STA 141C Final Report

Member 1 Jack Zheng
Member 2 Fengshuo Song
Member 3 Dingting Shen

Member 1 Contribution: Search for dataset, build up a proposal, communicate ideas with professor, design the modification plan, tuning parameters, conduct the final report.

Member 2 Contribution: search for dataset, build up proposal, data processing, design cnn architecture, tuning parameters, test the model, merge components, debug, handle communication between team members, conduct the final report.

Member 3 Contribution: build up the proposal, methodological research and develop algorithms, connect and test the model, conduct the final project

I. Introduction:

In this project, we are mainly focusing on classifications on figures about cats and dogs. We have built classification models of CNN and SVM, and then we are able to predict if a given image is of a cat or a dog and assess accuracy. Convolutional Neural Network (CNN) is an algorithm taking an image as input then assigning weights and biases to all the aspects of an image and thus differentiates one from the other. The dataset of cats and dogs is useful as the basis for learning and practicing how to develop, evaluate, and use convolutional neural networks for image classification. Support Vector Machine (SVM) is another classification algorithm that can give a decision boundary by maximizing the margin.

We have applied these methods to the dataset and compared the results. According to their different performance results, we are able to get a rough conclusion about which algorithm is suitable for what kind of dataset. We have built two different architecture editions of CNN. The first edition is a four-layer CNN. The second edition is a pre-trained neural network with VGG16. The third method is Support Vector Machine (SVM).

II. Dataset and Methodology

We have two folders named “test_set” and “training_set”, and there are 10,000 images in both folders. The folder “training_set” contains two sub folders “cats” and “dogs”, each holding 8000 images of the respective categories; the folder “test_set” also contains two sub folders “cats” and “dogs”, each holding 2000 images of respective categories.

The size and dimensions of each image are different. So, before manipulating all steps, we would take a deep look at the data and perform different types of operations. First, we read the original pictures from training and testing datasets and stored them separately. Through getting the insight of the data, we found that our image has a large size and parameters. And these parameters will only increase as we increase the number of hidden layers. This is where convolutional neural networks can be helpful. CNNs help to extract features from the images by extracting from low dimensional features to high dimensional features like the shapes. In the model, we use CNN and VGG architecture to extract features from the image. For CNN with VGG architecture, only a few layers near the end are fully connected. VGG is an implementation of CNN by the Visual Geometry Group.

Support Vector Machine (SVM) algorithm is one of the standard classification techniques. For classification problems, SVM may give an optimal solution. However, SVM is not suitable for classification of large datasets. There are some possible methods to apply SVM classification for large datasets. We choose to select representative training data from a large data set so that a normal SVM could handle it with extracted features.

III. Implementation Details

Neural networks can be trained by using batches of images, each of them having a label to identify the real nature of the image. Here we used images of cats and dogs. A batch can contain a few tenths to hundreds of images. For each image, the network prediction is compared with the corresponding existing label, and the distance between network prediction and the truth is evaluated for the whole batch. Then, the network parameters are modified to minimize the distance and thus the prediction capability of the network is increased. The training process continues for every batch in a similar way.

For the first edition multi-layer CNN, we reshaped the raw image to 64*64 pixels, and transform images to grayscale first. To construct a CNN architecture, we use 4 layers in total to extract features from the images. In the __init__ class, we configure the first three different trainable modules including convolution and affine layers with nn.Conv2d in a sequential container. Sequential class allows us to build PyTorch neural networks and equal stride layers by specifying ReLU andmax poolings function inside. And we use Linear function to fully connect dense layers to classify those features into their respective categories. Then we set the loss function and define the optimizer by using Stochastic Gradient Descent (SGD). Finally, we train the model for 20 epochs. As for the time of training data, it took a large amount of time if running on CPU, with a total around 30 minutes finishing running all epoches on a normal computer; however, the training time could be greatly reduced while running on GPU, with about 40 seconds running all epoches.

For the second edition CNN, we used a pre-trained CNN model for the convolutional base and designed the prediction layers. The pre-trained CNN model, VGG16, which had been trained from millions of high resolution images belonging to roughly 22000 categories (Hassan, 2018), was used to extract the features of a given image in our model. Since the model is already trained, we freeze the convolutional base (keep the weights unchanged) while training our model. The prediction layer we used was different from the model provided in VGG16, since the model was designed to predict 1000 classes. After trying several different prediction layers, we found that adding a log-softmax activation function at the end can perform the best classification. The VGG16 uses 224 x 224 RGB image, so the data used for training were resized. As for training the data, it took a large amount of time if running on CPU, with over 30 minutes per epoch on a normal computer; however, the training time could be greatly reduced while running on GPU, with about 40 seconds per epoch. This was done by using google colab, which provided the GPU accelerator for computations.

As for the SVM method, we used the resized gray-scale images (64*64 pixels) for training to reduce computations and save time. We also used the technique of PCA to further reduce the computation while also maintaining relatively the same accuracy. In order to obtain the best model for prediction, we had tuned several parameters before summarizing the final results. We first tuned the kernel function on the data with full dimension; then we tuned the dimension to be reduced to; and finally we tuned the bandwidth of the kernel function. (Figure 5 & Figure 6 & Figure 7) During these steps, we assumed that different parameters are independent of each other (i.e. RBF always outperforms other functions whatever the dimension and bandwidth is). We finally got the SVM model with the RBF kernel function, reduced to 13-dimensional space and gamma of 0.0003.

IV. Results and Interpretations

In the original CNN model, we could observe that with smaller image size and less convolutional layers, the accuracy would get smaller, because the features extracted become less. It is notable that our network's performance degrades if a single convolutional layer is removed. With the model we are using with 4 layers, the test accuracy is around 80%. (Figure 1 & Figure 2) The test accuracy we got from this CNN was not high enough to satisfy us. In order to make improvements, we introduced a pre-trained network -- VGG16 in the second edition. This pre-trained network helps to increase the test accuracy by 17% (from 80% test accuracy to 97% test accuracy). (Figure 3)

In CNN models, we also tried different activation functions. In the first version CNN, most layer function we used is ReLU (Rectified linear unit). It is the most widely used activation function and chiefly implemented in hidden layers of the Neural network. For some circumstances, it is less computational because it involves simpler mathematical operations. At a time only a few neurons are activated, making the network easy and making ReLU efficient for computation. In the Second CNN model, we added more layers and used VGG16 to create a multi-class classifier. The net contains eight layers with weights; the first five are convolutional and the remaining three are fully connected. Our network maximizes the multinomial logistic regression objective, which is equivalent to maximizing the average across training cases of the log-probability of the correct label under the prediction distribution.

The CNN with pre-trained VGG16 network gives an overall of 97% accuracy on the test dataset. The normal SVM model gives a final of 65% accuracy on the same test dataset. (Figure 4) In general, CNN with a pre-trained network outperforms the normal SVM. In order to find out the cause of this outcome, we dug deeper and made a comparison of these two approaches. After we looked closer at these two approaches, we found out that the reason for the bad performance of the normal SVM algorithm was the poor flexibility and feature extraction ability of the normal SVM. The cats and dogs image dataset we used in this project is more complicated. In that case, the hyperplane dimension needs to be changed from 1 to the n-th dimension. (Kumar, 2020) When we used the normal SVM to do the classification, the model was unable to extract the features within pictures well. Since SVM needs to solve the quadratic programming problems in order to find a separation hyperplane, which causes an intensive computational complexity. (Cervantes, 2008) This drawback contributes the most to the bad performance of the normal SVM. As a result, we concluded that the normal SVM algorithm is too simple to be able to capture all the useful features in the pictures. From an outside article (Cervantes, 2008), we also confirm our hypothesis that the nature of normal SVM, "normal SVM is not suitable for classification of large data sets", potentially constrains its performance on our dataset. We are indeed able to increase the accuracy of the SVM by increasing the dimension of PCA. However, the best we can improve is around 1 to 2 percent of accuracy. In the meanwhile, the training time that the SVM algorithm needs is going to increase dramatically.

On the contrary, with the help of pre-trained network VGG16, the CNN we trained is able to extract tons of features from the pictures. The VGG16 is such a powerful feature extraction tool that is able to "achieve 92.7% test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes" (Hassan, 2018). When processing our dataset, we only have two different categories, so it is fairly easy for VGG16 to detect the useful features.

Moreover, the network's size is limited mainly by the amount of memory available on current GPUs and by the amount of training time that we are willing to tolerate. For higher and future expectations, all the process of our experiments indicates that our manipulation and results can be improved simply by waiting for faster GPUs and bigger datasets to become available.

References

Cervantesa, J. & Li, X. & Yu, W. & Li, K. (2008). Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing (71).* https://www.sciencedirect.com/science/article/pii/S0925231207002962

Hassan, M. (2018). VGG16 – Convolutional Network for Classification and Detection.https://neurohive.io/en/popular-networks/vgg16/

Kumar, A. (2020). SVM (Support Vector Machine) for classification. SVM (Support Vector Machine) for classification | by Aditya Kumar | Towards Data Science

Appendix

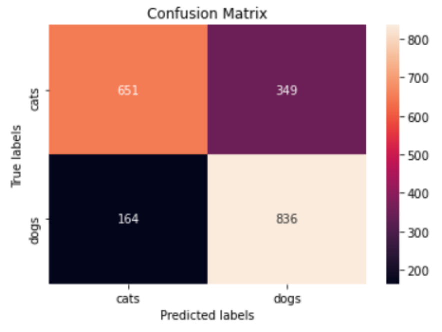
# Figure 1 (up left)
Confusion matrix plot under CNN model with image size 32*32 and 3 layers.
We can observe that the accuracy is 0.74.

# Figure 2 (up right)
Confusion matrix plot under CNN model with image size 64*64 and 4 layers.
We can observe that the accuracy is 0.80, which is higher than above with less layers.

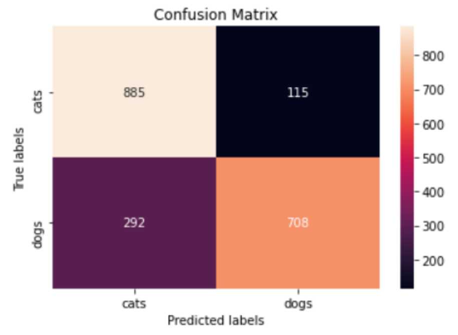
# Figure 3 (Below left)
Confusion matrix plot under CNN with image size 224*224 and VGG16 model.
We can observe that the accuracy is 0.97, which is the highest among all algorithms.

# Figure 4 (Below right)
Confusion matrix plot under SVM using SVC classifier with kernel of rbf.
We can observe that the accuracy is 0.65, which is the lowest among all algorithms.

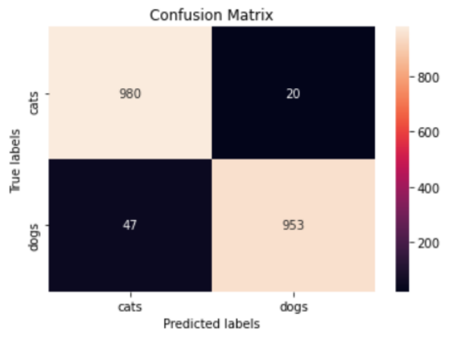
```
              precision    recall  f1-score   support

        cats       0.80      0.65      0.72      1000
        dogs       0.71      0.84      0.77      1000

    accuracy                           0.74      2000
   macro avg       0.75      0.74      0.74      2000
weighted avg       0.75      0.74      0.74      2000
```



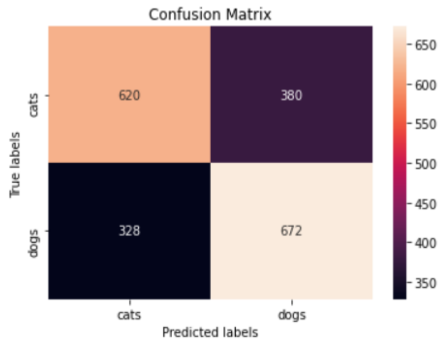
```
              precision    recall  f1-score   support

        cats       0.75      0.89      0.81      1000
        dogs       0.86      0.71      0.78      1000

    accuracy                           0.80      2000
   macro avg       0.81      0.80      0.79      2000
weighted avg       0.81      0.80      0.79      2000
```



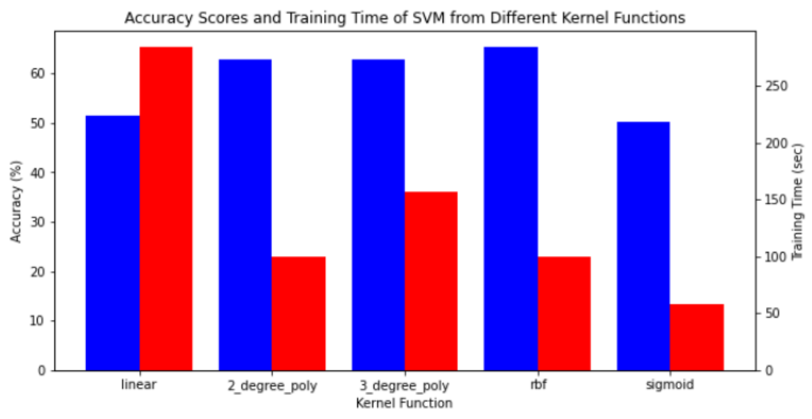
```
              precision    recall  f1-score   support

        cats       0.95      0.98      0.97      1000
        dogs       0.98      0.95      0.97      1000

    accuracy                           0.97      2000
   macro avg       0.97      0.97      0.97      2000
weighted avg       0.97      0.97      0.97      2000
```



```
              precision    recall  f1-score   support

        cats       0.65      0.62      0.64      1000
        dogs       0.64      0.67      0.65      1000

    accuracy                           0.65      2000
   macro avg       0.65      0.65      0.65      2000
weighted avg       0.65      0.65      0.65      2000
```

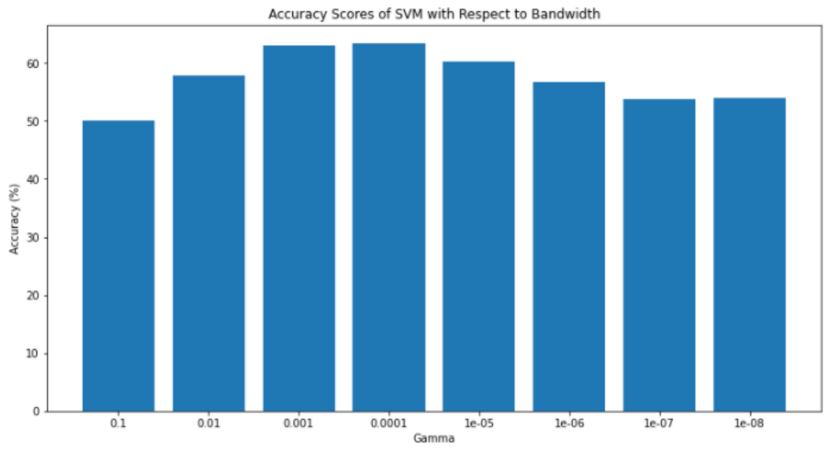


<u># Figure 5</u>
Barplot of Accuracy Scores and Training time of SVM from different kernel functions.



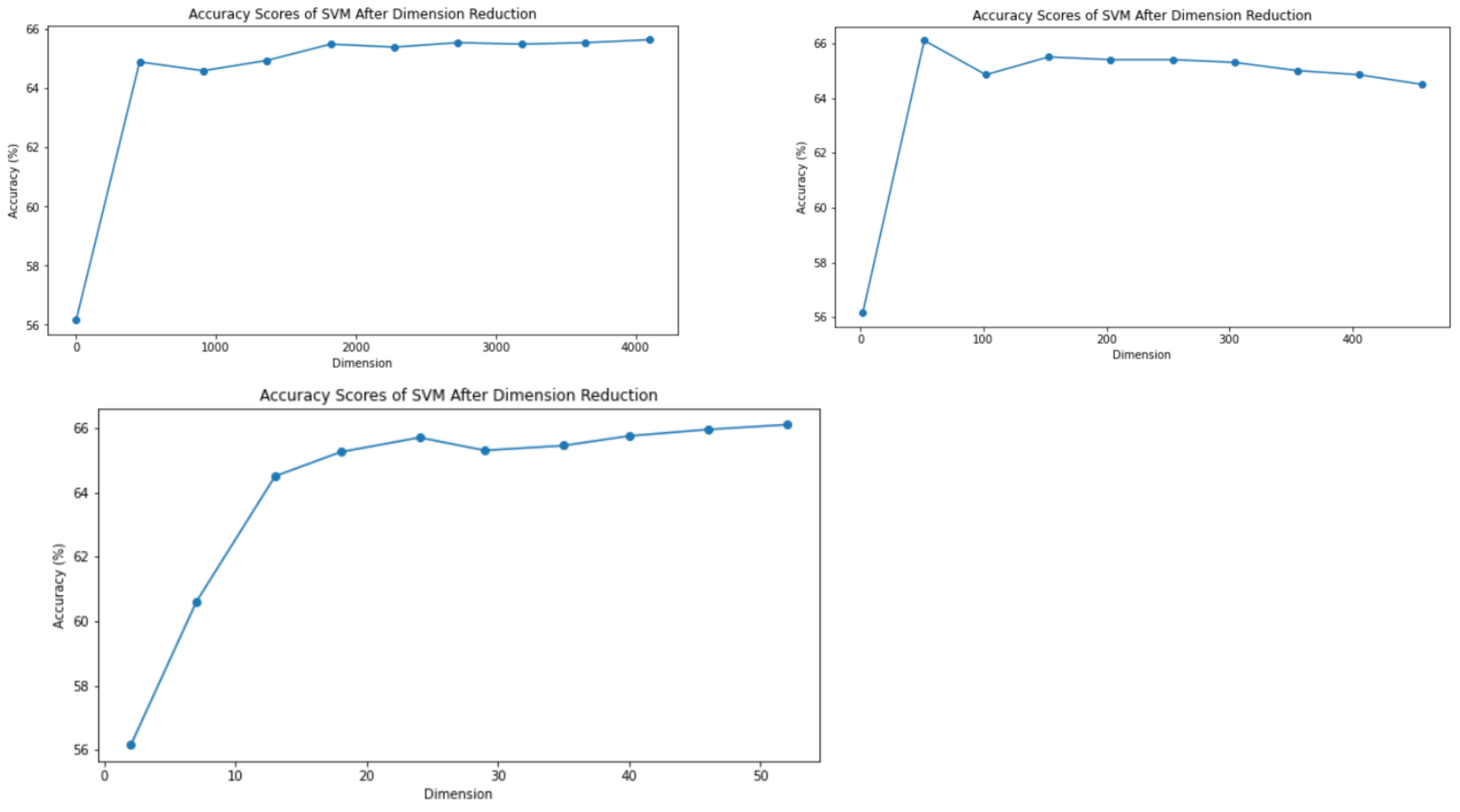
<u># Figure 6</u>

Barplot of Accuracy of SVM with Respect to Bandwidth



<u># Figure 7</u>

Pointplot of Accuracy Scores of SVM after Dimension Reduction


Accuracy Scores of SVM After Dimension Reduction
Accuracy (%)
Dimension
Accuracy Scores of SVM After Dimension Reduction
Accuracy (%)
Dimension
Accuracy Scores of SVM After Dimension Reduction
Accuracy (%)
Dimension