Tom and Jerry Image Classification: SVM, RandomForest and CNN

Mengyi Chen*

May 11, 2022

1 Introduction

In this project, we collect the images concerning Tom and Jerry. The data are downloaded from Tom and Jerry Image classification(https://www.kaggle.com/datasets/balabaskar/tom-and-jerry-image-classification). In the datasets, there are four kind of images, which decide on whether Tom and Jerry appears. We will apply three methods: SVM, RandomForest, CNN, which are classical methods in image classification. We will compare the accuracy of the three methods.

2 Data Description and Dimension Reduction

The Tom and Jerry image datasets contains 5478 images in total.

Type	Label	number
Neither Tom nor Jerry	0	1528
Tom	1	1930
Jerry	2	1240
Both Tom and Jerry	3	780

The four types have the following representatives:

- 'Neither Tom nor Jerry' means that in the image Tom and Jerry don't occur.
- 'Tom' means that we can only find Tom in the image, while can't find Jerry.
- 'Jerry' means that we can only find Jerry in the image, while can't find Tom.
- 'Both Tom and Jerry' means that both Tom and Jerry occur in the image.

Fig 1. is an illustration of the four types:

*E-mail: mc5215@columbia.edu

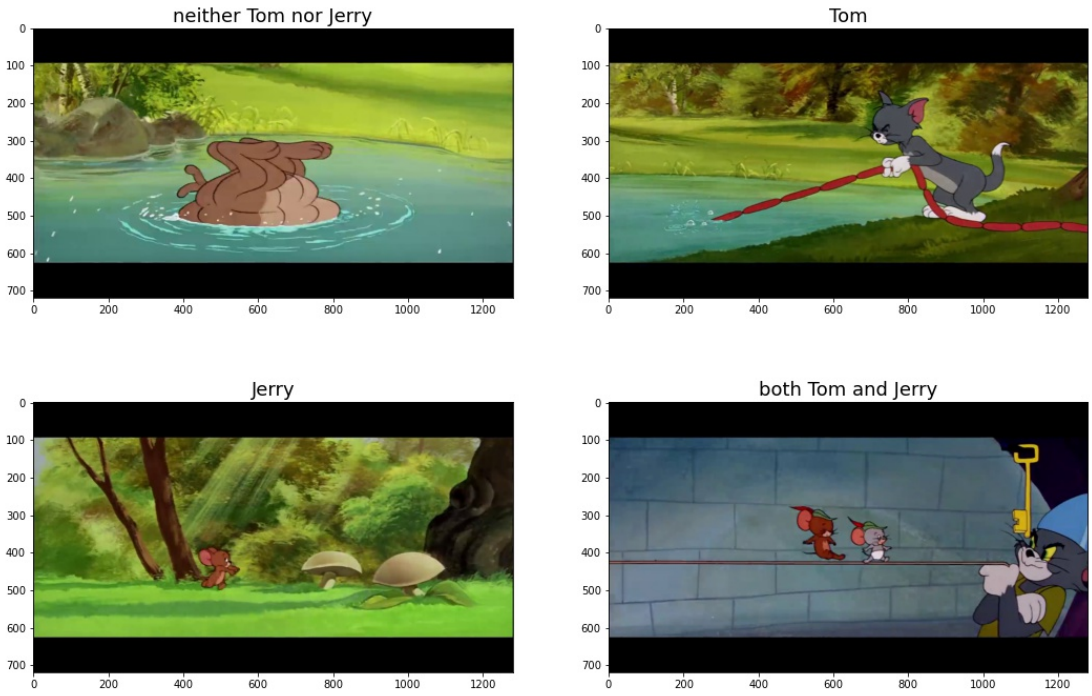

Figure 1: illustration of the four types of the images

These images have different sizes, we reshape them into shape $(90, 160, 3)$ for convenience. If we want to perform machine learning algorithms on the images, we algorithm will be too slow to proceed. This is because we have $90 * 160 * 3$ features! No doubt the large number of features slows down our program and may suffer from curse of dimension. Hence its a great idea to perform dimension reduction on the data first. We use PCA to extract 300 important features, then project the data to the 300 Principal components.

3 SVM

We divide the 5478 into training sets(3834points) and test sets(1644points). We perform svm.SVC with *rbf* kernel and defaulted parameters on the training data first. Then use the trained model to predict labels of test data and compare the outcome with the original labels of the test data.

- accuracy on training sets: 0.8257
- accuracy on test sets: 0.7244

Then we use cross-validation to choose the best parameter C and gamma:

$$\text{SVC}(\text{C} = 1245.3632237931529, \text{gamma} = 0.0001666033365920091)$$

Using the best C and gamma to train SVC model on the training data again:

- accuracy on training sets: 1.0
- accuracy on test sets:0.8266

The accuracy on both training sets and test sets improved greatly.

Figure 2: The confusion matrix on the test sets

4 Random Forest

Recall that in Random Forest method, each time we choose a node for a classification tree, we only select from a random sample of almost $\sqrt{n}$(n is the number of features) predictors. The aim of Random Forest is to decorrelate the trees. Suppose one predictor of the 300 predictors is very influential, then this predictor will be placed at the top node in every tree if we do not use Random Forest. Thus all the trees we build will look similar, which is exactly not what we expect. The method of Random Forest can help prevent overfitting.

We perform RandomForestClassifier of sklearn package to classify the data. There are two parameters in the function:

- n_estimators: the number of trees in the forest
- max_depth: the depth of each tree.

We use cross validation first to choose the best estimators. n_estimators is searched through $[1, 200]$ and max_depth is searched through $[1, 100]$. The best estimator we find is:

$$\text{RandomForestClassifier}(\text{max\_depth} = 68, \text{n\_estimators}=116, \text{random\_state}=0)$$

We adopt the best estimators and apply RandomForestClassifier to the data:

- accuracy of training:1.0
- :accuracy of test:0.7414841849148418

Figure 3: The confusion matrix on the test sets

The accuracy of training is 1, however the accuracy test is not very high. It really seems that Random Forest overfits. We fix one of n_estimators, max_depth to be the optimal value we solved using cross validation, then tune the other parameter. We have the following figure:

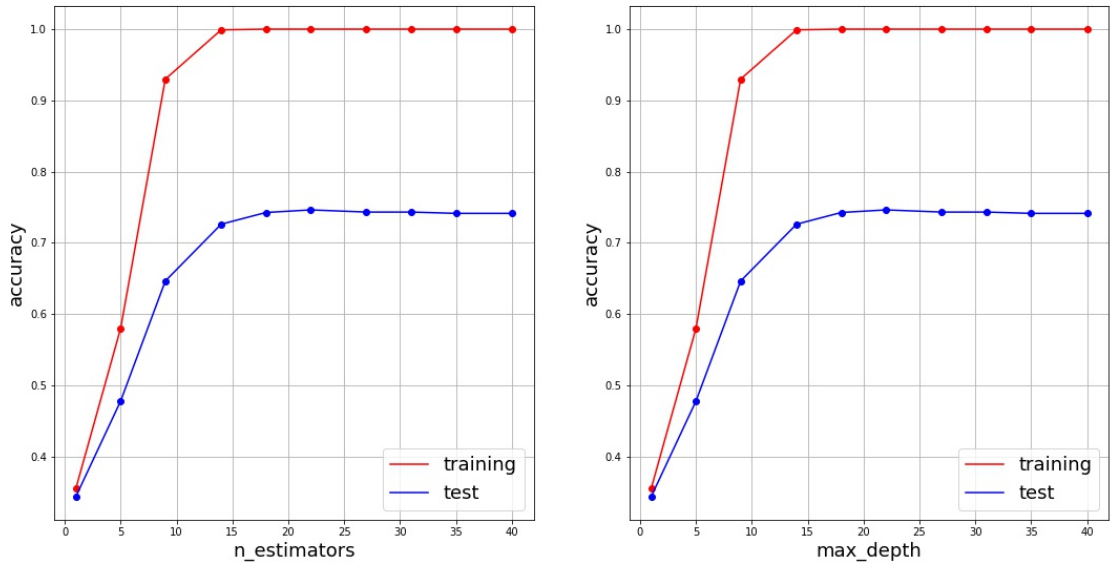

Figure 4: training/test accuracy as a function of n_estimators/max_depth

From Figure 4, we can see the training accuracy as well as test accuracy increases till n_estimators reach 15. Then their accuracy almost remains stable! In the right of Figure 4, we can observe similar patter when max_depth reach 15. Usually when overfitting occurs, we expect the training accuracy increases all the time and the test accuracy increases first then

decreases. However, in Figure 4, we can't observe obvious peak of training accuracy. We will explore more in discussion part.

5 CNN

Convolutional Neural Network is a popular image classification method which have shown wide range of success in numerous problems. CNN can extract specific patterns in the image, for example, eyes and mouths of human face. We expect CNN to perform well on Tom and Jerry Images.

When applying SVMRandom Forest method, we turn the image into a $(90, 160, 3)$ array. Then flatten the array and use PCA to obtain 300 principal components. We project the array onto the 300 principal projects. Each image is transformede into a one-dimensional array with length 300. We adopt tensorflow to perform CNN. Tensorflow has specific function for image classification. We handle 5478 images as follows:

1. For each types(neither Tom now Jerry, Tom,Jerry,both Tom and Jerry) of images, divide the groups of images into training group,test group, and validation group. Copy these images to the respective folders.

2. Now we have three folders: train, test, val. Each folders contain 4 subfolders: neither Tom now Jerry, Tom,Jerry,both Tom and Jerry.

3. Apply ImageDataGenerator of Tensorflow to create train, test, val object.

An advantage of using ImageDataGenerator rather than PCA is that we maintain all the features. We also do not need to create a huge array as in SVMRandom Forest to store the 300 features of all the data points, which can save memory.

We adopt the following CNN:

Layer (type)	Output Shape	Param
conv2d (Conv2D)	(None, 90, 160, 16)	1216
max_pooling2d (MaxPooling2D)	(None, 45, 80, 16)	0
dropout (Dropout)	(None, 45, 80, 16)	0
conv2d_1 (Conv2D)	(None, 45, 80, 32)	12832
max_pooling2d_1 (MaxPooling2D)	(None, 22, 40, 32)	0
dropout_1 (Dropout)	(None, 22, 40, 32)	0
conv2d_2 (Conv2D)	(None, 22, 40, 32)	25632
max_pooling2d_2 (MaxPooling2D)	(None, 11, 20, 32)	0
dropout_2 (Dropout)	(None, 11, 20, 32)	0
dropout_2 (Dropout)	(None, 11, 20, 32)	0
dense (Dense)	(None, 144)	1013904
dense_1 (Dense)	(None, 4)	580
Total params: 1,054,164 Trainable params: 1,054,164 Non-trainable params: 0		

We add Dropout layer to help prevent overfitting.

The accuracy of this CNN after 30 epoches is :

- train_accuracy: 0.9842
- val_accuracy: 0.8219
- test_accuracy: 0.8109339475631714

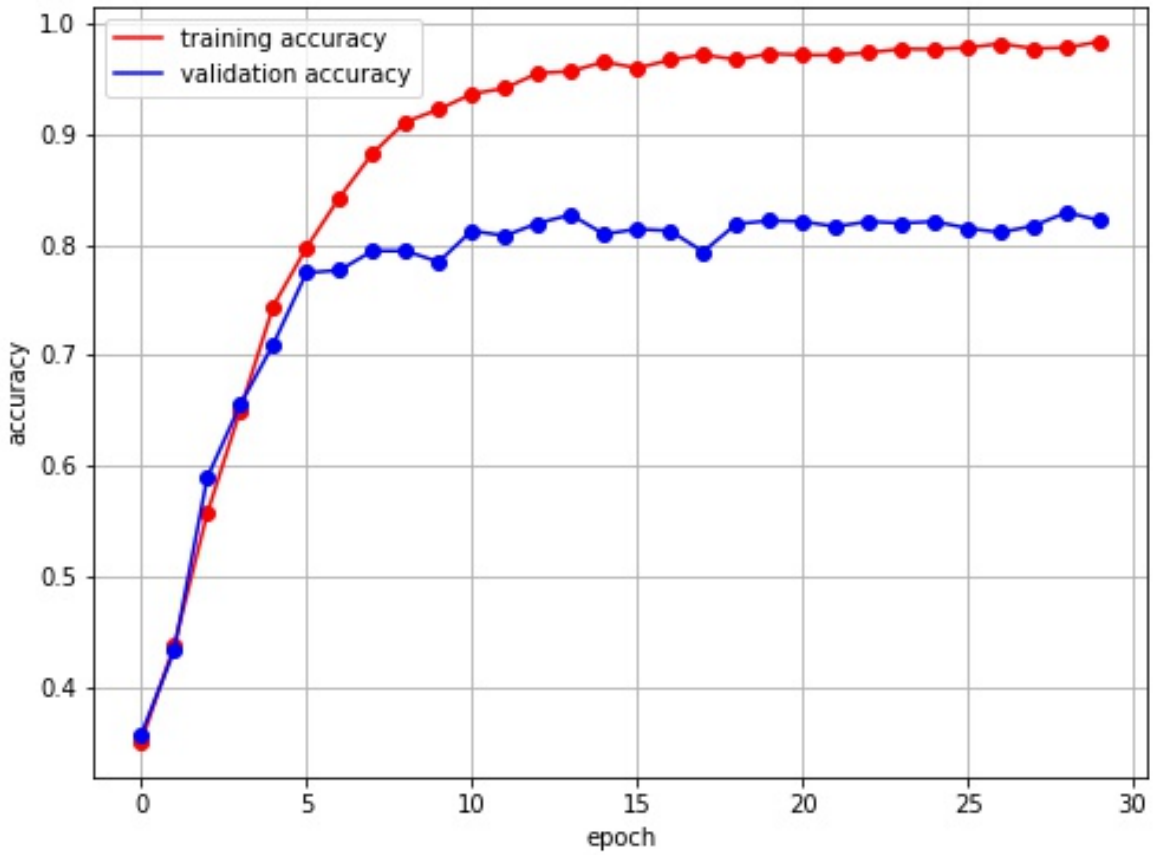

Figure 5: training/test accuracy as a function of epoches

6 Discussion and Comparisson

We compare the three method: SVM, Random Forest, CNN:

method	Train accuracy	Test accuracy
SVM	1.0	0.8266
Random Forest	1.0	0.74
CNN	0.9842	0.8109

What we really care is the test accuracy. We can see that SVM and CNN both outperform Random Forest method. Random Forest method is not very suitable for Tom and Jerry image classification.

We expect the CNN to perform much better than other machine learning methods, however, in this experiment, the performance of CNN is almost the same as SVM. I guess there are several reasons:

- The CNN model chosen is too simple. There are many CNN such as AlexNet, ZFNet, VGG16, VGG19, GoogleNet which perform well. However, I tried to apply VGG16 but the CNN is too complex for my laptop to run. The running time would be extremely long which is not realistic. If I am offered several GPU or TPU, probably I will apply more CNN to our data and compare the results. The CNN model chosen is limited to computational capacity.

- As we can see, the train accuracy of SVMRandom Forest are both 1. CNN's train accuracy is close to 1(Recall we add dropout layer). We use cross validation to choose the best parameters of SVMRandom Forest, but their test accuracy are still not very high. As shown in Figure 4, it's not very likely that the SVMRandom overfit. I guess our dataset-Tom and Jerry Images is not good enough. In most of the images, Tom and Jerry only occupy a very small part of the picture. Furthermore, many pictures are similar. I guess one of the main reasons why our train/test accuracy outcome are like this is that the data is not good enough.