Apply CNN on Dog vs Cat

Yiran Lyu
SIT-628 Final Project
CWID: 10441886
`ylyu3@stevens.edu`

April 27, 2020

Abstract

This report is for an individual project of the deep learning course. This project would judge if one image is dog or cat according to the model trained with the dataset of Dog vs. Cat project on Kaggle platform.The problem to be solved by the project is actually an image classification problem in the field of computer vision. This project choose VGG-19, a common CNN model to solve the problem.

1 Introduction

In just the past five years, deep learning has taken the world by surprise, driving rapid progress in fields like computer vision, natural language processing, statistical modeling and reinforcement learning. Every machine learning problem needs to build a model for transforming data that computer can learn from, find a loss function that quantifies the badness of model and program an algorithm to adjust the model's parameters for minmizing the loss. This project would also use these core components to solve a deep learning problem with contents included in course.

2 Related Work

Image classification is a popular topic for the field of deep learning, since most of time, it is absolutely easier for human to recognize what is on an image with human eyes than the computer. At present,The most popular solution for image classification is to use Convolutional Neural Network(CNN). Convolutional Neural Network is a multi-layer neural network that uses multiple convolutional layers and pooling layers with activation function like ReLU function to extract image features (called a feature map). After the multiple fully connected layers and using softmax, it can get the final classification result with final probability.

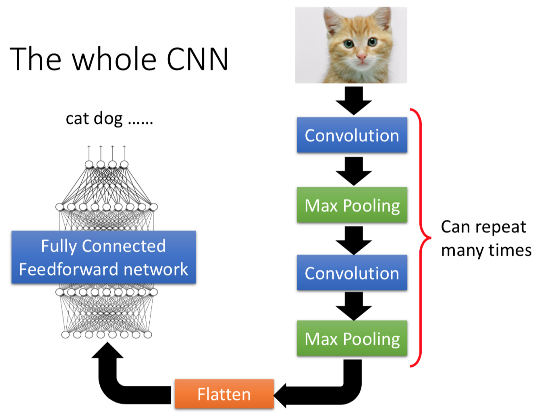


Figure 1: The whole CNN

Since 1998 with LeNet-5 model appearing, Convolutional Neural Network becomes more and more complex with higher accuracy. There are many common CNNs. Alexnet is the first large-scale network deployed to beat conventional computer vision methods on a large-scale vision challenge. The VGG network makes use of a number of repeating blocks of elements. GoogleNet makes use of networks with parallel concentrations and ResNet(Residual Networks) is currently the most popular go-to architecture today.

3 Algorithm

3.1 Dataset descrption and data preparation

The dataset is download from one competition on Kaggle platform called Dogs vs. Cats Redux: Kernels Edition. The train folder contains 25,000 images of dogs and cats. Each image in this folder has the label as part of the filename, such as cat.5.jpg and dog.100.jpg. The test folder contains 12,500 images, named according to a numeric id with no label or category in the file name, such as 1000.jpg. For more convenient and better training model, it is necessary to choose part of images in train folder as valid folder that the number of dog images and cat images are same. Since it needs too much time to run algorithm with too many iterations when the dataset is large, firstly only using 84 images as training data and 42 images as validation data is better to train the model. For the complete dataset, there would be 18000 images as training data and 7000 images as validation data.

3.2 Data preprocessing

Images are different with each other from some aspects. First, specifying the data path and transform them. Resizing each image as 224*224 and transforming the data shape as FloaderTensor.The value of the picture is in the interval [0,255]. In order to converge quickly during training and avoid saturation of the activation function, the value of the picture needs to be converted to the interval [0,1], which is called the normalization of data. An example of normalizing part of a picture's data is as follows:

3.3 Data visualization

Loading the processed data and try to visualize these tensor shape data. Normalizing input data to (0,1), then using function (x-mean)/std distribute each element to (-1,1). Changing the position of each dimension of tensor then limiting the element between minimum and maximum and ouputing a new tensor, after that, the data could be shown as image. Images in this dataset are colorful, which means these RGB images on red channel, green channel and blue channel can also be visualized separately. Graying images on each channel can also obtain three gray images. Choose one image as example.

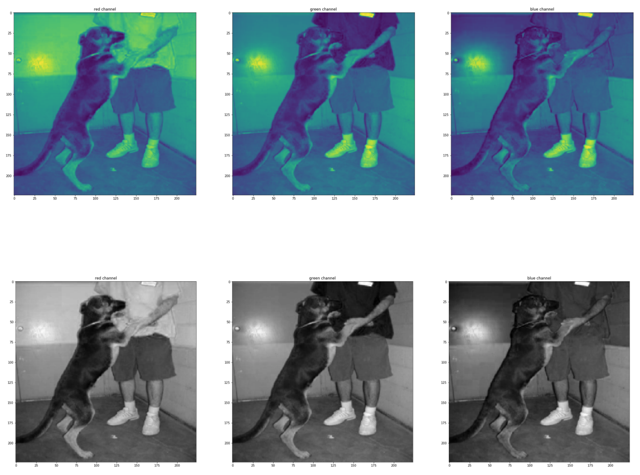
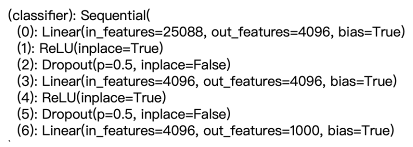


Figure 2: Three channels of RGB image

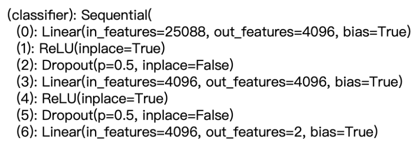
3.4 Model training and validating

VGG-19 model There are many well-known CNNs, but all of them have a deep network layer and a large number of parameters, which results in a very large amount of calculation and requires a long training time. Usually, using GPU to accelerate training procedure is required. In actual use, a pre-trained model trained on a large-scale data set is generally used for fine tuning (transfer learning). In this way, not only can the trained parameters be used, but also the training time can be greatly reduced. This project downloads and uses the pretrained VGG-19 model.This model have mulitiple layers for Conv layers and pooling layers with ReLU function, several times max pooling and one time average pooling. At the last layer, the pretrained model output 1000 features or catagory. In this project, we only have dog and cat two catagories. Therefore, we retain the preceding layers for feature extraction and only train the last layer, changing the model parameter, output features, as two.



```
(classifier): Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace=True)
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace=True)
  (5): Dropout(p=0.5, inplace=False)
  (6): Linear(in_features=4096, out_features=1000, bias=True)
```

(a) original parameter



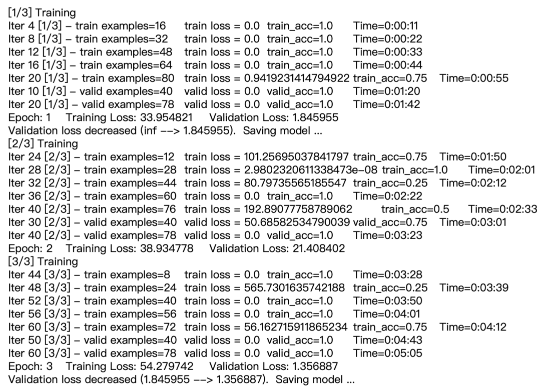
```
(classifier): Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace=True)
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace=True)
  (5): Dropout(p=0.5, inplace=False)
  (6): Linear(in_features=4096, out_features=2, bias=True)
```

(b) trained parameter

Figure 3: Changed VGG-19 model

Loss function and optimizer Choosing the cross entropy loss function and using Adam as the optimizer. Set learning rate as 0.01.

Train and validate the model To train a model, it needs clearing the gradients of all optimized variables, computing predicted outputs by passing inputs to the model, calculating the batch loss, computing gradient of the loss with respect to model parameters, updating parameter and traing loss. For each iteration, comparing the real and predicted outputs computes the training accuracy. Validating model is similar with np.lnf to trach change in validation loss, then saving the model when valid loss value is smaller then the best one. New model has been saved as 'model_vgg19.pth'. The iteration process with small dataset is as follows.

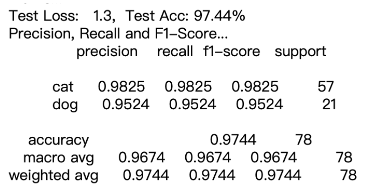


```
[1/3] Training
Iter 4 [1/3] – train examples=16     train loss = 0.0  train_acc=1.0      Time=0:00:11
Iter 8 [1/3] – train examples=32     train loss = 0.0  train_acc=1.0      Time=0:00:22
Iter 12 [1/3] – train examples=48    train loss = 0.0  train_acc=1.0      Time=0:00:33
Iter 16 [1/3] – train examples=64    train loss = 0.0  train_acc=1.0      Time=0:00:44
Iter 20 [1/3] – train examples=80    train loss = 0.9419231414794922 train_acc=0.75   Time=0:00:55
Iter 10 [1/3] – valid examples=40    valid loss = 0.0  valid_acc=1.0      Time=0:01:20
Iter 20 [1/3] – valid examples=78    valid loss = 0.0  valid_acc=1.0      Time=0:01:42
Epoch: 1    Training Loss: 33.954821     Validation Loss: 1.845955
Validation loss decreased (inf --> 1.845955).  Saving model ...
[2/3] Training
Iter 24 [2/3] – train examples=12   train loss = 101.25695037841797 train_acc=0.75   Time=0:01:50
Iter 28 [2/3] – train examples=28   train loss = 2.9802320611338473e–08 train_acc=1.0      Time=0:02:01
Iter 32 [2/3] – train examples=44   train loss = 80.79735565185547  train_acc=0.25   Time=0:02:12
Iter 36 [2/3] – train examples=60   train loss = 0.0  train_acc=1.0      Time=0:02:22
Iter 40 [2/3] – train examples=76   train loss = 192.89077758789062        train_acc=0.5      Time=0:02:33
Iter 30 [2/3] – valid examples=40   valid loss = 50.68582534790039 valid_acc=0.75   Time=0:03:01
Iter 40 [2/3] – valid examples=78   valid loss = 0.0  valid_acc=1.0      Time=0:03:23
Epoch: 2    Training Loss: 38.934778     Validation Loss: 21.408402
[3/3] Training
Iter 44 [3/3] – train examples=8    train loss = 0.0  train_acc=1.0      Time=0:03:28
Iter 48 [3/3] – train examples=24   train loss = 565.7301635742188  train_acc=0.25   Time=0:03:39
Iter 52 [3/3] – train examples=40   train loss = 0.0  train_acc=1.0      Time=0:03:50
Iter 56 [3/3] – train examples=56   train loss = 0.0  train_acc=1.0      Time=0:04:01
Iter 60 [3/3] – train examples=72   train loss = 56.162715911865234 train_acc=0.75   Time=0:04:12
Iter 50 [3/3] – valid examples=40   valid loss = 0.0  valid_acc=1.0      Time=0:04:43
Iter 60 [3/3] – valid examples=78   valid loss = 0.0  valid_acc=1.0      Time=0:05:05
Epoch: 3    Training Loss: 54.279742     Validation Loss: 1.356887
Validation loss decreased (1.845955 --> 1.356887).  Saving model ...
```

Figure 4: Iteration process

3.5 Evaluate the model

Define evaluate function to predict data and return the accuracy and average loss. Loading the trained model and setting thatthe last layer has two catagories as the model waiting for evaluation, needs to be tested and analyzied. The model trained by the small dataset has 97.44% accuracy. The evaluating parameter is as follows.



```
Test Loss:  1.3,  Test Acc: 97.44%
Precision, Recall and F1-Score...
              precision    recall  f1-score   support

         cat     0.9825    0.9825    0.9825        57
         dog     0.9524    0.9524    0.9524        21

    accuracy                         0.9744        78
   macro avg     0.9674    0.9674    0.9674        78
weighted avg     0.9744    0.9744    0.9744        78
```

Figure 5: Evaluating parameter

We can also choose some data to visualize the validation result. Plot 20 images with their predicted label and real label in blancket. If the predict result is wrong, the labels would be shown as red, else green.

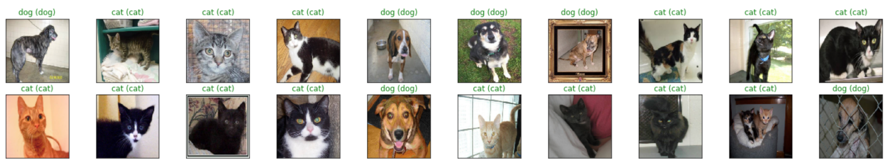


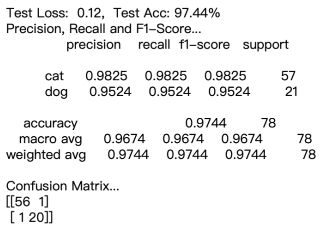
Figure 6: Predict result visualization

3.6 Predict test dataset with trained model

Load test data and transform it. Predict with trained model, then write the result in a csv file. Absolutely, it is another cost of time to run the predict process on CPU.

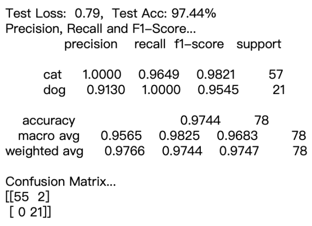
4 Comparison & Conclusion

Learning rate Usually, learning rate could be an important parameter to train one model. However, the dataset for training is very small. Test accuracy doesn't change whatever the learning rate is 0.1, 0.01 or 0.001. Even the other value like F-1 score is same in case when learning rate is 0.01 or 0.001. When the learning rate is 0.1, other parameters change but only a little. So just seems the model become stable after learnign rate is 0.01. Finally, when train the model with the whole dataset, it still trained with learning rate 0.1.



```
Test Loss:  0.12,  Test Acc: 97.44%
Precision, Recall and F1-Score...
              precision    recall  f1-score   support

         cat     0.9825    0.9825    0.9825        57
         dog     0.9524    0.9524    0.9524        21

    accuracy                         0.9744        78
   macro avg     0.9674    0.9674    0.9674        78
weighted avg     0.9744    0.9744    0.9744        78

Confusion Matrix...
[[56  1]
 [ 1 20]]
```

(a) learing rate = 0.01

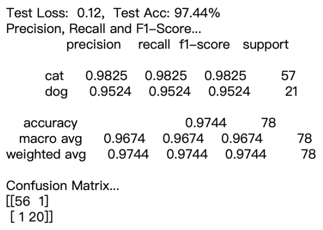


```
Test Loss:  0.79,  Test Acc: 97.44%
Precision, Recall and F1-Score...
              precision    recall  f1-score   support

         cat     1.0000    0.9649    0.9821        57
         dog     0.9130    1.0000    0.9545        21

    accuracy                         0.9744        78
   macro avg     0.9565    0.9825    0.9683        78
weighted avg     0.9766    0.9744    0.9747        78

Confusion Matrix...
[[55  2]
 [ 0 21]]
```

(b) learning rate = 0.1

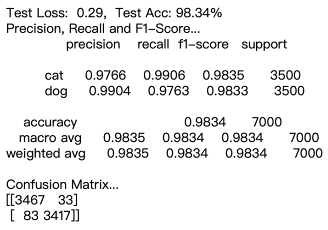
Figure 7: Tiny change when learning rate is different

Although it costs so much time to train a model with the whole dataset, the results is good. Test accuracy is higher when the dataset is larger. It is 98.34 %. The concrete parameters are as follow.



```
Test Loss:  0.12,  Test Acc: 97.44%
Precision, Recall and F1-Score...
              precision    recall  f1-score   support

         cat     0.9825    0.9825    0.9825        57
         dog     0.9524    0.9524    0.9524        21

    accuracy                         0.9744        78
   macro avg     0.9674    0.9674    0.9674        78
weighted avg     0.9744    0.9744    0.9744        78

Confusion Matrix...
[[56  1]
 [ 1 20]]
```

(a) small dataset



```
Test Loss:  0.29,  Test Acc: 98.34%
Precision, Recall and F1-Score...
              precision    recall  f1-score   support

         cat     0.9766    0.9906    0.9835      3500
         dog     0.9904    0.9763    0.9833      3500

    accuracy                         0.9834      7000
   macro avg     0.9835    0.9834    0.9834      7000
weighted avg     0.9835    0.9834    0.9834      7000

Confusion Matrix...
[[3467   33]
 [  83 3417]]
```

(b) lwhole dataset

Figure 8: Higher accuracy with larger dataset

Table 1: Comparision between small and large datasets

	small dataset	large dataset
accuracy	97.44%	98.34 %

This project is a good choice for reviewing the basic contents of Convulutional Neural Network and computer vision, what makes knowledge more clear. Since CNNs have mulitple layers and need a lot of time on computation and iteration, it makes hard to train the model and adjust parameters. On CPU, training model with the whole dog vs.cat dataset needs more than ten hours, however, this is not a very big dataset in actual use. Therefore, it seems better solve such problem on GPU. Fine tuning models are convenient and actually have been trained with a large scale of images. It let the users avoid thinking about the parameters between each layer.

How to improve the algorithm? First, there is still space to improve VGG-19 model. Do more testing with different parameter and choose the best one.

Second, each CNN model has it pros. It would be great to try another CNNs model on same dataset. Multiple models applying can provide different result for comparing. Third, it would be more accurate if we combine several models together to find the final result with summation or weighting. This way can combine advantage of each model and improve the accuracy of final model.

References

[1] Aston, Zhang & Zachary, C.Lipton (2020)Dive into Deep Learning.

[2] Kaggle Inc. Dogs vs. Cats Redux: Kernels Edition | Kaggle [EB/OL].

[3] Simonyan, K & Zisserman, A (2014) Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556.