Fine-Tuning in Machine Learning

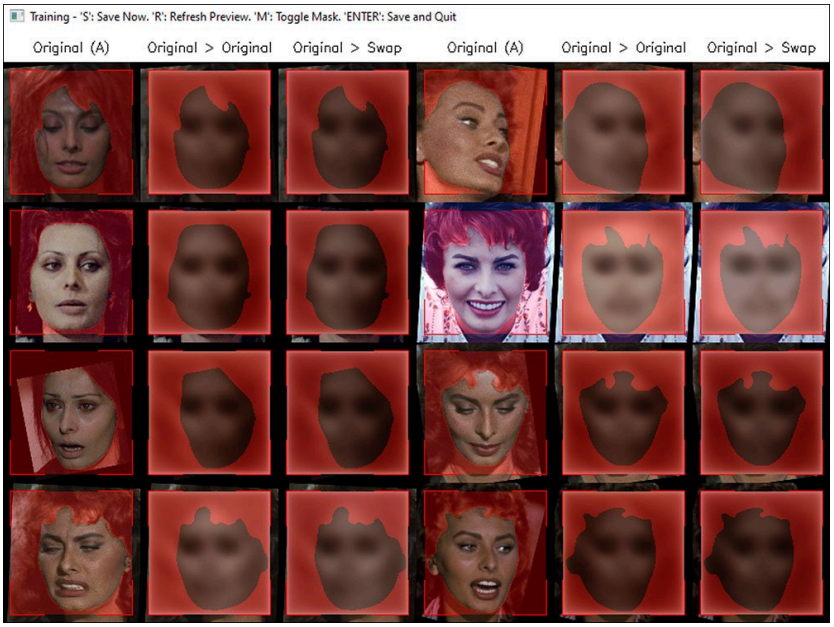
Originally published at https://blog.metaphysic.ai/fine-tuning-in-machine-learning/

January 9, 2023

Fine-tuning is the act of taking a trained generative machine learning model and adding new information to it, so that it can perform tasks that it was not originally trained for – such as reproducing a specific, perhaps obscure person that was not originally present in the database on which the model was first trained.
Fine-tuning is currently almost as much an art as a science, and has to contend with the fairly rigid architecture of a fully-trained AI model – which is not expecting to be revised *post-facto*.
Let's take a look at the essentials of what's involved.

Changing Face

Training a neural network from scratch is usually a labor-intensive, expensive and time-consuming endeavor. To boot, one is often 'reinventing the wheel' in the process, since the network will need to be taught, *every single time*, some of the most basic features that will be required in a final generative model.
One example of this is found in deepfake models that are trained entirely from zero. In the early phases of the training process, the material being discovered by the growing network is arguably very generic, in that it is learning that faces have two eyes, a centered nose underneath, a mouth, and some kind of delineation at the outer edges.



At the very beginning of a 'from scratch' deepfake training session, the previews are monotonously similar, as the neural network slowly learns to construct a 'basic' face. Some contend that even at this rudimentary stage, identity-specific foundations are being laid down, and broad parameters established which can not be completely overwritten later by subsequent identities. Others contend that the identities of these 'early faces' are so vestigial that no-one is likely to perceive their influence, when using a model pre-trained on generic, multiple, or different identities. In this case, the distribution being used is FaceSwap.

Many in the casual deepfake and professional VFX community believe that it is not worth constantly re-treading this ground for new models, and subscribe to the practice of *pre-training*, where a partially trained model is used as the basis for a specific new model.
For instance, DeepFaceLab (DFL) offers a generic model pre-trained on NVIDIA Labs' Flickr Faces HQ (FFHQ) dataset – a diverse and high-volume facial image collection featuring 70,000 high quality 1024×1024px images. DFL users can set this as a starting template for their face-swapping autoencoder models, and benefit from the model's innate per-trained understanding of all kinds of faces.

<YOUTUBE LINK OMITTED>

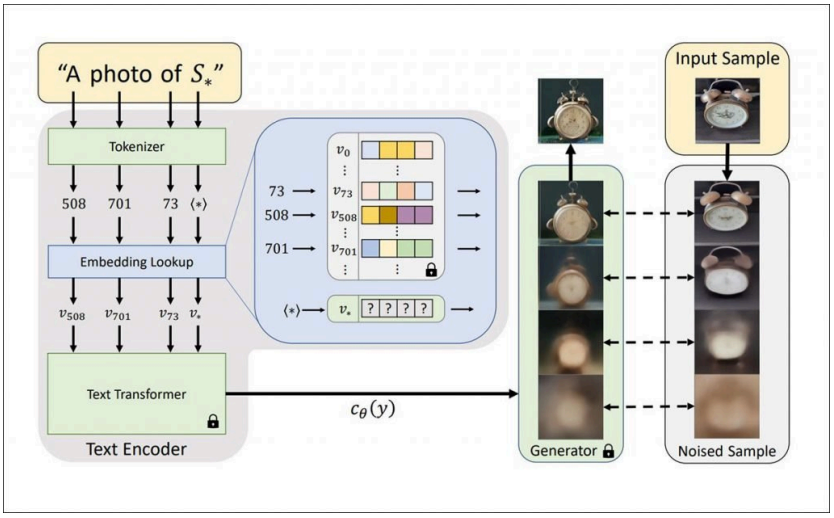
Since the identities in FFHQ are so varied, the assumption is that the features contained in this 'nascent' model are entirely generic, and can easily be 'taken over' by the two identities that the user chooses to swap, allowing for shorter training times.
This is the preemptive end of 'continuing' an already-existing model. In this case, the model has been trained only to a rudimentary quality, but has obtained a sketch-like comprehension of human facial physiognomy.
Though re-training represents a kind of fine-tuning (since some of the groundwork has already been laid with the seed model), it could more accurately be called 'major tuning'.

Fine-Tuning – And the Alternatives

The term 'fine-tuning' is more applicable to cases where the model is substantially or entirely complete, and where practitioners wish to benefit from the effort expended on the model by adding their own information to it.
The primary motivation for the practice is that the end-to-end training of a truly capable and complex model can be so extraordinarily effortful and expensive that it's not within the reach either of amateurs or of smaller research communities.
In such a case, the only full-blooded way to truly incorporate novel data into an already-trained model is to *resume the training of the existing model*, and introduce the new data in the process. This, in the strictest sense, constitutes fine-tuning.
An example of the scale of the challenge is the trained models behind the Stable Diffusion text-to-image and image-to-image generative system. The initial V1.0 release required 256 A100 GPUs (each with 40GB of VRAM) to train, running simultaneously for 160 hours, at an estimated cost of $600,000.

An example of fine-tuning, in this case with Google's DreamBooth. The four images on the left are trained into an already-existing and fully-trained latent diffusion model, making the model less performant overall, but suddenly able to insert the novel data into any variety of situations. Source: https://huggingface.co/docs/diffusers/training/dreambooth

Though this is not an easily repeatable achievement, there are actually quite a number of less exorbitant ways in which users can introduce their own material to the models that power the system. Some of them are neither difficult, expensive, nor particularly time-consuming, and mostly involve the creation of modifications to the way that Stable Diffusion associates image content with text content.
One approach of this type is called a hypernetwork, and was introduced by Tel Aviv University and NVIDIA shortly after the public release of Stable Diffusion in August of 2022.



Conceptual architecture of a hypernetwork for Stable Diffusion. Source: https://textual-inversion.github.io/

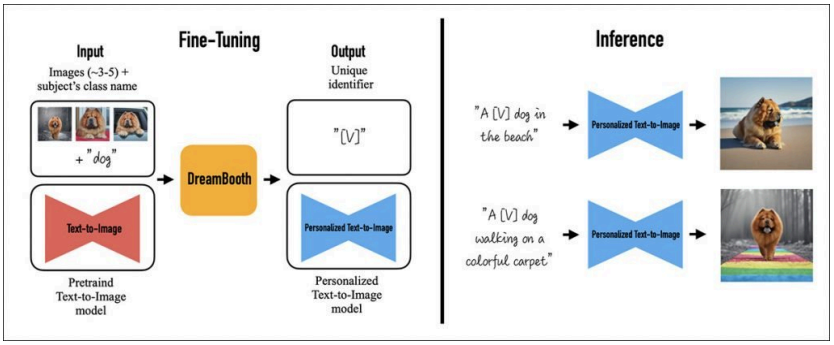
A hypernetwork intervenes in the way that the generative system looks for embeddings in the latent space (i.e. image content linked to semantic concepts, such as *'dog'*), and effectively creates new pathways through *already existing content*.
The user's part is to supply some images that they wish to be incorporated into Stable Diffusion. These will be used to condition the lightweight text encoder, so that it produces something novel and user-defined, and which was not native to the original training.

<YOUTUBE LINK OMITTED>

Nonetheless, this process does not actually affect the core trained data, and therefore has a limited or non-existent negative effect on general usage of the system (unlike 'real' fine-tuning – see 'Collateral Damage' below).
To boot, the files produced by a hypernetwork are very small and portable, and in many cases can be trained on consumer-level hardware in an order of hours, rather than the weeks or months that are more typical of full-fledged fine-tuning.
On the negative side, hypernetwork output is generally thought to be below the qualitative standard of higher-effort approaches such as genuine fine-tuning – and also of the most popular method of 'invading' Stable Diffusion – Google's DreamBooth.

DreamBooth

Released in late August of 2022, hard on the heels of the open-sourcing of Stable Diffusion, Google Research's DreamBooth performs genuine fine-tuning, in that it loads a fully LAION-trained checkpoint, usually weighing between 4GB-7GB, and *directly trains new material into it*.



DreamBooth requires the user to invent tokens (or 'unique identifiers') associated with the new images they are inserting into the original trained model. A unique token will 'summon up' the post-trained data at inference time, at the user's request. Source: https://dreambooth.github.io/

To boot, a customized DreamBooth model can be trained in as little as half an hour, depending on the number of user images, and various other settings available in popular implementations (though better results generally require high-effort curation and a training period of a few hours).
The user-material to be introduced into Stable Diffusion via the token and the images must belong to a class that already exists within the semantic hierarchy of the trained model, such as *'animal'*, *'man'*, *'woman'*, *'object'*, etc. In this way, the alien data becomes associated with the rich embeddings for its related class, and can benefit from the generative and interpretive power of that class – without having to do any real heavy lifting. Effectively, the new data is a kind of 'stowaway' on an already-existing class.
Not only has DreamBooth now become the *de facto* method of fine-tuning Stable Diffusion on user-contributed images, but the medium-level complexity of getting it to work (i.e., in a Google Colab, or within the generally-modest specs of a consumer GPU) has spawned a slew of 'middlemen' SaaS services and products, such as the controversial Lensa, many of which do no more than provide a friendly user-interface to the Colab scripts (which can otherwise be run without charge).
Though DreamBooth's relatively high compute requirements seemed destined to leave it in the shadow of less resource-intensive approaches when it first appeared last summer, the general consensus since then is that it provides superior results in terms of aesthetics and subject-accuracy.
Negatively, the process produces large files, usually of the same size as the 4GB+ checkpoints which were the starting point for a training session. Therefore even a modest collection of models is likely to eat up a lot of the user's disk space, and also to be less portable than methods based around text embeddings, such as the fairly lightweight textual inversion.

Collateral Damage

There is a difference between *resuming training* and fine-tuning. When you have access to the original training data, there is usually only a minimal penalty for interrupting training and starting it up again later.
If you're willing to save the complete end-session weights (they usually take up a lot of additional disk space), there's ordinarily no penalty at all, as the system can resume exactly where it left off, by loading those saved weights and re-initializing the paused session.
This is a very different situation from fine-tuning, where the user does *not* usually have access to the original data, but only to the weights that were derived from that data during the primary training.
In this scenario, the system has no opportunity to 'rethink' how the original data still relates to the entirety of the data, in light of the additional information that the user is adding, since the original source data is now missing, and cannot be properly reevaluated.
Therefore, the fine-tuned model's capacity for generalization tends to be adversely affected by the fine-tuning process, relative to the capabilities of the original model that was used as a starting point. In general, the fine-tuned model will perform well with respect to the new data that's been introduced, but not quite so well as the original model for more general usage.
For this reason, you can't just fine-tune (for instance) Stable Diffusion and replace the base model every time you add new data, because the *overall* capabilities of the model will deteriorate more with each fine-tuning, and your model will eventually only perform very well on queries related to data that you added yourself.
Additionally, the benefit of 'piggy-backing' off the quality of those expensive original weights will disappear over time, as the quality of those *original* weights will drop incrementally with each new fine-tuning session.

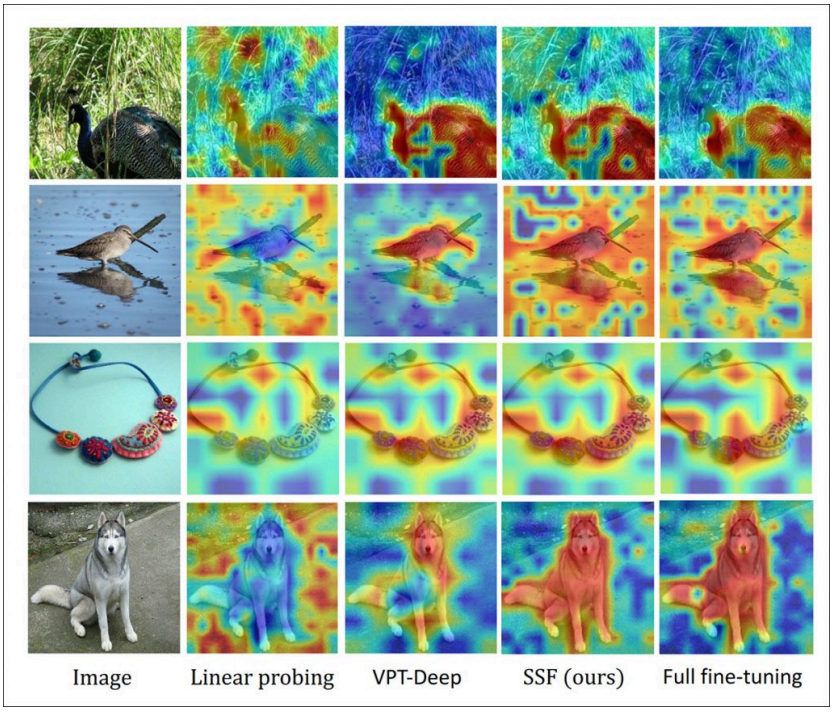
Don't Try This at Home

Since it would be better to fine-tune on the original data, and since, in many cases, the data is publicly available, why not just download it and avoid this particular negative effect of fine-tuning?
In most cases, the problems are logistical, in that the datasets are vast, often in the multi-terabyte range, and will require an amount of storage that's not commonly at the disposal of the mere machine learning enthusiast.
More importantly, to replicate the conditions under which the original data was trained, the hobbyist or small-scale fine-tuner will need comparable hardware resources. Unless the available hardware can replicate the original training environment, it won't usually be possible to 'continue' training

somebody else's model, even if all other requisites are met.
Since, in the case of models such as those created for Stable Diffusion, this may entail renting a gaggle of the most expensive GPUs in the world and running them for weeks or even months, it's not a realistic prospect – even if all the details (learning rate schedules, loss functions, etc.) necessary to 'resume' (rather than fine-train) have been published.

Linear Probing

Besides conditioning on the text in Variational Autoencoders (VAEs) or intervening in the more fluid text-based component of CLIP or other image/text components (as detailed above), there some other less invasive approaches to revising an existing model.
One of these is Linear Probing, where the changes are effected not on the entirety of the trained model, but only on the penultimate layer of the neural network (i.e., the last layer before the output layer that's exposed to the person using the generative system).



Attention maps derived from a number of approaches, including Linear Probing. Here, as we can see, full fine-tuning has obtained a better recognition of the subject. Source: https://arxiv.org/pdf/2210.08823.pdf

Though Linear Probing is popularly thought to obtain inferior results with respect to full fine-tuning, the exact mechanics of post-training revision are not entirely established, and some practitioners have found that fine-tuning itself can be more damaging to the integrity of the model than linear probing.

Conclusion

Just as it is difficult (though not impossible) to rewrite our own DNA, it is challenging to force an already-trained model to revisit the parameters and weights that were established originally – even under the best available conditions.
The negative collateral effects of fine-tuning signifies that it can't be done iteratively without obtaining ever-diminishing returns. This means that each time you want to add something to an existing full-scale model, you effectively need to discount ever using the fine-tuned model for its original broader purpose.
To boot, you're likely to need a disk-draining full-size model checkpoint for each single revision you make, and it won't be practical to build and improve the model over time – except by improving your methodology and starting the fine-tuning over again, using the 'full fledged' base model, from the beginning.
Like completed organic systems, original fully-trained models are quite brittle in terms of their fundamental composition and functionality. Once they're formed, intervention is difficult and often deleterious, with the challenge analogous to attempting to insert a functional additional cog into a very compact and complex Swiss watch, or to revising the deepest foundations of a building without destabilizing it.