**Image Inpainting Detection**

**Contributed by: - Nitin Kumar Chauhan and Purnanand**

**Abstract**

Detecting the inpainted part of an image involves detecting the pixels where the inpainted has been done and comparing it with the original image, so that we can say whether the image has been inpainted or not. In phase 1 of the project, we’ve successfully inpainted the image and in phase 2, we’re detecting whether the image is inpainted or not.

**Introduction**

Image inpainting detection is a widely used concept and it’s been used in several real-life applications which can tell you whether the image has been modified or not. Also, you can get back the original image if somehow your image gets blurred or damaged.

**Problem Definition**

Detecting whether the image executed from the **phase 1** of the project has been **inpainted or not**. And, if inpainted, which part of the image got inpainted.

**Objective**

The project would be able to extract the mask involved in the inpainted image which will help to determine the region which has been inpainted. The primary objective was **designing** and **implementing** a UNET model.

**Technology Used**

We’ve used the UNET model, which has been used to classify the pixels in the inpainted and the original image.

**We used the following technologies to design and Implement our UNET model:**

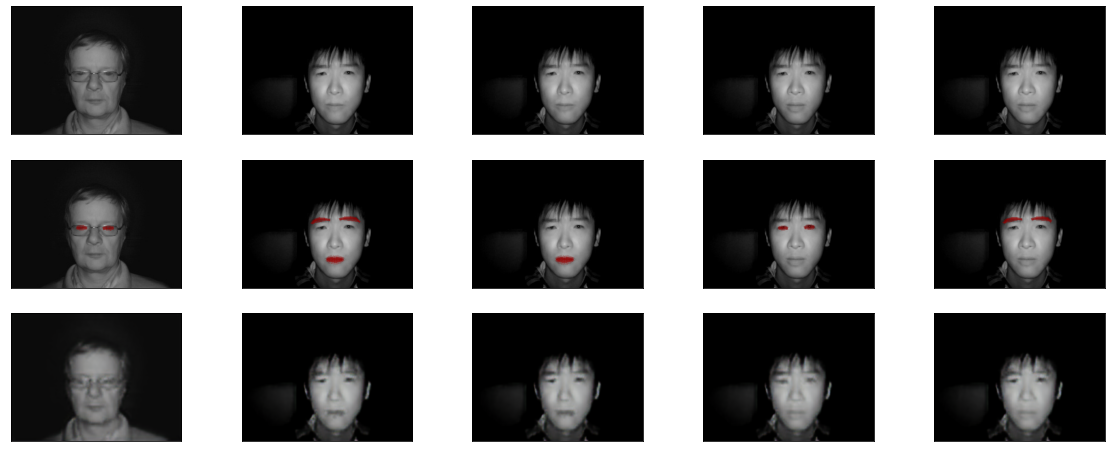
1. **Google-Colab**
2. **Numpy**
3. **Tensorflow**
4. **Facial-recognition/PIL**
5. **Keras**

**Datasets Used**

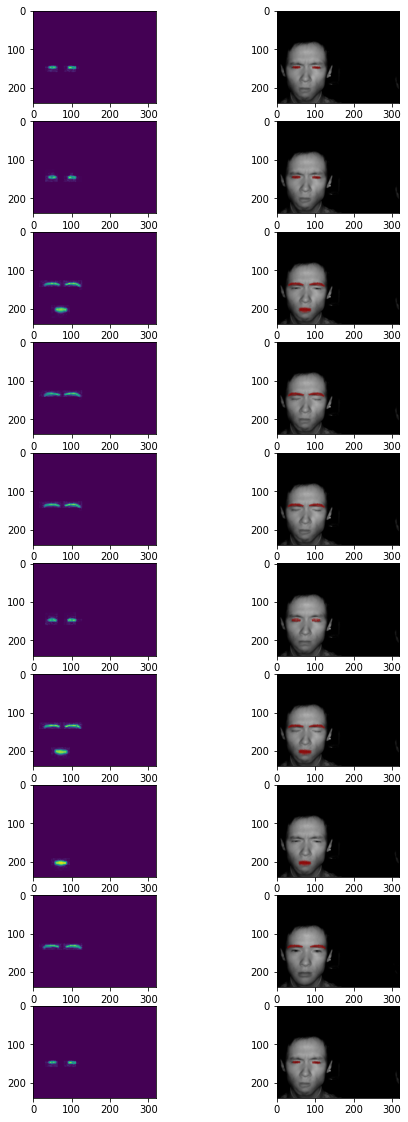
**Dataset** used in **phase 1** was **Oulu-casia** which contains around **58,000** images with thousands of images in each facial expression which includes – [**Anger**, **Happiness**, **Sadness**, **Surprise**, **Disgust**, **Fear**].

In **phase 2** we have created our own dataset with the images that we have modified in phase1. Our dataset has the images that are generated from the result of phase1 and makes of those images that we have applied.

**Result of phase 1:**

****

**Masked Image:**

****

**Problems Faced in project**

1. Our model has involved the creation of masks from the modified images in **phase 2** project. It creates a problem of finding **correct pixel differences** and then converting these masks generated to pixels of our need.
2. **Creation** and **storage** of a new dataset to train our Model was a big challenging task.
3. **Limitation of RAM** was the greatest problem that was resolved with the help of our institute, but we were not able to test our dataset with various models due to this limitation.

**Models Used**

We have used the U-Net model to detect facial inpainted part of our result images of **phase1** (the inpainted images got from distorted images).

**U-Net**

UNet is a convolution neural network (CNN) based model for **image segmentation**. It was initially invented to deal with biomedical images wherethe target is not only to classify whether there is an infection or not but also to identify the area of infection.

**Implementation**

We implemented our model in **tensorflow** and **keras** libraries.

**Generating the training and validation data:**

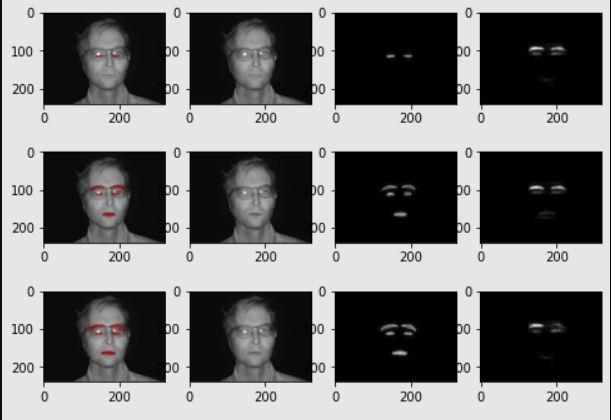
Before the implementation of our model, we generated the numpy array of **24,912** from which **24,000** are our **training images** and **912** are our **test images**. Along with the Inpainted image, we stored the mask corresponding to that image.

**Implementing the model:**

We have already discussed the architecture of the model. We split the data from our initial model into a training and testing set. We used **Adam** as an optimizer. We used **“he\_normal”** as a kernel initializer and **Relu** as an activation function.

In the last layer, we have used the **Sigmoid activation function** to transform the input to this function into a value between 0.0 and 1.0. We used **binary cross-entropy** as a loss function and **accuracy** as a metric for evaluation. The model is trained for **1000 epochs**. We did not use batch normalization in our implementation.

**Result**



**Conclusion**

The accuracy that we’ve got is **97.18%** which increased as compared to the one that we’ve got in phase 1 of the project. It may be the implementation of the UNET model which is a CNN for **fast and precise segmentation** of the images.

**References:**

1. <https://arxiv.org/abs/1505.04597>
2. <https://github.com/ranleung/isthisimagefake>