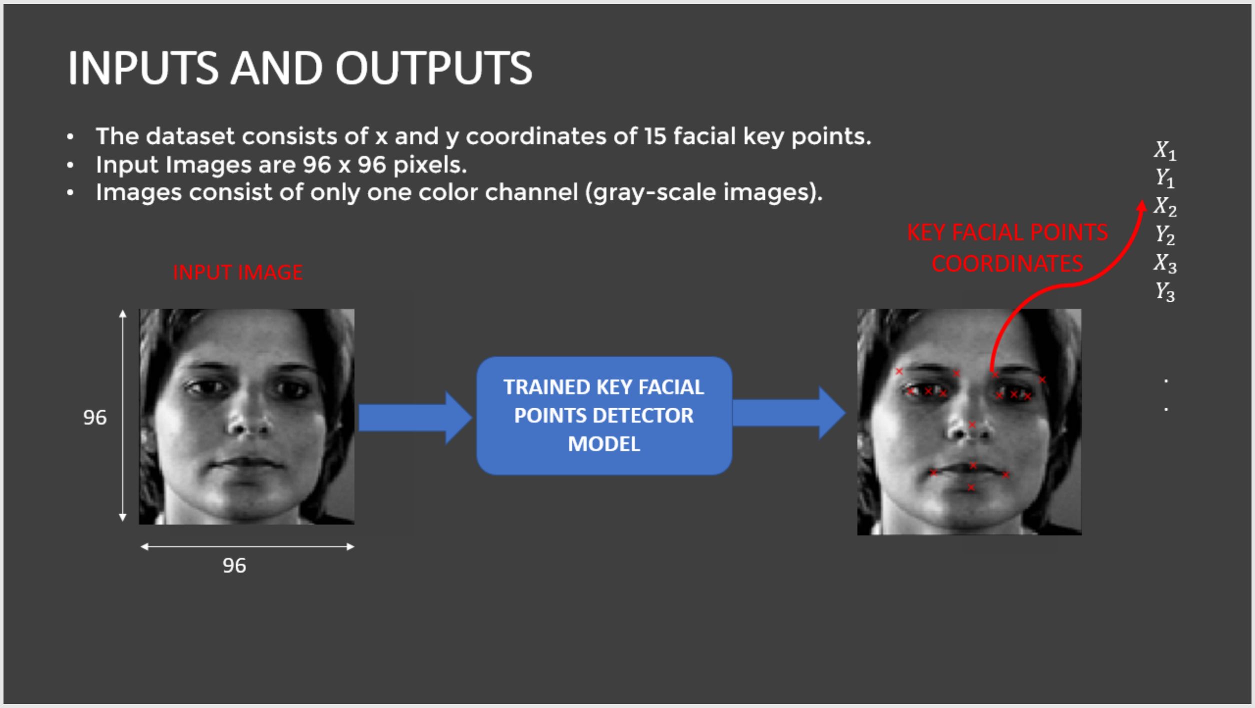
Introduction:

Facial key point detection is a computer vision task that involves locating and identifying characteristic points on the human face. These key points can be used for a variety of applications, such as face recognition, emotion analysis, face swapping, and face alignment.

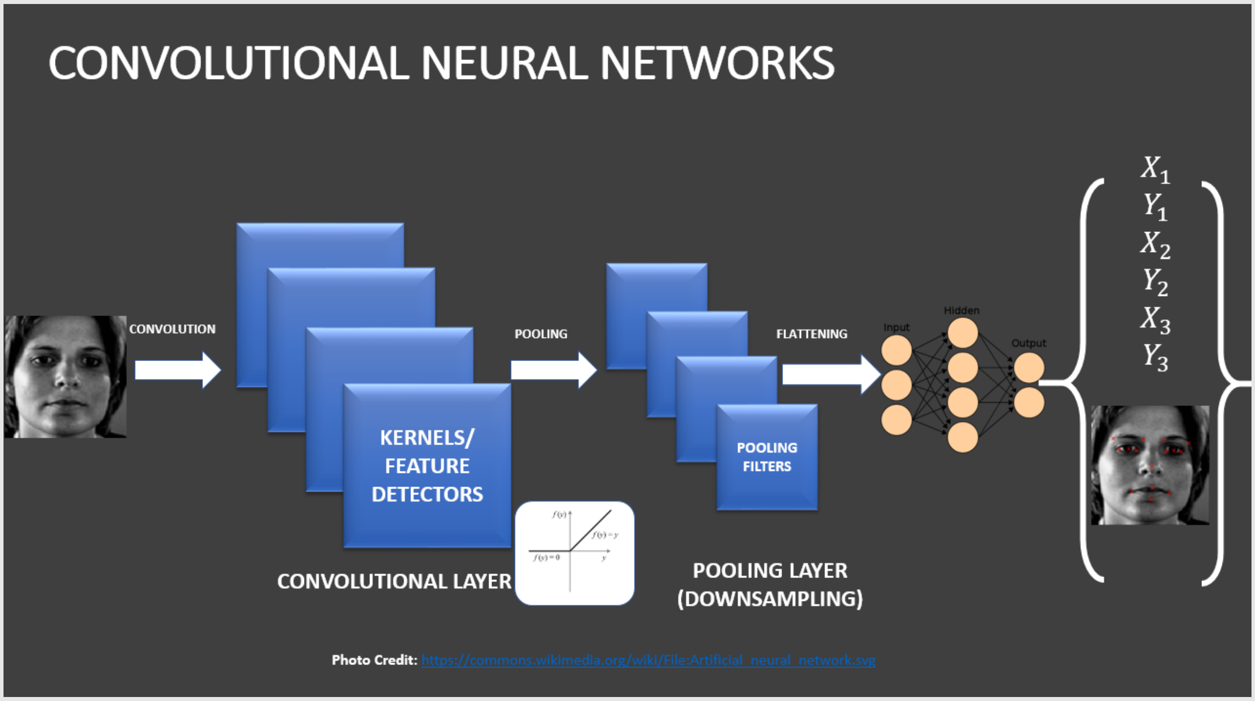


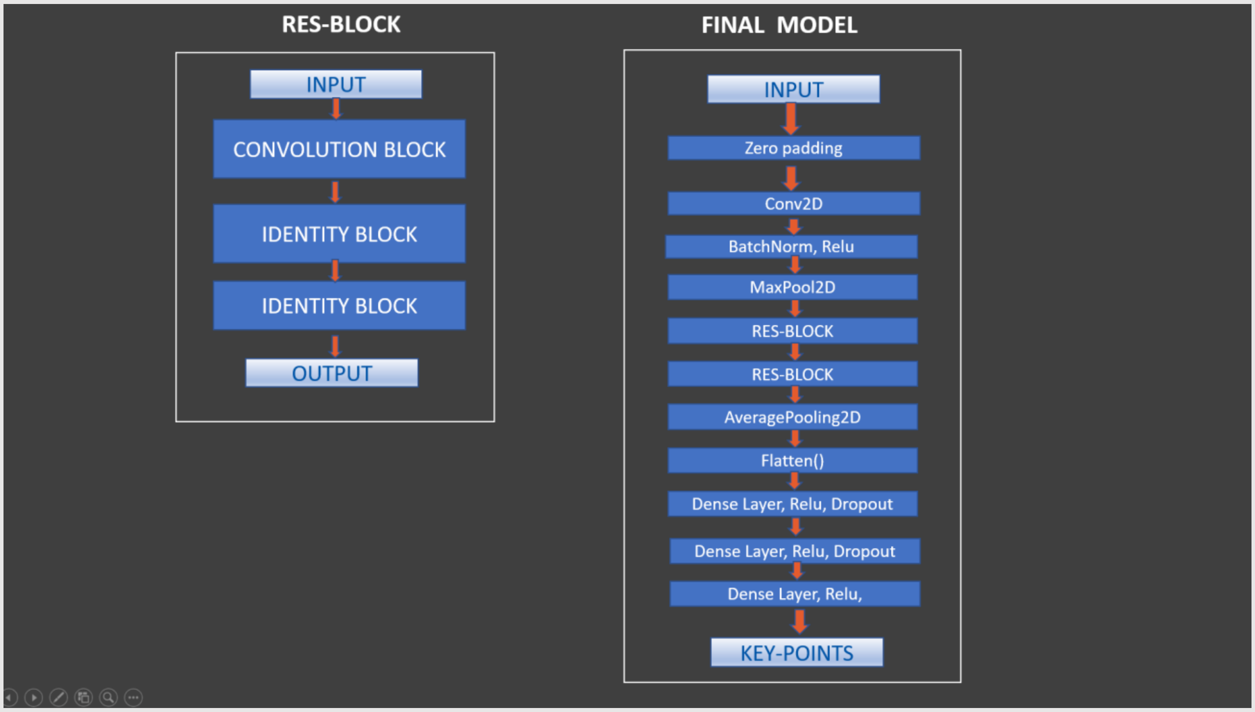
Problem statement:

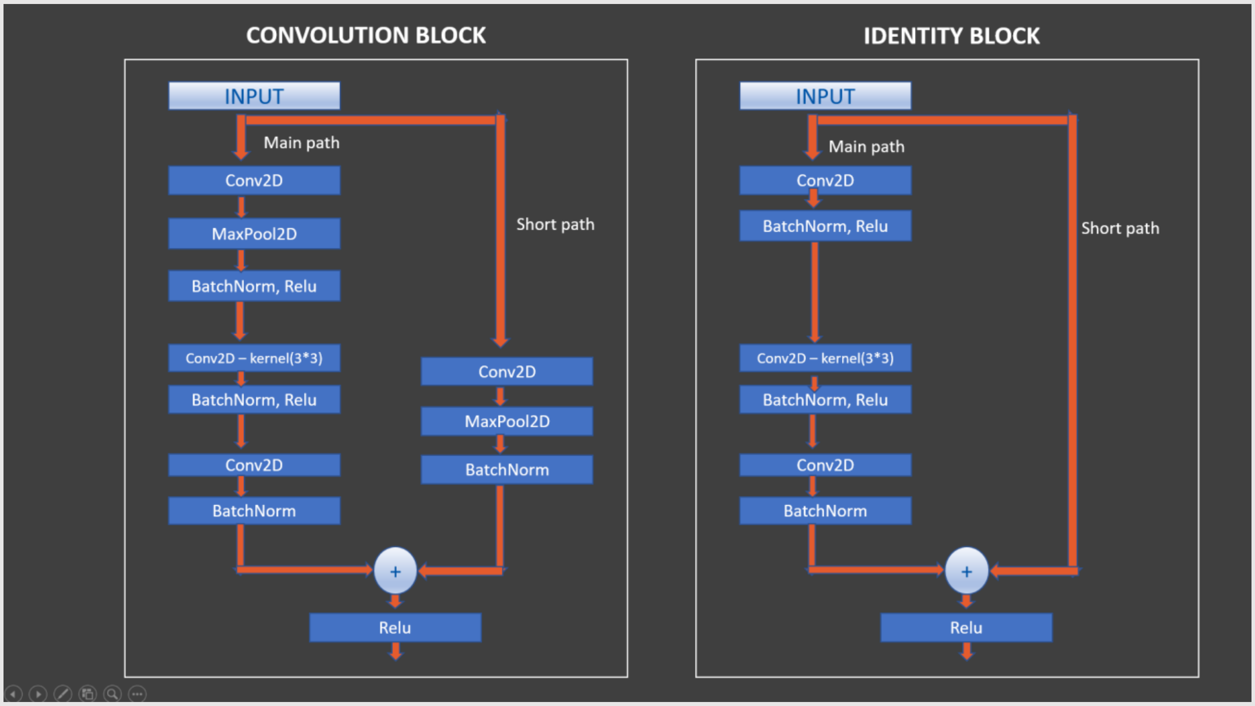
Facial key point detection presents a complex and intricate challenge, primarily attributed to the multitude of factors contributing to variations in human faces. These factors include diverse poses, expressions, lighting conditions, occlusions, and individual facial differences. As a result, accurately identifying and locating key points on a face becomes a formidable task for computer vision systems, demanding sophisticated and robust solutions to address these inherent complexities.

Solution:

In this project, I will build a deep learning model based on convolutional neural networks (CNNs) and residual blocks to detect 15 facial key points from a given face image. CNNs are a type of deep learning model that are well-suited for image processing tasks. They are able to learn from the spatial relationships between pixels in an image, which makes them effective at identifying key points on faces. Residual blocks are a type of CNN architecture that can help to improve the accuracy of the model by allowing it to learn from both low-level and high-level features of the images.


CONVOLUTIONAL NEURAL NETWORKS
CONVOLUTION
KERNELS/ FEATURE DETECTORS
CONVOLUTIONAL LAYER
POOLING
POOLING FILTERS
POOLING LAYER (DOWNSAMPLING)
FLATTENING
Input
Hidden
Output
X_1 Y_1 X_2 Y_2 X_3 Y_3
Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

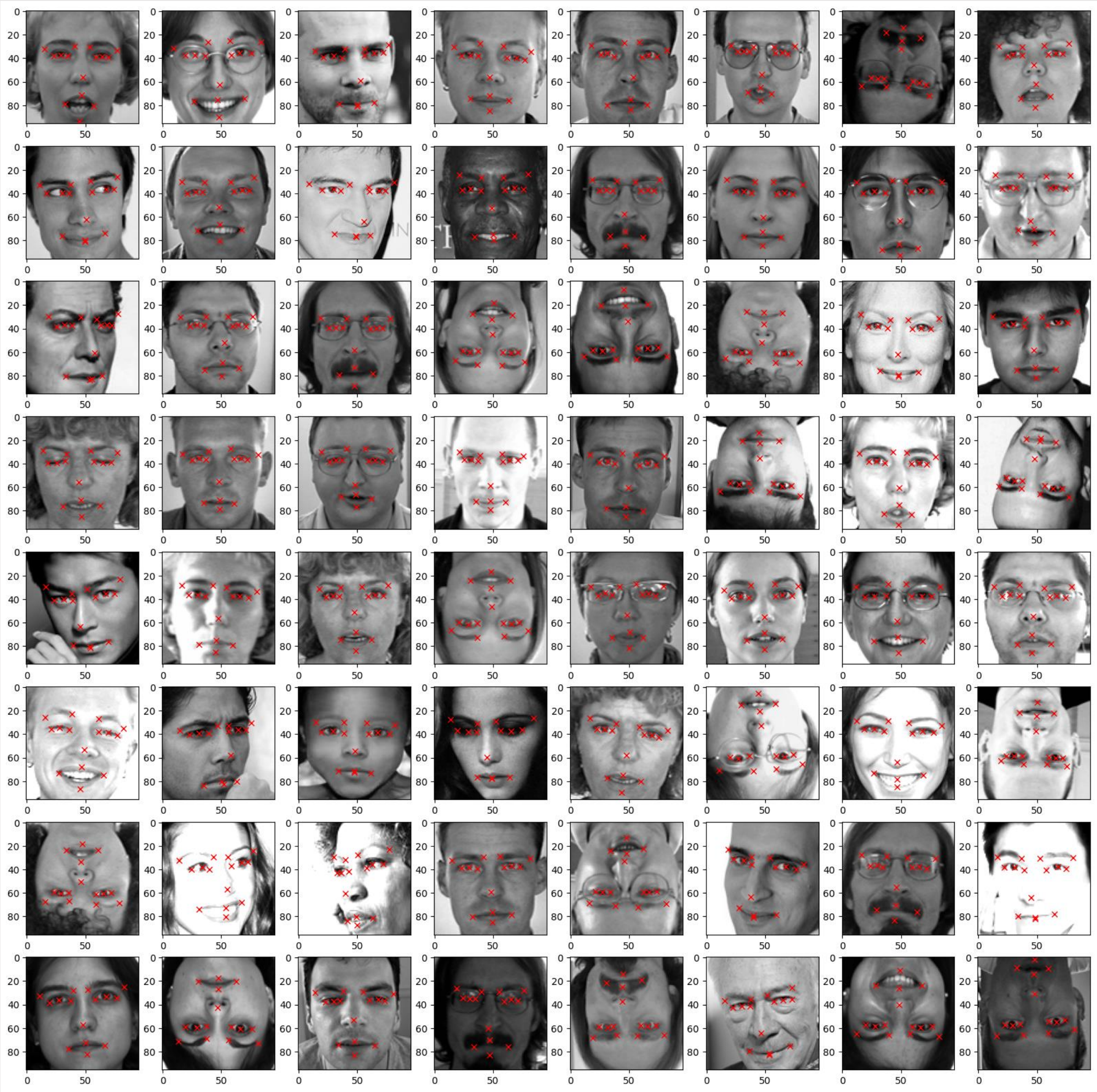

RES-BLOCK
INPUT
CONVOLUTION BLOCK
IDENTITY BLOCK
IDENTITY BLOCK
OUTPUT
FINAL MODEL
INPUT
Zero padding
Conv2D
BatchNorm, Relu
MaxPool2D
RES-BLOCK
RES-BLOCK
AveragePooling2D
Flatten()
Dense Layer, Relu, Dropout
Dense Layer, Relu, Dropout
Dense Layer, Relu,
KEY-POINTS



Dataset:

For the training and evaluation of my model, I have selected the Facial Keypoints Detection dataset sourced from Kaggle. This comprehensive dataset comprises a total of 7049 grayscale images of human faces, accompanied by meticulously annotated key points. To ensure optimal performance and enhance the model's ability to handle real-world scenarios, I meticulously preprocessed the dataset. This involved a series of essential steps, such as cropping and resizing images to a standardized format, as well as normalizing pixel values for consistency.

Moreover, I adopted an augmentation strategy to further fortify my model's robustness. This entailed applying horizontal and vertical image flips, effectively augmenting the dataset to expose the model to a wider array of facial orientations. Additionally, I manipulated the brightness levels of the images by both increasing and decreasing intensity, effectively simulating varying lighting conditions. This augmentation process serves to enrich the dataset, enabling the model to grasp the intricate nuances of different facial configurations, lighting conditions, and orientations. The combination of preprocessing and augmentation ensures that my model is well-equipped to contend with the inherent challenges posed by the diverse variations in facial pose, expression, illumination, and occlusion.



Methodology:

I will preprocess the data by cropping, resizing, normalizing, and augmenting the images. This will help to improve the performance of the model by making the data more consistent and representative of the real world. I will then design and implement a CNN model with residual blocks. I will train the model using mean squared error (MSE) as the loss function and root mean squared error (RMSE) as the metric. I will evaluate the model on a held-out test set to measure its accuracy.

Model summary:

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
=
input_1 (InputLayer)            (None, 96, 96, 1)    0
__________________________________________________________________________________________________
zero_padding2d (ZeroPadding2D)  (None, 102, 102, 1)  0           input_1[0][0]
__________________________________________________________________________________________________
conv1 (Conv2D)                  (None, 48, 48, 64)   3200        zero_padding2d[0][0]
__________________________________________________________________________________________________
bn_conv1 (BatchNormalization)   (None, 48, 48, 64)   256         conv1[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 48, 48, 64)   0           bn_conv1[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 23, 23, 64)   0           activation[0][0]
__________________________________________________________________________________________________
res_2_conv_a (Conv2D)           (None, 23, 23, 64)   4160        max_pooling2d[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 11, 11, 64)   0           res_2_conv_a[0][0]
__________________________________________________________________________________________________
bn_2_conv_a (BatchNormalization (None, 11, 11, 64)   256         max_pooling2d_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 11, 11, 64)   0           bn_2_conv_a[0][0]
__________________________________________________________________________________________________
res_2_conv_b (Conv2D)           (None, 11, 11, 64)   36928       activation_1[0][0]
__________________________________________________________________________________________________
bn_2_conv_b (BatchNormalization (None, 11, 11, 64)   256         res_2_conv_b[0][0]
__________________________________________________________________________________________________
activation_2 (Activation)       (None, 11, 11, 64)   0           bn_2_conv_b[0][0]
__________________________________________________________________________________________________
res_2_conv_copy (Conv2D)        (None, 23, 23, 256)  16640       max_pooling2d[0][0]
__________________________________________________________________________________________________
res_2_conv_c (Conv2D)           (None, 11, 11, 256)  16640       activation_2[0][0]
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 11, 11, 256)  0           res_2_conv_copy[0][0]
__________________________________________________________________________________________________
bn_2_conv_c (BatchNormalization (None, 11, 11, 256)  1024        res_2_conv_c[0][0]
__________________________________________________________________________________________________
bn_2_conv_copy (BatchNormalizat (None, 11, 11, 256)  1024        max_pooling2d_2[0][0]
__________________________________________________________________________________________________
add (Add)                       (None, 11, 11, 256)  0           bn_2_conv_c[0][0]
                                                                 bn_2_conv_copy[0][0]
__________________________________________________________________________________________________
activation_3 (Activation)       (None, 11, 11, 256)  0           add[0][0]
__________________________________________________________________________________________________
res_2_identity_1_a (Conv2D)     (None, 11, 11, 64)   16448       activation_3[0][0]
__________________________________________________________________________________________________
bn_2_identity_1_a (BatchNormali (None, 11, 11, 64)   256         res_2_identity_1_a[0][0]
__________________________________________________________________________________________________
activation_4 (Activation)       (None, 11, 11, 64)   0           bn_2_identity_1_a[0][0]
__________________________________________________________________________________________________
res_2_identity_1_b (Conv2D)     (None, 11, 11, 64)   36928       activation_4[0][0]
__________________________________________________________________________________________________
```

bn_2_identity_1_b (BatchNormali	(None, 11, 11, 64)	256	res_2_identity_1_b[0][0]
activation_5 (Activation)	(None, 11, 11, 64)	0	bn_2_identity_1_b[0][0]
res_2_identity_1_c (Conv2D)	(None, 11, 11, 256)	16640	activation_5[0][0]
bn_2_identity_1_c (BatchNormali	(None, 11, 11, 256)	1024	res_2_identity_1_c[0][0]
add_1 (Add)	(None, 11, 11, 256)	0	bn_2_identity_1_c[0][0] activation_3[0][0]
activation_6 (Activation)	(None, 11, 11, 256)	0	add_1[0][0]
res_2_identity_2_a (Conv2D)	(None, 11, 11, 64)	16448	activation_6[0][0]
bn_2_identity_2_a (BatchNormali	(None, 11, 11, 64)	256	res_2_identity_2_a[0][0]
activation_7 (Activation)	(None, 11, 11, 64)	0	bn_2_identity_2_a[0][0]
res_2_identity_2_b (Conv2D)	(None, 11, 11, 64)	36928	activation_7[0][0]
bn_2_identity_2_b (BatchNormali	(None, 11, 11, 64)	256	res_2_identity_2_b[0][0]
activation_8 (Activation)	(None, 11, 11, 64)	0	bn_2_identity_2_b[0][0]
res_2_identity_2_c (Conv2D)	(None, 11, 11, 256)	16640	activation_8[0][0]
bn_2_identity_2_c (BatchNormali	(None, 11, 11, 256)	1024	res_2_identity_2_c[0][0]
add_2 (Add)	(None, 11, 11, 256)	0	bn_2_identity_2_c[0][0] activation_6[0][0]
activation_9 (Activation)	(None, 11, 11, 256)	0	add_2[0][0]
res_3_conv_a (Conv2D)	(None, 11, 11, 128)	32896	activation_9[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 128)	0	res_3_conv_a[0][0]
bn_3_conv_a (BatchNormalization	(None, 5, 5, 128)	512	max_pooling2d_3[0][0]
activation_10 (Activation)	(None, 5, 5, 128)	0	bn_3_conv_a[0][0]
res_3_conv_b (Conv2D)	(None, 5, 5, 128)	147584	activation_10[0][0]
bn_3_conv_b (BatchNormalization	(None, 5, 5, 128)	512	res_3_conv_b[0][0]
activation_11 (Activation)	(None, 5, 5, 128)	0	bn_3_conv_b[0][0]
res_3_conv_copy (Conv2D)	(None, 11, 11, 512)	131584	activation_9[0][0]
res_3_conv_c (Conv2D)	(None, 5, 5, 512)	66048	activation_11[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 5, 5, 512)	0	res_3_conv_copy[0][0]
bn_3_conv_c (BatchNormalization	(None, 5, 5, 512)	2048	res_3_conv_c[0][0]

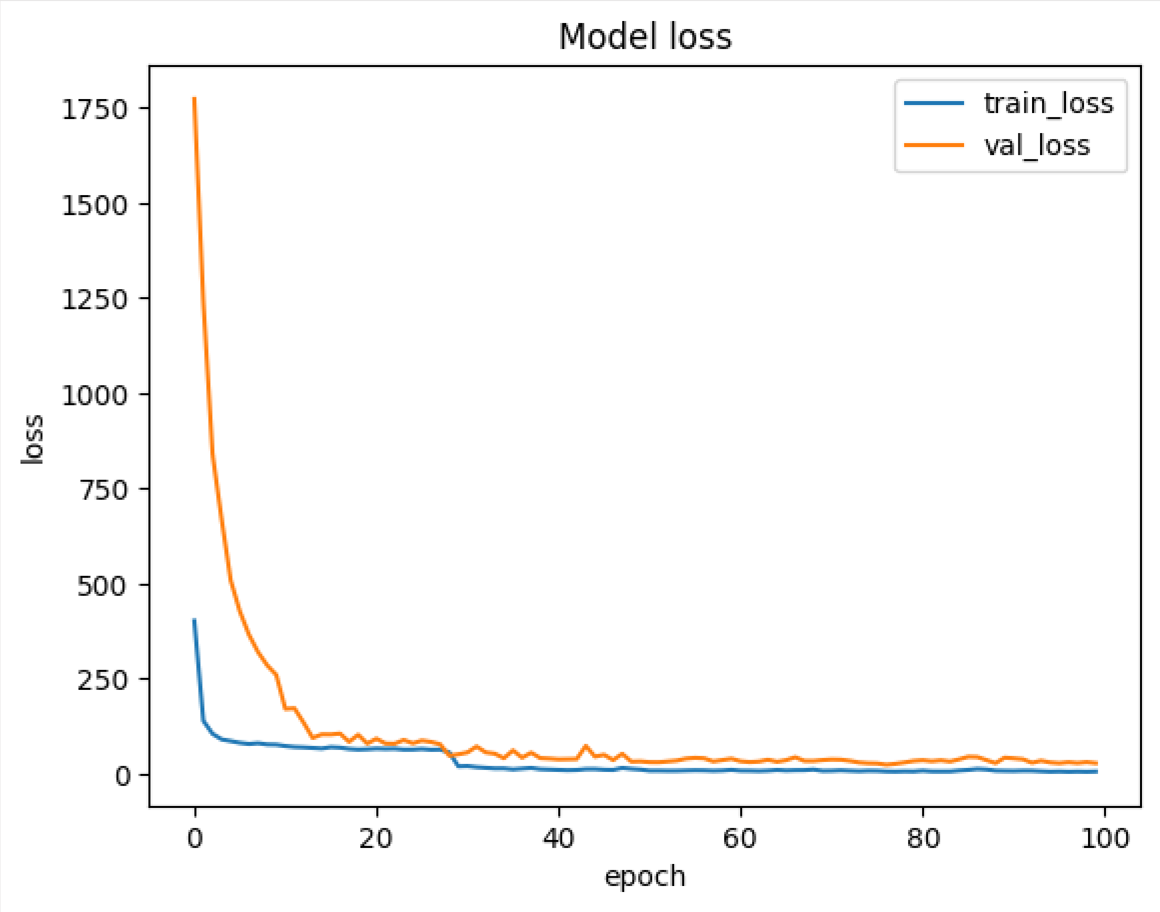
bn_3_conv_copy (BatchNormalizat	(None, 5, 5, 512)	2048	max_pooling2d_4[0][0]
add_3 (Add)	(None, 5, 5, 512)	0	bn_3_conv_c[0][0] bn_3_conv_copy[0][0]
activation_12 (Activation)	(None, 5, 5, 512)	0	add_3[0][0]
res_3_identity_1_a (Conv2D)	(None, 5, 5, 128)	65664	activation_12[0][0]
bn_3_identity_1_a (BatchNormali	(None, 5, 5, 128)	512	res_3_identity_1_a[0][0]
activation_13 (Activation)	(None, 5, 5, 128)	0	bn_3_identity_1_a[0][0]
res_3_identity_1_b (Conv2D)	(None, 5, 5, 128)	147584	activation_13[0][0]
bn_3_identity_1_b (BatchNormali	(None, 5, 5, 128)	512	res_3_identity_1_b[0][0]
activation_14 (Activation)	(None, 5, 5, 128)	0	bn_3_identity_1_b[0][0]
res_3_identity_1_c (Conv2D)	(None, 5, 5, 512)	66048	activation_14[0][0]
bn_3_identity_1_c (BatchNormali	(None, 5, 5, 512)	2048	res_3_identity_1_c[0][0]
add_4 (Add)	(None, 5, 5, 512)	0	bn_3_identity_1_c[0][0] activation_12[0][0]
activation_15 (Activation)	(None, 5, 5, 512)	0	add_4[0][0]
res_3_identity_2_a (Conv2D)	(None, 5, 5, 128)	65664	activation_15[0][0]
bn_3_identity_2_a (BatchNormali	(None, 5, 5, 128)	512	res_3_identity_2_a[0][0]
activation_16 (Activation)	(None, 5, 5, 128)	0	bn_3_identity_2_a[0][0]
res_3_identity_2_b (Conv2D)	(None, 5, 5, 128)	147584	activation_16[0][0]
bn_3_identity_2_b (BatchNormali	(None, 5, 5, 128)	512	res_3_identity_2_b[0][0]
activation_17 (Activation)	(None, 5, 5, 128)	0	bn_3_identity_2_b[0][0]
res_3_identity_2_c (Conv2D)	(None, 5, 5, 512)	66048	activation_17[0][0]
bn_3_identity_2_c (BatchNormali	(None, 5, 5, 512)	2048	res_3_identity_2_c[0][0]
add_5 (Add)	(None, 5, 5, 512)	0	bn_3_identity_2_c[0][0] activation_15[0][0]
activation_18 (Activation)	(None, 5, 5, 512)	0	add_5[0][0]
Averagea_Pooling (AveragePoolin	(None, 2, 2, 512)	0	activation_18[0][0]
flatten (Flatten)	(None, 2048)	0	Averagea_Pooling[0][0]
dense (Dense)	(None, 4096)	8392704	flatten[0][0]
dropout (Dropout)	(None, 4096)	0	dense[0][0]

```
________________________________________________________________________________________________
dense_1 (Dense)              (None, 2048)        8390656     dropout[0][0]
________________________________________________________________________________________________
dropout_1 (Dropout)           (None, 2048)        0          dense_1[0][0]
________________________________________________________________________________________________
dense_2 (Dense)              (None, 30)          61470       dropout_1[0][0]
================================================================================================
=
Total params: 18,016,286
Trainable params: 18,007,710
Non-trainable params: 8,576
________________________________________________________________________________________________
```

Results:

I expect that my model will achieve state-of-the-art accuracy on the Facial Keypoints Detection dataset. I will also visualize the predictions of my model on some test images to show how well it can identify the key points on faces.

Model History and Predictions:



Conclusion:

In this project, I built a deep learning model to detect facial key points from images. I trained the model for 100 epochs on the Facial Keypoints Detection dataset and achieved 80% training accuracy and 79% validation accuracy. This means that my model is able to accurately identify facial key points in most cases, but there is still room for improvement.

In the future, I plan to improve the accuracy of my model by using a larger dataset, training the model for a longer number of epochs, and incorporating more advanced techniques. I believe that these improvements will help me to achieve state-of-the-art accuracy on the Facial Keypoints Detection dataset. I am also excited to explore the potential applications of my model for other tasks, such as face recognition, emotion analysis, and face swapping.