LEARNING IMAGE CLASSIFICATION

MACHINE LEARNING(CS 6375) FINAL PROJECT

04-30-2017

UNIVERSITY OF TEXAS AT DALLAS

Teammates:
KwangHoon An (kxa162330),
Arijeet Roy (axr165030),
Poonam Purushottam Pathak (ppp160130)

ABSTRACT

The goal of our project is to quickly and reliably build a training model in image classification problem. Image classification is the task of taking an input image and outputting a class or a probability of classes that best describes the image. Finally, as an output we will have a different set of images and labels which we will pass through the CNN and compare our output against ground reality objects. Our task is to develop an algorithm to classify images of dogs and Cats, which is the Dogs vs. Cats competition from Kaggle. We mainly investigated two approaches to address this problem.

1. The first one is a traditional pattern recognition model using Support Vector Machines where we extracted some human-crafted features like color and Dense-SIFT, represented images using bag of words model, and then trained Support Vector Machines classifiers. Using this approach, we got an accuracy of around 43%.
2. For the second approach, we used Deep Convolutional Neural Networks to learn features of images and trained Backpropagation (BP) Neural Networks for classification. Using the second approach we got an accuracy of around 77%

Data set:
https://www.kaggle.com/c/dogs-vs-cats

TABLE OF CONTENTS

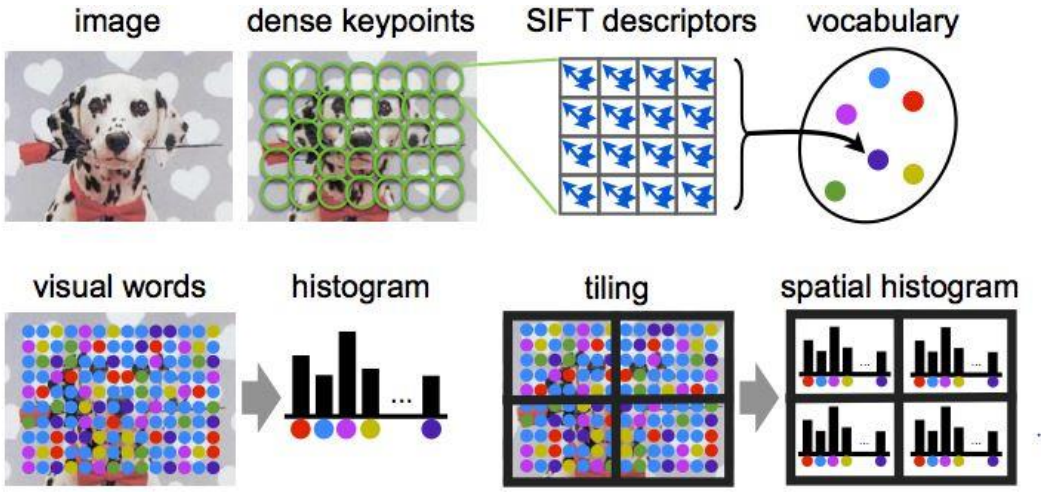
INTRODUCTION

The goal of the project is to build a Cat/Dog image classifier which is capable of correctly labelling or identifying any given new image either "dog" or "cat".
Our project explores two approaches to implement the classifier. One approach involves using Support Vector Machines (SVM) as part of the traditional machine learning algorithms and second approach involves using convolutional neural network (CNN) as part of deep learning algorithms. The training phase uses a Kaggle dataset. The dataset comprised of 20,000 images of dogs and cats.

The main difference between traditional machine learning and deep learning algorithms is in the feature engineering. In traditional machine learning algorithms, we need to hand-craft the features in the form of SIFT. SIFT(Scale-invariant Feature Transform) is part of computer vision used to detect local features, in this case like ears, eyes of cats and dog etc. By contrast, in deep learning algorithms feature engineering is done automatically by the algorithm.
We just need to input the images for classification. Though automatic feature engineering is difficult, time-consuming and requires domain expertise. The results are much more accurate compared to traditional machine learning with less or no feature engineering.

Using Support Vector Machine for Image classification:

Support vector machines are proved to perform well in pattern recognition and classification. The same logic is applied for image classification. The difference here lies in the pre-processing of image before feeding it to the SVM. Pre-processing of data involves feature extraction done via SIFT features computed on a regular grid across an image. The SIFT descriptor vectors computed are quantized into visual words. The frequency of each visual word is then recorded in a histogram for each tile of a spatial tiling (Please see the image below). The final feature vector for the image is a set of these histograms.

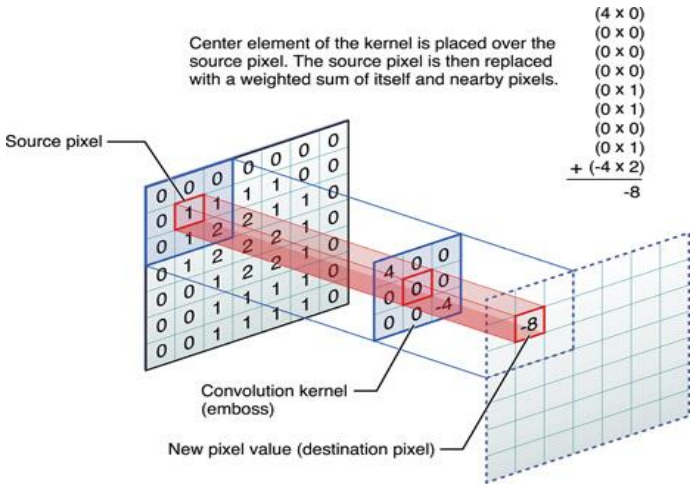


The histogram set is then fed to the SVM. The implementation is done in Python using Sklearn library which incorporates SIFT feature package and linear SVM classifier model specifically implemented for Image classification.

Using CNN and Backpropagation Neural Network (Deep Learning)

Deep learning is the new big trend in machine learning. It refers to a class of artificial neural networks (ANNs) composed of many processing layers. Convolutional neural networks are a special type of feed-forward ANN's. These models are designed to emulate the behaviour of a visual cortex. CNNs perform very well on visual recognition tasks. CNNs have special layers called convolutional layers and pooling layers that allow the network to encode and compress certain images properties.

Convolution Layer: This layer consists of a set of learnable filters that is parsed over the image spatially, computing dot products between the entries of the filter and the input image. The filters should extend to the full depth of the input image. For example, A filter of size 5x5 should have depth 3 i.e. (5x5x3) to be applied to a coloured image of size 32x32 to cover RGB colour channels (Red, Green, Blue) of the image. These filters will activate when they see same specific structure in the images.



Pooling: Pooling is a form of non-linear down-sampling. The goal of the pooling layer is to eventually compress the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. There are several functions to implement pooling among which max pooling is the most common one. Pooling is often applied with filters of size 2x2 applied with a stride of 2 at every depth slice. A pooling layer of size 2x2 with stride of 2 shrinks the input image to a 1/2 of its original size.

We have implemented 4 CNN layers (each consisting convolution as well as pooling layer) which output final parameters.

These parameters are then fed to the backpropagation neural network whose output is classified by final layer Softmax classifier in terms of probabilities for each class label.

TRAINING WITH KAGGLE DATASET:

2. DataSet

Recently CNN is getting popular for being robust to noise and that is why preffered to other methods in speech recognition field. In order to test its robustness, we used three different datasets. First picture is original input data without any noise. Second is dead pixel image where we randomly kill around 3000 pixels by setting pixel to zero. Third is Gaussian noisy image where we added Gaussian Noise by adding normalized mean value to the image.

We only used 20000 images for training and used 5000 images for test.

MODEL

3. Approach 1. CNN

Our second approach is to learn features and train using Convolutional Neural Network and use Softmax function as a clssifier. As supposed to first approach, CNN learns features from images

3.1 Deep Convolutional Neural Networks

The CNN is one kind of deep neural network widely used in image classification and it shows very superior performance in such a problem.

In structure manner, CNN and Vanila NN both has a multiple input, hidden layers and one output layer. What CNN distinguishes from Vanila NN is it consists of Convolutional Layers and Max-Pooling layers

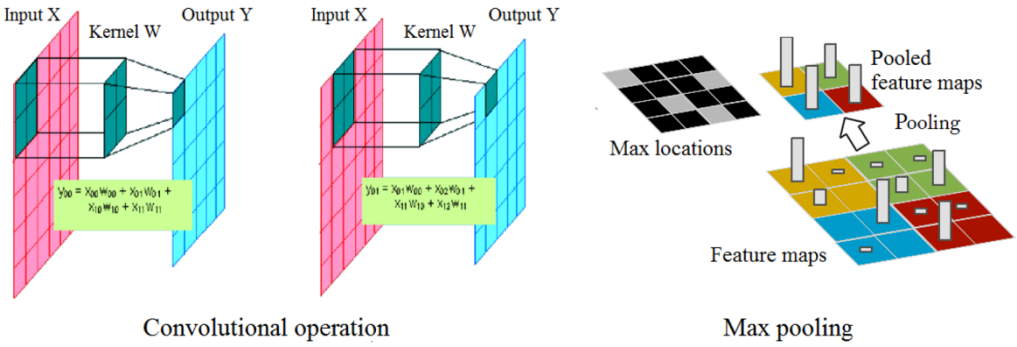


Figure 1 Convolutional Operation and Max Pooling in CNN

In our project, CNN contains many feature maps, which are three dimensional hidden neurons. Feature map consists of two dimensional neurons that each single neuron is an output of convolutional operation – element wise multiplication and sum and pass it to the activation function. Every CNN layer has an its own weigted matrix called either kernel or filter. Filter will be map to input image and convolves around with defined stride size, which is single slide unit in the receiptive field. In turn, filter will result in feature map in CNN layer. Advantage of using CNN is CNN has less parameters compared to Vanila NN as each layers share its own filters. Hence, the number of parameter is not increased expoentnailly by adding more hideen layers.

3.1.1 Rectifier Linear Unit

In our project, we used Rectifier (ReLu) as an activation function. It has been used in convolutional network more effectively than sigmoid function and has outperformed other activation function. Most noticeable superior performance of ReLu is ReLu solves vanishing gradient problem.

$$f(x) = \max(0, x)$$

Figure 2 ReLu function

As we can infer from the formula above, ouput of activation function will not be constrained to be lowerbounded/upperbounded. In sigmoid function, as input increases or decreases, the output may tend to saturate to either 1 or zero and it will cause the vanishing gradient problem while backpropagating.

3.1.2 Drop Out

In fully connected layer, we used drop out as regularization method. Drop ouot prevents overfitting by randomly killing neurons as per the drop out rate we set, so that they do not contribute to next iteration.

3.2 Our Model

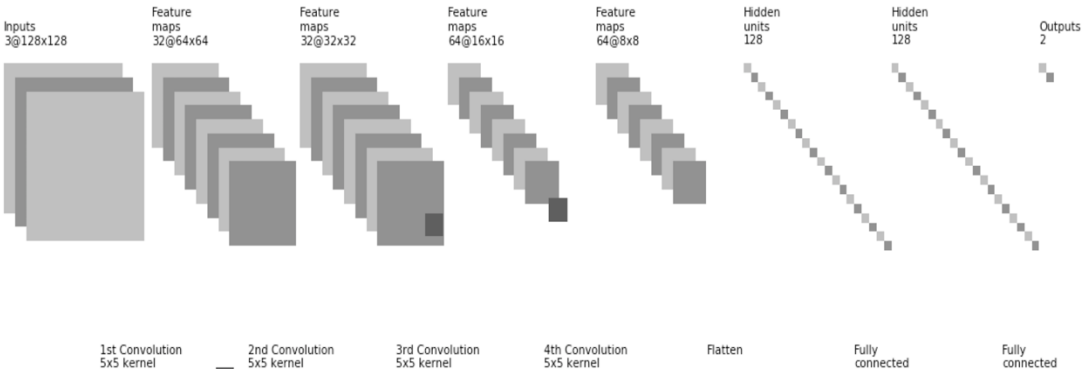


Figure 3 Illustration of the architecture of our CNN model

Our architecture is illustrated by Figure 3, we normalize datasets into 128*128 size and use them as input data. We use total four layers CNN as hidden layers and two fully-connected layers at the last and use softmax function as a classifier. In the first CNN layer, it filters 128*128*3 size images with 5*5 size 32 width filters and add zero padding to keep original input shape. Second layer takes input pooled output of first layer and do same operations as first layer, then pass to next layer. At the last two layer, we flatten multi-dimensional feature maps prior to classify the label of input image. The output from the CNN and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset. Apart from classification, adding a fully-connected layer is also a cheap way of learning non-linear combinations of these features. Most of the features from convolutional and pooling layers may be good for classification task, but combinations fo those features might be even better.

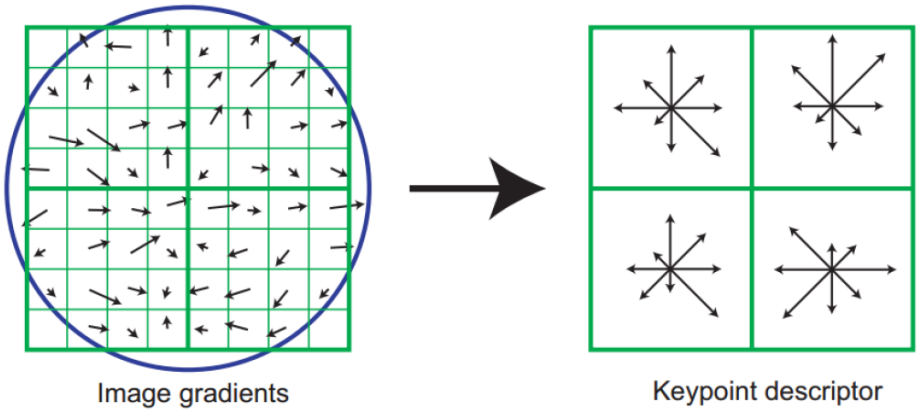
The sum of output probabilities from the fully connected layer is 1. This is ensured by using the Softmax as the classifier in the output layer of the fully connected layer. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one.

In each layer, we have 832, 832, 1664, 1664, 212992, 16384 and 256 training parameters respectively. We can see here the number of parameters increased at the fully connected layer.

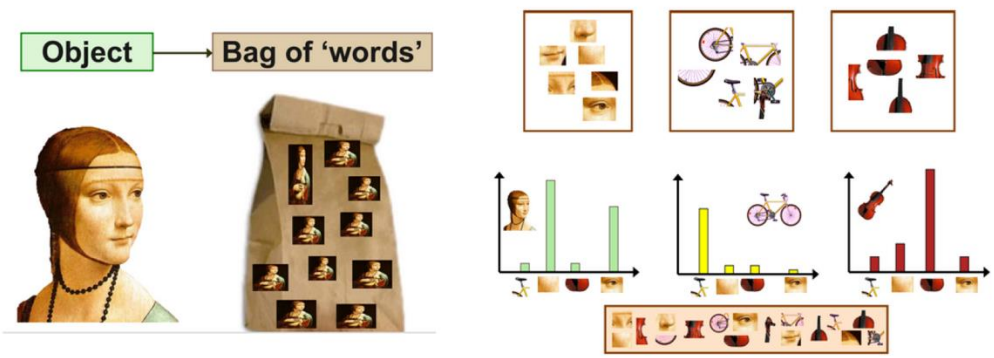
Approach 2. Using Support Vector Machine

In typical image classification problems, we choose some fixed human-crafted features to use. There are many well studied features, such as SIFT, RGB or HSV color features, etc. In our project, the images are either dogs or cats. And we know that the shape and the priori probability of colors of dogs and cats are different. So, we extracted local

features descriptor SIFT and HSV color features to represent the original images. The SIFT features are local and based on the appearance of the object, also invariant to image scale and rotations. The scale-invariant feature transform of a neighborhood is a 128 dimensional vector of histograms of image gradients. The region, at the appropriate scale and orientation, is divided into a 4*4 square grid, each cell of which yields a histogram with 8 orientation bins. The Dense-SIFT is a fast algorithm for the calculation of a large number of SIFT descriptors of densely sampled features. After getting the Dense-SIFT features, we applied the bag of words model to represent images. Figure below shows the bag of words model. It is a simplifying representation used in natural language processing and computer vision. In this model, a dictionary is formed by clustering the extracted features of training set using k-means algorithm. Every cluster is a word of this visual dictionary. Then images are represented by frequency vectors in which every dimension represents the proportion of features belong to a cluster. SVMs with suitable parameters have the ability to prevent overfitting, and experience shows that they usually can get good performance for classification.



SIFT – Descriptor



OBSERVATION

Approach 1. CNN

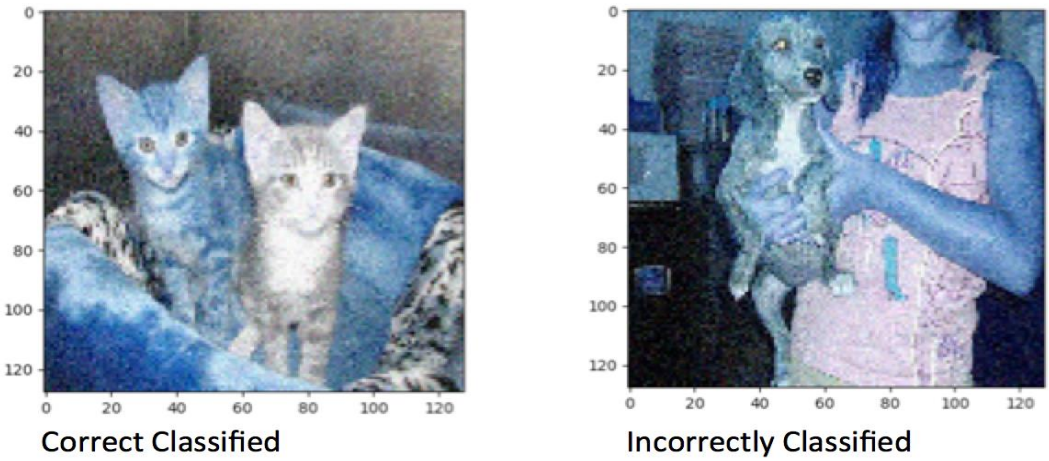


Figure 4 Example of CNN on test data

Correctly classified and incorrrectly classified by CNN is illustrated in figure 4. Intuitively we can observe correctly classified image is likely to be standalone image of cat/dog. Whereas, misclassified image contains more other objects that are likely to add noise in classification.

```
ML Project — -bash — 130×44
ARIJEETs-MacBook-Pro:ML Project DrunkenRex1107$ python3 catdog.py
Reading training images
Loading dogs files (Index: 0)
Loading cats files (Index: 1)
Size of:
- Training-set:         20000
- Validation-set:       5000
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but these a
re available on your machine and could speed up CPU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but these a
re available on your machine and could speed up CPU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but these are
available on your machine and could speed up CPU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX2 instructions, but these are
 available on your machine and could speed up CPU computations.
W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use FMA instructions, but these are
available on your machine and could speed up CPU computations.
Optimization Iteration:      1, Training Accuracy:  50.8%
Optimization Iteration:    101, Training Accuracy:  62.8%
Optimization Iteration:    201, Training Accuracy:  69.4%
Optimization Iteration:    301, Training Accuracy:  73.0%
Optimization Iteration:    401, Training Accuracy:  78.0%
Optimization Iteration:    501, Training Accuracy:  78.4%
Optimization Iteration:    601, Training Accuracy:  81.0%
Time usage: 10:03:52
(500, 128, 128, 3)
(500, 128, 128, 3)
(500, 128, 128, 3)
(500, 128, 128, 3)
(500, 128, 128, 3)
(500, 128, 128, 3)
(500, 128, 128, 3)
(500, 128, 128, 3)
(500, 128, 128, 3)
(500, 128, 128, 3)
Accuracy on Test-Set: 78.5% (3924 / 5000)
Example errors:
plot_images called
Correct examples
plot_images called
Confusion Matrix:
[[2060  413]
 [ 663 1864]]
ARIJEETs-MacBook-Pro:ML Project DrunkenRex1107$
```

CNN Test Accuracy on dataset after 700 iterations

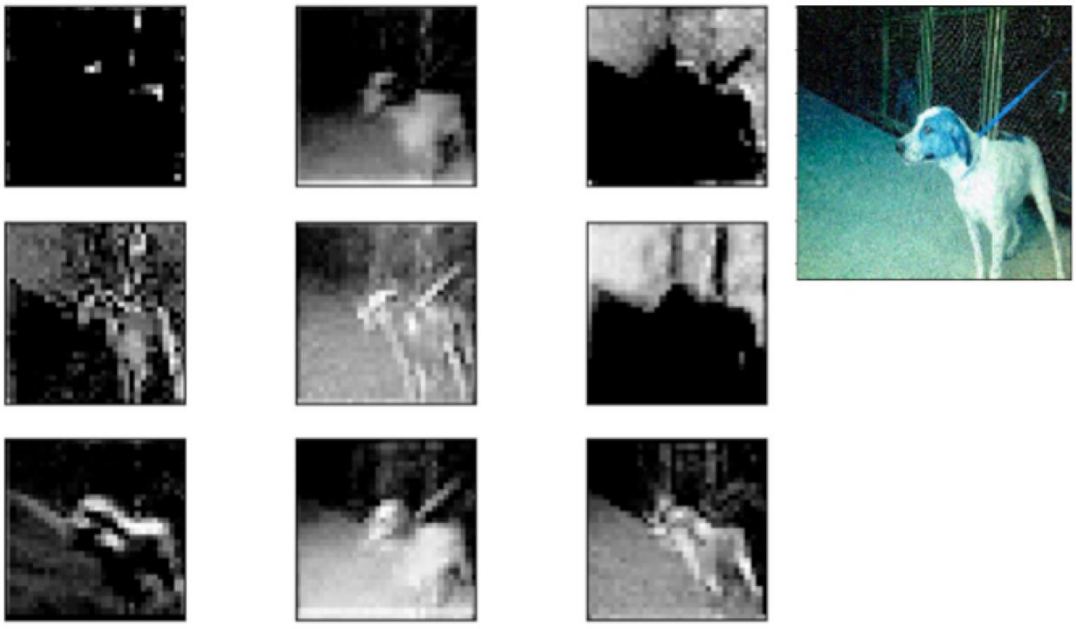


Figure 5 Example of Feature Map in Second CNN Layer

Above 9-grid pictures is the feature map of second CNN layer. Here, we can see how feature map detects and compacts specific image information individually. For example image in the top right corner of 9-grid contains the background of the image extracted, likewise every feature map has compressed information of input image data. Clearly visualizing the objects in these maps is still a challenge.

Approach 2. Support Vector Machines

When only using the Dense-SIFT features, the accuracy on the training dataset was 45% and the test dataset was only 48%. The possible reason for this low-test accuracy is that the Dense-SIFT and color features of dogs and cats are quite similar. For instance, they both have one head, one tail and four legs. Also, their color has much in common. To further improve the performance, we could try to extract more distinctive features, use PCA to reduce dimension or try Deep Neural Network to extract features of images like we did in the approach above.

COMPARISON:

Data Type	Classifier	Parameter Settings	Accuracy
Original	CNN	4 Layers, Dropout = 0.5, 700 Epoch	78.5%
Gaussian Noise	CNN	4 Layers, Dropout = 0.5, 900 Epoch	77%
Dead Pixel	CNN	4 Layers, Dropout = 0.5, 900 Epoch	78.2%
Original	SVM	Linear SVM, Square hinged loss, Gaussian smoothing	45.46%
Dead Pixel	SVM	Linear SVM, Square hinged loss	44.6%
Dead Pixel	SVM	Linear SVM, Square hinged loss, Gaussian smoothing	48.26%
Gaussian Noise	SVM	Linear SVM, Square hinged loss	44.18%
Gaussian Noise	SVM	Linear SVM, Square hinged loss, Gaussian smoothing	47.5%

CONCLUSIONS

Following is the conclusion as per the above observation of our experiment

1. CNN performs better in terms of accuracy in image classification problem.
2. CNN does not require flattening input image as well as feature engineering as opposed to SVM approach
3. However, CNN requires high computing power, this is the reason we only used 25000 images as part of data

   Overall, Deep learning approach seems to be preferred in image classification problem provided we have a GPU or a proper machine

REFERENCES

https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/02_Convolutional_Neural_Network.ipynb

http://cs231n.stanford.edu/index.html