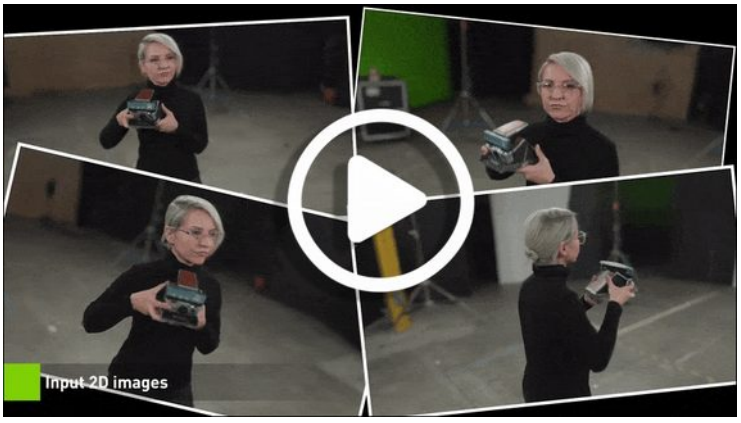
The Future of Autoencoder-Based Deepfakes

Originally published July 11, 2022 at https://metaphysic.ai/future-autoencoder-deepfakes/ -

Note: *This is now the only place on the web that a relatively complete version of the original post exists, including multimedia. It has been quite a job reconstructing it, as I did not have the original source material any more. Wrangling flat HTML archives and PDFs is some medieval...stuff. I beg your indulgence for errors such as text hyperlinks links that don't work, etc. In any case, most such links do not work any longer, due to 'selective' domain migration from metaphysic.ai to brahma.io.*

The way we refer to visual effects (VFX) work may be changing soon. For instance, at the time of writing, the relatively new technology of Neural Radiance Fields (NeRF) makes it possible to recreate entire scenes –including humans, homes, exterior environments, and almost anything else you can imagine – inside an AI's neural network, from a handful of static photos.



Is this a 'deepfake' or a 'simulation'? NVIDIA's 2022 Instant NeRF system derives an explorable neural 3D scene from a few source photos in as little as 5-10 seconds, and very little of the rendered material (only four frames) is 'real'. Source: https://www.youtube.com/watch?v=DJ2hcC1orc4

Since machine learning systems such as NeRF are run on computers, these brand-new technologies could eventually be included in the 'traditional' term *Computer-Generated Imagery*, aka 'CGI'.

Not Your Dad's CGI

However, in regard to their method and significance, NeRF and other emerging AI image synthesis methods, such as Generative Adversarial Networks (GANs) represent a potential quantum leap over the laborious manual processes that first stunned the world in *Jurassic Park* (1993) and *Terminator 2* (1992), and which continue to furnish the visual effects for summer blockbusters – at least, for now.

These emerging AI systems have the potential to attain new heights of realism across all sectors of visual effects, and to transcend the uncanny valley (the 'creepy' effect of sophisticated but ultimately unconvincing digital humans) – a challenge that has eluded traditional CGI techniques for decades.

Crucially, AI-driven processes like these could eventually democratize high-quality visual effects, making the very best quality VFX available not just to $500+ million superhero outings, but to more modest productions at the lower end of the film release cycle, and for television and streaming output.

Therefore, as machine learning-based image synthesis and editing systems graduate from ancillary tools in traditional CGI pipelines to full-fledged methods in their own right, we may need a new term for such approaches. CGAI? AIFX? Time will tell.

Continues on next page

In 2021, TikTok deepfaker NextFace incorporated modern singer Billie Eilish into Judy Garland's performance in 'Meet Me In St. Louis' (1944). Source: https://www.tiktok.com/@nextface/video/6936881472313806085

DeepFakes as 'Viral Deepfake Videos'

This is not just semantic pedantry – these new technologies are coming into existence and gaining traction and industry interest faster than we can distinctly name them, confusing the discussion.

For instance, 'deepfake' is beginning to be used in terms of AI-generated news content and audio-based fraud, as well as for simple mobile and web-based offerings that often produce crude, rudimentary, or limited output.

Therefore, this article is about 'deepfakes' in the context that the term has been most commonly understood over the past five years – AI-altered videos where autoencoder-based systems (which we'll look at shortly) are used to replace a face in a video with another face, at a level of sophistication and realism capable of inspiring awe and even alarm.

DeepFaceLab and FaceSwap

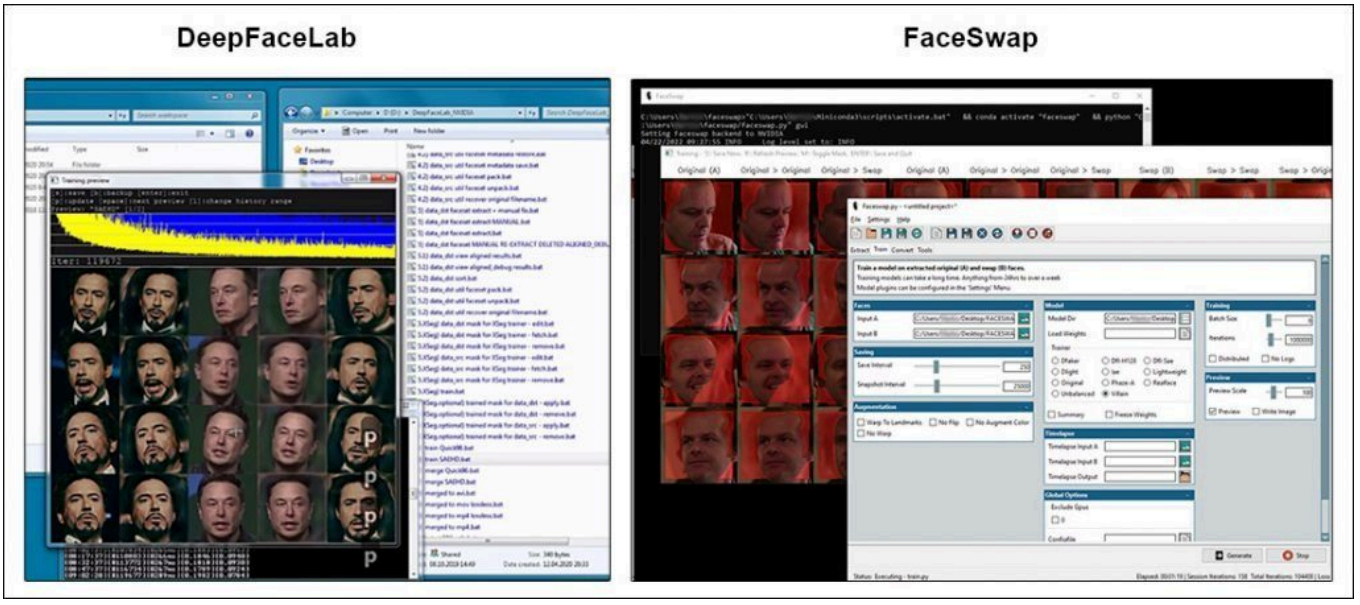
The term 'deepfake' was originally associated with the advent of AI-generated celebrity porn in 2017, though it has subsequently gained immense (and more acceptable) popularity as a method of 're-casting' and simulating actors in short video clips, and even bringing former celebrities back to life for documentaries, as well as slowly creeping into mainstream film and TV production.

Continues on next page

In a DeepFaceLab training session, actor Roly Hyde is transformed into the late and acclaimed science-fiction TV producer Gerry Anderson, for the 2022 documentary 'Gerry Anderson: A Life Uncharted'. The work was undertaken by the Anachronistic VFX company. Image courtesy of Christian Darkin.

The machine learning code behind the 2017 deepfakes that shocked the world was released in the r/deepfakes sub-Reddit — a forum that was blocked from public view almost as soon as the controversies broke. However, one user copied the code to GitHub before the ban, and that code has since been forked (i.e., other people have adopted or adapted the project) over 1000 times.

However, output from just two of those forks, DeepFaceLab and FaceSwap, has come to dominate what the public still currently conceives of as 'deepfakes'.



Left, DeepFaceLab (DFL). Right, the FaceSwap project. Unlike the command line-driven DFL, FaceSwap has a user-friendly GUI, and extensive Linux support. Source for DFL image: https://www.youtube.com/watch?v=1smpMsfC3ls

These software packages are maintained on a voluntary basis by enthusiast open source developers, and, at the time of writing, power practically every viral deepfake video you've ever seen.

The most dedicated (SFW) deepfakers using these packages have become YouTube and TikTok celebrities. Several, such as Metaphysic founder Chris Ume, ILM inductee Shamook and prolific faker Ctrl-Shift-Face, have crossed over into professional VFX production.

Partly due to DFL's deep association with deepfake porn — even its official user guides are hosted on an AI porn site — production studios rarely discuss their use of the software, while many prominent *non*-porn DFL creators still guard their anonymity.

As we'll see, some VFX companies and entrepreneurs have incorporated this open source code into closed, proprietary systems, or deconstructed the approach into new architectures.

Though professional production companies can avail themselves of expensive GPU training resources and personnel (to curate and refine the face datasets that power the deepfake models), as well as development resources to enhance the original packages, the slowly-evolving code behind DFL and FaceSwap remains available to anyone interested in putting in the considerable hours needed to master the learning curve, and to curate the data.

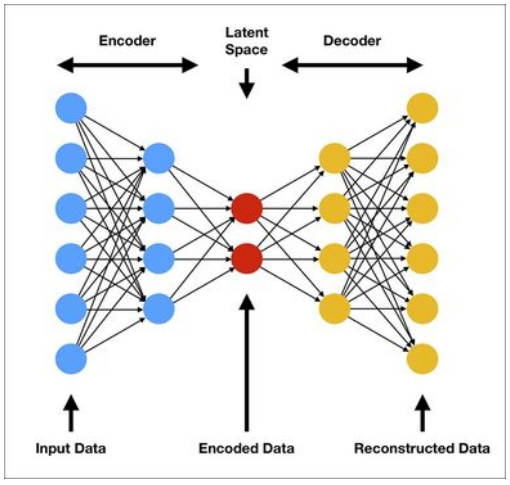
Many other systems, applications, and research projects have since emerged that perform the same or similar functionality. Some are quite simplified and exclusive to mobile devices, such as the Reface iOS and Android app.

However, model optimization and the increased computing power of mobile devices are beginning to offer more substantial deepfake capabilities to non-technical and mobile users: in April of 2022, an academic collaboration from China proposed MobileFSGAN, a full-fledged autoencoder system weighing little more than 10mb, which can perform faceswaps directly on iOS and Android phones (see image below).

MobileFSGAN in action: from the Chinese aca demic collaboration 'Migrating Face Swap to Mobile Devices: A lightweight Framework and A Supervised Training Solution', a sample video shows face-swap quality approaching that of desktop systems. Source: https://www.bilibili.com/video/BV1kh411i7A2

How Do Autoencoder-Based Deepfakes Work?

DeepFaceLab (DFL) and FaceSwap use an autoencoder architecture. An autoencoder neural network is designed to *encode* data (such as an image) into a *lower dimensional latent representation* (i.e. into mathematical vectors instead of pixels), compress the data, and then reconstruct the same data at the other end of the pipeline.



Conceptual schematic of an autoencoder. Source: https://www.compthree.com/blog/autoencoder/

Autoencoders are used for many purposes in computer vision, such as handwriting analysis and facial recognition.

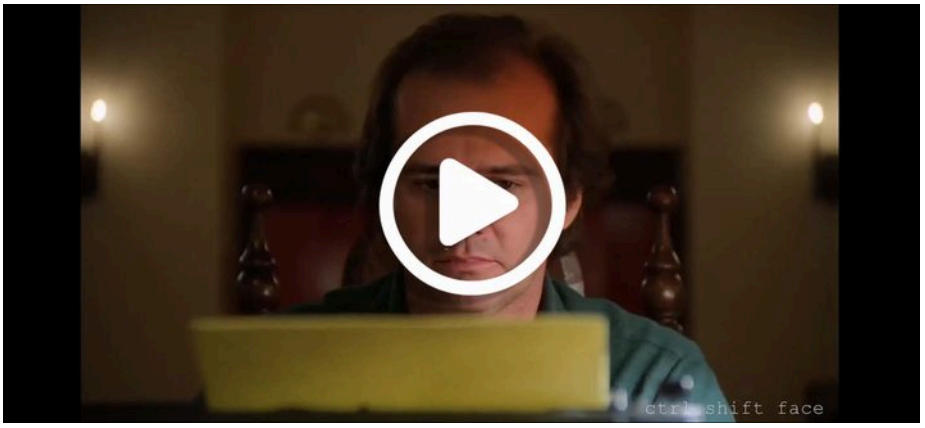
Since autoencoders seek 'essential' data from an image (such as a face image), they are very good at eliminating visual noise. In the case of deepfake software, this means that an autoencoder architecture can learn fundamental features and traits from a face image whilst largely ignoring extraneous factors such as grain, shadows, and other 'non-face' elements that may be present in the image, resulting in versatile and well-generalized models (i.e. models that can perform useful transformations on data that's different to their original training data).

Modern popular autoencoder-based deepfake packages such as DeepFaceLab incorporate various software libraries from the open-source machine learning research community. Very few of the essential components in such packages are written 'from scratch', for the sole purpose of producing deepfakes.

Data Gathering

In the first instance, it's essential to create a *face set*: a collection of face images from which the autoencoder will derive essential information about that particular identity.

Therefore we must obtain many, many pictures of the *source subject* and the *target subject*. For example, when replacing Jack Nicholson with Jim Carrey in a clip from Stanley Kubrick's *The Shining* (1980), Nicholson is the source subject and Carrey the target subject.



NOTE: *The demonstration images below are inspired by the popular YouTube parody by Ctrl Shift Face (video embedded above). All of the related process details featured as images in this article have been recreated by me, and do not necessarily represent either the workflow or software/method choices of Ctrl Shift Face.*

Most deepfake developers and practitioners recommend that 5-10,000 pictures should be extracted and curated for each subject, ideally from high quality video clips with varied lighting, and featuring diverse poses and expressions.

Some professional adaptations of these frameworks use *hundreds of thousands* of images, while some of the more popular deepfake artists use an even higher number of source images.

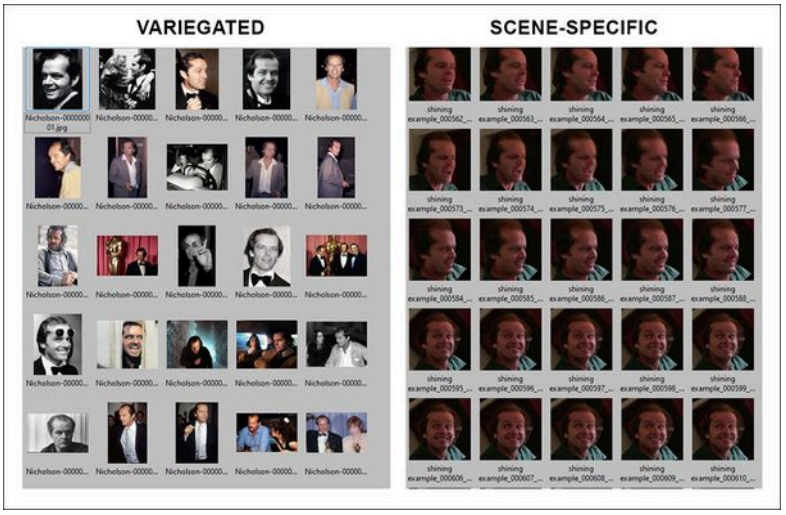
A typical approach to sourcing images for face sets, mixing in various 'social' and web-obtained pictures, such as publicity shots, together with extracted frames from movie clips.

Dedicated or 'All Purpose' Face Sets..?

Resource management is crucial during the entire process; not least here, at the data gathering and curation stage.

Since it can take 1-2 *weeks* to train a high quality model even on a well-specced GPU, it would be great if that model could be used to insert the target subject into *any clip* featuring the source subject.

However, a *dedicated* face set, solely comprising frames of the source subject *in one particular clip*, will produce a more accurate model – even if that model is likely to perform poorly on any *other* clips.



A model trained on a variegated dataset (images on left) is likely to generalize well and produce acceptable results in a wide range of situations. A model trained specifically on faces extracted from the target video clip (images on right) will produce a much more accurate swap for that clip, but will usually perform poorly on other clips, as it is 'overfitted' to a specific video.

This is because, during training, the neural network will dedicate a far larger part of its resources to the *exact characteristics* of the target clip, usually resulting in an excellent subsequent face swap – but producing an expensive and time-consuming model that can't really be used for any other task (though a 'snapshot' of the model's early training phase can be used as a 'pre-trained' template for later models that feature the same person, to save time in later projects).

Conversely, using non-specific and diverse face-images of a person will result in a model that is better generalized, and able to perform good swaps on many different videos — but not at the same quality as a model trained on a clip-specific dataset.

Obviously, the target face set (in this example, Jim Carrey) will *have* to be diverse and slightly random, because there is no *real* footage of Carrey playing Jack Nicholson's role; but clips of the Carrey identity should be chosen based on how similar they are to the clip we want to change (i.e., where possible, similar grain, lighting, expressions, etc.). The examples featured here are from Carrey's dramatic and intense role in *The Number 23* (2007).

Extraction, Pose Recognition, and Masking

Whether we choose to train an all-purpose or dedicated (i.e. clip-specific) model, the next stage is for the software to analyze all the images that we have curated so far; to estimate the poses — including facial expressions — featured in the face images; and to create *masks* that form boundaries on the extracted face, so that only *face material* (and not backgrounds, hair, earrings, and other extraneous elements) are trained in the neural network.

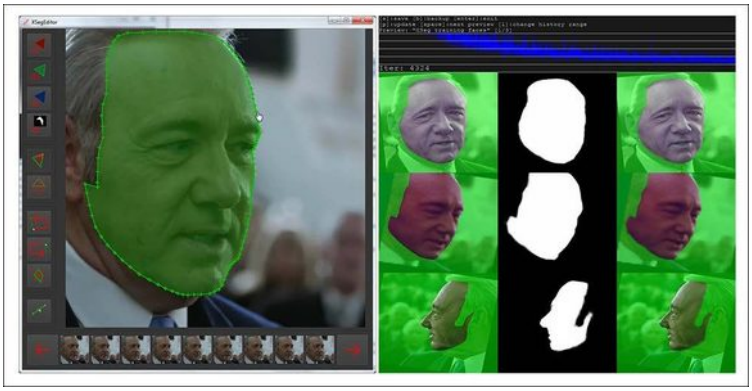
The extraction process iterates through the entire face set, using Adrian Bulat's Facial Alignment Network (FAN) to infer face poses, including facial expressions such as smiling or shouting.

Jack Nicholson's varying facial poses captured by the Facial Alignment Network (FAN), common to both DeepFaceLab and FaceSwap. This example is captured from FaceSwap (however, DeepFaceLab uses a more complex 3D landmark FAN extraction method for 'full head' deepfakes, i.e. deepfakes that include the subject's hair and ears).

These FAN alignments are used to generate masked areas revealing only face content. If faces are obstructed by stray elements such as fingers, hair, or even glasses, the masks should exclude that content.

Cut It Out!

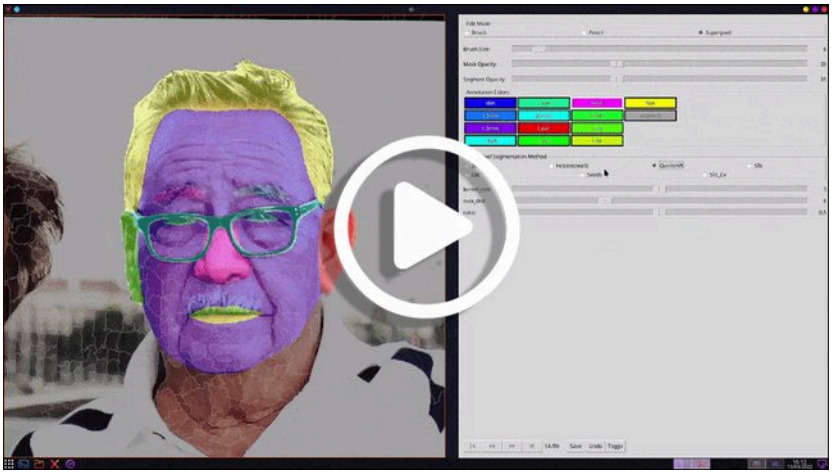
This is quite a challenge: DeepFaceLab approaches it with a dedicated program called XSeg, where the user manually draws masks every certain number of frames, and XSeg attempts to 'fill in'(or 'tween') the intermediate masks, based on that manual input.



FaceSwap also features a dedicated mask and alignment editor, and has recently introduced a highly effective masking algorithm based on the 2018 BiSeNet segmentation network (and a later PyTorch implementation).

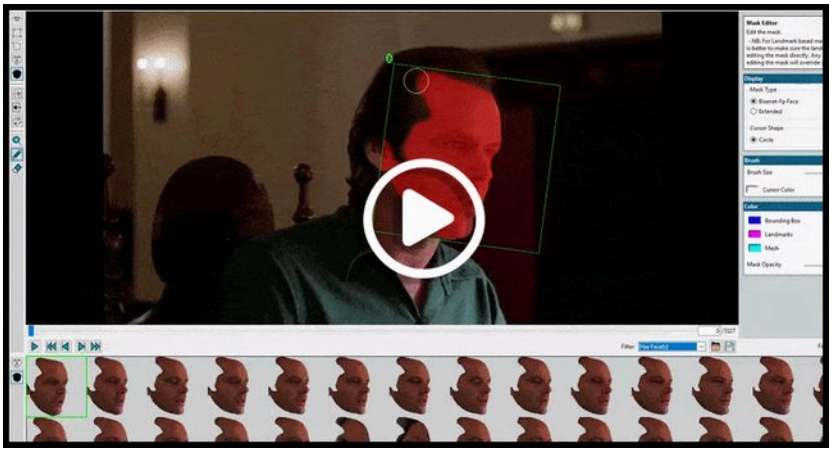
The BiSeNet-based automated segmentation algorithm in FaceSwap is trained on common obstructions, and here has successfully masked out periodic occlusions in the footage, without any manual intervention.

FaceSwap's Bisenet-Fp can hide or reveal ears, hair, and glasses, if the user wants to consider these elements in training. It was laboriously curated by core project developer Matt Tora, who manually annotated 40,000 examples of occlusions (glasses, cigarettes, hair, fingers, etc.), producing an automated masking algorithm that's incredibly discerning and accurate.



FaceSwap developer Matt Tora devised a bespoke labeling tool to create 40,000 edits for the Bisenet-Fp weights, and is currently working on another 10,000. Courtesy of Matt Tora.

In failure cases, individual masks can be hand-drawn or amended in both DeepFaceLab and FaceSwap:



The 'Manual' tool in FaceSwap allows the user to directly edit automatically extracted masks.

The now-bounded faces are extracted into smaller cropped images, which are saved into a new folder of 'training faces'. This is what will be fed into the autoencoder network.

Final extracted images for training. In both DFL and FaceSwap, each image contains metadata describing its facial alignments and mask shapes.

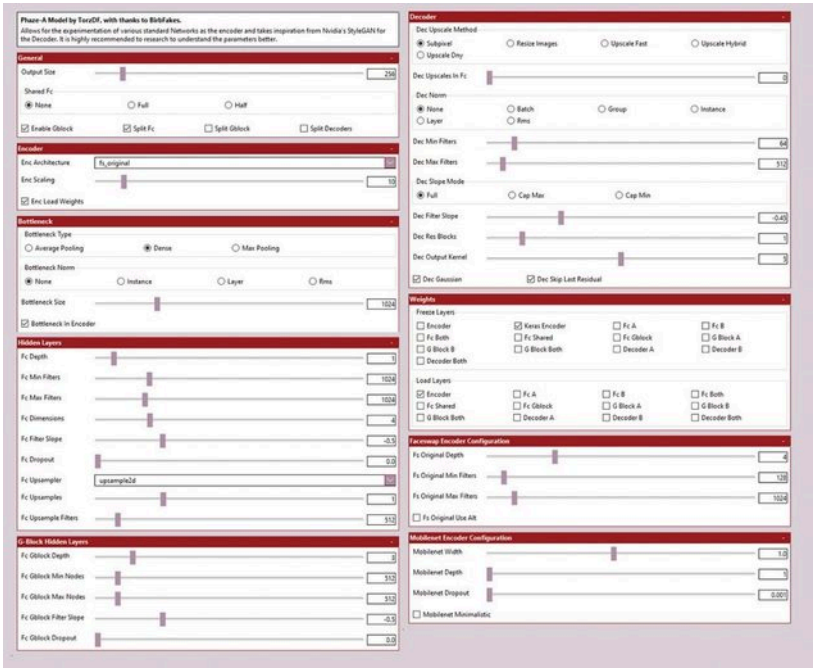
Both DFL and FaceSwap include tools to speed up manual removal of 'unwanted' identities and false recognitions, such as 'extra' people in the shot, or cases when the facial recognition component infers a face where none exists, such as in the pattern of a lace curtain, a *part* of a face, or even a dog:

Some typical facial recognition failures in FaceSwap's extraction processes, also common to DFL. These will need to be identified and removed from the dataset before training begins.

These preliminary procedures are run on both the source and target identity, resulting in a separate folder of images for each personality.

Training

At this point, training begins, and we have to choose a model type. Currently, nine models are available in FaceSwap, at varying levels of hardware requirements, flexibility, capability, and ease of use. Among these is the labyrinthine new *Phaze A*, a complex and highly configurable wrapper for 11 of the most popular open source encoding architectures.

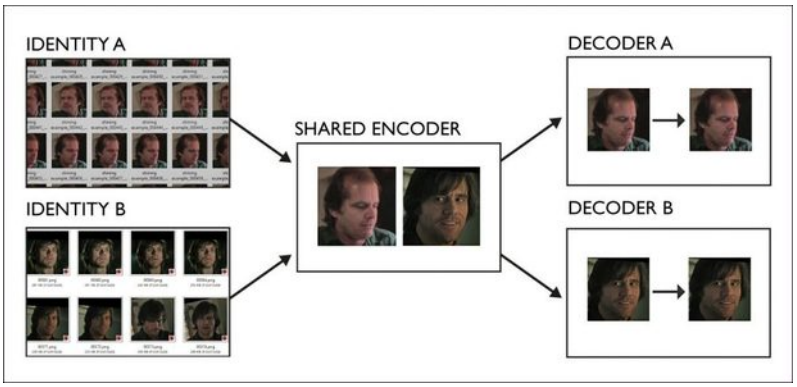


A dizzying array of options in FaceSwap's Phaze-A framework, which allows the user to create highly customized sub-models.

By contrast, DeepFaceLab currently offers only three models – SAEHD (the standard choice, and long-since ported to FaceSwap), the almost equally popular AMP, and a lightweight 'tester' model called Quick96.

The extracted faces are fed into the model in batches, and processed in the GPU. The model converts the images into vector information and proceeds to extract core traits from each image, slowly building up a database of information related to each identity.

In typical model configurations, the two identities are trained in a *shared encoder*, which means that the model slowly learns the relationships between *each face* across the two datasets.



As we can see in the image above, the model is learning to recreate each identity inside the latent space of the shared encoder, and will eventually produce two productive decoders, one for each identity.

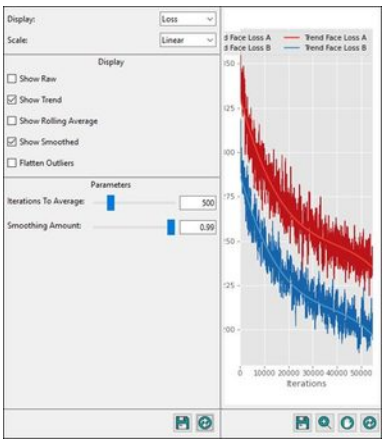
As we can see in the image above, the model is learning to recreate each identity inside the latent space of the shared encoder, and will eventually produce two productive decoders, one for each identity.

In the first hours of training, previews gradually demonstrate the assimilation of features across the two identities. Matched poses are presented during training in order to evaluate progress of the model. Depending on available hardware and settings, up to several weeks of continuous training may be necessary to obtain a convincing swap. When this screenshot was taken, the training had only been running for twelve hours.

For this reason, the datasets need to have *as many poses in common as possible*. If there's a picture of subject A looking *straight up*, and no such picture/pose for subject B, the model cannot learn to create a good transition between the two identities for that pose, because it's missing half of the necessary information. At best, the source identity will bleed through into the target identity; frequently, such data imbalances will lead to other types of distracting artefacts.

Likewise, a wide variety of matching *expressions* are needed across the two datasets. If subject A never smiles and subject B smiles a lot, the model can never learn to accurately represent subject A smiling.

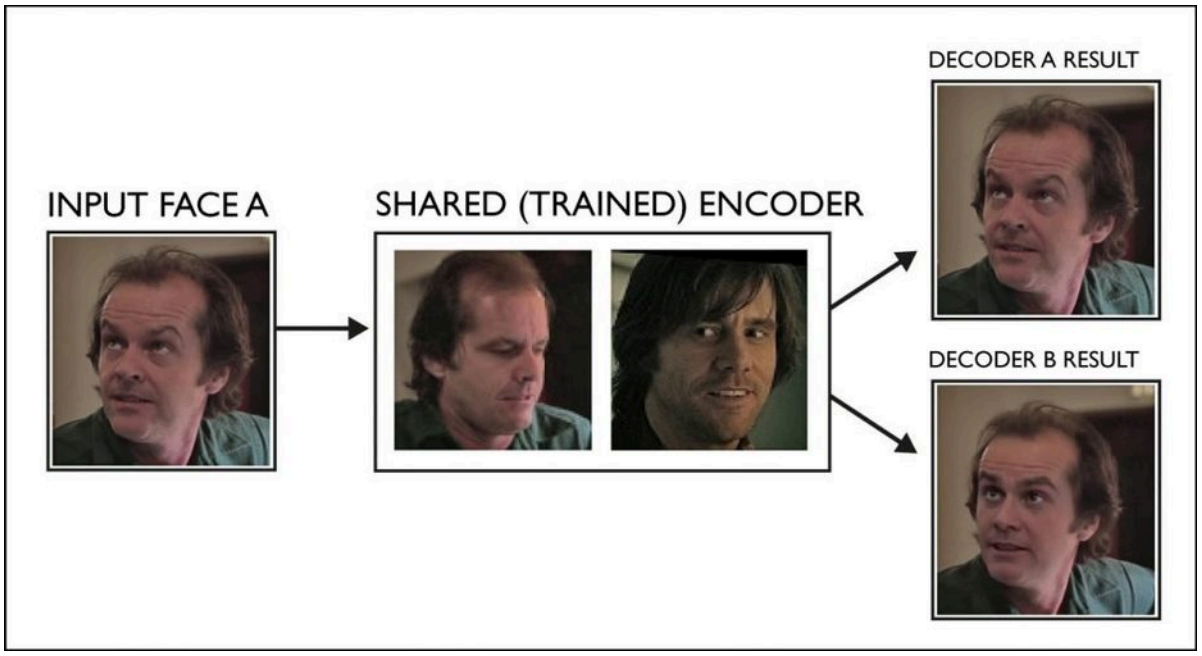
After a typical 3-14 days of training, depending on settings and data volume, the model reaches *convergence* – the point at which further training becomes redundant, or even detrimental.



A visualization of the reconstruction loss descending (that's good!) over a long period of non-stop training for a single FaceSwap model. Once target loss reaches a very low range (typically 0.07 – 0.01), the model has effectively 'converged', and training can be stopped.

Conversion

The trained model is now capable of recreating each identity quite well; but if you simply switch the decoder routings, the model can also now impose the *alternate* identity:



Newbie deepfakers sometimes ask if popular packages such as DFL and FaceSwap can also perform face swaps on single images. The truth is, that's *all* these applications do, since they operate on footage that comes in the form of hundreds or even thousands of individual images exported, manually or automatically, from the video clip (the altered images are automatically reassembled back into video at the end of the process).

Deepfaked face footage can also be exported as a series of masked images (i.e. with an alpha channel), instead of being directly incorporated into a new deepfake video clip:

This allows deepfakers to import the altered face footage as a 'floating' layer over the original footage, in video-compositing packages such as Adobe After Effects. In this way it's possible to tweak the masks, alter the color grading, blend the faces manually, and perform various other operations.

Are Deepfakes Really Going to Keep Getting Better?

Besides their potential for improving visual effects, deepfake videos (in the sense that they're addressed in this article) are a growing focus of public concern and legislation, fueled by the media's conviction that the quality of deepfake material is improving at a rapid and consistent pace.

But are deepfakes getting better because the technology is really evolving rapidly? It's not quite that simple, and the goal of continuous improvement faces a number of barriers. Let's take a look at some of them, and at potential roads forward.

Sleight of Hand

Most of the major improvements to the popular deepfake software distributions are at least a couple of years old, and many of the most impressive leaps in deepfake quality have occurred because deepfakers have now had years to explore the limits of the software, to become adept at working within the constraints of the code (and whatever hardware they can afford to run it), and to develop post-processing techniques that can improve on the output of the actual software packages.

Some deepfakers have learned to work around the technology's shortcomings by carefully selecting video clips that are suited to the process; by extensive conventional post-processing manipulation; or else by shooting custom footage (for instance of impersonators) that's likely to lend itself to more convincing deepfake output.

Additionally, a small cadre of the best deepfake developers (together with the research departments of notable VFX companies around the world) has actually started to extend the technology, often with resources far in excess of the 'casual' amateur deepfaker.

Let's now consider some of the more pragmatic obstacles to maintaining the growth in realism for autoencoder-based deepfakes.

The GPU Bottleneck

Though the GPU famine of the last couple of years seems set to abate, prices are unlikely to return to pre-pandemic levels, making high-quality deepfaking an increasingly expensive hobby — almost on a par with cryptocurrency mining in terms of hardware costs and electricity consumption.

Besides environmental concerns, the rising cost of the electricity required for the weeks (or even months) of training a single machine learning model may eventually become a notable obstacle to casual or malicious deepfaking, particularly where no money is being earned in the process.

But even with unlimited resources, autoencoder deepfakes are faced with an architectural bottleneck, in regard to how many training images can pass through the GPU at any one time; how large those images can be; and whether the architectures can scale up effectively.

For instance, DFL's GitHub features a gallery/history of increased training image input sizes since the advent of the project (check it out to see a wider range of correctly sized examples):

Representative examples, as of July 2022, of the increase in maximum size of training images and native output for DeepFaceLab since it launched. Note hat the latest (Morgan Freeman) example requires 24GB of video RAM (VRAM), with the cheapest suitable example currently priced around $1,500, subject to availability. Source: https://github.com/iperov/DeepFaceLab

State-of-the-art input training size and deepfake output still hovers around 512×512 pixels. For this reason alone, there have been relatively few examples of autoencoder deepfakes in professional film and TV production, and all have avoided 'close-up' shots.

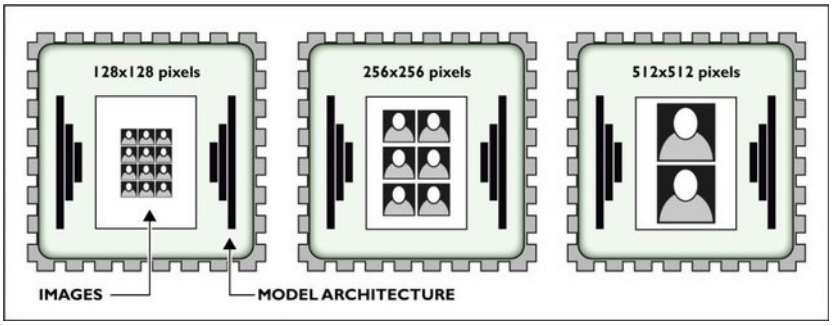
So, why not just get a bigger GPU, use bigger images, and get bigger output? After all, the models themselves can accommodate much larger images, and major VFX studios have deep pockets.

Part of the problem is related to batch sizes, i.e. how many images the training process can examine at any one time.

Batch Logistics

Images are trained in batches. Though typical batch sizes are between 6-24 images, you can set a batch size as low as 1. There is no *theoretical* upper limit either on batch size or image size (which are constrained instead by the limitations of the available hardware).

However, the larger the training images, the fewer of them can fit into the GPU in any single batch.



Consider also that the GPU has to accommodate not only the training images, but a considerable portion of the actual code of the deepfake software, diminishing the available space for the extracted face images.

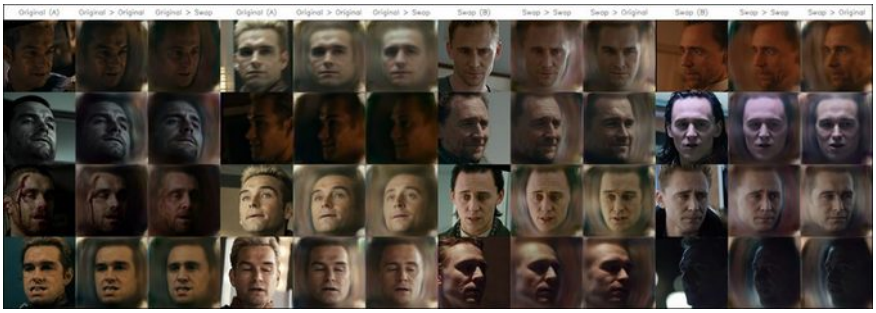
In the broad concept illustration above, we see that increasing the input training image size reduces the number of images that can be fit into the GPU in a single batch, while the software architecture also takes up a fixed allocation of GPU space, reducing space for the images.

A Difficult Balance

Where batch size is concerned, bigger is not always better. The more images in a batch, the more likely it is that the final model will *generalize* well but fail to capture detail that's intrinsic to the identity, resulting in a more 'vague' resemblance to the target.

Though smaller batch sizes may increase training time, they will also require less frenetic GPU activity, and allow the model to focus better on the (fewer) faces currently passing through the pipeline.

Therefore the general wisdom in the deepfake community is to 'start large' (high batches, moderate learning rates) and hone done until the smaller batch sizes and lower learning rates have the breathing space they need to concentrate on the finer details (such as teeth and inner eye areas), now that the central structures of the face are established.



Training a model with Phaze-A's StoJo preset, in FaceSwap. Source: Discord

However, any way you cut it, deepfaking is a VRAM-constrained activity, with 'card-envy' common in the user communities, along with the belief that many of the central challenges could be solved by throwing more VRAM at the problem. In these well-funded fantasies, users could train higher-res models at medium batch sizes, and standard-res models at *enormous* batch sizes (and far quicker, to boot, saving time and electricity!).

There are several reasons why neither money nor the VRAM it can purchase can solve all the problems.

Not that money is a minor issue: if you can access one of NVIDIA's A100 80gb GPUs, you certainly can have a few 512x512px images in a single batch for an HQ model, as well as higher batch sizes for smaller image in lower-res models; but the $30,000 USD price tag is prohibitive for hobbyist use, and there is in general a massive and jolting cost-leap between relatively affordable consumer GPUs such as the NVIDIA GeForce 30 (30xx) series (8-24gb) and higher-end cards often intended for data centers and industrial use.

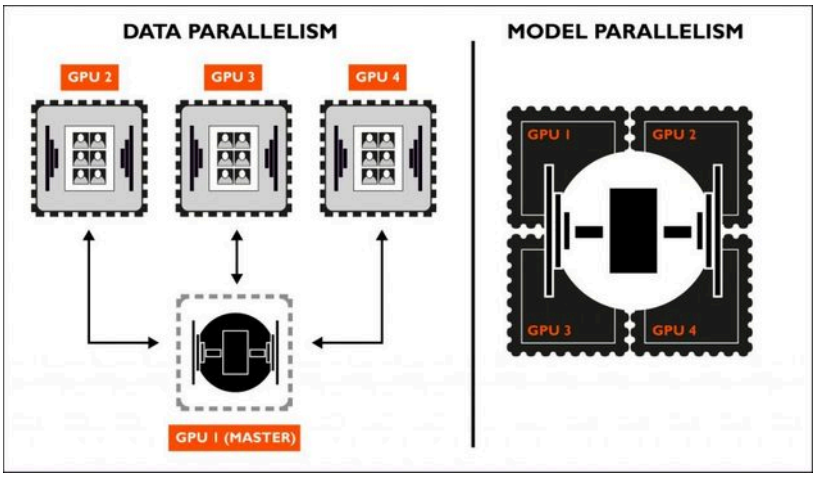
That said, 512px is still not production resolution; 1024x1024px barely gets you in the door in terms of movie and TV standards; but you'd usually need to compromise on speed or quality to obtain a pure 1024px pipeline (later, we'll take a look at the nascent 1024px deepfake scene).

Even if you *do* have a large GPU and are looking to train a smaller model, you can't usually speed up effective model training by imposing enormous batch sizes, because the model can't learn that fast; and the results, again, are likely to be substandard in comparison to standard and lengthy training on a consumer-level GPU.

Okay — so if we can't solve the problem with a single GPU, let's do what the processor industry did when Moore's Law expired, and spread the training challenge across *multiple* GPUs.

Data Parallelism and Model Parallelism

There are two standard ways that multiple GPUs can potentially scale up autoencoder deepfake systems: Data Parallelism and Model Parallelism.



With data parallelism (left, in the above image), the batch of training images are split across multiple GPUs, and orchestrated by a central controlling mechanism (pictured below), which controls the gradient descent and other training routines.

Each GPU in the group also has to make room for an instance of the model architecture, so an 'extra' GPU in the array is not entirely free to just 'fill up' with training images. This approach is already implemented in FaceSwap, though not commonly used.

Model Parallelism (right, in the above image) is more akin to virtualization, where the model is 'unaware' that the resources it is running on are composed of multiple GPUs instead of a single GPU.

Data Parallelism can be implemented in a FaceSwap project, though core developer Matt Tora warns that there are hard limits and diminishing returns to this approach:

"Data parallelism is not the best way to do this," Tora says. "It's what we do, because it's the easiest way, but there are at least a couple of issues here. Firstly, the model has to exist on every GPU in the array. So if your model takes up 10 gigs, it takes up 10 gigs on every single one of those GPUs."

"Secondly," Tora continues. "adding GPUs doesn't scale up the network in a linear way. You get something like a 1.5 increase per GPU. So for each GPU that you add, your speed-up's going to be worse than the one before. Because of these diminishing returns, I'd say that eight GPUs is the maximum you could really get any benefit from with this approach."

The chief advantage of this method is an increased batch size that's distributed across the GPUs — though we've already seen that this is not a 'magic bullet' for obtaining high-fidelity deepfakes.

GPU-based Model Parallelism is at a more nascent stage, though Google's Mesh-TensorFlow could in theory 'dump' an architecture such as DFL or FaceSwap directly into a single and coherent multi-GPU instance of any size, subject to the constraints of data transfer rates among the units.

Scaling Up Donald Trump

It's clear by now that these approaches exceed the usual scope of hobbyist machine learning enthusiasts. Many are chasing the elusive '1024 pixel pipeline', where 1024x1024px images are able to be trained productively into a model that can utilize and originate high quality, *native* 1024px input/output images, rather than relying on upscaling.

Examples of convincing footage from an end-to-end 1024 pipeline are scarce, both in the Discord forums of DFL/FaceSwap, and in the official 'behind-the-scenes' PR efforts of the many VFX companies that are currently experimenting with deepfakes packages as production tools.

However, that's not to say that no-one has done it. Over at PAGI studios in Sacramento, CA, former systems and security developer Dogan Kurt has developed an end-to-end 1024 pipeline that yields notable texture detail and temporal consistency:

Stability for the face output is helped by a proprietary landmark smoothing algorithm for the FAN alignments.

Kurt states that the model was developed through *progressive training*, an approach suggested by a 2018 NVIDIA paper, and also featured in Disney Research's *Rendering with Style* paper the following year.

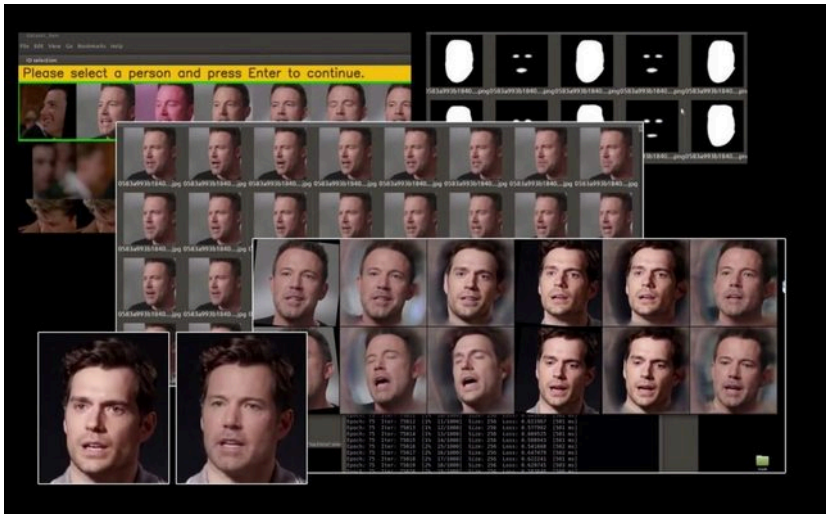
'We used progressive training as in ProGAN and Disney's paper. The good thing about progressive training is that you start with lower resolution, like 256 and grow the model progressively. Hence the initial iterations are very fast, and once the model is pretty good at lower resolution, you increase the resolution and get extra details.

Kurt says that it took two weeks to obtain the quality seen in the above YouTube video.

'But,' he adds 'we continued to train it for another two weeks, for more details, and at some point stopped it.'

The Navalny/Trump deepfake, Kurt says, was trained on a single 2080ti GPU, which is about 4x slower than an A100, and has only 11GB of VRAM. Part of the method involves redistributing a portion of the live architecture away from the GPU during training, freeing up space for training images.

A new model in this novel framework already has access to high-level extracted facial features from thousands of identity encodings that have been laboriously pre-trained from around half a million images. Some of these 'prior' images are from publicly available computer vision datasets, others hand-curated.



Stages from the PAGI deepfake workflow. Source: https://www.youtube.com/watch?v=A5J0s-6Uhu8

In the upper left corner of the image above, you can see that the user is allowed to select an identity that has been automatically recognized in the target video from this pretrained corpus.

The PAGI system, Kurt says, has only a tentative relationship to the original Keras source code from which DFL and FaceSwap evolved. He explains:

'I converted the original Keras code to PyTorch about 2 years ago. That's how I started my experiments…My architecture, however, is quite different. Is there any code copied literally? No, it's all written from scratch. Is there any idea from the original code? Definitely.

'As for DFL, I only checked its code when I wanted to port SAE and AMP to my own product. which is well after the Trump demo.'

1024px Training in FaceSwap

This week, the FaceSwap project also gets a native 1024px input/output model, in the form of a new setting for the Phaze-A framework.

Very early in a test training session for the new native 1024 preset in FaceSwap's Phaze-A. Click to see full-size image, and to get some idea of how big this training preview actually is. Image courtesy of Deep Homage (https://www.youtube.com/c/DeepHomage)

The new model setting is capable of training a 1024×1024 resolution model on a NVIDIA GPU with as little as 8GB of VRAM. However, this is a low spec for such a resource-heavy model, and would need to be run at a batch size of just 2, and in a Linux environment (which doesn't reserve VRAM for system usage).

Prominent deepfaker Deep Homage has been one of the first to experiment with the 1024 model. The lower part of the above image is a 1-1 representation of a live training preview (transforming Theresa Russell's 1985 performance as Marilyn Monroe into Monroe herself), from the very earliest stages of training. The training session took place on a NVIDIA A600 with 48GB of VRAM, at a batch-size of 10.

Matt Tora, the model's developer and originator, has at least started it running on a card as low as NVIDIA's GTX 1080 (8GB of VRAM, the very least you will need to run this model, at a batch size of 2, under Linux).

DeepFaceLab's table of increase in training image size, compared to 1024px native training resolution. These are broadly representative input/output sizes in the five years since deepfakes emerged, though better-specced deepfakers also use 384, 768, and various other resolutions.

Tora states:

'The 1024 preset is a symmetrical encoder/decoder network that trades off filter count for dimensional space. It also replaces the fully-connected layers with convolutions to reduce the memory overhead at the center of the model.

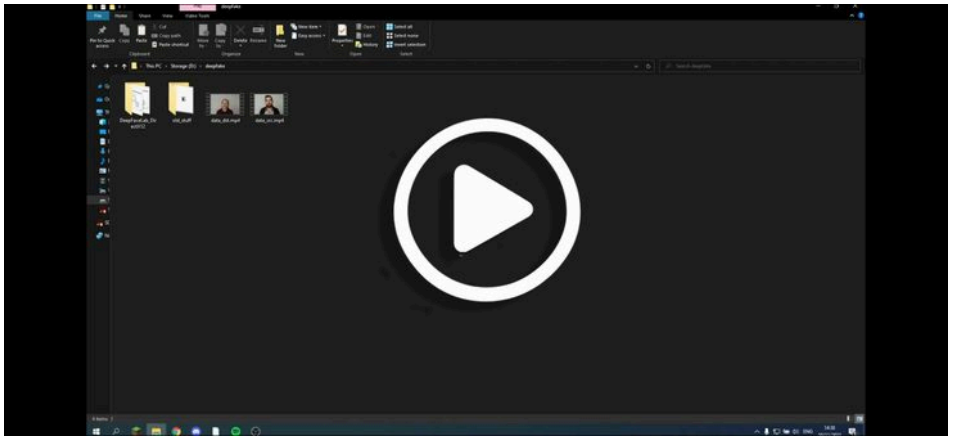
'This helps to create a model that can train to high resolutions with a relatively small memory footprint. Whilst it is possible, within the correct circumstances, to train this model on an 8GB card at very low batch-sizes, it is recommended to use a GPU with more than 8GB.'

The model, which has been available for some time to FaceSwap's Patreon supporters, enters the main branch on 16th July 2022.

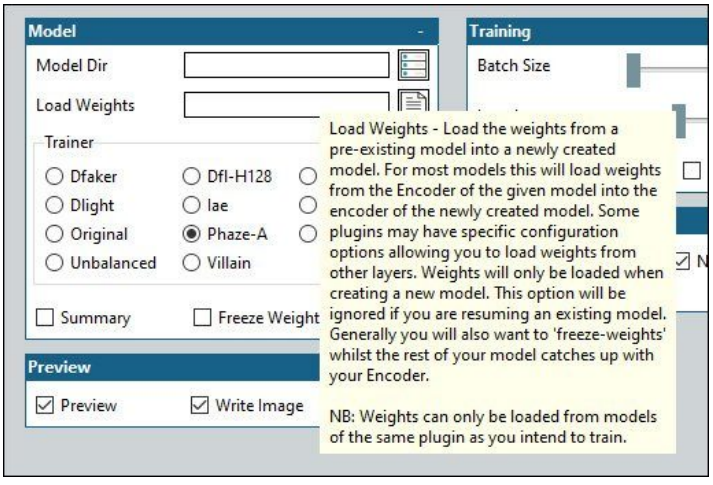
High Demands

A 1024 pipeline calculating loss values at this resolution and scale is likely to entail training times of several months, even on the highest-specced hardware. As resolutions increase, the days of training a new deepfake model 'from scratch' may be gone.

DeepFaceLab already maintains a culture of model re-use, as well as dedicated pre-training facilities within the software. By using pre-trained models, it's possible to exploit swap information already gained at great expense of time and resources, from advanced models trained on much better hardware than is likely to be available to the average user.



FaceSwap's Matt Tora has implemented an alternative scheme, where the weights of well-trained models can be imported into a new model, kick-starting training and cutting training times notably.



Importing trained weights from a fully-trained model into a new FaceSwap model.

‘The main difference’, Tora says ‘is that DeepFaceLab is training on many identities, while FaceSwap is re-using weights that have been trained on just two identities, which may help to obtain a more accurate resemblance. Due to the large training times involved, it’s difficult to conduct the lengthy testing that could really establish that one method is superior to the other, so that remains an open question.’

Additionally, Training at 1024px notably increases the minimum effective size for face-set images. Tora comments:

‘Basically, to get the best out of a 1024px model, you are going to want faces extracted that are, at a minimum, 1024px across each size. On 1080p footage, that is nigh-on impossible except in extreme close-ups.’

For the ‘deepfake recasting’ crowd, this limits the effective source data to 4K-resolution Blu-ray output, and increases the likelihood of having to deal with High Dynamic Range (HDR) source material, which produces unsuitable extractions without laborious pre-processing.

‘There are techniques,’ Tora says, ‘that attempt to map HDR footage to LDR, but these could best be described as hit-and-miss.’

Tora believes that 1024-based pipelines are better-suited to situations where the user is actually originating the source footage, rather than trying to manipulate Hollywood movies. Besides the increased hardware demands, this additional new pain-barrier is likely to limit 1024px+ deepfake model training to dedicated VFXers, rather than casual hobbyists.

‘The main problem with 1024px.’ Tora observes, ‘is your average user just does not have the time to either find useful data for it, nor the time to train it.’

Conclusion

We set out to consider whether autoencoder-based deepfakes really are likely to keep getting better, as current headlines predict. Perhaps the answer lies in the ongoing migration of the most talented deepfake developers and practitioners towards legitimate industry employment (or self-employment).

Deepfake research and development is expensive, and the proprietary systems that emerge from this investment of resources are either not likely to be put in the public domain at all, or else will require such considerable (and costly) computing power as to become out of reach to the casual user.

This slow trend towards a ‘de-democratization’ of open source image synthesis development is exemplified by the impressive offerings from OpenAI, such as the GPT and DALL-E series. In such cases, the source code is sometimes made available, but the weights that actually power the transformations cost millions to train — and they’re usually firewalled behind a metered (and profitable) API.

As the barrier to entry rises, in tandem with the public’s growing powers of discernment in regard to facial video synthesis, the deepfakes we’ll be seeing in movies and TV over the next five years are likely to leave the current crop of ‘hobbyist’ videos looking quite dated, and not so convincing as when they first appeared.

Without a substantial budget at their disposal, it simply may not be possible for casual deepfakers to keep up with the pros.

This means that while deepfake porn is unlikely to go away by any other method than the growth of legislation and enforcement, it isn’t likely to get much better either — certainly not in comparison to improvements in the quality of professional deepfakes in movies, TV and advertising.

Thanks to Matt Tora, Bryan Lyon, Dogan Kurt, Deep Homage, and Christian Darkin for their input and contributions to this article.