Face Recognition System for analysis of customer satisfaction and emotion

by Anushka Patil

Submission date: 27-Dec-2020 11:18AM (UTC+0400)
Submission ID: 1481431067
File name: thesis_report_turnitin_check.pdf (2.73M)
Word count: 5327
Character count: 27749

Chapter 1
INTRODUCTION

1.1. GENERAL INTRODUCTION TO THE THESIS AREA

Facial expression analysis has evolved over the time, from recognizing expressions in simple images to extending the same functionality in complex videos. One of the most upcoming technological development in the field of computer vision is the customer emotion recognition in retail. Comprehending customer emotion has many limitations when assessed by a human, however, facial recognition technology can prove to be a really powerful tool to attain better insights about customers' sentiment. This knowledge can not only provide the researchers with customer behavior patterns, but also predict the future purchasing's of that particular consumer. This will help make smarter business decisions and in turn furnish better customer experiences.

Facial emotion recognition or FER has three important steps:

i. Detection of the face
ii. Extraction of face features
iii. Emotion recognition or classification

Fig 1.1. Smiling Faces

In the past two decades, image classification has been extensively studied and major breakthroughs have been achieved with the help of deep convolutional neural networks. The architecture of these neural networks has two main components performing two main functions that is, feature extraction followed by classification. The first component helps identify simple as well as complex textures and produces describing features of the object of interest. A classifier requires large number of high-level features to be identified as accurate. Therefore, a strong classifier is generally trained on a large amount of data.

This training on large datasets is possible due to the presence of many large-scale image datasets available online for research purposes. One such dataset is the FER-2013 dataset made available on Kaggle for Facial Expression Recognition Challenge. This dataset is used in my research work. A compare and contrast between different convolutional neural networks such as Inception-v3, ResNet-50 and Mini-Xception is performed.

1.2. OBJECTIVES

The objective of this thesis is to achieve a comparable accuracy to that of the current state-of-the-art accuracy for recognizing seven basic expressions in a setup/environment, namely: anger, disgust, fear, happiness, sadness, surprise and neutral. Various convolutional neural networks algorithms used in this study include Inception, Mini-Xception and ResNet-50.

1.3. MOTIVATION

Emotion recognition can help substitute for the lack of survey and data availability. This information can provide an edge to the companies offering customer experiences. Today, a lot of new companies compete with the already established ones for better customer services. Sentiment recognition technology can help these businesses to strengthen customer loyalty and steer their sales. One of the suggested methods to research on customer emotion is by studying the reactions that are recorded while showing them different advertisements. This method is assumed to be better than product surveys as they do not require the consumer to recollect their memory of their use of the product and then put that emotion into words as surveys do.
In future, this technology will definitely be a great addition to implement customer feedback in retail with use case in restaurants, hospitals, Walmart, brand stores, showrooms, real agent businesses etc. It will not just provide more personalized and better customer experiences, but also help reduce malpractices like shoplifting to a great extent.

1.4. CHALLENGES

Recognizing customer emotion and satisfaction level, however, is full of challenges and limitations. Demographic analysis over different nationalities has shown that people from all over the world do not show similar levels of emotion responses. Also, different people have peculiar face structures that can be difficult to generalize and predict with the help of a machine learning algorithm. Another challenge is to capture the faces in real time

effectively in the presence of different lighting atmospheres. Age is another factor; face structures change as a person gets older. Therefore, it is important for a business to identify its consumer base or target customers and then customize the technology accordingly.

1.5. ORGANIZATION OF THE REPORT

This report is mainly divided into six chapters. First chapter is the general introduction about the thesis topic and highlights the challenges/limitations of the research work. Second chapter provides a brief about all the past related work done in this field. It also compares and contrasts different neural networks proposed by various researchers over the course of time. Chapter three describes the dataset and the preprocessing done on the same. It also introduces all the python libraries and modules used in this research. Chapter four provides details about the convolution neural network adopted to perform facial emotion recognition. The report ends with a result section and proposed future work.

Chapter 2
LITERATURE SURVEY

Traditional facial recognition methods like extraction of handcrafted features although successful to an extent, lacked the capacity to perform highly complicated and automatic recognition of faces. In recent years, researchers have adopted various deep learning algorithms to perform facial emotion recognition and have used several databases to obtain better results in terms of accurate emotion detection. All the research work cited below have been used to identify the seven basic facial expressions, namely: anger, disgust, scared, surprised, happy, sad and neutral.

Convolutional Neural Networks for facial emotion recognition was proposed by Mollahosseini [1] on various datasets. The images from the datasets were reduced to 48x48 pixels and data augmentation was performed. Data is augmented in order to expand the existing dataset as deep learning requires a large amount of data for training. Obtaining more supervised dataset is otherwise an expensive task. The used architecture was made up of two convolution-pooling layers and two modules of Inception which in turn contains convolutional layers of size 1x1, 3x3 and 5x5. This architecture models a network-in-network structure which increases the performance and deals with the overfitting problem.

Researchers argued that applying pre-processing steps to the data before using it to train the network can effectively increase the accuracy of the network to identify and classify the emotion correctly. Lopes [2] performed data augmentation, correction in the rotation, cropping the required part, down sampling and intensity normalization on the dataset, before training the CNN. The network consisted of two convolution-pooling layers and two fully connected layers. He performed these tests on three datasets, namely: JAFFE, CK+ and BU-3DFE. The impact of pre-processing techniques was also studied by Mohammadpour [3] in his CNN, which was primarily used to detect the action units of the face. The number of activated action units of the face was indicated using a network that contained two convolution-(max)pooling layers and two fully connected layers.

Sparse Batch Normalization or SBP was adopted in a CNN architecture in 2018 by Cai [4] to deal with the exploding gradient problem in neural networks. The architecture of this network consists of two successive convolution layers, max pooling and then followed by Sparse Batch Normalization. Randomly selected neurons are dropped out during training, a process called Dropout is applied in the middle of three fully connected layers to avoid over-fitting on the dataset.

Facial occlusion problem is another problem which is commonly faced during facial expression recognition. This was dealt by Li [5] in his novel CNN which uses VGGNet network to train the dataset, followed by attention mechanism-based CNN which ensures that the network provides more attention to the useful and relevant features in detecting facial emotions. This was trained and tested on RAF-DB, AffectNet and FED-RO datasets.

Yolcu [6] used three Convolution Neural Networks, each to detect a different part of the face. The images are cropped, and essential parts of the face(key-points) are focused on before using the images for training the network. Later, these images are introduced to a different kind of CNN which is used to identify the facial expression. This provided a better accuracy than using unprocessed images for facial emotion recognition.

Agrawal Mittal [7] used the FER2013 database to study the changes in recognition rate due to the change in different CNN parameters. Number and size of the filters were varied and used on 64x64 pixels images. Different types of optimizers were used on a CNN containing two convolution layers, followed by max pooling and softmax to classify. The researchers were able to achieve an average accuracy of 65%. Deepak Jain [8], in his research used CNN with two residual blocks containing four convolution layers each. The images obtained from datasets such as CK+ and JAFFE were cropped, and their intensity was normalized before training the model.

Spatio-temporal architect to study the variation of facial expressions in different frames was proposed by Kim [9]. They used a combination of CNN and LSTM to achieve this. CNN was used to learn the spatial features in different frames and lastly, LSTM preserved the entire sequence of these recorded spatial features. Yu [10] also studied the same but used nested LSTM instead. He used three sub networks, 3DCNN for identifying the features, T-LSTM for preserving and C-LSTM for tracking the multi-level features. Liang [11] used BiLSTM architecture and Multitask cascade CNN for preprocessing.

Chapter 3
DATASET AND LIBRARIES

3.1. DATASET DESCRIPTION

For this research, FER2013 (facial expression recognition 2013) dataset is used. This dataset was originally employed for the Facial Expression Recognition Challenge. The dataset contains 35887 gray scale images and is designed at 48x48 pixel. The faces in the image are centered and engage equal amount of space. The dataset is divided into three parts: training, evaluation and testing.

The split used in this research work for ResNet-50 and Inception- v3 is 80% training set, 10% validation set and the remaining 10% as the testing set.

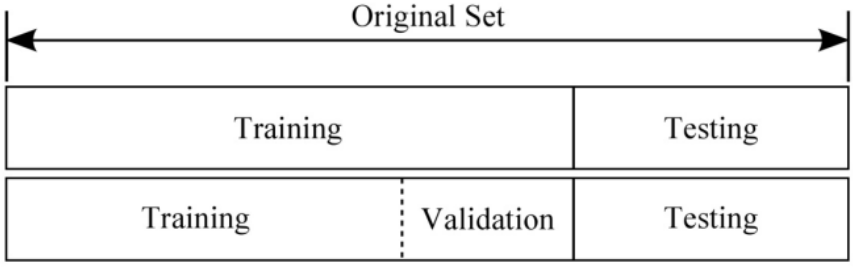

Fig 3.1. Split of the dataset

Training Dataset:
In my research work, training dataset consists of 28,709 images. This dataset is used to train the model with the help of weights and biases. This is the dataset that the model learns and in other word, 'sees'.

Validation/Evaluation Dataset:
In this research, 3589 images were used for evaluation. Validation dataset is a part separated from training dataset which is used to calculate an unbiased evaluation of a

model which is being trained and fitted on the training dataset. This is done while fine tuning the model hyperparameters. The validation dataset is used frequently to evaluate a model and then adjust the parameters accordingly. Even though the model sees the data quite frequently during evaluation, it does not learn this set of data. Therefore, the model is indirectly affected by the validation dataset. This dataset is also known as the development (Dev) set as it is used during the development stage of the model.

Testing Dataset:
35899 images were used for the purpose of testing the built model.

The dataset contains two columns, namely: pixels and emotion. The pixel column has a string for each image which represents space-separated pixel values in row major order. Emotion column represents the emotion in the form of a numeric code which ranges from 0 to 6. The seven categories that are represented by the 0-6 numeric codes are as follows:
0- Angry
1- Disgust
2- Fear
3- Happy
4- Sad
5- Surprise
6- Neutral

The highly imbalanced dataset (as seen in Fig 3.2.) is a major challenge faced while handling this dataset. For example, the class happy contains 8989 images whereas that of disgust contains only 547 images which makes the model biased towards one class if not dealt with properly.

Another challenge faced is the presence of invalid samples. Invalid samples include images that do not contain a face, inappropriate face cropping and wrongly labelled images.

Expression	Training	Validation	Testing	Total
angry	3995	467	491	4953
disgust	436	56	55	547
fear	4097	496	528	5121
happy	7215	895	879	8989
sad	4830	653	594	6077
surprise	3171	415	416	4002
neutral	4965	607	626	6198
Total	**28709**	**3589**	**3589**	**35887**

Fig 3.2. Summary of different classes and sample count

This dataset is perfect for evaluating facial emotions and performing different deep learning methods due to the large number of training samples.

3.2. LIBRARIES USED

NumPy: Powerful library that provides support for large arrays and matrices.

Pandas: Pandas help in performing various data manipulation needed to preprocess the dataset.

OpenCv: This is used to capture the video stream from the laptop camera to use it as in input to our Mini Xception model.

Keras: It provides an interface on python for developing deep learning networks.

Tensorflow: open-source backend required for using Keras functions.

Os: This library provides methods to interact with the operating system.

Imutils: Provides many functions required to perform image processing such as rotation, detecting edges, displaying Matplotlib images etc.

Scikit learn: Machine learning library of python.

Sys: It is a module having functions used to manipulate Python runtime environment.

Fig 3.3. Libraries

Chapter 4
CONVOLUTION NEURAL NETWORK

4.1. WHAT IS CNN?

A CNN or Convolutional Neural Network is a deep learning algorithm that takes an image as an input and attaches weights and biases in order to assign importance to specific objects of interest in that particular image. These specific objects or area of interest in the image should be differentiable than the other parts of the image.

One of the reasons why CNN is preferred over other classification algorithms is the minimal amount of pre-processing involved while using ConvNet. ConvNets possess the ability to learn and adapt to the filters, in contrast to the hand engineered filters that are employed in the primitive methods by extensive training.

Common terminologies used in CNN are discussed in the following sections.

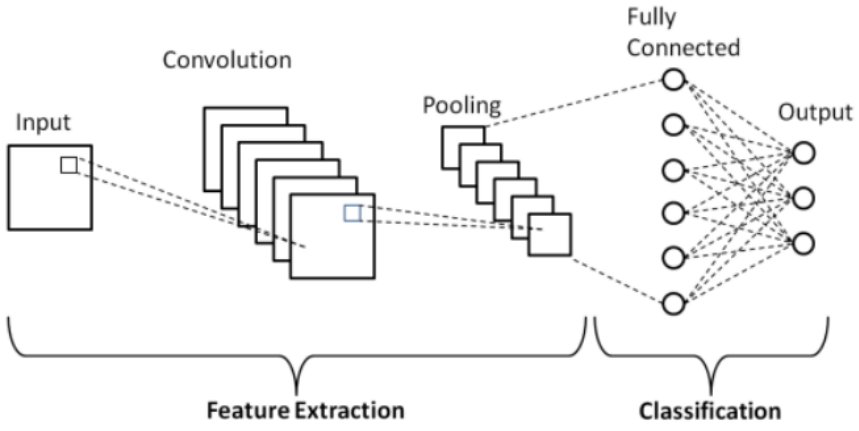

Fig 4.1. CNN

4.2. Layers in CNN

Input Layer:

This layer is nothing but the image data. Image data is in the form of a matrix or a list of pixels. The matrix is transformed into a single column matrix. In our research, this would convert the 48x48 dimension image to 2304 x1 matrix before feeding it into the CNN. The

images are already grey scaled, hence they are 48x48x1 dimension, where the depth is one. In case of RGB, the depth is 3.

We have 28709 training images; therefore, the dimension of the input would be 2304x28709.

Convo Layer:

Also known as the feature extractor layer, helps in extracting the features from the images. This is done so by performing the convolution operation, that is by calculating the dot product between the filter and the local region selected on the input image which is the same size as that of the filter. Convolution operation helps in edge detection. Result of this operation is a single integer. Following this, the filter is moved to the next portion of the input image by a Stride. This repeated till the whole input image is covered and the output is fed to the next layer.

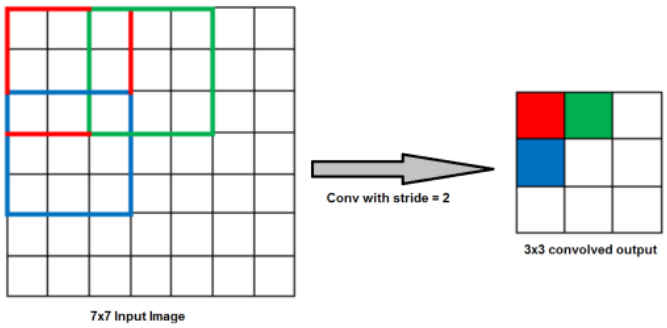

Fig 4.2. Convolution Layer

Stride, mentioned above, represents the number of steps to move the filter to decide the next receptive field. Generally, stride is taken as one.

Let's say a convolution operation is performed on an NxN input with a FxF filter with stride equals one, then the output would be (N-F+1)x(N-F+1).

Convo layer also consists of **ReLU activation**. ReLU stands for Rectified Linear Unity and is defined as the max function between 0 and the input. Basically, it makes all the negative values zero. This is one of the most efficient ways of turning the input non-linear.

Non-linearity is a required element in activation functions such as ReLU so as to produce a nonlinear decision boundary with the help of weights and inputs. The more complex the model, the more nonlinearity is required. The ability of the model to correctly identify and

classify complex input objects and not just restrict the classification based on linear relations is what makes it a strong classifier.

Pooling Layer:

Following convolution, the spatial volume of the input image is reduced using pooling layer. By applying pooling/ max pooling before the fully connected layer, we are making the network computationally less expensive. Pooling layer does not have a parameter, but it does have two hyperparameters, namely: Filter and Stride.

For instance, if a 2x2 filter is applied on a 4x4 dimension input with stride 2, then the input will reduce to 2x2. Which is one fourth of the initial volume.

Let F= Filter, S= Stride and Input Dimension = W_1 x H_1 x D_1, then, $W_2 = (W_1-F)/S+1$, $H_2 = (H_1-F)/S+1$ and $D_2 = D_1$, where W2, H2 and D2 are the width, height and depth of output.

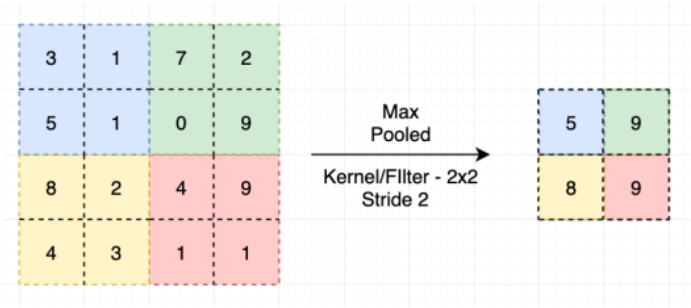

Fig 4.3. Pooling

Two types of pooling are Max Pooling and Average Pooling.

Average Pooling: This is used to output the average of all the input values of the matrix area which is covered by the filter/ kernel. Average pooling performs dimensionality reduction.

Max Pooling: Max pooling is used to output the maximum value out of the input matrix area which is covered by the filter/kernel. Max pooling does the job of a Noise/Disturbance Suppressant. The noisy activations are discarded, and dimension is reduced with the help of this layer. Max pooling is preferred over average pooling.

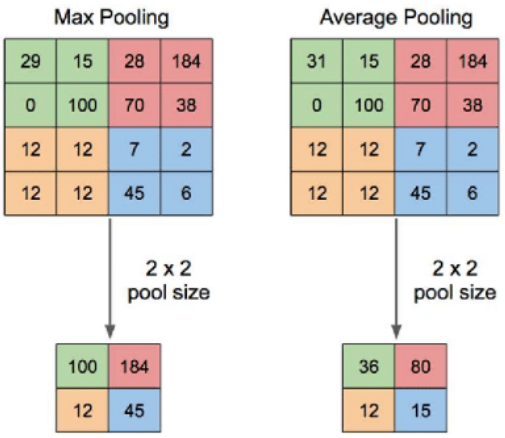

Fig 4.4. Max Vs. Average Pooling

Fully Connected Layer:

Fully Connected layer is responsible for connecting neurons in one layer that of another layer. It uses neurons, weights and biases. This layer is used to categorize and classify an input image into different categories/classes by training.

Softmax/Logistic Layer:

This layer is the last layer of the convolutional neural network and it is preceded by the Fully Connected layer. For binary classification, logistic is used whereas softmax is used. For multi-classification.

Output Layer:

Output layer is nothing but the one-hot encoding of the final predicted label.

4.3. CNN TERMS AND TERMINOLOGIES

Epoch:
One epoch refers to the complete cycle where the entire dataset moves forward and backward through a neural network. In other words, the network has parsed and seen the entire dataset once. One epoch is clearly too large to handle for computation, hence it is divided into certain batches.

Batch Size:
In brief, the batch size refers to the number of input images/ samples from the training dataset that the neural network works on before updating the weights of the neural network and proceeding to work on the next batch. The batch size can decide if the network is underfitted, overfitted or optimal.

Batch Gradient descent is a learning algorithm where the batch size is same as the training dataset. In Stochastic Gradient descent, the batch is equal to one and Mini-Batch Gradient descent refers to a algorithm where the size of the batch ranges between one and the size of the training set.

Dense Layers:
It refers to the layer that receives a input vector followed by its multiplication with the filter/matrix or the parameters and finally producing the output vector.

Activation Function:
Activation Functions are referred to as the non-linear functions that are used in the dense layers. Their job is to receive input and turn it into a non-linear output. For instance, tan(A+B). Here, A+B is turned to nonlinear with the help of the tan function. The most famous and commonly used activation function is the Rectified linear Unit (ReLU) which has been discussed above.

Training Loop:

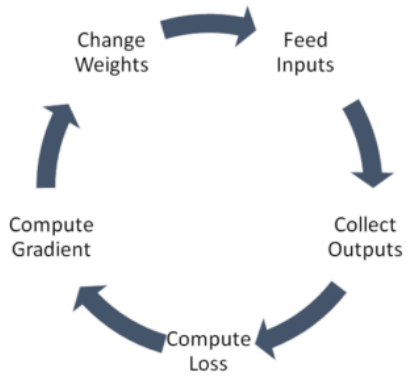

Fig 4.5. Training Loop

The training loop performs the learning of the neural network. Training includes feeding the input the model, collecting the outputs, analyze and contrast them with the expected outputs and then making the relevant required changes to the weights to correct the wrong

outputs. One of the most important part of the loop is computing the loss using the loss function.

Loss Function:
It is the function that gives a measure of the amount of wrong or error in the model. To optimize the network, we need to reduce this loss and make the model as accurate as possible. This can be done by finding the minimum value of loss function and this performed by a method called Gradient Descent.

Gradient Descent:
Given, Model(M), weights(θ), input x, output y and the loss function L, we can find out the gradient/slope of L(y, M(x; θ)). This slope(∇L) is an indicative of the change in loss corresponding to the changes made in the weights. This gives the optimal weights.

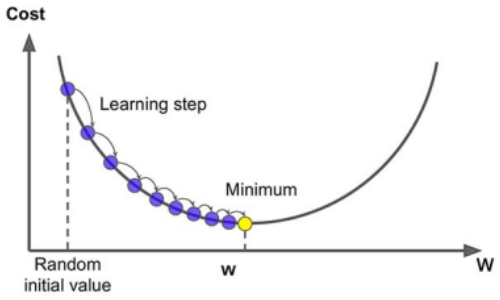

Fig 4.6. Gradient Descent

Learning Rate:
While training the network, the best way is to update weights very slowly. Even though it might take longer to train, it improves the convergence. The learning rate is a parameter of the optimizer.

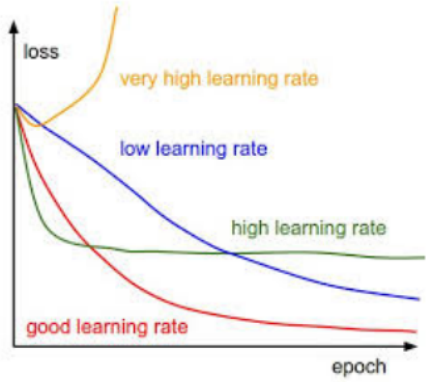

Fig 4.7. Different learning rates

Batch Normalization:
Batch Normalization is used in artificial neural networks to make them train quicker and provide stability by normalizing (re-scaling and re-centering) the input to the layer. It helps drastically reduce the number of training epochs required to train the network and hence reduces the time complexity.

4.4. TYPES OF OPTIMIZERS USED IN THIS RESEARCH

SGD Optimizer

SGD stands for Stochastic gradient descent and is a method for performing optimization the objective function with the help of smoothness properties like differentiable etc. It is nothing but the stochastic approximation of the gradient descent optimization.

ADAM Optimizer

The ADAM optimization algorithm is adopted extensively in deep learning applications like computer vision and can be considered as an extension to stochastic gradient descent. This method calculates adaptive learning rates individually for distinct parameters from estimates of first and second moments of the gradients. The algorithm is responsible for calculating an exponential moving average of the gradient and the squared gradient. Beta1 and beta2 are the parameters that control the decay rates of these moving averages.

4.5. TRANSFER LEARNING

Transfer learning refers to the process of making use of the knowledge gained from a previously solved problem and applying it to the current problem at hand.

Pre-Training

Pretraining is performed on a large dataset where all the parameters of the neural network are trained which takes hours on the GPU.

Fine Tuning

The new dataset is used to fine tune the pre-trained convolutional neural network.

If the new dataset is similar to the first one, then the same weights can be used.In case the new dataset is extremely small, only the last few layers are trained instead of training the entire network. The other layers are kept fixed. Few final layers from the pretrained model are removed and new final layers are retrained using the new dataset. In case the new dataset is too large, then it is proffered to train the entire model again with the initial weights.

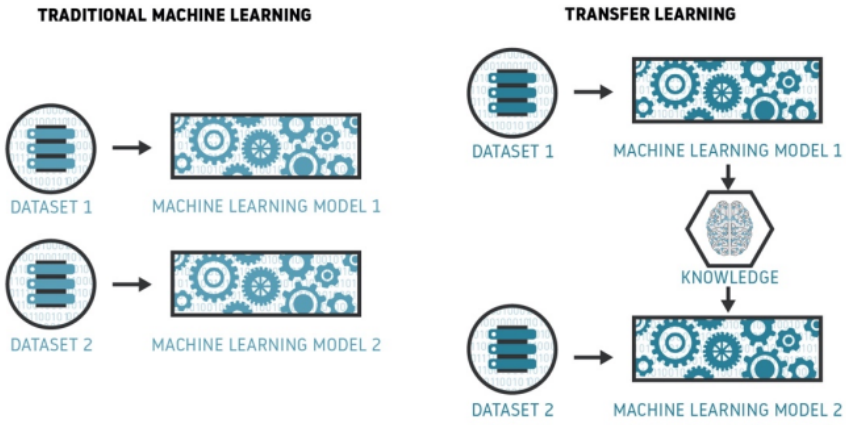

Fig 4.8. Transfer Learning Depiction

If the new dataset is not similar to the original one, then the earlier layers can be fixed and retrain the remaining layers in case of small dataset. In case of big dataset, the whole model can be retrained with the initial weights from the pre-trained network.

The earlier layers of the network contain general features like edge detector etc., and as we go deeper, it gets more specific to the classes contained in the original dataset for classification.

4.6. CNN FOR FACIAL EMOTION RECOGNITION

Convolution Neural Network has played an important role in achieving advancements in the field of Computer Vision. The basic role played by these networks is reducing the image into an easily processable form without actually losing the important characteristics required for an accurate prediction.

Few of the commonly used CNN architectures for facial emotion recognition are VGG16, Inception V3, ResNet-50 and Xception.

4.7. CNN MODELS BUILT IN THIS RESEARCH

In this research, three CNN models, namely: ResNet-50, Inception-v3 and mini-Xception are used. Following chapters explain their architecture, provide model summary and report the result and accuracy obtained.

Chapter 5
Mini-Xception

5.1. Mini Xception INTRODUCTION

Xception architecture utilizes a combination of depth-wise separable convolutions and residual modules. Residual modules change the mapping between two consecutive layers so that the difference between the original features and the desired ones are nothing but the learned features. Depth-wise separable convolutions separate the feature extraction process and combination inside the same layer, thereby reducing the number of parameters.

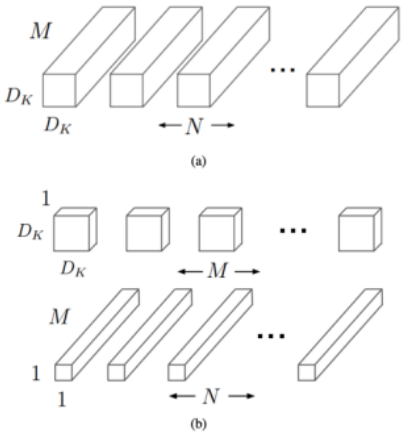

Fig 5.1. Depth-wise separable convolutions

Depth-wise convolution and point-wise convolution together combine into depth-wise separable convolutions. These layers serve the purpose of separating channel cross-correlations with spatial cross-correlations. This is done so by applying D x D filter for each channel M, followed by applying 1 x 1 x M filter N times to combine input channels (M) to output channels (N).

5.2. PRE-PROCESSING

The pixel values of the images are normalized by dividing it by 255 before introducing it to the neural network for training. This is done so as to ensure that the data is zero centered which allows the model to converge faster. It is divided by 255 in particular because otherwise a pixel value ranges between 0 to 255 (8 bits). Later 0.5 is subtracted and the result is doubled to obtain the range of pixels between -1 to 1.

```
def preprocess(x, v2=True):  # to keep the image btw. -1 and 1
    x = x.astype('float32')
    x = x/255.0
    if v2:
        x = (x - 0.5)*2.0
    return x
```

```
images = preprocess(images)
```

Fig 5.2. Code Snippet of pre-processing

5.3. DATA AUGMENTATION

Data augmentation is done artificially when the training set is not large enough for the network to learn effectively. It is performed by transforming the training dataset by flipping, rotating, cropping, normalizing, zooming etc. and creating modified versions of the same.

ImageDataGenerator class in the Keras library helps in fitting the model using image data augmentation.

```
data_generator = ImageDataGenerator(featurewise_center=False, featurewise_std_normalization=False,
                                    rotation_range=10, width_shift_range=0.1,
                                    height_shift_range=0.1, zoom_range=.1, horizontal_flip=True)
```

Fig 5.3. Code Snippet of Data Augmentation

5.4. Mini-Xception ARCHITECTURE

Mini Xception architecture was proposed by Octavio Arragia. It contains approximately 60,000 parameters and not more than 20 layers. It contains fully connected convolution networks which has four residual depth-wise separable convolutions. Every convolution ends with batch normalization and performing ReLU activation function. In the end, Global Average Pooling is done followed by softmax activation function to predict.

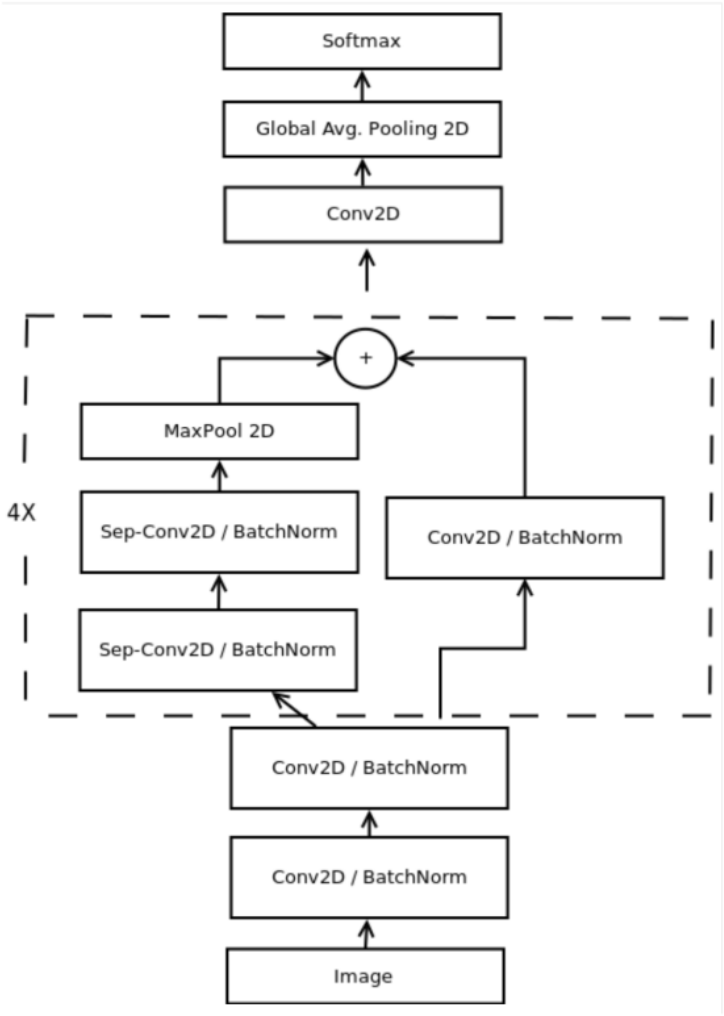

Fig 5.4. Mini Xception Architecture

Batch normalization helps in drastically reducing the required number of epochs to train the deep neural network. It does so by standardizing the inputs to a particular layer for every mini batch.

ReLU or rectified linear unit is a particular type of activation function which just needs to pick max(0,x) unlike Sigmoid, where extensive exponential operations are performed.

In the final layer, global average pooling is done by computing the mean value for every feature map. This mean value is then supplied to the softmax function which converts it to a probability.

5.5. MODEL SUMMARY

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 48, 48, 1)]	0	
conv2d (Conv2D)	(None, 46, 46, 8)	72	input_1[0][0]
batch_normalization (BatchNorma	(None, 46, 46, 8)	32	conv2d[0][0]
activation (Activation)	(None, 46, 46, 8)	0	batch_normalization[0][0]
conv2d_1 (Conv2D)	(None, 44, 44, 8)	576	activation[0][0]
batch_normalization_1 (BatchNor	(None, 44, 44, 8)	32	conv2d_1[0][0]
activation_1 (Activation)	(None, 44, 44, 8)	0	batch_normalization_1[0][0]
separable_conv2d (SeparableConv	(None, 44, 44, 16)	200	activation_1[0][0]
batch_normalization_3 (BatchNor	(None, 44, 44, 16)	64	separable_conv2d[0][0]
activation_2 (Activation)	(None, 44, 44, 16)	0	batch_normalization_3[0][0]
separable_conv2d_1 (SeparableCo	(None, 44, 44, 16)	400	activation_2[0][0]
batch_normalization_4 (BatchNor	(None, 44, 44, 16)	64	separable_conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 22, 22, 16)	128	activation_1[0][0]
max_pooling2d (MaxPooling2D)	(None, 22, 22, 16)	0	batch_normalization_4[0][0]
batch_normalization_2 (BatchNor	(None, 22, 22, 16)	64	conv2d_2[0][0]
add (Add)	(None, 22, 22, 16)	0	max_pooling2d[0][0] batch_normalization_2[0][0]
separable_conv2d_2 (SeparableCo	(None, 22, 22, 32)	656	add[0][0]
batch_normalization_6 (BatchNor	(None, 22, 22, 32)	128	separable_conv2d_2[0][0]
activation_3 (Activation)	(None, 22, 22, 32)	0	batch_normalization_6[0][0]
separable_conv2d_3 (SeparableCo	(None, 22, 22, 32)	1312	activation_3[0][0]
batch_normalization_7 (BatchNor	(None, 22, 22, 32)	128	separable_conv2d_3[0][0]
conv2d_3 (Conv2D)	(None, 11, 11, 32)	512	add[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 11, 11, 32)	0	batch_normalization_7[0][0]
batch_normalization_5 (BatchNor	(None, 11, 11, 32)	128	conv2d_3[0][0]
add_1 (Add)	(None, 11, 11, 32)	0	max_pooling2d_1[0][0] batch_normalization_5[0][0]
separable_conv2d_4 (SeparableCo	(None, 11, 11, 64)	2336	add_1[0][0]
batch_normalization_9 (BatchNor	(None, 11, 11, 64)	256	separable_conv2d_4[0][0]
activation_4 (Activation)	(None, 11, 11, 64)	0	batch_normalization_9[0][0]
separable_conv2d_5 (SeparableCo	(None, 11, 11, 64)	4672	activation_4[0][0]
batch_normalization_10 (BatchNo	(None, 11, 11, 64)	256	separable_conv2d_5[0][0]
conv2d_4 (Conv2D)	(None, 6, 6, 64)	2048	add_1[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0	batch_normalization_10[0][0]
batch_normalization_8 (BatchNor	(None, 6, 6, 64)	256	conv2d_4[0][0]
add_2 (Add)	(None, 6, 6, 64)	0	max_pooling2d_2[0][0] batch_normalization_8[0][0]
separable_conv2d_6 (SeparableCo	(None, 6, 6, 128)	8768	add_2[0][0]
batch_normalization_12 (BatchNo	(None, 6, 6, 128)	512	separable_conv2d_6[0][0]
activation_5 (Activation)	(None, 6, 6, 128)	0	batch_normalization_12[0][0]
separable_conv2d_7 (SeparableCo	(None, 6, 6, 128)	17536	activation_5[0][0]
batch_normalization_13 (BatchNo	(None, 6, 6, 128)	512	separable_conv2d_7[0][0]

conv2d_5 (Conv2D)	(None, 3, 3, 128)	8192	add_2[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 3, 3, 128)	0	batch_normalization_13[0][0]
batch_normalization_11 (BatchNo	(None, 3, 3, 128)	512	conv2d_5[0][0]
add_3 (Add)	(None, 3, 3, 128)	0	max_pooling2d_3[0][0] batch_normalization_11[0][0]
conv2d_6 (Conv2D)	(None, 3, 3, 7)	8071	add_3[0][0]
global_average_pooling2d (Globa	(None, 7)	0	conv2d_6[0][0]
predictions (Activation)	(None, 7)	0	global_average_pooling2d[0][0]

Fig 5.5. Model Summary

Chapter 6
ResNet 50

6.1. ResNet-50 INTRODUCTION

ResNet is a classical neural network which stands for Residual Networks and is used for many computer vision tasks. ResNet became popular due to its ability to train extensive deep neural networks that have more than 150 layers. Earlier, very deep neural networks were hard to train due to the vanishing gradient problem.

The concept of skip connection was introduced in ResNet. In the image below, left side shows the stacking of convolutional layers and the right side has the additional original input attached to the output of the convolutional block which is called the skip connection.

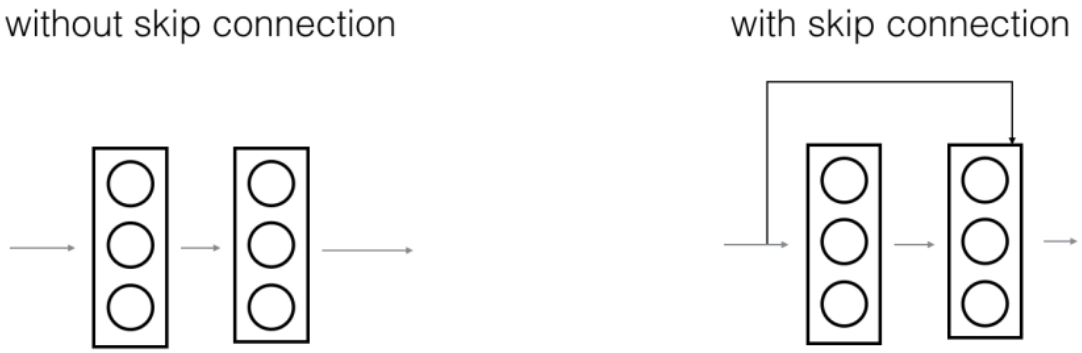

Fig 6.1. Residual Network with skip connection

6.2. ResNet-50 ARCHITECTURE

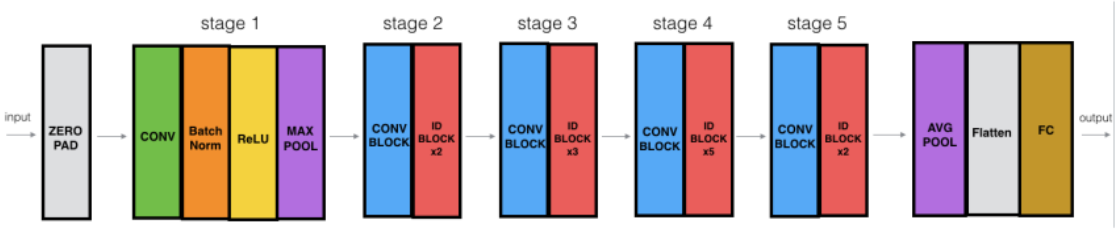

Fig 6.2. ResNet-50 Architecture

The ResNet-50 is a variant of the original ResNet which consists of 5 stages. Each stage contains a convolution and an identity block. Further, each convolution block has three convolution layers. Every identity block is also made up of three convolution layers. In total, ResNet-50 has more than 23 million trainable parameters.

It is common to use pretrained ResNet-50 models in Keras. Keras provides many backbone models with their ImageNet weights that are made available in its library.

In my research work, I have used a pretrained model with initial weights of that of VGGFace. . It takes 197x197 dimension images as input and trains on 50 layers. The top layers are removed and two additional layers, that is: fully connected layer and a logistic layer is added. It is then fine-tuned on the FER2013 dataset. First only the top layers are trained by freezing all the remaining layers. It is done so using the ADAM optimizer for 5 epochs. Later, the entire neural network is trained for 100 epochs with batch size 128 using the SGD optimizer.

Chapter 7
Inception V3

7.1. Inception V3 INTRODUCTION

Inception v3 is adopted widely in the field of image recognition and has proven to show greater than 78.1% accuracy on the ImageNet dataset. Inception V3 is aimed at reducing the computational power by making use of the previous Inception architectures.

The Inception model is made up of asymmetric as well as symmetric building blocks, convolutions, average/max. pooling, dropouts, concats and finally the FC (fully connected layer). Batch Normalization is also used extensively and is applied to the inputs of the activation function.

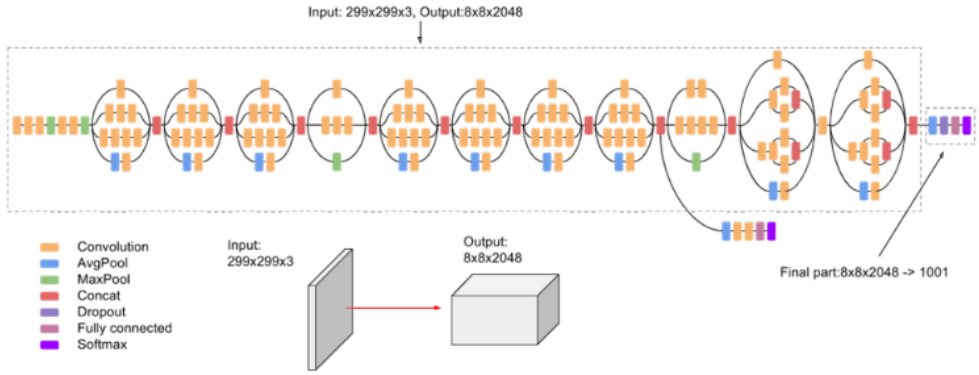

Fig 7.1. Inception V3 Architecture

7.2. Inception- V3 ARCHITECTURE

The architecture of Inception-v3 is built progressively as follows:

Factorized Convolution: it reduces the number of parameters involved in the network inurn reducing the computational complexity. It also provides a check on the efficiency of the network.

Smaller Convolutions: bigger convolutions are replaced by smaller convolutions so as to help with the faster training.

Asymmetric Convolution: Has lesser parameters than that of a symmetric convolution.

Fig 7.2. Asymmetric vs. Symmetric Convolution

Auxiliary classifier: This acts like a regularizer in Inception v3 network and I added in between layers during training.

In my research work, I have used a pretrained model with initial weights of that of ImageNet. It takes 139x139 dimension images as input and trains on 48 layers. The top layers are removed and two additional layers, that is: fully connected layer (dense layer) and a logistic layer is added. It is then fine-tuned on the FER2013 dataset. First only the top layers are trained by freezing all the remaining layers. It is done so using the ADAM optimizer for 5 epochs. Later, the entire neural network is trained for 100 epochs with batch size 128 using the SGD optimizer.

Chapter 8
EVALUATION METRICS

Confusion matrix

It is a table that describes the classifiers performance on a test data collection where the real values are known.

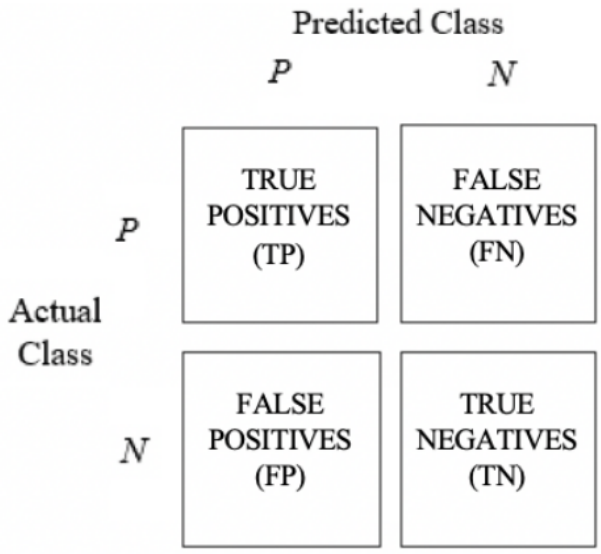

Fig 8.1. Confusion Matrix

8.1. RECALL

It is the probability that a relevant document is retrieved by a query. It is also known as sensitivity and is the number of correct test results upon the total number of results that are returned

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

8.2. PRECISION

It is defined as the total number of required documents retrieved divided by all the documents retrieved. In short it means that it is the percentage of documents retrieved that are pertinent .

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

8.3. ACCURACY

It is ration of the total number of correct predicted examples divided by the total number of examples. Accuracy is reliable only if all the classes are equally important

$$\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{\text{true positives} + \text{true negatives} + \text{false positives} + \text{false negatives}}$$

8.4. F1 Score

It is basically a statistical measure to rate performance and is defined as the harmonic mean between recall and precision. it measures how accurate a test or an individual is.

$$\text{F1 Score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Chapter 9
CONCLUSION

9.1. Mini-Xception

The state-of-the-art accuracy on FER2013 dataset, achieved by a combination of CNN extracted as well as handcrafted features is 75.4%. However, the accuracy achieved after running 5 epochs of batch size 32 is 53.3% in our research work.

```
model.fit(xtrain,ytrain,batch_size=32,epochs=5)

Epoch 1/5
898/898 [==============================] - 91s 101ms/step - loss: 1.6598 - accuracy: 0.3832
Epoch 2/5
898/898 [==============================] - 89s 99ms/step - loss: 1.3869 - accuracy: 0.4866
Epoch 3/5
898/898 [==============================] - 90s 100ms/step - loss: 1.2671 - accuracy: 0.5293
Epoch 4/5
898/898 [==============================] - 92s 103ms/step - loss: 1.1873 - accuracy: 0.5608
Epoch 5/5
898/898 [==============================] - 96s 107ms/step - loss: 1.1317 - accuracy: 0.5834

<tensorflow.python.keras.callbacks.History at 0x7ff48296d070>

test_accuracy = model.evaluate(xtest,ytest)[1]
print('test accuracy',np.round((test_accuracy)*100,2))

225/225 [==============================] - 3s 14ms/step - loss: 1.2832 - accuracy: 0.5330
test accuracy 53.3
```

Fig 9.1. Accuracy of Mini-Xception

The total number of parameters are 58,423, where 56,951 are trainable.

```
=================================================
Total params: 58,423
Trainable params: 56,951
Non-trainable params: 1,472
_________________________________________________
```

Fig 9.2. Mini-Xception parameters

9.2. ResNet-50

Our model achieved an accuracy of 71.2% which is the highest of all three models built in this study.

```
ResNet
Accuracy:
0.7124547227640011


Report:
              precision    recall  f1-score   support

       Anger       0.64      0.64      0.64       491
     Disgust       0.73      0.65      0.69        55
        Fear       0.58      0.49      0.53       528
   Happinness       0.88      0.91      0.89       879
     Sadness       0.58      0.60      0.59       594
    Surprise       0.81      0.80      0.80       416
     Neutral       0.68      0.74      0.71       626

    accuracy                           0.71      3589
   macro avg       0.70      0.69      0.69      3589
weighted avg       0.71      0.71      0.71      3589
```

Fig 9.3. Accuracy of ResNet-50

The total number of parameters are 25,666,503 , where 25,613,383 are trainable.

```
================================================================================================
Total params: 25,666,503
Trainable params: 25,613,383
Non-trainable params: 53,120
________________________________________________________________________________________________
```

Fig 9.4. ResNet-50 parameters

9.3. Inception V3

Our model achieved an accuracy of 63.8% .

```
Inception
Accuracy:
0.6386179994427417


Report:
              precision    recall  f1-score   support

       Anger       0.56      0.55      0.56       491
     Disgust       0.74      0.36      0.49        55
        Fear       0.49      0.38      0.43       528
   Happinness       0.85      0.89      0.87       879
     Sadness       0.47      0.50      0.48       594
    Surprise       0.76      0.75      0.75       416
     Neutral       0.58      0.66      0.62       626

    accuracy                           0.64      3589
   macro avg       0.64      0.58      0.60      3589
weighted avg       0.64      0.64      0.63      3589
```

Fig 9.5. Accuracy of Inception V3

The total number of parameters are 23,908,135 , where 23,873,703 are trainable.

```
================================================================================================
Total params: 23,908,135
Trainable params: 23,873,703
Non-trainable params: 34,432
________________________________________________________________________________________________
```

Fig 9.6. Inception V3 parameters