

POLYJUICE POTION

Timothy Joshua Dy Chua

Table of Contents

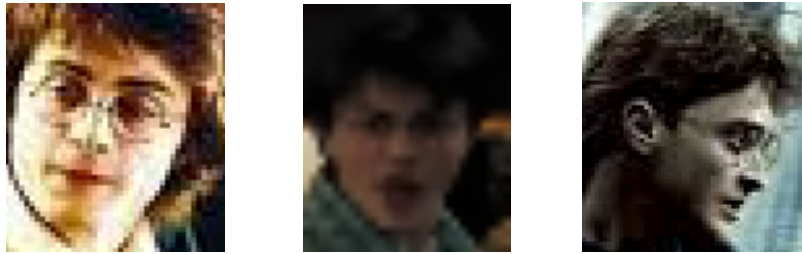
- I. Data Collection (Image Gathering)
 - a. Harry Potter
 - b. Timothy Chua
- II. Model
 - a. Keras – Basic CNN
 - b. Faceswap-GAN Simplified
- III. Experimentation
 - a. Training
 - i. Keras – Basic CNN
 - ii. Faceswap-GAN Simplified
 - 1. BatchSize=32
 - 2. BatchSize=64
 - 3. BatchSize=32, Epoch=100, Training Data : No Glasses
 - b. Testing
- IV. Generating the Video
- V. Files
- VI. Conclusion and Closing Remarks

Data Collection (Image Gathering)

A. Harry Potter

At first, to get these images for training data; I used an extension of Google Chrome called "Image Downloader." Basically, it downloads all of the images found on the active page. So, I went to Google Images and searched for Harry Potter keywords.

After the images have been shown, I simply clicked on the extension and the download began. I only paid attention to the number of training samples I had which was around 1100 at the time. I mean I looked through each photo and then cropped Harry's head, but after some image resizing; it looked really blurry. Some of the images that went into my training data set are found below.



Blurry images of Harry Potter

I didn't notice these at first, because I did not think of looking through my final training set of Harry Potter images. I only thought to look after my first generation of decoded images, since the generated images were blurry. I found out that 620 of my 1100 photos were blurry. This is the reason for the poor results of my model I thought. That is why I began to quickly fix my training data.

I would need better training samples this time, that is why I headed to Youtube. I looked for playlists of Harry Potter's videos from the Prisoner of Azkaban to the Order of the Phoenix. I paused and played the videos, while taking screen crop shots of Harry's face. My training data of Harry, again, grew to about a thousand photos after deleting the blurry photos of the initial collection.

My final Harry Potter with Glasses image count: 1082.

After training with the photos I've collected, the results were not excellent. I am assuming that since my training data has the face with the glasses; it would be hard for the model to generalize. With the help of my partner, I've thus re-collected pictures of Daniel Radcliffe without the glasses this time.

My final Harry Potter without Glasses image count: 1045.

B. Pictures of Myself

I am not a person who likes to take selfies for no reason. I am a person, though, in a long-distance relationship with a girl in Japan. Every time we would start the day and throughout the day, we would send selfies to each other to remind the other of our existence. So, this relationship has so far lasted from December of 2018. This means that I have an abundant number of selfies of my recent look. I just needed the means to aggregate it.

I first started out by downloading each photo one by one into my phone, and then transfer it to my laptop. But, I vaguely remembered something about Facebook allowing us to download all our information. So I requested for all of my message data, and that's where all of my photos were kept.

I again collected training data for my face, and I collected a total of 1000 photos of my face.

My final Timothy Chua with Glasses image count: 1000.

Likewise, after viewing the results; I was not happy. I thought to regather photos of myself this time without eyeglasses. I hope that this time, the algorithm would get accustomed to encoding/decoding my face. I used the burst mode of the iPhone to get all of these pictures extremely fast.

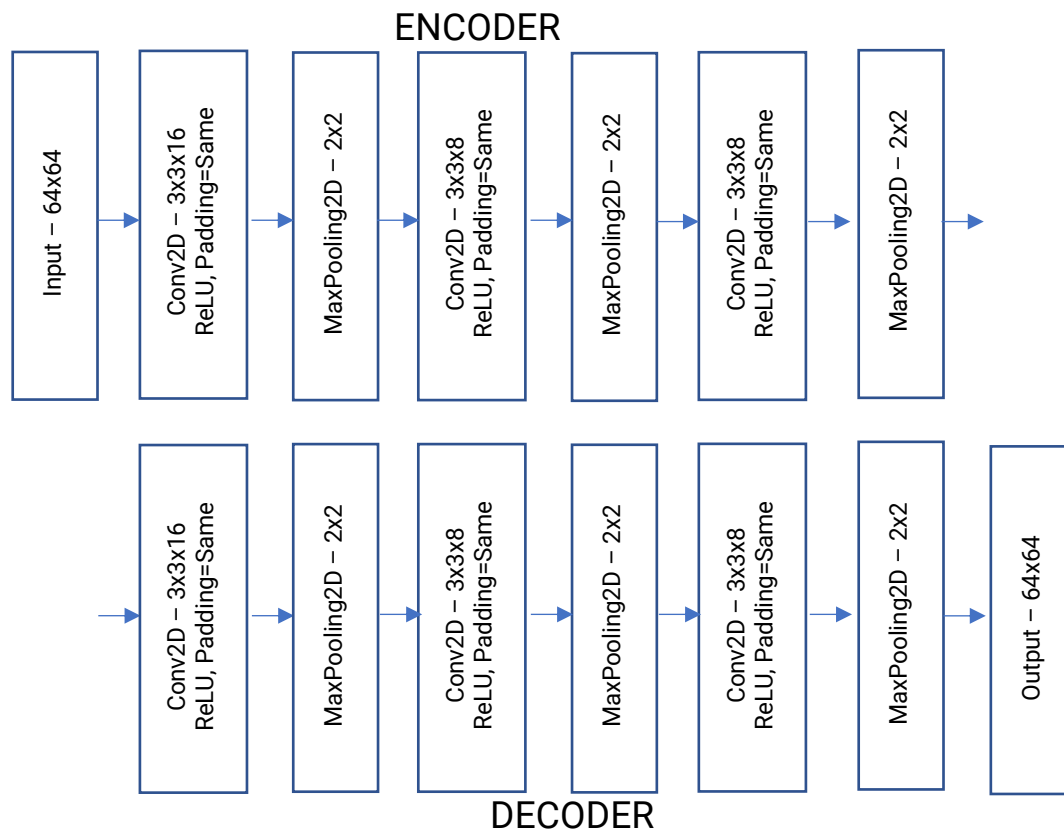
My final Timothy Chua without Glasses image count: 1318.

Model

A. Keras – Basic CNN Autoencoder

I am a beginner when it comes to Machine Learning projects. I can say that this is the first project wherein I'm the one responsible for the Machine Learning part. The first model I chose was the basic CNN encoder-decoder found on Keras's website. This is also the first time I've used Keras; and it's quite beginner-friendly. I tried learning Tensorflow before, but it was quite baffling; that style of programming. That is why Keras appealed to me.

The model provided by Keras to encode and decode MNIST numbers can be seen below. This model had a total of 4,963 parameters.

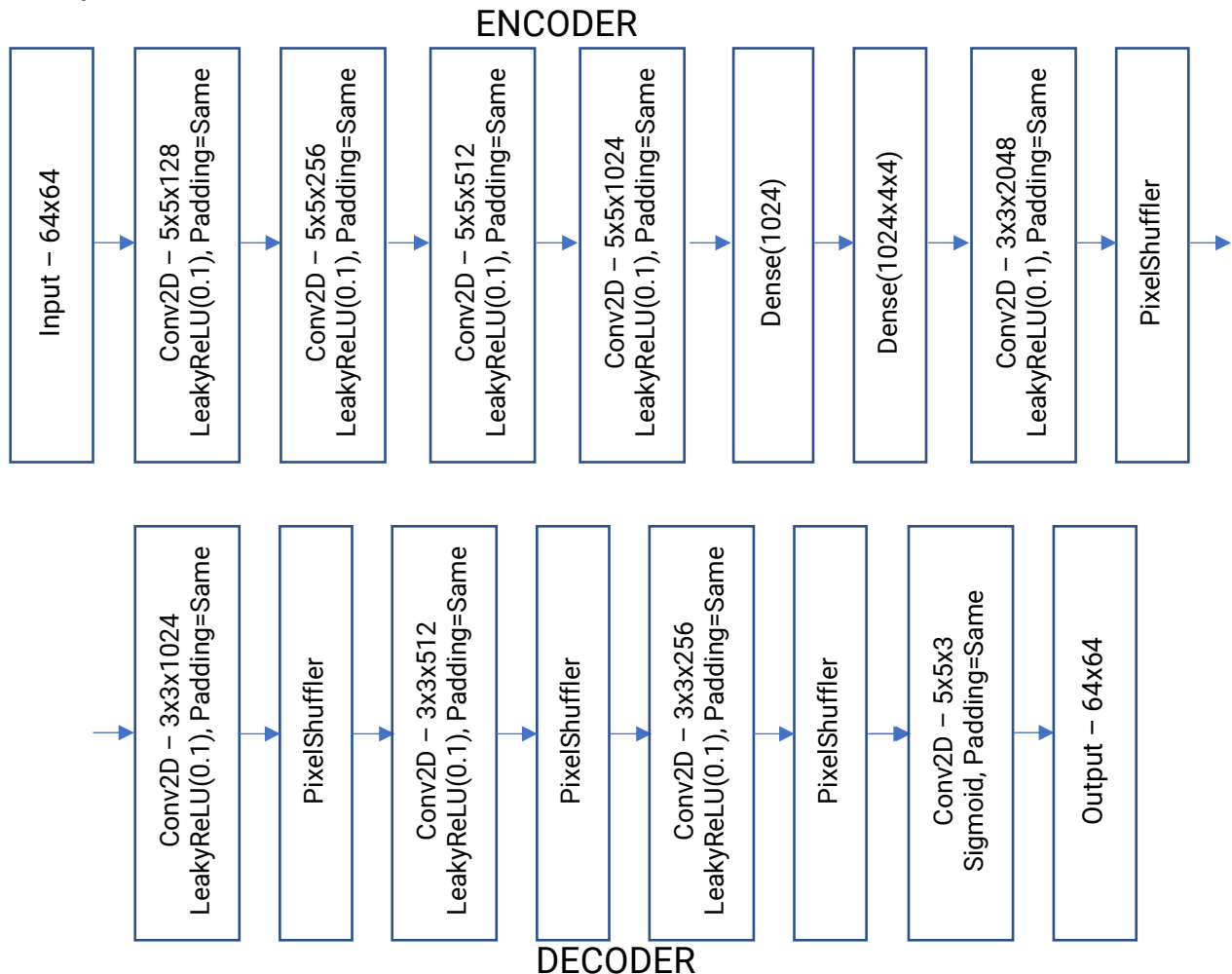


<https://blog.keras.io/building-autoencoders-in-keras.html>

B. Faceswap-GAN Simplified

The real implementation of this model is comprised of an Autoencoder and a Generative Adversarial Network(GAN). However, due to complexity constraint and processing power limitations, I will only be using the Autoencoder following the article from which I am basing on. The removal of GAN would prove harmful to the generated image, because the GAN would be a test of whether the image looks real or not.

Since we are only using an autoencoder for the whole system; the results would not be excellent. Although, this model does not have a GAN component, its number of parameters 70 million far exceed the previous model. This means that in terms of complexity this is a hundred times more complex than basic Keras model.



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

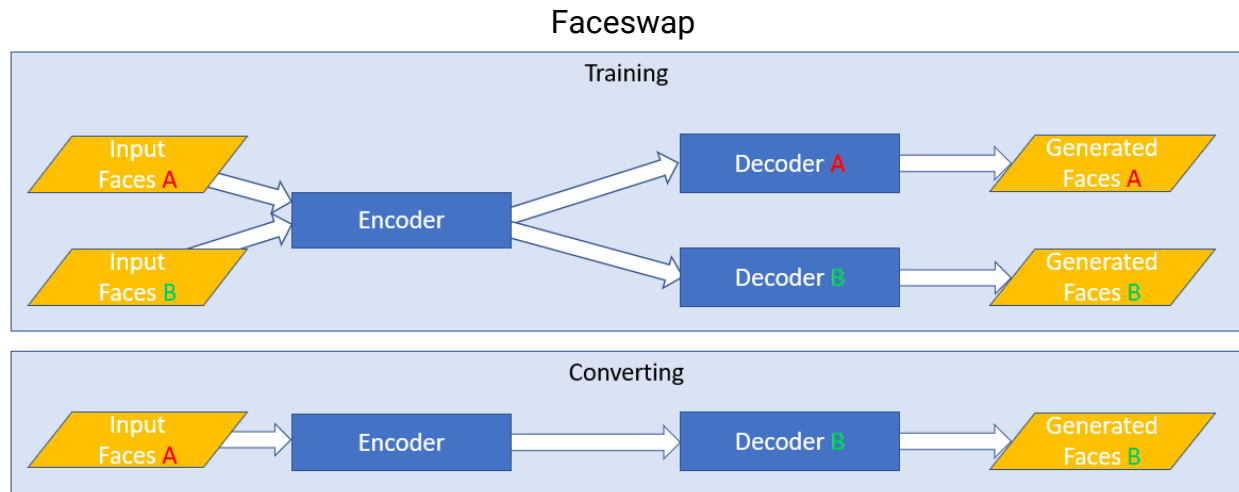
The PixelShuffler has to be found from somewhere else, since Keras was not offering the module in its library. It can be found in this link:

<https://gist.github.com/t-ae/6e1016cc188104d123676ccef3264981>

There was actually an error when I ran this module. It had something to do with *normalize_data_format*. The function could not be found in *keras.utils*, because it was in *keras.backend.common*.

After fixing that part, the model ran fine.

Experimentation



<https://forum.faceswap.dev/download/file.php?id=141&sid=f617aa20026ad3fdb1afce5b2963f43>

A. Training

According to the article linked on the bottom this page, to train a faceswap model. We would need a system shown on the figure above.

First, I would need to train a shared encoder for the two faces—mine and Harry's. There are two stages of the training as seen from the figure above. I would need to first train encoder on my face and couple that with a decoder A. This decoder would have to reconstruct my face from the encoded image given by the encoder. After which, I will, again, train the shared encoder with Harry's face. This time swapping decoder A for decoder B.

To measure the training loss, we would use Mean Absolute Error(MAE) as MAE is used for autoencoders. The decoders A and B will supposedly learn how to accurately produce an uncanny representation of the input depending on how the training goes.

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

B. Testing

Keras – Basic CNN

To train the Keras models, Keras provides a “fit()” function. The parameters for this first basic CNN autoencoder are as follows:

For the **fit()** function:

Epochs=50,

Batch_size=32,

Shuffle=True,


Validation_split=0.25

For the **adadelta optimizer**:

Loss=“mean_absolute_error”

Total training time : ~1hour

For this model, I tried encoding my photos and decoding them to see how it would look. The images do seem to resemble the inputs, but they are unsatisfactory as shown on the table below. I then proceeded to look for other more sophisticated models.

INPUT			OUTPUT		
					
					
					

Test run of Keras Basic CNN encoder and decoder

Keras – Faceswap-GAN Simplified BatchSize=32

After re-coding and choosing a more complex model, the first version of the parameters were

For the **fit()** function:

Epochs=50,

Batch_size=32,

Shuffle=True,

Validation_split=0.25

As prescribed by the article I based my model on, I also used the **Adam optimizer** with parameters:

lr=5e-5,


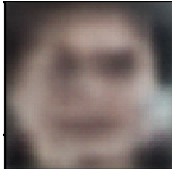
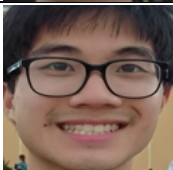
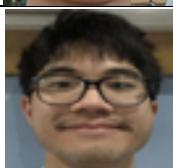
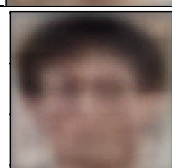
beta_1=0.5,

beta_2=0.999

Loss="mean_absolute_error"

Total training time : ~6hours

After training this new and improved model, I tested how it was able to faceswap my face and Harry's. Note that these faces have glasses, so it is harder to train them.

INPUT			OUTPUT		
					
					
					

Test run of Faceswap-GAN Simplified model, Batch Size = 32

The pictures remained blurry even though a better model was used, and the training set was filtered and recollected. I thought that maybe the reason for this blurriness was the batch size, so on the next run I decided to try it changing the batchsize from 32 to 64.

I also used this model to modify a video to include the faceswapping. The video can be found on this link:

https://www.youtube.com/watch?v=hvw6_dAY3Tk&feature=youtu.be

Please be careful. In the words of Carlson Cheng, a Thinking Machine, it's "nightmare fuel." I agree, unfortunately.

Keras – Faceswap-GAN Simplified BatchSize=64

I again trained the same model but with a different batch size(64).

For the **fit()** function:

Epochs=50,

Batch_size=64,

Shuffle=True,

Validation_split=0.25

As prescribed by the article I based my model on, I also used the **Adam optimizer** with parameters:

lr=5e-5,


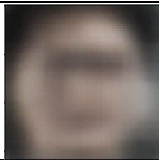

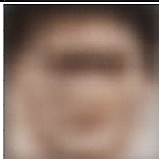
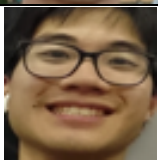
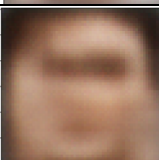
beta_1=0.5,

beta_2=0.999

Loss="mean_absolute_error"

Total training time : ~6hours

Below is the table of the faceswaps.

INPUT			OUTPUT		
					
					
					

The outline of my face can be distinctly seen, although the problem is that the eyes part is quite blurry. I thought that including eyeglasses in the training data of my face and Harry's was the one causing the issue. I then decided to change all of my training data into photos of me and Harry without glasses this time.

I have also used this model to generate a video found on the link below:

<https://www.youtube.com/watch?v=7hjKq75YJQM&list=PLwx27Znn4kiBwWwm8GtjBLwOIlj3bt0Rg&index=2>

Keras – Faceswap-GAN Simplified BatchSize=32, Epoch=100, Training Data: No Glasses

I again trained the same model but with a different batch size(32).

For the **fit()** function:

Epochs=**100**,

Batch_size=**32**,

Shuffle=True,

Validation_split=0.25

As prescribed by the article I based my model on, I also used the **Adam optimizer** with parameters:

lr=5e-5,

beta_1=0.5,


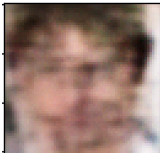


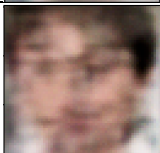
beta_2=0.999

Loss="mean_absolute_error"

Total training time : ~10hours

I noticed that my number of iterations was only around 50. I then researched and came across a publication [1]. It was written that they ran a number of 10,000 iterations on their model for training; so, I changed the number of my iterations to 100. That's the maximum number of iterations I could do in the limited time I had.

Below shows the results of the training.

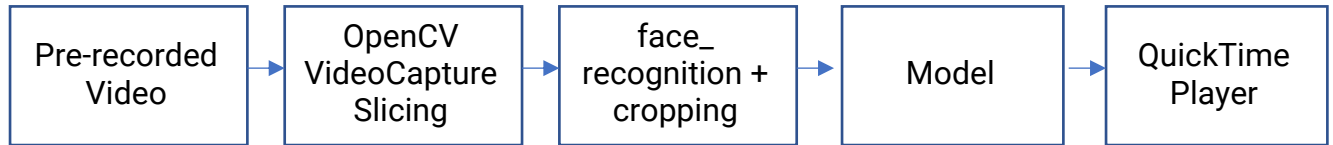
INPUT		OUTPUT	
			
			
			

[1] Fast Face-swap Using Convolutional Neural Networks by Iryna Korshunova, et.al.

I have also used this model to generate a video found on this link:
<https://www.youtube.com/watch?v=ESai9IKWcl0&feature=youtu.be>

I daresay that this video is the most promising out of all the other videos. This could be attributed to the removal of the glasses in the picture. It certainly helped the model learn a much simpler face.

Generating the Video



Pre-recording the Video:

I actually took around 30 minutes to video that ~10seconds of me looking into the camera to check how my appearance was after faceswapping.

This was the target video to use my faceswapping model on.

OpenCV VideoCapture Slicing:

For slicing the video, I use OpenCV's VideoCapture.read(). I actually ran into a problem, because the iPhone that I use embedded metadata which caused my photos to be rotated. Luckily fixing that wasn't a problem thanks to Apple's OS on the laptop.

Face_recognition + Cropping:

To detect and return the bounding boxes of my face in the frames from the sliced video, I used this opensource facial detection module. Found on this link:

https://github.com/ageitgey/face_recognition

The module was able to detect most if not all of my faces in each of the frames, so I was happy with that. It gave me the bounding boxes for the locations of my face, and all I had to do was to crop them.

Model:

Since the input to my model was a 64x64 pixel image, I had to reshape the image down to 64x64. Only then will I be able to feed it into my model. However, my model, so far, has turned my face into something far more sinister and not Harry Potter's face like it was supposed to.

QuickTime Player:

OS Catalina offers an application called QuickTime Player which I used to splice my frames back into the finished product in the form of a video.

Video Link:

<https://www.youtube.com/watch?v=r0Q-a9RTlCI&feature=youtu.be>

Files

I will not be including my data and my model saved weights in the submission, because it would make it a rather huge upload.

I apologize for any inconvenience this may cause.

For running the codes, I used Python 3.7.6

Also, the code can be found in Github:

https://github.com/tdchua/tim_potter_faceswap

-TM_Submission

- scripts

 - rename_files.py

 - resize_images.py

 - slice_video.py

- faceswap.py

- PixelShuffler.py

- list_of_faces

-Documentation

rename_files.py : a script to rename all of my images into a count starting from 1 in the format of <number>.png

resize_images.py : This resizes all the files into 64x64 which is the size the model accepts.

slice_video.py : uses OpenCV to slice the video into manipulatable frames

faceswap.py : contains the Model

PixelShuffler.py : contains the module used in the Autoencoder Model

list_of_faces: contains a list of the faces detected by the face_recognition module.

Documentation.pdf : contains the specifications for this project

Conclusion and Closing Remarks

It is 02:09AM of March 31 as I have started writing this. First and foremost, I would like to thank Thinking Machines for allowing me the opportunity to apply for their internship program. I want to thank my little sister and my partner who have helped me in data collection and video editing.

This is my first Machine Learning project that I've done on my own, and it was, indeed, quite the experience. Data collection was quite difficult and the most time-consuming I think. I have spent the most number of hours in this task collecting photos of me and Harry both with glasses and without. At one part of the project, I mishandled my data and ended up deleting a huge portion of it; so I had to expend time by recovering the lost number of images. I think I have learned to be more careful with data because of this project.

I have also learned that it is quite hard to manage your time if you are working during the day and wanted to relax at night. Although, this project was also fun to work through. It taught me the many facets of creating an ML project from data collection, model selection, and of course, research. I do not think there was any point wherein the code ran at the first try. I had to spend minutes to hours trying to debug and not give up.

Model selection is where I am not so familiar I think. Understanding the concept isn't quite enough, because you would have to choose a capable enough model to fit your project requirements. You cannot simply use a MNIST autoencoder for an application such as Faceswapping. Even if you did select the proper model for the endeavor, it would not amount to anything if the computer you had could not train it fast enough. Also, during training it is of paramount importance to double-check your code before committing to training. You may end up wasting hours because of an incorrect path to your data.

All in all, I found the project quite a challenge and engaging. Even if my model did not meet my expectations, I would still consider this to be a good learning experience for my future projects. I hope that you would allow me to further hone my skills and participate in project development in Thinking Machines. Thank you very much.