ImagSecu: **Challenge 3**

Schult, Julie Elisabeth

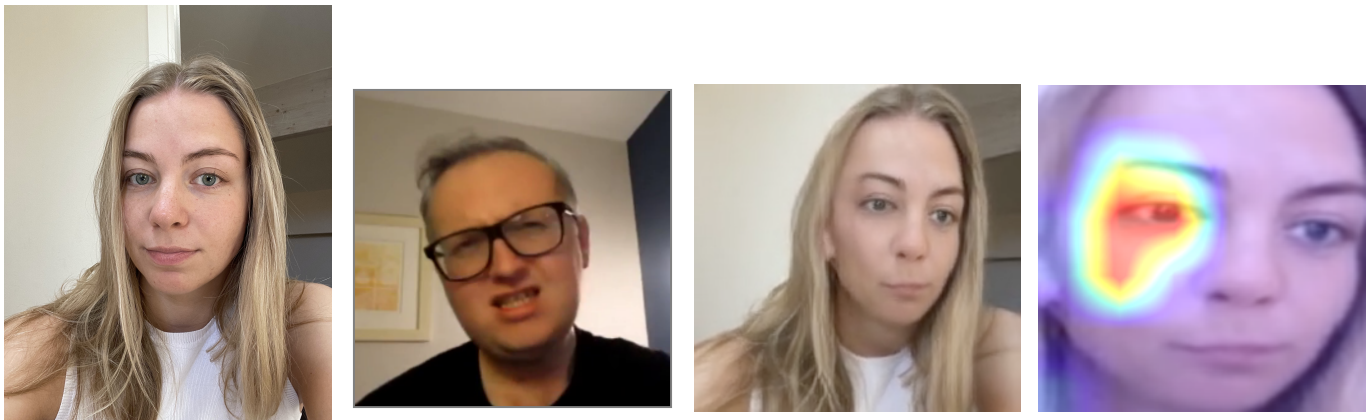
April 20, 2023

1 Task 1: How easy is it to create a deepfake?

The goal of this task is to aim more knowledge about deepfake generation and detection. It has been used a reenactment method to animate a source face image by a driving video. Reenactment is a methos used for recreating an event from the past so it looks like and feels like the original event. In our case we have a photo of ourselves and video clips of famous people giving speeches, and we want to make a video of ourself giving the speech instead of the famous person.

In the task, I have used 4 different selfies of myself having different poses, having different lightning and wearing different accessories. With each selfie I have used 4 driving videos to create deepfake videos.

In Figure 1 I used a selfie of myself where I look slightly to the left, but have some hair in the way so my whole face is not showing. This resulted that the detector detected the deepfake video to be fake with a 93% accuracy. As we see, it detects that it is a fake video the head is turning to the area covered by my hair is showing.



(a) A picture of myself.

(b) Snapshot from the driving video.

(c) Snapshot from the deepfake video.

(d) Results from deepfake detector, (99% fake).

Figure 1: 1st deepfake video.

In Figure 2 I have a selfie with the same pose as in Figure 1, but with a darker lightning and more face showing. The driving video is a speech given by the former American president Barack Obama. The results from the deepfake detector is that the snapshot from the video is real with a 93% accuracy. This is something I found surprising. I have not found a reasonable explanation for this, but this can be beacuse the the video make me look more to the left. If the deepfake video made me look to the right, it could be detected to be fake.

(a) A picture of myself.

(b) Snapshot from the driving video, of Barack Obama.

(c) Snapshot from the deepfake video.

(d) Results from deepfake detector (93% real).

Figure 2: 2nd deepfake video.

In Figure 3 I have a selfie where I look straight forward into the camera and with a baseball cap as an accessory. The video I created the deepfake with was a speech of Donald Trump, where he mostly look straight forward. This was detected as a fake video woth 100% accuracy. It looks like it detected the fake around my eyes.

(a) A picture of myself.

(b) Snapshot from the driving video, of Donald Trump.

(c) Snapshot from the deepfake video.

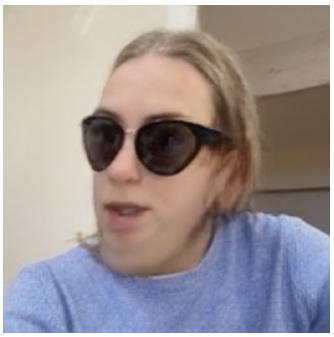
(d) Results from deepfake detector (100% fake).

Figure 3: 3rd deepfake video.

In Figure 4 I have a selfie where I look straight forward into the camera and with a pair of sunglasses as an accessory. The video I created the deepfake with was a speech of Leonardo Dicaprio, where he also mostly look straight forward. This was detected as a fake video woth 99% accuracy. It looks like it detected the fake on my chin, where the deepfake generator also used a part of my throat as my chin.

(a) A picture of myself.

(b) Snapshot from the driving video, of Leonardo Dicaprio.



(c) Snapshot from the deepfake video.

(d) Results from deepfake detector (99% fake).

Figure 4: 4th deepfake video.

To summarize, 3 of the 4 deepfake generated videos gets detected as fake. One of the main reasons why these 3 videos gets detected as fake is when the person driving video turns their head or do certain face expressions, my selfie struggles with following that event and it will therefore not look real. You can think that with accessories the video can appear more reliable as you are hiding more of your face, but that was not the case here so I am not sure.

There are multiple benefits and risks associated with deepfake technology. While seeing deepfake videos can be entertaining and fun to look at, it can also be used in cyberbullying and harassment to attack the person in the photo used. For example, you can make it look like a American democratic politician say it's opinions about a topic, when in real life it is a speech given by a republican politician. Deepfake can also be used for advertising, so make the advertisement more engaging and relatable to the viewer, but on the other hand this can lead to fraud and identity theft to create convincing fake identities. In addition can creating realistic simulations of historical events help students getting a better understanding in the education. But on the other side it can be used to create fake news, propaganda and disinformation. One last risk that is important to mention is created deepfake videos can be created by people without their consent, that is violating their privacy rights.

2 Task 2: From Data to Dollars: Using GANs to Generate Bags for Your Online Shop

The goal of this task is to generate images of handbags for a online fashion store. This is by improving a given base line model, and got doing everything from scratch.

The first step of improving this model is to change the Generator class. In the class, I have added 2 blocks. I did also change the activation function to *Tanh*, as it was shown it is the best function to use for GANs models. The code listing is shown below.

```
class Generator(nn.Module):
    '''
    Generator Class
    Values:
        z_dim: the dimension of the noise vector, a scalar
        im_dim: the dimension of the images, fitted for the dataset used, a scalar
          (fashion MNIST images are 28 x 28 = 784 so that is your default)
        hidden_dim: the inner dimension, a scalar
    '''
    def __init__(self, z_dim=10, im_dim=784, hidden_dim=128):
        super(Generator, self).__init__()
        # Build the neural network
        self.gen = nn.Sequential(
            get_generator_block(z_dim, hidden_dim),
            get_generator_block(hidden_dim, hidden_dim * 2),
            get_generator_block(hidden_dim * 2, hidden_dim * 4),
            get_generator_block(hidden_dim * 4, hidden_dim * 8),
            nn.Linear(hidden_dim * 8, im_dim),
            nn.Tanh()

        )
    def forward(self, noise):
        '''
        A forward pass function for the generator that takes a noise tensor as input and generates
    images as output.
        Parameters:
            noise: a noise tensor with dimensions (n_samples, z_dim)
        '''
        return self.gen(noise)

    # Needed for grading
    def get_gen(self):
        '''
        Returns:
            the sequential model
        '''
        return self.gen
```

The Discriminator class was also changed to improve the model. Like the Generator class, we added 2 more blocks. The code listing is shown below.

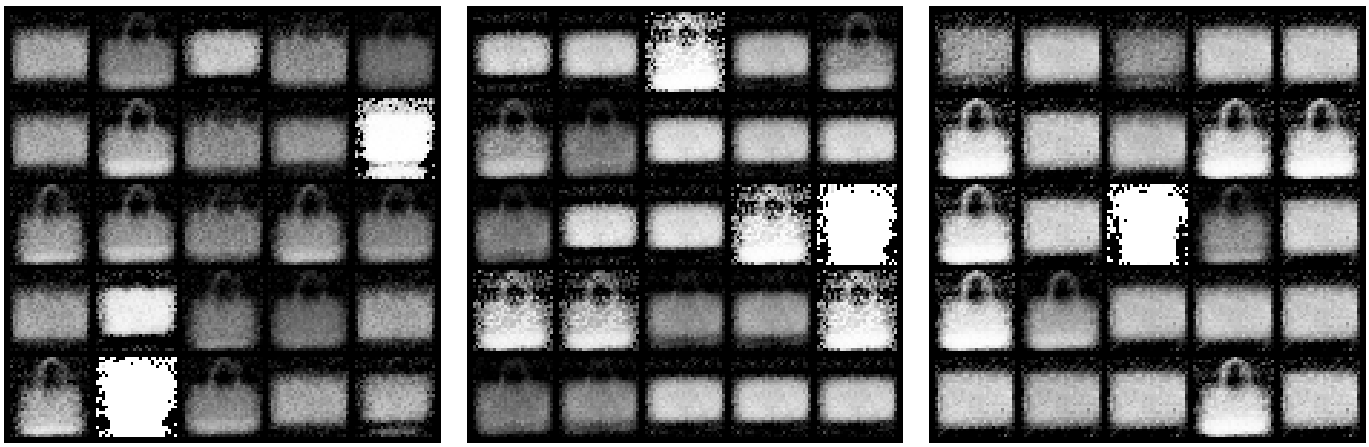
```
class Discriminator(nn.Module):
    '''
    Discriminator Class
    Values:
        im_dim: the dimension of the images, fitted for the dataset used, a scalar
            (MNIST images are 28x28 = 784 so that is your default)
        hidden_dim: the inner dimension, a scalar
    '''
    def __init__(self, im_dim=784, hidden_dim=128):
```

```
        super(Discriminator, self).__init__()
        self.disc = nn.Sequential(
            get_discriminator_block(im_dim, hidden_dim * 8),
            get_discriminator_block(hidden_dim * 8, hidden_dim * 4),
            get_discriminator_block(hidden_dim * 4, hidden_dim * 2),
            get_discriminator_block(hidden_dim * 2, hidden_dim),
            nn.Linear(hidden_dim, 1)
        )

    def forward(self, image):
        '''
        Function for completing a forward pass of the discriminator: Given an image tensor,
        returns a 1-dimension tensor representing fake/real.
        Parameters:
            image: a flattened image tensor with dimension (im_dim)
        '''
        return self.disc(image)

    def get_disc(self):
        '''
        Returns:
            the sequential model
        '''
        return self.disc
```

Finally, the hyper-parameters was changed. By trying a various of different it was found that the number of epochs needed to increase, z dimentions needed to decrease, display step needed to decrease, batch size needed to decrease, batch size needed to decrease and the learning rate needed to be very small. The following hyper-parameters are listed below.

Variable name	Value
n_epochs	30
z_dim	128
display_step	11
batch_size	32
lr	0.001
device	'cpu'

By running the code, we are left with a big amount of images. 3 of these generated images are shown in Figure 5. Here we can see that Figure 5c contains the best images of handbags.



(a) Generated plot 4807. (b) Generated plot 5137. (c) Generated plot 5269.

Figure 5: 3 generated images of handbags.

Figure 6 shows the handbag from Figure 5c that looks like to be the best.

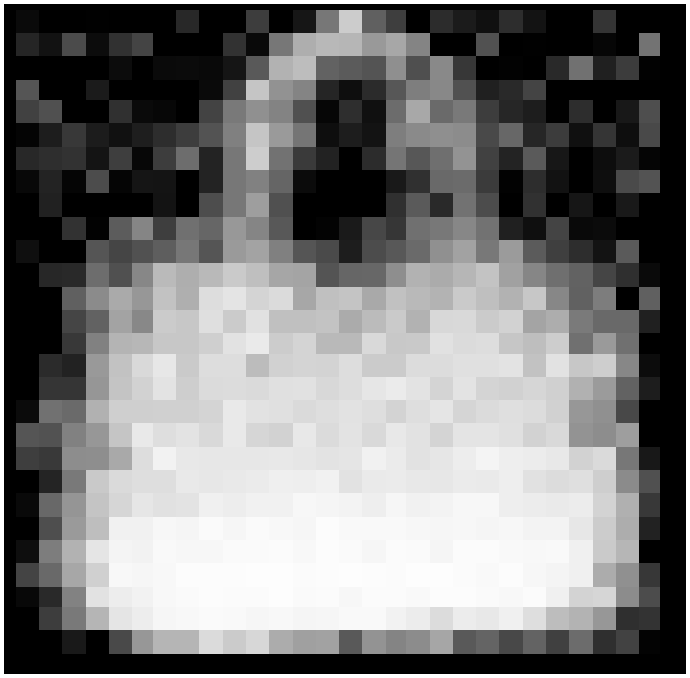


Figure 6: The best handbag.

Figure 7 shows the loss curve plot. We can see here that the discriminator loss (blue) is always low, while the generator loss is decreasing for each step.

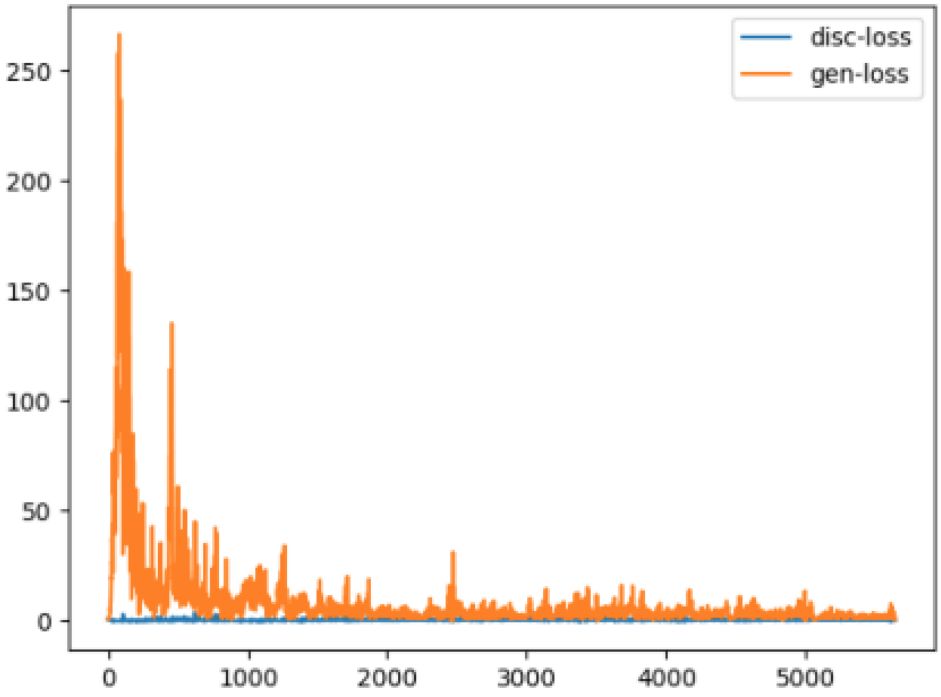


Figure 7: Generator loss and discriminator loss.