DS 200 Graduate Project Option 2

Image Classification

Abstract

This project is to do image classification. The task is to generate useful features from the images and design classifiers based on these features.

In this project, I design both scaler features and matrix features from various prospectives, including pixel intensity as well as object's shape.

Based on the features, I tried to use logistic regression, KNN, decision tree, random forest, SVM as classifier. By cross validation and grid search, I fine tune my random forest model's accuracy on validation set to 43%.

Apart from the above methods, I also design a CNN network. By adjusting the network structure and learning rate, I reach an accuracy for about 48% on validation set.

Project description

This is the option 2 of graduate student project for DS 200. This image classification project is divided into four parts: Data input, Exploratory data analysis and feature extraction, Classifier training and assessment, Neural networks(optional).

In data input part, the task is to read in the data from pictures and make preprocessing on the raw data to create the dataframe used in later parts.

In exploratory data analysis and feature extraction part, the task is to select 15 useful features based on the exploratory analysis on data. This part is the core for the whole project since the model's performance is largely decided by the features you select.

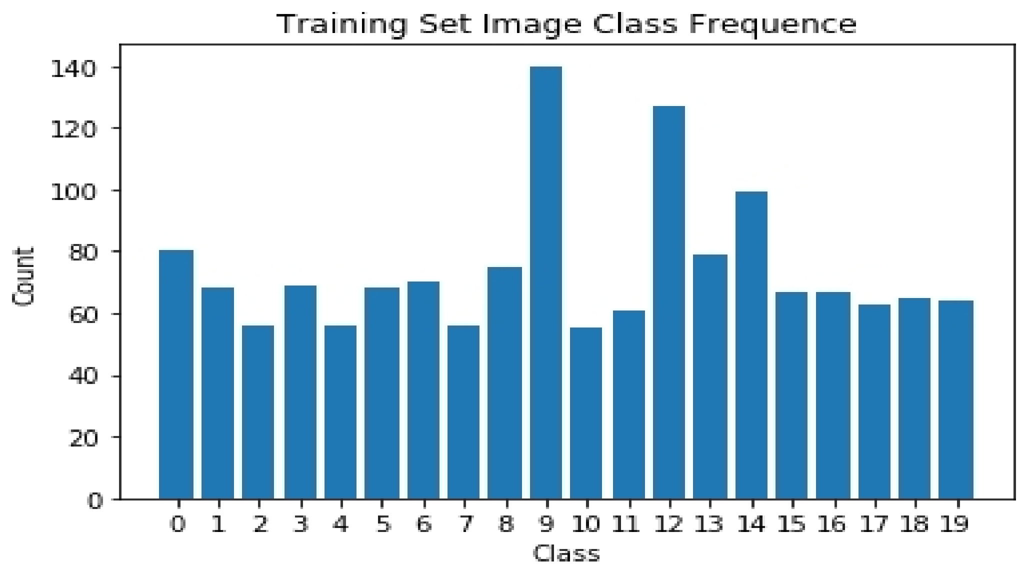
In classifier training and assessment part, the task is to try different classifiers and fine tune the model to get higher accuracy.

The neural networks part is optional, the task is to build a neural network to get a better prediction for the data.

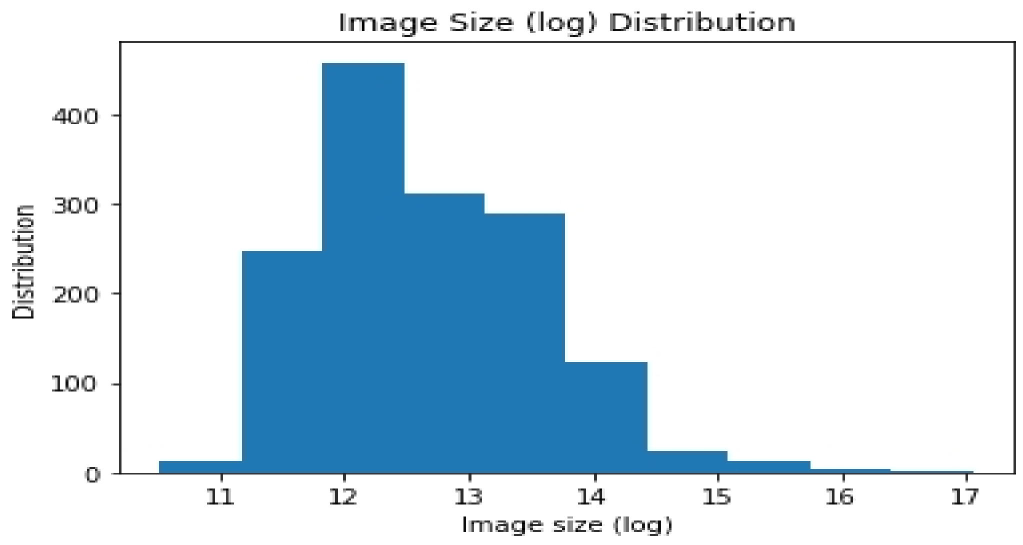
Data description

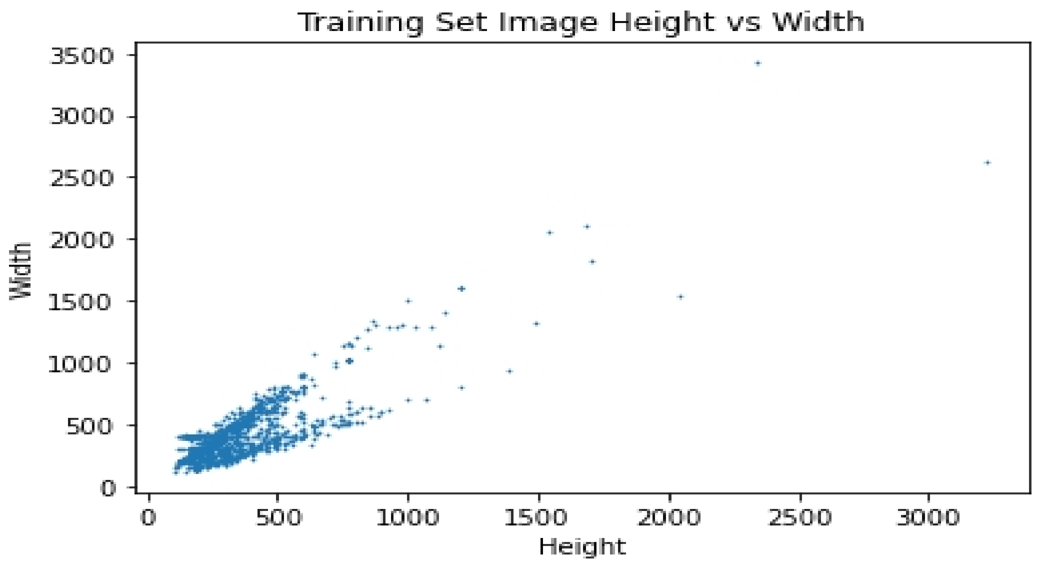
The dataset is divided into training set and validation set. There are totally 1501 images in the training set. Totally, there are 20 kinds of images.

The dataset is not equally distributed among different kinds of images.



Also, the image in these dataset is not the same size.





Apart from that, the image in the dataset is not totally composed of rgb three-channel image. In fact, some of them is one-channel gray image. To make the data in dataframe consistent, I convert these images into rgb images. The detailed code can be seen in PART1.ipynb.

Exploratory data analysis and feature extraction

I design a total of 16 features. The following passage is about how I find these features and why these features work.

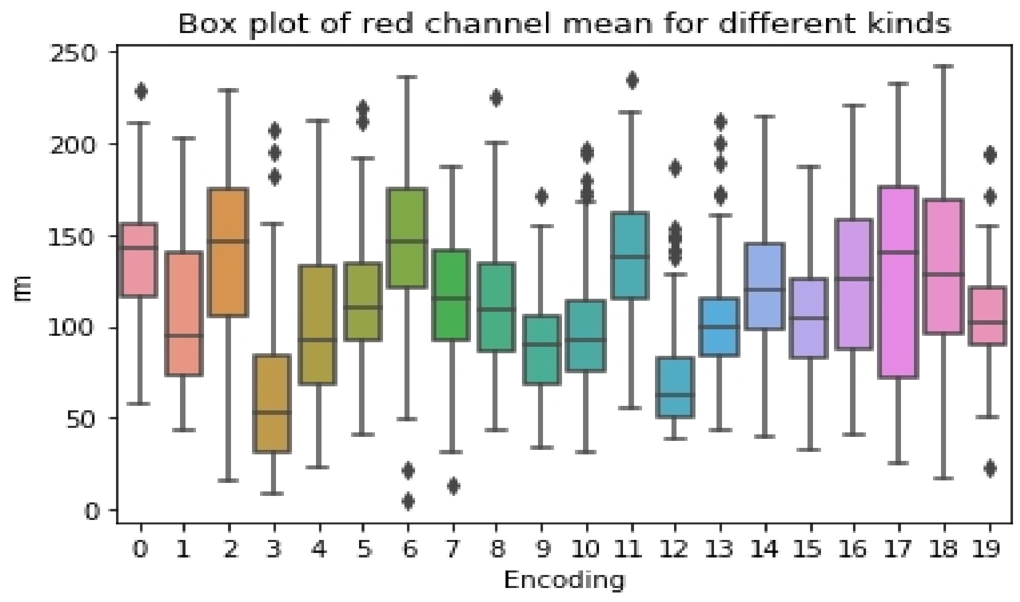
Image shape feature

Image shape is certain a kind of feature and is provided as an example in feature set. From pictures in data description part, we can see that the pictures vary in image size and height-width ratio. So, the size of picture and height-width ratio of picture are necessary as features since different size or height-width ratio can have influence on the picture's other features, even though the picture reflects the same object.

Color pixel feature

Color pixel feature is also certain a kind of feature. Since the pictures are rgb images, extract information on each channel is very natural. In the example, the teacher provides the mean intensity of red channel as one feature. Based on this, I also add mean intensity of green and blue channels as well as the standard variance of intensity of each channel as features. Apart from that, I also use the histgram of each channel as matrix features.

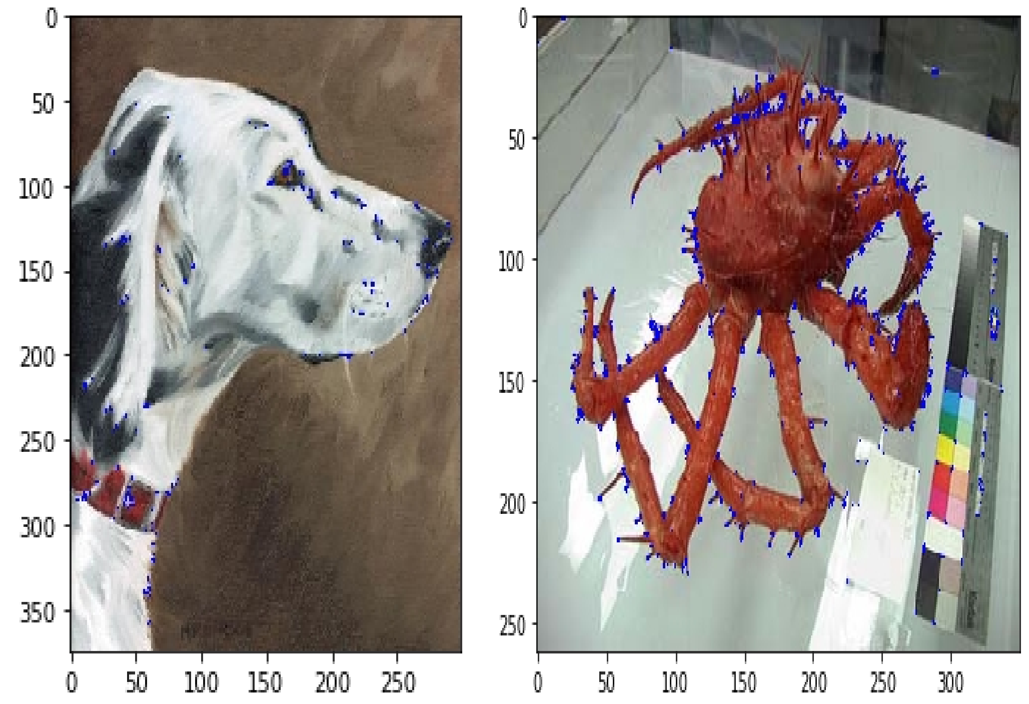
In PART2.ipynb, I use box plot to compare these features for different kinds. Detailed code and all six plots for scaler color pixel feature can be found there. From the plot, these features do vary a lot among different kinds.



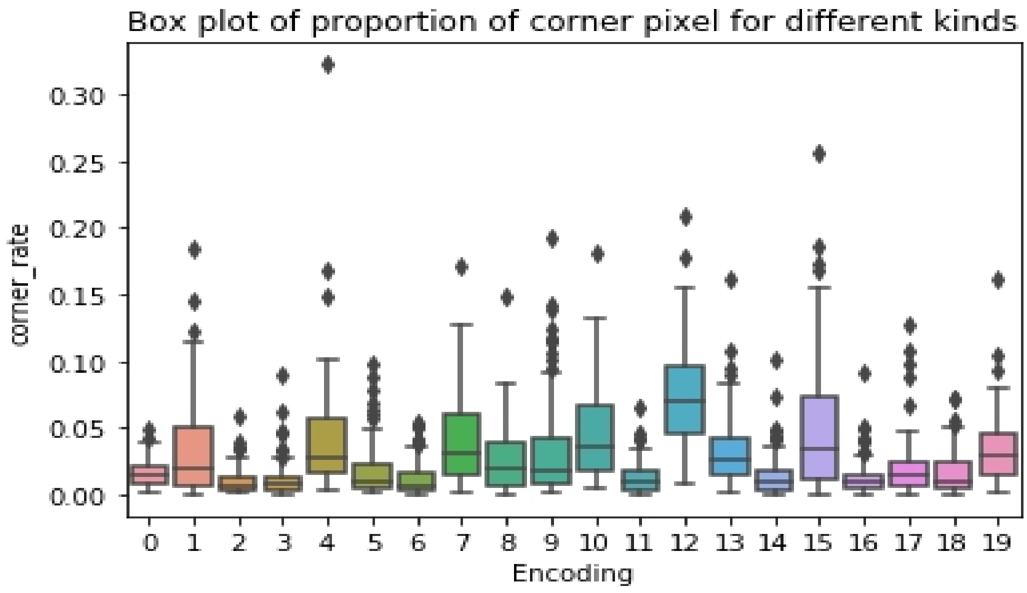
Corner feature

Corner feature refers to features generated by HarrisCorner method. From my point of view, the features mentioned above does not take the object's shape into consideration. So, adding some features about shape may be helpful.

To get an easy-generated shape feature, I think of HarrisCorner method. Since some object has more corners(for example, crab) than other objects. Use this feature may distinguish these kinds of objects well. Below is the comparison of corner information for dog and crab. The corners in the picture are set to completely blue color.



Based on the observation, I generate two features from HarrisCorner method. First, I use the corner rate as a sclaer feature, which is computed as the number of corners divided by the size of picture. The comparison of corner rate among different kinds can be seen below.

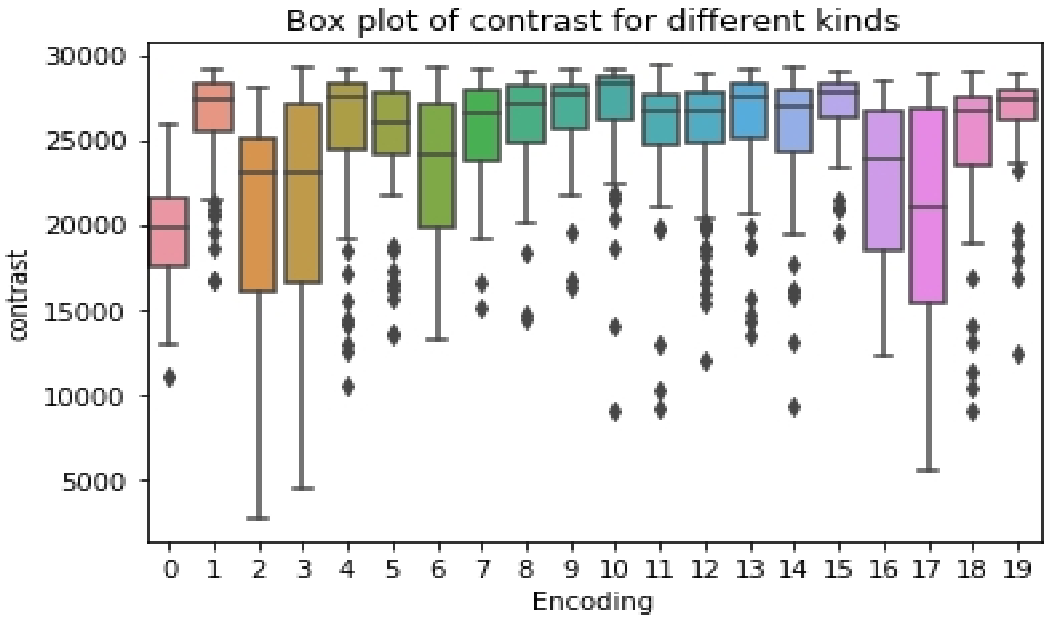


Apart from scaler feature, I also think where most of the corners lie can be a meaningful feature since the corner distribution for different objects is different. So, I divided the picture into 4*4 areas and make a matrix feature to record the corner rate in each area.

Contrast feature

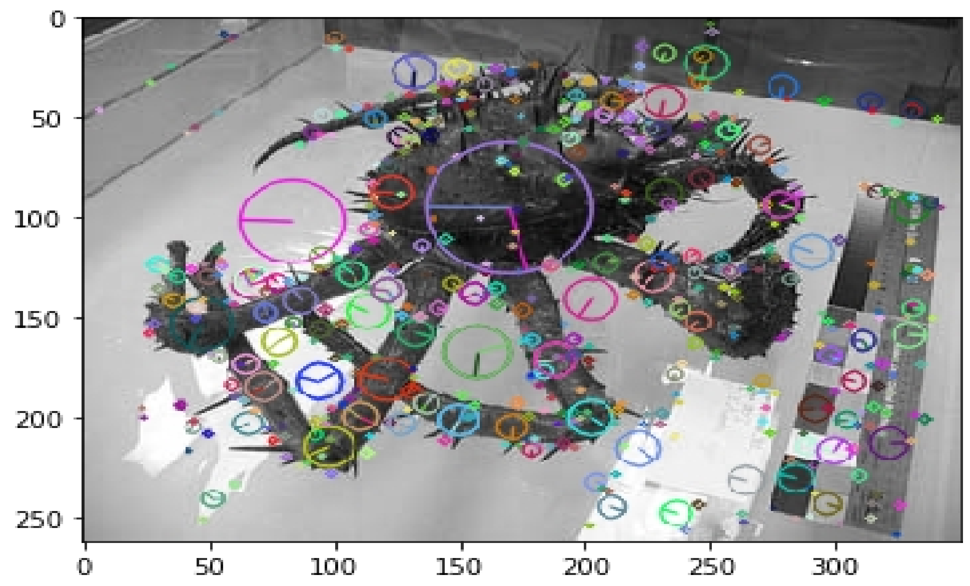
The aboved features have taken the picture's size into consideration. So, taking the object's size into consideration may also be helpful. This idea generates from the comparison between teddy bear and bear. Teddy bear is actually the toy model of bear. So, to some extent, they can be very similar in color and shape. The most significant difference between them is the size of object.

To reflect the size of the object, I use the contrast of picture as a feature. The contrast of image reflects the level of texture detail of the image (detailed caculation can be seen in PART2.ipynb). So, if we resize the image to the same size, the contrast can to some extent reflect the object's size in the image. Below is the comparison of contrast among different kinds.



Sift feature

Sift is a very useful and popular method in image processing. It can extract the keypoint from image. And each of the key point is represented by a vector called descriptor. Below is the keypoint extracted by sift.



However, there are too many keypoints. Considering the little dataset, using all of them as features will only result a mess. So, I use the bag of words technique to do k-means clusters on them. After that, I use the frequency of keypoints of the image belongs to each cluster as the feature. Detailed implementation can be seen in PART2.ipynb.

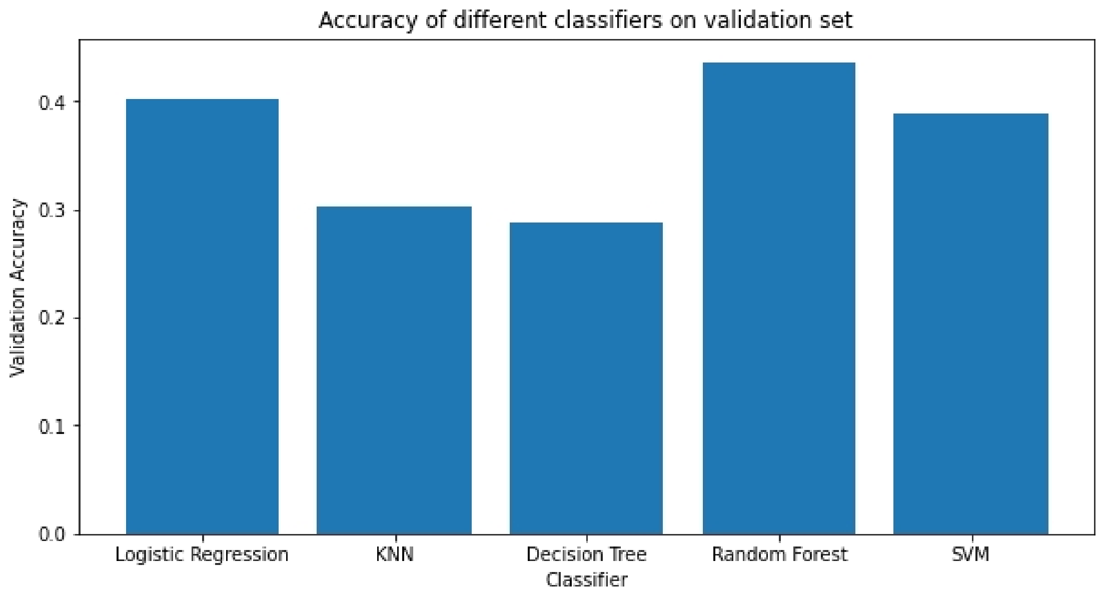
Among all these features, the most interesting features I came across are corner features, contrast and sift features. All these features are very powerful features. For corner features and contrast, as said before, I discovered them by coincidence, especially the use of 4*4 corner-rate matrix, which borrows some ideas from CNN and I just give it a try.

However, there are also some features that are useless in practice. For instance, the hist of three channel. I used to think them as useful since they reflect many information about the picture. However, in actual practice, they are useless. One reason maybe that the mean and standard variance of three channel have already reflect picture’s feature from pixel intensity prospective. Another reason maybe that the hist is a too long vector that our limited data cannot make the classifier learn it well.

Classifier

Non neural network classifier

For non neural network classifiers, I tried to use logistic regression, KNN, decision tree, random forest, SVM as classifier. Among them, random forest and logistic regression perform best. Below is their accuracy on validation set.(Here, I didn't fine tune the parameters, since this is just to select the most suitable model, a direct but simple view is enough.)



Then, among logistic regression and random forest, I choose random forest to do further improvement. There are two reasons. First, it's accuracy is slightly higher than logistic regression. What's more, random forest has more parameters and are more flexible, which means there is a larger range to fine tune the parameters and get better result.

After that, based on the random forest model, I use grid search and cross validation to fine tune the parameters. The best parameters are found as shown:

```
RandomForestClassifier(max_depth=20, min_samples_leaf=2, n_estimato
rs=300,oob_score=True)
```

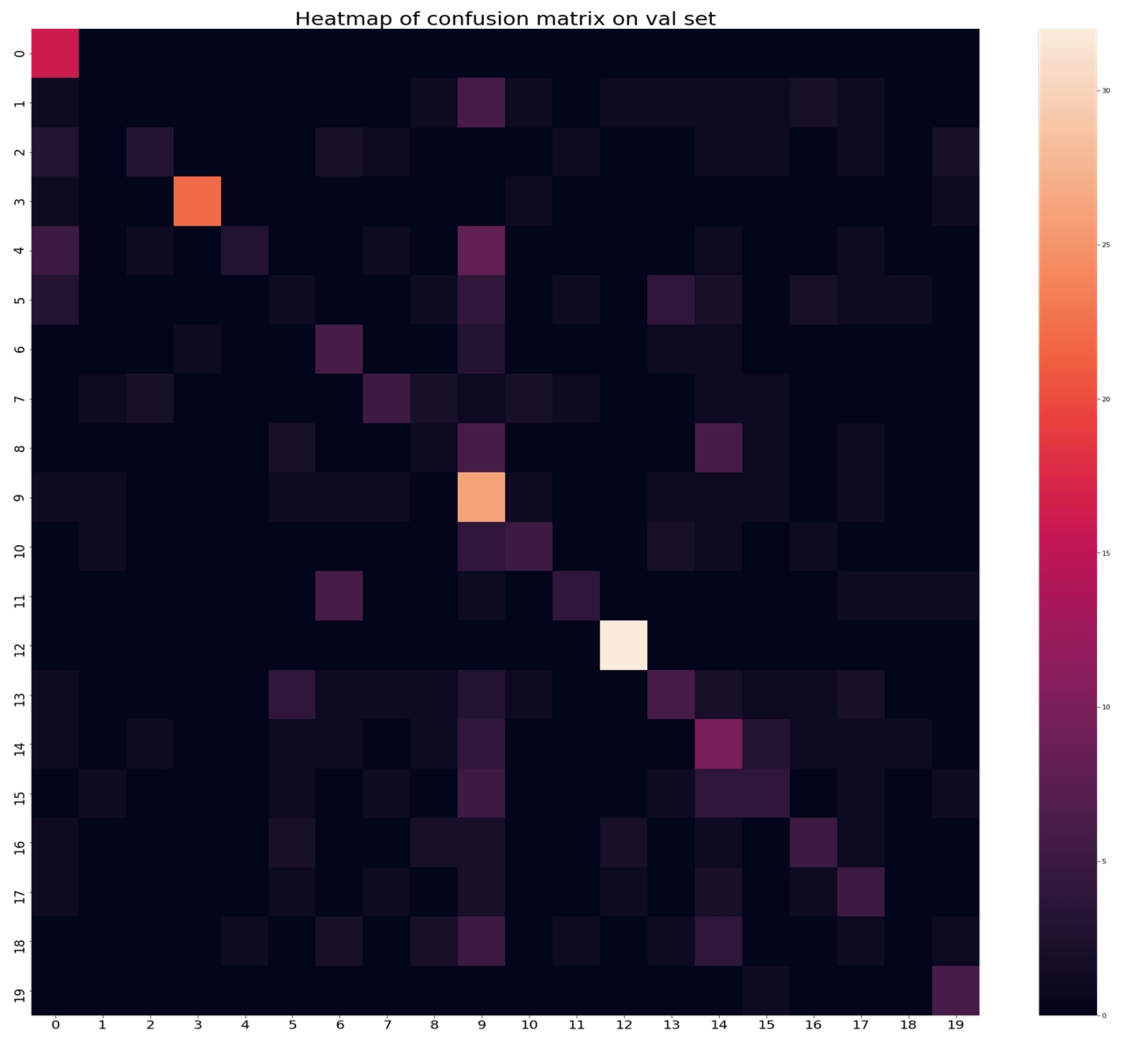
Finally, the random forest model can get about 43% accuracy on validation set, which significantly outperforms other models. This is not suprising for me and I think there are two reasons.

First, the features here are not equally-important and pictures with same label can have very different feature due to different image size or height-width ratio. For KNN, the underlying assumption that the nearest point in feature space is the most likely point is not suitable for this problem. Also, for decision tree, it relies the assumption that a single feature can seperate the sub dataset well and it always choose the best single feature in each term. However here, one single feature maybe meaningless, only the combination of certain features are meaningful.

Second, the decision boundary is not likely to be a straight line. Thus, logistic regression and SVM cannot learn the decision boundary very well here. Compared to random forest, they are not so flexible.

Evaluation of non-NN method

On the final model of random forest, I reach an accuracy for about 43% on validation set. But just single validation accuracy cannot reflect the whole thing. So, I generate the confusion matrix to see what the model did well and what did poorly.
From the plot, we can see that some kinds of pictures are predicted well, while some aren't. As the picture below shows, the model does well for some kinds, especially for airplanes, comet, gorilla, leopards and penguin, which are the brightest in the heatmap. However, for some other kinds, the model tends to confuse them with other kinds. For example, the model tends to predict a lot of other animals as gorilla. Also, the model tends to predict blimp as airplanes.



CNN

Since train convolutional neural network requires a lot of computation, I build my model on colab. Considering the limited train data and the policy that we cannot use pretrained weight, I didn't use any famous net structure such as vgg or res-net. Instead, I build a CNN network by myself. The network has two convolutional layer, each followed by relu activation and a 2*2 pooling layer. After that, I do a 4*4 maxpool to decrease the amount of parameters. Then, I use two fully connected layers with relu activation to get final result.

The network get about 48% accuracy on validation set. Detailed implementation can be found in PART4.ipynb