Project Report: Gender Classification of Masked Facial Images

CZ4042 Neural Networks and Deep Learning

Written by:
Jason Lee (U1727712L)
Lim Xuan Yu (U1721729J)
Ng Kai Chin (U1721647D)

Date: 23rd November 2020

Table of contents

1 Project Overview

1.1 Objectives

In this project, we aim to design a deep convolutional neural network (CNN) that is capable of performing gender classification on a facial image, wherein the face is occluded by a mask. Gender in this report refers to the biological, assigned sex of the subject rather than their personal gender roles. This corresponds to prompt D, Gender Classification of the CZ4062 Group Project guidelines. The specifications of this model and how we arrived at it are discussed later in this report.

Additionally, we explore the following tasks:

1. Exploration and modification of previously published architectures
2. Improving gender classification for faces obstructed by masks

1.2 Motivation

Gender (or more specifically, sex) recognition is an important task for animals in general, be it for competition or mating. In humans, this is mainly done visually. This is a deceptively complex task as both genders have the same general facial features - two eyes, a nose and a mouth. While some features are exclusive to either gender (e.g. beards, makeup styles), humans are still able to ascertain genders in their absence, and may sometimes even ignore them entirely if other cues indicate otherwise.

Automatic gender detection is an important task in Human Computer Interaction (HCI). Gender detection has many applications, including recommender systems, targeted advertising, as well as surveillance and security [1]. One such use case is in surveillance using images or video footages. By detecting the gender from an image or video frame, the set of subjects that a surveillance team will require to compare can be greatly reduced, leading to higher levels of efficiencies [2].

In a post-COVID-19 world, many countries continue to stay vigilant and consequently require their populus to wear face masks when outdoors. These masks usually obstruct more than half the face, covering the nose and mouth. In addition, they are not standardised at all, with colours and patterns varying wildly. This would immediately cause the performance of facial recognition systems to drop, especially if they are situated outdoors. We hope that by taking the possibility of masked individuals into account, we will be able to train a CNN that will be able to accurately classify gender in spite of the COVID precautions taken worldwide.

2 Literature Review

2.1 Review of existing techniques

Earlier works concerning facial classification focused on near-frontal face images. These types of images do not have complex variations. These works have produced very high accuracies on the FERET benchmark [3], which uses images that were taken under highly controlled conditions.

These earlier works mainly used human-designed features to represent facial images. These methods include the use of local binary patterns (LBP) [4] and Gabor features [5]. These proved effective on constrained benchmarks such as FERET.

More recently, the Labelled Faces in the Wild (*LFW*) dataset [6], as well as the more challenging *Adience* dataset have been used as benchmarks as they present facial images taken in more challenging, real-world conditions. These datasets are categorised as unconstrained, as opposed to constrained ones like FERET.

Eidinger et al (2014) [8] trained a model using the best representation methods of that time (LBP [4] and Four Patch LBP [7]) with a linear SVM classification, but found that it only achieved a modest accuracy of 76.1% on the more challenging, unconstrained Adience benchmark [8]. This proved the need for more complex models such as deep CNN architectures.

One of the foundational deep CNN works, Levi et al (2015) [9], proposed a 5-layer network (3 convolutional, 2 fully-connected) which achieved a reasonable accuracy of 86.8% on the Adience benchmark given its small network design.

Since then, there have been several state-of-the-art deep CNN architectures developed, including Inception (also known as GoogLeNet) [10] and ResNet [11]. Other works include Zhang et al (2017) [12], where they proposed a new CNN method which uses residual networks of residual networks (RoR). Their RoR model was pretrained on 2 datasets, namely ImageNet and IMDB-WIKI-101, and subsequently fine-tuned on Adience. This achieved a state-of-the-art accuracy of 93.2% on the Adience dataset.

2.2 Inception Model

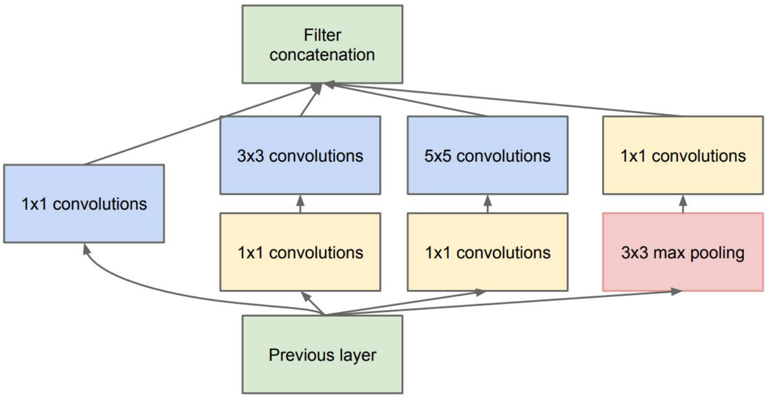


Figure 2.1: Inception module architecture [10].

The Inception model consists of many Inception modules stacked on top of each other, with regular max-pooling layers in between to manage the size of the grid. The key motivation for the Inception module architecture was approximating an optimal local sparse structure by using readily available dense components, which are computationally efficient [10].

2.2 ResNet Model

The ResNet model uses deep residual learning to combat degradation in accuracy in deep networks (degradation problem).

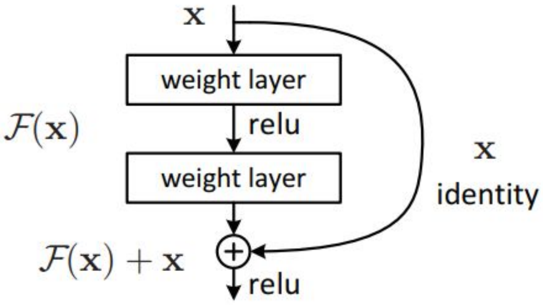

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.2: Building block of residual learning [11].

The residual mappings shown in Fig 2.2 were realised via shortcut connections, or connections skipping multiple layers, in the proposed model. This architecture allowed ResNet to effectively solve the degradation problem, and enjoy improved results with very deep models. A variety of ResNet models have been implemented, including models with 50, 101, and 152 layers.

Both Inception and ResNet models have been refined and reviewed since their original models. McNeely-White et al (2019) [13] found that the features extracted by Inception and ResNet models are qualitatively the same, which means that the 2 models extract approximately the same information from images. This could explain the similarity of their performances on common benchmarks.

2.3 Choice of Models

In this study, we looked at the InceptionV3 and ResNet50V2 models as implemented on Keras Applications. The Inception and ResNet models are considered state-of-the-art models presently, and are widely used in research. Although they exhibit similar performances on previous benchmarks (LFW, Adience), they have different architectures and core ideas, as discussed above. As such, it is worthwhile comparing the performances of these two models on a new dataset. These versions were chosen due to their similar parameter count, with 23,851,784 for InceptionV3 and 25,613,800 for ResNet50V2. Given our limited hardware capabilities, we are limited to training on these models which are not too large in size.

3 Methods

3.1 Overview

We employed State-of-the-art (SOTA) neural network architectures to accomplish our task of classifying a person's gender, given an image of a person wearing a mask. This is done through transfer learning. Based on our literature review, we have selected two SOTA models to carry out the project on, namely the **InceptionV3** and **ResNet50V2**.

In terms of project execution, the entire flow can be seen in Figure 3 below. From the initial Adience data set, we carried out preprocessing by adding masks to the images and augmenting the image data. Following that, we carried out transfer learning on our selected SOTA models and evaluated the preliminary results. Finally, we carried out a fine-tuning process where we unfroze more layers at the end of the neural network, and compared the new results with our preliminary results. The model that gave us the best prediction accuracy on our test set was selected as the final optimized model.

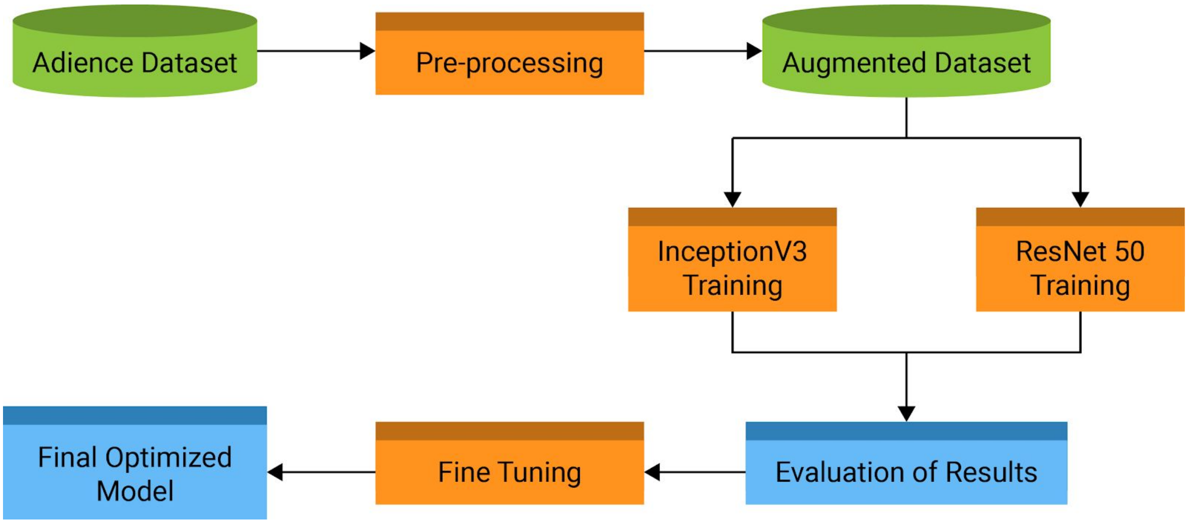


Fig 3: Project Pipeline

3.2 Pre-processing

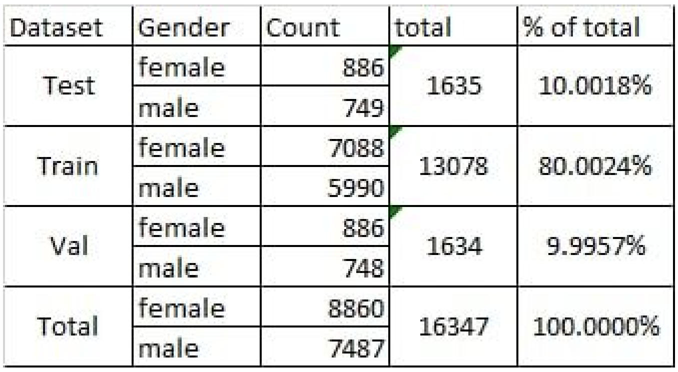
In order to train a Deep Learning model to detect gender of people wearing masks, we had to first acquire image data of people wearing masks, which we were not able to find online.

In order to remedy this problem, we decided to augment the data by placing masks onto existing images from the Adience Dataset. This in itself was not a trivial task, as facial recognition was needed to identify the right scale and location to overlay the mask images.

3.2.1 Dataset Source

A portion of the Adience Dataset [8], consisting of 26,580 photos with 2,284 subjects, was used to train, validate and test the model. The images feature a wide range of lighting conditions, pose and appearance to simulate real world conditions. The images are sourced from Flickr albums whose owners release the images under the Creative Commons (CC) License.

From a selected subset of 16,437 images, we created a train-test-validation split of 80-10-10 ratio. The datasets were adjusted to contain the same ratio of male to female faces in order to make the data more homogeneous. The breakdown of the data is as follows:



Dataset	Gender	Count	total	% of total
Test	female	886	1635	10.0018%
	male	749		
Train	female	7088	13078	80.0024%
	male	5990		
Val	female	886	1634	9.9957%
	male	748		
Total	female	8860	16347	100.0000%
	male	7487		

Figure 3.2 Breakdown of training data split

3.2.2 Cropping and Alignment

The images from the Adience Dataset come with some pre-processing applied. The faces are cropped to the region of interest and aligned such that the primary face in the image is aligned vertically with respect to the image frame. This step reduces some of the noise and helps the model train on the appropriate features. The aligned and cropped images were already available from the Adience dataset in the file aligned.tar.gz (1.9G) and thus we did not need to do it, though we consider it to be part of the training process.

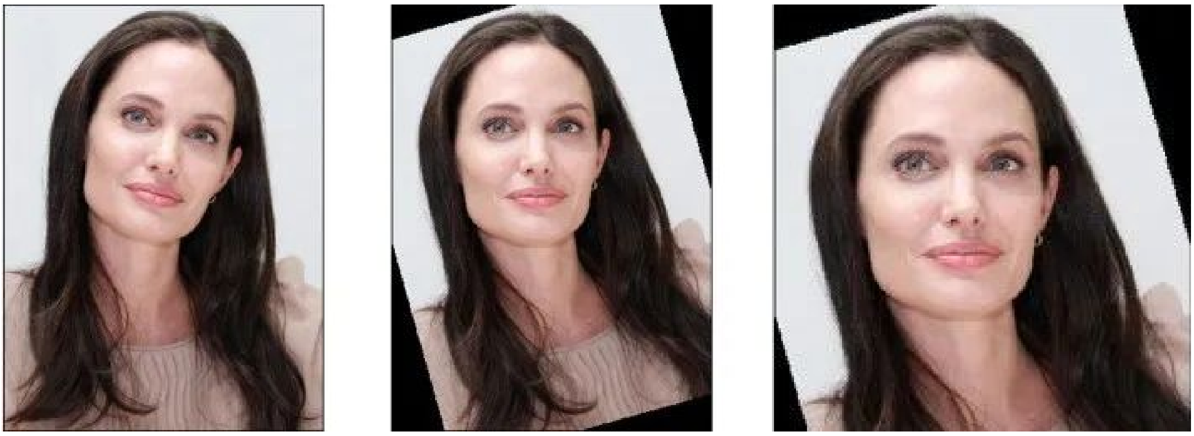


Figure 3.1.1 Alignment and Cropping Steps on Image. [14]

3.2.3 Facial Landmark Detection

The Face Recognition package [15], a machine vision library, was used to detect faces and identify the facial landmarks we could use to modify the base images. This library was built off dlib's state-of-the-art face detection deep learning model [16]. This detects 68 facial landmarks and notes their position on the image file. The landmarks are shown in Fig 3.1.2.

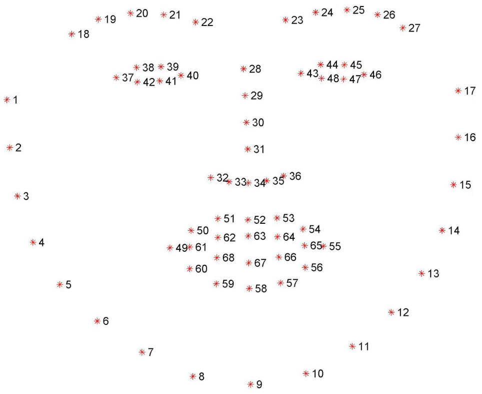


Figure 3.1.2 The 68 Facial Landmarks identified in dlib's model

The model then overlays the landmarks onto a face and modifies their position accordingly. This can be visualised in Fig 3.1.3 which shows screenshots of an implementation of the model onto a video feed. Each landmark is represented by a red dot on the overlay.

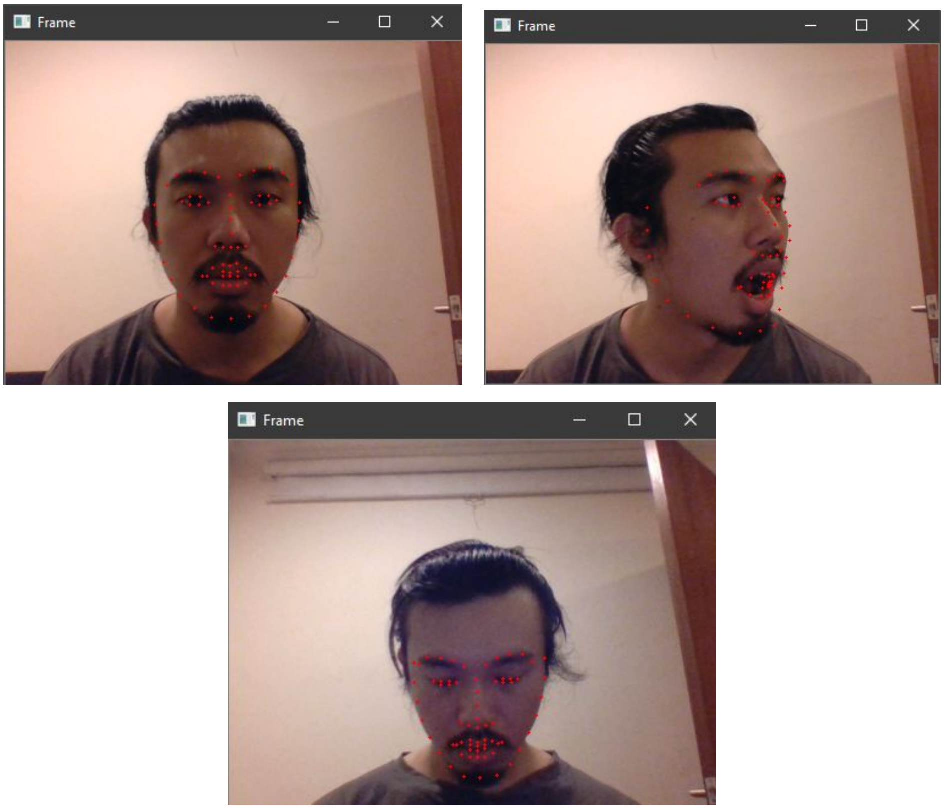
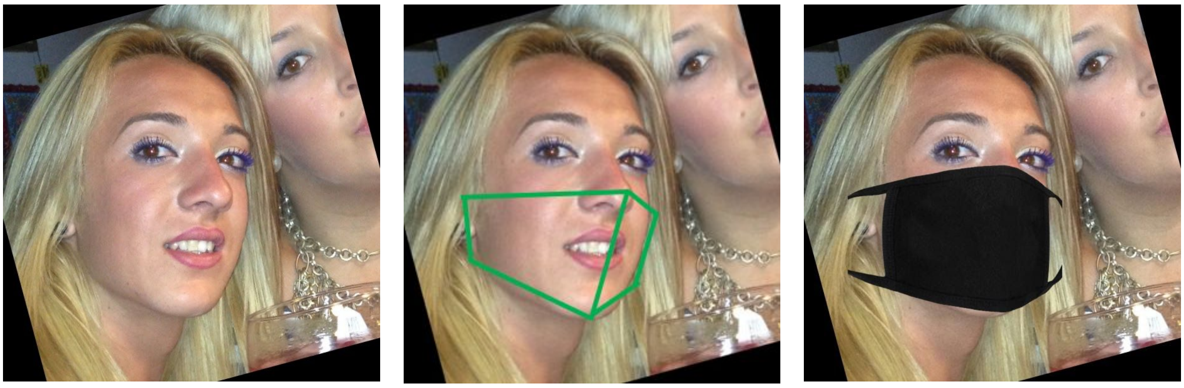


Figure 3.1.3 Applying shape_predictor_68_face_landmarks onto video feed.

3.2.4 Mask Overlay

We took reference from a code repository from Bhandary (2020) [17] that aimed to detect if faces were wearing masks or not, to build the data augmentation function. It would use the facial landmarks corresponding to the nose, chin and sides of the face to overlay a given mask image onto the face.

To account for non-frontalized images, the mask images would be sliced in half, then each half would be individually placed onto the face based on the appropriate landmarks. This modification can be seen in Fig 3.1.4.

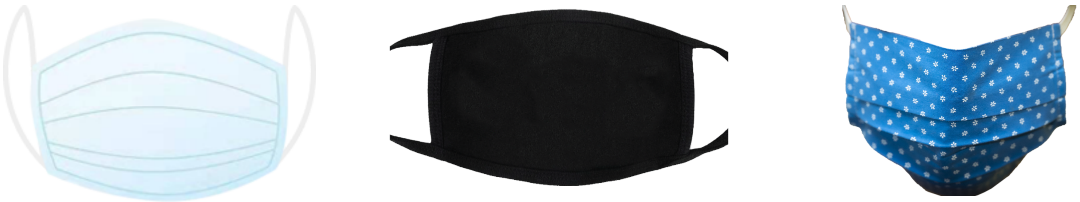


Original image *Features identified* *Mask overlaid*

Figure 3.1.4 Process of mask overlay

We chose a variety of mask designs to reflect the different types of masks people wear in reality. The popularity of cloth masks has increased in recent years, so populating the model with realistic data will make it more robust to real-world conditions.

The mask overlays used in this process were chosen to represent the 3 general classes of masks people wear, light, dark and patterned, as seen below in Fig 3.1.5. Patterned masks in particular could present a challenge for the model as in reality, there is an infinite number of patterns that could be printed onto a cloth mask, some of which may confuse landmark detection, hence, we simplify our problem by introducing a random, noisy design.



Light mask overlay *Dark mask overlay* *Patterned mask overlay*

Figure 3.1.5 Broad categories of masks worn in public under COVID-19 regulations

3.3 Model Training

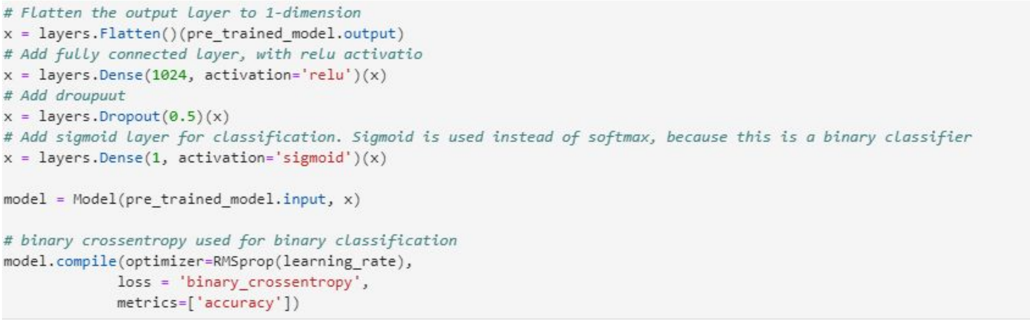
After the previous preprocessing phase, we now have a dataset of augmented images with masks on them. We proceed to the training phase, using cloud GPU platforms (Google Colan, Paper Space) as these have reasonably long usage provided to public users.

3.3.1 Transfer Learning

For this assignment, we used the Transfer Learning technique. Transfer Learning takes features learned on a previous problem and applying it on a new and similar problem. Transfer Learning helped to solve the problem of having a small dataset, which is insufficient to train a vast model such as InceptionV3 which has 331 layers and 23,851,784 parameters. Moreover, given our limited computational resources, transfer learning is the best approach to develop a robust model.

In this case, we used pre-trained models from Keras, namely, InceptionV3 and ResNet50V2. These models were trained on the *ImageNet* dataset, which is a commonly used visual database.

In terms of the implementation, we loaded the pre-trained weights of InceptionV3 and ResNet50V2 models, and freeze all layers, making them untrainable. We then replace the final loss output layer with a set of dense layers (with dropout), as seen in Figure 4.1.



```
# Flatten the output layer to 1-dimension
x = layers.Flatten()(pre_trained_model.output)
# Add fully connected layer, with relu activatio
x = layers.Dense(1024, activation='relu')(x)
# Add droupuut
x = layers.Dropout(0.5)(x)
# Add sigmoid layer for classification. Sigmoid is used instead of softmax, because this is a binary classifier
x = layers.Dense(1, activation='sigmoid')(x)

model = Model(pre_trained_model.input, x)

# binary crossentropy used for binary classification
model.compile(optimizer=RMSprop(learning_rate),
              loss = 'binary_crossentropy',
              metrics=['accuracy'])
```

Figure 4.1: Code snippet of new layers added to InceptionV3

3.3.2 Hyperparameters and loss function

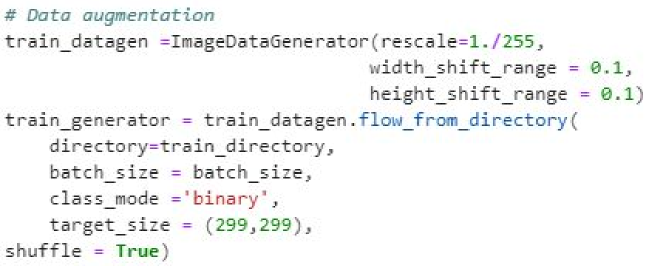
To ensure consistency between the training of both the *InceptionV3* as well as *ResNet50V2* models, we use the same hyperparameters for both trainings:

- **Optimizer**: RMSProp
- **Learning Rate**: 0.0001
- **Loss**: binary_crossentropy
- **Batch size**: 256
- **Colour mode**: RBG

The input shape for each model remains as the default input shape, (299,299,3) for InceptionV3 and (224,224,3) for ResNet50V2.

3.3.3 Training and validation setup

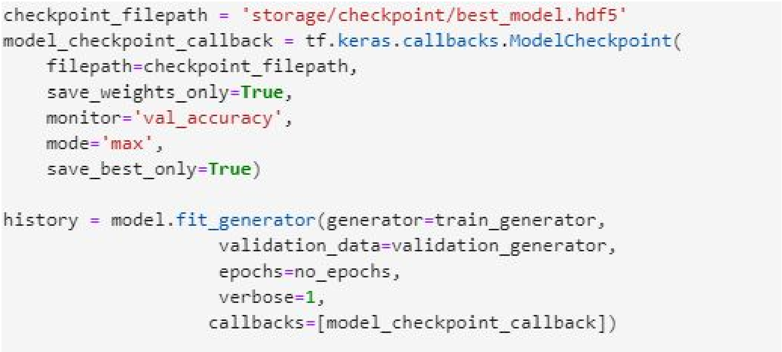
Generators: A training and validation generator was set up. Generators are used to reduce the memory consumption during training. Using the *ImageDataGenerator()* function, we rescale the RGB coefficient values in the image, such that the resultant values lie between 0 and 1. We also added *width_shift_range* and *height_shift_range* as a simple form of data augmentation, to increase the variance in the input data. The *flow_from_directory()* function is then called, which returns a DirectoryIterator object that reads input images from the specified directory. A similar method was used for the validation data. The code snippet can be seen in the figure below.



```
# Data augmentation
train_datagen =ImageDataGenerator(rescale=1./255,
                                  width_shift_range = 0.1,
                                  height_shift_range = 0.1)
train_generator = train_datagen.flow_from_directory(
    directory=train_directory,
    batch_size = batch_size,
    class_mode ='binary',
    target_size = (299,299),
shuffle = True)
```

(Figure 4.2: Keras generator for training data)

Using checkpoint: A callback was used during the model training. This is used to save the weights of the model where the validation accuracy is the highest. These weights will be used as the weights for the final model, which will be used for predictions and evaluations.



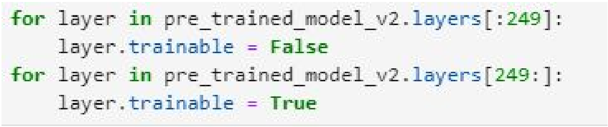
```
checkpoint_filepath = 'storage/checkpoint/best_model.hdf5'
model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_weights_only=True,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)

history = model.fit_generator(generator=train_generator,
                    validation_data=validation_generator,
                    epochs=no_epochs,
                    verbose=1,
                   callbacks=[model_checkpoint_callback])
```

(Figure 4.3: Setting up callback before fitting the model)

3.3.4 Fine-tuning

To optimise our models, we decided to fine-tune them after the first round of training. This is done by unfreezing some of the lower layers of the model. The motivation for doing so is that the SOTA models we use were pre-trained on *ImageNet*, while our task is to train the models on the processed Adience dataset. Imagenet consists of images of objects and animals as well. Given that these 2 datasets are inherently very different, we can expect the feature extraction of the pre-trained model to not be well-tailored for our new task. As such, we have also decided to experiment with this method of fine-tuning, for more optimal feature extraction.

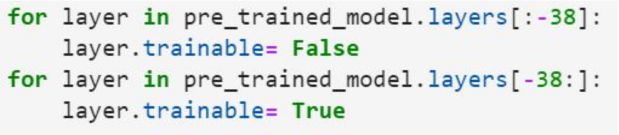
In terms of implementation, for the InceptionV3, we decided to unfreeze layer 249 onwards, which are the top 2 inception blocks in the InceptionV3 architecture. A code snippet for this is shown in Figure 4.4..



```
for layer in pre_trained_model_v2.layers[:249]:
    layer.trainable = False
for layer in pre_trained_model_v2.layers[249:]:
    layer.trainable = True
```

Figure 4.4: Unfreezing the top 2 inception blocks in InceptionV3

A similar approach was done for the ResNet50V2, where we decided to unfreeze the last 38 layers, which is the last convolution block of the 5 blocks in the architecture.



```
for layer in pre_trained_model.layers[:-38]:
    layer.trainable= False
for layer in pre_trained_model.layers[-38:]:
    layer.trainable= True
```

Figure 4.5: Unfreezing the last convolution block in ResNet50V2

4 Experiments & Results

All experimental codes and graphics of final results are available for viewing at https://github.com/omofish/CZ4062_final_project.

4.1 Comparison of training results

4.1.1 Training and validation accuracies and losses

In total, we trained a total of **4** models:

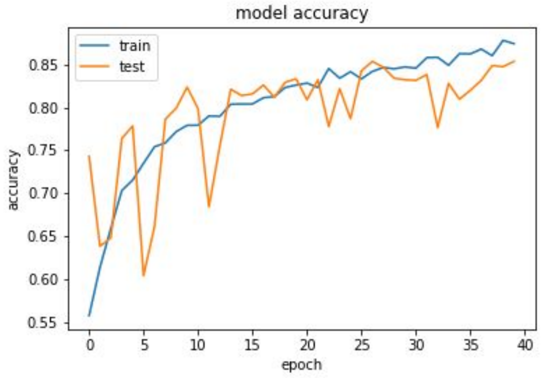
1) **InceptionV3_freeze_all**: *InceptionV3* where all original layers are frozen
2) **InceptionV3_fine-tuned**: *InceptionV3* where layers 0-249 are frozen; last 62 layers are unfrozen
3) **ResNet50V2_freeze_all**: *Resnet50V2* where all original layers are frozen
4) **ResNet50V2_fine-tuned**: *Resnet50V2* where layers 0-154 are frozen; last 38 layers are unfrozen

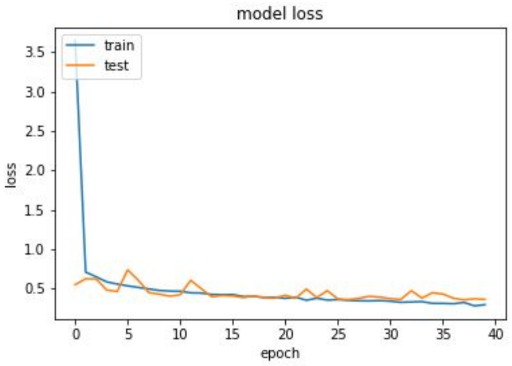
This section shows the results of the training process. For each chosen model, we plot 2 graphs: (1) Train and validation accuracy against epochs, and (2) Train and validation loss against epochs. The graphs can be seen in the subsections below.

In terms of accuracies, we see that the peak validation accuracies across all 4 models are above 85% despite limited computational resources and training epochs. In particular, the *ResNet50V2_fine_tuned* achieved the highest validation accuracy of about 94%, showing great potential. One point to note is that while the training for *InceptionV3_freeze_all* and *ResNet50V2_freeze_all* might not have converged, the fine-tuned models have both seemed to converge. This can be seen from the plateau in the training and loss graphs for *InceptionV3_fine-tuned* and *ResNet50V2_fine-tuned.*

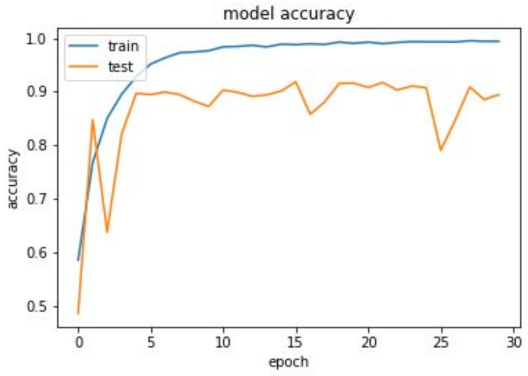
In terms of volatility, we see that the ResNet models have validation accuracies and losses that fluctuate a lot more than the Inception models. We postulate that the following reasons could have caused this (1) The regularization is too low, resulting in insufficient smoothing (2) The batch size is too small, resulting in uneven updates or (3) The learning rate is too large.

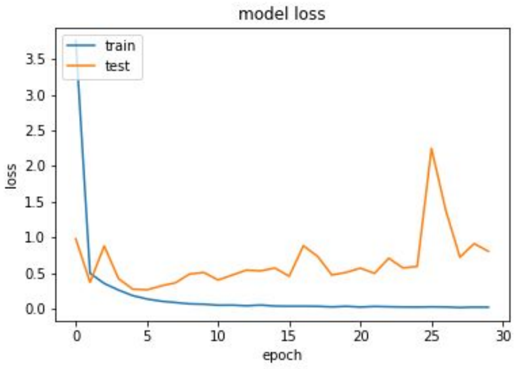
InceptionV3_freeze_all:
Img_size (299,299), 40 epochs



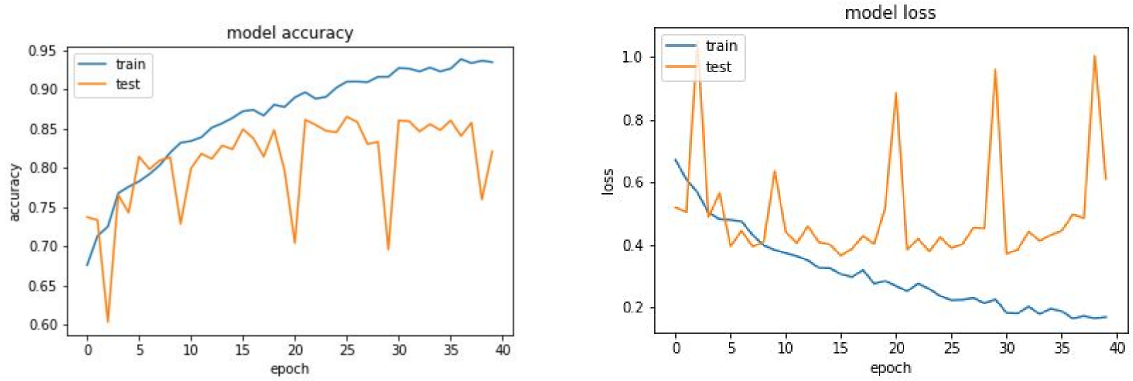


InceptionV3_fine-tuned:
Img_size (299,299), 30 epochs

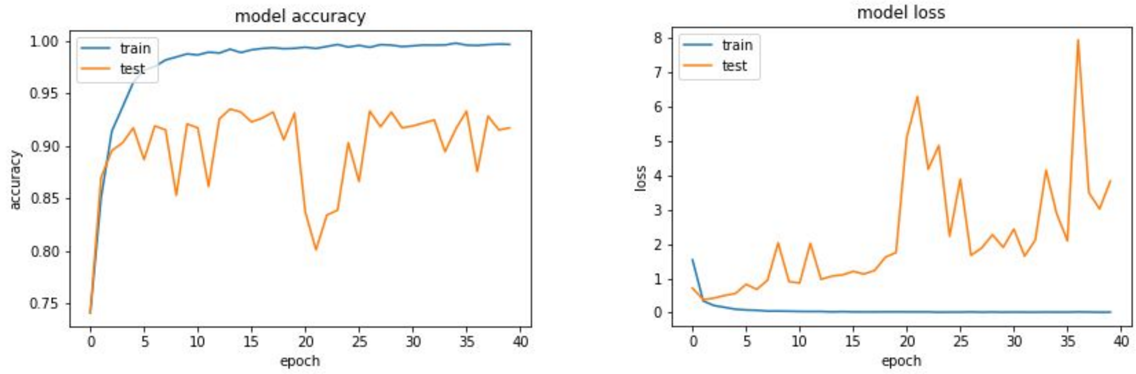




ResNet50V2_freeze_all:
Img_size (224,224), 40 epochs

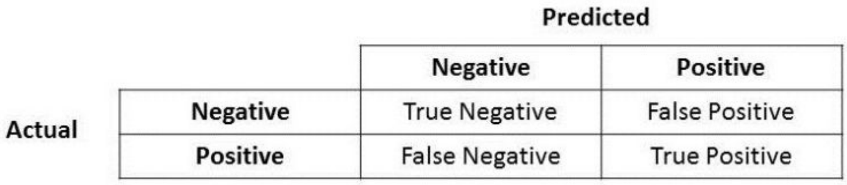


ResNet50V2_fine-tuned:
Img_size (224,224), 40 epochs



4.2 Comparison of evaluation results

To evaluate the level of accuracies of all our models, we loaded the weight of the model saved during the callback (max validation accuracy), and used it to predict the test data. A confusion matrix for each model was plotted, and we used it to calculate important metrics for comparison. The confusion matrix is generated using the *sklearn* library, which has a format as shown in the Figures 4.2 and 4.3 below. A predicted value of *Negative* refers to *Female,* and *Positive* refers to *Male.*

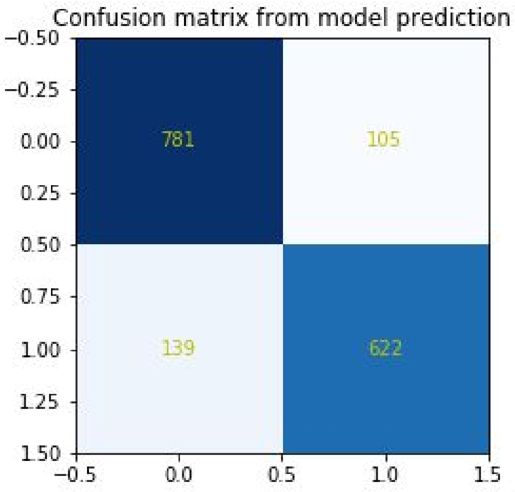
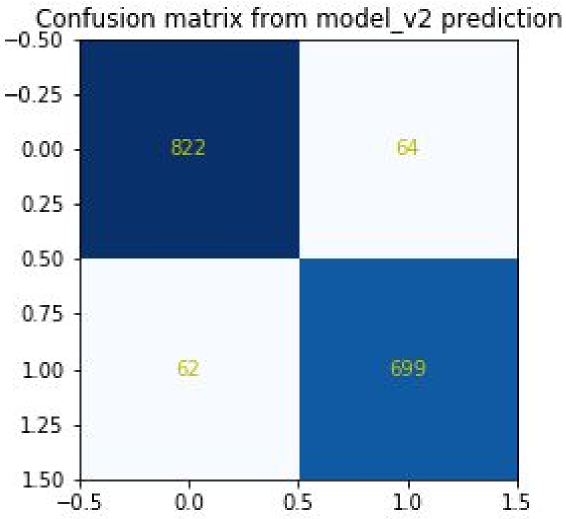


		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

(Figure 4.1: Format of confusion matrix results)

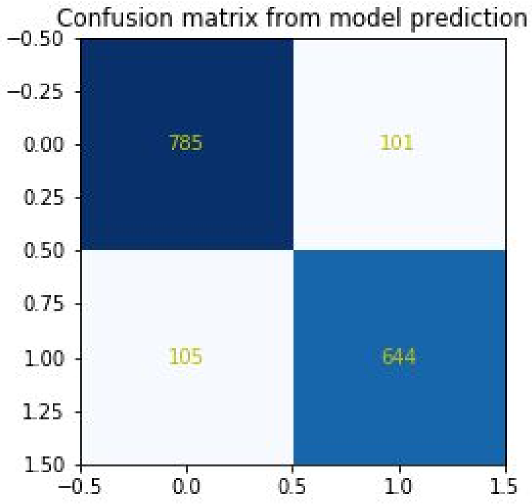
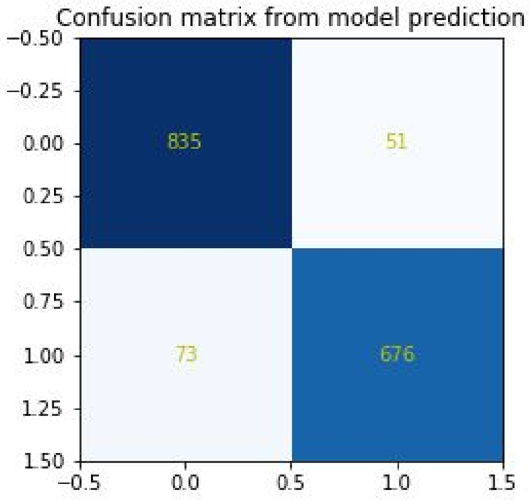
4.2.1 InceptionV3

Figure 4.2: Confusion matrices for InceptionV3_freeze_all and InceptionV3_fine-tuned

InceptionV3_freeze_all	**InceptionV3_fine-tuned**
	

4.2.2 ResNet50V2

Figure 4.3: Confusion matrices for ResNet50V2_freeze_all and ResNet50V2_fine-tuned

ResNet50V2_freeze_all	**ResNet50V2_fine-tuned**
	

4.2.3 Evaluation Scores

The following table (Figure 4.4) shows the Accuracy, Precision and Recall for each of the 4 models. These are calculated from the confusion matrices that we derived.

We see that **ResNet50V2_fine-tuned** is the top performing model, with the highest precision accuracy of **92.4%**. In the next segment, we investigate these metrics in greater detail to compare the performance of all 4 models.

Figure 4.4: Accuracy, Precision and Recall metrics for each model

Model	**Accuracy (%)**	**Precision male (%)**	**Precision female (%)**	**Recall male (%)**	**Recall female (%)**
InceptionV3_freeze_all	85.2	85.6	84.9	81.7	88.1
InceptionV3_fine-tuned	92.3	91.6	93.0	91.9	92.8
ResNet50V2_freeze_all	87.4	86.4	88.2	86.0	88.6
ResNet50V2_fine-tuned	92.4	93.0	92.0	90.3	94.2

5 Discussion

5.1 Determining the best approach and model

In this section, using the evaluation results above, we discuss a few key insights, and compare the effectiveness of the higher-performing models.

Effect of fine-tuning: By fine-tuning or unfreezing more layers, we see a better performance of the models. We see a 6.9% increase in accuracy for the *InceptionV3* model, and a 5% increase for the *ResNet50V2* models. This is as expected, because as mentioned before, the pretrained-weights are derived from a training on the ImageNet dataset, which is inherently different from the preprocessed Adience dataset that we are using. We can conclude that fine-tuning has helped the model learn lower-level features more effectively.

Accuracies, Precision and Recall: *ResNet50V2_fine-tuned* and *InceptionV3_fine-tuned* have comparable accuracy scores of 92.4% and *92.3%*. In choosing the better of the 2 models, it might be wise to compare their precision and recall scores as well. In terms of precision, we see that both models have different disadvantages, where the *ResNet50V2_fine-tuned* is more precise in predicting male, while *InceptionV3_fine-tuned* is more precise in predicting *female*. In terms of recall however, we see a more noticeable disparity. Recall, unlike precision, can be thought of as the model's ability to find all data points of interest. While the *ResNet50V2_fine-tuned* has a high recall for predicting *female*, it has a substantially lower recall for predicting *male* when compared to the *InceptionV3_fine-tuned.* This means that given that a subject/image is of a male, the model will have a lower chance of predicting that it is male.
Thus we conclude that while *ResNet50_fine-tuned* is the best model for overall accuracy, and *InceptionV3_fine-tuned* is a better model when we want to maximise the minimum recall for each class (ie. has the highest max(min(recall_male, recall_female))).

5.2 Comparing speed of the models

While the accuracy of our chosen model is important, the training and inference speed are crucial as well. This is especially important for real-time or time-sensitive use cases of automatic gender detection.

The following table compares the different parameters and metrics of InceptionV3 and ResNet50 [18]. While the InceptionV3 model has fewer parameters than the ResNet50, it has 1.5 times the number of floating point operations. This would mean that 1.5 times more addition and multiplication operations are required to run one full instance of InceptionV3 as compared to ResNet50. This could significantly lengthen the training and inference time of InceptionV3.

Figure 5.1: Comparing parameters and metrics of InceptionV3 and ReNet50

Model	**Number of parameters** *(millions)*	**FLOPs*** *(GFLOPs)*	**Number of layers****	**Trained Size** *(MB)*
InceptionV3	24	6	331	97
ResNet50	26	4	177	104

*FLOPs = floating point operations

**Number of layers in the *Keras* implementation, not the high-level layers in the schematic design

This is proven to be true in a research on benchmark analysis on neural network architectures [19]. The research showed that the inference time of InceptionV3 on 1 batch of data (79.39ms) is about 1.5 times that of ResNet50 (53.09ms). Thus, when considering training and inference speed, we conclude that ResNet50 outperforms InceptionV3.

5.3 Applications

While gender classification by itself has limited practical applications, it is an important building block on which other systems may be built. By training our model on masked faces, we allow it to function much more robustly in public settings, where wearing masks is still a widely-enforced safety requirement.

Gender classification on masked faces could be used to collect demographic data in public locations. A camera system placed at the front of mall stores could be used to better understand customer gender demographics, which will certainly be useful information to marketers. Businesses that rely on gender differences to determine their marketing approach may also use this information to improve their effectiveness.

Another potential use case lies in augmenting human-facing computer interfaces. Robotic assistants and smart terminals may use it to ascertain the proper salutation to use when greeting users. UX designers may also want to tailor their product experiences to the gender of the user. Automatic gender detection would be of great help in this area.

5.4 Future work

While our model already performs well on the task of classifying gender, we surmise it is possible to improve it further. One simple way would be to localise the model to the area it is intended to be used in. The training data is currently sourced from the Adience dataset, which comes from Flickr uploads and therefore is unlikely to conform to Singapore's racial demographic. Therefore, if we were to develop this model for practical use in a local context, it may be wise to retrain the top few layers of the model with a different set of images that reflect the local population.

As for possible extensions to the scope of our model, it may also be possible to train the model to classify not just the gender, but also the age of faces. There are plenty of facial characteristics, such as the relative size of individual features or presence of wrinkles that can help distinguish between young and old of either gender. This would help to make demographic data generated by this model become even more specific.

Lastly, in terms of areas of improvement, we can definitely experiment with more hyperparameter tuning and other training configurations. This could be experimenting with (1) number of layers to unfreeze, (2) smaller batch size for more stable training (3)use novel loss functions like Triplet Loss. Our model could also definitely improve if we were able to gather a more realistic set of data.

5.5 Conclusion

In this project, we looked at using 2 SOTA models, namely the *InceptionV3* as well as the *ResNet50V2* neural network architectures to predict the gender of people wearing face masks. We preprocessed the Adience dataset to create a cleaned set of images of people wearing face masks. Following that, we trained the models and analysed the evaluation results. We managed to achieve a reasonably high accuracy of 92.4%, using the *ResNet50V2* by fine-tuning the models in the transfer learning process. Our team also discussed many factors influencing the effectiveness of an automatic gender recognition system, including precision, recall and training/inference speed of the model.

Overall, we have shown that our model is able to perform the cognitively complex task of distinguishing genders fairly well, even when the faces are partially obstructed by masks. It is also shown to not require much preprocessing of test data, hence we are confident in its robustness if used in the real world. The fact that it is able to recognize facial features in the presences of masks also raises some ethical questions. Will AI surveillance eventually be able to bypass all human efforts to conceal our identities? Would we eventually need to consider our own faces as things to be hidden in the public space? Only time will tell.

6 References

1. Vaddi, R. S., Boggavarapu, L. N. P., Vankayalapati, H. D. and Anne, K. R. (2012). Real Time Human Gender Detection Based on Facial Features and Connected Component Analysis. (2012). Wireless Networks and Computational Intelligence. ICIP 2012. Communications in Computer and Information Science, vol 292. Springer, Berlin, Heidelberg.
2. Nistor, S. C., Marina, A. C., Darabant, A. S. and Borza, D. (2017). Automatic gender recognition for “in the wild” facial images using convolutional neural networks. 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP). DOI: 10.1109/ICCP.2017.8117018.
3. Phillips, P. J., Wechsler, H., Huang, J. and Rauss, P. J. (1998). The feret database and evaluation procedure for face-recognition algorithms. Image and vision computing, 16(5):295–306, 1998.
4. Ahonen, T., Hadid, A. and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 12, pp. 2037–2041, Dec. 2006.
5. Liu, C. and Wechsler, H. (2002). Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. IEEE Trans. Image Process., vol. 11, no. 4, pp. 467–476, Apr. 2002.
6. Huang, G. B., Ramesh, M., Berg, T. and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report, Technical Report 07-49, University of Massachusetts, Amherst, 2007.
7. Wolf, L., Hassner, T. and Taigman, Y. (2008). Descriptor based methods in the wild. Proc. Post-ECCV Faces Real-Life Images Workshop, 2008.
8. Eidinger, E., Enbar, R. and Hassner, T. (2014). Age and gender estimation of unfiltered faces. Trans. on Inform. Forensics and Security, 9(12), 2014.
9. Levi, G. and Hassner, T. (2015). Age and Gender Classification using Convolutional Neural Networks. 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). DOI: 10.1109/CVPRW.2015.7301352.
10. Szegedy, C. et al (2014). Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). DOI: 10.1109/CVPR.2015.7298594.
11. He, K. et al (2015). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). DOI: 10.1109/CVPR.2016.90.
12. Zhang, K. et al (2017). Age Group and Gender Estimation in the Wild With Deep RoR Architecture. arXiv:1710.02985.
13. McNeely-White, D., Beveridge, J. R. and Draper, B. A. (2019). Inception and ResNet features are (almost) equivalent. *Cognitive Systems Research 59 (2020), pp. 312-318*.
14. Serengil, S. I. (2020). Face Alignment for Face Recognition in Python within OpenCV. Retrieved from: https://sefiks.com/2020/02/23/face-alignment-for-face-recognition-in-python-within-opencv/

15. Geitgey, A. (2018). Face Recognition. Retrieved from: https://pypi.org/project/face-recognition/
16. Google (2020). dlib C++ library. Retrieved from: http://dlib.net/
17. Bhandary, P. (2020). Mask Classifier. Retrieved from: https://github.com/prajnasb/observations
18. Samuel. (2019). covnet-burden. Retrieved from https://github.com/albanie/convnet-burden
19. Bianco, S., & Cadene, R. (2018). Benchmark Analysis of Representative Deep Neural Network Architectures. cs.CV. r 10.1109/ACCESS.2017.DOI