NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CZ4042 Project Report

Members:

Ng Chen Ee Kenneth (U1721316F)

Goh Yong Wei (U1722195H)

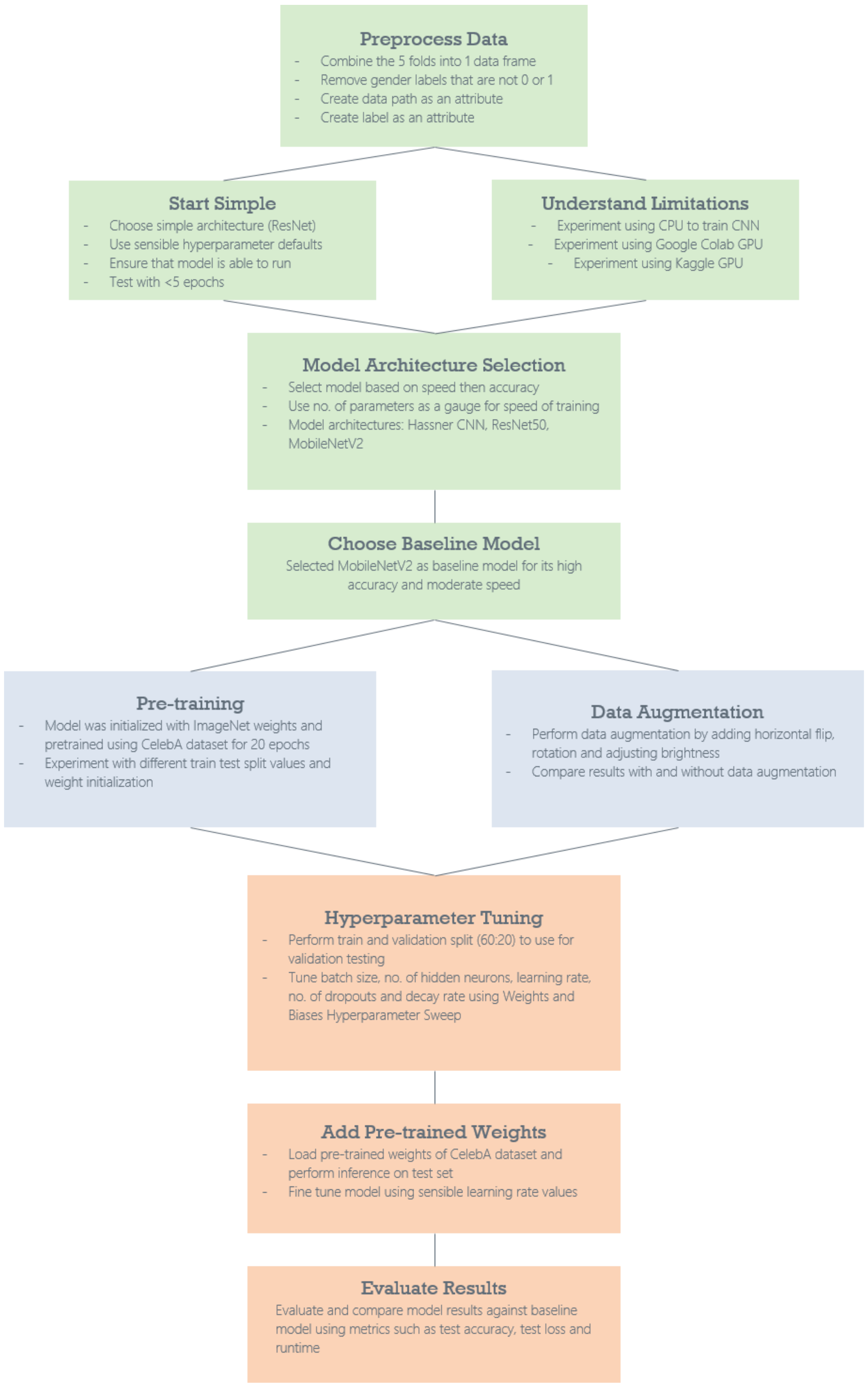
Ngo Jun Hao Jason (U1721978B)

Table of Contents

Project Objective

Automatic gender classification has been used in many applications including image analysis on social platforms. The goal of this project is to classify the gender of faces in an image.

Overview


Preprocess Data
- Combine the 5 folds into 1 data frame
- Remove gender labels that are not 0 or 1
- Create data path as an attribute
- Create label as an attribute
Start Simple
- Choose simple architecture (ResNet)
- Use sensible hyperparameter defaults
- Ensure that model is able to run
- Test with <5 epochs
Understand Limitations
- Experiment using CPU to train CNN
- Experiment using Google Colab GPU
- Experiment using Kaggle GPU
Model Architecture Selection
- Select model based on speed then accuracy
- Use no. of parameters as a gauge for speed of training
- Model architectures: Hassner CNN, ResNet50, MobileNetV2
Choose Baseline Model
Selected MobileNetV2 as baseline model for its high accuracy and moderate speed
Pre-training
- Model was initialized with ImageNet weights and pretrained using CelebA dataset for 20 epochs
- Experiment with different train test split values and weight initialization
Data Augmentation
- Perform data augmentation by adding horizontal flip, rotation and adjusting brightness
- Compare results with and without data augmentation
Hyperparameter Tuning
- Perform train and validation split (60:20) to use for validation testing
- Tune batch size, no. of hidden neurons, learning rate, no. of dropouts and decay rate using Weights and Biases Hyperparameter Sweep
Add Pre-trained Weights
- Load pre-trained weights of CelebA dataset and perform inference on test set
- Fine tune model using sensible learning rate values
Evaluate Results
Evaluate and compare model results against baseline model using metrics such as test accuracy, test loss and runtime

Data Exploration

CelebA

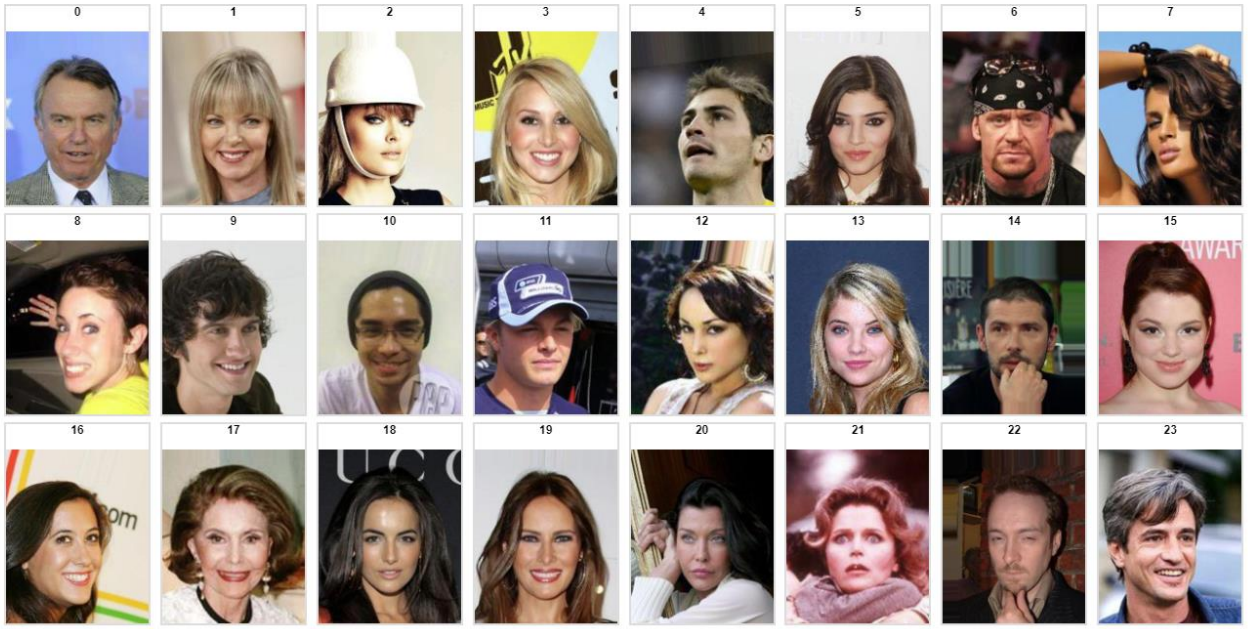


Figure 1: Images in CelebA dataset

Dataset Description

CelebFaces Attributes Dataset (CelebA) is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including:

- 10,177 number of identities,
- 202,599 number of face images, and
- 5 landmark locations, 40 binary attributes annotations per image.

The dataset can be employed as the training and test sets for the following computer vision tasks: face attribute recognition, face detection, landmark (or facial part) localization, and face editing & synthesis.

Adience

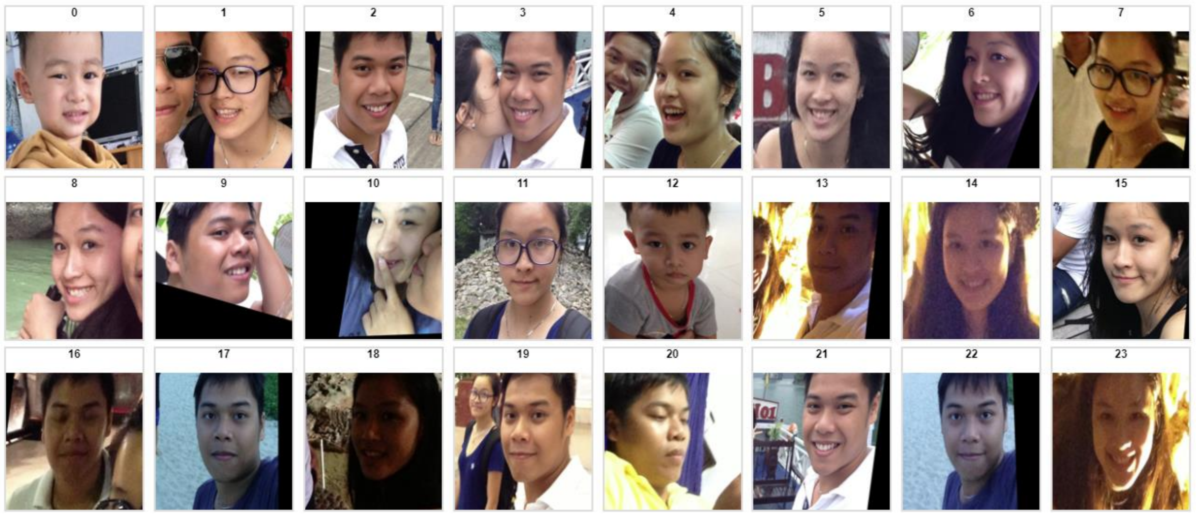


Figure 2: Aligned Images in Adience Dataset

Dataset Description

The Adience set consists of images automatically uploaded to Flickr from smart-phone devices. These images were uploaded without prior manual filtering, as is typically the case on media webpages, viewing conditions in these images are highly unconstrained, reflecting many of the real-world challenges of faces appearing in Internet images. Adience images therefore capture extreme variations in head pose, lightning conditions quality, and more. The breakdown of the dataset is as follows:

- 26580 images,
- 2284 subjects
- 8 Age Groups (0-2, 4-6, 8-13, 15-20, 25-32, 38-43, 48-53, 60-)
- Aligned or Original

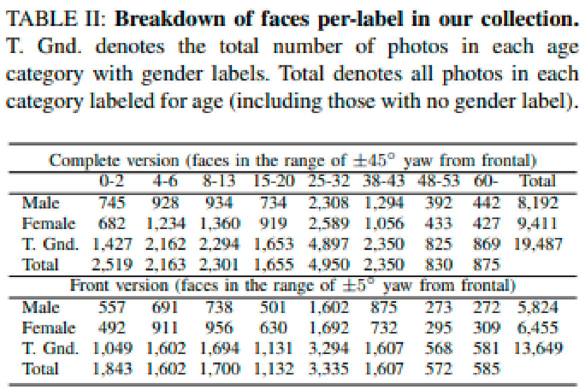


TABLE II: **Breakdown of faces per-label in our collection.** T. Gnd. denotes the total number of photos in each age category with gender labels. Total denotes all photos in each category labeled for age (including those with no gender label).

Complete version (faces in the range of $\pm 45^\circ$ yaw from frontal)									
	0-2	4-6	8-13	15-20	25-32	38-43	48-53	60-	Total
Male	745	928	934	734	2,308	1,294	392	442	8,192
Female	682	1,234	1,360	919	2,589	1,056	433	427	9,411
T. Gnd.	1,427	2,162	2,294	1,653	4,897	2,350	825	869	19,487
Total	2,519	2,163	2,301	1,655	4,950	2,350	830	875	
Front version (faces in the range of $\pm 5^\circ$ yaw from frontal)									
Male	557	691	738	501	1,602	875	273	272	5,824
Female	492	911	956	630	1,692	732	295	309	6,455
T. Gnd.	1,049	1,602	1,694	1,131	3,294	1,607	568	581	13,649
Total	1,843	1,602	1,700	1,132	3,335	1,607	572	585	

Figure 3: Age and Gender Estimation of Unfiltered Faces [1]

Choice of Dataset

There are 2 datasets made available for the Adience image dataset, original or aligned. The aligned dataset was selected in this project because it consists of photos that have been preprocessed using a frontalization method which helps to substantially boost the performance of face recognition systems. [2]

Changes to the Dataset

Due to the author's corrections of the gender labels, the size of the dataset has been reduced from 26580 to 13560. [3] With data preprocessing, the genders not belonging to 'f' or 'm' are removed and that leaves us with 12194 images. Figure 4 as shown below is the final number of images used for gender classification.

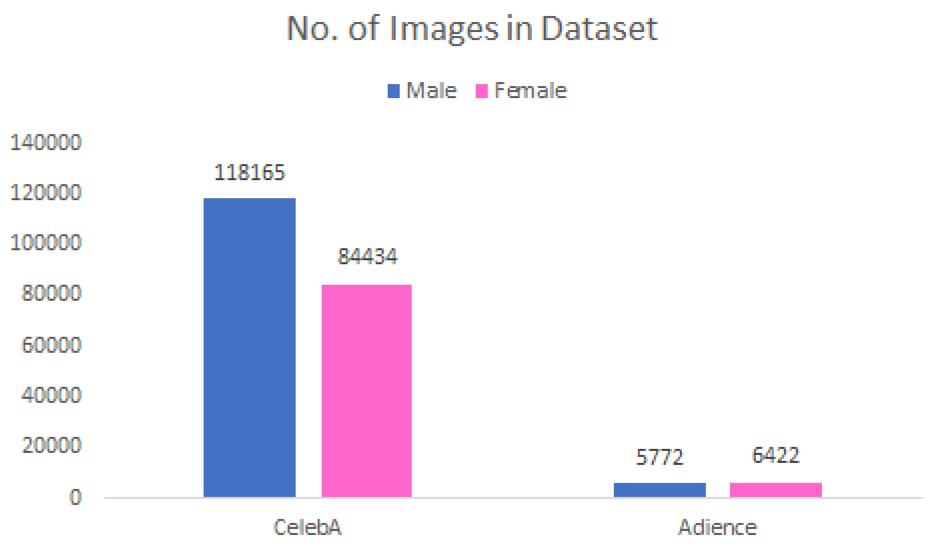


Figure 4: No. of images in each dataset (Adience & CelebA)

Data Preprocessing

We selected the keras .flow_from_dataframe() method in combination with ImageDataGenerator to load the Adience and CelebA dataset. This is because the .txt files are linked to the image location in unlabelled folders. Before loading the dataframes, the .txt files need to be preprocessed to remove irrelevant rows and columns.

Adience

There are a total of 5 .txt files labeled as fold_frontal_n.txt. We first transform the gender column for values 'f' to 0 and 'm' to 1. As there are a number of images that are not 'm' or 'f', they are removed from the dataset. The image directory is then created as a separate column (datadir) using user_id and original_image. These two attributes are then placed in a dataframe. After preprocessing the dataset, the dataset size is reduced from 13560 to 12194.

CelebA

Similarly for the CelebA dataset, we create 2 columns (image directory and gender). The original dataset consists of 202599 images with more than 40 different attributes. The Male attribute represents the gender of the individual in the image. -1 represents female while 1 represents male. We replace -1 with 0 in the Male attribute. The image directory is then created as a separate column (datadir). These two columns are then placed in a dataframe.

Modelling

Weights and Biases is a developer tool used to track, compare and visualize ML experiments. All the experiments that have been carried out in this project were recorded on Weights and Biases. The links to the table of results can be found in the Appendix section.

The process of modelling can be broken down into 7 parts.

1. Starting Simple
2. Understanding Limitations
3. Selection of Model Architecture
4. Data Augmentation
5. Hyperparameter Tuning
6. Pre-Training
7. Early Stopping

Starting Simple

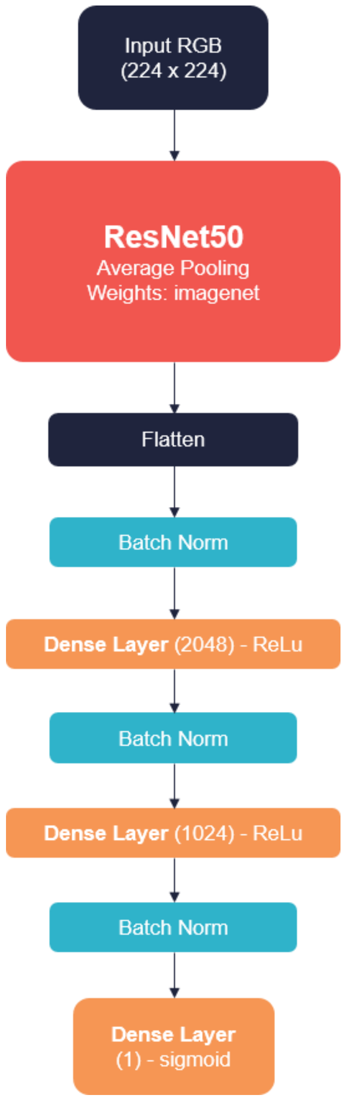


Figure 5: First Model

Training CNN models requires an extensive amount of time. The first step we took was to select a simple architecture such as ResNet50 to experiment on. The main goal is to ensure that the model is able to run smoothly without bugs. Sensible hyperparameter defaults were used. For example, we used imagenet weights as a base for ResNet50 and set the learning rate to 1e-3. No regularization was added. A batch size of 128 was selected. 2 fully connected layers of 2048 and 1024 neurons were added to progressively reduce the number of features as it reaches the base of the model.

Understanding Limitations

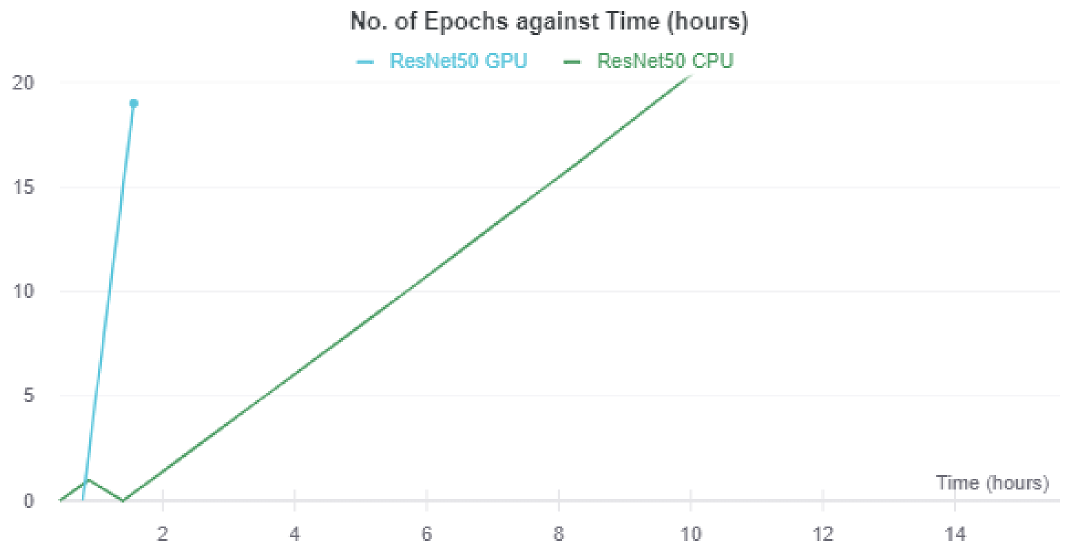


Figure 6: No. of Epochs against Time (hours)

Processor	**Runtime per epoch**
Intel(R) Core i7-7500U CPU @2.70GHz, 8GB RAM	26 mins 47 secs
Google Colab GPU Nvidia K80 0.82 GHz, 12GB RAM	7 mins 8 secs

We also faced an issue of training the datasets. Training the Adience dataset using CPU takes about 26 minutes per epoch which is not practical. We decided to use Google Colab GPU as it is free and is able to train the data at a faster rate. The training time was reduced to about 7 minutes per epoch. The Kaggle GPU was also used as it does not require the user to upload the image data to a set directory. That saves us the hassle of uploading more than 200,000 images in the CelebA dataset to the cloud.

Link to runs: https://wandb.ai/todayisagreatday/NN_Project_Test_Runs/
File: 3. Modelling/First Model -> resnet.py

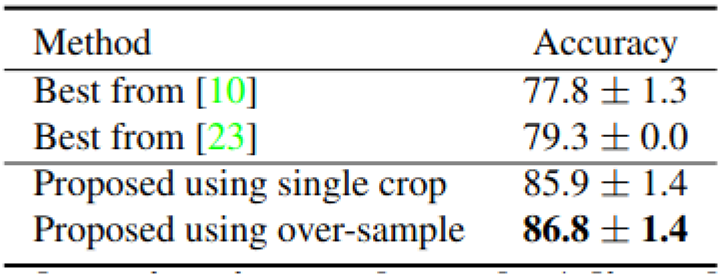
Selecting Model Architecture

Hassner CNN

Hassner CNN was the original CNN created for age and gender classification on the Adience dataset. [4]

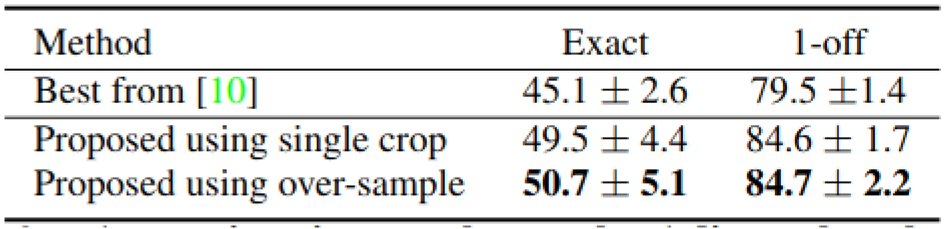
It has an architecture that is relatively simple compared to nowadays state-of-the-art CNNs. It is made up of 3 convolutional layers, followed by 2 fully-connected layers, all with ReLU activation except for the last layer which uses softmax. Additional, max pooling is done after each convolutional layer, and dropout is done for both fully-connected layers. Unlike in modern CNNs, Hassner CNN uses local response normalisation instead of batch normalisation after the first two convolutional layers.

It achieved decent results on the Adience dataset, though it is not possible to tell how it compares to other models on the same dataset since it is first to be used on this dataset.



Method	Accuracy
Best from [10]	77.8 ± 1.3
Best from [23]	79.3 ± 0.0
Proposed using single crop	85.9 ± 1.4
Proposed using over-sample	$\mathbf{86.8} \pm \mathbf{1.4}$

Figure 7: Hassner CNN's performance for gender estimation on the Adience dataset



Method	Exact	1-off
Best from [10]	45.1 ± 2.6	79.5 ± 1.4
Proposed using single crop	49.5 ± 4.4	84.6 ± 1.7
Proposed using over-sample	$\mathbf{50.7} \pm \mathbf{5.1}$	$\mathbf{84.7} \pm \mathbf{2.2}$

Figure 8: Hassner CNN's performance for age estimation on the Adience dataset

For our implementation of Hassner CNN, although we try to be faithful to the original design, we have made some practical changes. In the original model, the images are resized to a height and width of 256x256, before being cropped into 227x227 as input into the model. In our implementation, we used an input size of 224x224 to be consistent with other models that we were considering. The smaller input meant a reduced amount of parameters (which is still a whopping 115,023,873 parameters), at the expense of accuracy. Furthermore, the original model uses softmax for activation in the output layer, since it needs to perform classification of different age groups in addition to gender classification. For this project, the objective is to classify genders only (i.e. binary classification), so we used sigmoid which is sufficient.

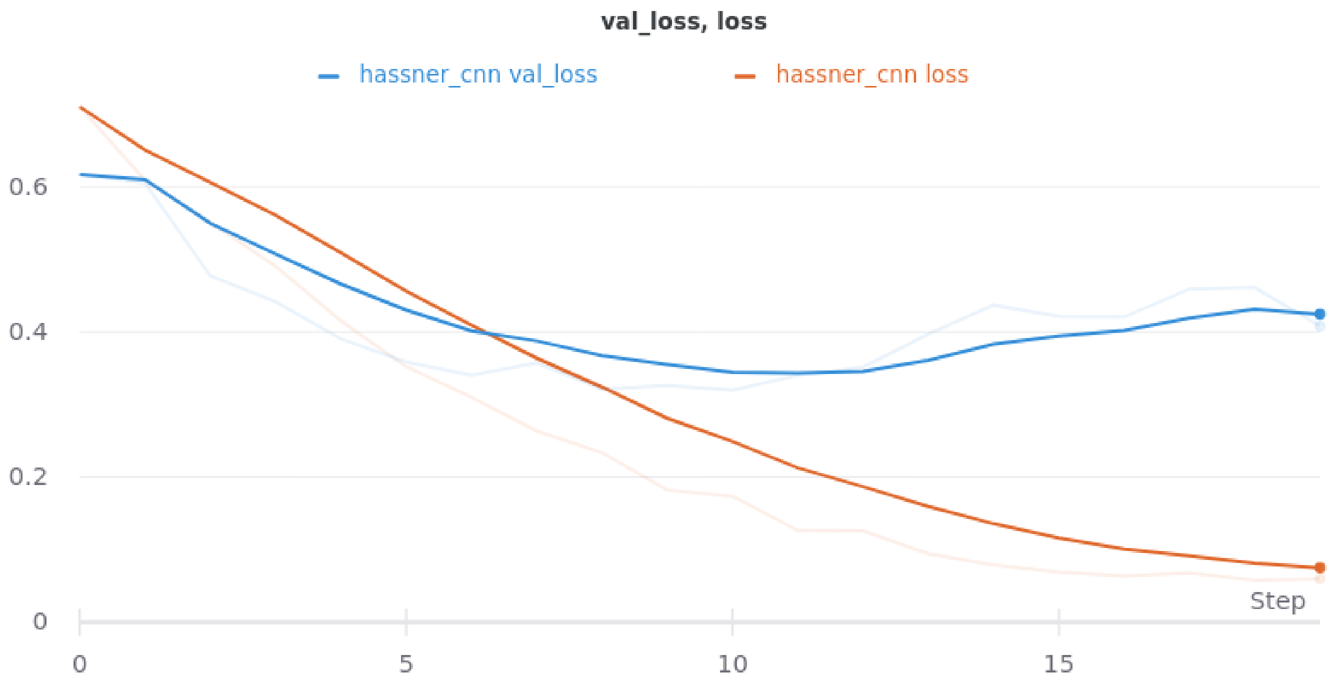


Figure 9: validation and training loss for Hassner CNN, for gender classification on Adience dataset

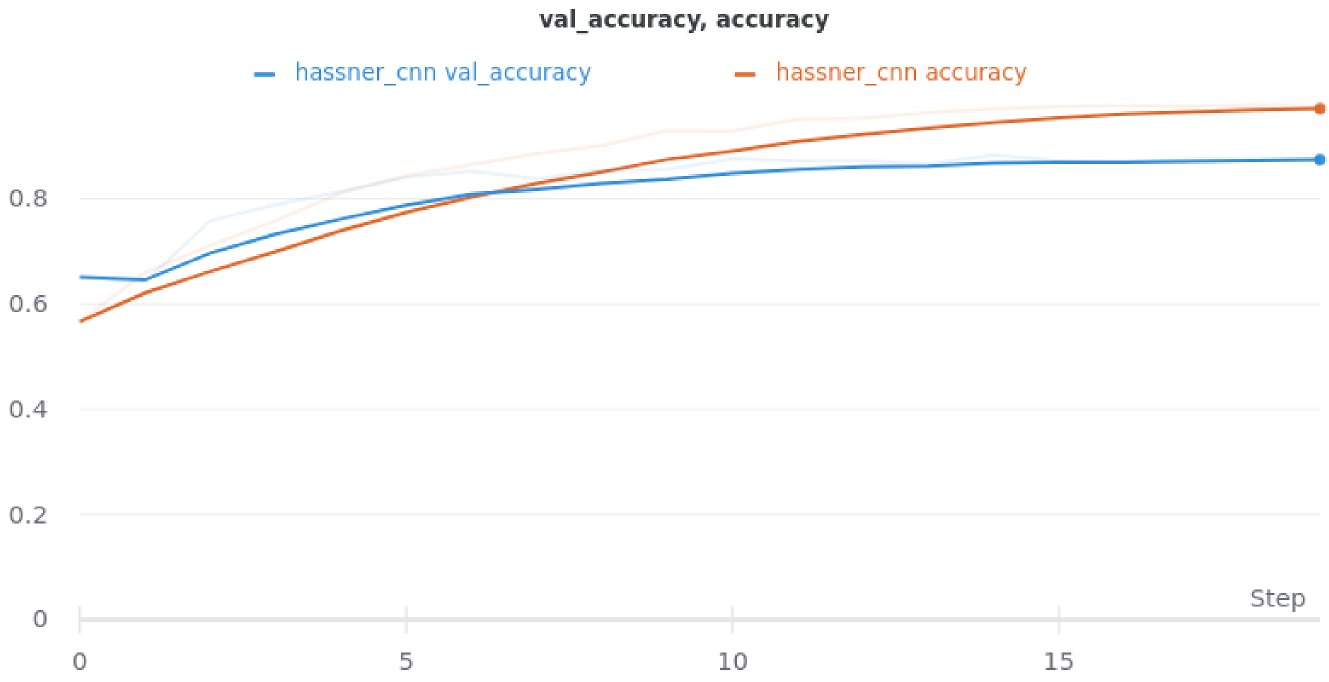
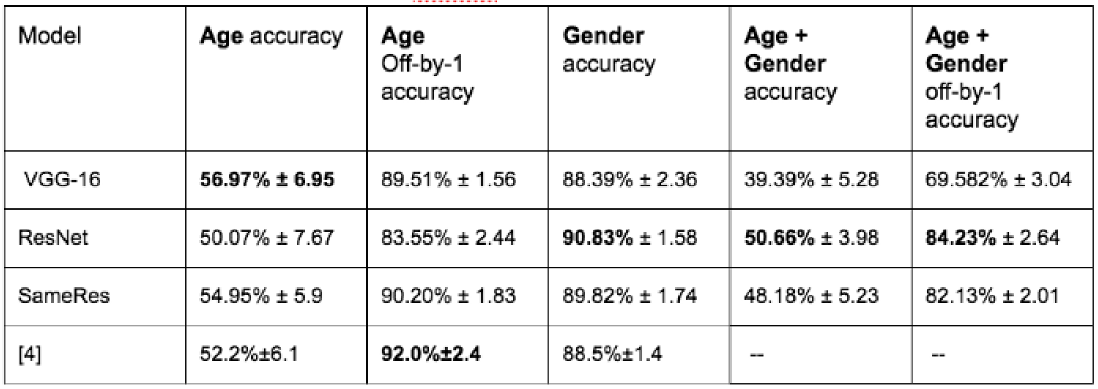


Figure 10: validation and training accuracy for Hassner CNN, for gender classification on Adience dataset

Upon evaluation of the model, we found that it performed well on the Adience dataset, with a validation accuracy of 0.8766.

Link to runs: https://wandb.ai/burntice/cz4042/runs/nml0vw6q?workspace=user-burntice
Notebook: '3. Modelling/Model Evaluation/hassner_cnn.ipynb'
Saved weights: '5. Saved Weights/Model Evaluation/hassner_cnn_eval.h5'

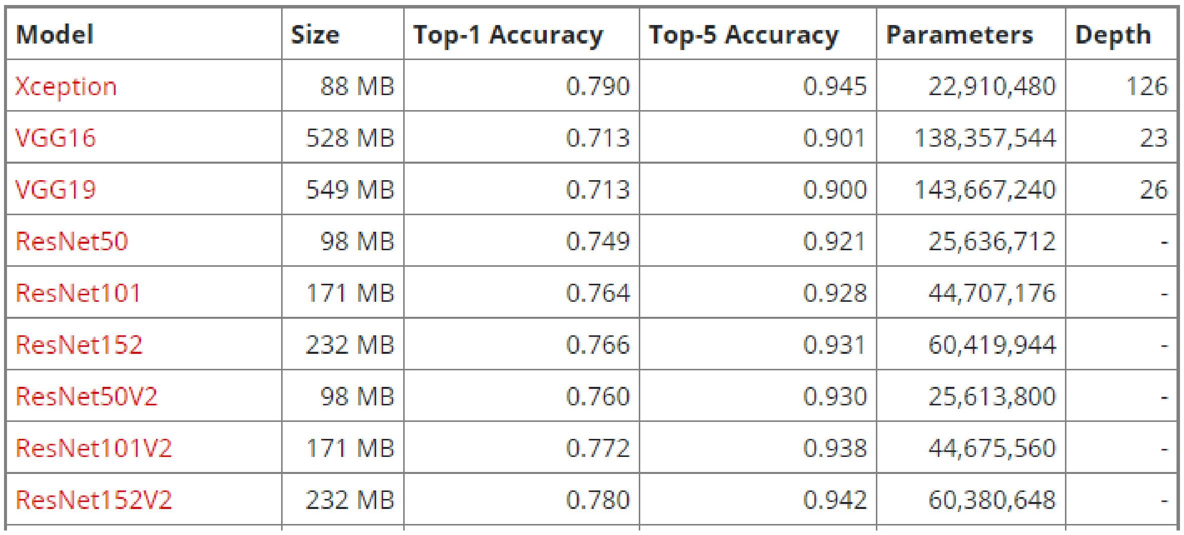
ResNet50



Model	**Age** accuracy	**Age** Off-by-1 accuracy	**Gender** accuracy	**Age + Gender** accuracy	**Age + Gender** off-by-1 accuracy
VGG-16	**56.97% ± 6.95**	89.51% ± 1.56	88.39% ± 2.36	39.39% ± 5.28	69.582% ± 3.04
ResNet	50.07% ± 7.67	83.55% ± 2.44	**90.83%** ± 1.58	**50.66%** ± 3.98	**84.23%** ± 2.64
SameRes	54.95% ± 5.9	90.20% ± 1.83	89.82% ± 1.74	48.18% ± 5.23	82.13% ± 2.01
[4]	52.2%±6.1	**92.0%±2.4**	88.5%±1.4	--	--

Figure 11: Gender Accuracy based on a 5-Fold Cross Validation by Andrey Karapetov

The ResNet model was selected as a possible architecture for image classification as it has won the 1st place on the ILSVRC 2015 classification task. [5] Additionally, it has over 60,000 Google citations and has been utilised by the creator of the Adience dataset (Andrey Karapetov). It has achieved a test score of 90.8% ± 1.58 when a 5-fold cross validation is done on the Adience data. [6]



Model	**Size**	**Top-1 Accuracy**	**Top-5 Accuracy**	**Parameters**	**Depth**
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-

Figure 12: State-of-the-Art deep learning models available in keras

The above figure shows the list of state-of-the-art models benchmarked on the ImageNet dataset. The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset. Considering the number of parameters, we decided to select ResNet50 as a possible option as it has the lowest number of parameters (25,636,712) among the ResNet models. Using an architecture with a lower number of parameters will reduce the overall runtime during training.

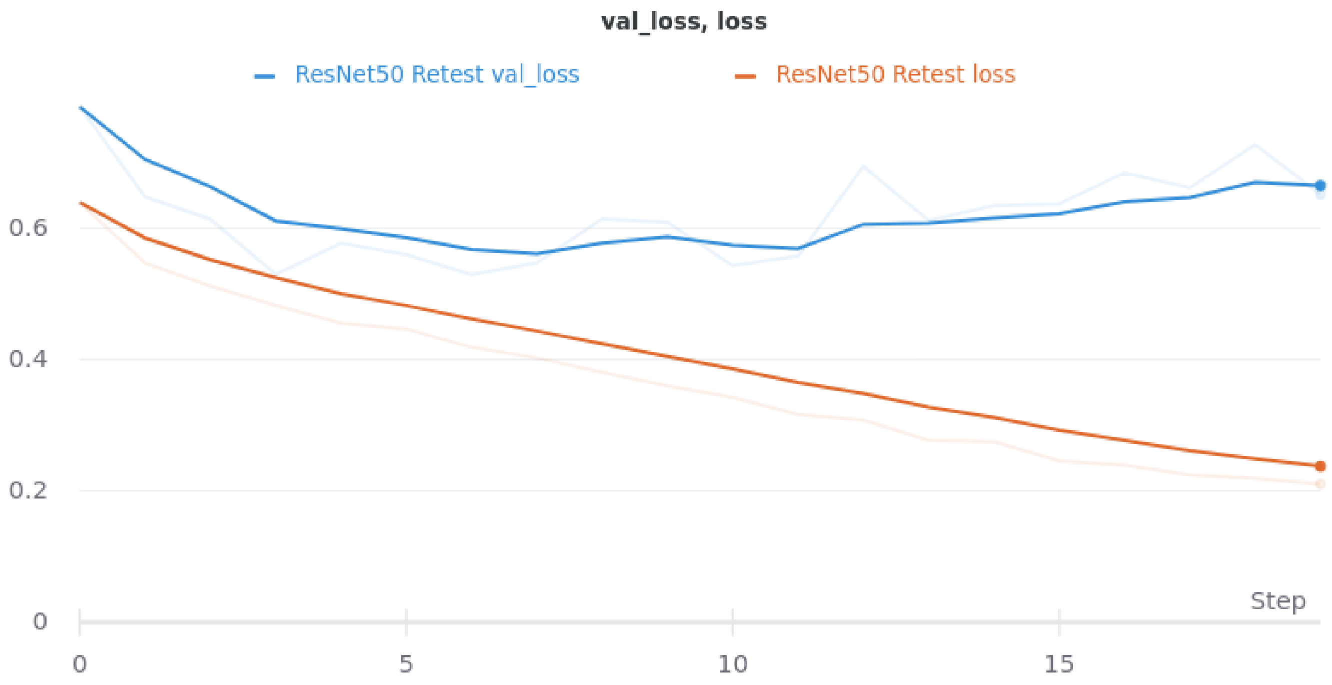


Figure 13: validation and training loss for ResNet50, for gender classification on Adience dataset

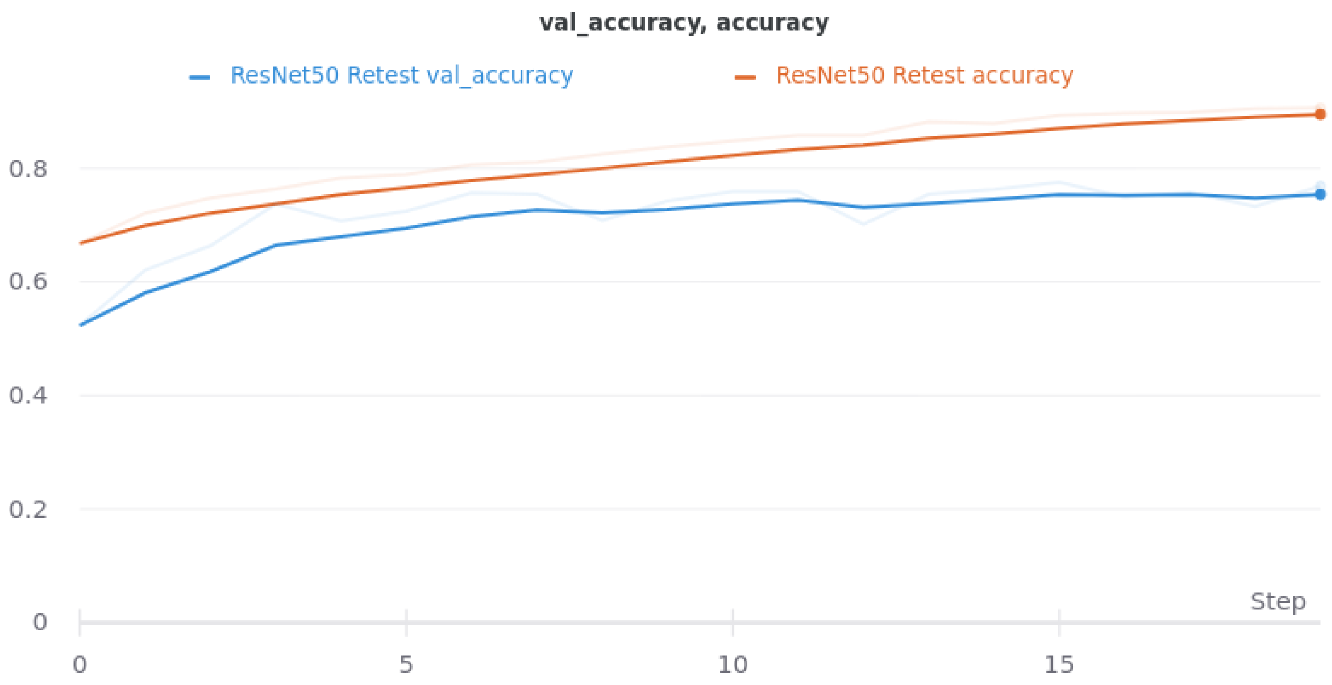


Figure 14: validation and training accuracy for ResNet50, for gender classification on Adience dataset

Upon evaluation of the model, we found that the performance was decent, with a test accuracy of 0.7688.

Link to runs: https://wandb.ai/todayisagreatday/NN_Project_Test_Runs/?workspace=user-todayisagreatday
Notebook: ‘3. Modelling/Model Evaluation/ResNet50.ipynb’

MobileNetV2

Although ResNet is a tremendous improvement over Hassner CNN, neural networks have come a long way since then. The models in the MobileNet family are supposed to be lightweight and fast (when performing inference), while not compromising too much on accuracy. We picked MobileNetV2 instead of MobileNetV3 for evaluation since its model (with pre-trained ImageNet weights) is available on Tensorflow.

MobileNetV2 has 3 features that allowed it to beat other state-of-the-art of its time.

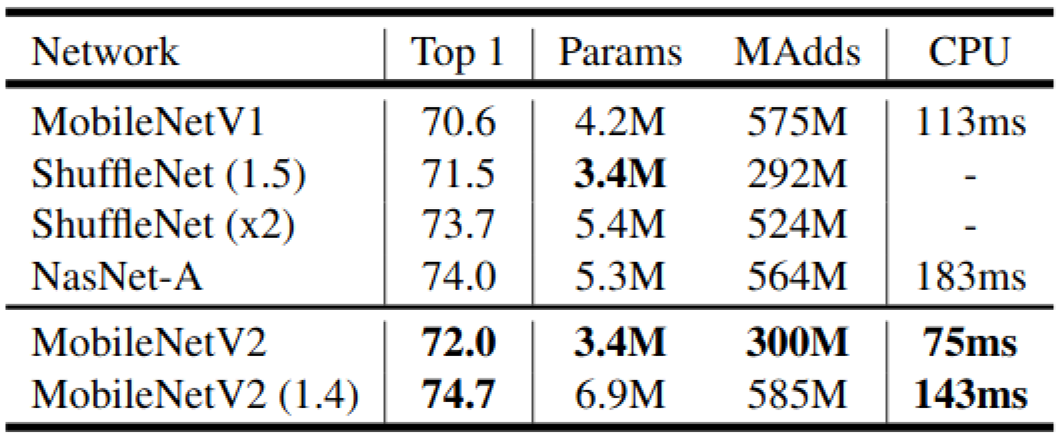
The first feature is depth-wise separable convolutions. Instead of using the standard convolution, MobileNetV2 splits the operation into 2 parts: a depthwise convolution, in which it applies a single convolution filter per input channel, followed by a pointwise convolution (1x1 convolution). Doing so allows it to perform convolutions at a lower computational cost while still working as well.

The second feature is linear bottlenecks. Due to the depth-wise separable convolutions, there is dimensionality reduction going on, which is more heavily affected by non-linearities like ReLU (since values less than 0 are lost). To deal with this problem, linear activation instead of non-linear activation is used at the narrow parts of the network.

The third feature is inverted residuals. These serve the same purpose as the residual connections in ResNet, in facilitating the backpropagation of gradients, while being more memory efficient than the traditional residual connection.

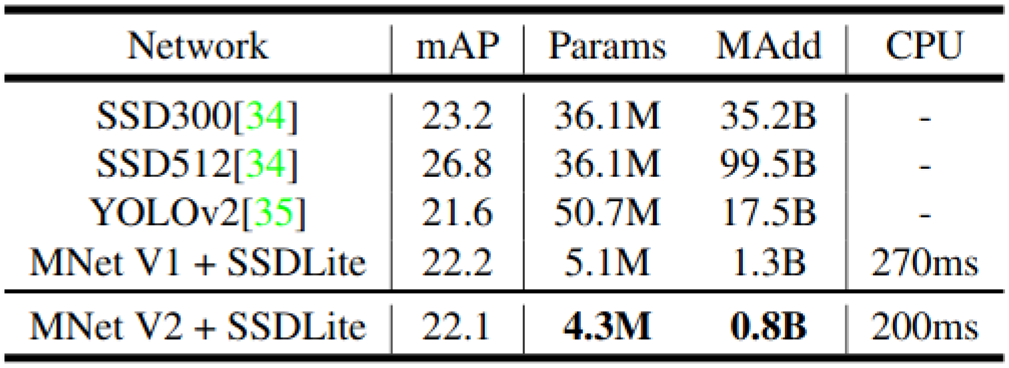
These 3 features are combined to form the bottleneck residual block, which is the mainstay of MobileNetV2. There are 19 of such layers before an input convolutional layer and the output convolutional layers in MobileNetV2's architecture.

MobileNetV2's design led to its impressive results in image classification, object detection and image segmentation, obtaining greater or equal results than its counterparts while occupying less space and requiring less time to run. [7]



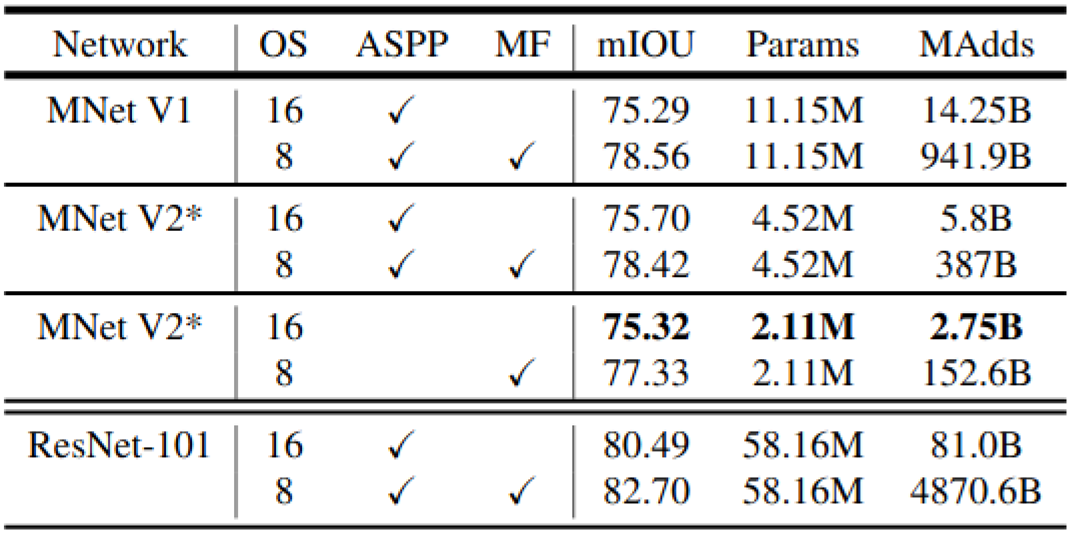
Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Figure 15: MobileNetV2's performance for image classification on ImageNet dataset



Network	mAP	Params	MAdd	CPU
SSD300[34]	23.2	36.1M	35.2B	-
SSD512[34]	26.8	36.1M	99.5B	-
YOLOv2[35]	21.6	50.7M	17.5B	-
MNet V1 + SSDLite	22.2	5.1M	1.3B	270ms
MNet V2 + SSDLite	22.1	**4.3M**	**0.8B**	200ms

Figure 16: MobileNetV2's performance for object detection on COCO dataset



Network	OS	ASPP	MF	mIOU	Params	MAdds
MNet V1	16	✓		75.29	11.15M	14.25B
	8	✓	✓	78.56	11.15M	941.9B
MNet V2*	16	✓		75.70	4.52M	5.8B
	8	✓	✓	78.42	4.52M	387B
MNet V2*	16			**75.32**	**2.11M**	**2.75B**
	8		✓	77.33	2.11M	152.6B
ResNet-101	16	✓		80.49	58.16M	81.0B
	8	✓	✓	82.70	58.16M	4870.6B

Figure 17: MobileNetV2's performance for image segmentation on Pascal VOC 2012 dataset

For our own evaluation of MobileNetV2, we found that it has 2,264,640 parameters, which is less than that of Hassner CNN and ResNet. In addition, it yielded a validation accuracy of 0.8852 on the Adience dataset without a prior tuning.

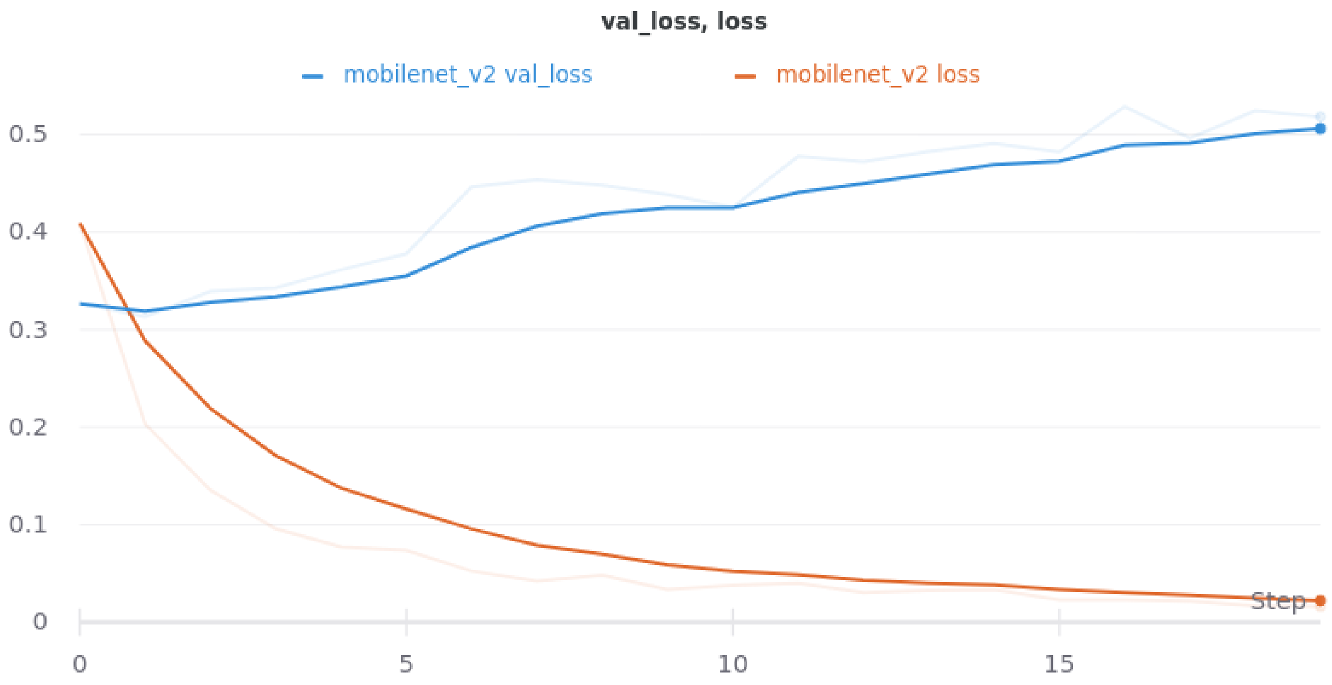


Figure 18: validation and training loss for MobileNetV2, for gender classification on Adience dataset

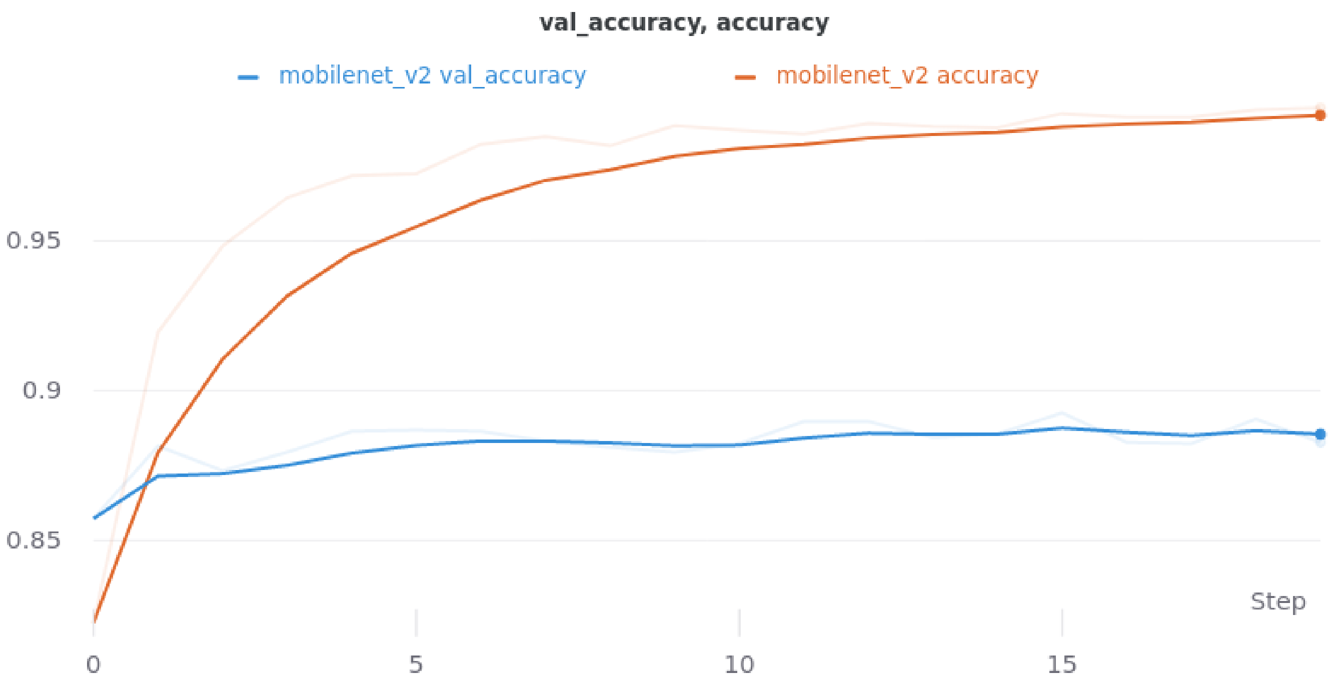


Figure 19: validation and training accuracy for MobileNetV2, for gender classification on Adience dataset

Link to runs: https://wandb.ai/burntice/cz4042/runs/3n5txt0r?workspace=user-burntice
Notebook: ‘3. Modelling/Model Evaluation/mobilenetv2.ipynb’
Saved weights: ‘5. Saved Weights/Model Evaluation/mobilenetv2_eval.h5’

Comparison Of Model Architectures

The Hassner CNN is eliminated as a viable architecture as it simply has too many parameters to train on. This greatly increases the time needed to train the network and would make the tuning process difficult.

This leaves the ResNet50 and the MobileNetV2 architecture for comparison. Aside from importing the base model from keras, the parameters used for the fully connected layers for both MobileNetV2 and ResNet50 are standardized to have the same values. The number of fully connected layers is increased from 2 to 3 and the number of neurons were changed to 1024>512>256 to reflect a progressive downsizing of the number of features.

These are the parameters used:

- Epochs: 20
- Learning Rate: 1e-3
- Batch Size: 64
- Optimizer: Adam
- Activation Function for hidden layers: ReLu
- Number of Neurons in Fully Connected Layer 1: 1024
- Number of Neurons in Fully Connected Layer 2: 512
- Number of Neurons in Fully Connected Layer 3: 256

Model Name	**Test Loss**	**Test Accuracy**	**Training Time**	**Test Time**
Hassner CNN	0.4080	0.8766	1h 41m 9s	971ms
ResNet50	0.6513	0.7688	1h 59m 11s	17s
MobileNetV2	0.5182	0.8827	49m 1s	707ms

The table above shows the comparison results for the evaluated model architectures. As the performance of MobileNetV2 is better in terms of accuracy and runtime (both during training and inference), we decided to use MobileNetV2 as our base model architecture.

Notebook: hassner_cnn.ipynb, ResNet50.ipynb, mobilenetv2.ipynb
Link to runs: https://wandb.ai/burntice/cz4042?workspace=user-

Data Augmentation

In data augmentation, our objective is to create more data from our existing dataset. This extra data will train the model for new exceptions. For example, our dataset may contain a few categories and one of the category might be taken by the same photographer. This photographer might tend to take photos of the object facing a specific direction for instance with the object facing the right. The model will then tend to associate photos with objects facing the right to be of that category. Data augmentation will change some of the image properties of that photo which will provide the model with augmented images. Below we shall explore some of the augmented techniques that we used for our image recognition model.

Sample image from dataset

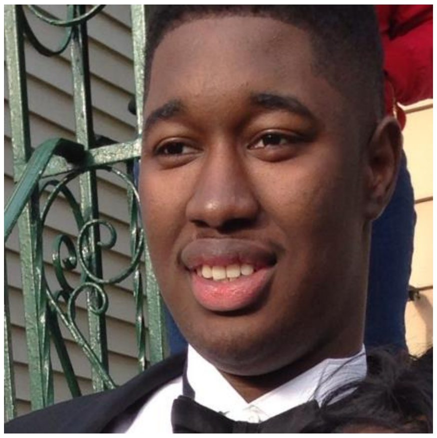


Figure 20: sample image from dataset

The above picture shows a sample from our training data. The dataset obtained is an aligned version of the original dataset. The author of this data has done some pre-processing to crop the images and align the faces to the middle of the frame. Therefore we will not be using shifting augmentation in our image preprocessing step.

Horizontal Flip

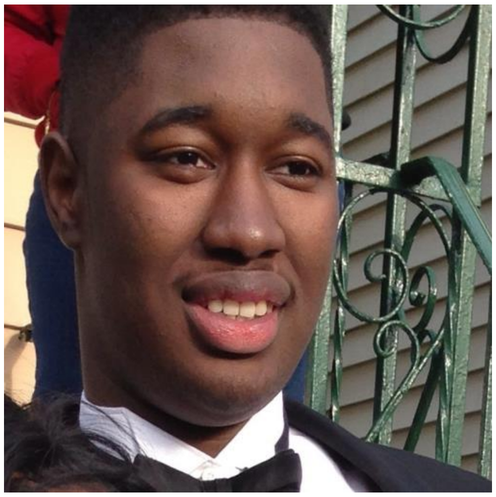


Figure 21: Flipping the sample image horizontally

In horizontal flipping, we changed the direction of the image in the horizontal direction. Comparing the above image with the one in the sample, we observed that the person is facing an opposite direction.

Image Rotation

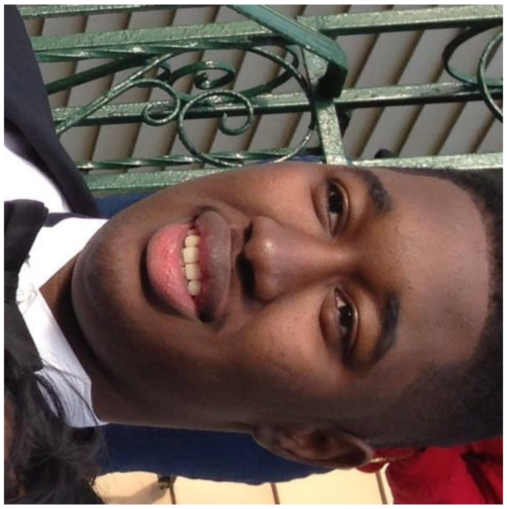


Figure 22: Rotating the image 90 degree

We will also be introducing rotation into our data augmentation. In the rotation, we will randomly sample images and rotate them between the angle of 0 to 90 degree in both the clockwise and anticlockwise direction.

Adjusting Brightness

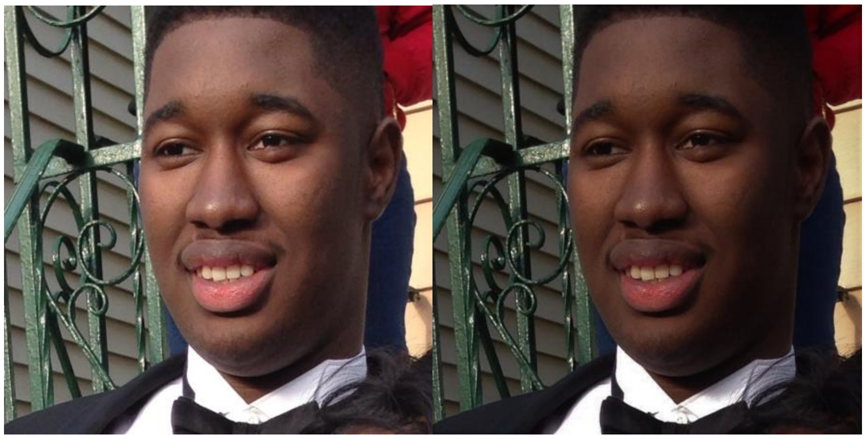


Figure 23: Decreasing the brightness of the image (Left - Original)

The last data augmentation is adjusting the brightness of certain images. A value of -0.5 to 1.5 with 1 being the original image brightness and a value greater than 1 making it brighter and less than 1 making it darker.

Results

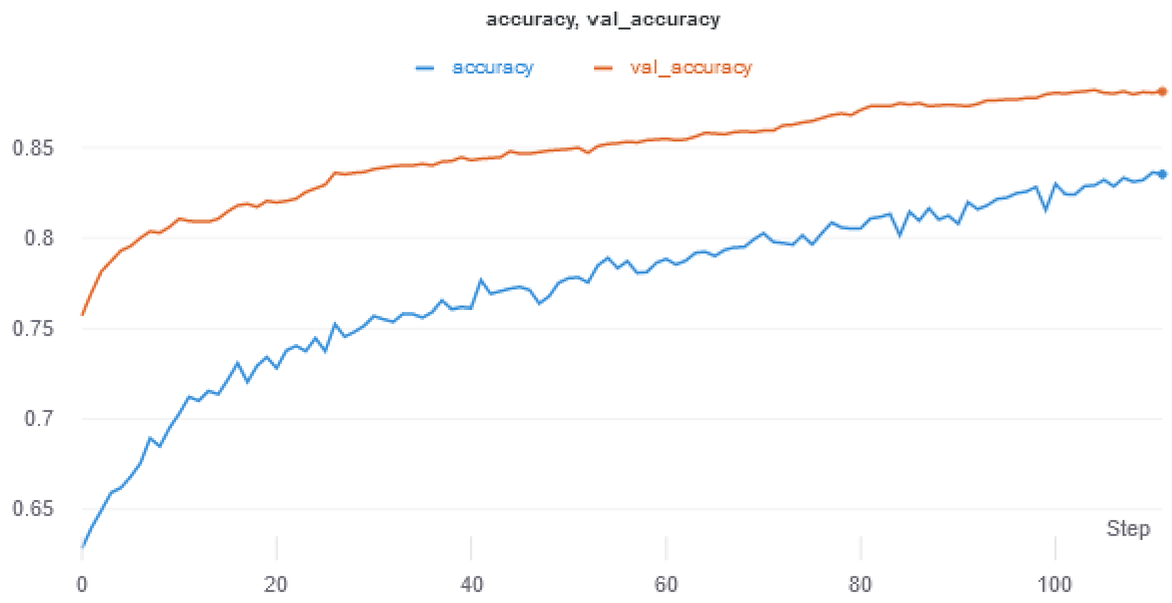


Figure 24: Data Augmentation Accuracy

After applying data augmentation, we achieved a max val accuracy of 0.88 after about 110 epochs. This is less than just training the dataset without augmentation that achieves an accuracy of over 0.9 accuracy. The reason for better accuracy without data augmentation could be due to preprocessing of the dataset by the author. While augmentation might not work as well for this project, it will tend to work better for images that have not been altered before.

Hyperparameters used:
- Epochs: 50
- Learning Rate: 1e-5
- Batch Size: 128
- Optimizer: Adam
- Activation Function for hidden layers: ReLu
- Number of Neurons in Fully Connected Layer 1: 1024
- Number of Neurons in Fully Connected Layer 2: 512
- Number of Neurons in Fully Connected Layer 3: 512

Comparison of with and without data augmentation

Data Augmentation	**Runtime (at 50 epochs)**	**Test Loss**	**Test Accuracy**	**No. of epochs**
Yes	5h 24m	0.3561	0.8491	50
No	4h 8m 25s	0.2764	0.9	50

Files:
3. Modelling/MNV2 Experiments -> mobilenetv2_celeba_v5.ipynb
5. Saved Weights/CelebA -> model_celeba_no_val.h5
6. Data Augmentation -> mobilenetv2_celeba_v5_data_aug.ipynb

Hyperparameter Tuning

For hyperparameter tuning, we split it up into 4 runs to reduce the number of combinations to search through. The assumption made here is that the hyperparameters to be tuned across different runs are independent of each other.

We searched for the optimal hyperparameters using grid search, with the help of the sweep function from Weights and Biases. For each combination of hyperparameters, the model is trained for 20 epochs.

Optimal Batch Size, Neurons and Learning Rate

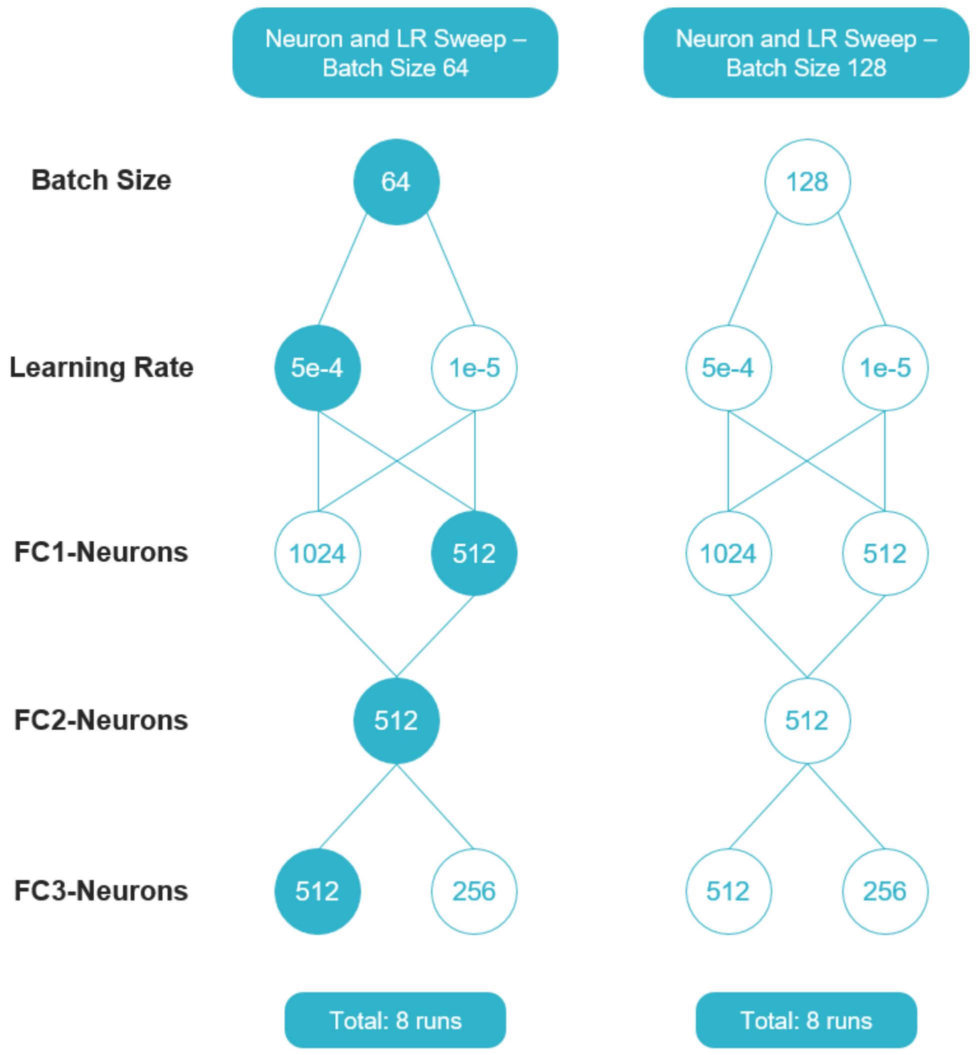


Figure 25: 2 Sweeps split by batch size (Optimal parameters highlighted)

For the first run, we performed a sweep for learning rate ∈ { 5e-4, 1e-5 }, number of neurons in FC1 ∈ { 1024, 512 } and number of neurons in FC3 ∈ { 512, 256 }, for a batch size of 128. The second run is similar to the first run, except that the batch size is set to 64. These two runs were done in parallel (on two Google Colaboratory notebooks) to speed up the tuning process. Note that the number of neurons in FC2 is set to 512 instead of being swept for. We reasoned that since the earlier layers in fully-connected layers tend to have more neurons, and the later layers tend to have less neurons, then the middle layers should have an in-between number of neurons (i.e. num_fc1_neurons ≥ num_fc2_neurons ≥ num_fc3_neurons).

Results of the first two runs (optimal parameters and results are underlined)

batch size	learning rate	FC1 neurons	FC3 neurons	val loss	val accuracy
128	5e-4	1024	512	1.314906	0.788313
128	5e-4	1024	256	0.723441	0.788313
128	5e-4	512	512	1.790023	0.776524
128	5e-4	512	256	2.294764	0.496155
128	1e-5	1024	512	18.312383	0.863659
128	1e-5	1024	256	17.236421	0.869810
128	1e-5	512	512	14.455925	0.867760
128	1e-5	512	256	13.259344	0.856996
64	5e-4	1024	512	1.132274	0.707842
64	5e-4	1024	256	1.190108	0.837006
64	5e-4	512	512	0.273368	0.926704
64	5e-4	512	256	0.231337	0.922091
64	1e-5	1024	512	15.417082	0.893900
64	1e-5	1024	256	14.634536	0.876473
64	1e-5	512	512	12.519720	0.882624
64	1e-5	512	256	11.563533	0.870835

Notebooks: '7. Hyperparameter Tuning/hyperparams_tuning_batchn.ipynb'

Link to sweep: https://wandb.ai/burntice/nn_project_tuning/sweeps/r4y8hd2a?workspace=user-
https://wandb.ai/burntice/nn_project_tuning/sweeps/30jubtgn?workspace=user-
https://wandb.ai/burntice/nn_project_tuning/sweeps/j0r78xbh?workspace=user-

Optimal Decay Rate

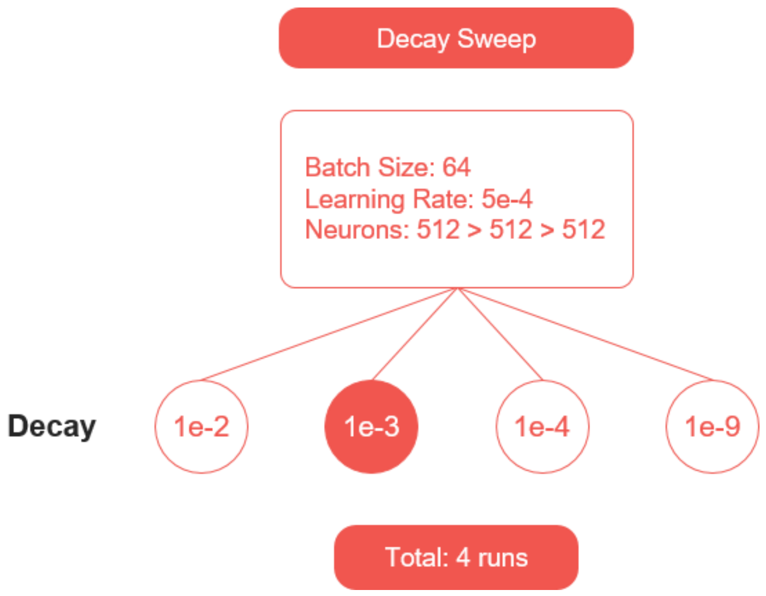


Figure 26: Decay Sweep (optimal parameter highlighted)

For the third run, the optimal hyperparameters based on the first two runs were used. In this run, we performed a sweep on weight decay ∈ { 1e-2, 1e-3, 1e-4, 1e-9 }.

Results of the third run (optimal parameters and results are underlined)

weight decay	**val loss**	**val accuracy**
1e-9	0.017376	0.892875
1e-4	0.763034	0.893388
1e-3	0.208280	0.942080
1e-2	0.439983	0.908764

Link to sweep: https://wandb.ai/burntice/nn_project_tuning/sweeps/rko63m53?workspace=user-
Notebooks: '7. Hyperparameter Tuning/hyperparams_tuning_L2.ipynb',
'7. Hyperparameter Tuning/hyperparams_tuning_L2 (Decay 1e-9).ipynb'

Optimal No. of Dropouts

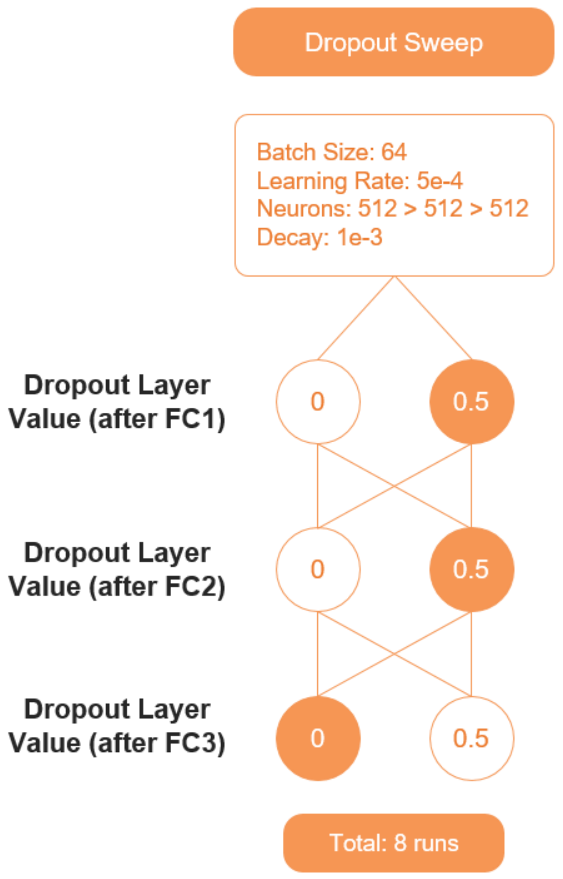


Figure 27: Dropout Sweep (optimal parameters highlighted)

For the fourth and last run, the optimal hyperparameters based on the first three runs were used. In this run, we performed a sweep on dropout rate for FC1 ∈ { 0, 0.5 }, dropout rate for FC2 ∈ { 0, 0.5 } and dropout rate for FC3 ∈ { 0, 0.5 }.

Results of the fourth run (optimal parameters and results are underlined)

dropout rate 1	**dropout rate 2**	**dropout rate 3**	**val loss**	**val accuracy**
0.5	0.5	0.5	0.772137	0.859559
0.5	0.5	0	0.255795	0.950281
0.5	0	0.5	2.447204	0.611993
0.5	0	0	1.893447	0.836494
0	0.5	0.5	0.924545	0.862634
0	0.5	0	0.462397	0.932342
0	0	0.5	0.533465	0.888262
0	0	0	0.496854	0.919528

Link to sweep: https://wandb.ai/burntice/nn_project_tuning/sweeps/001rfn8c?workspace=user-
Notebook: ‘7. Hyperparameter Tuning/hyperparams_tuning_dropout.ipynb’

Pre-training

This section will be divided into 2 parts. Before tuning and after tuning. Before tuning refers to experiments that were carried out without hyperparameter tuning. After tuning refers to experiments that were carried out after hyperparameter tuning of the model.

A pre-trained model is a model that was trained on a large benchmark dataset to solve a problem similar to the one that we are intending to solve. Pre-training a model that closely resembles the training data helps to greatly boost the accuracy of the model. In this specific example, we use the CelebA dataset as it is a large dataset of over 200,000 images and is similar to the Adience dataset.

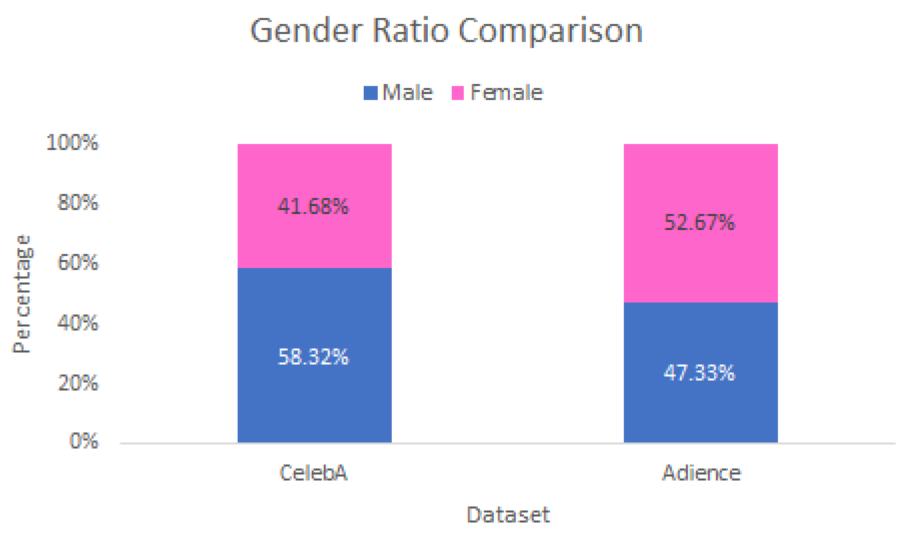


Figure 28: Class balance between Adience and CelebA

As we can observe from Figure 28, the classes are relatively similar to each other in terms of the gender ratio. This makes the CelebA dataset suitable for solving the task of gender classification like in Adience.

Experiments Before Tuning

Fixed Hyperparameters:

- Epochs: 20
- Batch Size: 128
- Optimizer: Adam
- Activation Function for hidden layers: ReLu
- Number of Neurons in Fully Connected Layer 1: 1024
- Number of Neurons in Fully Connected Layer 2: 512
- Number of Neurons in Fully Connected Layer 3: 512

Hyperparameters tested

- Learning Rate: 1e-3 or 3e-4
- Target Image Size: (112,112) or (224, 224)
- Train Test Split: 90:10 or 80:20
- Weights: None or ImageNet

Several experiments were carried out to determine the best way of carrying out pre-training.

Model	**Weights**	**Target Image size**	**CelebA Test Accuracy**	**Adience Test Accuracy**
CelebA Resized ZW	None	112 x 112	0.972	0.6052
CelebA Resized	Imagenet	112 x 112	0.9794	0.6371
CelebA	Imagenet	224 x 224	0.9669	0.7868
Baseline Adience Model	Imagenet	224 x 224	-	0.8827

3 models using the MobileNetV2 architecture were trained with 20 epochs and tested on the Adience dataset. The mobilenetV2 base model is frozen leaving only the fully connected layers to be trainable. Both datasets have a train test split ratio of 80:20. As we can observe from the table above, the test accuracy is very much lower as compared to the baseline model which has a test accuracy of 0.8827. Hence, we would need to perform fine-tuning on the model.

Link to runs: https://wandb.ai/todayisagreatday/CelebA%20Runs?workspace=user-
Files:
4. CelebA/Training -> celeb-a-pretrain_Vn.ipynb
3. Modelling/MNV2 Experiments -> mobilenetv2_celeba_vn.ipynb
Weights:
5. Saved Weights/CelebA -> model_celeba, model_celeba_resized.h5, model_celeba_resized_zerow.h5

Fine-Tuning

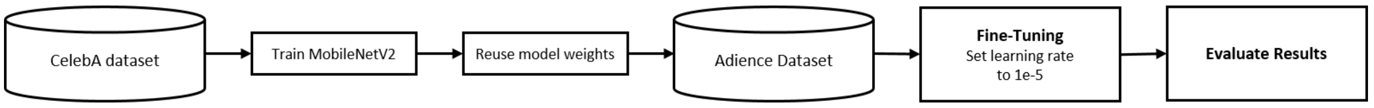


Figure 29: Workflow for fine-tuning

Following the keras documentation guide, the base model (MobileNetV2) was unfreezed and a low learning rate of 1e-5 was used. [8] This tremendously improved the performance of the model, increasing the test accuracy to 0.9147. The accuracy and loss curves over 50 epochs can be seen below.

Model	Learning rate	CelebA Test Accuracy	Adience Test Accuracy
CelebA	1e-3	0.9669	0.7868
CelebA-LR	1e-5	0.9669	0.9147

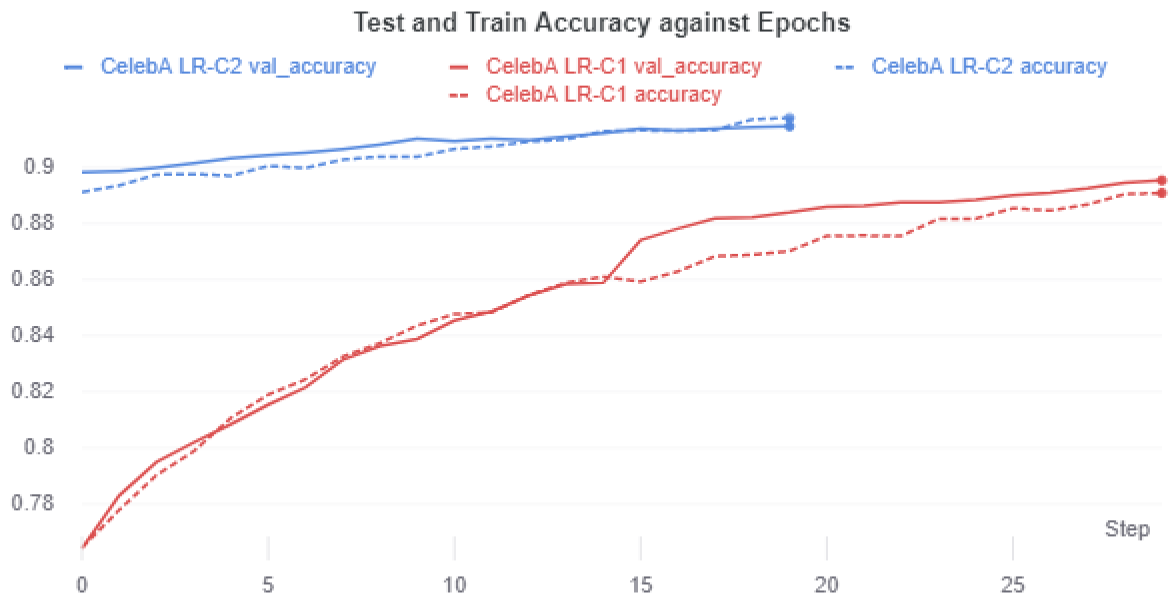


Figure 30: Test and Train Accuracy against epochs (blue line is a continuation of red line)

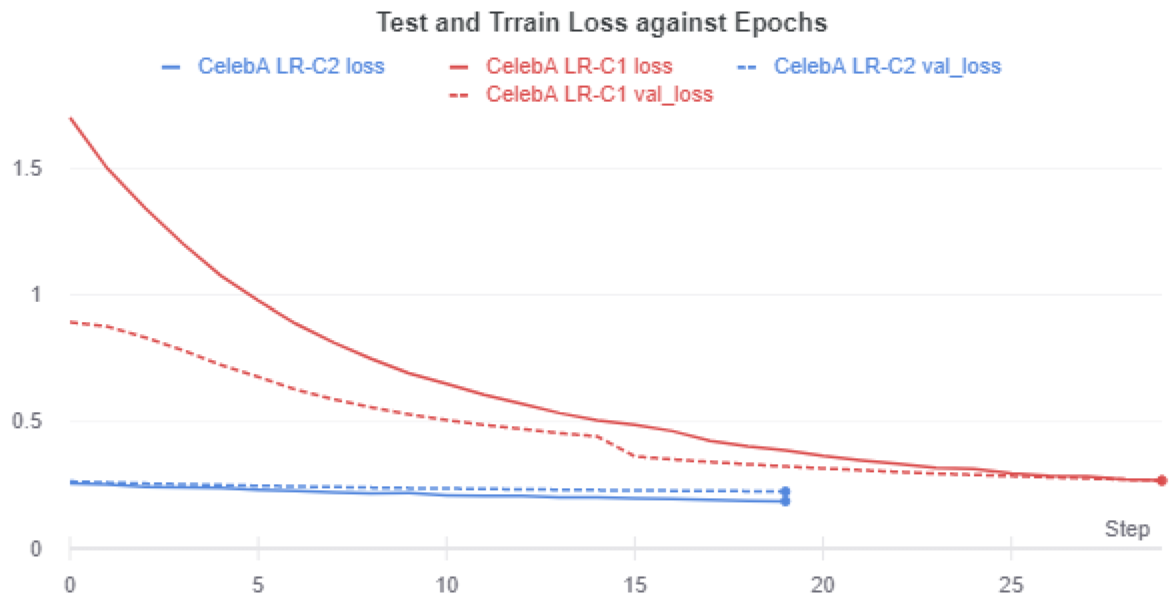


Figure 31: Test and Train Loss against epochs (blue line is a continuation of red line)

Using the Weights and Biases resume function, we are able to let the model run for an additional 20 more epochs after the cell has stopped. This explains why the accuracy and loss curve is split into 2 (red first then blue). As we can observe from Figure 30 and 31, there is minimal overfitting as the test and train curves are very close to each other. While 91.47% is a great score, the model has not been tuned yet.

Hyperparameters for Adience Testing:

- Epochs: 50
- Batch Size: 128
- Optimizer: Adam
- Learning Rate: 1e-5
- Target Image Size: (224, 224)
- Train Test Split: 80:20
- Weights: CelebA pretrained
- Activation Function for hidden layers: ReLu
- Number of Neurons in Fully Connected Layer 1: 1024
- Number of Neurons in Fully Connected Layer 2: 512
- Number of Neurons in Fully Connected Layer 3: 512

Link to runs: https://wandb.ai/todayisagreatday/transfer_learning_mnv2?workspace=user-todayisagreatday

Notebooks:
3. Modeling/MNV2 Experiments -> mobilenetv2_celeba_vn.ipynb
Weights:
5. Saved Weights/CelebA -> model_celeba, model_celeba_resized.h5, model_celeba_resized_zerow.h5

Experiments After Tuning

Without pretraining

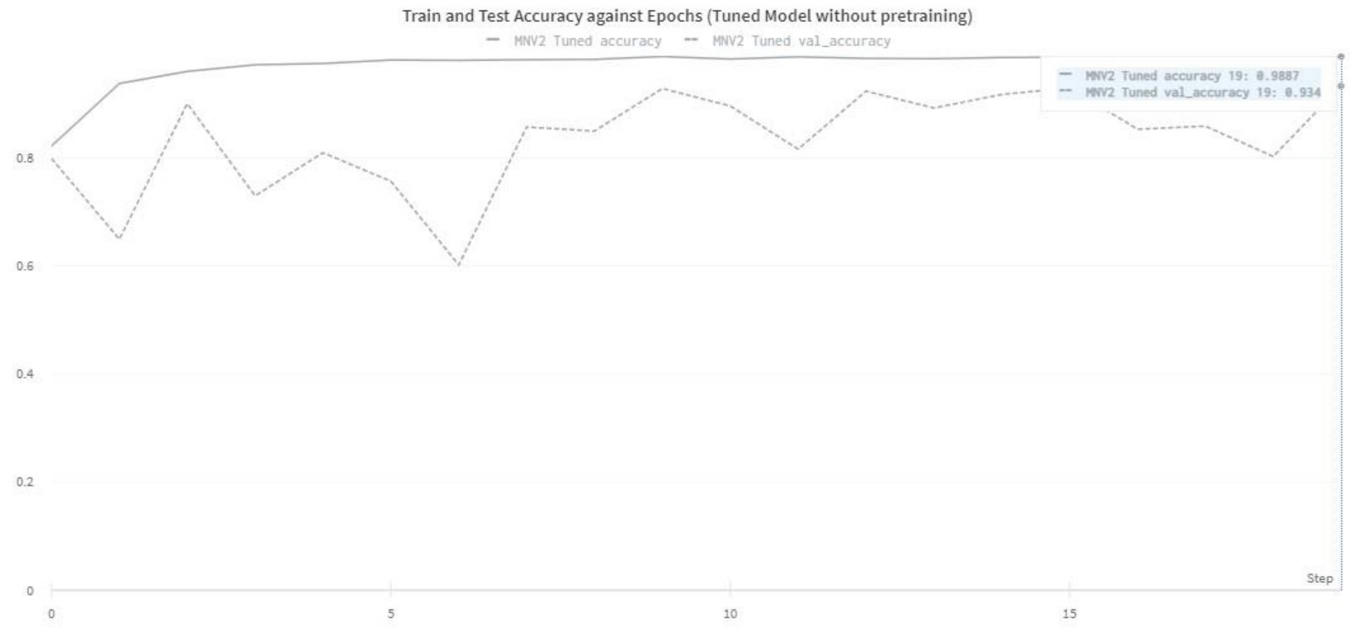


Figure 32: Train and test accuracy against epochs (tuned model without pre training)

Using the optimal parameters we obtained from hyperparameter tuning, we ran the model for 20 epochs and achieved a score of 0.934. Despite the great result, the instability in the model indicates that the learning rate is too high.

Hyperparameters for Adience Testing:

- Epochs: 20
- Batch Size: 64
- Optimizer: Adam
- Learning Rate: 5e-4
- Target Image Size: (224, 224)
- Train Test Split: 80:20
- Weights: CelebA pretrained
- Activation Function for hidden layers: ReLu
- Number of Neurons in Fully Connected Layer 1: 512
- Number of Neurons in Fully Connected Layer 2: 512
- Number of Neurons in Fully Connected Layer 3: 512

After Pretraining

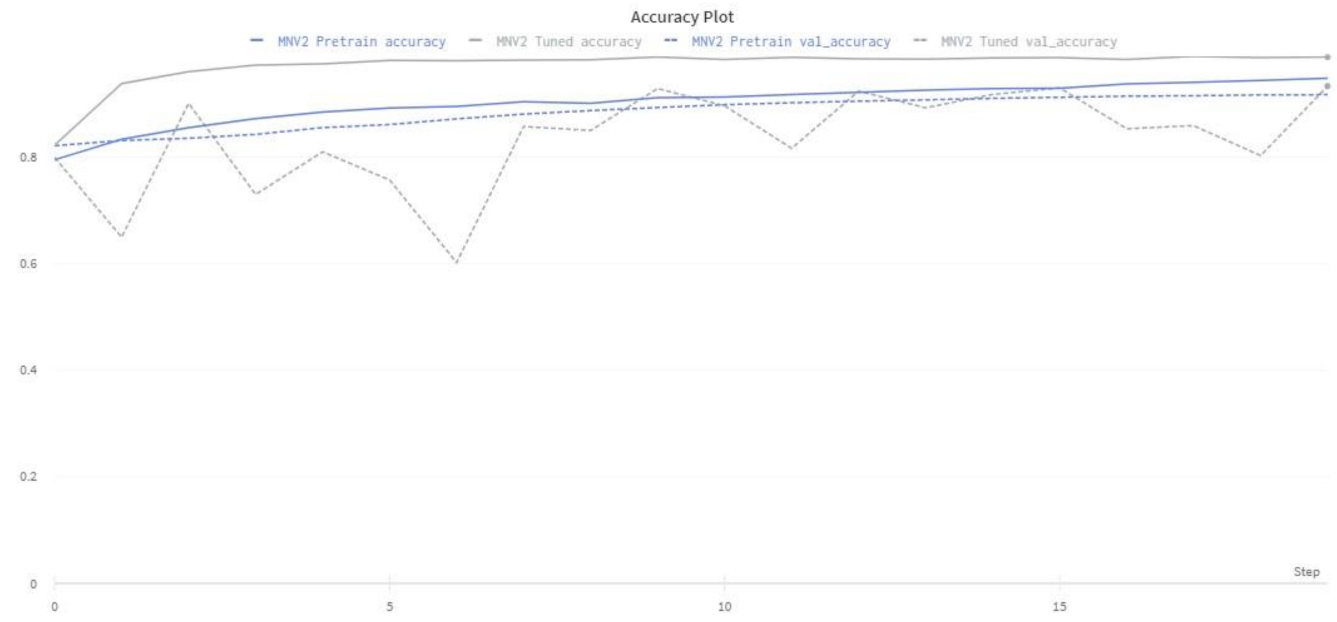


Figure 33: Comparison between CelebA and imagenet weights

MNV2 Pretrain (blue) represents the model with CelebA pretrained weights while MNV2 Tuned (grey) represents the model with Imagenet weights. By lowering the learning rate to 1e-5 and adding the pretrained CelebA weights, we are able to achieve a more consistent result as the number of epochs increases.

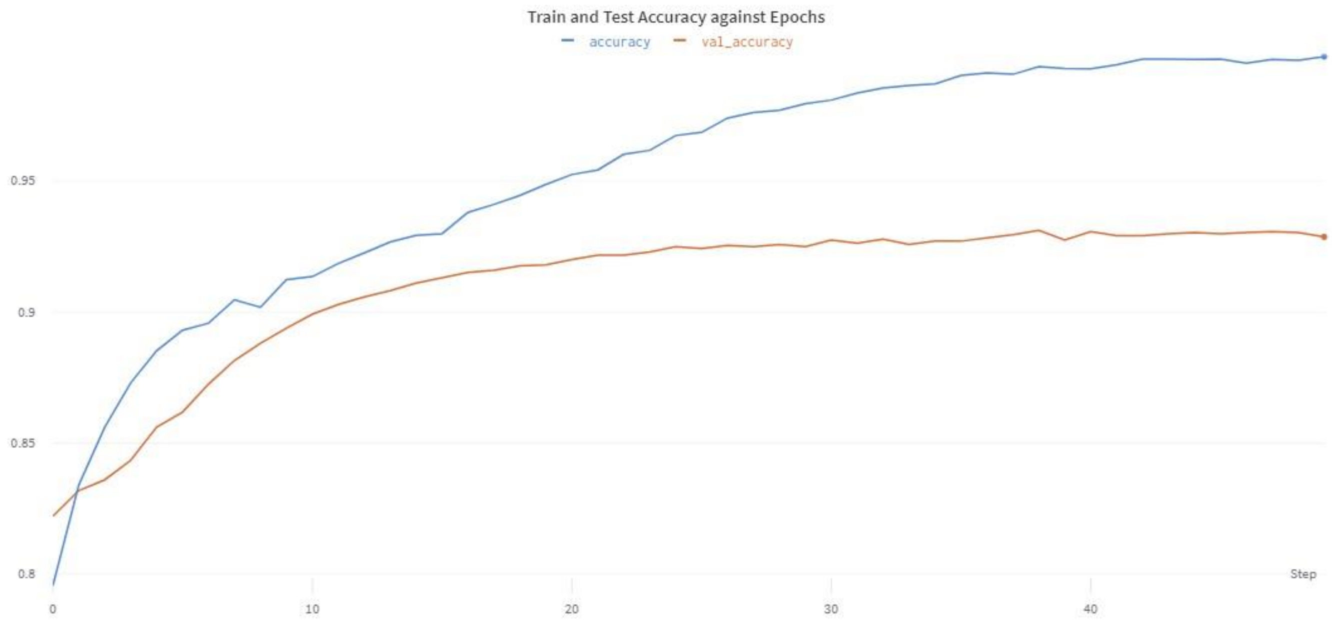


Figure 34: Train and Test accuracy against epochs after pre training and tuning

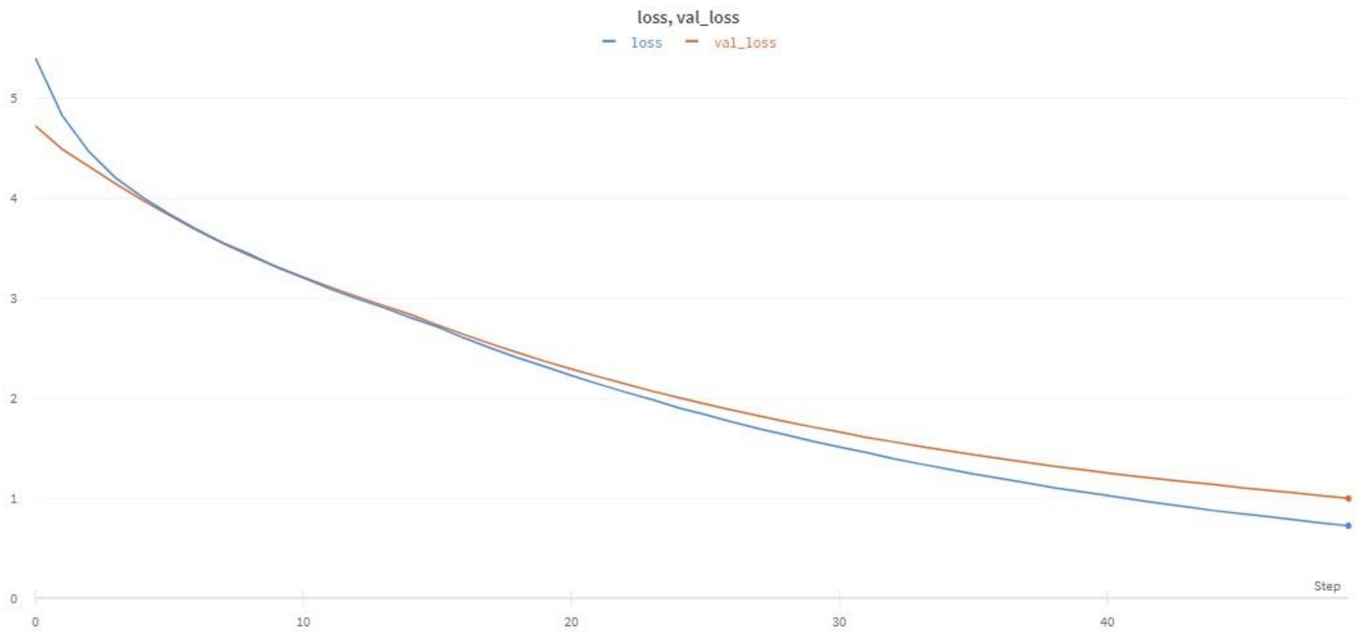


Figure 35: Train and Test loss against epochs after pre training and tuning

After running the model (MNV2 Pretrain) for a total of 50 epochs, we are able to reach a test accuracy of 0.9287. The model converges at a faster rate when comparing the model without tuning and the model with tuned parameters. There is also overfitting present as the number of epochs increases. The overall comparison will be evaluated in the next section.

Hyperparameters for CelebA Pretrain:

- Epochs: 20
- Batch Size: 128
- Optimizer: Adam
- Learning Rate: 3e-4
- Target Image Size: (224, 224)
- Train Test Split: 90:10
- Weights: Imagenet
- Activation Function for hidden layers: ReLu
- Number of Neurons in Fully Connected Layer 1: 512
- Number of Neurons in Fully Connected Layer 2: 512
- Number of Neurons in Fully Connected Layer 3: 512

Hyperparameters for Adience Testing:

- Epochs: 50
- Batch Size: 64
- Optimizer: Adam
- Learning Rate: 1e-5
- Target Image Size: (224, 224)
- Train Test Split: 80:20
- Weights: CelebA pretrained
- Activation Function for hidden layers: ReLu
- Number of Neurons in Fully Connected Layer 1: 512

- Number of Neurons in Fully Connected Layer 2: 512
- Number of Neurons in Fully Connected Layer 3: 512

Link to runs: https://wandb.ai/burntice/nn_project_tuning?workspace=

Files:
3. Modelling/MNV2 Experiments -> mobilenetv2_celeba_tune.ipynb, mobilenetv2_tuned.ipynb
4. CelebA/Training -> celeb-a-pretrain_tune.ipynb

Weights:
5. Saved Weights/CelebA -> model_celeba_tune.h5

Evaluation of Results

Model	**Test Accuracy**	**Test Loss**	**Epochs**
Baseline MNV2	0.8827	0.5182	20
MNV2 Pretrained	0.9147	0.1865	50
MNV2 Tuned	0.934	0.06276	50
MNV2 Tuned + Pretrained	0.9287	0.7237	50

The above table shows the improvement of results from the baseline model to the final tuned and pretrained model. After tuning and pre training, the test accuracy has increased by 4.6%.

Future Considerations

To further improve and validate the results would require additional testing. Below are a few ways that can possibly improve the model's performance

Learning Rate Scheduler

As the model is observed to be overfitting, by setting the learning rate to decrease over time, this would allow the model to speed up the training process at the beginning and obtain optimal weights as the number of epochs increases.

Cross Validation

In Hassner's research paper, he conducted a 5-fold cross validation to ensure the validity of the results. Likewise, cross validation could be performed using the MobileNetV2 model to validate the results obtained.

Conclusion

In this project, we performed gender classification of images using the Adience and CelebA dataset. The dataset directory text files were first preprocessed into dataframes. Next, we limited our selection of models to 3 different architectures (Hassner CNN, ResNet50, MobileNetV2). After testing the model architectures, we selected MobileNetV2 as our base model to use for classification due to its great accuracy and moderate speed. Following the selection of MobileNetV2, we experimented with augmenting the data, tuning the model as well as pre training the model with CelebA dataset. This gave us an overall test accuracy of 0.9287.

Appendix

CelebA Pre Training Results

https://wandb.ai/todayisagreatday/CelebA%20Runs?workspace=user-todayisagreatday

Modelling Results

Before Pre Training
Model Comparison (ResNet, MobileNet V2 and Hassner CNN)
https://wandb.ai/burntice/cz4042

ResNet (Initial Testing)
https://wandb.ai/todayisagreatday/NN_Project_Test_Runs/

After Pretraining
MobileNetV2
https://wandb.ai/todayisagreatday/transfer_learning_mnv2?workspace=user-todayisagreatday

Hyperparameter Tuning Results

https://wandb.ai/burntice/nn_project_tuning?workspace=user-burntice

Glossary

Term	**Description**
MNV2	MobileNetV2
FC1/2/3	Stands for the 1st, 2nd and 3rd **F**ully **C**onnected layers in the model after the flatten layer
Val loss	Validation loss
Val accuracy	Validation accuracy
CNN	Convolutional neural network
ReLU	Rectified linear unit
Training time	Time taken for a model to train on the training set
Test time	Time taken for a model to evaluate the test set, for 1 step
wandb	Weights and Biases. A developer tool for recording and visualizing ML experiments
Hyperparameter Sweep	Hyperparameter sweeps (unique to wandb) provide an organized and efficient way to conduct a battle royale of models and pick the most accurate model. They enable this by automatically searching through combinations of hyperparameter values (e.g. learning rate, batch size, number of hidden layers, optimizer type) to find the most optimal values.

Bibliography

[1] E. Eidinger, R. Enbar and T. Hassner, "Age and Gender Estimation of Unfiltered Faces," in IEEE Transactions on Information Forensics and Security, vol. 9, no. 12, pp. 2170-2179, Dec. 2014, doi: 10.1109/TIFS.2014.2359646.

[2] T. Hassner, S. Harel, E. Paz and R. Enbar. Effective Face Frontalization in Unconstrained Images. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Boston, 2015.

[3] T. Hassner, "Face Image Project - Data", Talhassner.github.io, 2014. [Online]. Available: https://talhassner.github.io/home/projects/Adience/Adience-data.html#agegender. [Accessed: 22- Nov- 2020].

[4] G. Levi and T. Hassner, Age and Gender Classification Using Convolutional Neural Networks. IEEE Workshop on Analysis and Modeling of Faces and Gestures (AMFG), at the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Boston, 2015.

[5] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", arXiv.org, 2015. [Online]. Available: https://arxiv.org/abs/1512.03385. [Accessed: 22- Nov- 2020].

[6] A. Karapetov, "Age and Gender Classification using Deep Neural Networks", Medium, 2018. [Online]. Available: https://medium.com/@andreykar_79244/age-and-gender-classification-using-deep-neural-networks-a8ded298a838. [Accessed: 22- Nov- 2020].

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks", arXiv.org, 2018. [Online]. Available: https://arxiv.org/abs/1801.04381. [Accessed: 23- Nov- 2020].

[8] F. Chollet, "Keras documentation: Transfer learning & fine-tuning", Keras.io, 2020. [Online]. Available: https://keras.io/guides/transfer_learning/. [Accessed: 22- Nov- 2020].