Dataset Creation

Workspace layout

```
├── artificial_dataset_creation
│   ├── advanced_artificial_facemask_overlay
│   ├── artificial_facemask_overlay
│   ├── duplicate_image_detector
```

For specific commands and expected inputs for each of the scripts please consult the provided README‘s within their respective directories.

The purpose of these scripts/programs is to facilitate the creation of a dataset for training/producing a model with learning_algo.py (contained within the main mask_detection project folder).

advanced_artificial_facemask_overlay and artificial_facemask_overlay contain two different methods of applying a mask to a face, the first having a range of options for manipulation of the location of the mask (this is useful for creating a dataset of incorrectly worn masks i.e. not covering the nose or mouth), the second script has fixed positions for the position of the masks on the detected faces but has the advantage of not requiring labelling of the facial landmarks on each mask added to the mask source folder. In short, the advanced script will result in a more realistic output but requires some extra setup for each mask whereas the second more basic script allows for a quicker turn around as it does not require this extra step.

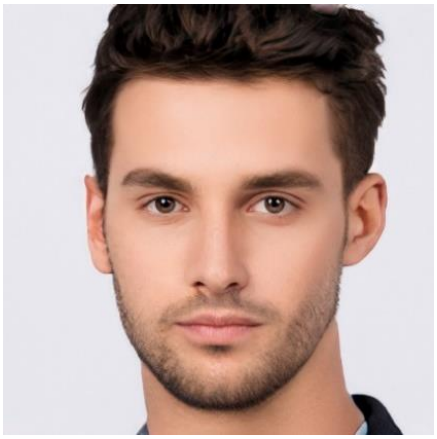
Recommendations for optimal results

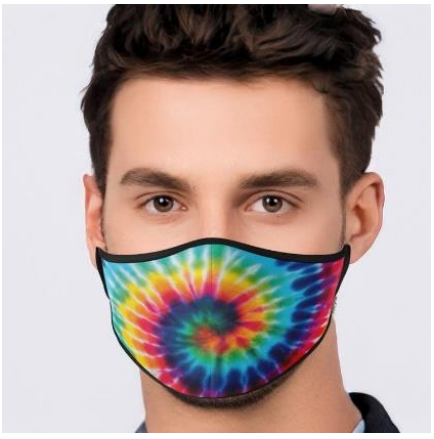
The dataset labels are as follows:

- with_mask
- no_mask

For each label the images should be as close to the use case as possible i.e. pick natural looking portrait style images, a mixture of physical attributes: gender, race, jewellery, glasses, etc.
For the with_mask label we recommend a mix of real masked individuals and artificially generated images.
For the masks themselves we recommend images of high-resolution masks which are as close to the source image’s perspective as possible.

An example image after processing with the artificial_facemask_overlay script:





Model Training (learning_algo.py)

Generating a model

Workspace layout

```
├── dataset
│   ├── with mask
│   ├── without_mask
├── learning_algo.py
├── model1.model
```

Learning_algo.py takes in 4 arguments:

- **-d** or **--dataset**, this is used to provide the path to the directory containing the dataset. For the above example, the argument would be **-d .\dataset**.
- **-p** or **--plot** can be used to generate the plot with regards to the accuracy of the model on the train and the test set. For example, the argument would be **-p** or **-p "Graph.png"**.
- **-m** or **--model** argument to provide the make for the model to be saved as, the file extension is **.model**. for example, the argument would be **-m** or **--model "first_model.model"**.
- **-tm** or **--trainmodel** is used to provide a path to a pre-trained model to be re-trained. For example, for the above workspace layout let us say we want to train model1.model then the argument would be **-tp** or **--trainmodel .\model1.model**.

When create a model or retraining one make sure to specify the path to the dataset from the directory containing the learning_algo.py. If the user does not provide a name for the trained model to be saved as then the model would be saved as **mask_detector.model**. Even when retraining a pre-trained model, the **-m** or **--model** can be used to provide a name for the new model to be saved. For the above workspace layout, the command to train a new model on the images in the dataset folder, produce an accuracy plot with the name **acc_plot** and save it as **new_model.model** is as follows:

```
python learning_algo.py -d .\dataset -m "new_model.model" -p "acc_plot"
```

Expanding the pretrained model

Workspace Layout

```
├── dataset
│   |---with mask
│   |---without_mask
├── dataset2
│   ├── newDS
│   │   ├── with mask
│   │   ├── without_mask
├── learning_algo.py
├── model1.model
```

In-order to retrain a pre-trained model the **-tm** or **--trainmodel** argument should be used as mentioned above. When training a pre-trained mode, the user can use the **-m** or **--model** argument to provide a new name for the model to be saved as else it will be saved as the default (**mask_detector.model**). for the above workspace example, the command to re-train **model1.model** on **newDS** and save it as **DSmodel.model** is as follows:

```
python learning_algo.py -d .\dataset2\newDS -tp .\model1.model -m "DSmodel.model"
```