Assignment 1

Tingjun Yuan, Muhammad Malik

Abstract

This assignment aims to train a CNN model on a dataset of two classes. This model is used to determine whether a pet image is a cat or a dog, and calculate the probability of each cases. We first trained the model on the Cats and Dogs dataset according to an online tutorial [2], trained a separate model on the Stanford Dogs dataset with some modification. Then, we replaced some convolutional layers and compare the performance between models. This report discusses the result of the trained models and reasons behind these.

Result and Evaluation

This assignment is divided into four experiments. Here are the results and evaluations of each experiment:

Experiment 1 This experiment simply follows the Keras tutorial [2] with the original parameter, except that the learning rate is set to 0.0001 (i.e. 1e-4). For the test image shown in the Notebook, the model gives 85.44% confidential rate that the image represents a cat. Next, we trained a separate model on the Stanford Dogs dataset [3]. This model has 120 classes, which is unsuitable for solving the cats-and-dogs problem. This model based on the Stanford Dogs dataset is subject to change in the following experiments.

Experiment 2 This experiment is to modify only the output layer of the model based on the Stanford Dogs dataset. Since the model has the output shape `(None, 120)`, which is incompatible with the dataset with 2 classes, we modified the `Dense` output layer to make it has only 1 output unit `(None, 1)`, and changed its activation function into `selu`. We then train the modified model for 50 epochs. The performance of the last epoch is:

```
acc: 0.9825 - loss: 0.1180 - val_acc: 0.9531 - val_loss: 0.1952
```

and it gives that there is a probability of 85.04% that the cat image shown in the Notebook is a cat, and 99.99% that the dog image shown is a dog.

Experiment 3 This experiment is to replace the output layer of the model, as well as the first two convolutional layers. The modification of the output layer is the same as Experiment 2. The first two `Conv2D` convolutional layers, namely `conv2d_128` and `conv2d_256`, which previously used `linear` activation function, now changed to use `selu`. Again, we trained the modified model for 50 epochs, and the performance of the last epoch is:

```
acc: 0.9710 - loss: 0.1376 - val_acc: 0.9416 - val_loss: 0.2091
```

For the test images shown in the Notebook, it gives 85.24% that the cat iamge is a cat, and 99.99% that the dog image is a dog. However, the validation accuracy becomes lower, and the validation loss becomes higher. The fact that the validation loss is slightly higher than the training loss signals that the model becomes slightly more overfitting.

Experiment 4 This experiment is to replace the output layer of the model, as well as the last two convolutional layers. Again, we modified the output model the same way as Experiment 2. This time, we keep the other convolutional layers unchanged, and modified the last two `Conv2D` convolutional layers, namely `conv2d_512` and `conv2d_728`. These layers are changed to use `selu`, as well as the previous 2 experiments. After 50 epochs of training, the performance of the last epoch is:

```
acc: 0.9786 - loss: 0.1249 - val_acc: 0.9346 - val_loss: 0.3121
```

For the test images shown in the Notebook, it gives that the test cat image is 85.30% a cat and the test dog image is 99.73% a dog. However, the validation accuracy becomes even more lower, and the validation loss becomes even more higher than the one with Experiment 3. The fact that the validation loss is extremely higher than the training loss signals that the model becomes extremely more overfitting.

Discussion and Conclusion

In the original model structure in Experiment 1, the `Conv2D` layers mentioned in the last section are all `linear`, which is simply:

$$f(x) = x \tag{1}$$

This behaves quite well, and it indeed has good performance. However, this has some limitations on accuracy. As a comparasion, we used another activation function called Scaled Exponential Linear Unit (`selu`) [1], which is:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha \cdot \left(\exp\left(\frac{x}{\alpha}\right) - 1\right) & \text{if } x < 0 \end{cases} \tag{2}$$

where $\alpha = 1$ by default. In this case, we used the default value. According to the result, although that the accuracy stays almost the same, but after replacing the activation functions with `selu`, the performance of the models become worse. This might because that the default α value provided by TensorFlow makes the function value nearly constant, and also makes it the same as the `elu` activation function. According to [4], maybe $\alpha = 1.6733$ is a better choice.

We have also tried `relu` with default parameters, but the model behaves even more worse – the model predicts every image as 50% probability of being either a cat or a dog.

In conclusion, to train a good model, it is important not only to choose a right activation function, but also tune the parameters in a right way.

References

[1] Jonathan T. Barron. Continuously differentiable exponential linear units, 2017.

[2] fchollet. Image classification from scratch, September 2023.

[3] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

[4] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks, 2017.