Solution explanation

Three methods for this challenge

1. OCR: Since every logo has bank's name, at first I was thinking about use OCR package to extract text data from the document. Moreover, most bank statements have logo in header thus I can match bank's name to first few lines of the document. But this is not a learning algorithm so I didn't try.
2. ORB feature matching: I tried using ORB descriptors to extract features from two images (logo vs. document) and checked whether they matched. But it failed, the logo was matched to text in the document. It may because bank's logo is kind of simple and they shared similar characteristics with text (such as a curve or other things).
3. Classification model (CNN) + sliding windows: within limited time, the most straightforward algorithm for me is to train a classification model, classifying logos to banks. Then, I can use sliding windows with different sizes to scan the document and put every catch to the classification model. If the bank statement belongs to one of five banks, the sliding window will catch the logo and return the bank's name. Otherwise it will just return 'Other'.

Details of the classification model and sliding windows

1. Classification model (on logo, six classes):
    a. Dataset: I got 300 images (50 images per classes) from google and reshape them to the same size, splitting it to training data and testing data with 4:1 ratio. Then, I generate more data during training by using data generator tool in Keras.
    b. Model structure: since logo is not that complex, I first tried constructing model by myself, but the performance is not that good. Then I used vgg16 without top layer as the base model (not trainable) to extract features and add a flatten and two dense layers with one dropout as the final model. The result of the model will give six probabilities for each class.
2. Sliding windows:
    a. Since the sizes for those five logos are limited, the ratio between height and width is 1:1 or 1:1.5. Thus, I tried four sliding windows: (100,100), (100,150), (200,200), (200,300). These numbers are just the results of experiment.
    b. When there is a document, the sliding windows will scan the

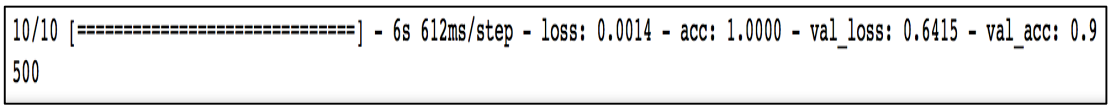
document from left to right, top to bottom (the logo is usually existing in the top of the bank statement) with stride 25. If the prediction of one catch is higher than 0.95 (except it belongs 'Other' class), I can just return that result, since I am sure that model detects the logo and the statement belongs to that bank. Otherwise, we can go through the whole image and return 'Other'.

The process of the prediction

1. Load image and change it to RGB mode since the model need three channels.
2. Check the whole image as it is a logo at first and see the result of the prediction.
3. If the image has a side longer than 1000, it may be a document, then using sliding windows to scan the document.
4. If the the document is too big (height > 2000 or width > 2000), in order to increase the efficiency, the image will be cropped to ¼*height with same width since the logo should be always on the top of the bank statement (it will make mistake sometimes, the defeat will be mentioned below). After tailoring, the image will be resized in order to increase computing speed (the height should no larger than 600 while the width should no bigger than 1600.
5. Scan the image with four sliding windows from left to right, top to bottom to predict every patch it catches. The program will return the classifying result.

Algorithm's performance

Since I only collected several bank statements, it is not easy to get the accuracy of the prediction for the bank statements (It passed all the bank statement test data downloaded from GitHub). However, the classification model on logos provides the glance of the performance. I tried different NN structure, optimizers, epochs, learning rate and so on, the best accuracy on validation dataset is 95%, which means it is good at classifying different logos. But it should also be considered that the size of testing data is small (10 images per classes).



```
10/10 [==============================] - 6s 612ms/step - loss: 0.0014 - acc: 1.0000 - val_loss: 0.6415 - val_acc: 0.9
500
```

Defeat of the Algorithm

1. When there is a document, the program will prepress the input and crop it. Thus, the document should be in the right direction (it can be rotated but shouldn’t be too much, otherwise it will be cropped).
2. If the logo is not in the top ¼ part, it will give the wrong answer.
3. The document should have an acceptable resolution. The sizes of the sliding windows are fixed and the logo should have at least 30 pixels in one side. Otherwise it is not easy to detect that logo.
4. Among those classes, the model performs worst in classifying Citi group logo and other texts in the document. It may because I put Citi group logo name as well as its logo together in the training dataset (I found another company which also had an umbrella logo, I thus put name and logo together in this case). Thus, if a document belongs to other banks (not those five banks), it may return Citi group since it detects texts in the bank statement. The model performs well in other classes.

Next step

1. As I mentioned above, model may misclassify Citi group logo and other texts in the document. It can be solved by collecting more data and training model with larger dataset. Moreover, the tuning on hyper-parameter should also be considered.
2. New Algorithm: it will be more accurate if I use a model to detect the logo directly. But it needs annotations of the box around the logo, which will take much longer time. If time allowed, more work should be done with this challenge.