Three Classes Face Detection

Ting-Liang Huang
Oregon State University
huangtin@oregonstate.edu

1. Problem Statements

Deep neural network can efficiently solve image classification problem. The most intuitive way to train the model is to flatten images to a vector and then use Convolution Neural Network(CNN) model to train datasets. In recent years, technological progress in GPU that provides multi-threads functionality can efficiently increase the time performance of training process. Therefore, people start to try bigger model and even directly feed high pixel of images to CNN without using any size reduction method. Even though GPU can provide powerful computational power, this strategy will still need numerous datasets and long-time training processes. In order to address this issue, using expert's recommendation will be another technique that people were used a lot. In the following sections, I will use image detection methods to extract meaningful points to train CNN.

1.1. challenges of the problem

Dogs, cats, and human face datasets are three classes classification problem. I will use one-hot encoding to label classes. The challenge will be finding meaningful features to differentiate those images since dogs, cats, and humans share lots of similar features e.g., eyes, nose, month in Figure 1. The number of datasets will be changed up or down depending on the later training results.

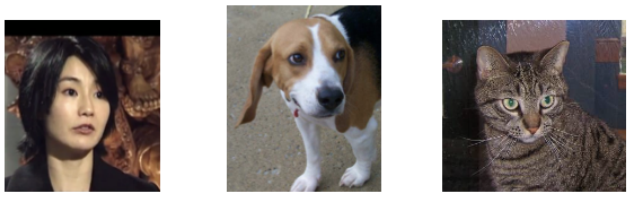

Figure 1. Human, dog, and cat images

1.2. Motivation

When human look at each others, or even look at animals. The most easier way to differentiate the differences of them is to look at their faces first. A lot of animal classification datasets care not only animals' faces but also their body shape. This brought my attention and I think image detection can helps addressing this problem domain. The following sections will introduce how this method works.

2. Approach

In this paper, I implement multiple computer vision and deep learning techniques to train three classes datasets. First, image augmentation is an important step to clean the data so that it can match the CNN model. Good data prepossessing always leads excellent results. Secondly, image detection methods will change image to multiple keypoints showing the features of data. Once getting image keypoints, I will feed the keypoints to CNN model. Lastly, several metrics will be used to evaluate the training and testing results (Figure 2).

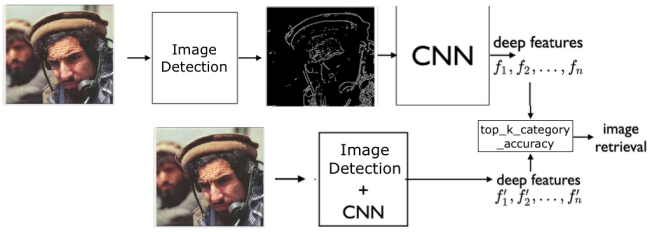

Figure 2. Approach flow

Image Augmentation. has been shown as an effective way to improve model generalization and reduce overfitting in computer vision field. In this paper, I use the general image augmentation method includes: horizontal flip and dimensional reduction. After image augmentation, I generalize two times more data from the original datasets.

Image Detection. Before the boom of AI, people usually use keypoint detection to help program differentiating images. Advantage of the method is that it requires less number of datasets. In this paper, edge, gray-scale, and colour detection techniques (Figure 3) will be used to train CNN.

CNN Model. To select an appropriate model, we mainly focus on CNN with dropout. Dropout is a method that helps reducing time complexity during training and addressing

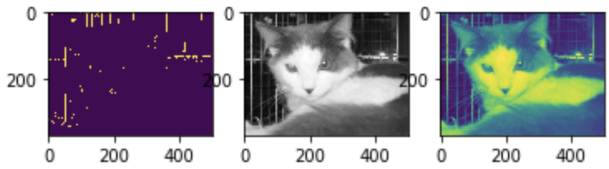

Figure 3. **Detection** left:edge, middle:gray-scale, right:colour

overfitting problem. At first, 5x5 filters will be used to track larger features. Next, 3x3 filters will be used to address more speific part of featrues(Figure 4).

Layer (type)	Output Shape	Param #
conv2d_67 (Conv2D)	(None, 60, 60, 64)	1664
activation_90 (Activation)	(None, 60, 60, 64)	0
max_pooling2d_67 (MaxPooling	(None, 30, 30, 64)	0
dropout_48 (Dropout)	(None, 30, 30, 64)	0
conv2d_68 (Conv2D)	(None, 28, 28, 32)	18464
activation_91 (Activation)	(None, 28, 28, 32)	0
max_pooling2d_68 (MaxPooling	(None, 14, 14, 32)	0
dropout_49 (Dropout)	(None, 14, 14, 32)	0
conv2d_69 (Conv2D)	(None, 12, 12, 32)	9248
activation_92 (Activation)	(None, 12, 12, 32)	0
max_pooling2d_69 (MaxPooling	(None, 6, 6, 32)	0
dropout_50 (Dropout)	(None, 6, 6, 32)	0
flatten_23 (Flatten)	(None, 1152)	0
dense_63 (Dense)	(None, 50)	57650
activation_93 (Activation)	(None, 50)	0
dropout_51 (Dropout)	(None, 50)	0
dense_64 (Dense)	(None, 1)	51
dense_65 (Dense)	(None, 3)	6

Total params: 87,083

Figure 4. CNN summary

Optimizer. Different optimizer for comparison is intended to get a best performance on image classification. To get the best performance with less number of epoch, I focus on two optimizers which are Adam and SGD. In theory, the optimizer Adam will remember both previous change of gradient and current change of gradient. Also, it can converge to local minimum faster compare to SGD. The optimizer SGD that only take previous training result as references, so it will take more time to train but this cause the better results in achieving local minimum. I compare the performance of these two optimizer by using them on the CNN model I mentioned above.

3. Evaluation

The following sections explain the results of computer vision and deep learning techniques that I'm going to implement in this paper.

3.1. Software and Hardware Environments

I use pelican server with GTX1080 hardware environment. Tensorflow [2], OpenCV [1], and Tensorboard [2]are used for training datasets and visualization work.

3.2. Parameters Setup

Input parameters. image will be resized to 64x64x1 for the case in edge detection. The scales in matrix will be either 0 or 255. Therefore, CNN model can train and predict the location of those non-zero pixel.

Learning Rates. I will use Tensorflow's default setup and learning rates = 0.001. Since 0.001 can converge to local or global minimum in this datasets, I won't change this parameter.

Batch Size. I will use Tensorflow's default setup and batch size = 32. Since 32 can converge to local or global minimum and training time isn't not the issue in this datasets, I won't change this parameter.

Number of Training Epochs. Number of training epochs will set to be 100. However, tensorflow provides early stop function which may cause program stop earlier.

3.3. Evaluation Metrics

To evaluate the performance of training results. Top k catrgorical accuracy can calculates the top-k categorical accuracy rate, i.e. success when the target class is within the top-k predictions provided(Figure 5).

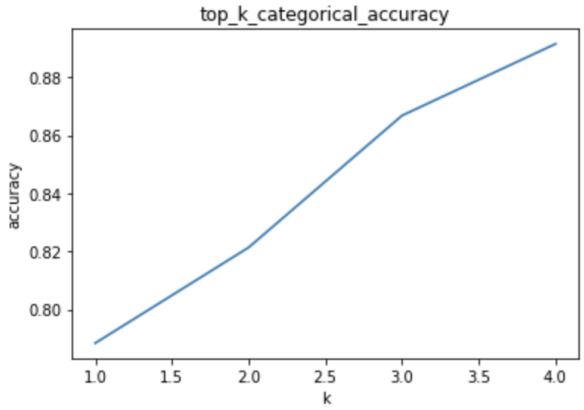

Figure 5. top-k predictions

3.4. Datasets

Dogs, cats, and human face datasets are three classes classification problem. The three classes have a lot of com-

mon features (e.g., eyes, nose, mouth).

Ground Truth. One-Hot labeling method.

Number of training and test images. 9000 for training sets and 3000 for testing sets.

Number of classes. three classes

3.5. Baseline approach

Baseline approah will use only 4500 training sets and 1500 testing sets. Data won't do any image preprocessing and CNN model won't use dropout method either. Optimizer will be used Adam. Highest accuracy:training=85.51%, validate=71.5%. Figure 6 shows the validation accuracy,validation loss, and time of the traninig results.

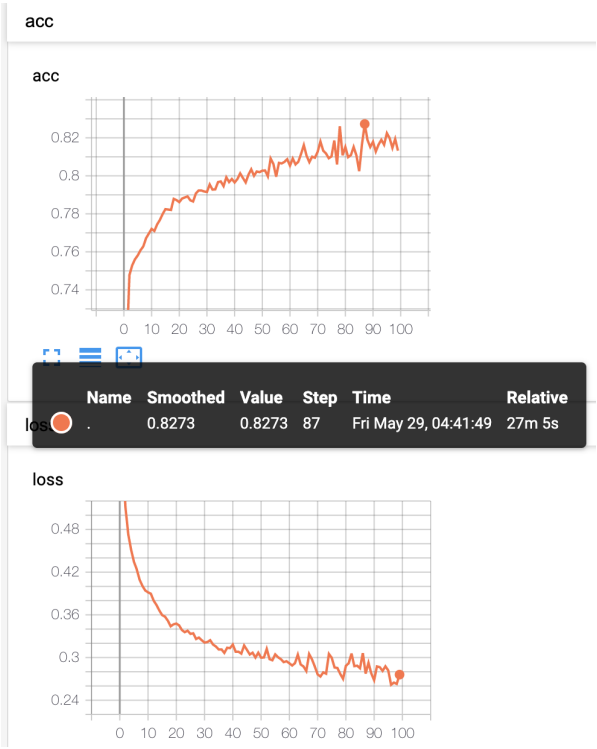

Figure 6. Baseline model training results

3.6. Optimizer Comparison

In this section, I use SGD optimizer to check that if it can show better results compared to Adam optimizer. From Figure 7, I concluded that SGD is slightly better than Adam when epoch becomes larger and larger. However, SGD sometimes traps in saddle points. Therefore, the following section will still use Adam as my optimizer.

3.7. Image Detection Methods Comparison

In this section, I will make comparison for three detection methods(edge, colour, gray-scale) and original pixel-based method. Figure 8 shows four methods' validation accuracy and loss. Figure 9 summarizes overall performance of four methods. Considering time and accuracy performance for the methods under the same CNN model, I concluded that edge is the most efficient method because this method use less time but can perform same as other methods. On the other hand, pixel-based method is time consuming because of large size of input data.

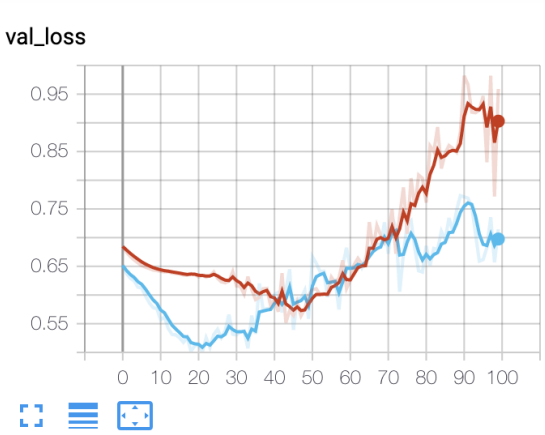

Figure 7. optimizer comparison between Adam(red) and SGD(blue)

Figure 8. **Image Detection Methods** colour(green), gray-scale(gray), pixel-based(orange), edge(red)

	Validation Loss	Validation Accuracy	Training Time	Epoch
Edge	0.3371	82.62%	35m 34s	100
Colour	0.2999	83.01%	1h 11m 11s	100
Gray-scale	0.3171	82.54%	1h 2m 3s	100
Pixel-based	0.3267	78.52%	1h 0m 33s	10

Figure 9. Table for Image Detection Methods

3.8. Datasets Visualization

Figure 10 shows correctly predicted sample and the corresponding ground truth. It's an amazing good result that edge detection can detect correct edge for human, dog, and cats.

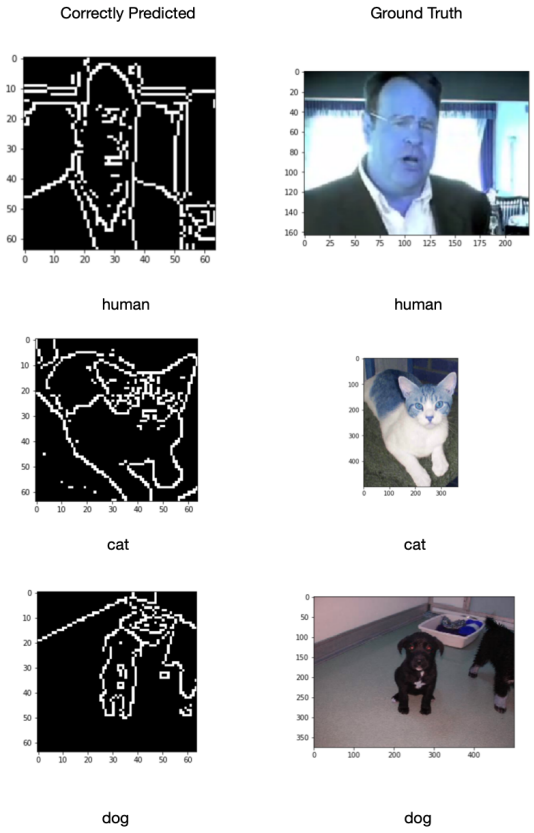

Figure 10. Correctly Predicted V.S. Ground Turth Visualization

Figure 11 shows incorrectly predicted sample and the corresponding ground truth. The predicted sample looks intuitive for human eyes but machine framework.

4. Conclusions

This paper make a comparison of Adam and SGD optimizers. In three classes dataset , I obtained highest 83% test accuracy merely use 9000 training sets. What's more, I try different detection methods and evaluate the performace

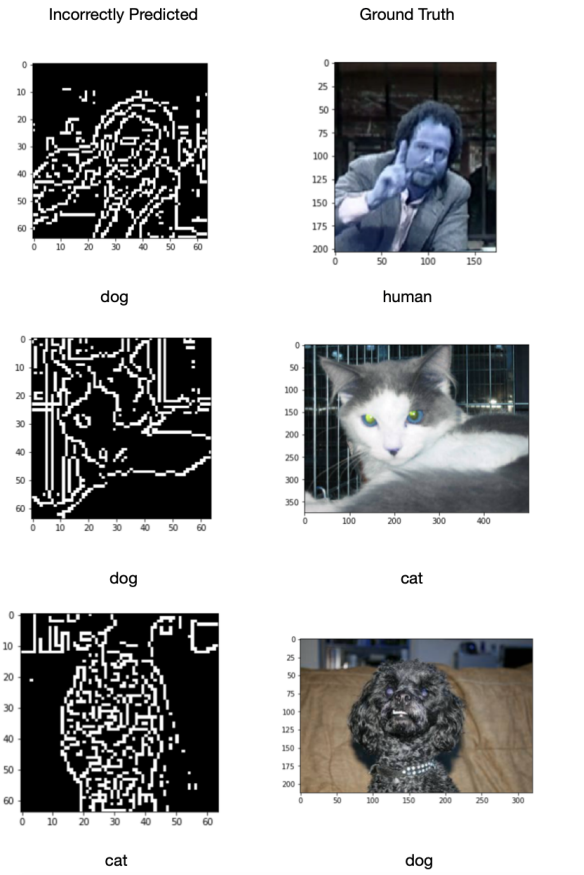

Figure 11. Incorrectly Predicted V.S. Ground Turth Visualization

of them. Overall, this paper summarizes various detection methods and optimizer to further improve the performance in image classification.

5. Future Works

Future work will focus on the structure of models, I can try to use more advanced model like efficientNet and DenseNet. Also, recent published optimizer 'Ranger' is another good direction I can work on.

References

[1] https://opencv.org/

[2] https://www.tensorflow.org/