UNIVERISTY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc «Informatics»

ΠΜΣ «Πληροφορική»

MSc Thesis

Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	**Predicting Facial Beauty with Convolutional Neural Networks** Πρόβλεψη Ελκυστικότητας Προσώπου με Χρήση Συνελικτικών Νευρωνικών Δικτύων
Student's name-surname: Ονοματεπώνυμο φοιτητή:	**Konstantinos Bozikis** Κωνσταντίνος Μποζίκης
Father's name: Πατρώνυμο:	**Marinos** Μαρίνος
Student's ID No: Αριθμός Μητρώου:	ΜΠΠΛ18051
Supervisor: Επιβλέπων:	**Dionisios Sotiropoulos, Assistant Professor** Διονύσιος Σωτηρόπουλος, Επίκουρος Καθηγητής

May 2022/ Μάιος 2022

3-Member Examination Committee

Τριμελής Εξεταστική Επιτροπή

Dionisios Sotiropoulos	**Georgios Tsihrintzis**	**Efthimios Alepis**
Assistant Professor	**Professor**	**Associate Professor**
Διονύσιος Σωτηρόπουλος	Γεώργιος Τσιχριντζής	Ευθύμιος Αλέπης
Επίκουρος Καθηγητής	Καθηγητής	Αναπληρωτής Καθηγητής

Περίληψη

Καθώς η επιστήμη της Μηχανικής Εκμάθησης εξελίσσεται, η ανάγκη για ανάπτυξη μοντέλων που μιμούνται την ανθρώπινη κρίση αυξάνεται. Ένα κομμάτι επάνω στο οποίο εστιάζουν πολυάριθμες έρευνες είναι η ανάλυση οπτικών σημάτων με στόχο την εξαγωγή συμπερασμάτων σύμφωνων με την ανθρώπινη λογική. Στην παρούσα διατριβή αναλύεται η πρόβλεψη της ελκυστικότητας ενός προσώπου με τη χρήση Συνελικτικών Νευρωνικών Δικτύων και εξετάζεται η επιρροή που μπορεί να έχουν μεροληπτικά δεδομένα κατά τη διαδικασία της εκπαίδευσης. Για το σκοπό αυτό, αναπτύχθηκαν πολλαπλά μοντέλα με διαφοροποιήσεις στα δεδομένα εκπαίδευσης και αναλύθηκαν τα αποτελέσματα με γνώμονα τις διαφορές αυτές. Σύμφωνα με τα αποτελέσματα της μελέτης, γίνεται φανερό ότι η εκπαίδευση μοντέλων με χρήση μεροληπτικών δεδομένων δύναται να αποφέρει μη αξιόπιστα αποτελέσματα και να οδηγήσει σε εσφαλμένα συμπεράσματα.

Abstract

As the science of Machine Learning evolves, the need to develop models that mimic the human crisis increases. One area on which many researchers focus is the analysis of optical signals with the aim of drawing conclusions in accordance with human logic, with the most important tool for this analysis being Convolutional Neural Networks. The subject of this dissertation is the prediction of facial beauty using Convolutional Neural Networks and the impact that discriminatory training data may have on the extracted predictions. For this purpose, multiple models were developed with differences in the training data and the results were analyzed based on the introduced bias in each case. According to the results of the study, it becomes clear that model training using biased data can yield unreliable results and lead to erroneous conclusions.

Table of Contents

1. INTRODUCTION

Facial attractiveness has been a subject of many studies over the years. Although previous studies indicate that characteristics like facial averageness and symmetry are defining factors that influence the facial attractiveness perception of a human, a universal definition of beauty remains a mystery. This is more prominent when comparing data across countries and ethnicities. Different cultures usually have different perception of attractiveness and value different characteristics. Also, each person's definition of beauty is different and can be influenced by numerous factors such as personal feelings, clothing, hairstyle, and social status.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning applications are the key technology behind many recent services such as natural language recognition in virtual assistants, self-driving cars etc. One of the most popular deep neural networks are Convolutional Neural Networks that are commonly used for analyzing visual data and can be used to build attractiveness prediction models. Using Convolutional Neural Networks for image recognition has been proved a challenge as it requires a large amount of data in order to create a model that can generate reliable and consistent results. Building an effective facial beauty recognition model presents two main challenges: the amount of data and data quality.

The Transfer Learning method can be used to overcome the first challenge. The basic premise of transfer learning is simple: take a model trained on a large dataset and transfer its knowledge to a smaller dataset. This method provides a base model that can extract all the useful characteristics needed for beauty recognition and can be trained on a smaller dataset in order to be able to predict the desired characteristic. In this thesis we applied the transfer learning technique to the VGG-Face model developed by the Visual Geometry Group of the University of Oxford (Parkhi et al. [1]). This model was created using a dataset of 2.6 million face images of 2,622 individuals.

Data quality is a defining factor in machine learning systems and biased data is a common problem. Bias as a general term defines the tendency to lean towards a certain direction either in favor of or against the given topic, person or thing. When it comes to data science, a biased dataset can be classified as one that doesn't represent a model's use case fully and therefore produces results that are skewed or inaccurate. There are many examples where applications utilizing deep learning were characterized as racially biased due to faults in the process of data collection and labeling. On numerous platforms, AI-based facial recognition systems have problems recognizing women and people of color as accurately as they can identify white men. Much of this problem is due to the lack of diverse training data the models were trained on. A study from Joy Buolamwini [2], a researcher at the MIT Media Lab, showed that nearly 35% of black women were misidentified by prominent facial recognition systems.

Especially for facial beauty, which is a subjective attribute, data needs to be diverse and unbiased. In this thesis, the dataset selected is the Chicago Face Database provided by the University of Chicago which contains a diverse sample of standardized photographs of male and female faces of varying ethnicity. Several convolutional neural networks were implemented using different subsets of the main set as training data, based on race. Using the full dataset as control, the role of racial bias in data collection and how it affects final predictions was investigated.

1.1 Previous research

Facial beauty has been a topic of debate for centuries. In medieval times, renaissance painters used unique ratios named "The Golden Ratios" to represent through paintings what the perfectly shaped human face would look like [3]. The Golden Ratios are ratios based on the value of 1.6, which was considered by the Greeks to be a perfect number. This golden ratio was applied to facial beauty, where different facial ratios were calculated and compared against the value of 1.6.

Over the last years, facial beauty prediction has become an increasingly popular research area due to the many potential applications that it can benefit, such as aesthetic surgery planning, cosmetic recommendation, and others. Rhazi et al. [4] proposed a method to predict facial beauty based on golden ratios calculated from the extracted feature corners. Schmid et al. [5] have proposed a model to calculate facial beauty based on golden ratios, symmetry, and neoclassical canons. The neoclassical canons are ratios used by medieval painters in their paintings to represent their understanding of human beauty. Thornhill et al. [6] proved that there is a strong correlation between average faces and facial beauty, but the most attractive faces are usually not average. Facial symmetry, however, does increase with facial averageness, as stated in Grammer et al. [7].

Recently, machine learning approaches have been developed to assess facial attractiveness with facial shape features and landmarks considered as decisive factors. Kagian et al. [8] analyzed facial beauty depending on the shape and facial geometry. Various facial features that describe facial geometry, color and texture, combined with an average human attractiveness score for each facial image were used to train various prediction models. Facial attractiveness ratings produced by the final model were found to be highly correlated with the respective human ratings. Wei et al. [9] assessed facial symmetry and attractiveness based on Support Vector Machines (SVM) and linear regression using a predefined dataset. Hong Y-J et al. [10] presented a novel framework for automatically assessing facial attractiveness based on facial landmark extraction. Lin et al. [11] used an attribute-aware CNN to predict the facial beauty using the SCUT-FPB5500 dataset [12] and utilizing powerful GPU systems for training. Q. Xiao et al. [13] developed Beauty3DFaceNet, which makes use of a CNN to predict the attractiveness of 3D faces. The research team created a 3D facial attractiveness dataset called ShadowFace3D, which contains information such as the point clouds, texture images, and texture mappings of 3D faces as well as their public aesthetic criteria. Although the approach is promising, the nature of the source data is limiting and furthermore the training is quite resource demanding. Zhai, Yikui et al. [14], based on previous research on fine-grained image classification using the multiscale architecture, implemented a model for unconstrained facial beauty prediction. In order to mitigate the overfitting phenomenon which is caused by the scarcity of labeled facial beauty samples, the transfer learning strategy was used, improving the networks performance. Finaly, Cao, Kerang et al. [15] presented a novel network design for facial beauty prediction aiming to better performance using a residual-in-residual (RIR) structure for the prediction model. Final results showed that this kind of network was able to predict attractiveness closer to a human’s assessment than previous attempts.

2. DEEP CONVOLUTIONAL NEURAL NETWORKS

2.1 Neural Networks

Artificial neural networks are information-processing systems whose structure and functionality are reminiscent of the nervous system and especially the brain of humans and animals. A neural network often consists of a large number of simple units working in parallel, the so-called neurons. These neurons send each other information, in the form of activation signals, via weighted connections. A neural network is arranged in several layers: the input layer, the output layer and the "hidden layers" in between.

From the input layer, each node propagates its value to all hidden layer nodes connected to it. In order to learn from the data, each node connection receives a weight and a bias for the evaluation and interpretation of the information. These are the flexible adjusting components in a network that make learning possible in the first place. Input values that are important for correct classification are strengthened and unimportant ones are attenuated. These weighted values are then summed and passed through an activation function. Activation functions simulate the biological neurons by deciding whether a signal is passed on and at what level. In the human brain, this happens by "overloading" the neuron with electrical energy. The activation functions ultimately decide whether the input features are significant enough to be passed on. The same procedure then takes place between the hidden layer and the output layer.

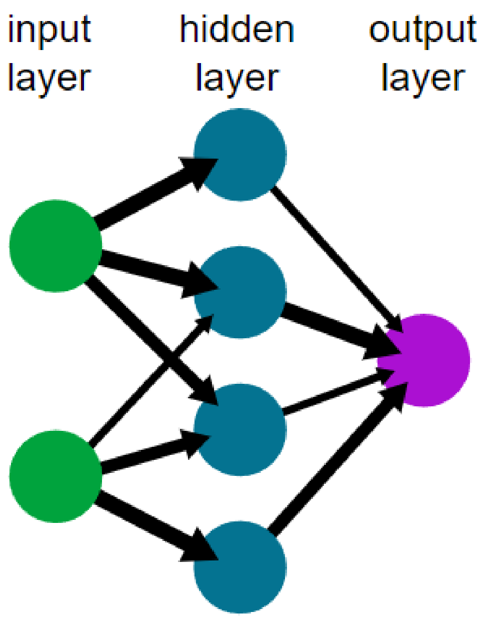


Image 1: Depiction of a basic neural network

2.2 Basic network principles

2.2.1 Weights and structure

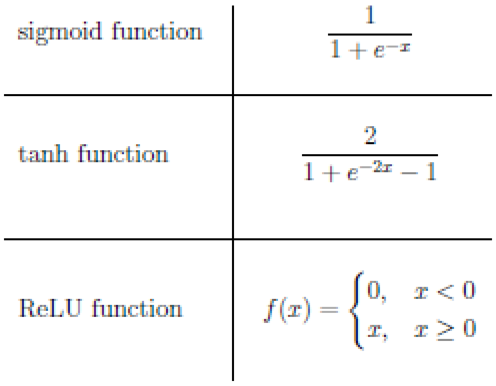
The connections between the units in a neural network are weighted, meaning that the weight indicates how much the input from a previous unit affects the output of the next unit. To mathematically compute an artificial neuron, all the products of all the inputs and their corresponding weights are added, then a bias is added to that sum and finally the resulting value is fed into an activation function to form the output.

2.2.2 Biases

A bias is an extra input to a neuron and it is technically the number 1 multiplied by a weight. The bias makes it possible to move the activation function curve left or right on the coordinate graph, enabling the neuron to create the required output value.

2.2.3 Activation functions and the ReLu

By definition, an activation function decides if a neuron should be activated ("fired") or not. It introduces non-linearity to the output of a neuron. A neural network without activation functions would be just a linear regression model. The most common activation function for a CNN the ReLu. It outputs x when x is positive or zero and outputs 0 otherwise (Figure 5).



sigmoid function	$\frac{1}{1+e^{-x}}$
tanh function	$\frac{2}{1+e^{-2x}-1}$
ReLU function	$f(x)=\begin{cases}0, & x<0\\ x, & x\geq 0\end{cases}$

Figure 1: Popular activation functions

ReLu is less computationally expensive than some other common activation functions like tanh and Sigmoid because the mathematical operation is simpler and the activation is sparser. Since the function outputs 0 when $x \leq 0$, there is a considerable chance that a given unit does not activate at all. Sparsity also means more concise models with more predictive power and less noise or overfitting. In a sparse network, neurons are more likely to process meaningful information. For example, a neuron which can identify human faces should not be activated if the image is actually about a dog. One more advantage, which the ReLu possesses over the others, is that it converges faster. Linearity (when $x \geq 0$) means that the slope of the line does not plateau when x increases. Therefore, ReLu does not have the vanishing gradient problem suffered by some other activation functions, such as Sigmoid or tanh. Another common activation function used in CNNs is the Softmax function. It is often used in the output layer, where a multiclass classification happens.

2.2.4 Backpropagation & Gradient descent

Backpropagation is an algorithm which, based on prediction errors, helps neural networks to learn their parameters. This algorithm works by calculating the gradient of the loss function, which points us in the direction of the value that minimizes the loss function. A loss function is an error metric, a way to calculate the inaccuracy of predictions. The aim of deep learning models is to minimize this loss function value, and the process of minimizing the loss function value is called optimization. Using gradient descent, we can iteratively move closer to the minimum value by taking small steps in the direction given by the gradient. Gradient descent is an optimization algorithm which modifies the internal weights of the neural network to minimize the loss function value. After each iteration, the gradient descent algorithm attempts to decrease the loss function value by tweaking weights, until it reaches the point where further tweaks produce little or no change to the loss function value, also called convergence. In conclusion, backpropagation and gradient descent are two different methods that form a powerful combination in the learning process of neural networks.

2.2.5 Learning rate – optimizers

A learning rate is the step size of each iteration in the gradient descent or other optimization algorithms. If the learning rate is too small, convergence will take a long time to happen, but if the learning rate is too large, there might be no convergence at all. There are many optimization algorithms (optimizer), one of which is the Adam, which computes individual learning rates for different parameters. Adam is the optimization algorithm used in this thesis.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (ConvNet/CNN) are one of the most popular and effective learning algorithms for image analysis. A ConvNet is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other, extract localized features from input images and unfold these image fields using filters. First, the CNN recognizes simple structures such as lines, splashes of color or edges in the first levels. As the levels progress, the convolutional neural network learns combinations of these structures such as simple shapes or curves. With each level, more complex structures can be identified. The data is repeatedly resampled and filtered in the levels. In the last step, the results are assigned to the classes or objects to be recognized.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the human brain and was inspired by the organization of the visual cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area. CNN is usually constructed by three types of layers: convolution, pooling and fully connected layers. Convolutional and pooling layers are used for feature extraction and fully connected layers perform the final mapping of extracted features into the final output.

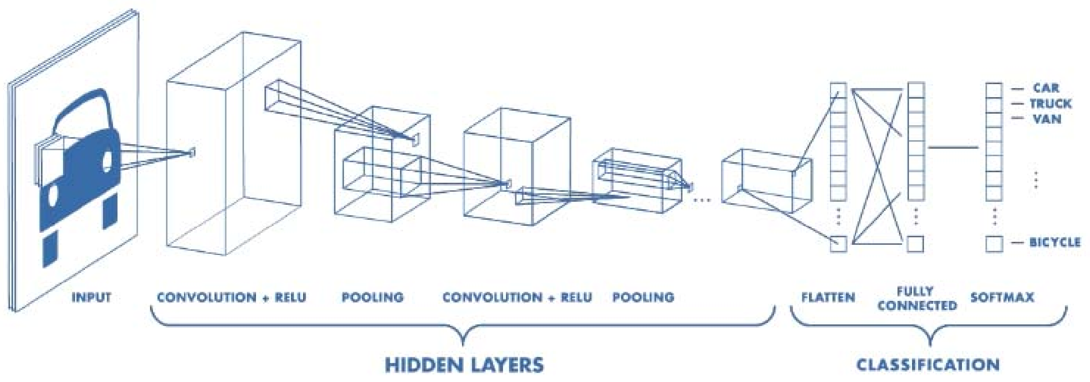


Image 2: Convolutional Neural Network architecture

2.3.1 Convolutional Layer

Convolutional layers are the most important building blocks of any CNN. A convolutional layer is able to recognize and extract individual features in the input data. In image processing, these can be features such as lines, edges, or specific shapes. In this layer, the image is analyzed by various filters that have a certain pixel size and gradually scan the graphic for its properties. This process can be compared with a small magnifying glass that scans the image from left to right and top to bottom. The filter records the results of this scanning process in a matrix. This result matrix is now also scanned through a smaller filter - in the same way as the original image was scanned. The results of this new scanning process are also recorded in matrices. This process is repeated several times at lower and lower levels, allowing the convolutional layer to analyze the original input in high detail.

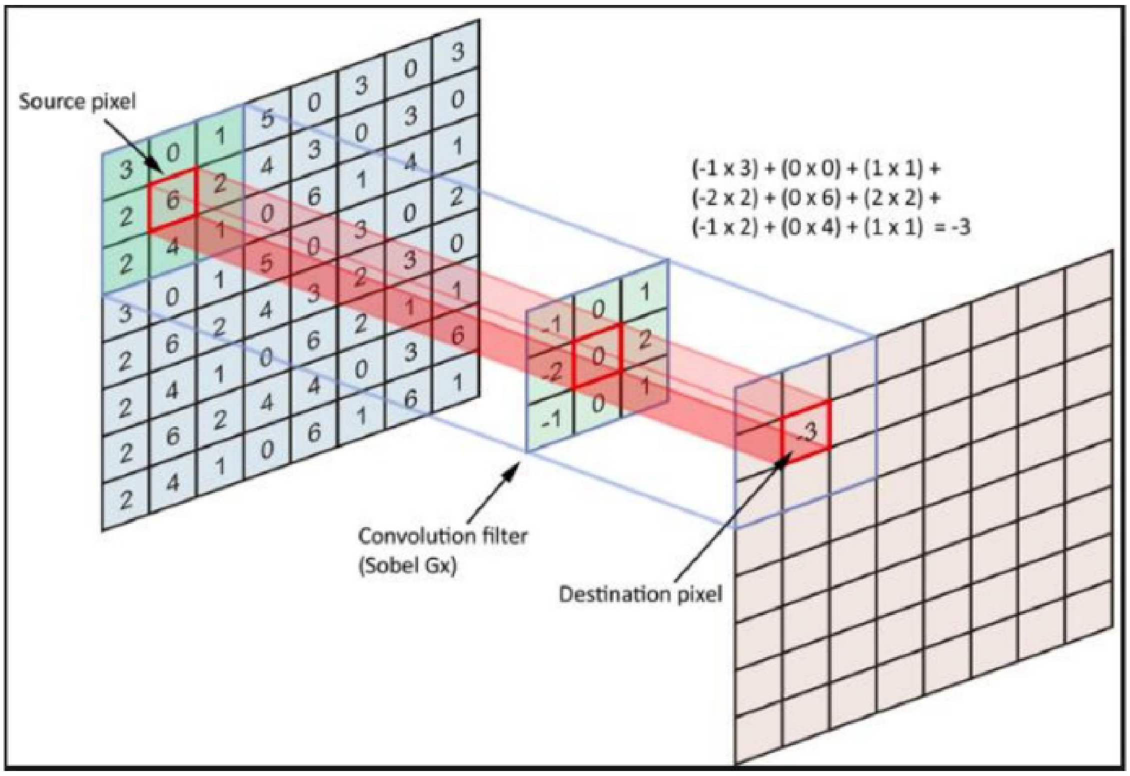

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Image 3: Convolutional layer

In Image 3, an element wise multiplication is made between a 3x3 sized filter matrix and a 3x3 sized area of the input image’s matrix. The elements of the resulting matrix are summed, and the sum is the output value (“destination pixel”) on the feature map. The filter then slides over the input matrix, repeats the multiplication with every remaining combination of 3x3 sized areas and completes the feature map. Multiple filters are used for one input and the resulting feature maps are joined together for the final output of one convolutional layer.

There are two other important concepts in convolutional layers: strides and padding. Strides are the number of pixels a kernel or a filter moves over the input matrix. Padding is what is used when the filter does not fit the input matrix. There are two types of padding: valid padding, when the border pixels of the input matrix are discarded, and zero or “same” padding, when zeros are added to the borders so that the filter fits the input matrix.

2.3.2 Pooling Layers

Pooling layers are responsible for reducing the dimensionality of feature maps, specifically the height and width, preserving the depth. By doing so, the required processing power is decreased, while at the same time the dominant features in feature maps are extracted. There are two types of pooling layers: max pooling and average pooling. Max pooling outputs the maximum value of the elements in the portion of the image covered by the filter (Image 4), while average pooling returns the average value. Max pooling is usually the preferred method as it is better at extracting dominant features and is generally performs better.

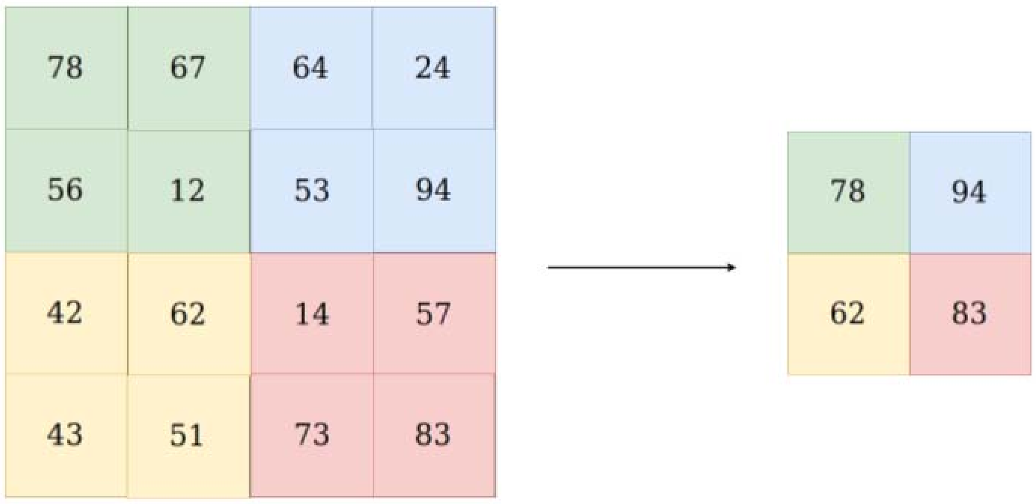


Image 4: Max Pooling layer

2.3.3 Fully Connected Layers

Fully connected layers are where classification actually happens. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer. Each fully connected layer (called Dense layer) is passed through an activation function (e.g. tanh or ReLu), but the output Dense layer is passed through Softmax. In the Softmax multiclass classification, the loss function used is Cross Entropy. The output of the Softmax function is an N-dimensional vector, where N is the number of classes the CNN has to choose from. Each number in this N-dimensional vector represents the probability that the image belongs to each certain class.

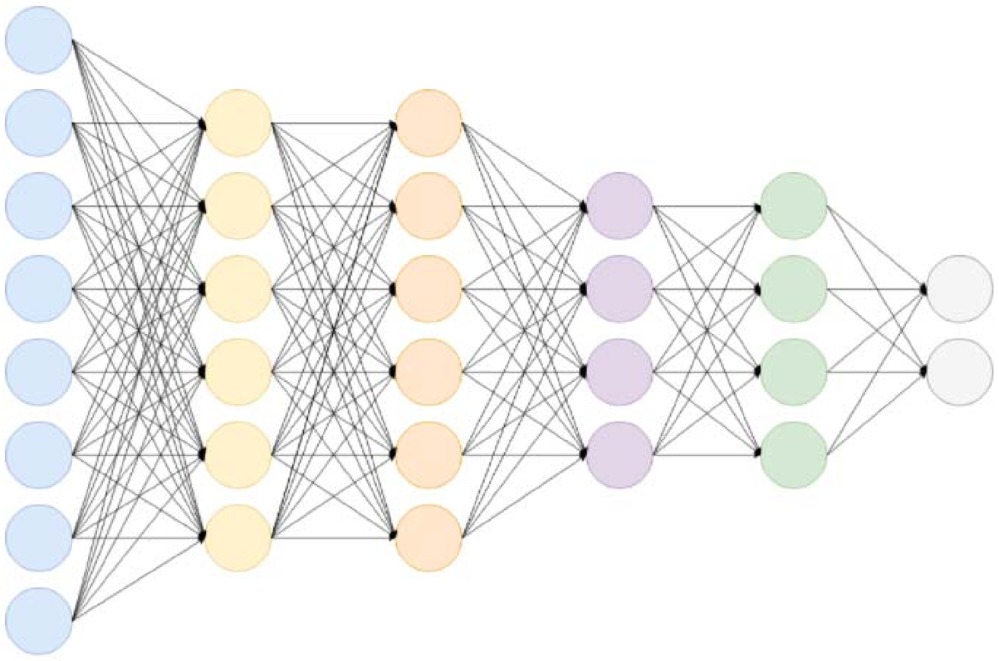


Image 5: Fully Connected layers

2.4 VGG-Face model

The VGG-Face model was used as a base for implementing the models for this thesis. It is a CNN implemented by Omkar Parkhi in the 2015 paper titled "Deep Face Recognition". It is a modified implementation of VGG-16, trained especially for face feature extraction.

The model's architecture consists of clusters of convolutional layers and max pooling layers. The activation function used after each convolution is ReLu. It has 22 layers and 37 deep units. In Table 1 all the layers are listed in order with shape details and number of trainable parameters for each.

Layer (type)	Output Shape	Number of parameters
conv2d_input (InputLayer)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 224, 224, 64)	1792
conv2d_1 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_2 (Conv2D)	(None, 112, 112, 128)	73856
conv2d_3 (Conv2D)	(None, 112, 112, 128)	147584
max_pooling2d_1 (MaxPooling 2D)	(None, 56, 56, 128)	0
conv2d_4 (Conv2D)	(None, 56, 56, 256)	295168
conv2d_5 (Conv2D)	(None, 56, 56, 256)	590080
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590080
max_pooling2d_2 (MaxPooling 2D)	(None, 28, 28, 256)	0
conv2d_7 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_8 (Conv2D)	(None, 28, 28, 512)	2359808
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2359808
max_pooling2d_3 (MaxPooling 2D)	(None, 14, 14, 512)	0

conv2d_10 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2359808
max_pooling2d_4 (MaxPooling 2D)	(None, 7, 7, 512)	0
conv2d_13 (Conv2D)	(None, 1, 1, 4096)	102764544
dropout (Dropout)	(None, 1, 1, 4096)	0
conv2d_14 (Conv2D)	(None, 1, 1, 4096)	16781312
dropout_1 (Dropout)	(None, 1, 1, 4096)	0
conv2d_15 (Conv2D)	(None, 1, 1, 2622)	10742334
flatten (Flatten)	(None, 2622)	0
activation (Activation: 'Softmax')	(None, 2622)	0

Table 1: VGG-Face layer architecture

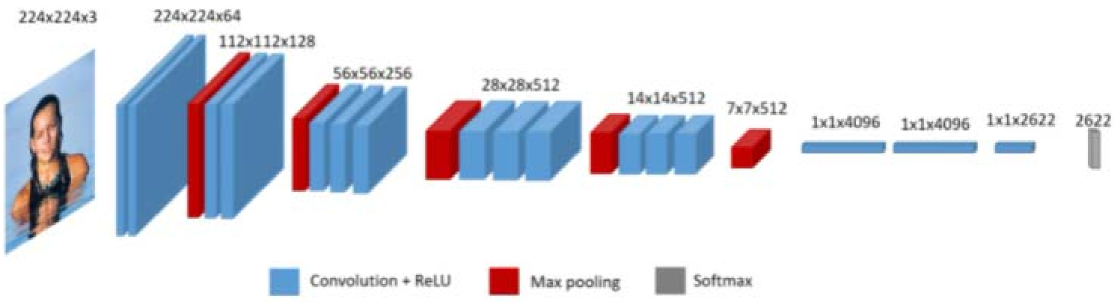


Image 6: Visualization of VGG-Face

3. DATASET DESCRIPTION

In this thesis, the data used for training the CNN is a subset of the Chicago Face Database that was developed at the University of Chicago by Debbie S. Ma, Joshua Correll, and Bernd Wittenbrink. It provides high-resolution, standardized photographs of male and female faces of varying ethnicity between the ages of 17-65. Extensive norming data are available for each individual model. These data include both physical attributes (e.g., face size) as well as subjective ratings by independent judges (e.g., attractiveness).

The database consists of three datasets: CFD, CFD-MR and CFD-INDIA. The CFD dataset consists of images of 597 unique individuals. They include self-identified Asian, Black, Latino, and White female and male models, recruited in the United States. All models are represented with neutral facial expressions. A subset of the models is also available with happy (open mouth), happy (closed mouth), angry, and fearful expressions (Image 7). The CFD-MR extension set includes images of 88 unique individuals, who self-reported multiracial ancestry. The CFD-INDIA extension set includes images of 142 unique individuals, recruited in Delhi, India.

For this thesis, the CFD dataset was used in order to have a more diversified sample. Subjective rating norms such as attractiveness, which will be the focus of this study, are based on a U.S. rater sample. The question presented to each rater was to rate each person's attractiveness with respect to other people of the same race and gender. Ratings were recorded based on the Likert scale, which is a standard classification format. The range is 1-7: 1 = Not at all, 4 = Neutral and 7 = Extremely. A mean attractiveness value was then calculated for each model.

Image 7: Images with different expressions

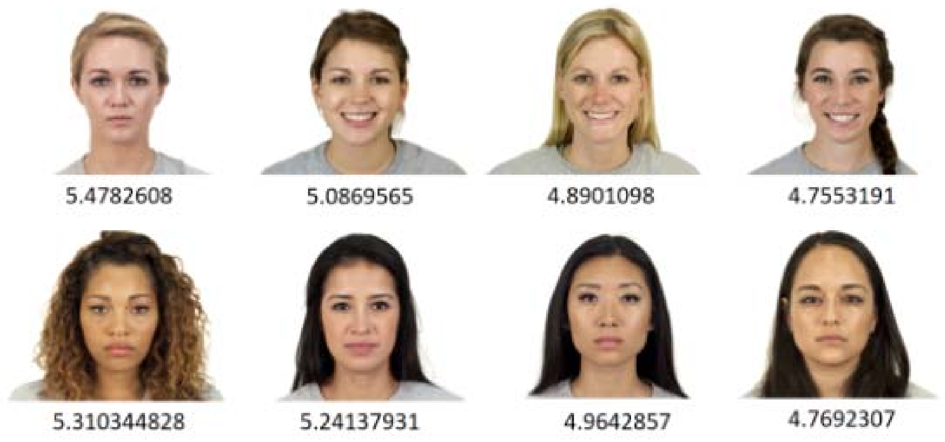


Image 8: Attractiveness score examples

4. EXPERIMENTATION SETUP

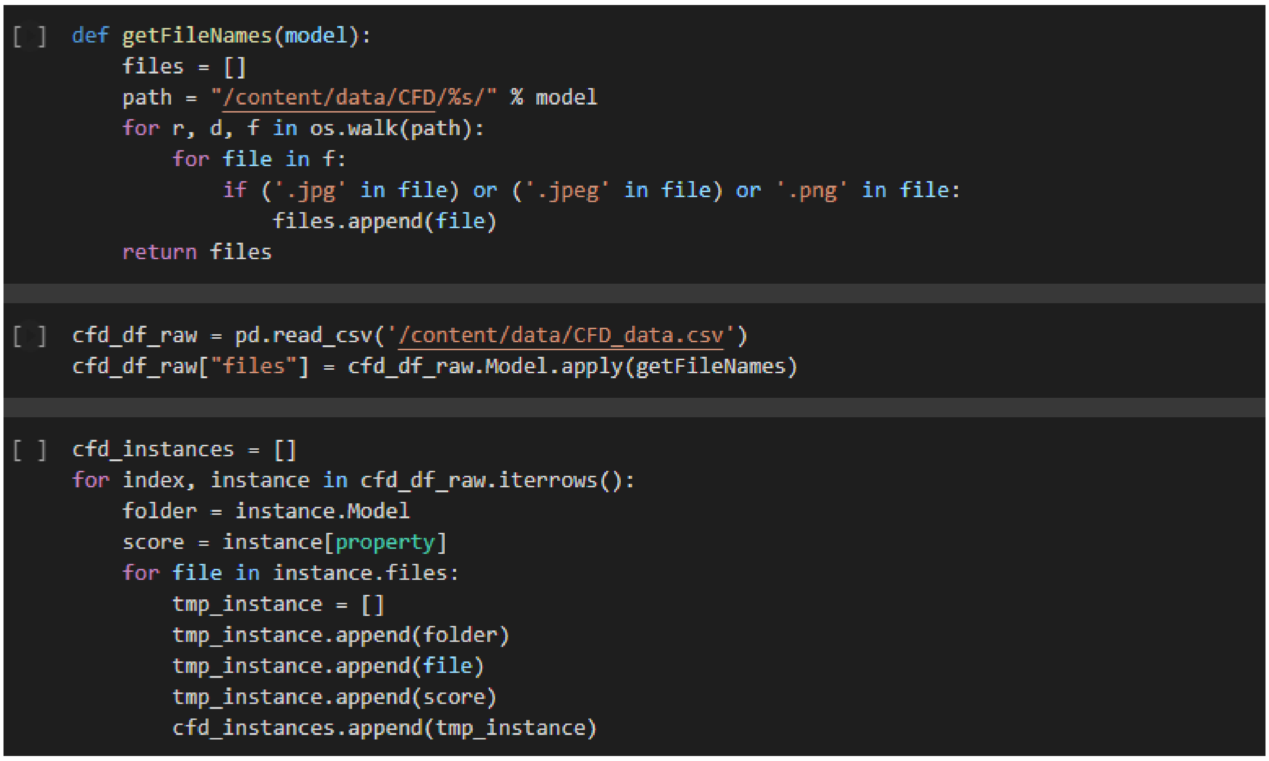
4.1 Tool description

Google Colab was used as the development environment for this experiment. Google Colab provides hosted runtimes for Python development with free access to GPUs and TPUs. This was an important feature as training a model on image data can be very resource demanding and the process could not be executed on available local machines. A hosted Tensor Processing Unit (TPU) runtime was used for the training process.

The CNN model was developed with Keras, an open-source library built on top of TensorFlow 2 designed to provide a high-level API towards Tensorflow to simplify and speed-up the development of deep learning models. Tensorflow is an end-to-end open-source platform for machine learning. It is a symbolic math library that uses dataflow and differentiable programming to perform various tasks focused on training and inference of deep neural networks.

4.2 Data pre-process

The dataset contains 1207 images of 597 unique individuals categorized by gender and race (White, Black, Latino, Asian). The first step of the dataset processing was to read the norming data provided and match each entry with the respective image file. All relevant data were then gathered in a dataframe. The properties included in the final dataframe were folder, filename, attractiveness score, emotion, race, gender. As mentioned previously, for some models there are images with different emotions. For the purposes of this experiment, all expressions except neutral and happy (closed mouth) were excluded from the training data. The final dataset consisted of 750 images.

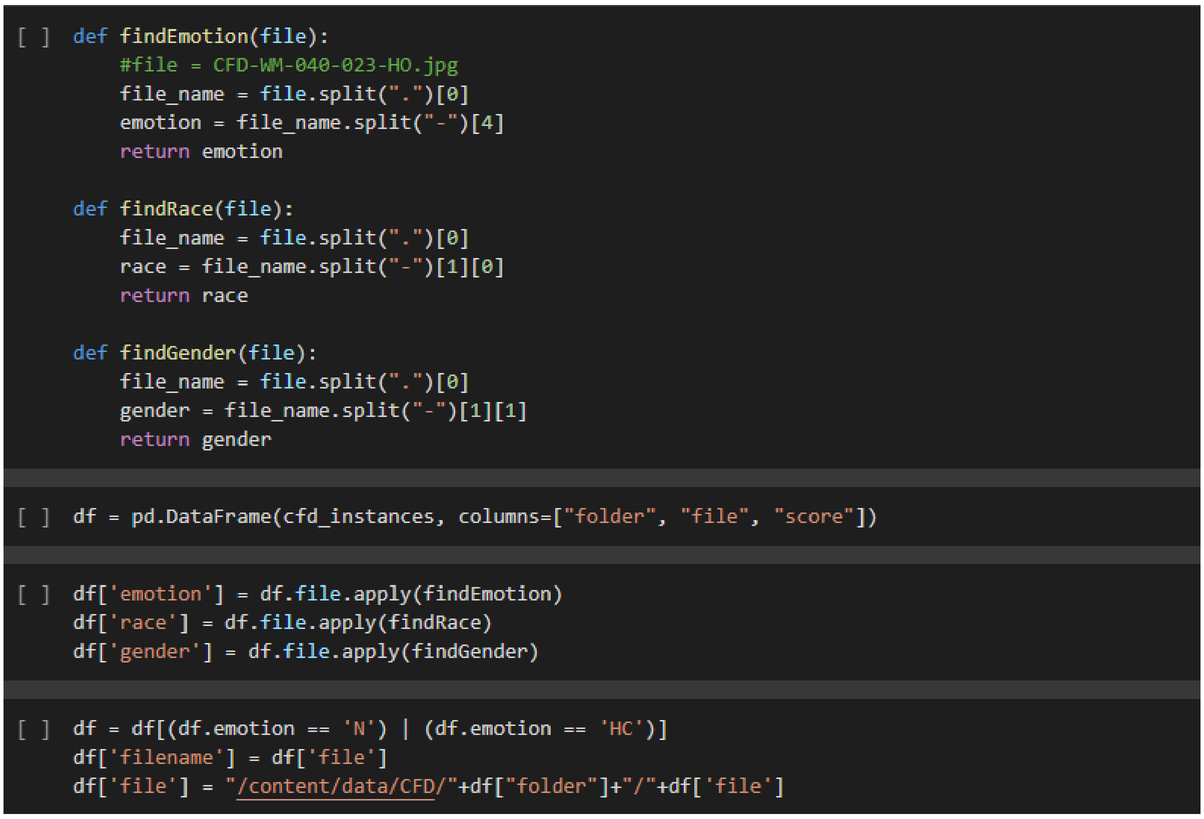


```
[ ] def getFileNames(model):
        files = []
        path = "/content/data/CFD/%s/" % model
        for r, d, f in os.walk(path):
            for file in f:
                if ('.jpg' in file) or ('.jpeg' in file) or '.png' in file:
                    files.append(file)
        return files
```

```
[ ] cfd_df_raw = pd.read_csv('/content/data/CFD_data.csv')
    cfd_df_raw["files"] = cfd_df_raw.Model.apply(getFileNames)
```

```
[ ] cfd_instances = []
    for index, instance in cfd_df_raw.iterrows():
        folder = instance.Model
        score = instance[property]
        for file in instance.files:
            tmp_instance = []
            tmp_instance.append(folder)
            tmp_instance.append(file)
            tmp_instance.append(score)
            cfd_instances.append(tmp_instance)
```

Image 9: Norming data extraction



```
[ ] def findEmotion(file):
        #file = CFD-WM-040-023-HO.jpg
        file_name = file.split(".")[0]
        emotion = file_name.split("-")[4]
        return emotion

    def findRace(file):
        file_name = file.split(".")[0]
        race = file_name.split("-")[1][0]
        return race

    def findGender(file):
        file_name = file.split(".")[0]
        gender = file_name.split("-")[1][1]
        return gender
```

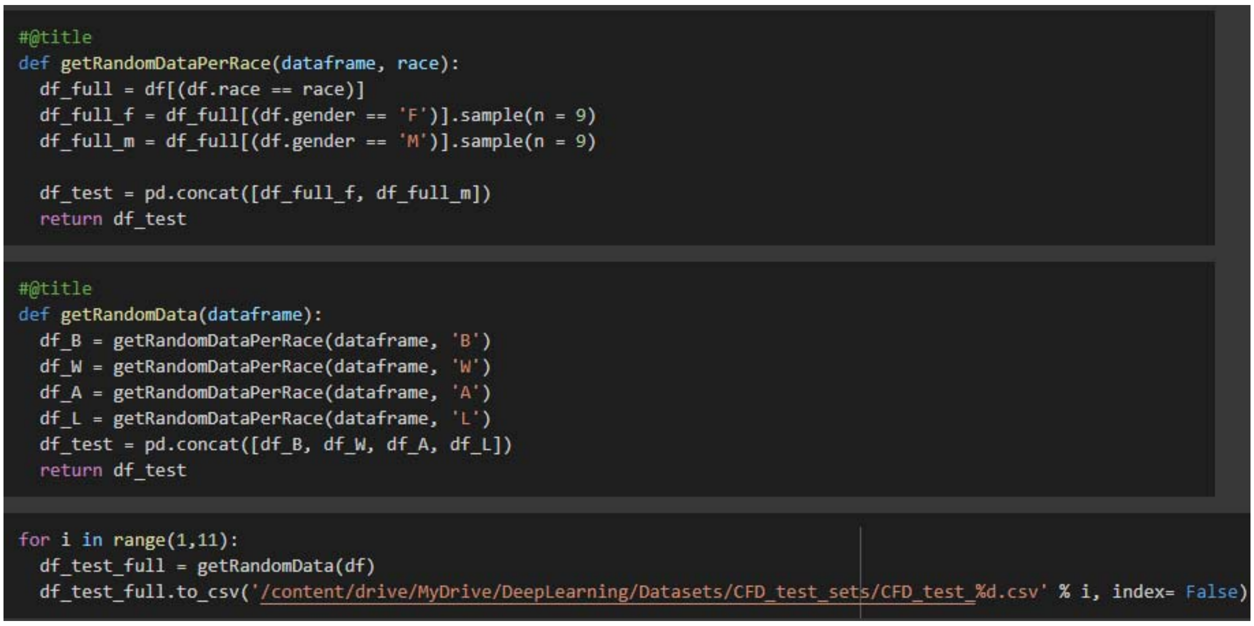
```
[ ] df = pd.DataFrame(cfd_instances, columns=["folder", "file", "score"])
```

```
[ ] df['emotion'] = df.file.apply(findEmotion)
    df['race'] = df.file.apply(findRace)
    df['gender'] = df.file.apply(findGender)
```

```
[ ] df = df[(df.emotion == 'N') | (df.emotion == 'HC')]
    df['filename'] = df['file']
    df['file'] = "/content/data/CFD/"+df["folder"]+"/"+df['file']
```

Image 10: Dataset pre-process

The test sets had to be extracted and saved at the pre-processing stage in order to be used for predictions on all the models that would be subsequently created. Considering that the number of images in the test set was relatively small due to the overall small dataset, aiming to improve the reliability of the final results, ten different test sets were randomly created applying the same method.



```
#@title
def getRandomDataPerRace(dataframe, race):
  df_full = df[(df.race == race)]
  df_full_f = df_full[(df.gender == 'F')].sample(n = 9)
  df_full_m = df_full[(df.gender == 'M')].sample(n = 9)

  df_test = pd.concat([df_full_f, df_full_m])
  return df_test
```

```
#@title
def getRandomData(dataframe):
  df_B = getRandomDataPerRace(dataframe, 'B')
  df_W = getRandomDataPerRace(dataframe, 'W')
  df_A = getRandomDataPerRace(dataframe, 'A')
  df_L = getRandomDataPerRace(dataframe, 'L')
  df_test = pd.concat([df_B, df_W, df_A, df_L])
  return df_test
```

```
for i in range(1,11):
  df_test_full = getRandomData(df)
  df_test_full.to_csv('/content/drive/MyDrive/DeepLearning/Datasets/CFD_test_sets/CFD_test_%d.csv' % i, index= False)
```

Image 11: Test set extraction

4.3 CNN implementation & training

As mentioned in the Introduction chapter, the purpose of this experiment is to determine how racially biased data affect the prediction results on an attractiveness prediction model. For this purpose, 5 different types of CNN models were implemented. The Type F models were trained on all available images, and the rest were trained on a subset created by excluding one of the four races (Table 2).

Model type	Information
Type F	Model trained on the full dataset
Type W	Model trained on a subset, excluding White ethnicity individuals
Type B	Model trained on a subset, excluding Black ethnicity individuals
Type A	Model trained on a subset, excluding Asian ethnicity individuals
Type L	Model trained on a subset, excluding Latino ethnicity individuals

Table 2: Model type categorization

The training versus validation ratio was 85/15 for all models and the number of images in each test set was set to 72 (Table 3). For Type F, the full test set (72 images) was used for predictions while for the rest, an ethnicity was excluded from the test set depending on the model type. For example, for Type A models, images of Asian models were excluded from the test set resulting in a final set of 54 mages.

Model type	Total set	Training set	Validation set	Test set
Type F	678	576	102	72
Type W	440	374	66	54
Type B	420	357	63	54
Type A	594	505	89	54
Type L	594	505	89	54

Table 3: Number of images per model type and set

All Convolutional Networks implemented had the same architecture, based on the VGG-Face model (Image 12). The model was reconstructed, and the available trained weights were applied. The last seven layers of the base model were trained on the dataset while the rest were kept unmodified in order to take advantage of the model's existing ability to recognize facial features and patterns (Image 13). As this is a regression problem, the loss function selected was mean squared error (Image 14). Mean squared error is calculated as the average of the squared differences between the predicted and actual values. The result is always positive regardless of the sign of the predicted and actual values and a perfect value is 0.0. The squaring means that larger mistakes result in more error than smaller mistakes, meaning that the model is punished for making larger mistakes. ADAM was selected as the optimization algorithm, which is best suited for the problem. According to Kingma et al., 2014, the method is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters". Finally, the epoch number was set to 2000, but an early stopping callback was used with a patience number of 50, meaning that if the validation loss didn't decrease for 50 rounds the training would be terminated. The checkpointer was responsible for saving the best performing model and weights through the course of the training.

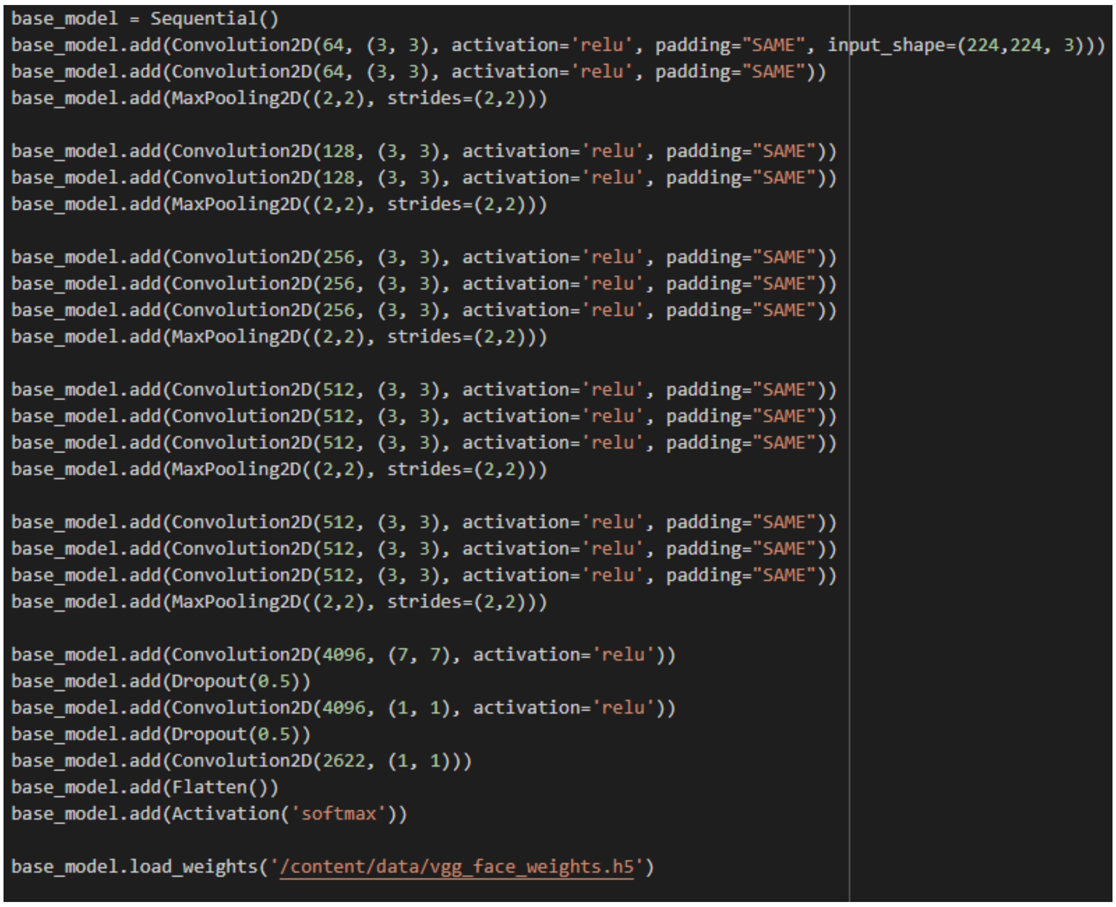
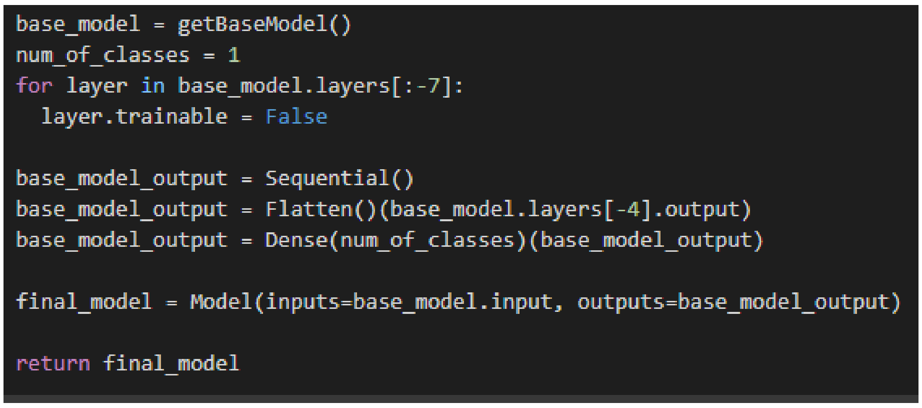
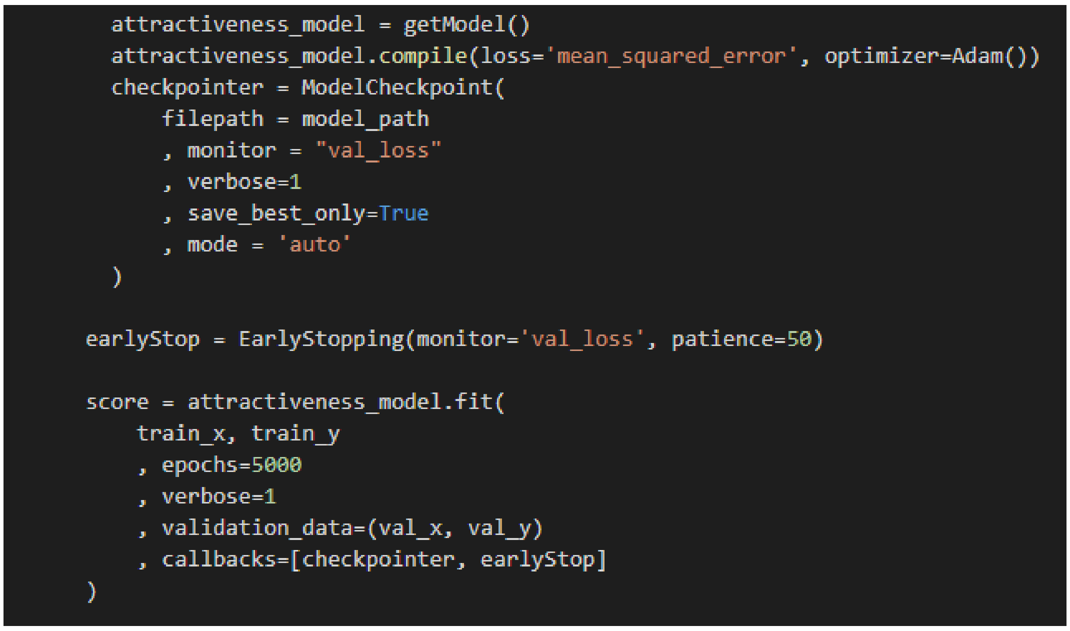

```
base_model = Sequential()
base_model.add(Convolution2D(64, (3, 3), activation='relu', padding="SAME", input_shape=(224,224, 3)))
base_model.add(Convolution2D(64, (3, 3), activation='relu', padding="SAME"))
base_model.add(MaxPooling2D((2,2), strides=(2,2)))

base_model.add(Convolution2D(128, (3, 3), activation='relu', padding="SAME"))
base_model.add(Convolution2D(128, (3, 3), activation='relu', padding="SAME"))
base_model.add(MaxPooling2D((2,2), strides=(2,2)))

base_model.add(Convolution2D(256, (3, 3), activation='relu', padding="SAME"))
base_model.add(Convolution2D(256, (3, 3), activation='relu', padding="SAME"))
base_model.add(Convolution2D(256, (3, 3), activation='relu', padding="SAME"))
base_model.add(MaxPooling2D((2,2), strides=(2,2)))

base_model.add(Convolution2D(512, (3, 3), activation='relu', padding="SAME"))
base_model.add(Convolution2D(512, (3, 3), activation='relu', padding="SAME"))
base_model.add(Convolution2D(512, (3, 3), activation='relu', padding="SAME"))
base_model.add(MaxPooling2D((2,2), strides=(2,2)))

base_model.add(Convolution2D(512, (3, 3), activation='relu', padding="SAME"))
base_model.add(Convolution2D(512, (3, 3), activation='relu', padding="SAME"))
base_model.add(Convolution2D(512, (3, 3), activation='relu', padding="SAME"))
base_model.add(MaxPooling2D((2,2), strides=(2,2)))

base_model.add(Convolution2D(4096, (7, 7), activation='relu'))
base_model.add(Dropout(0.5))
base_model.add(Convolution2D(4096, (1, 1), activation='relu'))
base_model.add(Dropout(0.5))
base_model.add(Convolution2D(2622, (1, 1)))
base_model.add(Flatten())
base_model.add(Activation('softmax'))

base_model.load_weights('/content/data/vgg_face_weights.h5')
```

Image 12: VGG-Face model implementation



```
base_model = getBaseModel()
num_of_classes = 1
for layer in base_model.layers[:-7]:
  layer.trainable = False

base_model_output = Sequential()
base_model_output = Flatten()(base_model.layers[-4].output)
base_model_output = Dense(num_of_classes)(base_model_output)

final_model = Model(inputs=base_model.input, outputs=base_model_output)

return final_model
```

Image 13: Final model implementation



```
      attractiveness_model = getModel()
      attractiveness_model.compile(loss='mean_squared_error', optimizer=Adam())
      checkpointer = ModelCheckpoint(
          filepath = model_path
          , monitor = "val_loss"
          , verbose=1
          , save_best_only=True
          , mode = 'auto'
      )

    earlyStop = EarlyStopping(monitor='val_loss', patience=50)

    score = attractiveness_model.fit(
        train_x, train_y
        , epochs=5000
        , verbose=1
        , validation_data=(val_x, val_y)
        , callbacks=[checkpointer, earlyStop]
    )
```

Image 14: Model compiling and training

In the data pre-processing section, it was mentioned that ten different test sets were created. As each test set needed to be excluded from the training data, the training of each of the five model types needed to be repeated for every test set , resulting in ten iterations of training, each producing a new model, resulting in a final number of fifty trained models. Following are the training/validation plots for all model types during the first iteration (Figures 2-6).

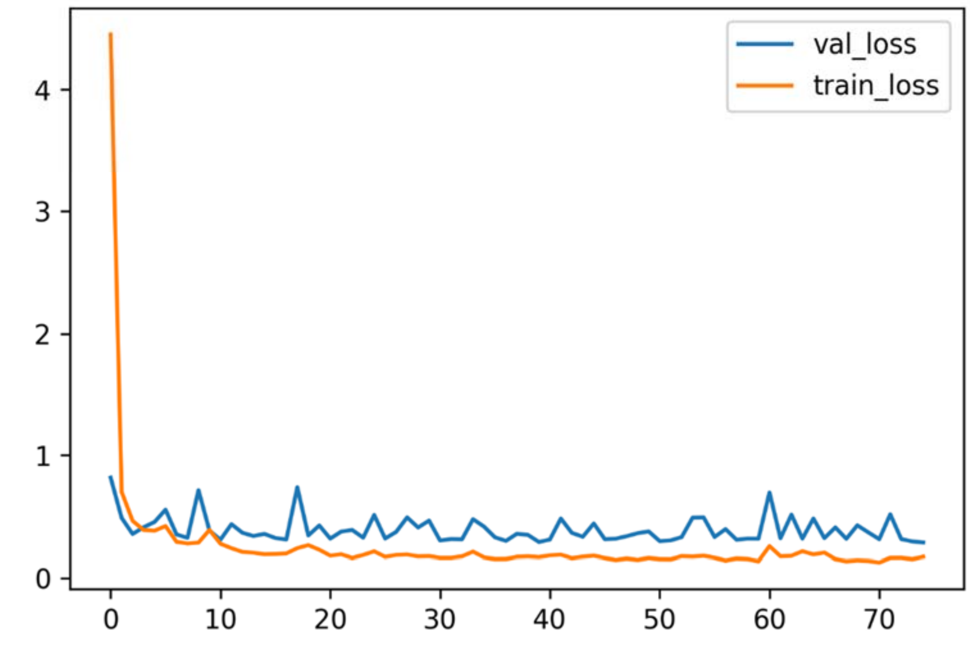


Figure 2: Model Type F - Train & validation loss

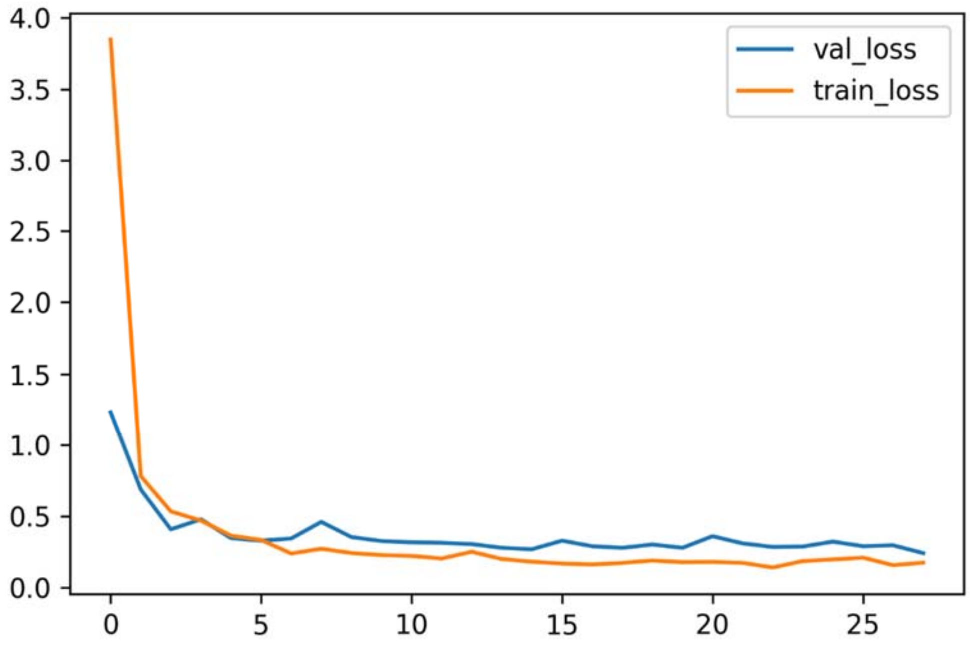


Figure 3: Model Type W - Train & validation loss

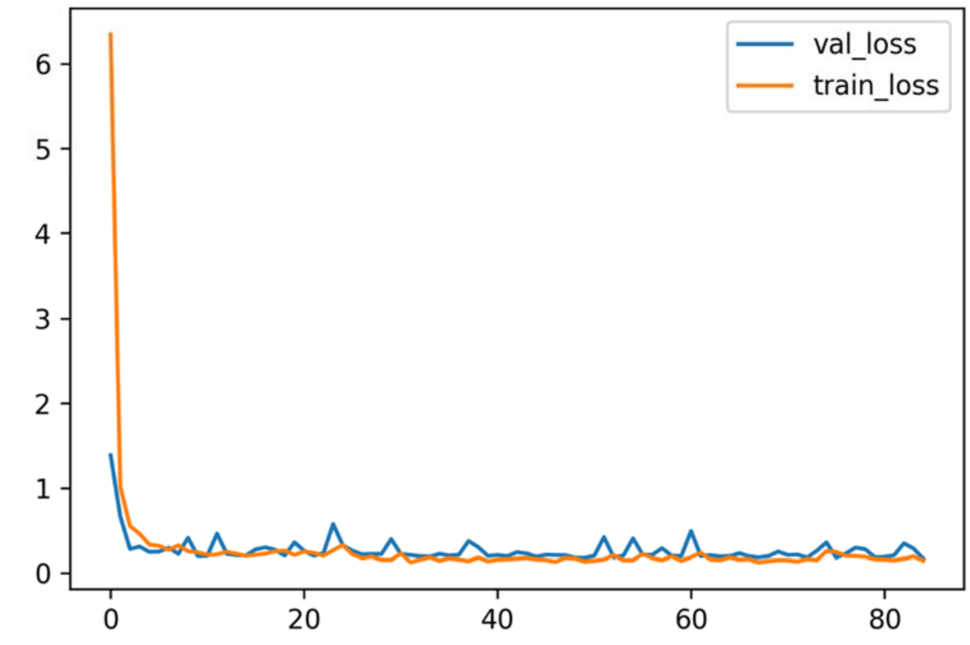


Figure 4: Model Type B - Train & validation loss

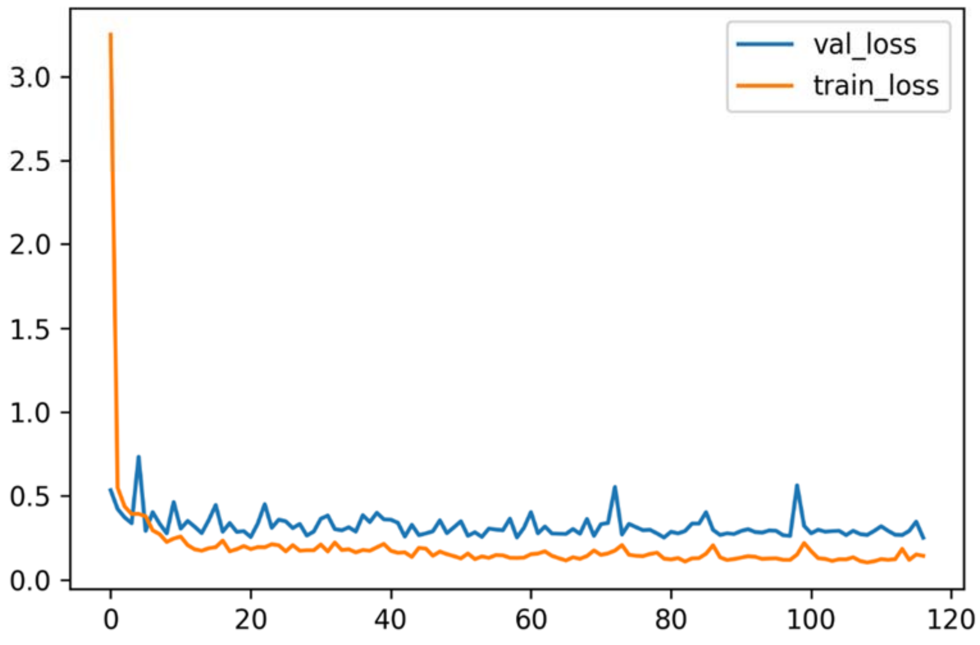


Figure 5: Model Type A - Train & validation loss

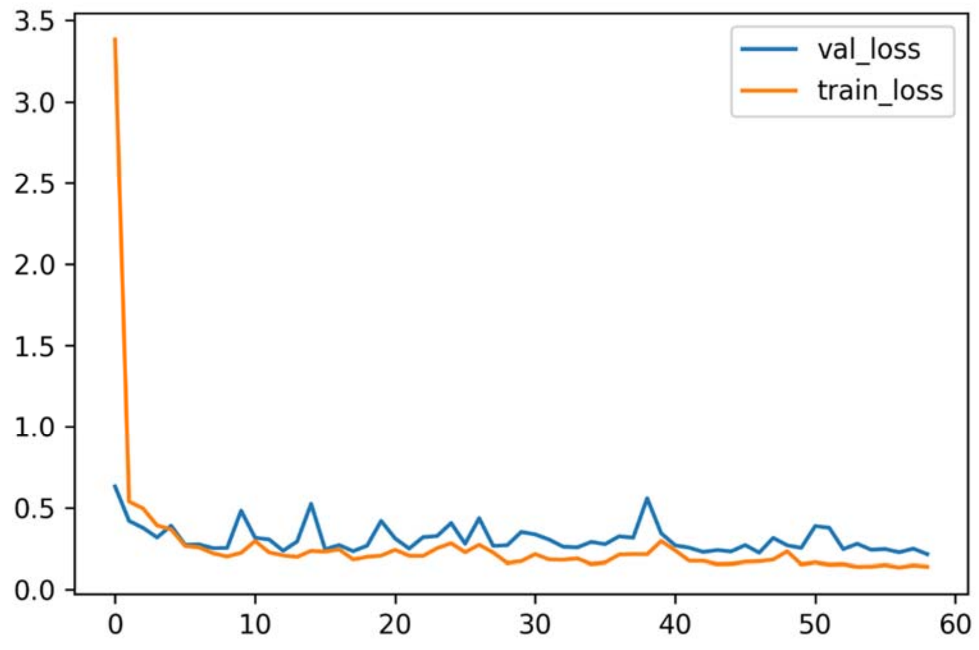


Figure 6: Model Type L - Train & validation loss

4.4 CNN layer visualization

Visualizing CNNs can give us a better understanding of how the network extracts information from the input image and how each layer and filter processes the input to come to a final result. The technique used, takes an input image and plots what each filter has extracted after a convolution operation in each layer.

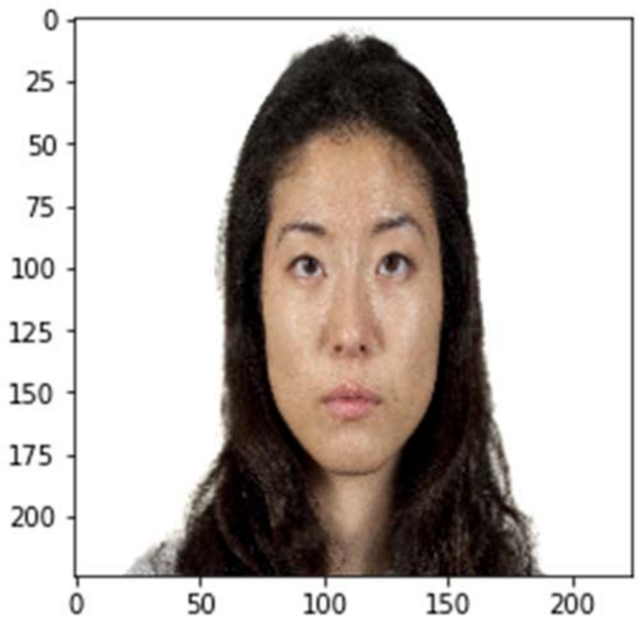
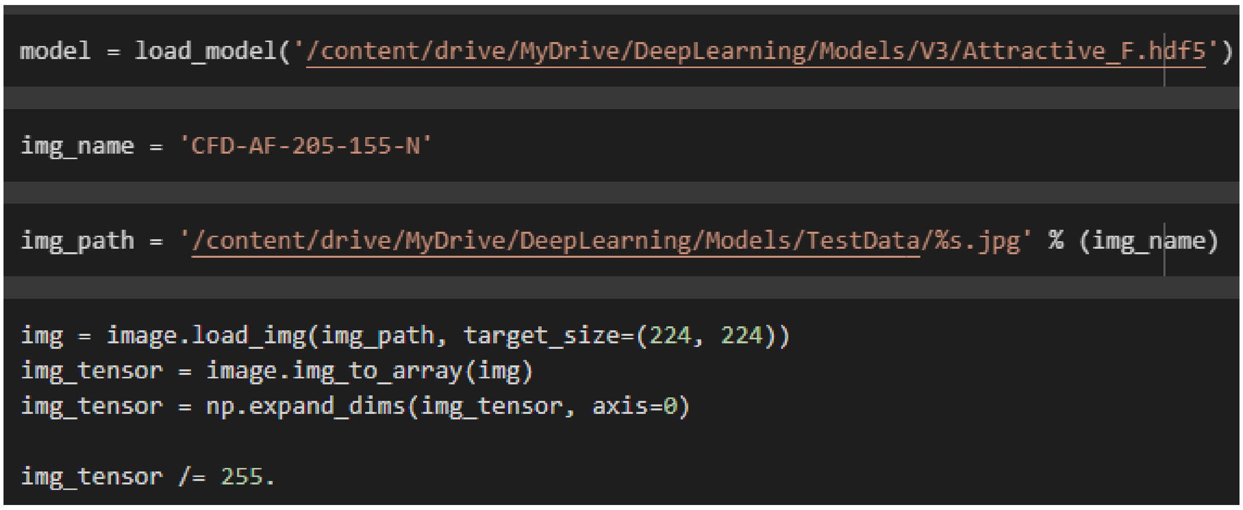


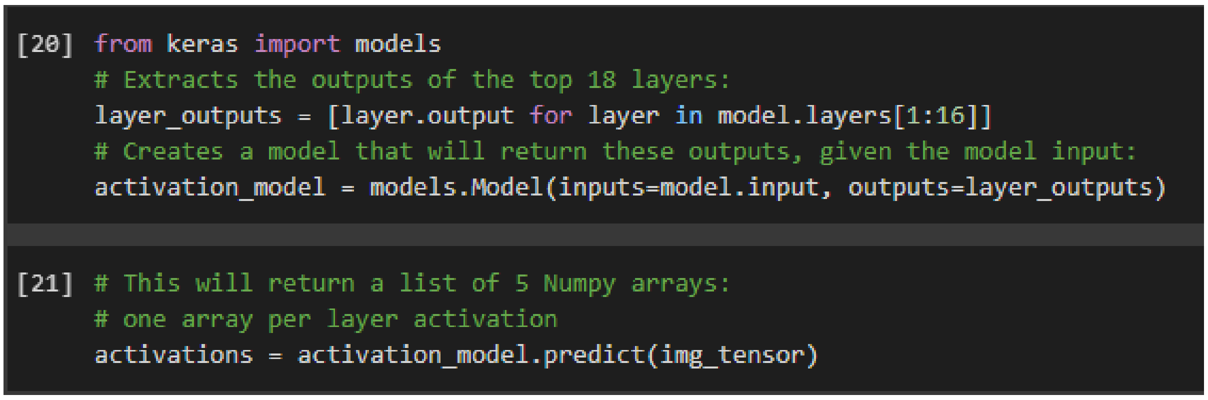
Image 15: Input image

For this process a Type F model was used, and the input image was randomly selected. The first step was to load the previously trained model and the input image. In order to extract the feature maps, a Keras model was created, that takes batches of images as input, and outputs the activations of all convolution and pooling layers. When fed an image input, this model returns the values of the layer activations in the original model.



```
model = load_model('/content/drive/MyDrive/DeepLearning/Models/V3/Attractive_F.hdf5')

img_name = 'CFD-AF-205-155-N'

img_path = '/content/drive/MyDrive/DeepLearning/Models/TestData/%s.jpg' % (img_name)

img = image.load_img(img_path, target_size=(224, 224))
img_tensor = image.img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis=0)

img_tensor /= 255.
```

Image 16: Model & input image loading



```
[20] from keras import models
     # Extracts the outputs of the top 18 layers:
     layer_outputs = [layer.output for layer in model.layers[1:16]]
     # Creates a model that will return these outputs, given the model input:
     activation_model = models.Model(inputs=model.input, outputs=layer_outputs)

[21] # This will return a list of 5 Numpy arrays:
     # one array per layer activation
     activations = activation_model.predict(img_tensor)
```

Image 17: Implementing model for visualization

Finally, a complete visualization of all the activations in the first 15 layers was plotted and saved. Each image created shows the visualized activation of each filter in a layer.

Image 18: conv2d

Image 19: conv2d_1

Image 20: max_pooling2d

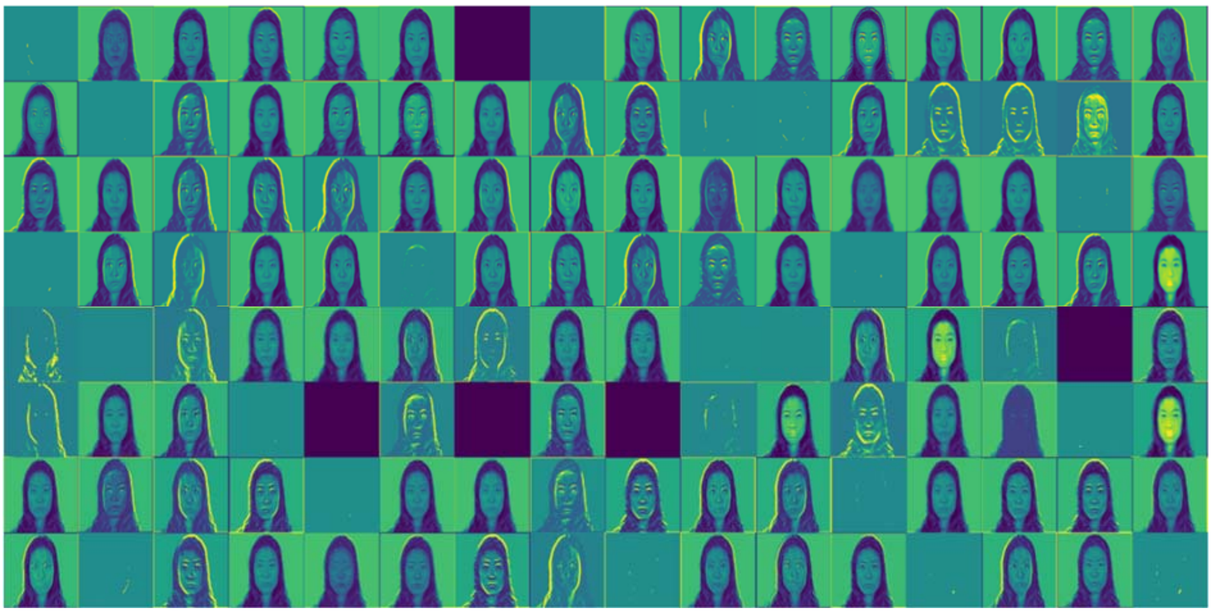


Image 21: conv2d_2

Image 22: conv2d_3

Image 23: max_pooling2d_1

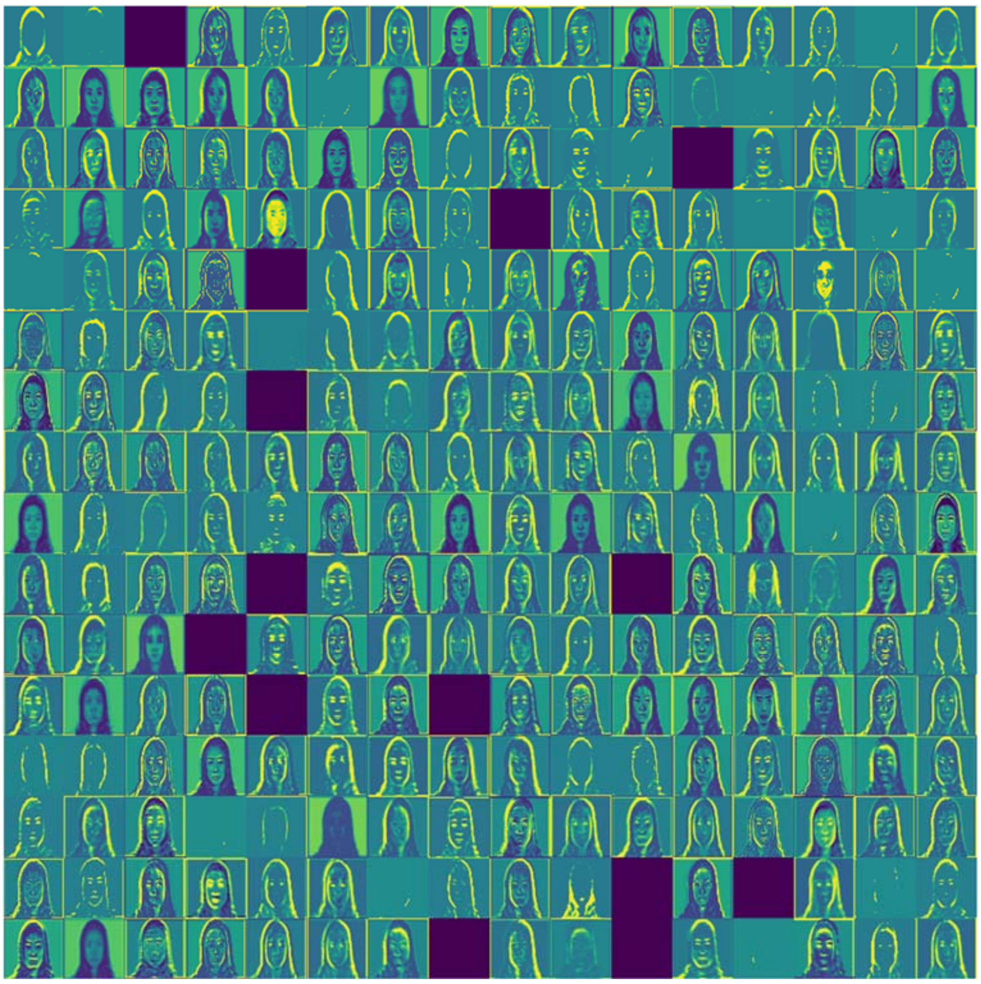


Image 24: conv2d_4

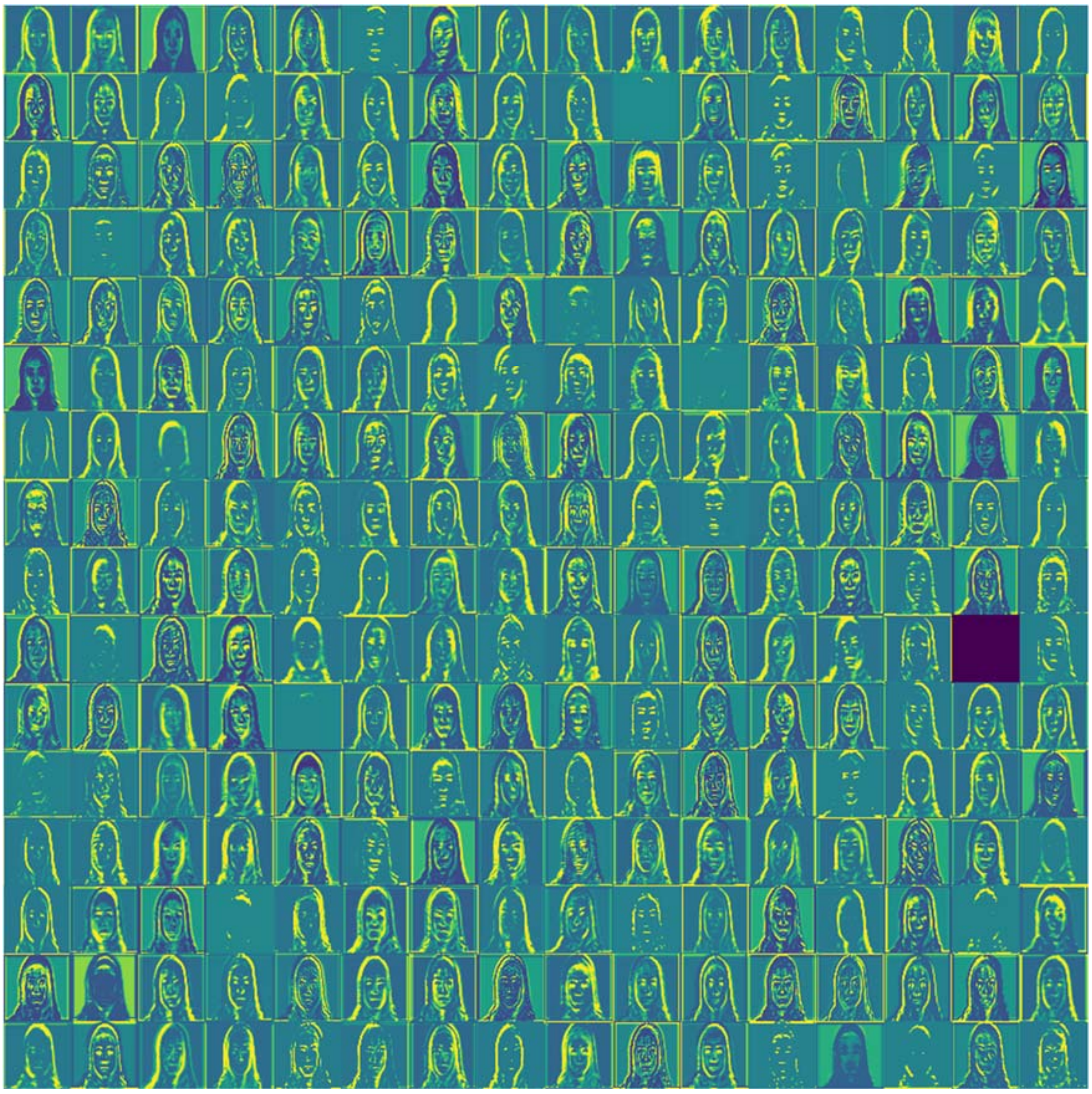


Image 25: conv2d_5

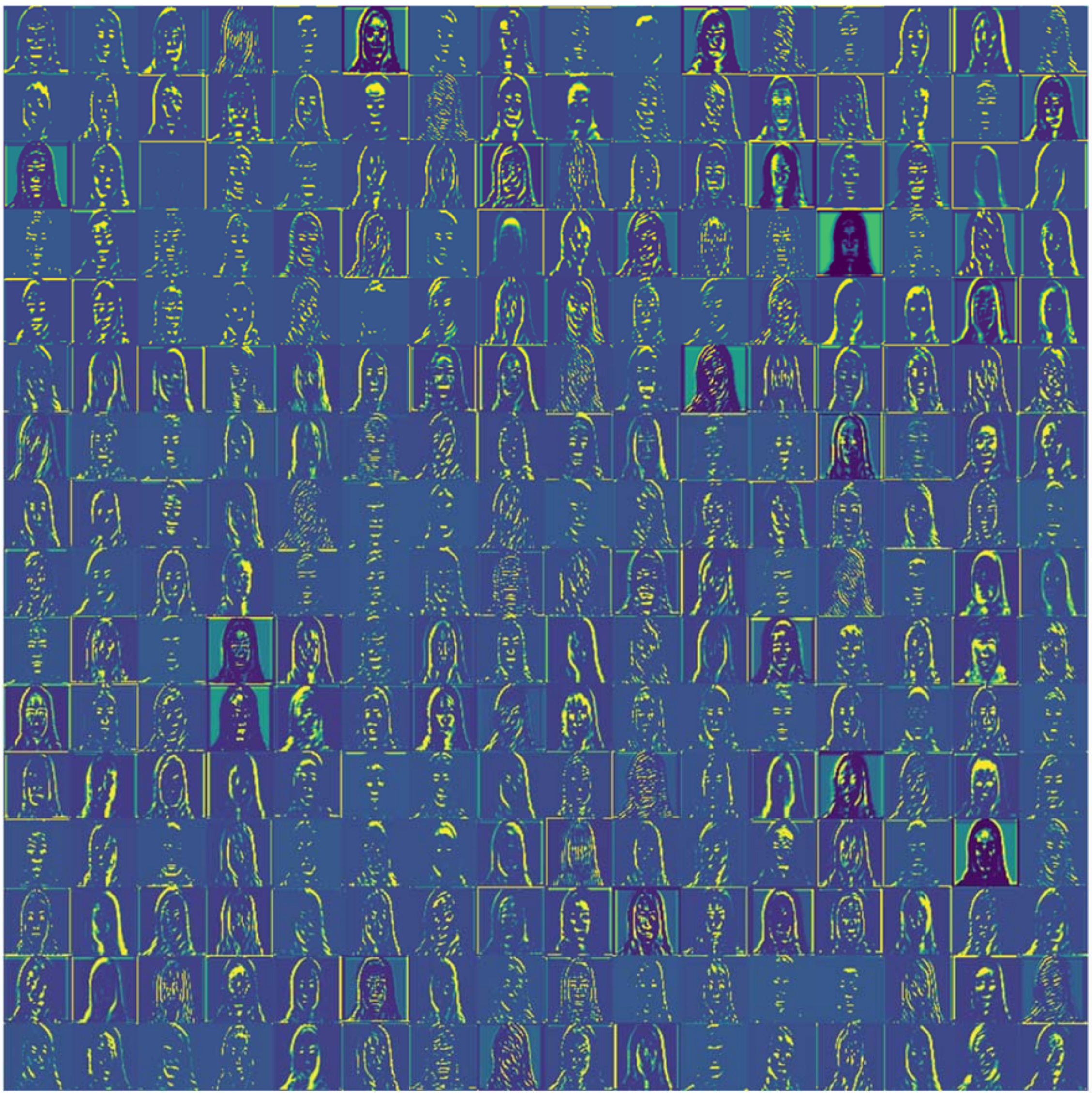


Image 26: conv2d_6

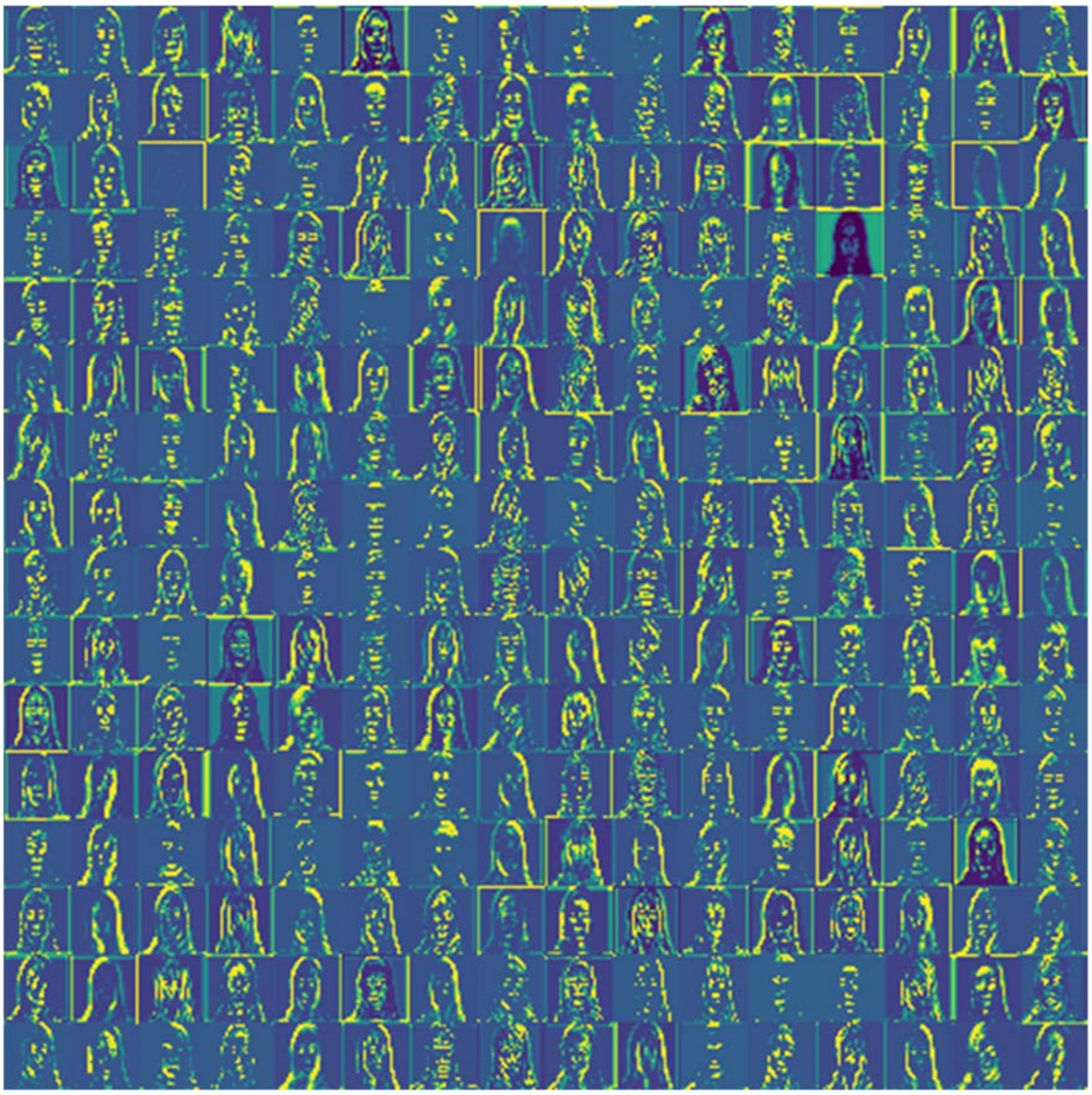


Image 27: max_pooling2d_2

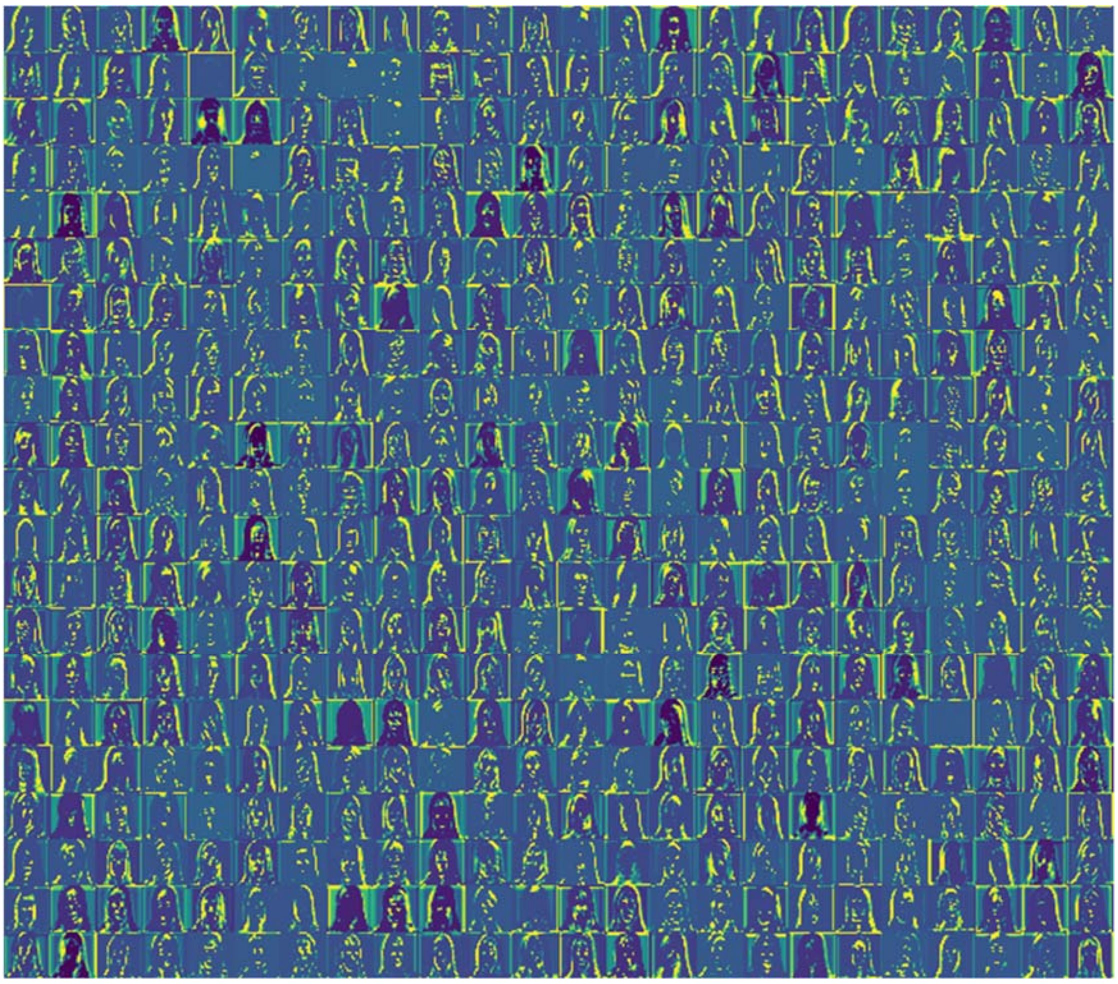


Image 28: conv2d_7

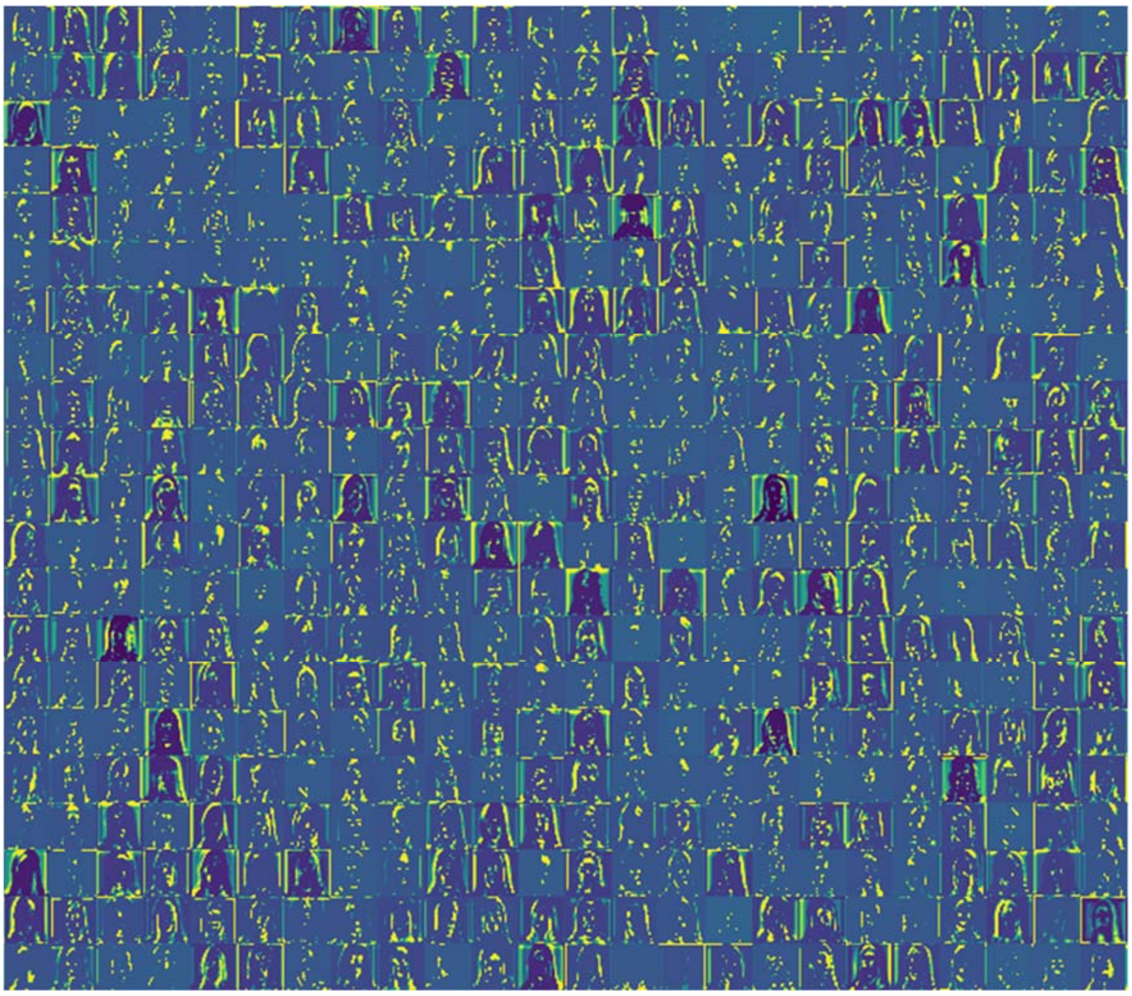


Image 29: conv2d_8

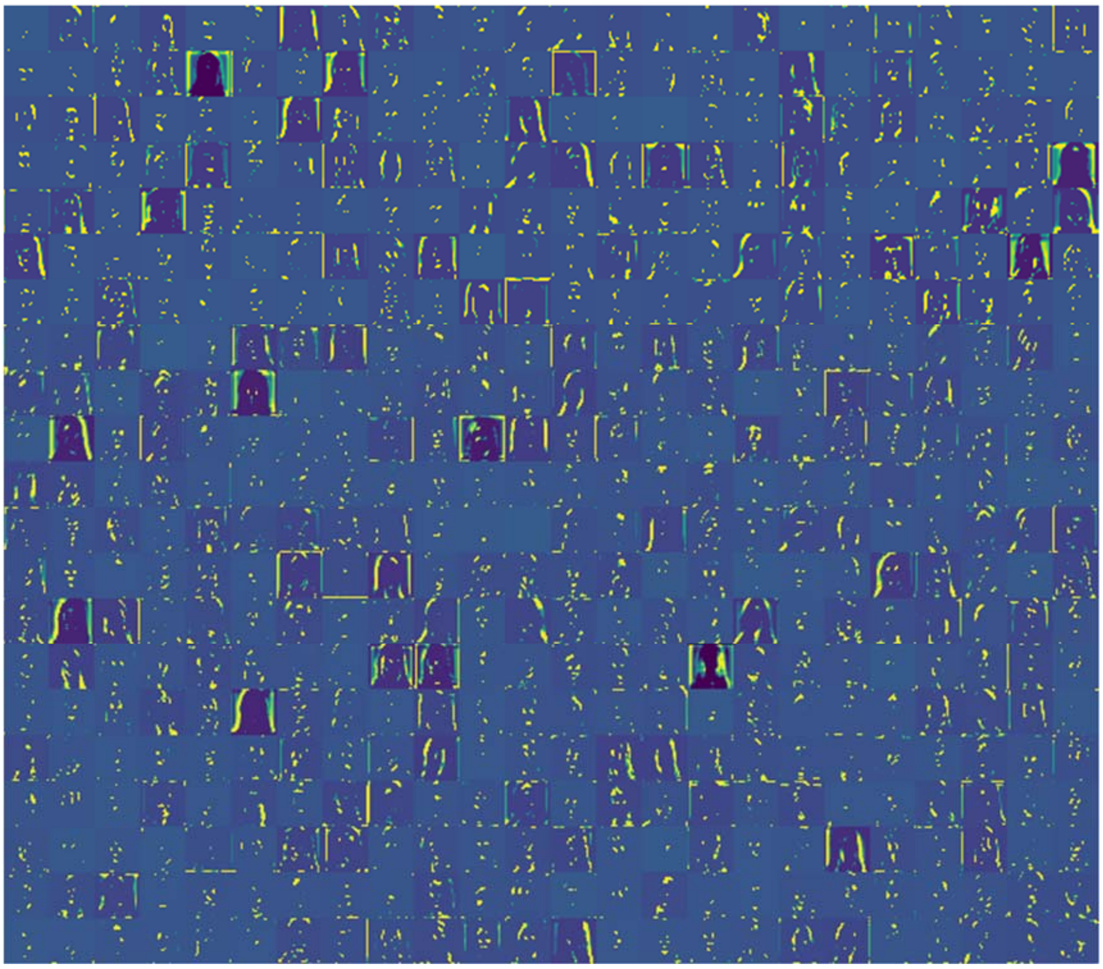


Image 30: conv2d_9

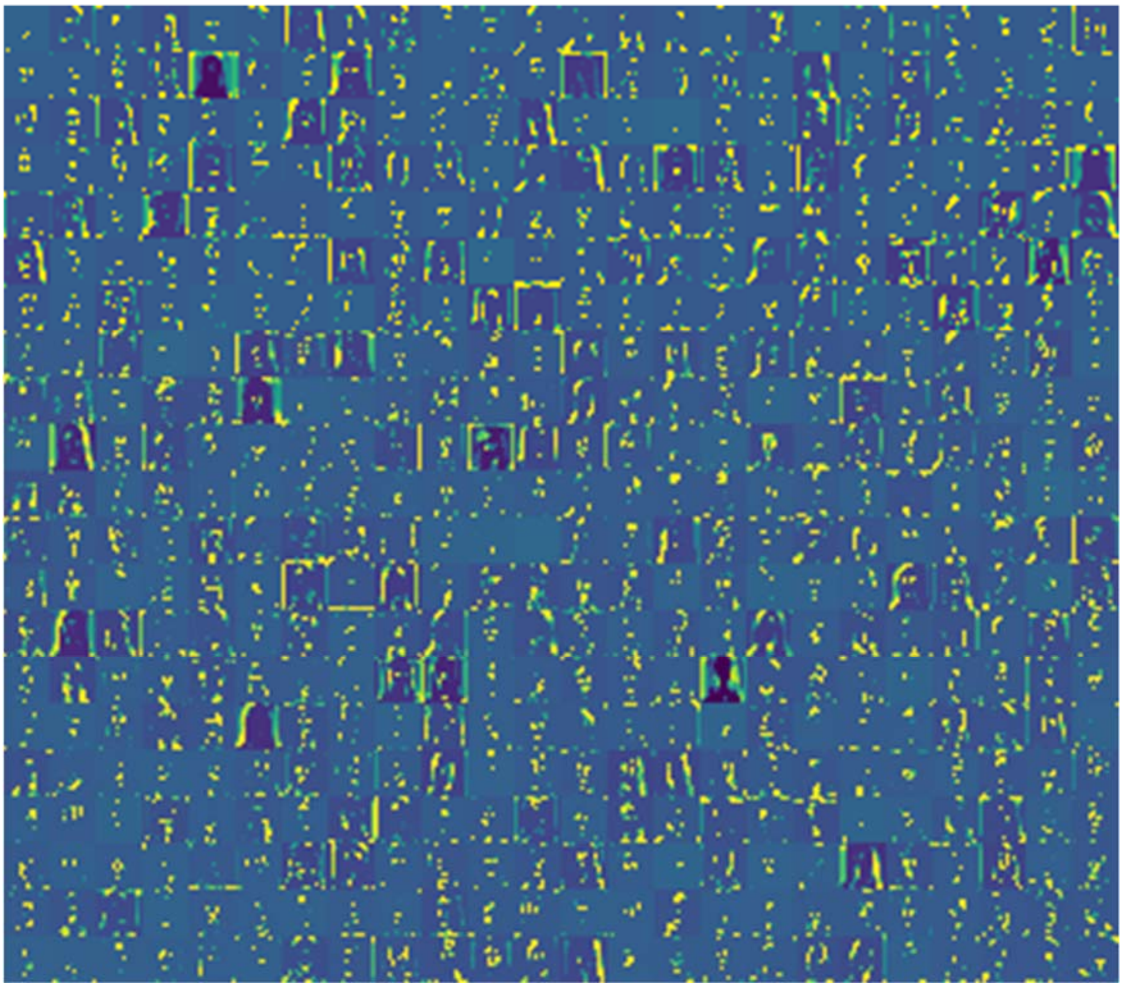


Image 31: max_pooling2d_3

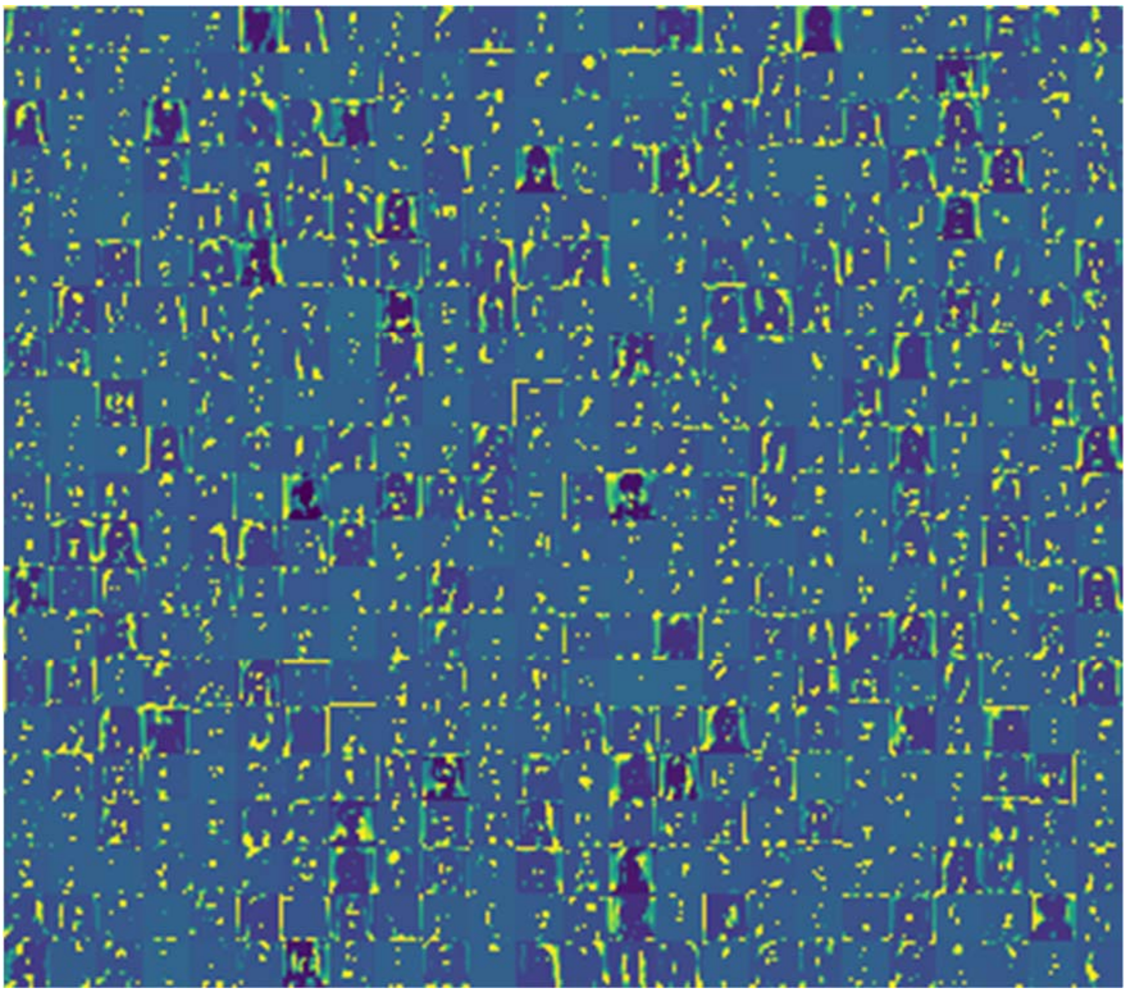


Image 32: conv2d_10

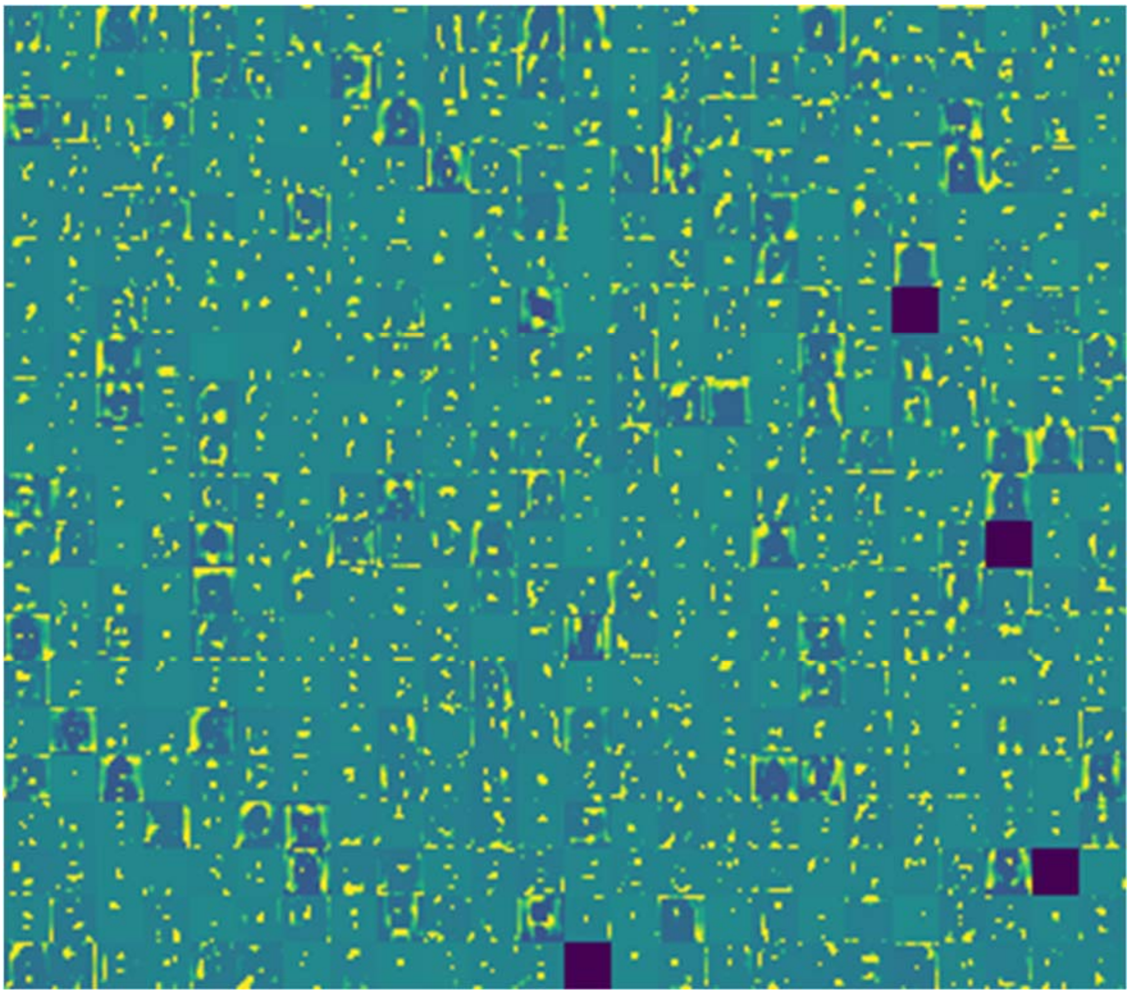


Image 33: conv2d_11

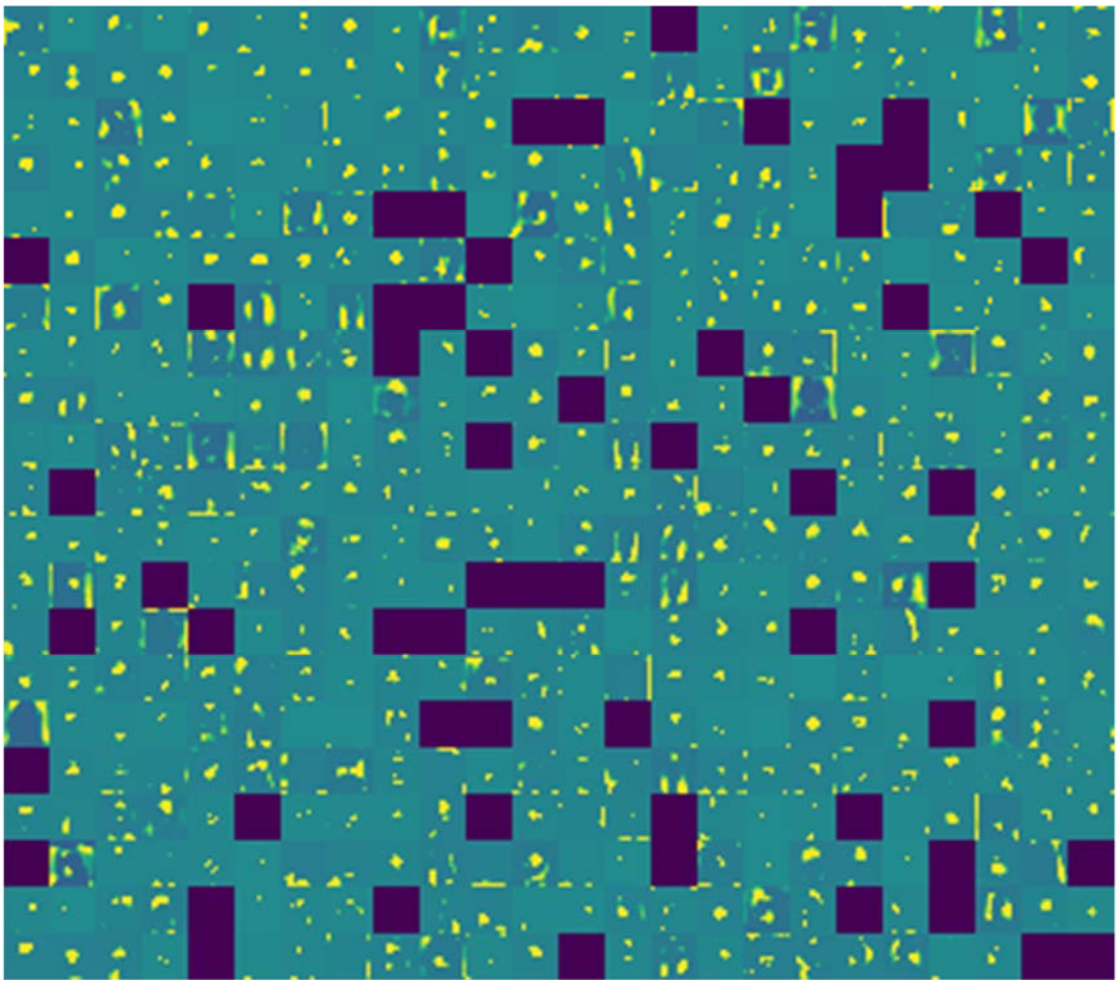


Image 34: conv2d_12

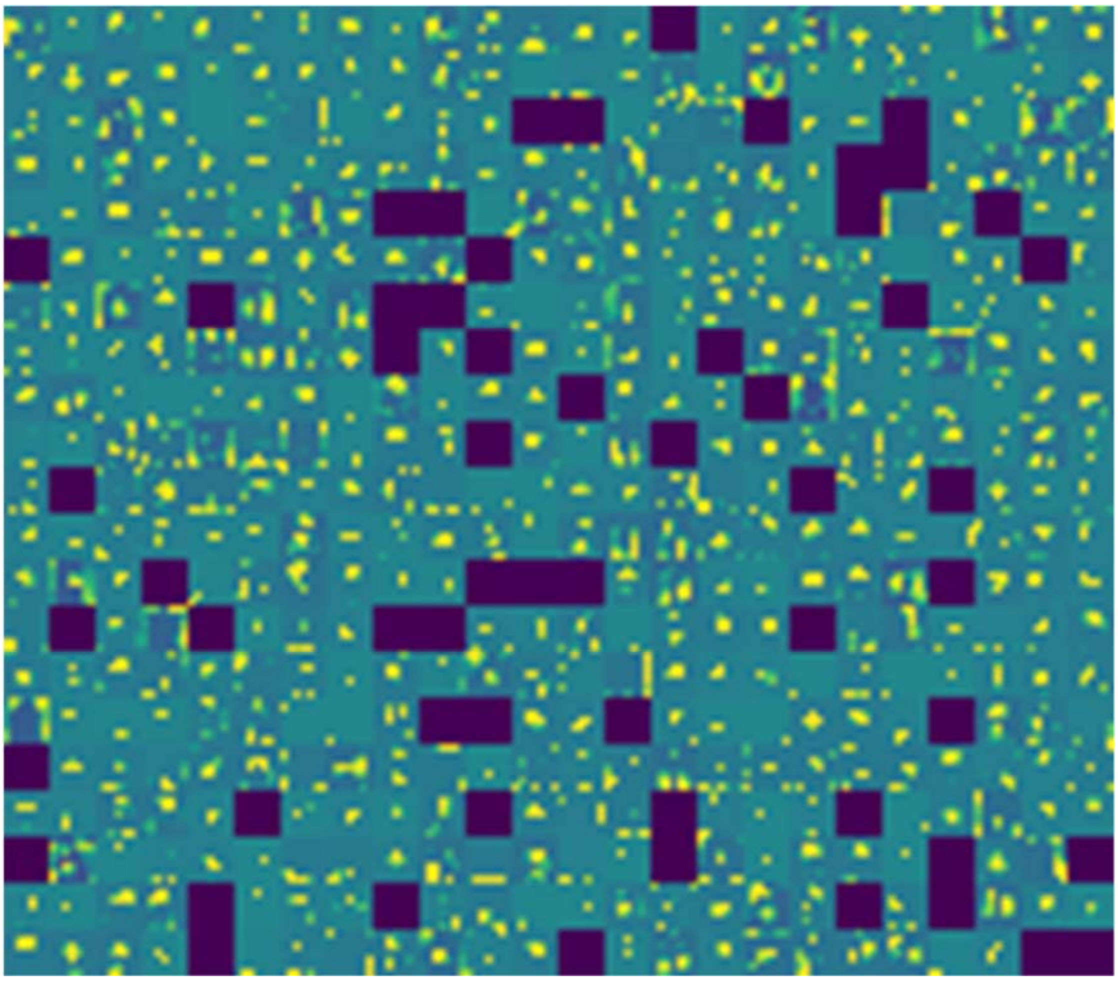


Image 35: max_pooling2d_4

The initial layers (Images 18-26) seem to retain most of the input image features. It looks like the convolution filters are activated at every part of the input image. This gives us an intuition that these initial filters might be primitive edge detectors.

As we go deeper (Images 27-35) the features extracted by the filters become visually less interpretable. An explanation for this can be that the network is now abstracting away visual information of the input image and trying to convert it to the required output.

In some images (especially Images 34-35) there are a lot of blank convolution outputs. This means that the pattern encoded by the filters was not found in the input image. As we progress through the network the filters try to extract more complex shapes and patterns that in some cases don’t exist in an image, which results in a blank output.

5. EXPERIMENTATION RESULTS

In this chapter, all prediction results will be displayed and analyzed. Two basic metrics were calculated in order to evaluate each model's performance: Pearson correlation and mean absolute error. The Pearson correlation coefficient is a measure of linear correlation between two sets of data. Mean absolute error is a measure of errors between paired observations expressing the same phenomenon. In our case, it is the error between attractiveness predictions and actual values. Additionally, scatter plots of actual vs predicted values were created for each case, containing paired values of all ten models of each type.

5.1 Type F model performance

Type F model, as mentioned in the previous chapter, is a model trained on the full dataset containing images of all four ethnicities. Ten models of this type were trained, each with a different test set.

The Pearson correlation for the validation set is acceptable (Table 4) with an average value of 0.6981 and mean square error is at 0.43. Considering that the attractiveness score is between 1 and 7, the model can predict it with ±0.4 error. Additionally, the test set results are almost identical, which indicates that the model has the ability to generalize and predict scores for input images that weren't included in the training process.

ModelNo	Validation set		Test set	
	Pearson correlation	Mean absolute error	Pearson correlation	Mean absolute error
1	0.7114608706	0.3891693419	0.5851807451	0.5400051229
2	0.6488464311	0.4268629942	0.7069867723	0.4020390598
3	0.6114909335	0.4974872831	0.6821329715	0.4098491618
4	0.6686423875	0.4671703047	0.7689630167	0.362511383
5	0.7708477094	0.3718037034	0.7897271135	0.4013349699
6	0.7384085026	0.4268569263	0.6942402076	0.4217622864
7	0.7370681529	0.4103251549	0.6600841722	0.4627719748
8	0.6502897626	0.456986166	0.7073352683	0.4519757964
9	0.7345835474	0.3849640041	0.7449945108	0.4129380476
10	0.7094588302	0.4569267838	0.6571672391	0.4844723094
Average	**0.6981097128**	**0.4288552662**	**0.6996812017**	**0.4349660112**

Table 4: Type F model - Validation/test set performance

In the respective scatter plots (Figures 8, 9) a positive linear relationship between actuals and predictions is depicted. The grouping is slightly tighter for the validation set, which is in line with the metrics on Table 4.

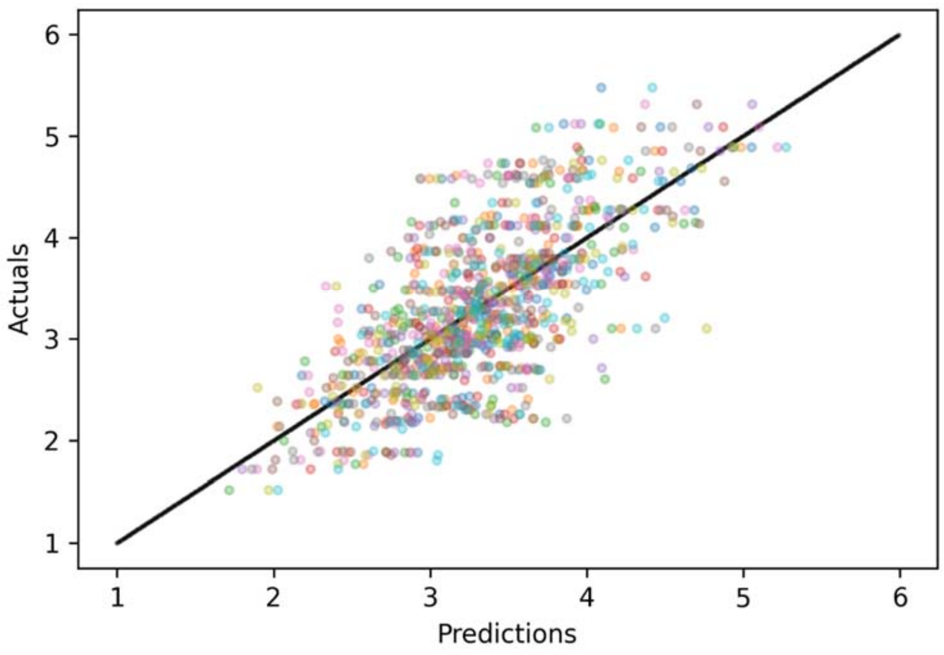


Figure 7: Type F model - Scatter plot on validation set predictions

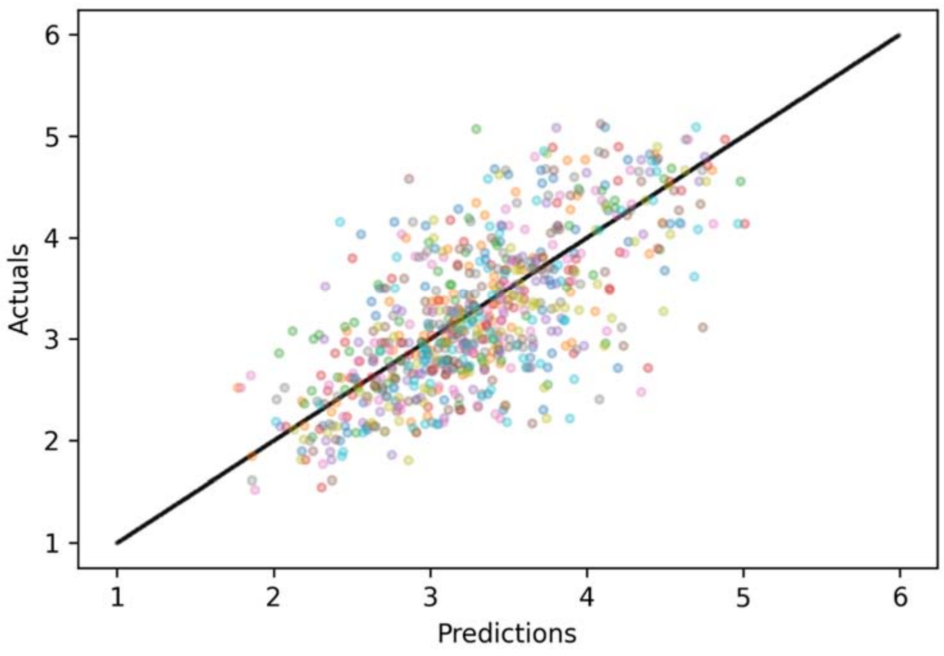


Figure 8: Type F model - Scatter plot on test set predictions

5.2 Type W model performance

Type W model, as mentioned in the previous chapter, is a model trained on a subset, excluding White ethnicity individuals. Ten models of this type were trained, each with a different test set.

The Pearson correlation for the validation set is acceptable (Table 5) with an average value of 0.7054 and mean square error is at 0.41 which means that the model can predict attractiveness with ±0.4 error. Additionally, the prediction performance results for the test set are slightly lower with a moderate correlation. These results supported by the scatter plots for validation and test sets (Figures 10-11), where there is a strong linear relationship between actual and predicted values, with a slightly tighter grouping for the validation set.

ModelNo	Validation set		Test set	
	Pearson correlation	Mean absolute error	Pearson correlation	Mean absolute error
1	0.7522903866	0.3746734471	0.5191768892	0.5524340469
2	0.7076249169	0.4225660206	0.6419403264	0.4307983066
3	0.7267429516	0.4028885525	0.7191278017	0.3814956607
4	0.6921820072	0.4366546432	0.6915883937	0.4319676529
5	0.6986949243	0.3977268467	0.7396080468	0.4670180879
6	0.7465498614	0.4157233258	0.6241018735	0.4476720286
7	0.7346080891	0.3881537467	0.6064066948	0.4899079148
8	0.6997446262	0.3794407606	0.7768880973	0.399471898
9	0.5937631784	0.4351835797	0.6477580512	0.4218431584
10	0.7013180302	0.4237639553	0.6232046173	0.4933024214
Average	**0.7053518972**	**0.4076774878**	**0.6589800792**	**0.4515911176**

Table 5: Type W model - Validation/test set performance

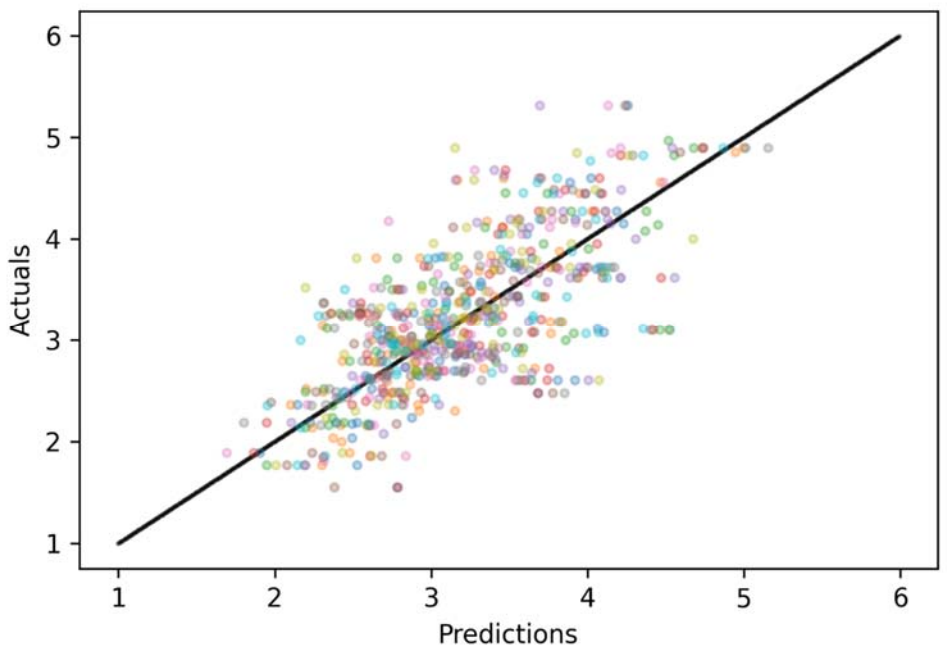


Figure 9: Type W model - Scatter plot on validation set predictions

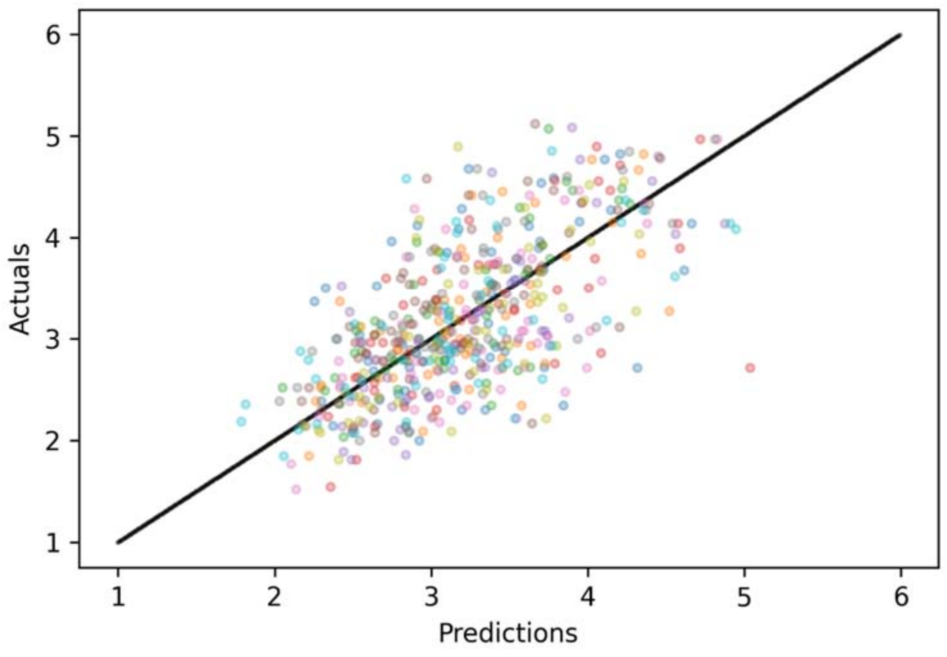


Figure 10: Type W model - Scatter plot on test set predictions

For the purpose of testing the effects of excluding an entire group of individuals from the training data, predictions were calculated for the excluded group on both the type W and type F models. From the results displayed in Table 6, it is clear that the predictions on the type W model had a low degree of correlation as an average and high error values. In some cases, like model number 7, the correlation nears zero and error value is almost 0.8. On the other hand, the predictions of the same set of images on the type F model are clearly more accurate with a high average correlation of 0.8585 and 0.40 average error.

ModelNo	Type W - Excluded Test set		Type F - Excluded Test set	
	Pearson correlation	Mean absolute error	Pearson correlation	Mean absolute error
1	0.4074935666	0.6673390262	0.8066008993	0.4124333667
2	0.7662039926	0.4523312389	0.8826492274	0.3115141459
3	0.5901058163	0.5106764415	0.7376284336	0.3413824274
4	0.2332453885	0.4921994914	0.6013614998	0.3836462457
5	0.1517063933	0.5380770538	0.8584553068	0.2882150592
6	0.4877419835	0.6558088893	0.8554576931	0.4250530929
7	0.06143093995	0.7999063731	0.6020476274	0.499391643
8	0.09370903376	0.810165702	0.5989702988	0.5240724138
9	0.5586956412	0.5528191668	0.6781391886	0.4376925084
10	0.5907160873	0.6305290722	0.8859977355	0.3849337008
Average	**0.3941048843**	**0.6109852455**	**0.750730791**	**0.4008334604**

Table 6: Type W model - Excluded test set (White) - Type W/F comparison

Examining the scatter plots (Figures 12-13)shows a weak positive relationship between actuals and predictions for the Type W models while the same data have a strong, linear and positive relationship on Type F models. These results support the metrics in Table 6 and visualize the effect of the introduced bias on the model.

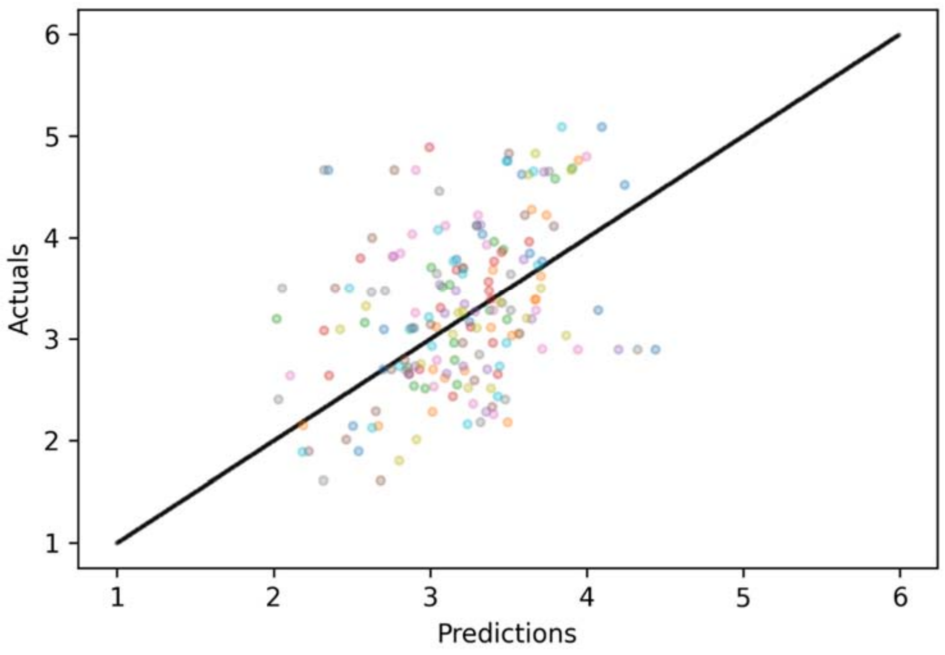


Figure 11: Type W model - Scatter plot on excluded test set predictions

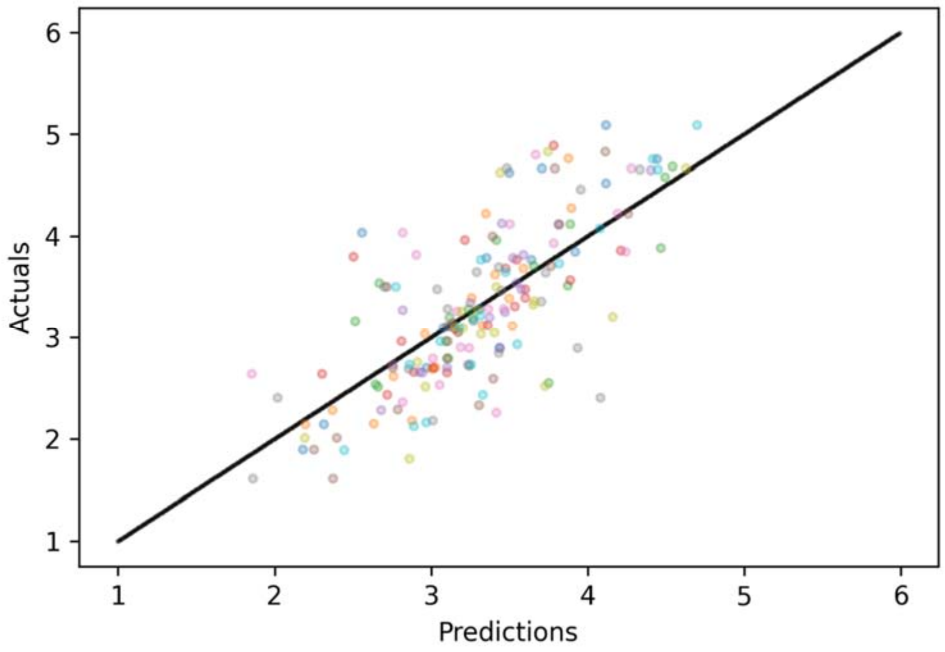


Figure 12: Type F model - Scatter plot on excluded test set predictions

5.3 Type B model performance

Type B model, as mentioned in the previous chapter, is a model trained on a subset, excluding Black ethnicity individuals. Ten models of this type were trained, each with a different test set.

The Pearson correlation for the validation set is high (Table 7) with an average value of 0.7474 and mean square error is at 0.3677 which means that the model can predict attractiveness with ±0.37 error. Additionally, the prediction performance results for the test set are slightly lower with a moderate correlation.

ModelNo	Validation set		Test set	
	Pearson correlation	Mean absolute error	Pearson correlation	Mean absolute error
1	0.8008635274	0.2950863208	0.5949393526	0.5697499071
2	0.713284108	0.377150961	0.7711832414	0.3850495579
3	0.7286109359	0.3704864425	0.684635278	0.3639285332
4	0.7591041935	0.3991122531	0.7281237818	0.397985391
5	0.6878293411	0.3960688389	0.7871874495	0.3802769484
6	0.7453820828	0.3864011331	0.6593320446	0.4563800653
7	0.7391158474	0.4256376284	0.7008715354	0.443287309
8	0.7703685791	0.3207339004	0.7218015759	0.4476571583
9	0.7087185652	0.3668553577	0.701470454	0.4154738662
10	0.8202630286	0.3397371093	0.7408565353	0.4157496919
Average	0.7473540209	0.3677269945	0.7090401249	0.4275538428

Table 7: Type B model – Validation/test set performance

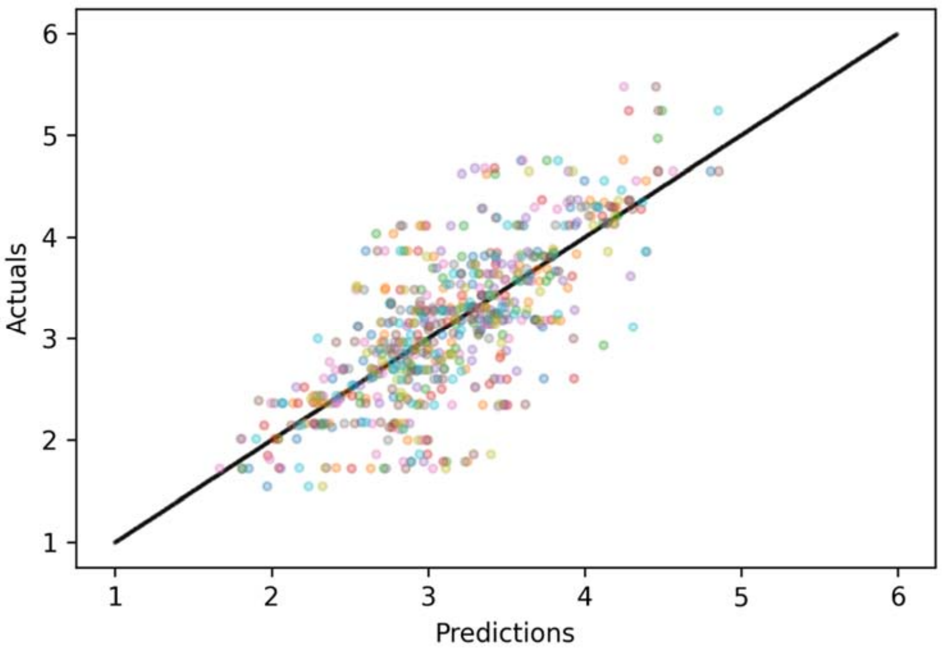


Figure 13: Type B model - Scatter plot on validation set predictions

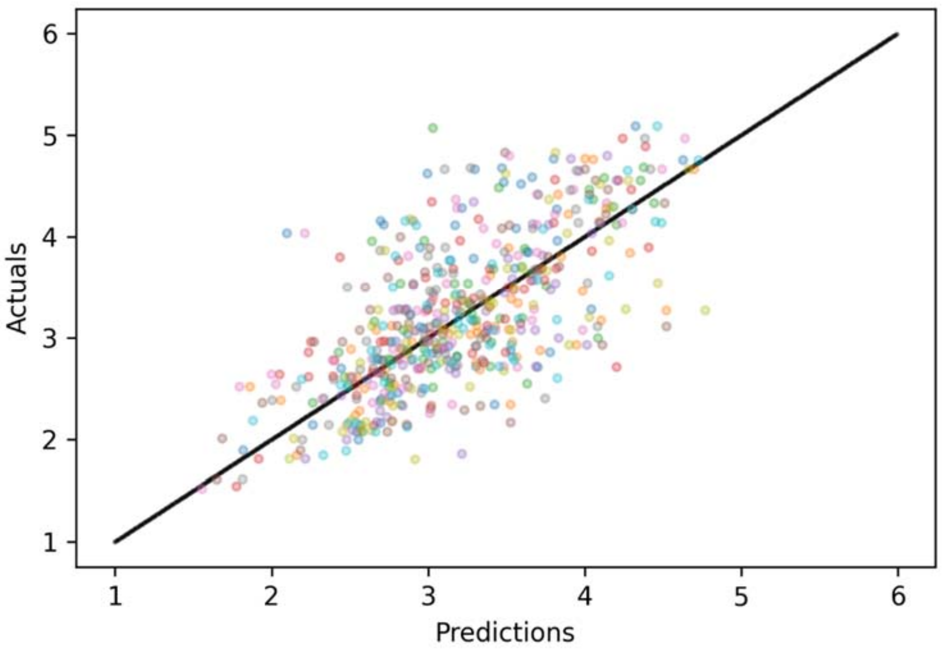


Figure 14: Type B model - Scatter plot on test set predictions

Regarding predictions for the excluded group on both the Type B and Type F models, from the results displayed in Table 8, it is clear that the predictions on the Type B model had a low degree of correlation as an average (0.445) and high error values (0.70). On the other hand, the predictions of the same set of images on the Type F model are clearly more accurate with a moderate average correlation of 0.6494 and 0.45 average error.

ModelNo	Type B - Excluded Test set		Type F - Excluded Test set	
	Pearson correlation	Mean absolute error	Pearson correlation	Mean absolute error
1	0.6657600103	0.581662987	0.5133249718	0.511128949
2	0.6041490478	0.7309349693	0.5632838208	0.4899690109
3	0.4978413884	0.69752481	0.5735912699	0.4511501922
4	0.6838238944	0.6202281324	0.8250189315	0.3279072557
5	0.5564948394	0.5358872068	0.7221493098	0.4702108203
6	0.4712052143	0.9685865225	0.739286221	0.4380403242
7	0.04388510445	0.8603541111	0.4447802713	0.5059208567
8	0.4548847431	0.7147458721	0.8340853034	0.4478253791
9	0.3782038599	0.5505686086	0.7260536075	0.3396844079
10	0.09328420133	0.7731214266	0.5523518568	0.5445585749
Average	0.4449532303	0.7033614646	0.6493925564	0.4526395771

Table 8: Type B model - Excluded test set (Black) - Type B/F comparison

The metrics above are depicted in Figures 16, 17. Grouping of values in model B looks shifted to the left, meaning that images on the excluded set scored consistently lower that the actual attractiveness values. On the other hand, actuals/predictions on Type F models show a good linear relationship.

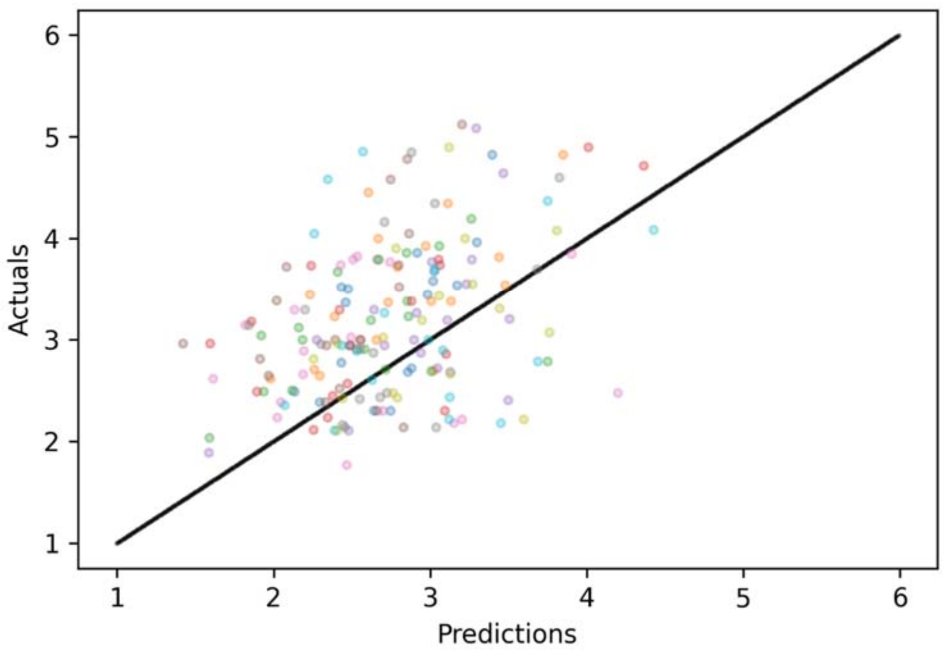


Figure 15: Type B model - Scatter plot on excluded test set predictions

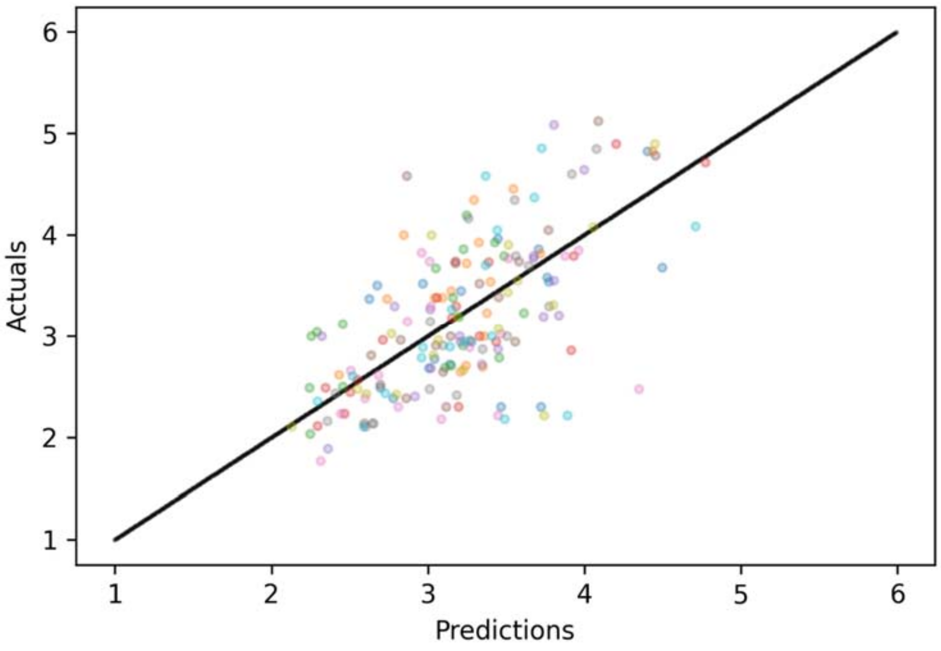


Figure 16: Type F model - Scatter plot on excluded test set predictions

5.4 Type A model performance

Type A model, as mentioned in the previous chapter, is a model trained on a subset, excluding Asian ethnicity individuals. Ten models of this type were trained, each with a different test set.

The Pearson correlation for the validation set is high (Table 9) with an average value of 0.7259 and mean square error is at 0.3747 which means that the model can predict attractiveness with ±0.37 error. The prediction performance results for the test set are lower with a moderate correlation (0.6330) and higher average error (0.45).

ModelNo	Validation set		Test set	
	Pearson correlation	Mean absolute error	Pearson correlation	Mean absolute error
1	0.7790136542	0.3363166791	0.5509466897	0.5344106943
2	0.7375181364	0.3592557339	0.6079663659	0.4556312842
3	0.7764488864	0.3511565253	0.6780335958	0.4140937201
4	0.7313311696	0.3807832228	0.6860787658	0.4243501828
5	0.6293286617	0.4197092451	0.6823135146	0.4327685774
6	0.7079445649	0.3921269535	0.7315948737	0.4157022699
7	0.7259164432	0.351757423	0.466441419	0.5062755364
8	0.6859025908	0.3674618801	0.6321781777	0.4650022034
9	0.7252427747	0.405608064	0.6220297386	0.4415359808
10	0.7605881825	0.3830803068	0.6726402782	0.4802457303
Average	0.7259235064	0.3747256034	0.6330223419	0.457001618

Table 9: Type A model – Validation/test set performance

The lower performance for the test sets on model Type A is observable on the scatter plot (Figure 19), where the values are slightly shifted to the left in comparison with the validation set plot (Figure 18). However, both plots display a good linear relationship between actual and predicted values.

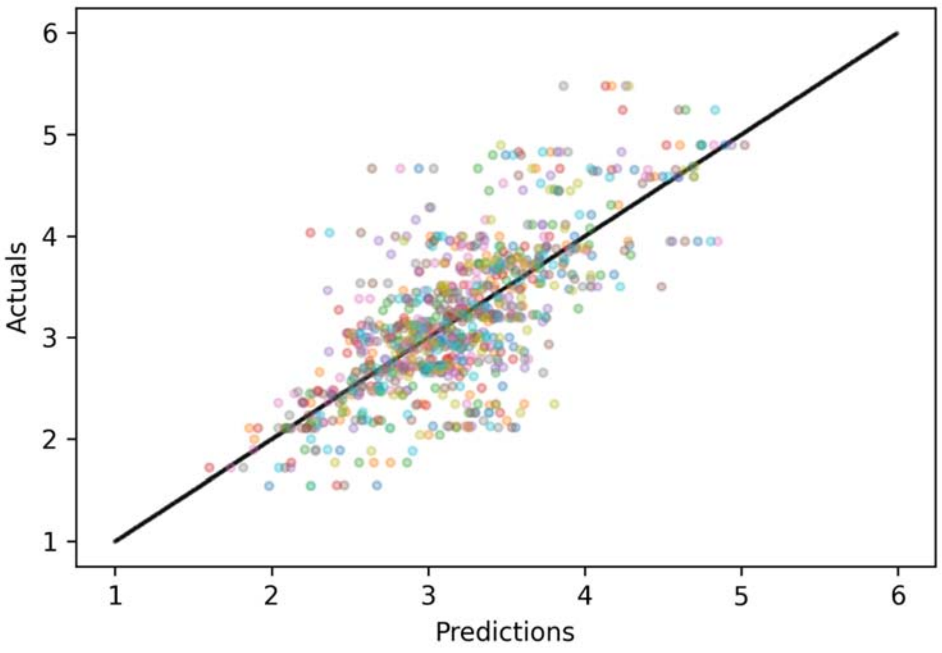


Figure 17: Type A model - Scatter plot on validation set predictions

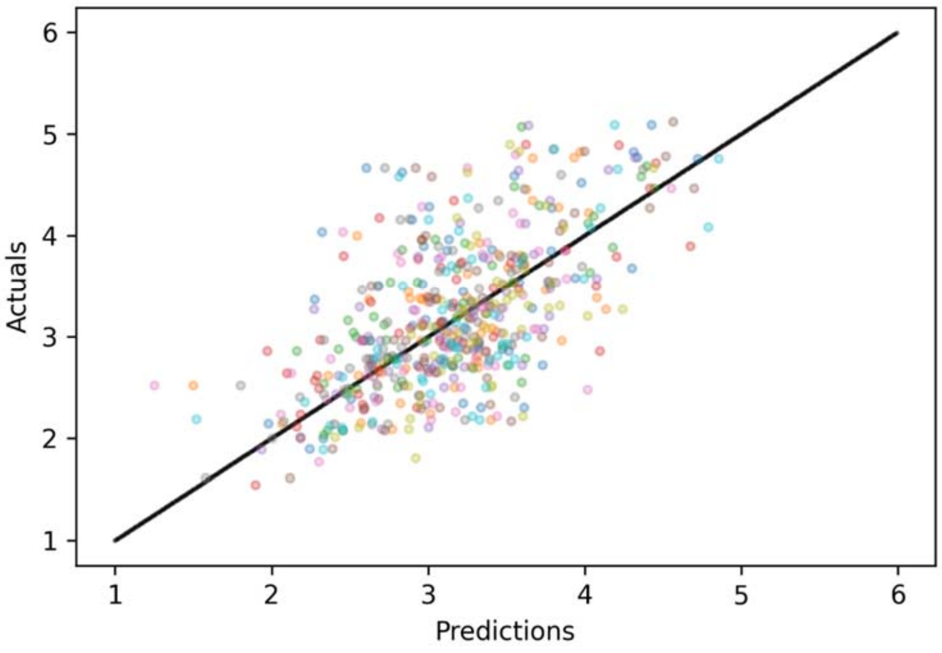


Figure 18: Type A model - Scatter plot on test set predictions

Regarding predictions for the excluded group on both the Type A and Type F models, the results are inconclusive (Table 10). Excluding the Asian group from the training data didn’t seem to have an impact on prediction performance. This is also visible in the respective scatter plots (Figures 20, 21), where in both cases the grouping of values is similar, with a strong linear correlation. This could be explained by the lower number of images for the Asian group compared to White or Black, and also by facial characteristics that might make the group less susceptible to biased training data.

ModelNo	Type A - Excluded Test set		Type F - Excluded Test set	
	Pearson correlation	Mean absolute error	Pearson correlation	Mean absolute error
1	0.5605758751	0.6185375805	0.5144375768	0.6069098648
2	0.8423549584	0.32380907	0.8814362951	0.2716675202
3	0.808978486	0.2814593368	0.7713284969	0.3784251143
4	0.8635573137	0.3603317123	0.8268615343	0.3268554097
5	0.8112934686	0.468606279	0.8335959375	0.4554526583
6	0.5079950046	0.4997010019	0.5614072468	0.4095774262
7	0.8150798878	0.3954848796	0.8713157295	0.3719708483
8	0.6914193048	0.5476302576	0.7470348729	0.4736194613
9	0.8061502187	0.45549438	0.8538980527	0.3783307099
10	0.6663721805	0.4896766098	0.6798833408	0.5517110412
Average	0.7373776698	0.4440731107	0.7541199083	0.4224520054

Table 10: Type A model - Excluded test set (Asian) - Type A/F comparison

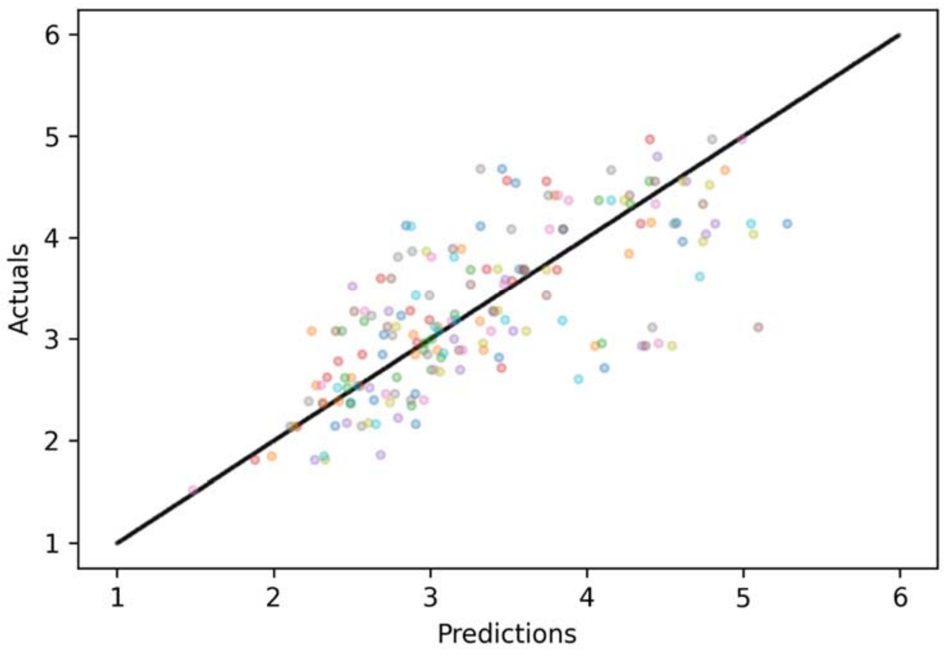


Figure 19: Type A model - Scatter plot on excluded test set predictions

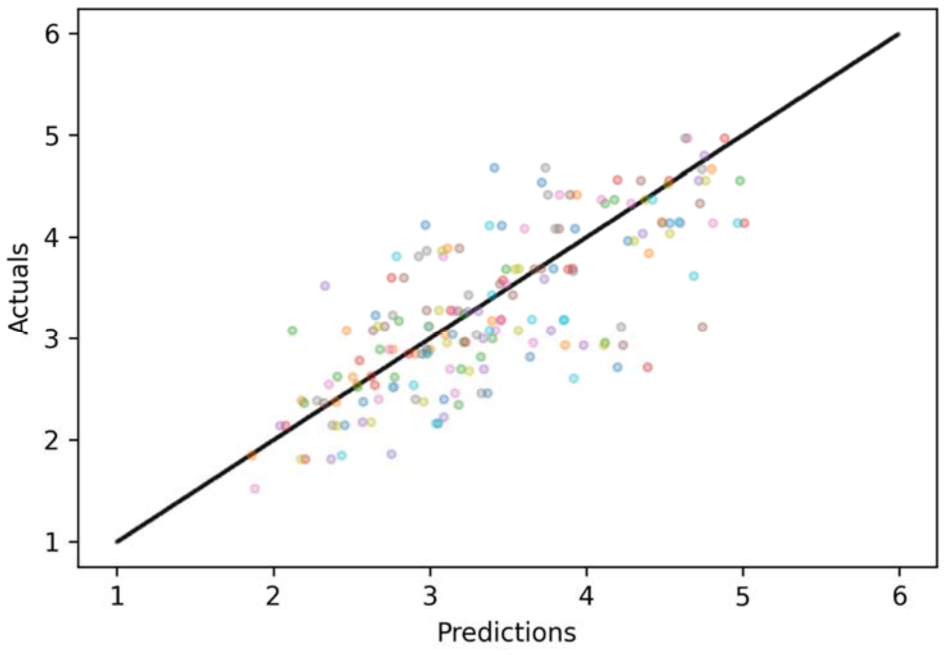


Figure 20: Type F model - Scatter plot on excluded test set predictions

5.5 Type L model performance

Type L model, as mentioned in the previous chapter, is a model trained on a subset, excluding Latino ethnicity individuals. Ten models of this type were trained, each with a different test set.

The Pearson correlation for the validation set is moderate (Table 11) with an average value of 0.6754 and mean square error is at 0.3908 which means that the model can predict attractiveness with ±0.39 error. Additionally, the prediction performance results for the test set are slightly higher with a stronger correlation albeit higher error.

ModelNo	**Validation set**		**Test set**	
	Pearson correlation	**Mean absolute error**	**Pearson correlation**	**Mean absolute error**
1	0.7160112536	0.3323401808	0.6046989879	0.5262706637
2	0.6482604911	0.4307984866	0.785233173	0.374705388
3	0.6193688013	0.4362318242	0.7689628212	0.3250118931
4	0.6511514548	0.4426543262	0.7843580066	0.3414114139
5	0.6334481472	0.3984320881	0.7883489001	0.3898562602
6	0.6641791613	0.3967074845	0.7370693142	0.4167473067
7	0.7747588464	0.3363746343	0.6525105639	0.5077570679
8	0.7096553327	0.3275324308	0.7532955825	0.4602677015
9	0.7030647736	0.3949674393	0.7672953648	0.3709079484
10	0.6338765951	0.4122167136	0.6945963649	0.4732141368
Average	0.6753774857	0.3908255608	0.7336369079	0.418614978

Table 11: Type L model – Validation/test set performance

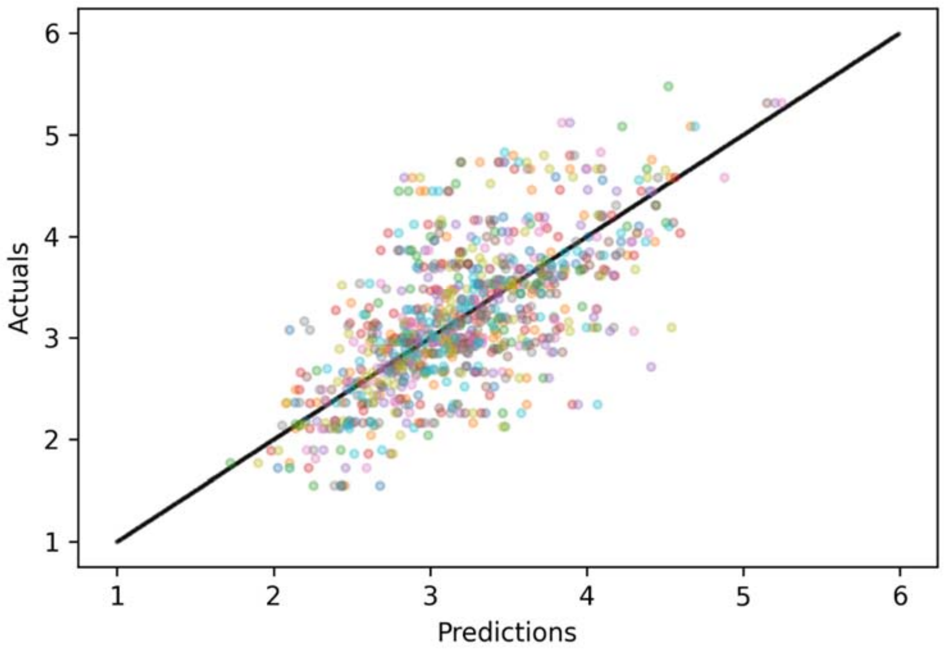


Figure 21: Type L model - Scatter plot on validation set predictions

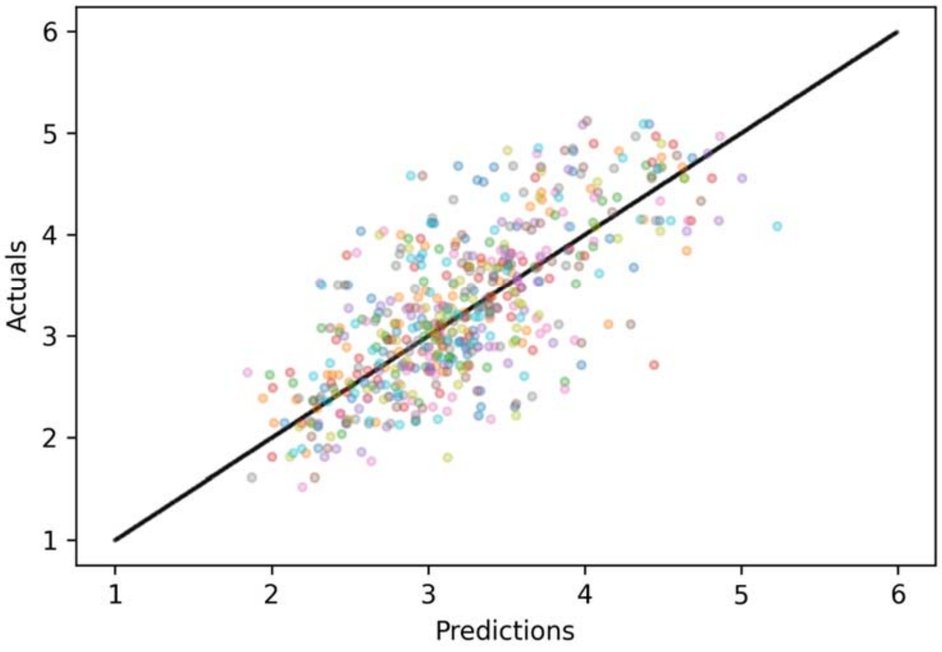


Figure 22: Type L model - Scatter plot on test set predictions

Regarding predictions for the excluded group on both the Type L and Type F models, the results are inconclusive (Table 12). Like with the Asian group, excluding the Latino group from the training data didn't seem to have an impact on prediction performance and this is also visible in the respective scatter plots (Figures 24, 25). This could be explained by the lower number of images for the Latino group compared to White or Black, and also by facial characteristics that might make the group less susceptible to bias.

ModelNo	Type L - Excluded Test set		Type F - Excluded Test set	
	Pearson correlation	Mean absolute error	Pearson correlation	Mean absolute error
1	0.5405781953	0.6270277249	0.505981102	0.6295482184
2	0.5329833733	0.5167786597	0.5402907682	0.5350055754
3	0.5940382685	0.5707245171	0.646839298	0.4684390327
4	0.5751808614	0.558535802	0.7501187755	0.4116366077
5	0.69370438	0.4426014272	0.8032424362	0.3914614212
6	0.7809101365	0.3385424	0.7122318118	0.4143782492
7	0.5520114041	0.4704808255	0.5267299518	0.4738045776
8	0.7250212586	0.3406508664	0.6679593967	0.3623860375
9	0.5612776237	0.4133750078	0.7188023401	0.496044617
10	0.8115515198	0.3054898364	0.4183414789	0.4566858678
Average	0.6367257021	0.4584207067	0.6290537359	0.4639390205

Table 12: : Type L model - Excluded test set (Latino) - Type L/F comparison

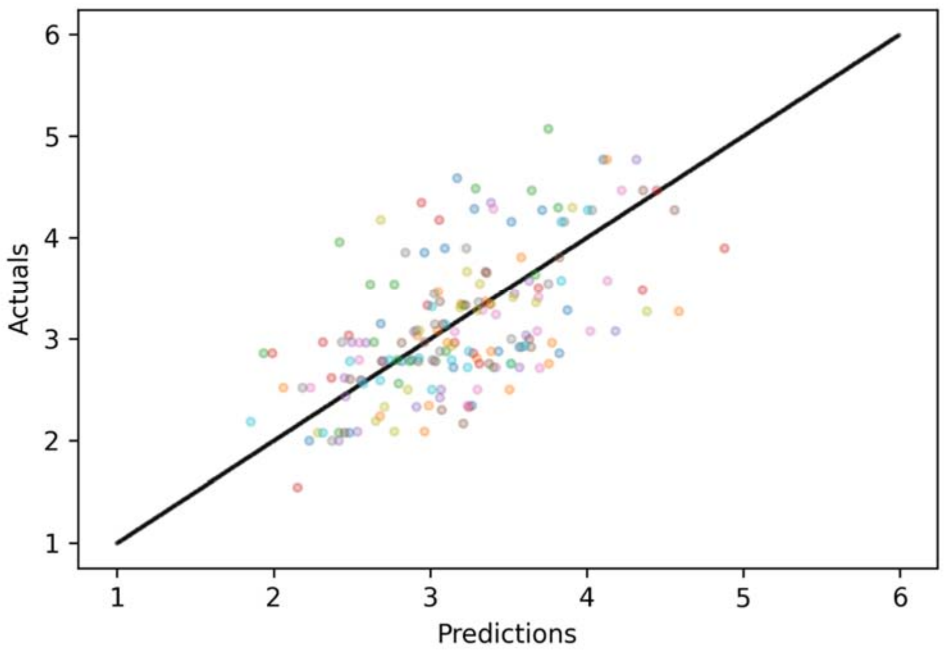


Figure 23: Type L model - Scatter plot on excluded test set predictions

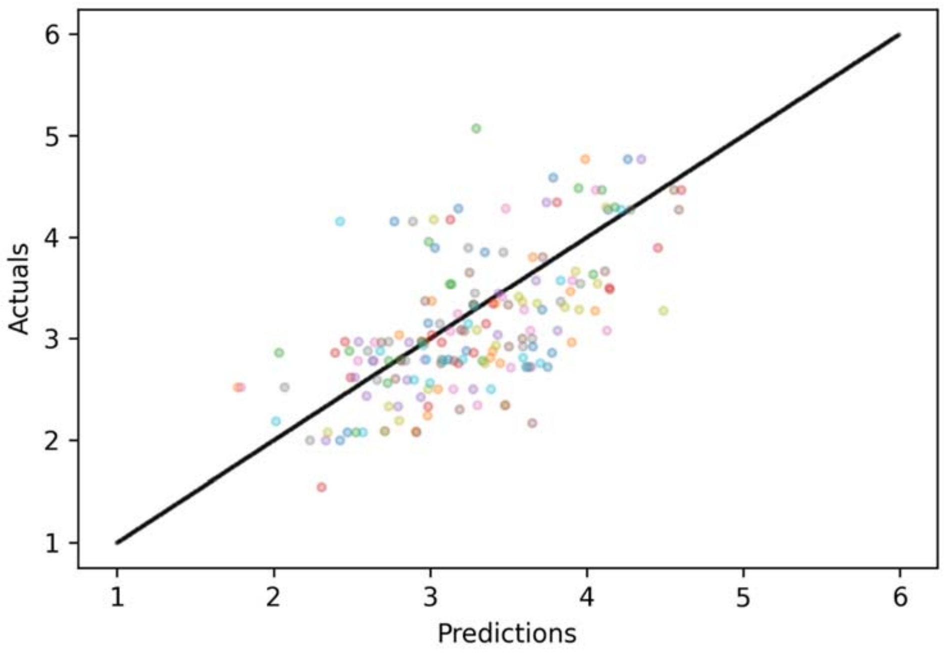


Figure 24: Type F model - Scatter plot on excluded test set predictions

6. CONCLUSION - FUTURE WORK

The problem addressed in this thesis was to implement an attractiveness prediction model and evaluate its performance along with the effects of bias introduction. This was achieved by implementing a convolutional neural network based on the VGG-Face model and training it on data from the Chicago Face Database. Five model categories were created based on the training data and ten different test sets were randomly generated, resulting in fifty models.

The training process was consistent across all models, with the only differentiating factors being the training and test data. Training time varied between 25 and 120 epochs as an early stop technique was used, meaning that the process stopped after 50 rounds of no validation accuracy decrease.

Despite the relatively small dataset available, the performance results were acceptable for all cases. Test set results for all model types show good correlation of predicted versus actual values and in most cases are close with the respective validation results. That means that the models have the ability to generalize and accurately predict attractiveness on images not included in the training process.

Regarding the introduction of bias in the training data, excluding White or Black ethnicities from the data showed a clear effect on prediction results, with low correlation and error being evidently higher. In both cases the scatter plots showed a shift to the left, revealing a trend of predicted attractiveness values being lower than the actual scores. This supports the case that when data collection is flawed, without diversity or even a complete group exclusion, the resulting model would potentially output compromised results. For Asian and Latino ethnicities, the results were different, with the exclusion of the groups from training data not affecting the prediction performance. This could be due to specific facial traits or the amount of data representing these two groups.

6.1 Future work

As mentioned above, the results for the Asian and Latino ethnicities weren't in in line with the other two groups. It would be interesting to examine the reasons behind those results and what makes the two groups apparently less susceptive to bias introduction.

The filter visualization process provided some insight on the convolution process, but couldn't provide the factors that define a low or high attractiveness score. Future research could focus on the visualization process and define the patterns and facial features that define attractiveness.

Another future approach would be to gather a bigger dataset. The Chicago Face Database consists of diverse, quality data, but the number of images is relatively low. Additionally, excluding data from the initial dataset in order to introduce bias definitely didn't help with this problem. A bigger dataset would produce more reliable results and provide a bigger tolerance for data exclusion. The current models were trained on images created in studio, with consistent lighting conditions, background and model expressions. A bigger more diverse dataset would also help to create a more capable model, able to output predictions on images with a variety of backgrounds and expressions.

REFERENCES

[1] Omkar M. Parkhi et al., "Deep Face Recognition", 2015.

[2] Buolamwini, J., "Gender Shades: Intersectional Phenotypic and Demographic Evaluation of Face Datasets and Gender Classifiers", MIT Master's Thesis, 2017.

[3] E. P. Prokopakis, I. M. Vlastos, V. A. Picavet et al., “The golden ratio in facial symmetry,” Rhinology journal, vol. 51, no. 1, pp. 18–21, 2013.

[4] M. E. Rhazi, A. Zarghili, A. Majda et al., “Facial beauty analysis by age and gender,” International Journal of Intelligent Systems Technologies and Applications, vol. 18, no. 1, pp. 179–203, 2019

[5] K. Schmid, D. Marx, and A. Samal, “Computation of a face attractiveness index based on neoclassical canons, symmetry, and golden ratios,” Pattern Recognition, vol. 41, no. 8, pp. 2710–2717, 2008.

[6] R. Thornhill and S. W. Gangestad, “Human facial beauty,” Human Nature, vol. 4, no. 3, pp. 237–269, 1993.

[7] K. Grammer and R. Thornhill, “Human (Homo sapiens) facial attractiveness and sexual selection: the role of symmetry and averageness,” Journal of Comparative Psychology, vol. 108, no. 3, pp. 233–242, 1994.

[8] A. Kagian, G. Dror, T. Leyvand et al., “A humanlike predictor of facial attractiveness,” in Advances in Neural Information Processing Systems, pp. 649–656, MIT Press, Massachusetts, MA, USA, 2007.

[9] W. Wei, E. S. Ho, K. D. McCay et al., “Assessing facial symmetry and attractiveness using augmented reality,” Pattern Analysis and Applications, vol. 28, pp. 1–7, 2021.

[10] Hong Y-J, Nam GP, Choi H, Cho J, Kim I-J. A Novel Framework for Assessing Facial Attractiveness Based on Facial Proportions. Symmetry. 2017

[11] L. Lin, L. Liang, L. Jin et al., “Attribute-aware convolutional neural networks for facial beauty prediction,” in Proceedings of the 2019 International Joint Conference on Artificial Intelligence, pp. 847–853, Macao, China, August 2019.

[12] Duorui Xie, Lingyu Liang, Lianwen Jin, Jie Xu, Mengru Li, "SCUT-FBP: A Benchmark Dataset for Facial Beauty Perception", 2015.

[13] Q. Xiao, Y. Wu, D. Wang et al., “Beauty3DFaceNet: deep geometry and texture fusion for 3D facial attractiveness prediction,” Computers & Graphics, vol. 98, pp. 11–18, 2021.

[14] Zhai, Yikui et al. “BeautyNet: Joint Multiscale CNN and Transfer Learning Method for Unconstrained Facial Beauty Prediction.” Computational intelligence and neuroscience vol. 2019 1910624. 28 Jan. 2019, doi:10.1155/2019/1910624.

[15] Cao, Kerang, Kwang-nam Choi, Hoekyung Jung, and Lini Duan. 2020. "Deep Learning for Facial Beauty Prediction" Information 11, no. 8: 391.

[16] Debbie S. Ma, Joshua Correll and Bernd Wittenbrink, "The Chicago face database: A free stimulus set of faces and norming data", 2015.

[17] C. Nebauer et al., "Evaluation of convolutional neural networks for visual recognition", 2018.

[18] Keiron O'Shea, Ryan Nash, "An Introduction to Convolutional Neural Networks", 2015.

[19] Wazir Muhammad et al., "An Introduction to Deep Convolutional Neural Networks With Keras", 2020.

[20] Reny Jose et al., "A Convolutional Neural Network (CNN) Approach to Detect Face Using Tensorflow and Keras", 2020.

[21] [Online] Dr. Varshita Sher, "Beginner’s guide to building Convolutional Neural Networks using TensorFlow’s Keras API in Python", towardsdatascience.com, 2020

Available: https://towardsdatascience.com/beginners-guide-to-building-convolutional-neural-networks-using-tensorflow-s-keras-api-in-python-6e8035e28238

[22] [Online] Himanshu Rawlani, "Visual Interpretability for Convolutional Neural Networks", towardsdatascience.com, 2018

Available: https://towardsdatascience.com/visual-interpretability-for-convolutional-neural-networks-2453856210ce

[23] [Online] Will Koehrsen, "Transfer Learning with Convolutional Neural Networks in PyTorch" , towardsdatascience.com, 2018

Available: https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce