This project is also listed in github:

<https://github.com/tdchua/tim_potter_faceswap>

This link is quite a helpful website in understanding how does a faceswap model achieve its function and the concepts that go along with it.

<https://forum.faceswap.dev/viewtopic.php?t=146>

[03-26-20]:

The result of the training can be seen here:

<https://imgur.com/a/ppo5z7N>

As you can see the decoded image looks vaguely similar to the input.

I am pinning this problem on maybe a faulty dataset(a lot of images are actually blurry to begin with). Another problem could be that the model is too simple to produce realistic images.

I am now going to check my data.

[03-27-20]:

Training data such as these will be removed.

A blurry photo of a person making a face for the camera

Description automatically generated  A blurry photo of a person

Description automatically generated

I realized I was able to find these types of blurry photos quicker by sorting the pictures by file size. Generally, the less the file size, the more blurry and pixelated it is.

I initially started with around 1100 photos of Harry Potter. After filtering the blurry photos out, the number of photos left is around 620 photos. It was indeed a huge oversight not checking at the raw photos. I will proceed on filtering my own photos.

The method of gathering photos initially which might have included the low quality ones was using “Image Downloader” extension on Chrome, and then browsing google images. This might have been a mistake.

[10:29PM]

Training Sample Count

Harry Potter: 1082

Tim: 1000

[03-28-20]:

Github commit branch: [399c75c](https://github.com/tdchua/tim_potter_faceswap/commit/399c75cd43c06458db592eac2b0ceec96dddf4c0)

I finished training my simple Keras model from the examples shown on their website.

<https://blog.keras.io/building-autoencoders-in-keras.html>

I chose the Convolutional autoencoder, because I had an idea on how it operates given that I took the courses of Andrew Ng.

input\_img = Input(shape=(image\_size,image\_size,3))

x = Conv2D(16, (3, 3), activation='relu', padding='same', name='enc\_conv2d\_1')(input\_img)

x = MaxPooling2D((2, 2), padding='same', name='enc\_maxpool\_1')(x)

x = Conv2D(8, (3, 3), activation='relu', padding='same', name='enc\_conv2d\_2')(x)

x = MaxPooling2D((2, 2), padding='same', name='enc\_maxpool\_2')(x)

x = Conv2D(8, (3, 3), activation='relu', padding='same', name='enc\_conv2d\_3')(x)

encoded = MaxPooling2D((2, 2), padding='same', name='enc\_maxpool\_3')(x)

x = Conv2D(8, (3, 3), activation='relu', padding='same', name='dec\_B\_conv2d\_1')(encoded)

x = UpSampling2D((2, 2), name='dec\_B\_upsampl\_1')(x)

x = Conv2D(8, (3, 3), activation='relu', padding='same', name='dec\_B\_conv2d\_2')(x)

x = UpSampling2D((2, 2), name='dec\_B\_upsampl\_2')(x)

x = Conv2D(16, (3, 3), activation='relu', padding='same', name='dec\_B\_conv2d\_3')(x)

x = UpSampling2D((2, 2), name='dec\_B\_upsampl\_3')(x)

decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same', name='dec\_B\_conv2d\_4')(x)

The model produce these pictures from my inputs

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| A person wearing glasses and smiling at the camera  Description automatically generated | A close up of a mans face  Description automatically generated |
| A person wearing glasses and smiling at the camera  Description automatically generated | A picture containing sitting, blurry, kitchen, bird  Description automatically generated |
| A person wearing glasses and smiling at the camera  Description automatically generated | A blurry image of a person  Description automatically generated |

As you can see, my output picture bares an uncanny resemblance to Harry Potter..

Who am I kidding? The photo does not show harry potter in any facet.

If anything, it only blurred the photo and warped my smile which makes it only creepier…

To be fair, this model was demonstrated on MNIST pictures meaning just digits from 0 -> 9. It was able to construct the shape of the numbers, but it was too much to expect that a clear face could be generated as well. So, it boils down to finding a better model for the task of autoencoding. Just to compare, this model only has 4,963 parameters.

[11:30AM] I will now try finding a better model that is open sourced.

[11:50AM] I have found a model like this that did not have excellent results, however I think that it is good enough for this application. This model, however, has 70million parameters.

A stark contrast, I think compared to the first model I used. It seems reasonable to believe that to swap faces would take more parameters than producing pictures. The link to this article is found below:

<https://medium.com/gradientcrescent/deepfaking-nicolas-cage-into-the-mcu-using-autoencoders-an-implementation-in-keras-and-tensorflow-ab47792a042f>

“Our implementation relies on a simple autoencoder without the use of a GAN component, and is based on a [simplified implementation](https://github.com/OValery16) of the FaceSwap-[GAN repository by Lu et al](https://github.com/shaoanlu/faceswap-GAN).”

[02:36PM] I have finished coding the new model, however grouped convolutions aren’t supported on CPU. I would have to find a GPU to be able to train this model…

[03:02PM] The model has started training, and I hope that it would bear good fruits!

The model also used a PixelShuffler function that uses Keras which I derived from github

[03-29-20]

[10:05AM] The model is still currently training, because there were two instances I forgot to dump the weights after 3 hours of training.

[11:51AM] Model finished training. Will test it now.

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| A person wearing glasses and smiling at the camera  Description automatically generated | A close up of a device  Description automatically generated |
| A person wearing glasses and smiling at the camera  Description automatically generated | A picture containing bird  Description automatically generated |
| A person wearing glasses and looking at the camera  Description automatically generated | A picture containing bird  Description automatically generated |

Using the new model, I was able to achieve better results. For example, my skin tone became paler and my nose became narrower at the third picture. However, It is far from perfect; and this is performed on 64x64 pixels of images. The pictures still remain blurry, and it is quite frustrating.

The medium article mentioned using pre-trained weights so that the loss would converge faster, but I was not able to acquire the pre-trained weights.

[01:57PM]

I will begin converting the target video into frames. This is the video that will be uploaded on Youtube to satisfy the requirement of the Internship program.

[02:26PM]

I have used an example from StackOverflow that utilizes opencv. However, installing opencv itself was a problem; since “cv2” did not exist for the pip install command. That is why the command should be “pip install opencv-python”.

[03:18PM]

I ran into a problem that was internal to the VideoCapture function of opencv.

After slicing the videos into frames it would then rotate those frames because it did not keep the metadata of the video. I used an iPhone selfie camera for the video; the dimensions should be 1080x1920; but after slicing it becomes 1920x1080.

Luckily apple, allows opening multiple images and rotating them simultaneously to address this issue.

I will now search for an open source face detection that will return the bounding box for my face at each image.˜

<https://github.com/ageitgey/face_recognition>

This facial recognition module will help me find my face in the picture.

The plan is to output the bounding box containing my face. Resize it to 64, feed it into my faceswap, and finally paste the output back into the image.

[05:13PM]

I finished generating the frames with the harry potter faceswap module. I will now splice all the frames into one video.

<https://theailearner.com/2018/10/15/creating-video-from-images-using-opencv-python/>