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

This project is mainly inspired by **Baidu AI development platform - mask wearing detection products**.

`https://ai.baidu.com/tech/body/driver`

Related/Similar work:

1. https://github.com/AIZOOTech/FaceMaskDetection
2. https://github.com/chandrikadeb7/Face-Mask-Detection
3. https://arxiv.org/abs/2003.09093

3 Dataset

This dataset consists of 3835 images belonging to two classes:

- with_mask: 1916 images
- without_mask: 1919 images

The images used were real images of faces wearing masks. The images were collected from the following sources:

1. https://github.com/chandrikadeb7/Face-Mask-Detection
2. RMFD dataset(https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset)
3. MAFA dataset(https://www.kaggle.com/rahulmangalampalli/mafa-data)

The following is the directory structure:

```
└─dataset
  ├─with_mask ...  1916 images.
  └─without_mask ...  1919 images.
```

4 Approach

4.1 Preprocessing

- `data_dir = "../dataset"`
  `PATH = list(paths.list_images(data_dir))`
- Load images:
  The RGB channel values are in the [0, 255] range. We don't need all the information of the three color channels, just convert the image to grayscale.
- Rescale all images to the same size.
  `image = cv2.imread(img_path, 0)`
  `image = cv2.resize(image, (img_width, img_height))`
- Standardize data:
  `image = image.flatten()/255.0`

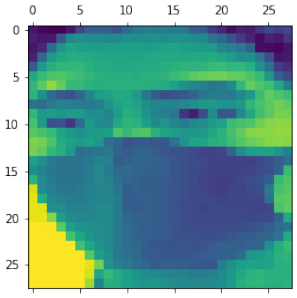
So far, every image has been transformed into a vector with a length of 50176. This will make it more convenient to use PCA to extract the principal components.

4.2 Visualize the data

4.3 Data augmentation

Due to the small number of training samples in this data set, overfitting generally occurs. Data augmentation takes the approach of generating additional training data from existing examples by augmenting them using random transformations that yield believable-looking images. This helps expose the model to more aspects of the data and generalize better.

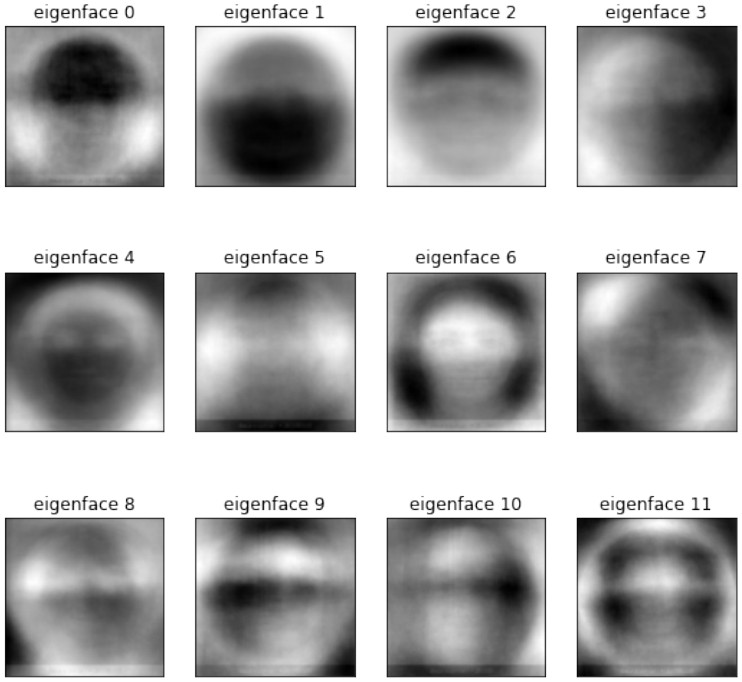
4.4 Dimensionality reduction



It can be seen that the picture contains many similar pixels, so referring to the idea of face recognition, we can perform appropriate dimensionality reduction processing through principal component analysis. **sklearn.decomposition.PCA** is used here.

```
n_components = 15
pca = PCA(
  n_components=n_components,
  svd_solver='randomized',
  whiten=True
).fit(X_train)
eigenfaces = pca.components_.reshape(
  (n_components, img_height, img_width)
)
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

Obviously, after extracting the first 15 main features, we can still clearly distinguish whether people are wearing masks.



4.5 Classification algorithms

4.5.1 Support Vector Machines

I chose C-Support Vector Classification(SVC) with Radial basis function kernel(RBF). Use GridSearchCV to find the optimal SVC parameters C and gamma. **sklearn.model_selection.GridSearchCV** and **sklearn.svm.SVC** is used here.

```
param_grid = {
    'C': [1e3, 5e3, 1e4, 5e4, 1e5],
    'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1]
}
SVM = GridSearchCV(
    SVC(kernel='rbf',
        class_weight='balanced',
        probability = True,
        verbose = True
        ),
    param_grid,
    verbose=10,
    n_jobs=-1
)
SVM = SVM.fit(X_train_pca, y_train)
print(SVM.best_estimator_)
```

4.5.2 Nearest Neighbors

For the n_components, consider the nearest neighbor classifier.

4.5.3 Decision Trees

4.5.4 Bayesian Learning

4.5.5 Logistic Regression

5 Results

6 Conclusions and Future Work

7 References