IN3060/INM460 Computer Vision Coursework report

- **Student name, ID and cohort:** Sykat Howlader (210018421) - UG
- **Google Drive folder:** https://drive.google.com/drive/folders/1V9vmBAnllW1wh9UHgAyoAEABrhLMG1s6?usp=sharing

This report outlines the development and evaluation of three computer vision models that tries to correctly predict face covering using machine learning.

Data

The dataset used in this project comprises of cropped coloured images of human faces. It consists of respectively 2394 training and 458 testing images. Labels are provided for each image (in accompanying txt files) as an integer from 0 to 2: 0 if no mask is worn, 1 if it is worn, and 2 if it is worn improperly. The distribution of the dataset is as follows, which indicates class imbalances:
Correctly Worn Mask (1): 1,940 instances
No Mask (0): 376 instances
Improperly Worn Mask (2): 78 instances
For a more robust testing strategy a personal dataset was created, consisting of images sourced from the internet. For each category 7 images were allocated.

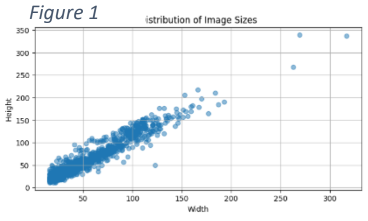
Implemented methods

For this project three different computer vision models were implemented:

- Support Vector Machine with Hog (histogram of oriented gradients) feature descriptors
- Multi-layer Perceptron with Hog feature descriptors
- Convolutional Neural Network

HOG + SVM

Figure 1



The first classification model implemented was a Support Vector Machine (SVM) trained on HOG features extracted from the images. SVM was chosen because of its robustness and effectiveness in high dimensional spaces. HOG feature descriptors, on the other hand, capture well edges and gradient structures, which are fundamental in delineating facial structures and detecting the presence or absence of masks.
The images in the dataset have different sizes as shown in Figure 1 , so I decided to resize them all to a specific size 150x128 pixels. After resizing them, some image preprocessing have been applied to the images. The images were turned into greyscale, because information about the colour is not that relevant for the mask detection, and computation is faster on greyscale images than RGB images. A histogram equalization was applied to improve the contrast in the images and subsequently a sharpening filter was applied to enhance the edges between the face and the mask if present. Once the preprocessing has been completed, the HOG feature extraction began, following a same procedure as outlined in Tutorial 5. The ‘train’ dataset has been split into training and validation set. A SVM classifier was initialized with kernel model rbf (radial basis function) and gamma parameter 0.001. The SVM was then trained using the HOG features of the training set and the model was saved. The classification report on the validation set showed that the model did not predict any instances of classes 0 (No mask) and class 2 (Mask worn improperly). This might have happened because of the class imbalance. So, to try to overcome this challenge a new SVM was initialized with these hyperparameters: classifier = svm.SVC(kernel='linear', C=1.0, class_weight='balanced', random_state=42) where class_weight='balanced' parameter will automatically adjust weights inversely proportional to class frequencies in the input data. This approach yielded better results.

HOG + MLP

The second classifier implemented was a Multi-layer perceptron trained also on Hog features. The choice in this case was to compare the two different models using the same HOG features descriptors. For this classification model, the same preprocessing of the images were applied. So, images were resized, converted into greyscale, went through a histogram equalization and then a sharpening filer. After HOG features were extracted, train set was split into training and validation, and a MLP was initialized with these hyperparameters (hidden_layer_sizes=(100,), max_iter=300, activation='relu', solver='adam'). After training the MLP on the HOG features, the training score of 1.00 indicated possible overfitting during the training process, so the hyperparameters of the MLP were slightly changed, to include L2 normalization: (hidden_layer_sizes=(100,), max_iter=100, alpha=1e-4, solver='sgd', learning_rate_init=.1, early_stopping=True, validation_fraction=0.1, n_iter_no_change=10) .

CNN

The third model developed was a convolutional neural network (CNN), selected for its ability to perform well in image classification tasks. In this case there was no need to do feature extraction as it is done automatically by the convolutional layers. The challenge in this case was to define an optimal architecture that balances depth (number of layers) with complexity (number of parameters). The CNN architecture consisted of two convolutional layers with 5x5 kernels followed by max pooling, and three fully connected layers, ending in an output layer designed for three classes. The network was trained using a custom dataset where images were pre-processed using a series of transformations that include conversion to PIL format, resizing to 128x128 pixels, normalization, and standardization to ensure uniform input. The results on the validation set, suggested that the model was performing poor on people wearing mask improperly. To try to improve the model, dynamic data augmentation technique was applied in the transformation process, to include random flips, rotations and colour adjustment to images where mask were worn improperly. However this did not lead to any improvements.

IMPLEMENTATIO ON THE IMAGES IN THE WILD

For the implementation of MaskDetection function in the wild, at first I tried to apply Haarcascade Frontalface model to detect the faces in my personal dataset. However, the face detector did not perform well as in the images people have different poses and angles. Consequently, I abandoned this approach and opted for a more controlled preprocessing step. Images were cropped to just show the faces, so that the models could be applied only to these specific areas.

Results

This section will outline the results of the three different models implemented. The performance of these models has been significantly influenced by the class imbalance present in the dataset. In fact, a model efficiency depends on the quality and structure of the data its trained with. Models trained on well-balanced and representative datasets tend to be more robust and perform better.
For the SVM and MLP models, specific image preprocessing steps were implemented, as detailed in the implementation section. As we can see from the figures 2,3 and 4, these preprocessing steps enhanced the image quality which in turn facilitated more effective feature extraction.

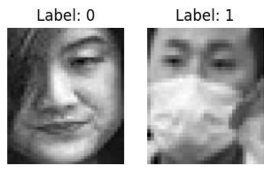


Figure 2 - greyscale

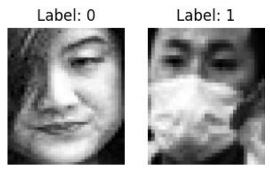


Figure 3 - histogram equalization

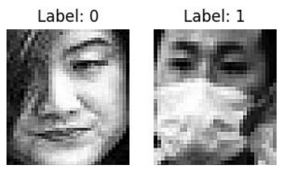


Figure 4 - sharpening filter

The SVM, after adding the hyperparameter class_weight='balanced', which would automatically adjust weights inversely proportional to class frequencies in the input data, has improved its ability to detect people wearing mask improperly. Even though it is a slight improvement, this model predicted correctly 5 instances out of 19 instances of test images with label ‘2’, as shown in figure 5.
The MLP and CNN on the other hand, even after L2 regularization applied to the first one and data augmentation to the latter one, did not show any progress in predicting test images of label ‘2’.
The SVM and MLP are relatively similar in terms of overall accuracy. They both achieved an accuracy of 85% in the test set of the images. However if we consider the confusion matrix of the two models we can see that the SVM performs slightly better on the category ‘2’ (mask worn improperly) compared to the MLP. In

fact, as mentioned before the SVM identified correctly 5 instances of 'people wearing mask incorrectly' compared to 0 in MLP, as shown in figures 5 and 6.
The CNN emerged as the top-performing model, achieving an impressive 92% accuracy on the test set. However, this high accuracy should be interpreted with caution. Despite its overall accuracy, the CNN failed to identify any instances of 'people wearing masks incorrectly.' Its performance was relatively high for labels 0 (no mask) with accuracy of 92%, and 1 (mask worn correctly) with accuracy of 97%, but its inability to detect label 2 highlights a critical area for further improvement and model tuning.
In terms of size, the models can be classified as follows:

- SVM, with a size of 61,3 MB
- MLP, with a size of 12,5 MB
- CNN, with a size of 6,2 MB

Interestingly, the CNN, despite being the smallest model in size, it delivers the highest performance. This underscores the efficiency of CNN architectures in utilizing computational resources while achieving superior accuracy.

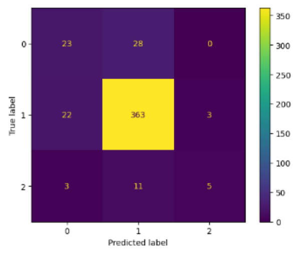


Figure 5- SVM CM

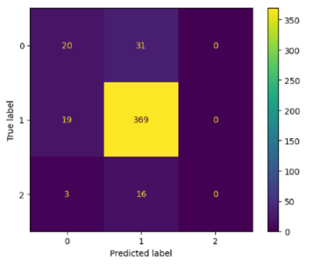


Figure 6- MLP CM

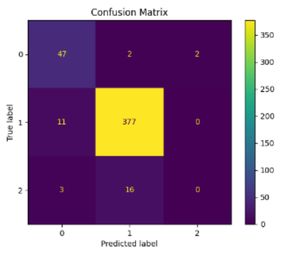
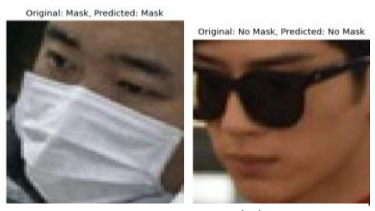


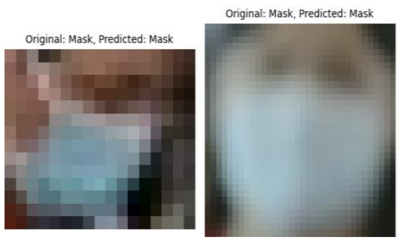
Figure 7- CNN CM

Below, some test images from the test set are displayed along with their ground truth labels and predicted labels from each model. As we can see some images are also hard to interpret even with human eyes due to their quality.



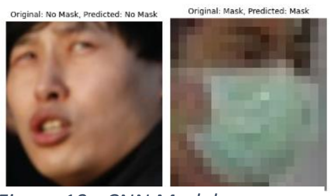
Original: Mask, Predicted: Mask
Original: No Mask, Predicted: No Mask

Figure 8- HOG+SVM model



Original: Mask, Predicted: Mask
Original: Mask, Predicted: Mask

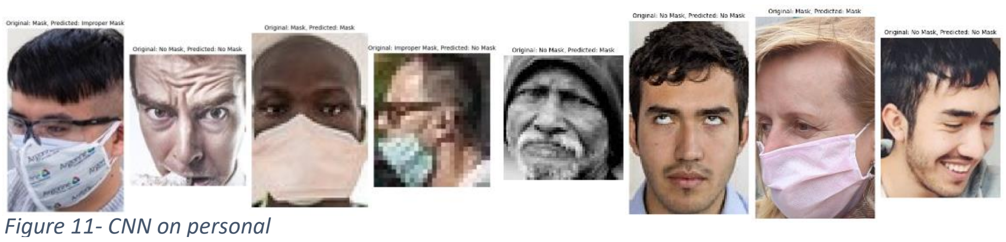
Figure 9- HOG+MLP model



Original: No Mask, Predicted: No Mask
Original: Mask, Predicted: Mask

Figure 10 - CNN Model

I will now present some results from tests conducted by the CNN on my personal dataset, which consists of images collected from the internet. As we can see the 'improper mask' is category where the model makes the most mistakes.



Original: Mask, Predicted: Improper Mask
Original: No Mask, Predicted: No Mask
Original: Mask, Predicted: Mask
Original: Improper Mask, Predicted: No Mask
Original: No Mask, Predicted: Mask
Original: No Mask, Predicted: No Mask
Original: Mask, Predicted: Mask
Original: No Mask, Predicted: No Mask

Figure 11- CNN on personal

References

For this project the following codes were taken and adapted from the tutorials and external sites:

- HOG feature extraction, from Tutorial 5
- Loading images and labels from selected directories, and defining MLP classifier: Tutorial 6
- Defining a CNN network architecture, loss function, optimizer, and transformations: Tutorial 7
- Data augmentation for CNN: https://pytorch.org/vision/stable/transforms.html
- Custom Dataset for CNN: https://medium.com/analytics-vidhya/implementing-cnn-in-pytorch-with-custom-dataset-and-transfer-learning-1864daac14cc