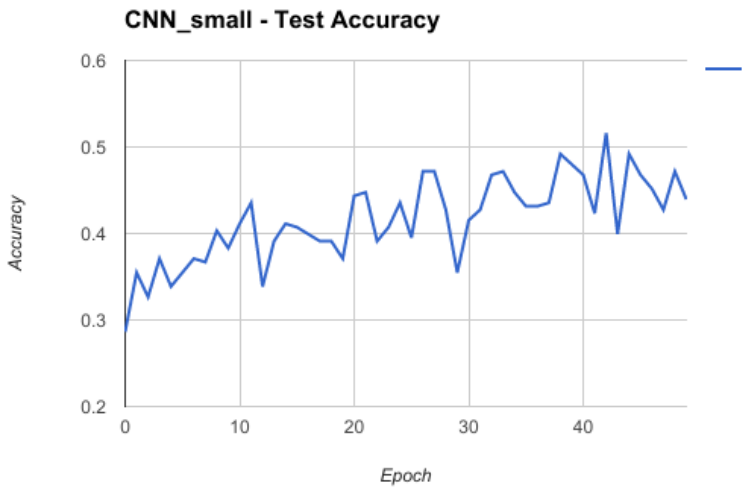
Convolutional neural networks are the state of the art approach for image classification.

4.3.1 CNN_small

The code for this architecture is in the "capstone-project/notebooks/3.3 CNN_small.ipynb" notebook. It loads the data from a npz file, the architecture of the network is based on the Cifar10 example (https://github.com/fchollet/keras/blob/master/examples/cifar10_cnn.py) provided in the Github Repository of Keras.

Below are the charts for the train and test accuracy. The data is taken from Tensorboard.





Train - Accuracy	Test - Accuracy
0.64	0.51

These results are not bad, the training time was lower than with the SVM because we didn't need to find the hyperparameters. But this is a very simple architecture, the training time increases if we increase the size of the architecture.

5 Comparison of Methods

In this section we will compare the different models we trained in the last section.

Library	Model	Train - Accuracy	Test - Accuracy
sklearn	K - Nearest	0.40	0.42
sklearn	SVM	0.19	0.16
Keras	CNN	0.64	0.51

In the table above we can see that the CNN outperforms the other models. So we will now use a CNN to train it on the whole dataset.

6 Train Final Model - Fine Tuning

In this section, we will train a CNN on the whole dataset.

Fine Tuning

Fine Tuning is a method where you take pretrained networks and change only some layers. The reason is, if we have pre-trained convolutional layers the layers near the input are for feature extraction (edges, corners, …) and training a full network is very time consuming and we don't need that if we have layers they do a good job in extracting features. So, we will use a network pretrained on the ImageNet dataset and change only the last layers.

Image Augmentation

Image augmentation is a technique to produce more data from the same dataset. I rotate, flips, shifts or zooms the image, it is very useful to prevent overfitting. I also rescaled the images to values between [0, 1] to improve training.

- rotation_range=40,
- width_shift_range=0.2,
- height_shift_range=0.2,
- rescale=1./255,
- shear_range=0.2,
- zoom_range=0.2,
- horizontal_flip=True,
- fill_mode='nearest'

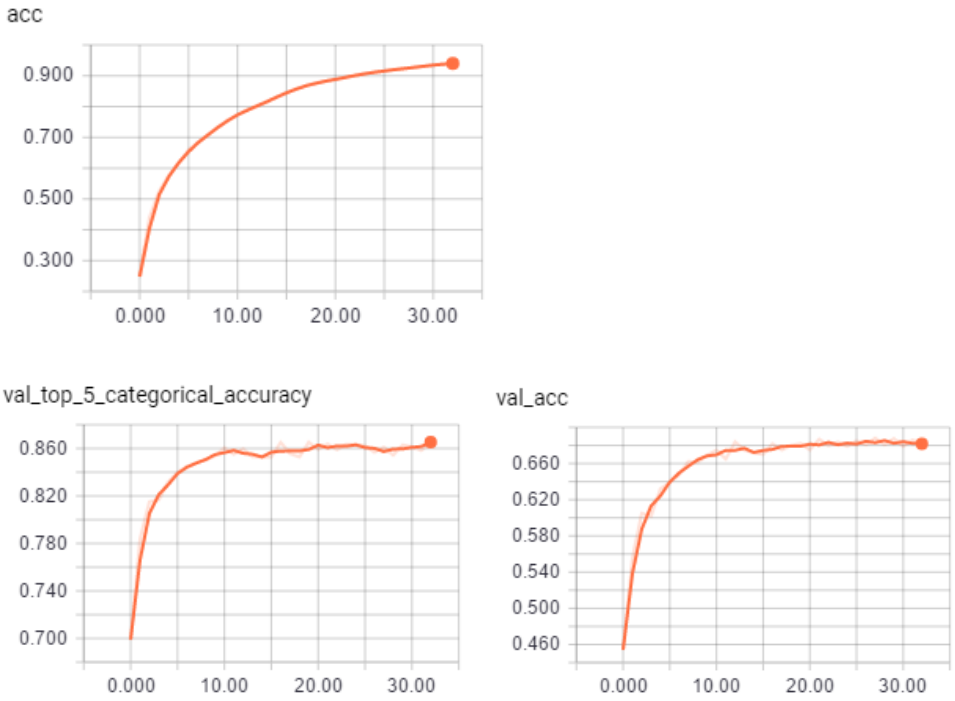
6.1 CNN - VGG16

The VGG16 - architecture is from the paper of Karen Simonyan & Andrew Zisserman (https://arxiv.org/pdf/1409.1556.pdf). We use a pre trained network on the ImageNet Dataset and replaced the last fully connected layers. We freezed the convolutional layers and trained only the last three fully connected layers. Keras makes it easy to do that. The code for this model is provided in "capstone-project/notebooks/3.4 CNN - VGG16.ipynb".

Below is a table for the accuracy, the validation accuracy, the top 5 categories validation accuracy and the graphs of training.after training for 32 epochs

Accuracy:	0.93
Validation Accuracy	0.68

Top 5 Validation Accuracy	0.87



The above table shows that there is a about 20% accuracy difference between training and testing data, so it seems that the network started overfitting. We should consider using more data or to do more data augmentation on the images, to avoid overfitting. But nice to see is that our top-5-categorie accuracy on the validation data is near 90 %.

7 Conclusion and Recommendations

It's nice to see that we could beat the benchmark provided by Caltech256, but we also have to say this benchmark is from 2006 and they used a SVM. They had an accuracy of 35% compared to our model with 68% accuracy on the validation data.

We also saw that CNN's outperformed the other models, I think we could also improve the model if we would have cleaner data, because as we mentioned in the problems section some things are hard to classify also for humans if objects are not centred in the images.

Also, important to mention is that with Keras, it's very easy to build such networks. You don't need to think about where you get pretrained weights or the VGG16 models, Keras contains all of that.

The project showed that CNNs are good performing in image classification, also on not so clean images, but the challenging thing is the training time, one epoch did

take about 10 minutes on a very fast GPU. I think a big problem was that the whole data did not fit in memory so we had to load every batches from hard disk this is very time consuming.