Multi-Label Pedestrian Attributes Detection by CNN

Yidong Fu

April 12, 2020

1 Introduction

Financial frauds happened more and more frequently but, in many cases, they all started from inside. There is a newly invented fraud named Multiple Mortgages by the Same Collaterals. Some employees in a specific bank stole collaterals such as Title Deeds and used them to apply for mortgages in another bank. Since information is not shared between different banks, for those collaterals that have been mortgaged will likely get approval from other banks.

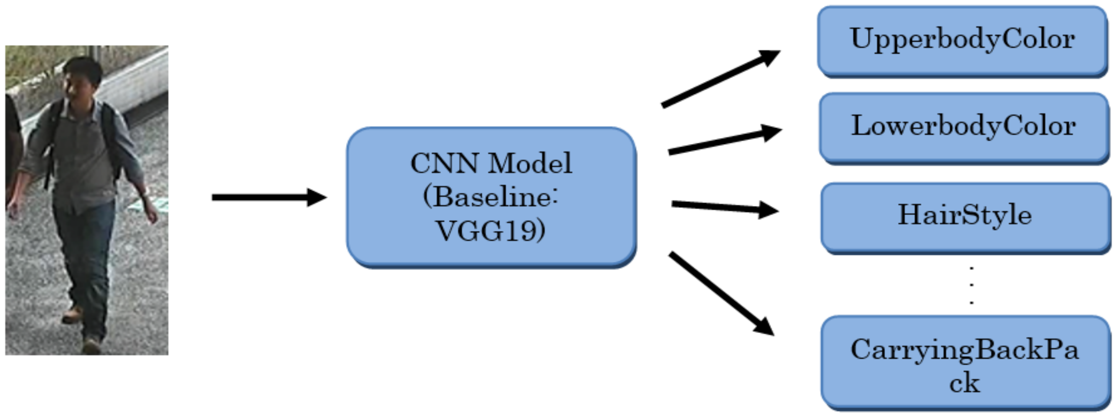


Figure 1: Model Overview

This has caused billions of losses in China. However, those approvers often found out the truth after a long period of time and they could not remember how exactly those cheaters looked like because they need to face hundreds of people every day. But they could remember some of their characters such as a man with a black backpack, or a woman with blonde hair. Therefore, I want to develop a multi-label image classifier to help people find who they are looking for.

2 Dataset Information

The data comes from PETA dataset which is developed by Yubin Deng and his team. ”It consists of 19000 images, with resolution ranging from 17-by-39 to 169-by-365 pixels. Those 19000 images include 8705 persons, each annotated with 61 binary and 4 multi-class attributes”[1].

It has numerous camera angles, viewpoints or resolutions and it’s hard to unify them, so I chose only to use CUHK dataset which has same resolution and camera angle for all images. The CUHK dataset contains 160×80 RGB images of 4563 pedestrians. It has 14-17 labels for each image such as Upperbody and Lowerbody styles, hairstyles, and so on.

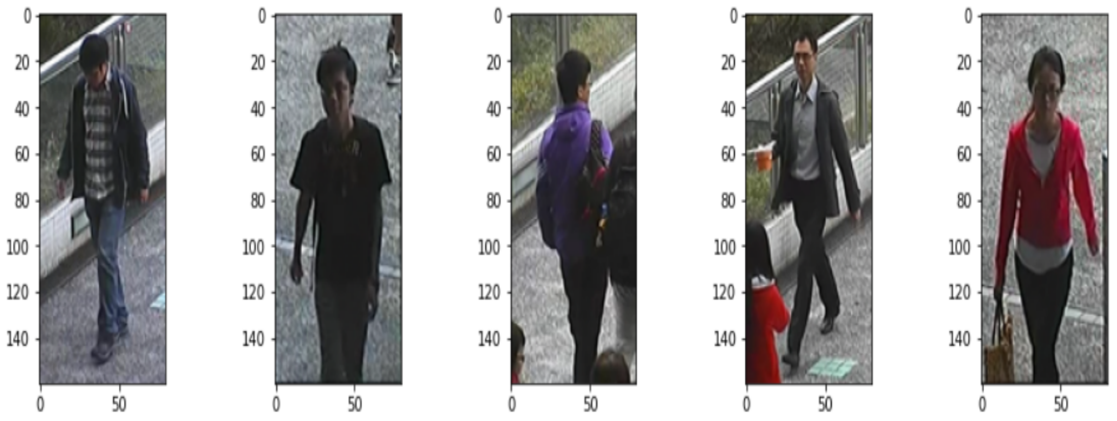


Figure 2: Data Examples

3 Single attribute with multiple labels

3.1 Carrying attribute Labelling

This part mainly focuses only on "carrying" attributes which indicate what people carry in the image, so I chose the specific 10 'carrying' categories which are 'carryingBabyBuggy', 'carryingBackpack', 'carryingFolder', 'carryingLuggageCase', 'carryingMessengerBag', 'carryingNothing', 'carryingOther', 'carryingPlasticBags', 'carryingSuitcase', 'carryingUmbrella',

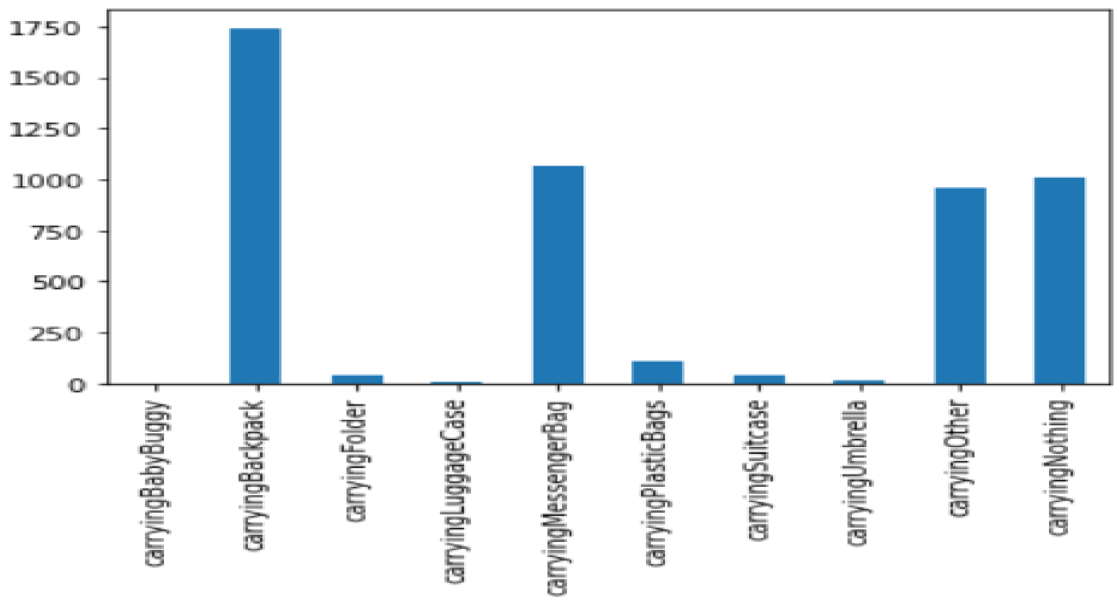


Figure 3: imbalanced carrying attributes

As we can see from the above Chart, the dataset is very imbalanced because 'carryingBabyBuggy', 'carryingPlasticBags', 'carryingFolder', 'carryingLuggageCase', 'carryingSuitcase', and 'carryingUmbrella' have lower than 300 samples but other categories have at least 1000 samples. As long as samples are less than 10% of the whole dataset, we do not think any model could classify those labels correctly. Therefore, I would categorize these three categories as 'carryingOther'.

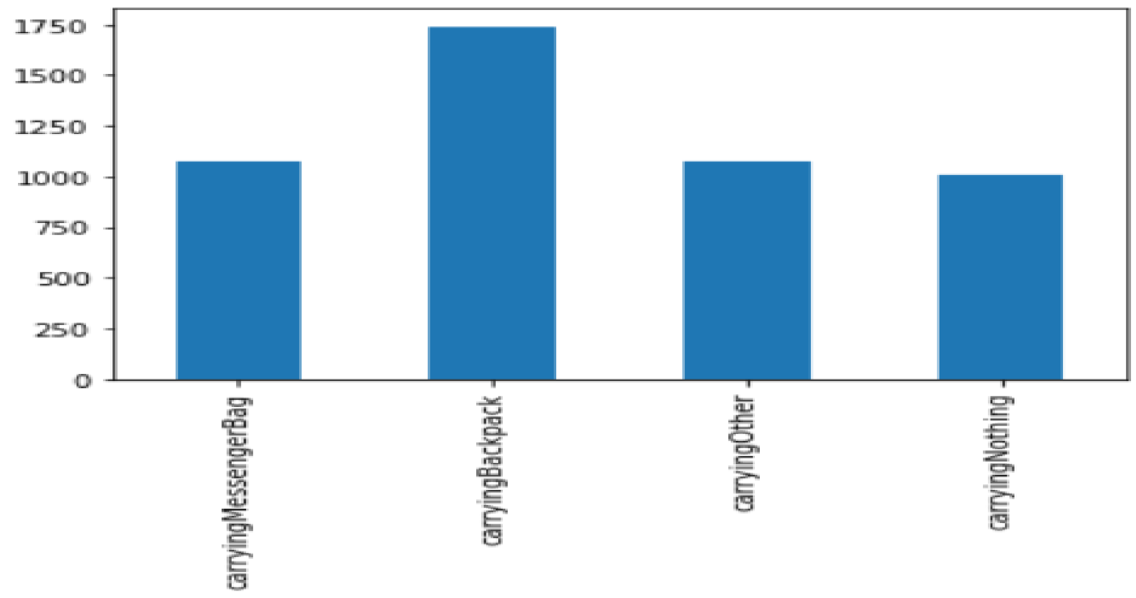


Figure 4: balanced carrying attributes

3.2 Model Preparation

To train the model, we implemented of the VGG19 as my baseline model for the reason that it's famous for image classifying. 4563 images were split into training and test samples by 0.1. Therefore, we have 4107 training samples and 456 test samples. And we further split 4107 training samples into 3286 training and 821 validation samples.

Data Augmentation is implemented since carrying attributes are hard to be detected. For example, if a pedestrian wore black jacket and carried a black backpack, it would be really hard for machine to learn that because black jacket and black backpack look very similar to each other. Not only that, sometimes attributes might be occluded because of camera angle or other obstacles.

The purpose of data augmentation here is trying to eliminate appearance ambiguity as much as possible and highlight the true distinctive feature. Four augmentation methods are carefully chosen to fulfill this purpose.

(a) rotation_range=20. Rotate image randomly so that a backpack might be rotated so it looks bigger in the image.

(b) zoom_range=0.4. Zooms in the crucial part of the image and makes a particular attribute much bigger and easier to be detected. It also wipes out irrelevant features in the image such as backgrounds or other pedestrians, so this step is crucial to the data augmentation.

(c) shear_range=0.5. Fix the x-axis pixel and move y-axis pixel horizontally. It has similar effect as rotation_range.

(d) horizontal_flip=True. Flip images horizontally which produce more images to train in the model.

As a result, the carrying attributes are more obvious after data augmentation is implemented.

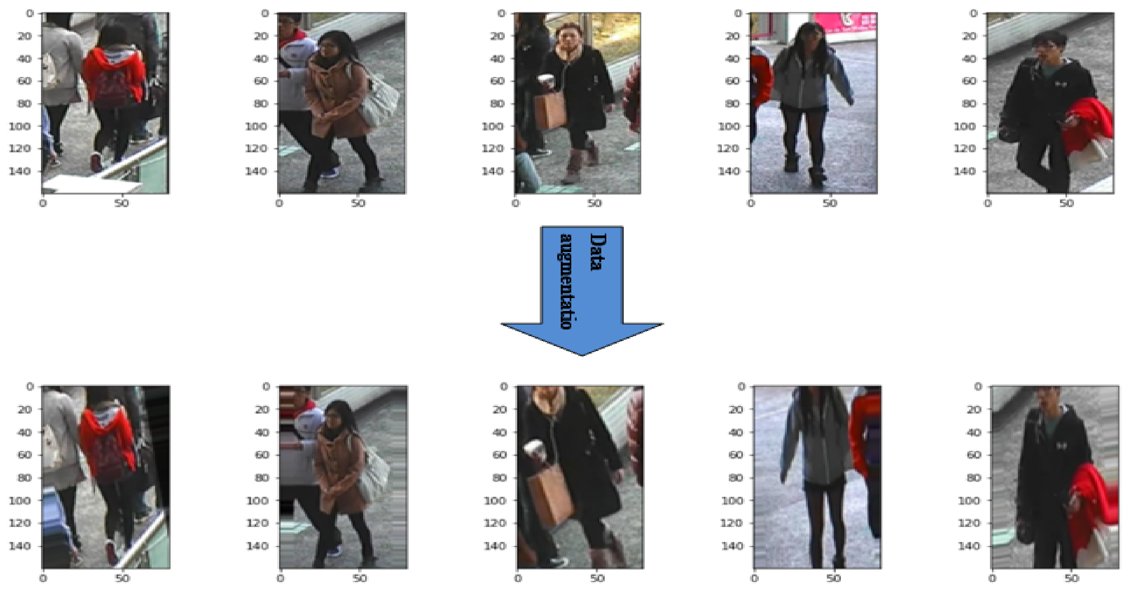
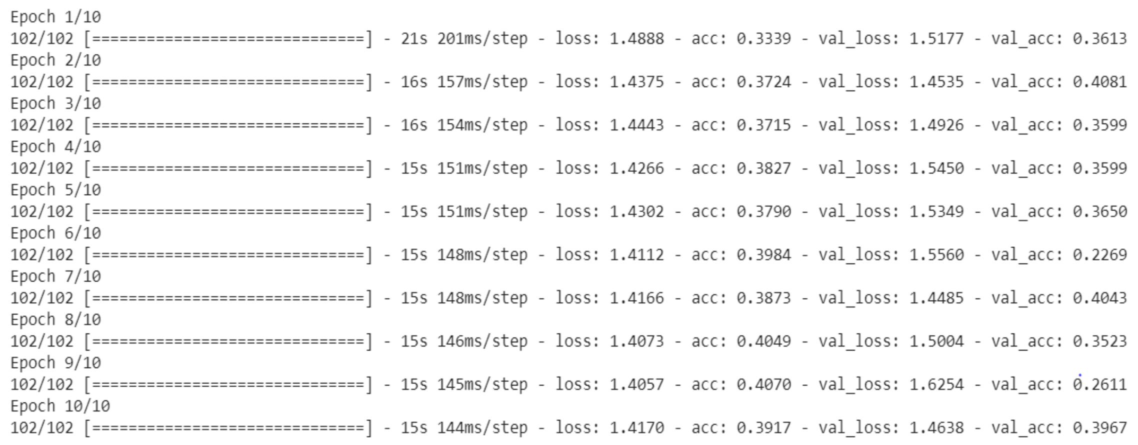


Figure 5: Data Augmentation Transformation

3.3 Baseline Model training

I would use VGG19 as my baseline model with optimizer ‘Adam’, and ‘softmax’ activation function at the output layer. Categorical cross entropy loss is used here to coordinate with the softmax activation function.

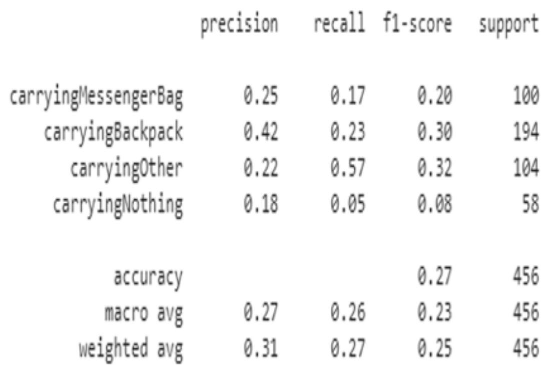
As the result shows, the training accuracy rate is just about 30



```
Epoch 1/10
102/102 [==============================] - 21s 201ms/step - loss: 1.4888 - acc: 0.3339 - val_loss: 1.5177 - val_acc: 0.3613
Epoch 2/10
102/102 [==============================] - 16s 157ms/step - loss: 1.4375 - acc: 0.3724 - val_loss: 1.4535 - val_acc: 0.4081
Epoch 3/10
102/102 [==============================] - 16s 154ms/step - loss: 1.4443 - acc: 0.3715 - val_loss: 1.4926 - val_acc: 0.3599
Epoch 4/10
102/102 [==============================] - 15s 151ms/step - loss: 1.4266 - acc: 0.3827 - val_loss: 1.5450 - val_acc: 0.3599
Epoch 5/10
102/102 [==============================] - 15s 151ms/step - loss: 1.4302 - acc: 0.3790 - val_loss: 1.5349 - val_acc: 0.3650
Epoch 6/10
102/102 [==============================] - 15s 148ms/step - loss: 1.4112 - acc: 0.3984 - val_loss: 1.5560 - val_acc: 0.2269
Epoch 7/10
102/102 [==============================] - 15s 148ms/step - loss: 1.4166 - acc: 0.3873 - val_loss: 1.4485 - val_acc: 0.4043
Epoch 8/10
102/102 [==============================] - 15s 146ms/step - loss: 1.4073 - acc: 0.4049 - val_loss: 1.5004 - val_acc: 0.3523
Epoch 9/10
102/102 [==============================] - 15s 145ms/step - loss: 1.4057 - acc: 0.4070 - val_loss: 1.6254 - val_acc: 0.2611
Epoch 10/10
102/102 [==============================] - 15s 144ms/step - loss: 1.4170 - acc: 0.3917 - val_loss: 1.4638 - val_acc: 0.3967
```

Figure 6: Baseline Model training

And we need more than one metric to measure the performance. To predict the result by our validation dataset, I want to introduce precision-recall and f1-score to our model. From the chart below, one can easily tell that the test accuracy rate is 27% which is a terrible result. And one of the reasons can be explained that the model categorizes many ‘carryingBackpack’ pedestrian to ‘carryingOther’ pedestrian. The relatively high precision rate and low f1-score for ‘carryingBackpack’ can tell that even though the model correctly categorizes some ‘carryingBackpack’ pedestrians, but it categorizes more pedestrians incorrectly as the confusion matrix shows.



```
                     precision    recall  f1-score   support

carryingMessengerBag      0.25      0.17      0.20       100
    carryingBackpack      0.42      0.23      0.30       194
       carryingOther      0.22      0.57      0.32       104
     carryingNothing      0.18      0.05      0.08        58

            accuracy                          0.27       456
           macro avg      0.27      0.26      0.23       456
        weighted avg      0.31      0.27      0.25       456
```

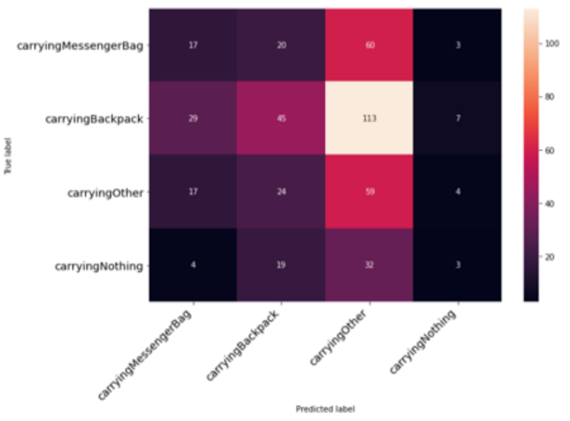


Figure 7: Baseline Model training

The model does an unsatisfied work because we used 'softmax' and 'categorical_crossentropy' for the model. They do not work well on multi-label classifying tasks because 'softmax' convert values of output layers into probability values which take all outputs into consideration.

$$softmax(y_{nj}) = \frac{exp(U^L_{nj})}{\sum_k exp(U^L_{nk}))}$$

Besides, categorical cross entropy loss only takes one category into consideration as well as it will only calculate loss of one particular categorical which is y_{nj}.

$$H_p(q) = -\sum y_i \log(p(y_i))$$

However, some pedestrians might carry more than one item. For instance, a pedestrian can carry a backpack and a plastic bag. As a matter of fact, there are 390 samples out of 4563 samples who carried more than one item. Softmax would do a terrible job in this scenario. If a person carries a backpack and an umbrella, the machine would only classify him as carrying the most probable item and ignore the other item which makes the model inaccurate.

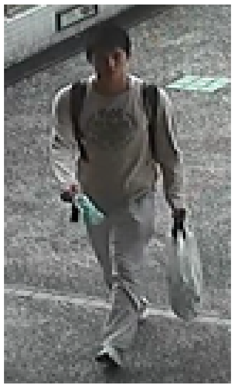


Figure 8: Pedestrian carries multiple items

3.4 Sigmoid and Binary Crossentropy Training

Therefore, to avoid this problem, we should consider the something else. To calculate possibility for each attribute independently, we should implement an activation function that does not consider other attributes such as 'tanh', 'sigmoid', and so on.

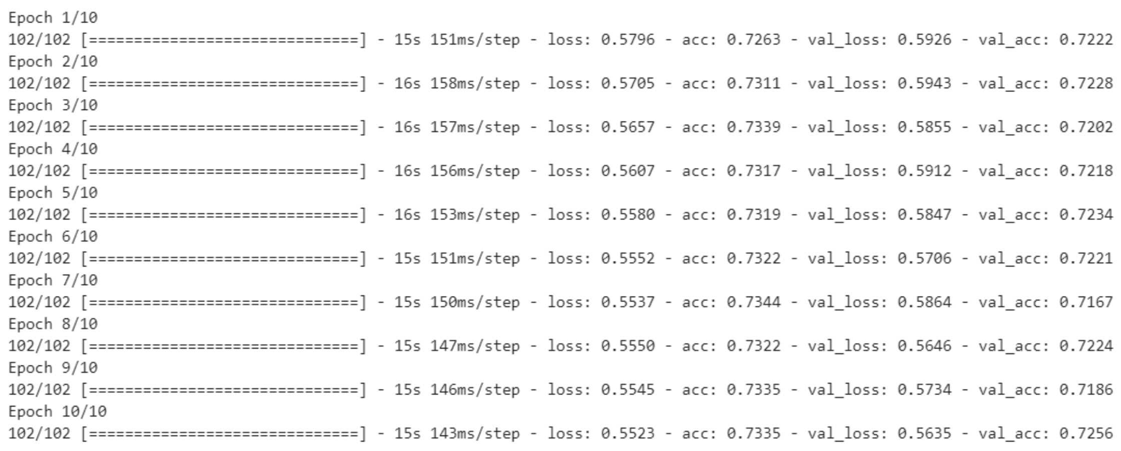
Based on my testing, sigmoid does a better job because it ranges from (0,1) but it never reaches 0 or 1. Therefore, the less possible attribute will not vanish and more possible attribute will not explode. 'Tanh' ranges from (-1,1) so it has severe vanishing problems as many attributes vanish during training and 'ReLU' avoids vanishing problems, but it would explode so only the most frequent category is predicted. As a result, 'sigmoid' activation function was chosen.

$$sigmoid(U) = \frac{1}{1 + exp(-U)}$$

Loss function needs to be changed as well, and binary cross entropy loss function always works with sigmoid function. From formula below, one can tell that this function only considers one attribute at a time, which gives huge log loss to positive class and low log loss to positive class.

$$H_p(q) = -\frac{1}{N}\sum y_i \log(p(y_i) + (1 - y_i)\log(1 - p(y_i))$$

After implementing both 'sigmoid' activation function for the output layer and 'binary_crossentropy' loss function, the training accuracy increases dramatically to at least 70



```
Epoch 1/10
102/102 [==============================] - 15s 151ms/step - loss: 0.5796 - acc: 0.7263 - val_loss: 0.5926 - val_acc: 0.7222
Epoch 2/10
102/102 [==============================] - 16s 158ms/step - loss: 0.5705 - acc: 0.7311 - val_loss: 0.5943 - val_acc: 0.7228
Epoch 3/10
102/102 [==============================] - 16s 157ms/step - loss: 0.5657 - acc: 0.7339 - val_loss: 0.5855 - val_acc: 0.7202
Epoch 4/10
102/102 [==============================] - 16s 156ms/step - loss: 0.5607 - acc: 0.7317 - val_loss: 0.5912 - val_acc: 0.7218
Epoch 5/10
102/102 [==============================] - 16s 153ms/step - loss: 0.5580 - acc: 0.7319 - val_loss: 0.5847 - val_acc: 0.7234
Epoch 6/10
102/102 [==============================] - 15s 151ms/step - loss: 0.5552 - acc: 0.7322 - val_loss: 0.5706 - val_acc: 0.7221
Epoch 7/10
102/102 [==============================] - 15s 150ms/step - loss: 0.5537 - acc: 0.7344 - val_loss: 0.5864 - val_acc: 0.7167
Epoch 8/10
102/102 [==============================] - 15s 147ms/step - loss: 0.5550 - acc: 0.7322 - val_loss: 0.5646 - val_acc: 0.7224
Epoch 9/10
102/102 [==============================] - 15s 146ms/step - loss: 0.5545 - acc: 0.7335 - val_loss: 0.5734 - val_acc: 0.7186
Epoch 10/10
102/102 [==============================] - 15s 143ms/step - loss: 0.5523 - acc: 0.7335 - val_loss: 0.5635 - val_acc: 0.7256
```

Figure 9: Sigmoid and Binary Crossentropy

However, we cannot implement precision-recall or f1-score as metrics because they can only measure whether a category is right or wrong. We need to implement more complicated metrics and we found two better metrics which are Micro_averaging and Macro_average ROC.

In Micro_averaging ROC metric, we need to sum up all individual True Positive Rate (TPR) and all False positive Rate (FPR) and apply them. It treats every element of the label indicator matrix as a binary prediction.

$$MicroavergingTPR = \frac{\sum TPs(c_i)}{\sum TPs(c_i) + FNs(c_i)}$$
$$MicroavergingFPR = \frac{\sum FPs(c_i)}{\sum FPs(c_i) + TNs(c_i)}$$

In Macro_average ROC metric, we need to take the average of TPR and FPR of the system and apply them. It gives the same weight to the classification of each label.

$$MacroavergingTPR = \frac{\sum TPR(c_i)\phi(FPRs(c_i) - FPR(c_i))}{|C|}$$

$$MacroavergingFPR = \frac{\sum_{c_i \in C} FPR(c_i)}{|C|}$$

Where C is the number of different classes.

If one wants to examine the overall model performance, Macro_averaging ROC should be considered. It takes the average TPR and FPR of unique classes so it's better to look at it when categories have relatively similar number of samples. In our cases, we should consider Micro_averaging ROC more because it sums up all categories' TPR and FPR so that a single class distribution will not effect as much as in Macro_averaging ROC.

As shown in the plot below, the Macro_average ROC with AUC equal to 0.51 is almost equivalent to the diagonal line which means that it predicts as good as a random prediction model even though it has much higher training accuracy than the previous model. However, since our dataset varies in size as figure4 shows, Macro_averaging is not the best metric to examine.

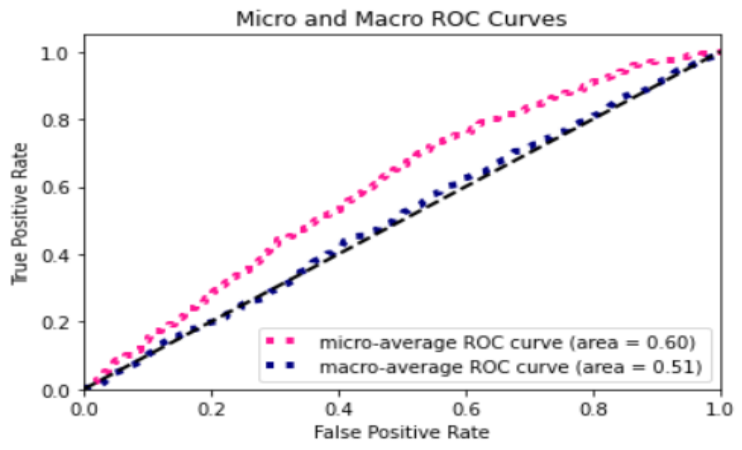
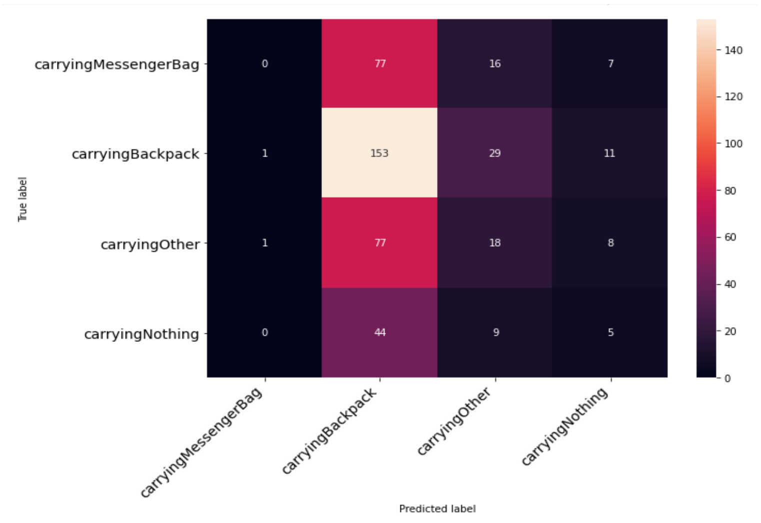


Figure 10: Micro and Macro ROC curves

While taking a look at Micro_average ROC, it has AUC of 0.6 which performs slightly better than a random prediction model. However, overall it performs better than the previous model because it focuses more on individual class. As figure11 shows, this model actually categorizes 'carryingBackpack' much better than the previous model.



True label \ Predicted label	carryingMessengerBag	carryingBackpack	carryingOther	carryingNothing
carryingMessengerBag	0	77	16	7
carryingBackpack	1	153	29	11
carryingOther	1	77	18	8
carryingNothing	0	44	9	5

Figure 11: Confusion Matrix for new Model

Below is a figure of our labeled samples and their test results. As one can see, if a pedestrian is carrying a backpack obviously, the model can recognize it. However, if there are some obstacles, it would be harder. For example, in the second figure, the woman carries a shoulder bag which is labelled as 'carryingOther' but the model thinks she carries a backpack because the man beside her carries a backpack. Especially the strap on his shoulder is a very obvious feature to notice. Therefore, even though I tried to zoom in every image to avoid this problem, sometimes obstacles still cause ambiguity. Another example is the fifth figure, where a man clearly carries nothing but the black shadow on his back might cause the model to classify it as 'carryingBackpack' which is usually black as well.

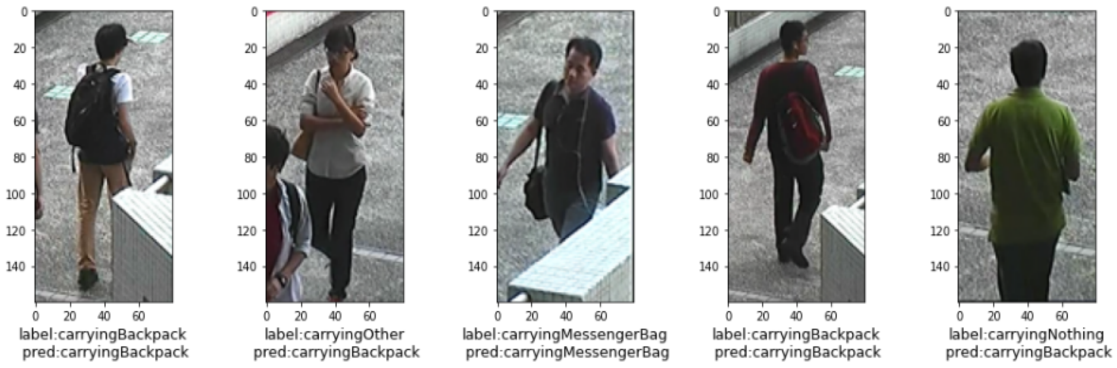


Figure 12: Pedestrian Test Result

Therefore, even though the model can predict some obvious cases, it's not practical at all. I believe that adding more attributes in the model and use their correlation can help the model. For example. If the machine knows that the man in the fifth figure is labelled 'upperbodyGreen', it might not recognize the black shadow as a backpack. If the machine know that the women is labeled 'personfalFemale', it might have larger probability to classify the small shoulder bag as 'carryingOther'.

4 Full Attribute

4.1 Full attribute Labelling

To take more attributes into consideration, we need to first find out all unique attributes. There are 101 unique attributes which can be divided into 7 major categories:

1. 'accessory'
2. 'carrying'
3. 'footwear'
4. 'hair'
5. 'lowerBody'
6. 'upperBody'
7. 'Personal'

However, not all attributes have enough samples to train. For example, there is only 1 pedestrian with a kerchief;there are only 3 pedestrians with shawls; there are only 5 pedestrians who carry luggage cases. Therefore, any attribute that has less than 100 samples is dropped because they can be considered as outliers. In the end, 54 attributes are left. The distribution of data is shown below.

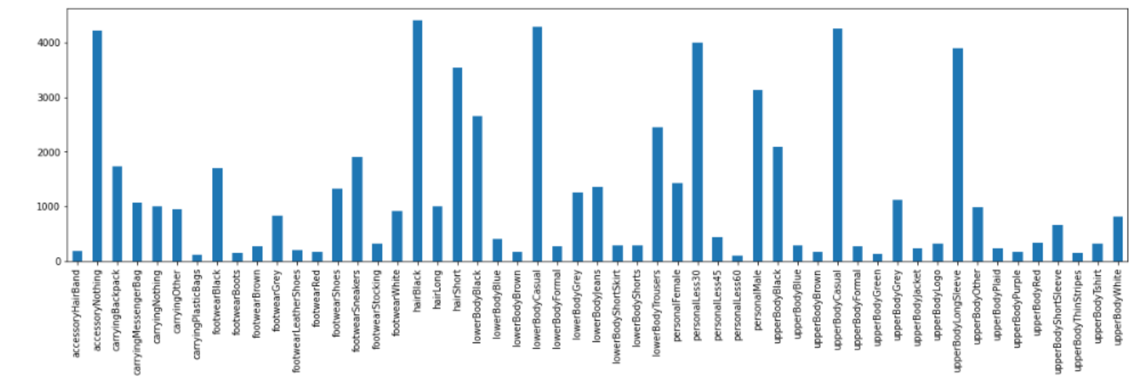
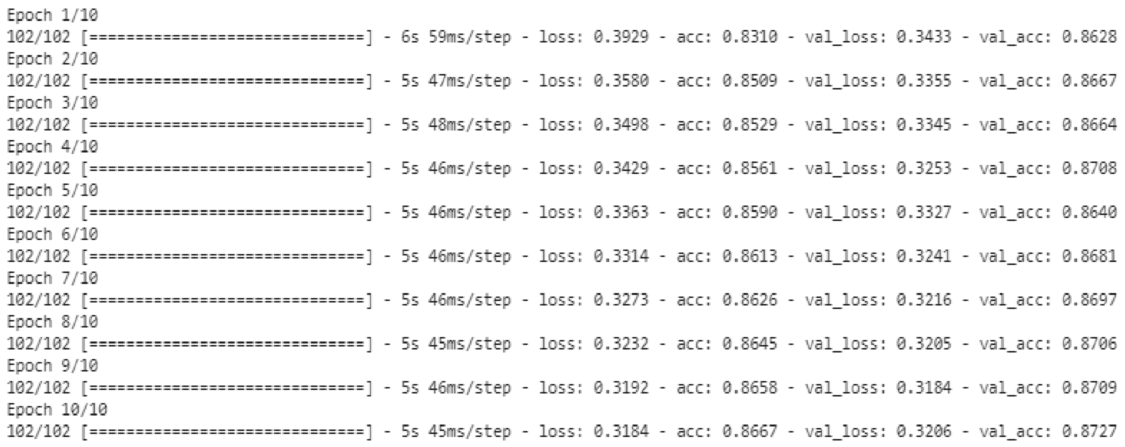


Figure 13: full attributes distribution

4.2 Model Training

The same model in the previous section was implemented, even though the data augmentation was changed because zoom_range was reduced to only 0.1 for the reason that more complete image should be taken into consideration. As a result, the result was surprisingly good. The training accuracy increased from 72% average to 87% average as shown below.



```
Epoch 1/10
102/102 [==============================] - 6s 59ms/step - loss: 0.3929 - acc: 0.8310 - val_loss: 0.3433 - val_acc: 0.8628
Epoch 2/10
102/102 [==============================] - 5s 47ms/step - loss: 0.3580 - acc: 0.8509 - val_loss: 0.3355 - val_acc: 0.8667
Epoch 3/10
102/102 [==============================] - 5s 48ms/step - loss: 0.3498 - acc: 0.8529 - val_loss: 0.3345 - val_acc: 0.8664
Epoch 4/10
102/102 [==============================] - 5s 46ms/step - loss: 0.3429 - acc: 0.8561 - val_loss: 0.3253 - val_acc: 0.8708
Epoch 5/10
102/102 [==============================] - 5s 46ms/step - loss: 0.3363 - acc: 0.8590 - val_loss: 0.3327 - val_acc: 0.8640
Epoch 6/10
102/102 [==============================] - 5s 46ms/step - loss: 0.3314 - acc: 0.8613 - val_loss: 0.3241 - val_acc: 0.8681
Epoch 7/10
102/102 [==============================] - 5s 46ms/step - loss: 0.3273 - acc: 0.8626 - val_loss: 0.3216 - val_acc: 0.8697
Epoch 8/10
102/102 [==============================] - 5s 45ms/step - loss: 0.3232 - acc: 0.8645 - val_loss: 0.3205 - val_acc: 0.8706
Epoch 9/10
102/102 [==============================] - 5s 46ms/step - loss: 0.3192 - acc: 0.8658 - val_loss: 0.3184 - val_acc: 0.8709
Epoch 10/10
102/102 [==============================] - 5s 45ms/step - loss: 0.3184 - acc: 0.8667 - val_loss: 0.3206 - val_acc: 0.8727
```

Figure 14: final training result

The test result also improved tremendously. Micro_average AUC increased from 0.58 to 0.81 which means that the model predicts result much better. As shown below, even though Macro_average AUC almost stayed as same level as the previous model, which represent that the overall model performance does not improve, prediction for each class is more accurate in this model.

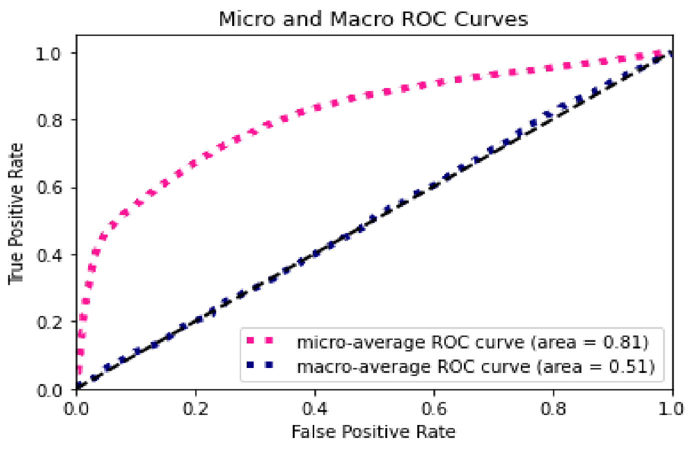


Figure 15: Micro and Macro ROC curves

5 Conclusion

In the end, I can state that more attributes can definitely help increase accuracy of multi-label model as I proposed in section 3. However, there are more aspects that we can dive deeper.

1. Class imbalance is a problem that need to be addressed. As you can see, in the last model, there are classes with 100 samples and classes with 4000 samples and methods need to be found out to solve it. Otherwise, over-fitting cannot be avoided.
2. The full PETA dataset can be learned. For the sake of concision, only CUHK dataset is used in this paper. However, the more pedestrians it learns, the more accurate the model can be.
3. More metrics can be found out to measure the accuracy of our model performance. For example, Hamming-loss, Exact Match Ratio and so on might be good metrics for the model and point out the way to improve.

6 Reference

[1] Y. Deng, P. Luo, C. C. Loy, X. Tang, "Pedestrian attribute recognition at far distance," in Proceedings of ACM Multimedia (ACM MM), 2014[PDF]

[2] V. Asch, "Macro- and micro-averaged evaluation measures," 2013 [PDF]

[3] S. Verma, "Multi-label Image Classification with Neural Network: Keras," 2019 [towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede]