Binary Image Classifier for Guilty Dog

Jiaming Li
Project ID: 3
Courant Institute of Mathematical Sciences
251 Mercer St, New York, NY 10012
jl10321@nyu.edu

Abstract

This project examines the performance of some commonly used models in the context of binary image classification on a small dataset of dog facial expressions. Different variations of Support Vector Machines (SVM), Convolutional Neural Network (CNN), and CNN-SVM are employed to investigate the relation between testing performance and the values of hyperparameters, as well as the general performance of the three models.

The findings reveal that CNN with smaller filter numbers, dropout regulations, and fewer MaxPooling layers are optimal for this dataset (accuracy about 0.9). In general, SVMs are not complicated enough to capture the patterns using this small image-based dataset (accuracy about 0.5), but its performance can be improved by about 10% to 20% if implementing a CNN for feature extraction before applying SVM.

Keywords: binary image classification, SVM, CNN, CNN-SVM, dog emotions

1 Introduction

Understanding and interpreting facial expressions in dogs is crucial for comprehending their emotional states and behavior. For dog owners everyday scenarios like torn blankets or knocked-over objects can be challenging. When it's unclear which dog is responsible for such incidents, valuable lessons might not be effectively learned by the guilty party. With the recent emerging use of machine learning techniques in image classification, this report introduces a binary image classifier by exploring Support Vector Machine (SVM), Convolutional Neural Network (CNN), and the combination of the two (CNN-SVM) to discern whether a dog exhibits a "guilty face" or not.

2 Related Work

There has been a lot of research showing that dogs, compared to other animals in the Canidae, developed different muscle structures around the eyes [1]. These muscles allow domestic dogs to display facial expressions of happiness, anger, pain, etc, which are used in the clinical recognition of moods[2]. As a result, research has been conducted to determine a dog's emotional state based on its facial expressions. One of which, closely related to this project, compares the authors' neural network with the existing Support Vector Machine, LeNet-5, and other models.[3].

While guilt is an intricate concept, the "guilty look", characterized by a tucked tail, visible whites of the eyes, hunched posture, flattened ears, etc, surely exists. All these features are consistent with fear/stress emotions in dogs, which are categorized as possible emotions in dogs. In his previous work, Beerda concludes that a lowered posture of the dog (lowered position of the ears, tail and body) "appears to be a more consistent indicator of stress"[4].

3 Method

The models chosen for this project are SVM with linear, polynomial, and Radial Basis Function(RBF) kernels, CNN, and CNN-SVM.

- SVM is a relatively simpler model, which makes it suitable to perform a binary classification task on a small dataset since it is less likely to overfit. However, since the patterns in the image data are not very obvious in this project, an SVM might not be able to capture the corresponding features.
- In the meantime, CNN is also commonly adapted for image-based classifiers since the complex structure enables the network to capture hidden patterns/ more complicated features of data. Yet, the enormous number of trainable parameters, a result of CNN's complexity, makes it more likely to overfit the training data, which may result in poor performance in testing scores.
- Therefore, other than applying pure SVM and CNN, this project also explores the combination of the two, which is a CNN-SVM model to see if this combination could result in a better performance than a pure SVM/CNN.

This project will first perform a grid search to find the best kernel type and hyperparameters for the SVM model, and serve it as a baseline algorithm. Then the project will try CNN-SVM with different layer structures and regulations to find out the best-performing one. Finally, a pure CNN with the same layers except for the final dense layer and loss function(the SVM structure) will be performed to compare with the CNN-SVM.

The majority of the dataset is from an existing Kaggle dataset: Dog Emotions Prediction `https://www.kaggle.com/datasets/devzohaib/dog-emotions-prediction`. The others are collected via Google images. Due to the small size of this dataset (300 images in each class), all images are manually checked and classified. The standards for classifying and examples of images can be found in the Dataset section of this report.

4 Dataset

The dataset for this project consists of two classes: Guilty and Not Guilty, which represent images of "guilty-looking" dogs and the ones who don't look "guilty". The tagging is manually done and follows the stress indicators concluded by Beerda[4]. Examples of the two classes are as follows. The images are of different sizes and various resolutions but will be scaled to the same size (256×256) during preprocessing.

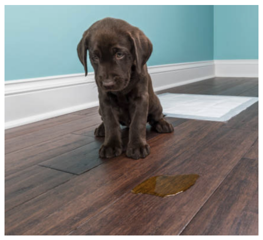


Fig 1: flat ear

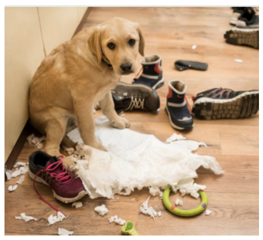


Fig 2: hunched back

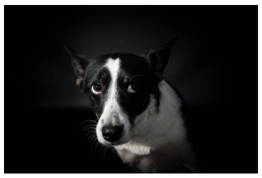


Fig 3: visible whites of the eye

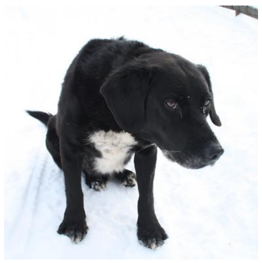


Fig 4: avoid eye contact

Guilty class examples

Fig 5: happy

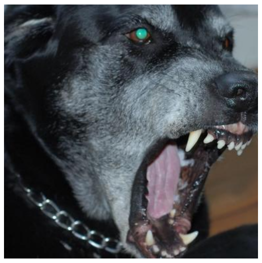


Fig 6: angry

Fig 7: relaxed

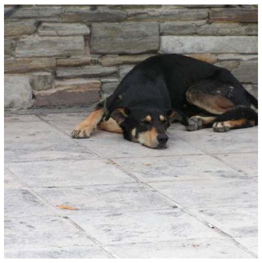


Fig 8: sleeping

Not Guilty class examples

5 Experiments

5.1 Load and Preprocess Data

The program loads the two classes and scales all images to the same size (256× 256). Every image data now is a 3d array of size (256,256,3), where the last dimension of 3 represents the RBG color value of one pixel. The program rescales the RBG value of all images from (0,255) to (0,1) by dividing the values by 255, and performs a split of 70% train, 20% validation, and 10% test.

5.2 SVM

The program tries three different Support Vector Machines: linear kernel, polynomial kernel, and RBF kernel to select the best model.

- Linear kernel (hyperparameter: C - regulation parameter)
- Polynomial kernel (hyperparameter: degree - polynomial degree)
- RBF kernel (hyperparameter: gamma - influence parameter)

The result from one run is as follows (accuracy for validation set):

```
({'C': 0.1, 'kernel': 'linear'}, 0.4479166666666667),
({'C': 1, 'kernel': 'linear'}, 0.4479166666666667),
({'C': 10, 'kernel': 'linear'}, 0.4479166666666667),
({'gamma': 0.1, 'kernel': 'rbf'}, 0.4583333333333333),
({'gamma': 0.01, 'kernel': 'rbf'}, 0.4583333333333333),
({'gamma': 0.001, 'kernel': 'rbf'}, 0.4583333333333333),
({'degree': 2, 'kernel': 'poly'}, 0.4479166666666667),
({'degree': 3, 'kernel': 'poly'}, 0.4375),
({'degree': 4, 'kernel': 'poly'}, 0.4375),
({'degree': 5, 'kernel': 'poly'}, 0.4479166666666667)
```

As shown, RBF kernel SVMs perform generally better than linear and polynomial kernel SVMs in terms of accuracy. However, after 10 consecutive tests, the best parameter varies from "RBF" to "linear", from "gamma = 0.1" to "gamma = 0.001". Though the best parameters for the SVM model may vary, the validation accuracy and the final test accuracy don't alter much. Without the loss of generality, the project selects the RBF SVM with gamma = 0.1 for testing, and gets a result of:

```
Accuracy:          0.5625
Recall:            0.42857142857142855
Precision:         0.5
```

5.3 CNN

A grid search in CNN structure would be harder to realize than SVM. However, it is possible to start with some reasonable guesses given the features of the dataset. Firstly, the total trainable parameters in the model cannot be too large due to the small size of the dataset. Therefore, the CNN should have a relatively smaller filter number in the convolution layers and fewer layers in general to prevent overfitting. The CNN model should be simpler as well as smaller than a CNN that is trained on a large dataset (e.g. AlexNet). Secondly, adding dropout layers with a larger dropout rate should be helpful to prevent overfitting. The dropout layer serves as a regulation in CNN as an l2 regulation in SVM.

Base CNN The base CNN model (**model A**) is as follows:

```
 1  Conv2D(16, (5,5), 1, activation='relu', input_shape=(256,256,3)
 2  MaxPooling2D(pool_size=(2, 2))
 3  Conv2D(16, (3,3), 1, activation='relu')
 4  MaxPooling2D(pool_size=(2, 2))
 5  Conv2D(32, (3,3), 1, activation='relu')
 6  MaxPooling2D(pool_size=(2, 2))
 7  Conv2D(32, (3,3), 1, activation='relu')
 8  MaxPooling2D(pool_size=(2, 2))
 9  Flatten()
10  Dense(256, activation='relu')
11  Dense(1, activation='sigmoid')
```

The model takes the default "`Adam`" optimizer (learning rate = 0.001), binary cross-entropy loss function, and accuracy as metrics.

Variation 1 The variation 1 model (**model B**) has the same layer structure as model A, except for the filter numbers of all convolution layers are multiplied by 2. For example, the third line would be `Conv2D(32, (3,3), 1, activation='relu')`.

Variation 2 The variation 2 model (**model C**) adds two dropout layers to model A. `Dropout(0.3)` are added before line 5, and before the final output layer (line 11).

Variation 3 The variation 3 model (**model D**) adds two more Maxpooling layers to model A. `MaxPooling2D(pool_size=(2, 2))` are added before line 6 and 8, which reduces the model size significantly.

Model Choosing Run 15 epochs of every variation of the CNN above, the program obtains the following results.

The final epochs (A-B-C-D) return:

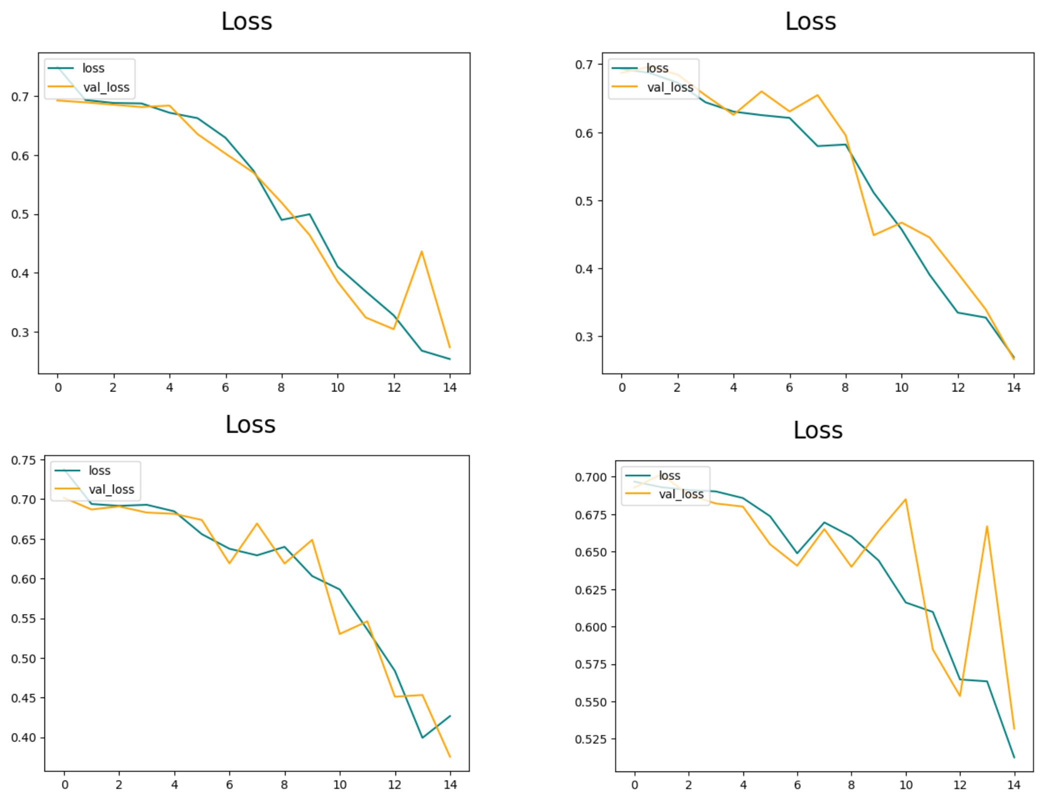
```
loss:0.2540-accuracy:0.9062 - val_loss:0.2740-val_accuracy:0.8646
loss:0.2693-accuracy:0.8966 - val_loss:0.2667-val_accuracy:0.8750
loss:0.4267-accuracy:0.8077 - val_loss:0.3754-val_accuracy:0.9062
loss:0.5126-accuracy:0.7476 - val_loss:0.5317-val_accuracy:0.7083
```

From the graphs, it is easy to see that models A, B, and C all reached a validation accuracy above 0.85 except for model D. The gap between validation and training loss is larger and the loss curve is much less stable for model D. Compare to model A, the validation accuracy is not improving as much in model D. We can conclude this is a result from a underfitting network (model too simple to capture complex features).

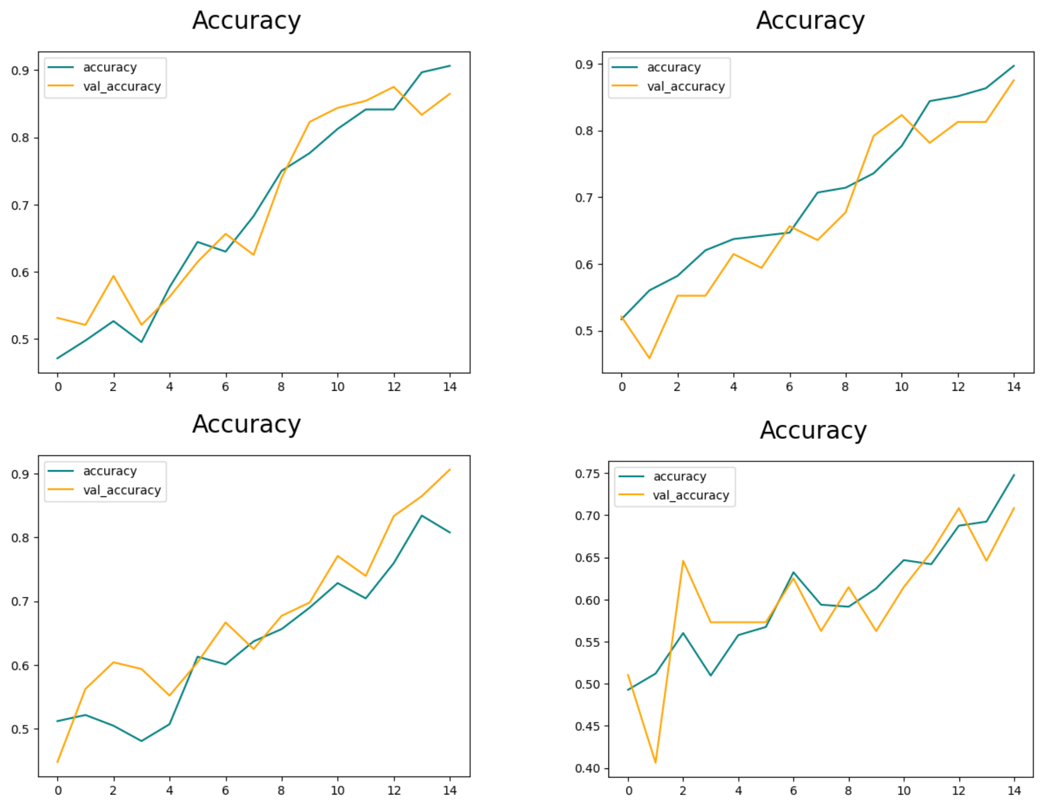
Both models B and C get a better validation accuracy after 15 epochs than the base CNN. Here the project chooses model C for testing since the higher the validation accuracy, the stronger the ability to make correct classification.

Test Result The test result using model C is as follows:

```
Accuracy:        0.90625
Recall:          0.8500000238418579
Precision:       1.0
```



loss from left to right, up to down: Model A, B, C, D



accuracy from left to right, up to down: Model A, B, C, D

5.4 CNN-SVM

In this model, the project combines the 2 best-performing models together to see if the usage of a CNN for feature extraction will improve the performance of SVM, as well as if SVM serves as an extra effective way for preventing CNN from overfitting.

Train CNN Train the CNN on the dataset. This is already done in the previous 5.3 CNN section.

Extract Features Use the trained CNN to extract features from the dataset. Here the program uses `Model()` from `keras.models` and define its input as the input for model C, and output as the dense layer output, which will be fed into the SVM model.

Train SVM Train the SVM (RBF kernel with gamma = 0.1) on the training data.

Evaluate Evaluate the CNN-SVM model using the 10% testing data.

The CNN-SVM model obtains a result of :

```
Accuracy:        0.625
Recall:          0.6
Precision:       0.6
```

6 Conclusions

SVM, CNN, and CNN-SVM models are all efficient models for completing a binary image classification task. This project aims to choose the best-performing hyperparameters for the above three models when dealing with a small image dataset and compare their results.

SVM doesn't have an ideal performance when dealing with an image-based small dataset. The best model this project decides is an RBF kernel SVM with gamma value = 0.1, which gets a test accuracy of around 0.5. CNN on the other hand, has a significantly better performance. After 15 epochs, the best-performing model obtains a test accuracy of around 0.9. CNN-SVM model in this project combines the two best-performing models. Yet, the performance of the CNN-SVM does not exceed a pure CNN. While it still shows an obvious improvement of around 10% to 20% in accuracy compared to SVM after combining it with CNN.

Although not discussed in detail, the initial guess for the base CNN in this project was determined after more than 20 trial models. The first few CNN guesses yielded poor performance, sometimes even worse than a linear SVM, with an unchanging validation accuracy and increasing validation loss. It became evident that executing a CNN on a small dataset had a high likelihood of leading to overfitting.

The results in this report are from one representative execution, the specific value may vary in the code appendix.

References

[1] Juliane Kaminski, Bridget M. Waller, Rui Diogo, Adam Hartstone-Rose, and Anne M. Burrows. Evolution of facial muscle anatomy in dogs. *Proceedings of the National Academy of Sciences*, 116(29):14677–14681, 2019.

[2] Daniel et al. Mota-Rojas. Current advances in assessment of dog's emotions, facial expressions, and their use for clinical recognition of pain. *Animals : an open access journal from MDPI*, vol. 11,11 3334. 22, Nov. 2021.

[3] Liu Y. Mao, Y. Pet dog facial expression recognition based on convolutional neural network and improved whale optimization algorithm. *Sci Rep*, 13, 3314, 2023.

[4] Jan.A.R.A.M. van Hooff Hans W. de Vries Bonne Beerda, Matthijs B.H. Schilder. Manifestations of chronic and acute stress in dogs. *Applied Animal Behaviour Science*, 52:307–319, 1997.