Face2BMI Paper Report

Implementation Notebook Links

For Kaggle Notebooks:
Go to the Link -> Click on Copy and Edit -> Click Run All
For Colab Notebooks:
Click Runtime -> Click Run all

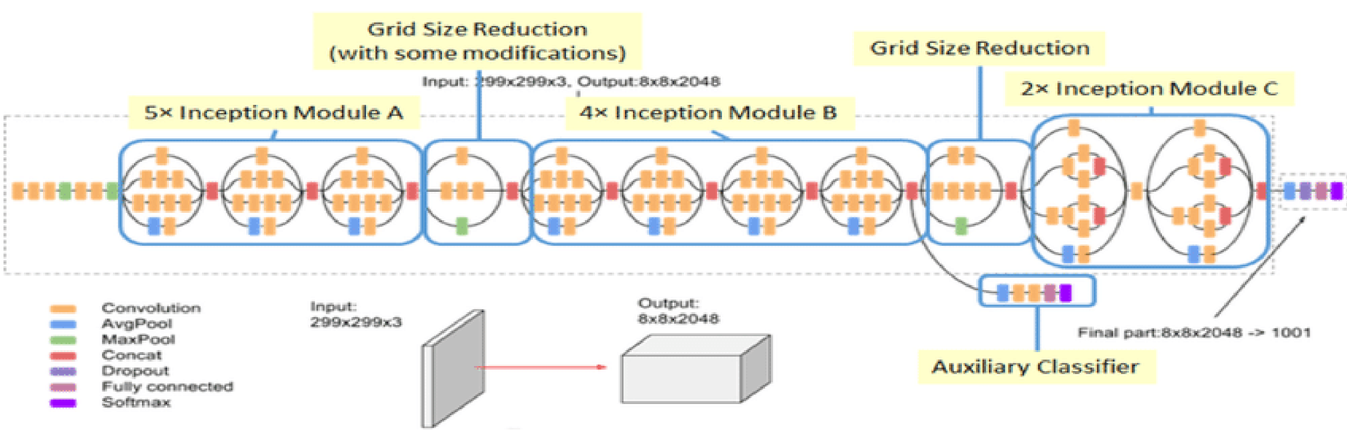
1. Notebook used for Training
2. Analyzing the Performance of models on Training Dataset
3. Analyzing the Performance of models on Prison Dataset
4. Analyzing the Performance of models on VIP_attribute Dataset

Important Links

1. https://www.tensorflow.org/addons

Pre-trained Models and their architecture

1. **Inception V3**



Source: https://www.researchgate.net/publication/339296714_IMCFN_Image-based_Malware_Classification_using_Fine-tuned_Convolutional_Neural_Network_Architecture

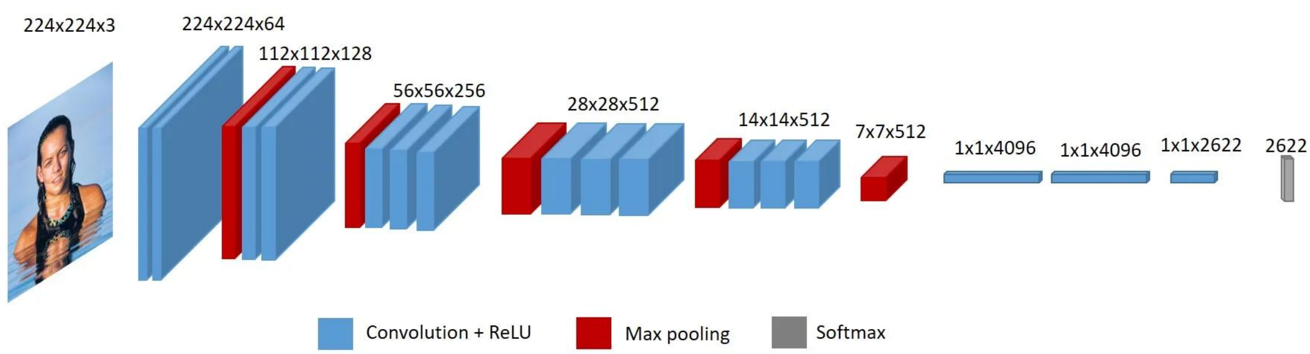
type	patch size/stride or remarks	input size
conv	$3\times3/2$	$299\times299\times3$
conv	$3\times3/1$	$149\times149\times32$
conv padded	$3\times3/1$	$147\times147\times32$
pool	$3\times3/2$	$147\times147\times64$
conv	$3\times3/1$	$73\times73\times64$
conv	$3\times3/2$	$71\times71\times80$
conv	$3\times3/1$	$35\times35\times192$
$3\times$Inception	As in figure 5	$35\times35\times288$
$5\times$Inception	As in figure 6	$17\times17\times768$
$2\times$Inception	As in figure 7	$8\times8\times1280$
pool	8×8	$8\times8\times2048$
linear	logits	$1\times1\times2048$
softmax	classifier	$1\times1\times1000$

Source: https://pytorch.org/hub/pytorch_vision_inception_v3/

2. **VGG-Faces**

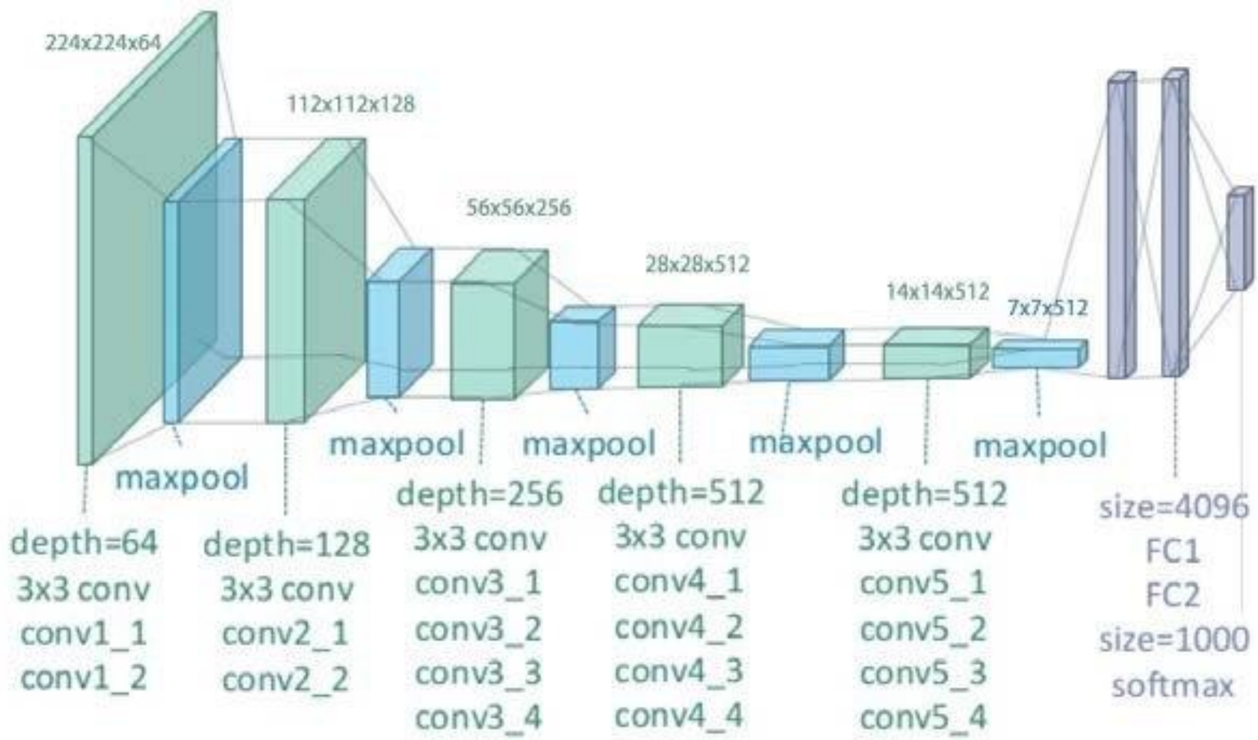
layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
type	input	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv
name	–	conv1_1	relu1_1	conv1_2	relu1_2	pool1	conv2_1	relu2_1	conv2_2	relu2_2	pool2	conv3_1	relu3_1	conv3_2	relu3_2	conv3_3	relu3_3	pool3	conv4_1
support	–	3	1	3	1	2	3	1	3	1	2	3	1	3	1	3	1	2	3
filt dim	–	3	–	64	–	–	64	–	128	–	–	128	–	256	–	256	–	–	256
num filts	–	64	–	64	–	–	128	–	128	–	–	256	–	256	–	256	–	–	512
stride	–	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2	1
pad	–	1	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1

layer	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
type	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	softmx
name	relu4_1	conv4_2	relu4_2	conv4_3	relu4_3	pool4	conv5_1	relu5_1	conv5_2	relu5_2	conv5_3	relu5_3	pool5	fc6	relu6	fc7	relu7	fc8	prob
support	1	3	1	3	1	2	3	1	3	1	3	1	2	7	1	1	1	1	1
filt dim	–	512	–	512	–	–	512	–	512	–	512	–	–	512	–	4096	–	4096	–
num filts	–	512	–	512	–	–	512	–	512	–	512	–	–	4096	–	4096	–	2622	–
stride	1	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

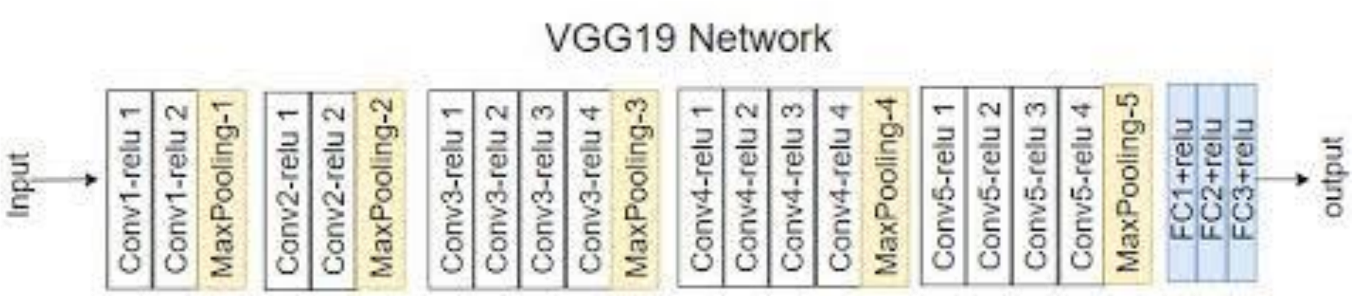


Source: https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/

3. VGG-19

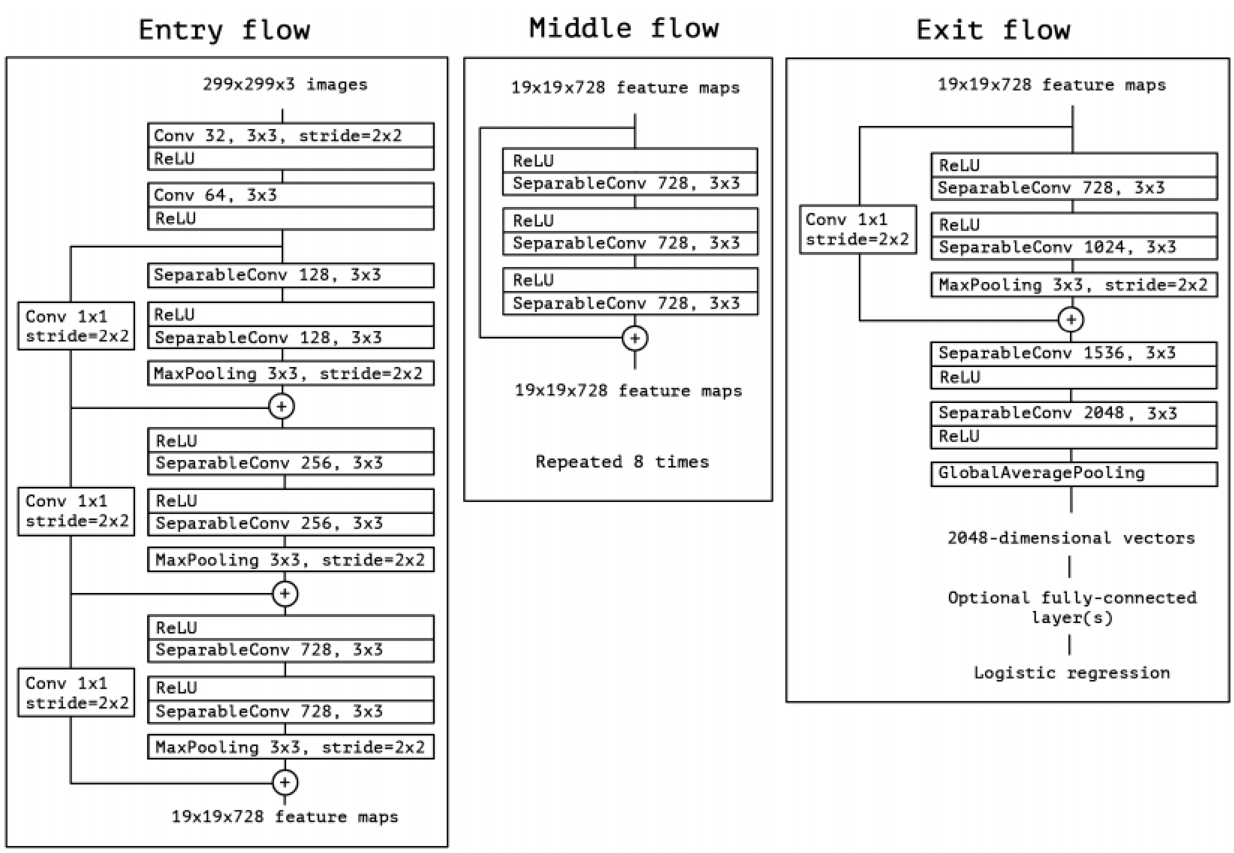


Source:
https://www.researchgate.net/figure/Illustration-of-the-network-architecture-of-VGG-19-model-conv-means-convolution-FC-means_fig2_325137356



Source:
https://www.osapublishing.org/DirectPDFAccess/EDF3B596-4F68-430A-B0E0E650BB4C758E_426045/oe-28-2-2511.pdf?da=1&id=426045&seq=0&mobile=no

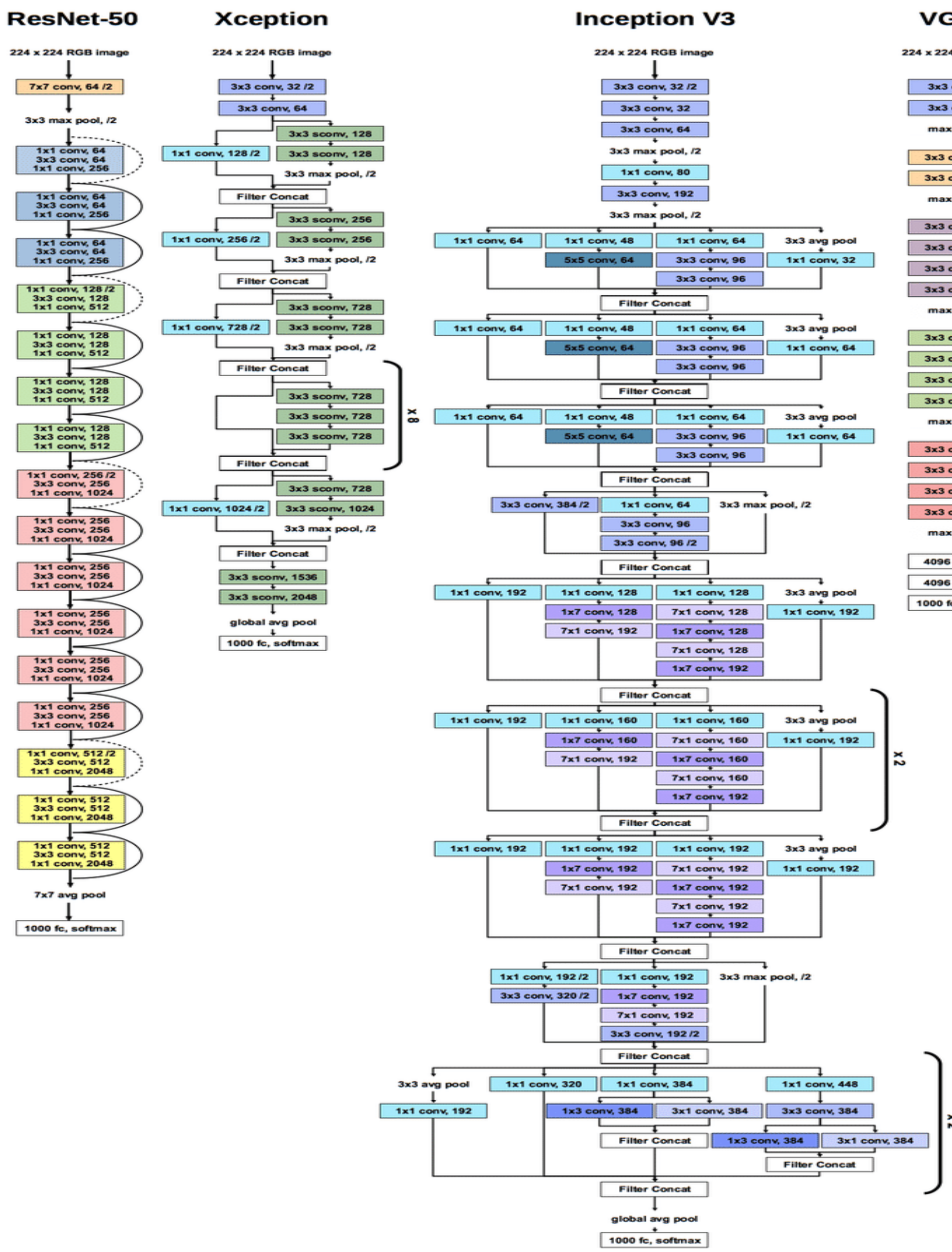
4. Xception



Source: https://towardsdatascience.com/xception-from-scratch-using-tensorflow-even-better-than-inception-940fb231ced9

Source: https://maelfabien.github.io/deeplearning/xception/#ii-in-keras

5. Comparison



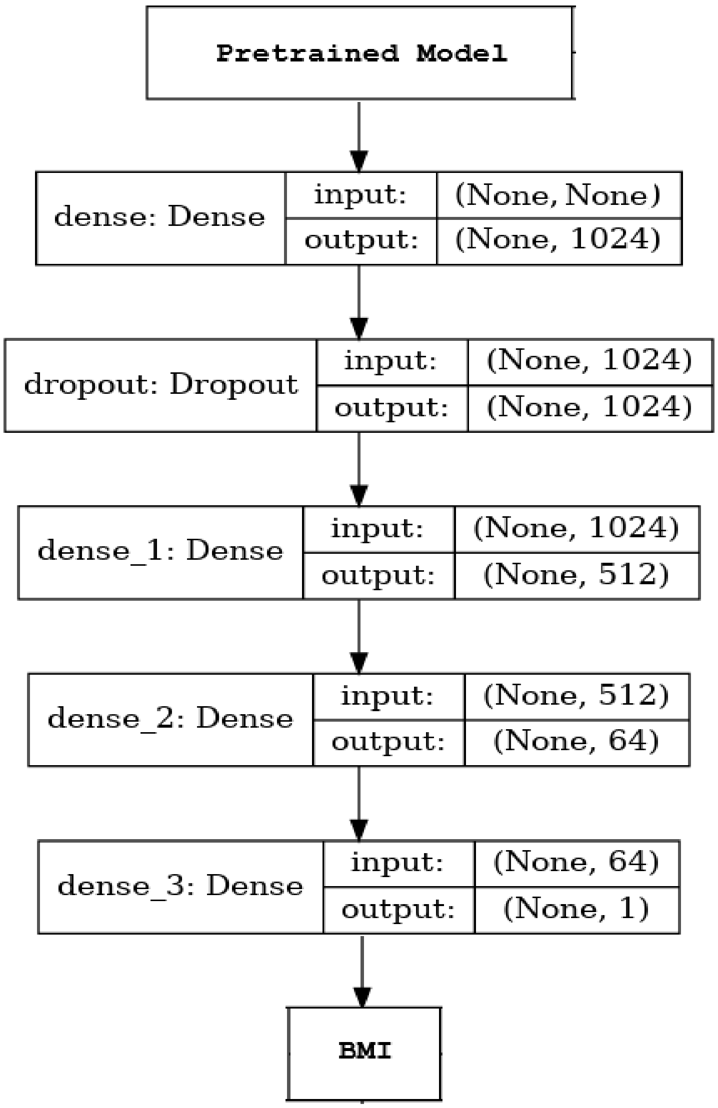
Pre-Trained Models Information

Model	Description	No. of Layers
VGG-Faces	The VGG-Face model is developed by the researchers of the Visual Geometry Group (VGG) at Oxford. This model uses the VGGFace2 dataset that consists of 3.31 million facial images of 9131 subjects. It was created with the primary intention of training robust face recognition models.	50 (Resnet50 Architecture)
Xception	Feature extraction base of the Xception model is formed by 36 convolutional layers. It is an extension of the Inception model architecture where it uses depth wise Separable Convolutions in place of the standard inception modules. It was also trained on ImageNet Database and achieved a Top-1 accuracy score of 0.79 and a Top-5 accuracy score of 0.94 with around 22 million parameters.	36 Conv Layers
Inception_v3	Inception-v3 is a deep convolutional neural network. It is the 3rd edition model of the Inception CNN developed by Google. This model is trained on more than a million images from the popular ImageNet database. It gave a Top-1 accuracy score of 0.779 and a Top-5 accuracy score of 0.937 with around 24 Million parameters only and thus computationally effective when compared to other models.	48
VGG19	VGG-19 model is a deep convolutional neural network, a successor of the VGG-16 model and developed by the researchers of the Visual Geometry Group. Like the Inception-v3 model, it is trained on the popular ImageNet database. It achieved a Top-1 accuracy score of 0.752 and a Top-5 accuracy score of 0.925 with around 143 million parameters.	19

Implementation Details

We used the same set of fully connected layers as top of all 4 pre-trained models.

Fully Connected Layers for all the Models



We used a dropout layer with value 0.5 so 50% of neurons in that layer do not take part in training to avoid overfitting.We used 2 types of activation functions namely Rectified Linear Unit (relu) and Gaussian Error Linear Units (gelu). Gelu is a relatively new activation function because it tends to work better when there is more noise in the data.
Generally only one optimizer is used in the model for training. Convolutional Neural networks learn low level features at initial layers and then abstracting on this low level feature, as we move through layers it learns high level complex features. Mostly all the computer vision tasks will have similar low level features and regarding to task high level

features can change. Due to which it makes more sense to train end layers more and initial layers less, thus discriminative learning.
In our application we used 4 different Adam optimizers, each with a different learning rate applied to the last 35 layers.

Last 35 - 25 layers → ADAM(lr = 1e-8)
Last 25 - 15 layers → ADAM(lr = 1e-7)
Last 15 - 5 layers → ADAM(lr = 1e-6)
Last 5 layers → ADAM(lr = 1e-5)

As VGG19 has relatively few layers, we used 2 Adam optimizers only.

During training the model if the validation loss does not improve after 5 epochs then the training will be terminated. We used the EarlyStopping callback provided by Keras library to implement it.

Why did we use these models ?

VGGFace	**VGG19**	**Inception-v3**	**Xception**
It is used for the task of one shot face verification. Given that it’s been already trained on facial images it makes sense to transfer that knowledge to our problem statement using transfer learning.	It is one of the earlier Convolutional Networks, to check the capability of effectiveness of this network we decided to use VGG19.	It was developed with the primary intention of achieving high accuracy and low computational cost so that it can be used on devices with low processing power as well.	It has the same number of parameters as Inception-v3. It combines the output of all convolutional filters and performs 1x1 depthwise convolution to capture output.

Results obtained in terms of Mean Absolute Error(MAE) for Overall, Male and Female

1. **ILLINOIS TRAINING DATASET**

	MAE-Overall	MAE-Male	MAE-Female
VGGFaces	3.63	3.61	3.93
Inception-v3	2.86	2.83	3.59
Xception	2.82	2.79	3.54
VGG-19	2.97	2.89	4.04

2. **VIP_Attributes DATASET**

	MAE-Overall	MAE-Male	MAE-Female
VGGFaces	3.42	4.16	2.68
Inception-v3	3.1	3.04	3.17
Xception	3.91	2.78	5.03
VGG-19	3.2	3.33	3.06

3. **Arrest Records DATASET**

	MAE-Overall	MAE-Male	MAE-Female
VGGFaces	3.73	3.35	5.09
Inception-v3	3.87	3.57	4.68
Xception	3.93	3.56	5.11
VGG-19	3.79	3.75	3.99

Loss Function and Evaluation Metric

For our research study, we used Mean Squared Error (MSE) as our regression loss function. It is calculated by computing the summation of squared differences of actual value and the predicted value, divided by the total number of samples. Because of squaring the difference, this loss function penalizes the model more heavily.

$$MSE = (\frac{1}{n}) \sum_{i=1}^{n} (y_i - x_i)^2$$

We used Mean Absolute Error (MAE) to evaluate our model's performance. It is calculated by computing the summation of absolute values of differences of actual value and predicted value , divided by the total number of samples.

$$MAE = (\frac{1}{n}) \sum_{i=1}^{n} |y_i - x_i|$$

Gelu Activation Function →

Computes gaussian error linear:

$$\text{gelu}(x) = x\Phi(x),$$

where

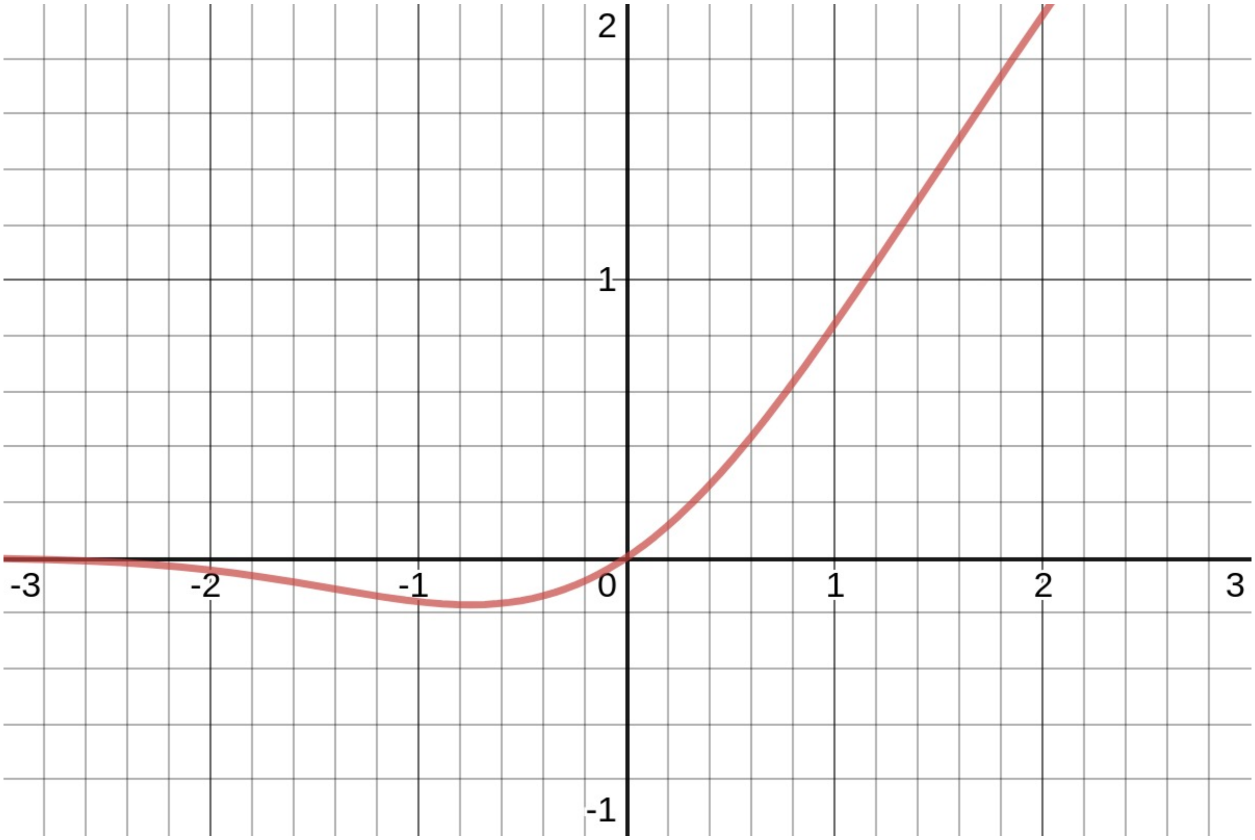
$$\Phi(x) = \frac{1}{2}\left[1 + \text{erf}(\frac{x}{\sqrt{2}})\right] \$$$

when `approximate` is `False`; or

$$\Phi(x) = \frac{x}{2}\left[1 + \tanh(\sqrt{\frac{2}{\pi}} \cdot (x + 0.044715 \cdot x^3))\right]$$

when `approximate` is `True`.

Graph →



Batch Normalization Layer

Batch normalization is a layer that allows every layer of the network to do learning more independently. It is used to normalize the output of the previous layers. The activations scale the input layer in normalization. Using batch normalization learning becomes efficient also it can be used as regularization to avoid overfitting of the model. The layer is added to the sequential model to standardize the input or the outputs. It can be used at several points in between the layers of the model. It is often placed just after defining the sequential model and after the convolution and pooling layers.