CS 760: Machine Learning Fall 2020

Face Mask Detection

AUTHORS: Zijie Zhang

GO GREEN. AVOID PRINTING, OR PRINT 2-SIDED OR MULTIPAGE.

Write your abstract here

1 Introduction

With the global outbreak of COVID-19, it's even more important to have some policy to mitigate risk. For public safety and health, people are recommended to wear face masks and coverings to control the spread of the COVID-19.

In hospitals and various COVID-19 testing places. If only people wearing masks are allowed to enter, the risk of infection for doctors and staff can be reduced. However, if a potential COVID-19 infected person who is not wearing a mask suddenly appears, there is a high risk of infection in face-to-face communication.

At many intersections, pedestrians will gather briefly while waiting for traffic lights. At this time, the risk of COVID-19 spreading among the population is high. But it is impossible to hire a person to stay at the intersection and remind pedestrians to wear masks.

Similarly, requiring Uber drivers and passengers to wear masks at all times can also effectively reduce the spread of the COVID-19. However, it is impossible to supervise the wearing of masks on moving vehicles in real time.

There are many application scenarios such as this. If it is all supervised by manpower, the investment cost is high, and the health risks of the staff are also high.

Therefore, the need for artificial intelligence to determine the wearing of masks came into being. The main idea of this project is to construct a classifier to judge whether the photo cropped from the face detector is wearing a mask.

2 Related/Similar work

3 Dataset

3.1 Source

The data comes from the following sources:

1. **Real-World Masked Face Dataset, RMFD**
   `https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset`
2. **CASIA-WebFace-Dataset**

3.2 Description

1. The dataset contains 3400 images, which belong to two categories:
   (a) **with_mask: 1470**
   (b) **without_mask: 1930**
2. Classes:{'with_mask', 'without_mask'}
3. The directory structure:

```
└─dataset
   ├─with_mask ...  1470 images.
   └─without_mask ...  1930 images.
```

4 Approach

4.1 Pre-processing

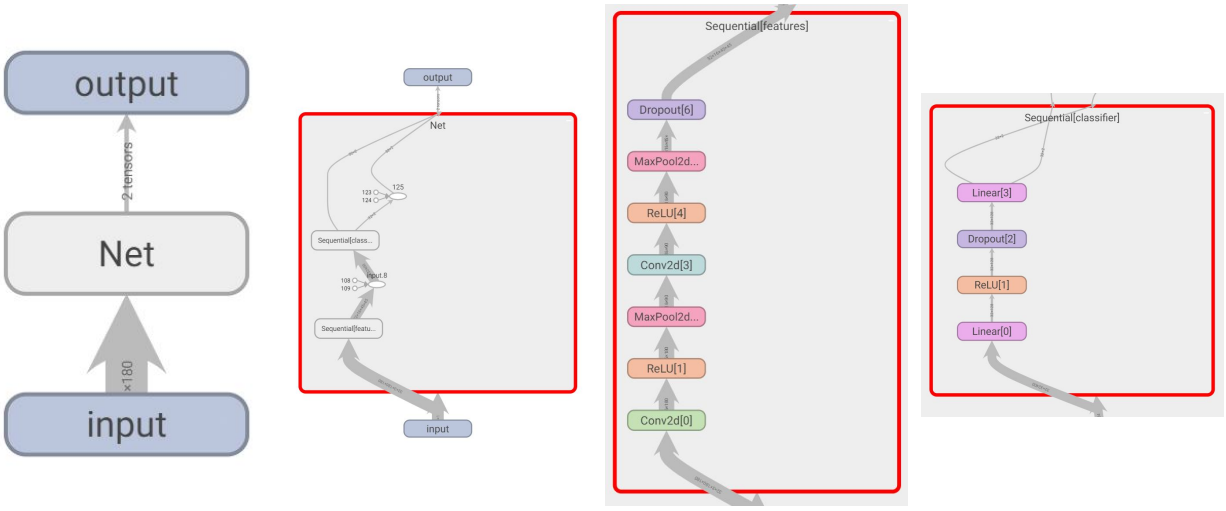
1. Since the shapes of the images in the data set are not the same, the first step of preprocessing is to resize all images to the same shape.(180x180)
2. Data augmentation:
   Due to the small sample size of the data set, over-fitting is prone to occur. Therefore, the operation of data augmentation was added. Generate some effective new data on the basis of existing data by using random transformation. This helps expose the model to more aspects of the data.
   (a) ColorJitter
   (b) RandomRotation
   (c) RandomVerticalFlip
   (d) RandomHorizontalFlip
3. For each RGB image, it has three channels. We can convert it to a [3, 180, 180] **Tensor**.
4. For each element in the tensor, its range is [0,255]. We should normalize them.
5. Data preview:

```
Transforms = transforms.Compose([
    transforms.Resize((IMAGE_HEIGHT, IMAGE_WIDTH)),
    transforms.RandomApply(torch.nn.ModuleList([
        transforms.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.1),
        transforms.RandomRotation(degrees=15),
    ]), p=0.3),
    transforms.RandomVerticalFlip(p=0.1),
    transforms.RandomHorizontalFlip(p=0.1),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
```

4.2 Neural Networks Method

Generally, Convolutional Neural Networks (CNN) are complex feed forward neural networks. CNNs are used for image classification and recognition because of its high accuracy.

So the first method is Neural Networks.



1. The main structure of the neural network consists of two parts: Feature learning and classifier.
2. The first part includes two **Conv2d** layer.
3. The second part is a Full-Connected Network.

```
Net(
  (features): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=(1, 1), ceil_mode=
        False)
    (3): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation=(1, 1), ceil_mode=
        False)
    (6): Dropout(p=0.2, inplace=False)
  )
```

```
  (classifier): Sequential(
    (0): Linear(in_features=32400, out_features=128, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=128, out_features=2, bias=True)
  )
)
```

4.3 Nearest Neighbors Algorithm

Considering the use of K-NN algorithm to achieve face recognition, we can try to use K-NN for mask wearing recognition.

Based on the previous preprocessing, flatten each Tensor into a 1-dimension vector.

Choose algorithm = 'kd_tree' to speed up. Implement with parameters:

```
n_neighbors = 15, weight = 'distance', metric='euclidean'
```

For a new image after preprocessing, we will gave a point in the 32400-dimension vector space. Calculate the Euclidean distance from this point to every known point. The predicted value of this unknown point is obtained by weighted voting of the 15 known points closest to it.

4.4 Random Forest Algorithm

As we all know, decision trees can be very sensitive to the specific data we observe. In fact, they can be quite biased, and tend to overfit.

After randomly slicing the original data set, generate a decision tree on each data set after the split. Use information entropy as the criterion for selecting nodes.

After generating a hundred decision trees like this, for a new image after preprocessing, every decision tree will give a prediction. Choose the class with the most votes as the predicted value of the new image.

4.5 Packages

1. pytorch
2. sklearn

5 Results

5.1 Description of experiments

For the above three methods, after randomly slicing the original data set at a ratio of 0.2, train on the training set to obtain accuracy on the test set.

1. **Neural Networks**:
   (a) Accuracy of the network on the test images: 92 %.

(b) Accuracy of with_mask : 95 %.

(c) Accuracy of without_mask : 91 %.

2. **Nearest Neighbors**:

```
[ Nearest Neighbors ]
classification report on the test set
done in 22.212s
              precision    recall  f1-score   support

           0       0.92      0.73      0.81       294
           1       0.82      0.95      0.88       386

    accuracy                           0.86       680
   macro avg       0.87      0.84      0.85       680
weighted avg       0.87      0.86      0.85       680
```

3. **Random Forest**:

```
[ Random Forest ]
classification report on the test set
done in 0.061s
              precision    recall  f1-score   support

           0       0.92      0.84      0.88       294
           1       0.88      0.95      0.91       386

    accuracy                           0.90       680
   macro avg       0.90      0.89      0.90       680
weighted avg       0.90      0.90      0.90       680
```

5.2 Comparisons

5.3 Results

6 Conclusions and Future Work

7 References