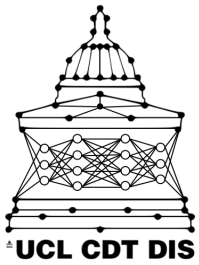


UCL CDT DIS Note

26th May 2020

Cyber-security implications of deepfakes

Francesca Gerardi[a], Nikolay Walters[a], and Tomas James[a]

[a]University College London

Deepfake content — typically a video whose subject has been faceswapped — poses a real and growing threat to online security, both from a political and consumer-centric perspective. To assess the risk that this content has, as well as how straightforward it would be for an attacker to produce deepfaked content, we produce an example deepfake using two leading open-source frameworks, as well as an open-source facial synthesis model before evaluating each framework's results and effectiveness. We subsequently suggest risk mitigation techniques, as well as deepfake detection methods. Our demonstrations show that, whilst technologically impressive, deepfaked content produced on consumer computing hardware are not truly realistic. However, this content can be impactful and pose a threat should it be produced by an experienced and motivated user with access to a high-end GPU. To combat this, we suggest content is marked prominently with a watermark and that detection techniques, such as facial recognition cross-referencing, is used more widely on video hosting services.

Contents

1 Introduction **3**
1.1 The history of deepfakes . . . 3
1.2 Deepfake applications . . . 4
1.3 Deepfakes as a cyber-security risk . . . 4

2 Qualitative deepfake production **5**
2.1 VAE: Variational Auto-Encoder . . . 5
2.2 A basic deepfake . . . 7
2.3 A more advanced example . . . 8

3 Deepfake production workflow **9**
3.1 FaceSwap . . . 11
3.2 DeepFaceLab . . . 12
3.3 Other approaches . . . 13

4 Difficulties and limitations **15**
4.1 Traditional deepfakes . . . 15
4.2 Facial synthesis models . . . 19

5 Results and discussion **20**

6 Combating the deepfake threat **21**
6.1 Current detection methods . . . 22
6.2 Prevention mechanisms . . . 22
6.3 Proposed solutions . . . 23
6.4 Ethical concerns . . . 23
6.5 Detection pipeline . . . 24

7 Conclusion **25**

1 Introduction

It can be argued that nothing defines the 21st century as synonomously as the internet. What began as a communication tool at CERN (Berners-Lee 1989) has outgrown itself to form a generation-defining entity that forms the backbone of our day-to-day lives. Nowhere has this improvement become more evident than in the way we consume media.

The popularity of YouTube and other video hosting sites, as well as the affordability of computing and filming resources, has allowed largely anybody to produce, edit and upload videos on a variety of often unchecked subjects. Furthermore, the popularity and availability of these services has led to more traditional media sources, for example the BBC, using online video as a supplementary method of content delivery.

For the general public this approach allows more access than ever before to global and local events. However, a consequence of this is the availability of material that can be repurposed for malicious intent. This enormous dataset, coupled with the power of desktop computing machines and sophisticated mathematical models, has led to the emergence of a growing threat: the "deepfake".

Deepfakes — a portmanteau of deep-learning and fake — are an increasingly frequent online occurence. In brief, a deepfake is a video that utilises deep-learning to take the face of a target person and implant it over the face of a source within a source video.

1.1 The history of deepfakes

Despite deepfakes quickly becoming part of the 21st Century zeitgeist, their origins can be traced back to academic research within the early 1990s. The first piece of software that could be classed as recognisably deepfake was the Video Rewrite Project (Bregler, Covell and Slaney 1997). This project utilised machine learning to detect phonemes in a training dataset of spoken audio tracks, to which it then applied to dub the subject in to speaking a different audio track.

Despite producing successful and convincing outputs, the Video Rewrite Project had a number of limitations stemming from the computing facilities available in 1997. Research has continued as these resources have improved, and likewise so has the quality of deepfakes produced by these research efforts. For instance, Generative Adversarial Networks (GANs) have been noted as being particularly effective in image synthesis applications. An example of a GAN-based code is the StyleGAN project (Tero Karras et al. 2019), which is responsible for http://thispersondoesnotexist.com.

However, these projects reside within a research domain, largely out of the eye of the general public. Deepfakes weren't truly a part of the public consciousness until 2017 — 20 years after the publication of the Video Rewrite Project — when a subreddit r/deepfakes was created by the user u/deepfake, in turn coining the moniker of "deepfake". Within r/deepfakes users uploaded their own homemade deepfake videos, though this quickly devolved into a hub of celebrity face-swapped pornography. Understandably, Reddit — the subreddit's host — shut r/deepfakes down in 2018. This high-profile closure served to elevate deepfakes up in to the mainstream media's attention.

Since then, deepfakes have rarely been out of the news. Their potential for harm, especially in political events such as presidential elections, has led to technology companies such as Facebook banning deepfakes that aren't apparent to the user on their services (Bickert 2020). However, this is not currently enforced programatically. To achieve this in an online service at scale and on demand, the Deepfake Detection Challenge (AWS et al. 2019) was established in 2019. The challenge, which is still live as of this publication, focuses on detection techniques, with the eventual winner receiving a prize of $500,000.

1.2 Deepfake applications

Despite the negative attention — and history of unpleasant applications — deepfakes do have some legitimate uses. Indeed, the Video Rewrite Project was designed to begin a series of software applications that would be able to dub the cast of a movie to change the movie's language.

Even outside of pure audio dubbing, deepfakes in cinema have a host of potential uses. For instance, stunt people within movies are usually shot such that their faces aren't visible, or are obscured by distance or a camera panning blur. The application of deepfakes to this field would allow greater diversity in the type of shots available that contain stunt people. Furthermore, the "de-aging process" could utilise face-swapping technology to simply swap the face of a performer with a younger version of themselves. A similar approach could be taken to performers that are not able to perform in a production or movie but would still like to consensually take part. These applications would therefore circumvent the need for complex and potentially inaccurate CGI, at least within facial synthesis.

Outside of cinema, the acceptable use cases for deepfakes become more difficult to rationalise. Indeed, their identity-changing nature is suited far more towards criminal activity than it is to legitimate applications, thus posing an inherent cyber security risk.

1.3 Deepfakes as a cyber-security risk

We have already shown in Section 1.1 that deepfakes have been used for unsavoury means. These immoral breaches have, until now, concerned celebrities and people in the public eye. However, the landscape is swiftly moving towards using deepfakes for domestic cyber crime.

2019 saw the world's first deepfake fraud (Stupp 2019) and, as with the deepfake history highlighted in Section 1.1, the beginnings of this cyber crime lie in audio synthesis. This event saw the CEO of a British energy firm being deceived in to paying $240,000 to a 'contractor' by an AI-synthesized voice that imitated their superior.

As of the date of this publication, this event is the only example of an AI-mediated crime. However, Kirat, Jang and Stoecklin (2018) demonstrated a malware attack called DeepLocker that used AI to conceal itself until it reached its intended host. Worringly, DeepLocker was able to identify its target by facial and voice recognition, thereby opening the door on a new landscape of AI-powered cyber-crime.

The possibility also exists that an attacker could work in a similar manner to that highlighted in Stupp (2019) but instead utilise deepfake technologies to produce a video call. This type of crime could take many forms, from requesting money transfer for unpaid bills to calling and imitating

a member of a bank or other utility provider requesting access to a consumer's account. Less rooted in finance but still as devastating is the possibility of an attacker producing forged video of a member of the public committing a crime, or saying something they wouldn't otherwise have said.

AI-powered cyber-crime is an arms race, with enforcement agencies racing to identify potential attack methodologies before attackers can implement them successfully. In reality, stopping an attack before it has begun — essentially intelligence gathering — won't always be possible. The scope of this problem is also enormous: the sheer wealth of information available about almost anybody online via social networks makes anybody a viable target of an AI-based cyber-crime. As such, the requirement for methods of detecting these crimes, be it deepfake or otherwise, in situ are more important now than ever.

This report will focus on efforts to produce a viable and convincing deepfake using readily available, consumer-computing resources. Our objective is to determine just how straightforward it would be for these deepfakes to be produced at home, largely on demand. We test this by attempting to swap the faces of Daniel Craig in *Casino Royale* with that of Matt Lewis, Technical Director at NCC Group. Our focus was doing this with open source software, specifically DeepFaceLab and FaceSwap, in order to better replicate the tools that a member of the public would have. We also investigate more bespoke tools in the Speech Driven Facial Animation code to determine whether advanced facial synthesis codes pose a viable threat as well.

2 Qualitative deepfake production

In brief, an algorithm constructed to produce a deepfake (DF) video would aim to generate a set of realistic and believable synthetic images or masks representing our target's face in a new environment. Although a number of sophisticated algorithms capable of generating synthetic faces were developed in recent years (Bitouk et al. 2008; Dale et al. 2011; Suwajanakorn, Seitz and Kemelmacher-Shlizerman 2015; Thies, Zollhofer et al. 2016; Korshunova et al. 2017; Suwajanakorn, Seitz and Kemelmacher-Shlizerman 2017; Pham, Wang and Pavlovic 2018; T. Karras, Aila et al. 2018; Kim et al. 2018; Chan et al. 2019; T. Karras, Laine and Aila 2019; Thies, Zollhöfer and Nießner 2019), most of these have not been implemented in publicly available software. Instead, a more straightforward framework loosely based on the UNsupervised Image-to-image Translation (UNIT) networks (Liu, Breuel and Kautz 2017) is realized in the vast majority of open-source DF software. Although multiple modifications are indeed present, often sourced from the latest research developments, the backbone of such software remains the same. In this section we will outline the basic building blocks of a DF network, how they fit together to produce synthetic images and conclude this section by describing a state-of-the-art model used in this investigation.

2.1 VAE: Variational Auto-Encoder

At the core of DF implementations is an auto-encoder. Originally developed as a dimensionality-reduction and feature-learning technique, it played a major part in neural network research for decades (LeCun 1987; Bourlard and Kamp 1988; Hinton and Zemel 1994). An auto-encoder is a type of Artificial Neural Network (ANN) consisting of two sub-networks, an encoder and a

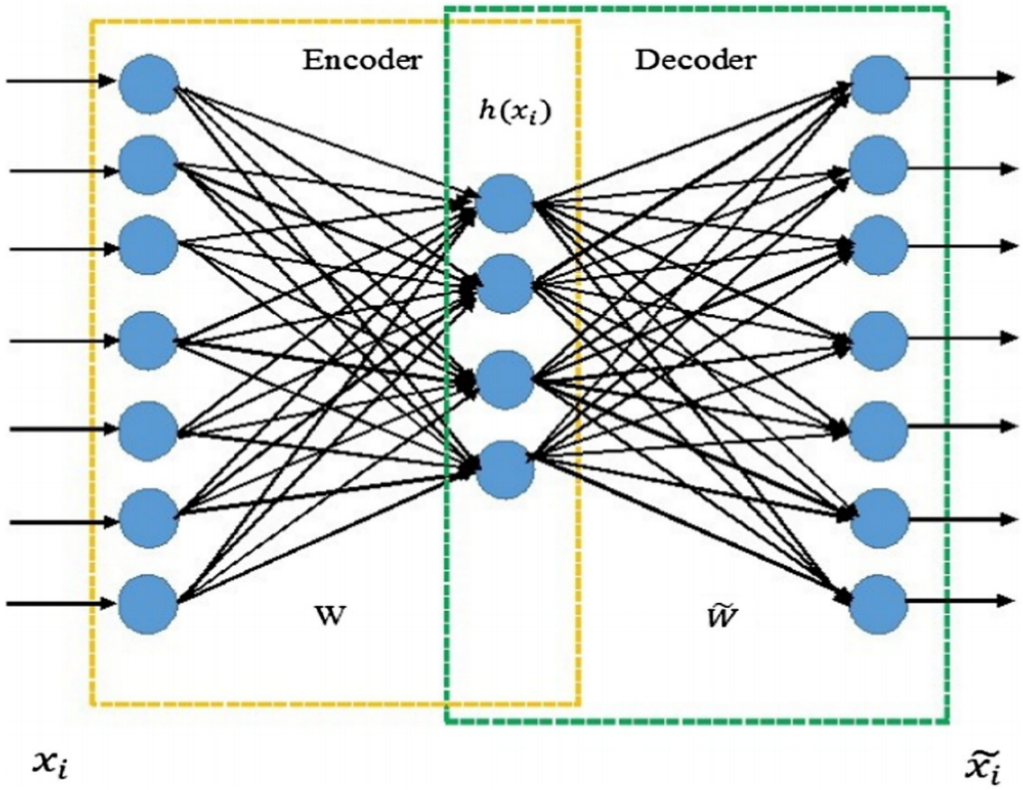


Figure 1: Schematic representation of an auto-encoder network reproduced from Ahmed, Wong and Nandi (2018). The yellow box highlights the encoder subnetwork and green outlines the decoder. The 'bottleneck' can be seen as the shared layer between the two subnetworks.

decoder. Any network with a smaller number of output dimensions than inputs can be considered to be an encoder. A decoder, on the other hand, has less input dimensions than outputs. Connecting the two together, such that the encoder's outputs become the decoder's inputs, producing an auto-encoder network with a 'bottle-neck' in the middle (Fig.1). As we are working with images, the use of convolutional networks (LeCun 1989) for both the encoder and the decoder is advisable.

The convolutional layer examines a subsection of an image (referred to as receptive field) at each step. Consecutive subsections are obtained by shifting a receptive field by a specified number of pixels. Element-wise multiplication is performed between each receptive field and a filter. A close match between a receptive field and a filter results in a large numerical value. Conversely, bad matches would generate values close to or equal to zero. After sliding through a full image and obtaining a single value for each receptive field, one converges on a lower dimensional feature map. Initially, filters are represented by matrices populated with random values, or weights. During the training process, the network learns the appropriate weights for the filters via backpropagation.

Consider a simple image compression task. We can train our auto-encoder to learn a lossy compression; this lossy compression will be used to reconstruct the original, uncompressed image. Passing an image to the encoder results in the encoder being forced to produce a mapping or

encoding to fit the original image in a smaller number of dimensions. This encoding is then read by a decoder that aims to reconstruct the original image from the encoding. This aim is motivated by comparing the original image with the decoded image and the overall performance is quantitively measured by some loss function. By minimizing the reconstruction error, the encoder learns useful lower-dimensional representations of the original data (i.e. feature extraction), while the decoder improves on reconstructing the original image from an encoded representation.

Once training is complete, the network can be separated into encoder and decoder portions. If we pass an image to the encoder, we obtain a compressed, lower-dimensional representation of the image. In short, the encoder created an effective representation of the observed data in a low dimensional 'latent space'. More interestingly however, we can provide an encoding, previously unseen by the network, to the decoder. The decoder will then generate a brand-new image, and thus it begins to act as a generative model. Unfortunately, classical autoencoders do a relatively poor job when it comes to realistic data generation. This is because the decoder struggles to interpolate between large gaps in latent space.

To improve on this, a Variational Auto-Encoder (VAE) (Kingma, Salimans and Welling 2015) was developed. Simply put, instead of an encoder trying to encode an image as a set of discrete coordinates in the latent space, it starts to represent each image as a distribution, where that distribution is often Gaussian by design. This establishes a well-populated volume of the latent space, with each distribution contributing a certain amount based on how close it is to that point. Moreover, a second loss function is often introduced (classically Kullback-Leibler divergence (Kullback and Leibler 1951)) to make the combined distributions appear more Gaussian in the latent space. This way the learned data representation is well-behaved in the latent space, with not many individual distributions too far away from the origin, and somewhat uniform angular density. In other words, we create a well-sampled environment for the decoder, resulting in superior image generation.

2.2 A basic deepfake

VAE might appear useful, but its applications in face-swapping might not be obvious. Consider two sets of images, one representing our target's face (Set A) and another containing faces of our destination (Set B). Our aim is to generate images that look analogous to Set A but match the facial expressions from Set B in order to ultimately replace it. Let us start by creating two VAE networks for each set where the encoder is shared between both sets. The resultant encoding will then be passed to either decoder A if the image is from Set A or to decoder B if the image is from set B.

After these networks have been trained, the encoding from Set A can be passed to decoder A, resulting in the image of our target. However, if we pass an encoding from Set B (i.e. our destination's face) to decoder A (i.e. decoder that was trained on our target) we generate an image of a face that physically resembles our target, but with a facial expression that closely matches the destination's expression.

During training the shared encoder captured identity-independent attributes such as facial expressions, lighting conditions and face angle. Each independent decoder learnt how to effectively reconstruct a physically similar face from these attributes. Thus, on passing our destination's

encoding through a target's decoder it should not be surprising that we obtained an image that resembles our target but displays the same facial expression as the destination.

Thus far we have an unsupervised model that is capable of generating faces of our target that mimics the expression of our destination. All that remains in producing a deepfake is to pass every frame of destination's face through the encoder, record its encoding and feeding it to the target's decoder. This is of course performed automatically by the algorithm.

2.3 A more advanced example

Let us consider a more advanced and realistic DF algorithm that can be used to generate a high quality deepfake. We will focus on the SAEHD (High Definition Styled Auto-Encoder) model from DeepFaceLab with the 'df' architecture. SAEHD consists of two auto-encoders. While one of the auto-encoders retains the original purpose outlined in the previous section, the second aims to capture the "style" of an image. In principle, this stems from the separability of an image's content and an image's style. In reality, images are often affected by a number of nuisances such as lens aberration, motion blur and random noise. These effects can be grouped together as image style, separated from the actual content of an image. Therefore, we can change the image style without having an effect on its content. To generate a realistic DF it is not only important to capture the content, but also to reproduce the style effects within the image such that the deepfaked style matches the destination's environment. The second styled auto-encoder (Ma et al. 2019) receives a low-resolution version of an image. Once encoding of this image is constructed, it is used as an input to the first decoder, transferred using Adaptive Instance Normal (AdaIN) transformation (Gatys, Ecker and Bethge 2016).

"Df" model architecture is usually used when the target has a similarly-shaped face and head compared to the destination. This produces a more natural result since it does not morph faces. However, because of this it does require all of the viewing angles of the destination set. The encoder consists of four downscale blocks. Each downscale block is made of a single convolutional layer paired with a leaky ReLU activation function (Maas et al. 2013). The 'bottle-neck' consists of two fully connected layers and a single convolutional layer, again paired with a leaky ReLU. The decoder is composed of three upscale blocks, followed by four up-down residual blocks. Similarly to downscale blocks, upscale blocks contain convolutional layers that increases image dimensionality. A single up-down residual block is made up of two convolutional layers of the same dimensions, sandwiched between one upscale and one downscale block. Residual blocks have the same purpose as in ResNet (He et al. 2015), so that different training images will train different parts of the network with variable rate, depending on error backpropagation. This is similar to training an ensemble of models, which tends to improve overall accuracy. The network itself is trained via a Learning Rate Dropout optimization algorithm (Lin et al. 2019).

Contrary to popular belief, Generative Adversarial Networks (GANs) (Goodfellow et al. 2014) are seldomly used for DF production. Although GANs produce much sharper images to VAEs, their training process is highly unstable (Goodfellow et al. 2014), with frequent network collapse. This makes GANs comparatively unsuitable for DF software that strives to be user-friendly. However, provided we have trained our model for sufficient time, it would be possible to add a GAN implementation on top, especially when trying to benefit from its positive qualities. As such, we append a GAN to the end of both decoders (i.e. A and B). Adversarial discriminators see the

original images labelled as true and reconstructed images labelled as generated. Therefore, the discriminator learns to distinguish fake images from true, while the generative part of the network tries to generate images that would fool the discriminator, thus making generated images more difficult to distinguish from the true set. The overall loss of the network is computed as a sum between VAE total loss and GAN loss multiplied by a constant. As the model progresses through the training the constant is gradually increased, so that GAN starts to contribute more and more to the overall loss. This way at the early training stages weights and biases are optimized mostly based on VAE's judgement, while at later stages when high resolution of smaller details is required GAN's opinion becomes more significant. The overall training process remains more stable as the unexperienced network is less likely to collapse (i.e. become stuck in a local minima) due to the smaller effect from the GAN.

3 Deepfake production workflow

Logically, the very first step in the deepfake production workflow is choosing two candidates to face swap (target/destination A and B respectively). Next, we collect at least a couple of hundred images/video frames where they appear. These photos/frames should capture different lighting conditions, expressions and faces perspectives in order to maximize the capability of the network to learn how to reconstruct the input targets faces. Indeed, the neural network will only be able to learn what the user supplies as input. For this reason it is important to not only provide several different images but also to furnish possible matching pictures for set A and B.

Once the two sources are selected, the workflow of a DF code consists of three main steps:

- **extraction** of the faces from the video/pictures, for both sources.

  At this stage the code will take as input the video/images collected and extract the faces, where the latter will be automatically detected and stored in an alignment file or encoded into a `.png` that contains the mapping information of a mask to the face landmarks, as shown in Fig.2, for each extracted face.

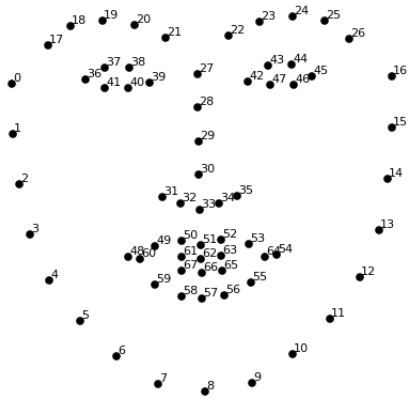


Figure 2: 68 point landmarks. Image taken from `https://forum.FaceSwap.dev/`

  It is essential to note that all the faces which are present in the alignment file are detected automatically via a face-detection algorithm which may result in a partial or complete

mask-face misalignment (see Fig.7). For this reason, before moving forward, it is incredibly important at this stage to check the alignment files and to manually adjust the masks, thereby removing any unwanted faces/facial components, as well as adding any missed faces. This may be a time-consuming process, but it is essential for the success of the swap. This is due to the mask being the main neural network's reference point as it identifies whether a part of the image is face, background or obstruction; without a good alignment even training the network could be a challenge. The extraction must be done for both the faces/frames that will be converted and for those that will be fed into the training neural network. The latter don't necessarily need to be a subset of the former. The training won't be done for a particular target picture/video, but once the model will learn how to swap two faces for which it has been trained it will be able to do that for any picture/video in which one of the two sources is present.

Figure 3: Jack Nicholson and Jim Carrey alignment masks. In light blue there are the lines connecting the 68 landmarks, the blue box is the bounding box, i.e. the area within which are included all faces components identified by the internal landmarks (from n.17 to 67 referring to Fig.2), and the green box is the extraction box, which includes all the face detected. Both these boxes are oriented accordingly to the face.

- **training** of a chosen model.

  Each code comes with its own models, i.e. different neural networks models that may vary by architecture (activation function, hidden layers, hidden units etc) and/or input/output resolution and hence computational cost. The user must input the verified alignment files of the pictures/frames chosen for sets A and B for the training. Then, the user must set a batchsize, choose a training model and then adjust any particular plugin required by the code. While the model is training it is possible to monitor how the learning is proceeding by outputting the preview for the swap and by displaying the learning plot, which illustrates the trend of the loss function over the training epochs. As long as the loss function is decreasing as a function of the number of epochs, the neural network is learning how to map the faces. Throughout the training, it is possible to check how the model is doing not only by looking at the previews, but also by making intermediate back-ups and performing the swap, in order to perceive how the number of epochs affect the conversion, which is the next and final step.

- **conversion** of video/frames of input A, swapping the face with input B (default option), or viceversa.

DF creation takes place via a number of script files. Script files are grouped by the workflow stages, that pass on required environmental parameters to more general python scripts. Python

scripts rely on several libraries and tools, such as TensorFlow (Abadi et al. 2015) for deep learning, FFMpeg for video handling and OpenCL or CUDA frameworks for GPU/CPU execution. Frequently, upon executing a script, a command window can trigger allowing the user to set additional parameters via a command line interface.

Before embarking on deepfaking the *Casino Royale* scene we attempted faceswaps on a variety of subjects first in order to become better acquainted with the available techniques and their limitations. We do this with three open-source frameworks: FaceSwap, DeepFaceLab and a facial synthesis code called SDA (Speech Driven facial Animation).

3.1 FaceSwap

The primary advantage of FaceSwap[1] is that it supports a GUI (Graphical User Interface), making it very user-friendly. Indeed, one can understand the workflow by just looking at the actions menu: extract, train, convert, tools (as in Fig.4). In the Figure, the 'tools' option includes the manual fix to the alignment file, unwanted face removal and many more options.

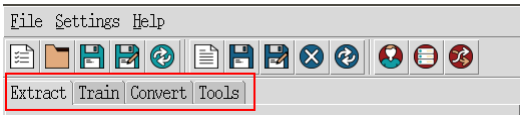


Figure 4: GUI menu. The workflow steps are highlighted in the red rectangle.

Secondly, FaceSwap allows the user to choose an additional mask to be placed on top of the original 68-landmark mask defined during the extraction step. In particular, there are five masks that all differ in properties and purposes. For instance, one is optimized for obstructions, such as hands or glasses, whilst another extends the mask to the forehead in order to avoid the "double eyebrows" problem. The latter is a very common issue that consists of having both source's eyebrows overlapping in the final converted image/frames. As was discussed earlier, it may transpire that alignments need to be manually fixed. In that case the new created mask will comprise the adjusted 68 landmarks, though the user might also want to keep the additional mask. For this reason FaceSwap has a tool called Mask that allows the desired mask to be applied on top of the basic mask without going through the whole extraction process. The mask chosen for the extraction must be then the same for training and conversion.

In the training step, FaceSwap provides ten different models, amongst which it is not so clear and obvious to understand which will work best for the given applications. This is mainly due to the fact that there are many factors involved and each model has its own pros and cons. FaceSwap offers a wide variety of models, from a less computationally intensive model designed to work on most affordable machines to very complex architecture that requires a huge computational effort and even weeks of training.

Moreover, apart from the choice of the specific model, it is possible to easily configure some global plugins such as mask (which must be the one chosen in the extraction phase) and coverage, i.e. the amount of the image fed in the neural network, loss function and learning rate.

Since training is a very hard and costly process, FaceSwap gives the possibility of stopping and resuming it or to recover and restore the model in case it corrupts, as may rarely happen.

[1] `https://github.com/deepfakes/FaceSwap`

The details of FaceSwap are largely discussed in its useful dedicated Forum [2], along with a guide to the workflow's main steps.

3.2 DeepFaceLab

Unlike FaceSwap, DeepFaceLab[3] does not have a GUI, relying solely on a chain of scripts mentioned above. Basic image viewing capabilities are implemented via a portable XNViewMP independent software. A native image editor is also implemented that can be optionally used to define alignment masks interactively and merge learnt faces with a destination video.

Although missing GUI and a mix of script files does impede the users' learning process, numerous steps were taken to simplify the DF creation process. Based on the recent updates it appears that the developer is aiming to simplify the overall workflow by cutting out multiple models, whilst focusing on implementing more features on the few selected models of choice. Shaping the entire software to suit a few models seems more reasonable and thus in the author's opinion is an improvement over previous versions. Although previous architectures and models can still be found by dissecting the software, we highly discourage such an approach as the user is likely to run into many version incompatibility issues that might not be obvious at a first glance. With the latest version of DFL running on 'Leras' (the developer's implementation of Keras (Chollet et al. 2015) with fewer limitations) it seems likely that soon older models will become completely unusable, slicing the number of available models from seven to two. This should be seen as an overall positive to casual users.

From the very first steps, a DFL user is likely to encounter some questionable workflow implementations. DFL requires a very specific environment for the scripts to work, and so any DF creation processes start by running a script that creates a 'workspace' directory with several subdirectories. This is already problematic because a previous project's workspace directory would either need to be renamed or its contents will be erased automatically. As well as this, the source and destination videos would need to be placed in this directory and follow a strict naming convention. Although not having to specify paths and names of files might seem to be a user-friendly approach it does create some difficulties. This means the software is only able to interact with one version of videos, models and images. To prevent older versions from being overwritten they need to be manually backed up by the user away from the active workspace.

Face extraction and mask alignment is performed automatically via the S3FD algorithm (Zhang et al. 2017). Semi-manual interactive extraction is also available. During interactive extraction the user is only allowed to change the size of a region inside which the target's face is situated, as well as set a central point for that region (usually middle of a nose works well). However, it is not possible to automatically set mask landmarks, meaning the user is still relying on the automatic detection algorithm with a search region adjusted. This can be extremely problematic if some environmental artefact confuses the algorithm resulting in incorrect placement of landmarks that cannot be adjusted. If a face is also highly obstructed by some foreign object, face detection might not be possible at all. Whilst the first issue can sometimes be fixed by manually cropping out parts of a mask (i.e. if mask edges are outside of the target area), there is currently no method of manually adjusting incorrect landmark placement inside a mask (i.e. eye landmarks unaligned

[2] `https://forum.FaceSwap.dev/`
[3] `https://github.com/iperov/DeepFaceLab`

with target's eyes) nor to force the software to accept a face that it cannot detect. To add further frustration, homemade interactive editors work in fixed resolution, with images automatically fit to the vertical axis of a screen. Because of this, it is possible to have blind zones: areas to the far left and right of an image that cannot be accessed by the interactive viewer. This can be especially problematic when dealing with footage from movies, with the image's horizontal axis spanning a far greater length than vertical. In such cases, if a target's face is in a blind zone the only hope of extraction is placed upon automatic, non-interactive detection. Practically, users should stay away from using footage with partially covered face frames or faces located close to the left or right edges.

The latest version of DFL provides two models with a choice of two architectures, resulting in four possible training configurations. The Quick96 model should be treated as a test model in order to build an idea of what the final product may look like without employing too many resources. Alternatively, it can be treated as a model dedicated to GPUs limited to 2-4GB VRAM. On the other hand, SAEHD requires at least 6 GB of VRAM and is the model that would be used to produce realistic DFs. We refer the reader to DFL's user guide[4] for more information on training. Finally, is is preferable to leave the network's AE dimensions on default parameters and change batch size and resolution first until all the available memory can be utilised. If the minimum desired batch size and resolution still produces out of memory errors only then should the network's parameters be changed.

After training is complete, the learnt target's face needs to be merged with the destination images. This can be done either interactively, allowing a user to set variable merging parameters for each frame, or automatically by specifying values that will be applied to all frames. DFL's user guide does a good job at explaining these parameters. Other than being time consuming, there are no other drawbacks to interactive merging, since the effect of varying each parameter can immediately be seen. Once a good set of merging values is found it can easily be propagated through the remaining frames. Frames can then be combined into a video using the original audio track. Users can choose between `.mp4` or `.avi` and specify a bitrate.

3.3 Other approaches

As Sections 3.2 an 3.1 have shown, producing a deepfake to a convincing standard is both time consuming and computationally-expensive. These two factors form significant barriers against the more widespread use of malicious deepfakes within society. By introducing limitations in the nature of the deepfake output, as well as releasing "pre-trained" datasets, the runtime of a deepfake code can be reduced significantly. One such example is the Speech Driven Facial Animation (SDA) code[5] (Vougioukas, Petridis and Pantic 2019).

The SDA code shares more in genealogy with the Video Rewrite Project than either DeepFaceLab or FaceSwap. Both DeepFaceLab and FaceSwap attempt to substitute the face of a source on to the destination face within a video, thus producing a typical deepfake. Uniquely, the SDA code takes an image of a target face and a source audio track and synthesizes a video of the face within the still image "speaking" the audio within the audio track. This is distinct from the

[4] https://mrdeepfakes.com/forums/thread-guide-deepfacelab-2-0-explained-and-tutorials-recommended
[5] `https://github.com/DinoMan/speech-driven-animation`

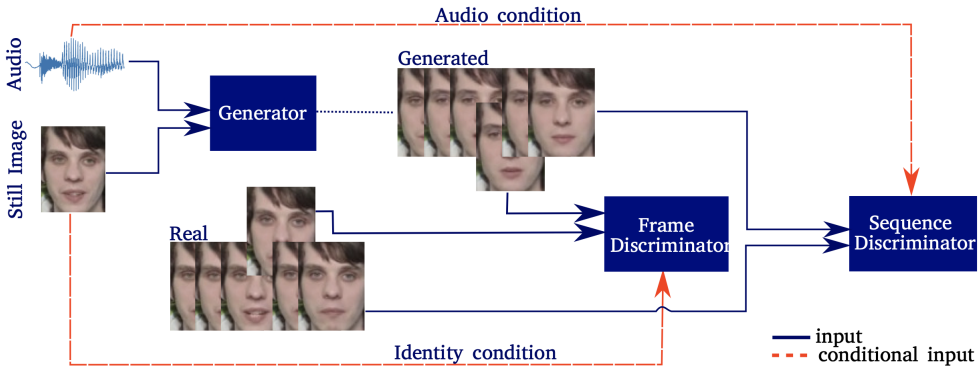


Figure 5: SDA's deep model for facial synthesis. Note the use of 2 discriminators, both of which work in unison to produce SDA's trademark eye blinking and brow movements.

Video Rewrite Project as SDA can synthesize a still source image and does not require a source video.

SDA utilises a GAN (as shown in Fig. 5) based model. The user supplied still image, alongside 0.16s divisions of the audio track, are used as inputs to the RNN-generator (Recurrent Neural Network). This produces a synthetic video of the subject speaking the 0.16s snippet of the audio track. A frame from the output of the generator is then fed to a pre-trained Frame Discriminator, with the objective of fooling the discriminator in to believing that the frame from the generated synthetic video is indeed real.

Should this condition succeed, the full synthetic video is used as an input, alongside the audio track, to a Sequence Discriminator. This discriminator ensures that the full synthetic video — essentially 0.16s chunks knitted together — forms a cohesive, convincing output that is synchronised to the audio. In turn, it also ensures that natural facial movemenets, such as eye blinking, is present within the video.

The SDA code has a number of immediate advantages over traditional deepfake models. Firstly, the discriminators are pre-trained meaning that the end user is not required to perform any time consuming, computationally-expensive training of their own. Furthermore, this advantage runs deeper in that the runtime of the code is around the length of time of the input audio track, rendering it a very fast and efficient code at runtime.

Secondly, the ability to use only an image as an input is wide reaching. A user does not need to have a dataset of video material containing the subject to use as an input, or indeed train their model on, making it relatively straightforward to produce something classified as a deepfake.

Finally, SDA's ability to identify phonemes in the audio means that the output is language independent. Its use of two discriminators is also advantageous in producing the facial movements including blinking, leading to a more natural, convincing result.

4 Difficulties and limitations

4.1 Traditional deepfakes

- Double eyebrows issue, as shown in Fig.6, consists of having both sources eyebrows overlapping in the final converted image/frames, as explained above. This problem might be resolved by choosing an additional mask that pushes the upward limit of the mask up to the forehead.

Figure 6: Attempts to swap Nicholas Cage face to Kate Winslet's. Frame taken from a scene of *Titanic*

- As already recalled in Sec.3, verifying the alignment file created in the extraction process is of vital importance for the success of the whole process. As shown in Fig.7, the first guess of the face is not always successful and it is for this reason that checking and eventually fixing by hand is important. FaceSwap's tool for manually fixing the alignment is once again very user friendly, even though there is no fix option for single face components (i.e. eyes, mouth etc).

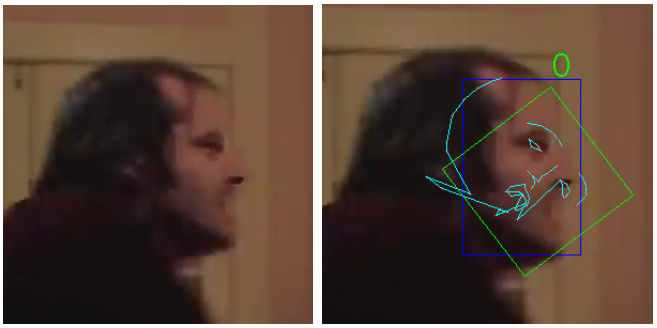


Figure 7: Frame taken from a scene of *The Shining*

- One of the most common challenges in faceswapping is the resolution. Unfortunately, this problem arises at the end of a very time-expensive training rendering it difficult to combat. In FaceSwap there are multiple models available, many of which give the freedom to choose the input and output size of the image, i.e. the resolution. Frustratingly, the higher the resolution the longer the training, meaning that the optimal result is not granted. The challenge is then to find the tradeoff between a sufficiently high resolution and a reasonably long training process. In the case of Fig.8 for the adopted model it can be seen that the smaller face is much better converted than the face on the right as a result of the different face resolutions.

Figure 8: Attempt to swap Jim Carrey face to Jack Nicholson's.Frame taken from a scene of *Shining*

- As a consequence of bad alignments and improperly selected images, occassional third dimension loss is evident. Fig.9 is an example of this where the code overlays a nearly flat face over the original one. This demonstrates how important it is to thoughtfully select the sources images, attempting to match for both the targets the same angles.

Figure 9: Attempt to swap Nicholas Cage face to Kate Winslet's. Frame taken from a scene of *Titanic* on the left and from an interview on the right.

- Visibility of mask edges, i.e. a boundary between superimposed and original face, can impact the realism of a DF by providing obvious visual evidence of video modification. This effect is usually evident by either eroding or diffusing mask edge. A more complex but sophisticated technique is to train another network focused on learning the mask. Even with intelligent color matching, the edges may stand out due to physical features, such as an abrupt end to an eyebrow or inconsistent shading. Fig. 10 shows a noticeable mask boundary on the left that was not blurred effectively. The hadow on the left cheek caused by a slight unmatched facial orientation between the target and destination makes this effect more visible, compared for example with the top boundary near the forehead. On the right, a black patch of the target's hair clearly defines the corner edge of the mask. This is caused by the target's hair inclusion inside a mask when the face was extracted.
- Incorrect color matching (Fig. 11) may also introduce some visible artifacts. Despite the example in Fig. 11 being greatly exaggerated, we can see that the algorithm is darkening the left part of the target's face, thereby simulating a shadow, whilst the right side is much brighter. However, this is perceptively incorrect when studying the destination's face: it is unevenly lit such that direct light illuminates the left side, leaving the right side partially in shadow. Overall, the darkening of the target's face makes it difficult to identify from which direction the light source is applied with multiple inconsistent shadowed regions. This creates a sense of inconsistency when viewed by a human, thus making detection of video altering simple.
- Presence of artificial 3D objects on target's face overcomplicates things by creating direct

Figure 10: Example of suboptimal mask overlay where mask edges are clearly visible.

Figure 11: Example of suboptimal color matching resulting in unnatural shadows and overall brightness.

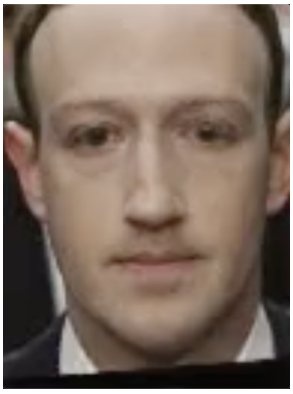
obstructions. Unfortunately, even with somewhat transparent objects — such as glasses — a few issues arise. One of the more striking problems is caused by a 3D object being learnt by the network as a 2D mask representation (Fig. 12). In this case glasses appear flattened or 'tattooed' on the face. This effect would appear especially stark in a profile shot. Part of the glasses would either be missing (if they were outside of a mask) or contain very little depth information.

- Finally, it shouldn't be surprising that a network would struggle with learning dynamical features (such as eyes or teeth) compared to more static ones (such as eyebrows, facial hair or birthmarks). On undertrained models this effect can be rather significant, for example incorrect or inconsistent eye direction is usually an obvious clue. Even after many iterations, certain blurriness in the eye and mouth regions can still present themselves (Fig. 13).

Figure 12: Example of a 3D object being flattened by the mask. Eyeglass temples are abruptly cut off by the mask edge.

Figure 13: Showing poorly-resolved, slightly granular eyes.

(a) The input image to the SDA code of Mark Zuckerberg. Note the high resolution and amount of background visible in the image. This crop was chosen to emulate a video call style of frame as much as possible.



(b) The first frame of the output from the SDA code. In comparison, this image has been cropped aggressively by the SDA code and also downgraded in resolution by a large degree. The alignment has also changed slightly.

Figure 14: Images showing the input to the SDA code and the first frame of the output. Overall, good results are observed despite the decrease in resolution and change in colour accuracy.

4.2 Facial synthesis models

Aggressive crop and resolution change In the case of the SDA code, regardless of the image and image resolution that a user supplies, the model will always crop the image to a close crop of the face, thereby omitting a lot of background material. Alongside this, the model also decreases the resolution significantly to around 96×128 pixels. The combination of these two factors results in a small, pixelated video output. In turn, the aggresive close crop on the face results in a lot of contextual background material being lost, thereby making the output easier to discern as being a deepfake.

Figure 14 shows an example of this in terms of the input image to the SDA code as well as the first frame of the output video.

Immediately obvious when comparing Fig. 14a and 14b is the aggressive crop, alongside the decrease in resolution. The SDA code has also suffered from a degree of bleed-through of the underlying model in Fig. 14b, especially in the regions containing the eyes. The colour of the face has also become paler and lost definition.

Pre-trained datasets Whilst we highlighted the advantage that the pre-training datasets can bring in Section 3.3, this implementation also brings a number of major disadvantages.

Firstly, the SDA code is not supplied with the functionality to train a user's own discriminators. Consequently, the accuracy of the results available is wildly variable depending on the supplied dataset used and the number of datapoints each of those datasets contains.

Secondly, the demonstrated results in Vougioukas, Petridis and Pantic (2019) are highly idealistic as the inputs are representative of the datapoints within the trained datasets. In our tests, we found our results, which were not as representative, were not as accurate or precise in either the facial animations or the number of artifacts visible in the output.

5 Results and discussion

During this investigation we had the unique opportunity to learn applications of DF technology from scratch, potentially following a similar path to an attacker with a range of resources from a conventional desktop PC to a more powerful 12 GB VRAM GPU. Several of the DFs produced by our group can be found at the link below[6]. We find these videos quite impressive from a technological point of view. However, due to a number of artifacts they can be visually classified as modified videos, being unlikely to fool without further postprocessing.

We identified three major factors that have a significant effect on the realism of our outputs: human error, lack of computational resources and model limitations.

Human error We believe human error remains the main cause of unrealistic videos. This can be further subdivided into two classes: errors due to a lack of experience and errors due to time constraints. The software that we used relies on highly sophisticated and customizable algorithms. Each algorithm offers dozens of parameters to be determined either from trial and error or personal experience. Although some documentation does exist online it is far from being complete. Overall, a lack of peer-reviewed white papers describing the machinery behind each model often leads to incorrect assumptions and confusion. Notabley, a great proportion of understanding had to be inferred from other user's comments and feedback, thereby presenting an unreliable source of knowledge. This resulted in suboptimal parameter selection for our models and thus had a degrading effect on the final videos.

Time constraints were problematic in a less usual way. It quickly became clear that the second most time-consuming process is data preparation and post processing, with network training being the first. Unsurprisingly, better quality preparation steps do lead to better final results. However, it remains a balancing act. For example, one can individually extract faces from each frame of a video manually, ensuring mask landmarks match extremely well. Crucially, the main motivation behind DF technology is to free the user from having to manually adjust every frame using editing software, so returning to frame by frame extraction starts to erode the difference between older methods of face replacements and DF. A certain shift in the mindset of DF users is observed, with many preferring to take their network outputs to apply them in dedicated video editing software. In this scenario DF algorithms act simply as generators capable of mimicking their destination, with their products cleaned up and merged via professional video tools. One may ask whether it would be possible to create an automatic pipeline that generates realistic DF videos without human intervention; the answer would be no in the vast majority of cases. This implies that a potential user would need to dedicate a considerable amount of time to generate a realistic product.

Computational resources Computational resources place a straightforward constraint on the DF quality. Larger VRAM GPUs facilitate more complex, deeper network architectures with higher resolution and larger batch sizes. Increasing architecture complexity allows the model to learn more complex patterns in the data, while higher batch size allows the network to see more data in one go before performing weight adjustments. The videos we created on 4GB GPU

[6] `https://liveuclac-my.sharepoint.com/:f:/g/personal/zcapnwa_ucl_ac_uk/Eq4sAl6aT35JhLTmblau5uYBUDFLcoCgwGJgbvalIGj73w?e=SPsdwe`

remained blurry after 200,000 iterations, thus imposing a strict limit on the quality and hence believability. Videos produced on 12GB GPU were far better, with multiple small details well-resolved. Potentially, such hardware could be sufficient for a motivated individual to produce a realistic DF of a typical internet quality. Smaller video cards can be used for testing purposes but are unlikely to output anything of high quality.

Model limitations Certain issues stem from the way DF algorithms are programmed, meaning no amount of training will improve the result. One such example is the use of 2D mask, resulting in foreign objects like glasses to appear glued on a face, lacking any depth. Potential generation of a 3D mask is likely to be beneficial, but we are unaware whether such approach is implemented in any software. The fixed shape of the mask could potentially be an issue; however, steps are taken to move forwards with full face masks or flexible mask learnt by a network. Automatic placement of mask landmarks that relies on face detection algorithms is also limiting but remains an active area in research. Generally speaking, provided an unlimited amount of computational power, well-matching datasets, ideal mask alignment and absence of 3D objects that obstruct or interact with a face we see no factors that can prevent the model from learning a realistic mapping from source to destination face.

Data limits This is an issue we did not encounter during this investigation, as we had potentially unlimited imagery of our target. Nonetheless, one can imagine a situation where lack of data can start to become highly problematic. With certain angles absent, the user would have to rely on network's face morphing to predict a missing frame. Although it can suffice for a few 'in-between' frames, if a large proportion of orientations is missing it is unlikely that anything of quality can be generated. Some improvements can be made by using a similar person's face to pretrain the network but ultimately with small quantities of data very few improvements can be made. This remains the most effective tool for DF prevention.

Facial synthesis In practice, facial synthesis models are capable of producing convincing results. The lack of reproducibility highlighted here is, on reflection, a very good outcome. These codes have limitations that, by design or not, limit their applications for harm. For example, the aggressive crop and resolution change means that it isn't straightforward for an attacker to produce a fake video call using the SDA output.

6 Combating the deepfake threat

We have shown that whilst the technology to produce deepfakes is flawed in certain aspects, it is only a matter of time before these flaws are improved to the point that deepfakes may pose a very real threat to the general public. Naturally, this turns our attention towards preventing people becoming a victim of a deepfake attack, as well as identifying deepfaked content.

6.1 Current detection methods

Current detection methods such as LSTM lip-sync (P. Korshunov and S. Marcel 2018) and seperable CNNs (Yu, Chang and Ti 2019) have been in development for some time, however their success rates are often low and unreliable. Indeed, Pavel Korshunov and Sebastien Marcel (2018) show that the method utilising lipsync are at best able to successfully identify deepfakes $\approx 40\%$ of the time, a marginal improvement on guesstimation. However, Pavel Korshunov and Sebastien Marcel also highlight a IQM (Image Quality Metrics) with SVM (Support Vector Machines) that has a success rate closer to 90%. Whilst this is still not perfect, this approach is a significant improvement over other implementation's success rate.

However, even IQM with SVM is imperfect. It's principle metric to which it measures whether a source is a deepfake or not relies on the inherent image defects introduces in to the deepfake by the model. For example, Fig. 10 shows the suboptimal colour matching and unnatural shadows within the frame, meaning that an IQM approach would be ideal to highlight this. Furthermore, Li et al. (2019) highlight a number of these artifacts, as well as datasets built for deepfake forensics. However, it needs to be highlighted that as the quality of deepfakes improve the number of artifacts and defects within them will decrease, rendering this IQM approach less accurate with time.

In addition to this, the release of the "Deepfake Detection Challenge" on Kaggle (AWS et al. 2019) has highlighted the severity of the threat that deepfakes pose, as well as the efforts that technology companies are going to in order to combat them.

6.2 Prevention mechanisms

Deepfakes, by their nature, are deeply adaptive. Sections 3.2, 3.1 and 3.3 have shown that an entirely automatic "one-size-fits-all" solution isn't capable of producing convincing results; manual intervention is still required. However, such intervention is capable of hiding a multitude of computational sins, meaning that with current technology a would be belligerent needs only determination and patience to produce a convincing deepfake.

This renders the idea of actionable prevention mechanisms incredibly challenging. Much like the deepfake models themselves, no "one-size-fits-all" solution is effective here. It could be suggested that the model's inability to successfully incorporate eyewear means that those that appear in media wearing glasses are safer from becoming a deepfake victim. In turn, videos that show multiple different facial geometries are also more challenging to produce from a deepfake perspective, leading to further security against this type of attack.

A less sophisticated albeit effective method relies on the source video containing a prominent watermark. A watermark that enroaches on the face would reduce the accuracy of the model and increase the frequency of manual alignment required. Crucially however, a watermark of this nature within the original video may prove distracting and offputting to a consumer. A more sophisticated suggestion would utilise blockchain technology such that a transactional database is able to track the creation, edits and uploads of a video or image. This chain would therefore be able to be queried in order to identify whether a media source has been tampered/altered. However, an implementation like this would require a fundamental re-engineer of all aspects of

the web to move towards a privacy-centric model. One such example of this type of model is SOLID (Berners-Lee 2016).

Furthermore, an effective suggestion is already in place for the vast majority of individuals: privacy on social networks. In reality, social networks would be the best place for attackers to scrape images and content with which to produce a deepfake of a member of the public. Therefore, it is recommended that privacy settings within social networks are configured such that personal images are not indexed or searchable by search engines and that only confirmed, vetted people have access to those images. This therefore limits the access that attackers have to content that could be used to train their models and produce their deepfakes.

6.3 Proposed solutions

However, a much more effective suggestion relies on an ethical argument: corporations and companies that host video can incorporate deepfake detection methods on their services. Challengingly, such a task is non-trivial as it requires the application of quantitative methods to a qualitative problem that often relies on human perception. Indeed, solving this problem is a research field in and of itself, though there could be a number of fast implementations that could be applied to begin detecting deepfakes in the short term. One such approach is a cross-reference process that compares media against pre-existing uploaded content.

The objective of this is to identify whether the uploaded content contains any similarities to pre-existing content and, if it does, where these similarities lie as a function of pixel position within the frame. Subsequent application of facial recognition algorithms can then identify whether these similar pixels lie within the classified face. If the number of these identified pixels falls below some theshold value, i.e. the face in the uploaded content does not match that in the pre-existing source, then the video can be marked as tampered.

However, this method only works if pre-existing content is found. Should the uploaded deepfake not contain pre-existing content then this method obviously fails. To combat this, facial recognition technology could be re-purposed to identify an individual in an image and cross-reference the facial geometry against that determined from extant content. For instance, this method could identify the Prime Minister in a video and then determine, from a bank of images of that person, facial measurements such as interpupillary distance. By comparing the equivalent measurements in the deepfake to the "known" measurements the model could begin to determine how likely it is that the video is a deepfake. Expanding this further, the model could also track the positions of points on the face, for example the tip of the nose, the eyes and the ears, and determine whether their movements do not match the global movement of the head.

6.4 Ethical concerns

It is important to stress however that classifying media as "deepfaked" — rather than edited — relies on absolute certainty. In practise, such certainty will not be forthcoming. Instead, classification should rely on a probabilistic approach. Should these statistics indicate that a video is tampered, for instance to $> 3\sigma$ confidence, then it can be flagged as such.

Another ethical consequence of this comes when we consider what should be done with a video that is classified as a deepfake. The removal of content that is illegal or breaks a service's Terms of Service/Use is a given. Content that is deepfake classified that does not satisfy either of these conditions is more challenging to deal with. From one perspective, the continued propogation of that content could feed misinformation; on the other hand, such content may be being used for pure entertainment purposes (e.g. in a movie or production). An approach to mitigating this is to suggest that content that does not satisfy the removal conditions could not be removed and could instead be prominently flagged as "Altered" or "Tampered". An accompanying message could tell the consumer that the content they're looking at should be viewed with this in mind. In this way, media sources can remain impartial and are not controlling the content users can and cannot view. This concern could be nullified if a service ensures that deepfakes on their platform, regardless of their intention, are against their Terms of Service.

However, this approach does not limit a user's ability to subsequently share deceitful media. The challenge here is a balancing act: avoiding censorship and therefore large corporations controlling media whilst also avoiding the proliferation of harmful media designed to sabotage. In truth, perhaps the fairest and most ethically sound approach is the one that has already been adopted by Facebook: to ban anything under the umbrella of 'AI altered media designed to deceive' outside of that uploaded for entertainment or comedic purposes.

6.5 Detection pipeline

This does not, however, stop an attacker with an agenda from uploading a deepfake video to a service. Hence these services need to have a proactive and effective approach to screen, detect and action videos it deems to be deepfaked. A suggested approach to mitigating the effectiveness of deepfakes, alongside detecting them in situ, is a combination of the suggestions presented earlier. For instance, an example flow is as follows:

- The creator of a video places a recognisable watermark within it; if the subject of the video wears glasses, it is suggested that these are worn
- Uploading this video to a video-hosting site triggers the cross-reference procedure at upload
  - Pre-existing detection algorithms for illegal or inappropriate content runs as usual at this stage
- Classifying the video as being similar to a pre-existing video triggers the facial recognition stage
- Pixel positions are compared and those that lie within the boundaries of the faces within the content are identified
- This information is then used to determine whether the content is altered and the resulting video labelled as such

7 Conclusion

In this work we described two leading open-source software applications freely available to anybody and their implementation to produce deepfaked (DF) videos, as well as a facial synthesis model. Our aim was to create a DF video that could act as a demonstration of a cyber-security risk. The videos we produced throughout this investigation, although can be judged as technologically impressive, cannot be deemed realistic. We outlined common issues and conclude that such issues arise due to human error, lack of computational resources and model imperfections, sorted from more to less impactful. We can conclude that DF technology has a potential to be a security risk provided it is employed by an experienced and motivated user with access to high-end GPU. Available data does impose further constraints, with current models struggling to produce realistic imagery from unconventional angles, irregular lighting conditions, complicated facial obstructions and different facial morphology of target and destination. This makes recordings of interviews and public speeches a fantastic source for a successful DF, which coincides well with a probable source video that would be used for fraud purposes. We also assess the current, as well as future, deep detection methods and find that detection largely relies on imperfections introduced by the models in to the deepfake output. As models improve and their quality increases, these metrics will begin to become less accurate in their detections. We propose qualitative methods, such as facial regonition and cross referencing, as an alternative to the quality based metrics.

References

Abadi, M. et al. (2015). "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". arXiv preprint.

Ahmed, H. O. A., M. L. D. Wong and A. K. Nandi (2018). "Intelligent condition monitoring method for bearing faults from highly compressed measurements using sparse over-complete features". In: *Mechanical Systems and Signal Processing* 99, pp. 459–477.

AWS et al. (2019). *The Deepfake Detection Challenge*. URL: `https://deepfakedetectionchallenge.ai`.

Berners-Lee, Tim (1989). *Information Management: A Proposal*. Report. CERN.

– (2016). URL: `https://solid.inrupt.com`.

Bickert, Monika (2020). "Enforcing Against Manipulated Media". In: URL: `https://about.fb.com/news/2020/01/enforcing-against-manipulated-media/`.

Bitouk, D. et al. (2008). "Face swapping: automatically replacing faces in photographs". In: ACM Transactions on Graphics (TOG).

Bourlard, H. and Y. Kamp (1988). "Auto-association by multilayer perceptrons and singular value decomposition". In: *Biological Cybernetics* 59.502, pp. 291–294.

Bregler, Christoph, Michele Covell and Malcolm Slaney (1997). *Video Rewrite: Driving Visual Speech with Audio*. URL: `https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/human/bregler-sig97.pdf`.

Chan, C. et al. (2019). *Everybody dance now*. ICCV.

Chollet, F. et al. (2015). In: URL: `https://keras.io`.

Dale, K. et al. (2011). *Video face replacement*. ACM Transactions on Graphics (TOG.

Gatys, L. A., A. S. Ecker and M. Bethge (2016). "Image style transfer using convolutional neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2414–2423.

Goodfellow, I. et al. (2014). "Generative adversarial nets". In: *Advances in neural information processing systems*, pp. 2672–2680.

He, K. et al. (2015). "Deep residual learning for image recognition". arXiv preprint.

Hinton, G. E. and R. S. Zemel (1994). "Autoencoders, minimum description length, and Helmholtz free energy". In: *NeurIPS* 1993, p. 502.

Karras, T., T. Aila et al. (2018). *Progressive growing of GANs for improved quality, stability, and variation*. ICLR.

Karras, T., S. Laine and T. Aila (2019). *A style-based generator architecture for generative adversarial networks*. CVPR.

Karras, Tero et al. (2019). *Analyzing and Improving the Image Quality of StyleGAN*. arXiv: 1912.04958 [cs.CV].

Kim, H. et al. (2018). *Deep video portraits*. ACM Transactions on Graphics (TOG.

Kingma, D. P., T. Salimans and M. Welling (2015). "Variational dropout and the local reparameterization trick". In: *Advances in Neural Information Processing Systems*, pp. 2575–2583.

Kirat, Dhilung, Jiyong Jang and Marc Ph. Stoecklin (2018). "DeepLocker: Concealing Targeted Attacks with AI Locksmithing". In: URL: https://i.blackhat.com/us-18/Thu-August-9/us-18-Kirat-DeepLocker-Concealing-Targeted-Attacks-with-AI-Locksmithing.pdf.

Korshunov, P. and S. Marcel (2018). "Speaker inconsistency detection in tam- pered video". In: European Signal Processing Conference (EUSIPCO).

Korshunov, Pavel and Sebastien Marcel (2018). *DeepFakes: a New Threat to Face Recognition? Assessment and Detection*. arXiv: 1812.08685 [cs.CV].

Korshunova, I. et al. (2017). *Fast face-swap using convolutional neural networks*. ICCV.

Kullback, S. and R. A. Leibler (1951). "On information and sufficiency". In: *Ann Math Stat* 22, pp. 79–86.

LeCun, Y. (1987). "Modèles connexionistes de l'apprentissage". In: *Ph.D. thesis, Université de Paris VI* 18.502, p. 515.

– (1989). *Generalization and network design strategies*. Technical Report. CRG-TR-89-4, University of Toronto, 330, 352.

Li, Yuezun et al. (2019). *Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics*. arXiv: 1909.12962 [cs.CR].

Lin, H. et al. (2019). "Learning rate dropout". arXiv preprint.

Liu, M. Y., T. Breuel and J. Kautz (2017). *Unsupervised image-to-image translation networks*. NeurIPS.

Ma, X. et al. (2019). "Style-based Variational Autoencoder for Real-World Super-Resolution". arXiv preprint.

Maas, A. et al. (2013). "Rectifier nonlinearities improve neural network acoustic models". In: *ICML* 30.

Pham, H. X., Y. Wang and V. Pavlovic (2018). "Generative adversarial talking head: Bringing portraits to life with a weakly supervised neural network". arXiv preprint.

Stupp, Catherine (2019). "Fraudsters Used AI to Mimic CEO's Voice in Unusual Cybercrime Case". In: URL: https://www.wsj.com/articles/fraudsters-use-ai-to-mimic-ceos-voice-in-unusual-cybercrime-case-11567157402?utm_campaign=the_algorithm.unpaid.engagement&utm_source=hs_email&utm_medium=email&utm_content=76535820&

_hsenc=p2ANqtz-_HP64YUhkAXhxAYRrsaGBy1NcusfB8RBDpoNn3Ie2_GdN85NJIwUWmmCPzGwLg7LHx2su7J9ha
_hsmi=76535820.

Suwajanakorn, S., S. M. Seitz and I. Kemelmacher-Shlizerman (2015). *What makes Tom Hanks look like Tom Hanks*. ICCV.

– (2017). *Synthesizing Obama: learning lip sync from audio*. ACM Transactions on Graphics (TOG.

Thies, J., M. Zollhofer et al. (2016). *Face2face: Real-time face capture and reenactment of rgb videos*. CVPR.

Thies, J., M. Zollhöfer and M. Nießner (2019). *Deferred neural rendering: Image synthesis using neural textures*. SIGGRAPH.

Vougioukas, Konstantinos, Stavros Petridis and Maja Pantic (2019). *Realistic Speech-Driven Facial Animation with GANs*. arXiv: 1906.06337 [cs.CV].

Yu, Chia-Mu, Ching-Tang Chang and Yen-Wu Ti (2019). *Detecting Deepfake-Forged Contents with Separable Convolutional Neural Network and Image Segmentation*. arXiv: 1912.12184 [cs.CV].

Zhang, S. et al. (2017). "S3fd: Single shot scale-invariant face detector". arXiv preprint.