Distinguish real and fake shoes

Zheng Wu
Computer Science
New Mexico State University
Las Cruces, New Mexico
zwu@gmail.com

Jiefei Liu
Computer Science
New Mexico State University
Las Cruces, New Mexico
jiefei@nmsu.edu

Abstract

Nowadays, more and more counterfeits were made. Most of the fake products were made like the real, even the identification code. So there have lots of people working on product recognition, there are lots of companies and small Apps help people to distinguish whether or not the product is fake, but all of the products were distinguished by a human. Therefore, those Apps and companies put a lot of human resources on the product recognition. During this project, We will propose a way to help those companies and Apps reduce utilizing the human resources. During this project we will utilize the Convolutional neural network(CNN) to predict the results.

1. Instruction

We found that there has lots of people would like to let the third party certificate what they just bought, especially, the very expensive goods. Zheng Wu was designing an App called "FakeOffUS", you can find it in the Apple App Store, This App is helping people distinguish you bought the shoes are fake or real. And also there are lots of Apps and companies are helping people distinguish the counterfeits. But as we know, all kinds of these recognitions are worked by human, therefore the companies need to hire lots of people to distinguish the fake or real. The user needs to take several pictures of the shoes which contains the front, side, back, the tongue of shoes, inside of shoes, insole, box label, and box cover and upload those pictures through the App. the background crew will split the pictures and go through the picture one by one. That takes lots of human resources to split and distinguish the pictures. And also we know that some of the counterfeits were made like the real, and it needs to carefully identify that still need to done by a human, but there are lots of counterfeits are easy to distinguish, so, we will propose a way to help companies and Apps reduce the utilization of the human resource. The System can help them split the different part of the shoes, and distinguish the real or fake. Due to the machine learning disadvantage, we can not guarantee that the predicted accuracy is 100%, but this system still can help them to reduce the large human resources.

I attach the sample input here, which are real shoes and fake shoes:

Figure 1: Real shoes

Figure 2: fake shoes

2. CNN

Definition:

"The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers."

During this project, we propose a way to get them as much high accuracy as we can. And also we tried to use CNN to split the pictures too, which also can save human resources.

3. Experimental

For the dataset, we collected more than 2000 pictures, 1000 for the real dataset, and 1000 for the fake dataset, which the sample I showed in Figure 3 and Figure 4. We utilized the Google Colab to do this project, and also due to the large size of the dataset, we requested a higher GPU and a bigger size of memory.

When we split the dataset into the different files it takes a very long time, we need to split it one by one, so

we were thinking, is that can use CNN to split the pictures too? So, we implement another CNN method, in order to split the pictures more easily.

Therefore, there have two methods in this project.

The first method

The first method is to predict the shoe pictures by utilizing CNN. This method can predict which part of the shoes is like the Figure 3 shows. and it can help the background crew to split the pictures.

Figure 3: Split samples

Model: "sequential_9"

Layer (type)	Output Shape	Param #
conv2d_18 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_18 (MaxPooling	(None, 74, 74, 32)	0
conv2d_19 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d_19 (MaxPooling	(None, 12, 12, 64)	0
flatten_9 (Flatten)	(None, 9216)	0
dense_18 (Dense)	(None, 512)	4719104
dense_19 (Dense)	(None, 7)	3591

Total params: 4,742,087
Trainable params: 4,742,087
Non-trainable params: 0

Figure 4: First testing case model summary

The first test case, we have two conv_2d layers and two pooling layers, the input shape is (148, 148, 32), we have the model summary shows in Figure 4.

```
Epoch 390/400
25/24 [==============================] - 7s 268ms/step - loss: 0.6199 - accuracy: 0.7864
Epoch 391/400
25/24 [==============================] - 7s 266ms/step - loss: 0.6248 - accuracy: 0.7838
Epoch 392/400
25/24 [==============================] - 7s 267ms/step - loss: 0.6420 - accuracy: 0.7623
Epoch 393/400
25/24 [==============================] - 7s 265ms/step - loss: 0.6491 - accuracy: 0.7799
Epoch 394/400
25/24 [==============================] - 7s 264ms/step - loss: 0.6461 - accuracy: 0.7734
Epoch 395/400
25/24 [==============================] - 7s 263ms/step - loss: 0.6809 - accuracy: 0.7714
Epoch 396/400
25/24 [==============================] - 7s 266ms/step - loss: 0.6444 - accuracy: 0.7812
Epoch 397/400
25/24 [==============================] - 7s 261ms/step - loss: 0.5951 - accuracy: 0.7870
Epoch 398/400
25/24 [==============================] - 7s 266ms/step - loss: 0.6232 - accuracy: 0.7825
Epoch 399/400
25/24 [==============================] - 6s 257ms/step - loss: 0.6182 - accuracy: 0.7740
Epoch 400/400
25/24 [==============================] - 6s 257ms/step - loss: 0.6651 - accuracy: 0.7799
```

Figure 5: First testing case training accuracy

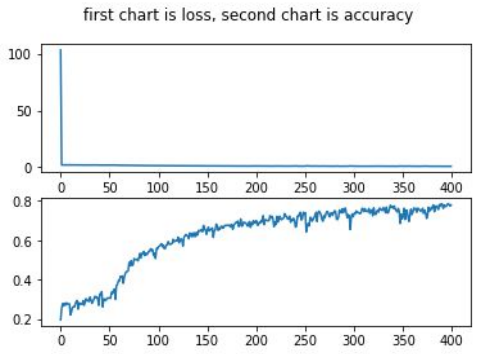

Figure 6: loss/accuracy curve

For this testing case we set 400 epoch and the training accuracy can reach 77.9%, and the accuracy of the testing is 63%, the overfitting happened. And from the Figure 6, we can see that the loss curve can reduce to the low level, and the training accuracy curve can reach the 0.8 but it has the noise on it, that because the background of the image might affect the training curve.

In the second test case, we adjusted the size of the input data size to be 10, and combined 5 models, the results show in Figure 7

```
#start testing....
print("testing data/1————————",test_accuracy("test_data/1"))
print("testing data/2————————",test_accuracy("test_data/2"))
print("testing data/3————————",test_accuracy("test_data/3"))
print("testing data/4————————",test_accuracy("test_data/4"))
print("testing data/5————————",test_accuracy("test_data/5"))
print("testing data/6————————",test_accuracy("test_data/6"))
print("testing data/7————————",test_accuracy("test_data/7"))
print("testing data/8————————",test_accuracy("test_data/8"))
print("testing data/9————————",test_accuracy("test_data/9"))
print("testing data/10————————",test_accuracy("test_data/10"))
```

```
[[0.00105545]]
c3 0.0
testing data/1———————— real
[[0.11343185]]
c3 0.0
testing data/2———————— real
[[0.19613656]]
c3 0.0
testing data/3———————— real
[[0.52744406]]
c3 1.0
testing data/4———————— fake
[[0.00022731]]
c3 0.0
testing data/5———————— real
[[0.99844366]]
c3 1.0
testing data/6———————— fake
[[0.08806472]]
c3 0.0
testing data/7———————— real
[[0.806854]]
c3 1.0
testing data/8———————— fake
[[0.00113818]]
c3 0.0
testing data/9———————— real
[[0.57130593]]
c3 1.0
testing data/10———————— fake
```

Figure 7: Second testing result

We can see that the prediction only has one mistake, therefore the accuracy can reach 90%.

The second method

The second method is to predict the shoes real or fake also utilized by CNN, this method can filter the most fake shoes, which can save lots of human resources.

The first meritorious testing case, we have two conv2d layers and 2 pooling layers, the input shape is (148, 148, 32), we have the model summary shows in Figure 8.

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_6 (MaxPooling2	(None, 74, 74, 32)	0
conv2d_7 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d_7 (MaxPooling2	(None, 6, 6, 64)	0
flatten_3 (Flatten)	(None, 2304)	0
dense_6 (Dense)	(None, 512)	1180160
dense_7 (Dense)	(None, 7)	3591

Total params: 1,203,143
Trainable params: 1,203,143
Non-trainable params: 0

Figure 8: First testing case model summary

```
Epoch 90/100
6/6 [==============================] - 0s 16ms/step - loss: 1.1231e-04 - accuracy: 1.0000
Epoch 91/100
6/6 [==============================] - 0s 16ms/step - loss: 1.0940e-04 - accuracy: 1.0000
Epoch 92/100
6/6 [==============================] - 0s 16ms/step - loss: 1.0684e-04 - accuracy: 1.0000
Epoch 93/100
6/6 [==============================] - 0s 15ms/step - loss: 1.0642e-04 - accuracy: 1.0000
Epoch 94/100
6/6 [==============================] - 0s 15ms/step - loss: 1.0150e-04 - accuracy: 1.0000
Epoch 95/100
6/6 [==============================] - 0s 16ms/step - loss: 9.9315e-05 - accuracy: 1.0000
Epoch 96/100
6/6 [==============================] - 0s 16ms/step - loss: 9.7267e-05 - accuracy: 1.0000
Epoch 97/100
6/6 [==============================] - 0s 16ms/step - loss: 9.6017e-05 - accuracy: 1.0000
Epoch 98/100
6/6 [==============================] - 0s 16ms/step - loss: 9.3417e-05 - accuracy: 1.0000
Epoch 99/100
6/6 [==============================] - 0s 16ms/step - loss: 9.1396e-05 - accuracy: 1.0000
Epoch 100/100
6/6 [==============================] - 0s 15ms/step - loss: 8.9406e-05 - accuracy: 1.0000
```

Figure 9: First testing case training accuracy

And for the first testing case from figure 9, we set the epoch as 100, and as you can see the training accuracy can reach 100%, but the testing accuracy only has 55%, the overfitting happened which is almost equal to the random guess. We analyzed the issues, there have several problems, for the dataset direction, the whole dataset might be too small or the training data was too much. For the model direction, The model layers and parameters might need to be adjusted.

The second meritorious test case, we adjust the input dataset, and adjust the parameters in the CNN model, and the model summary shows in Figure 10.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_4 (MaxPooling2	(None, 74, 74, 32)	0
conv2d_5 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d_5 (MaxPooling2	(None, 12, 12, 64)	0
flatten_2 (Flatten)	(None, 9216)	0
dense_4 (Dense)	(None, 512)	4719104
dense_5 (Dense)	(None, 2)	1026

Total params: 4,739,522
Trainable params: 4,739,522
Non-trainable params: 0

Figure 10: Second testing case Model summary

```
Epoch 490/500
2/1 [==============================] - 0s 46ms/step - loss: 0.0511 - accuracy: 0.9882
Epoch 491/500
2/1 [==============================] - 0s 47ms/step - loss: 0.0841 - accuracy: 0.9647
Epoch 492/500
2/1 [==============================] - 0s 53ms/step - loss: 0.0900 - accuracy: 0.9765
Epoch 493/500
2/1 [==============================] - 0s 45ms/step - loss: 0.0838 - accuracy: 0.9765
Epoch 494/500
2/1 [==============================] - 0s 47ms/step - loss: 0.1423 - accuracy: 0.9412
Epoch 495/500
2/1 [==============================] - 0s 42ms/step - loss: 0.1058 - accuracy: 0.9412
Epoch 496/500
2/1 [==============================] - 0s 48ms/step - loss: 0.0531 - accuracy: 0.9765
Epoch 497/500
2/1 [==============================] - 0s 47ms/step - loss: 0.0494 - accuracy: 0.9765
Epoch 498/500
2/1 [==============================] - 0s 46ms/step - loss: 0.2329 - accuracy: 0.9176
Epoch 499/500
2/1 [==============================] - 0s 43ms/step - loss: 0.1027 - accuracy: 0.9647
Epoch 500/500
2/1 [==============================] - 0s 46ms/step - loss: 0.0693 - accuracy: 0.9765
```

Figure 11: Second testing case training accuracy

We adjust the epoch to be 500, We can see that the training accuracy was decreased a little bit due to adjusting the input dataset, but the testing accuracy was increased to 72.7%, the overfitting still happened, but it is much better than the previous testing case.

The third meritorious test case, we adjusted the parameters in the CNN model, and the train accuracy shows in figure 12

```
Epoch 490/500
2/1 [==============================] - 0s 127ms/step - loss: 0.3168 - accuracy: 0.8824
Epoch 491/500
2/1 [==============================] - 0s 124ms/step - loss: 0.2741 - accuracy: 0.8471
Epoch 492/500
2/1 [==============================] - 0s 121ms/step - loss: 0.3377 - accuracy: 0.8235
Epoch 493/500
2/1 [==============================] - 0s 121ms/step - loss: 0.3117 - accuracy: 0.8706
Epoch 494/500
2/1 [==============================] - 0s 121ms/step - loss: 0.2541 - accuracy: 0.9412
Epoch 495/500
2/1 [==============================] - 0s 120ms/step - loss: 0.2475 - accuracy: 0.8941
Epoch 496/500
2/1 [==============================] - 0s 121ms/step - loss: 0.3051 - accuracy: 0.8824
Epoch 497/500
2/1 [==============================] - 0s 123ms/step - loss: 0.2192 - accuracy: 0.8941
Epoch 498/500
2/1 [==============================] - 0s 125ms/step - loss: 0.2536 - accuracy: 0.9059
Epoch 499/500
2/1 [==============================] - 0s 119ms/step - loss: 0.2246 - accuracy: 0.9176
Epoch 500/500
2/1 [==============================] - 0s 123ms/step - loss: 0.2760 - accuracy: 0.8706
```

Figure 12: third testing case training accuracy

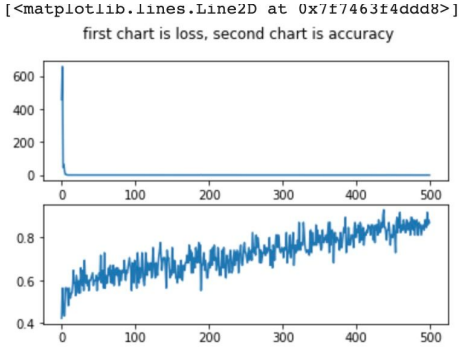

Figure 13: loss and accuracy curve

For this testing case, we achieved that the train accuracy to 87% and the test accuracy reached 77%. And from the loss and accuracy curve we can see that the loss curve can reach a very low level, and the curve of the training accuracy has noise, that because the input images are not the “clean image” the background of the image might affect the training curve. This is the best result we got. The overfitting still happened, but the results were much better than the first testing case. The final testing accuracy was not that much high, that due to the size of

the dataset, we only can collect more than 2000 pictures, because of the time limit of the semester-long project.

4. Conclusion

During this project, we generated two kinds of program, first one recognizes which part of shoes belongs to, the second one is to recognize the shoes fake or real.
The First method reached the very high accuracy. The second method did not reach the very high accuracy, the main reason is the size of the dataset was not big enough, and we tried our best to reduce the overfitting. Finally, we got much better results than the first testing case. So that this project can help distinguish counterfeits Apps and companies filter the customer uploaded images and save lots of human recourse.

5. Future to do

We will continue to collect the data, and let the program reach higher accuracy. This system might be used in the "FakeOffUS" App, as I mentioned at the beginning of this report.

Reference

[1] Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016). *Deep Learning*. MIT Press. p. 326.