Transflow Learning: Repurposing Flow Models Without Retraining

Andrew Gambardella
University of Oxford
gambs@robots.ox.ac.uk

Atılım Güneş Baydin
University of Oxford
gunes@robots.ox.ac.uk

Philip H. S. Torr
University of Oxford
phst@robots.ox.ac.uk

Abstract

It is well known that deep generative models have a rich latent space, and that it is possible to smoothly manipulate their outputs by traversing this latent space. Recently, architectures have emerged that allow for more complex manipulations, such as making an image look as though it were from a different class, or painted in a certain style. These methods typically require large amounts of training in order to learn a single class of manipulations. We present Transflow Learning, a method for transforming a pre-trained generative model so that its outputs more closely resemble data that we provide afterwards. In contrast to previous methods, Transflow Learning does not require any training at all, and instead warps the probability distribution from which we sample latent vectors using Bayesian inference. Transflow Learning can be used to solve a wide variety of tasks, such as neural style transfer and few-shot classification.

1. Introduction

One of the greatest challenges in modern machine learning is few-shot learning [13, 14]. Whereas a human is capable of learning a task such as handwritten digit recognition after only having seen a few samples of each digit, even simple machine learning classifiers require training multiple epochs over a relatively large dataset. When it comes to more complicated tasks, machine learning algorithms become even more data-hungry—whereas humans can learn to play Atari games in a matter of minutes, even the most sample-efficient of reinforcement learning algorithms take hundreds of hours of gameplay [24].

What advantages do humans have over machines that allows us to consistently beat them with regards to sample efficiency? One might argue that humans are constantly utilizing their experiences in other domains in order to draw parallels between tasks. Even when it comes to very disparate sets of tasks, such as “natural language understanding" and “video game playing," studies have shown that it is possible to transfer knowledge between these domains for major sample efficiency gains in both machine learning algorithms and human learning [24, 2]. The idea of taking knowledge from one domain or task, and using that knowledge in another domain, is known as “transfer learning” [20].

Figure 1: Transflow Learning allows us to warp the latent distribution of any trained invertible generative model, so that we can instead sample data similar to that we provide post-hoc. This works by treating the latent distribution as the prior in a Bayesian posterior inference setting where we condition on the provided data. In this example, given only 18–24 instances of images in the art domains on the left, a flow model trained on CelebA [17] is able to generate human faces with matching attributes, even though these attributes are not contained in the CelebA dataset.

When attempting to transfer knowledge about one task to another, two broad questions must be asked: how is the prior knowledge from other tasks stored, and how can it be used? We propose an answer to these questions in the context of generative models.

Modern generative models, such as Generative Adversarial Networks (GANs) [8], normalizing flow models [22], and

autoregressive models such as Conditional PixelCNN [25] differ in their learning mechanisms, but all share a common thread: they learn a function $f_{\boldsymbol{\theta}} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ which transforms samples of a latent random variable $\mathbf{z} \sim q(\mathbf{z})$, where $\mathbf{z} \in \mathbb{R}^d$ and $q(\mathbf{z})$ is a known distribution, to a data point (*e.g.*, an image) $\mathbf{x} \in \mathbb{R}^d$. The latent distribution $q(\mathbf{z})$ is usually a simple distribution such as the multivariate Gaussian, $\mathcal{N}(\vec{0}, I)$. When learning generative models we keep $q(\mathbf{z})$ fixed, and concentrate all of our efforts on optimizing the parameters $\boldsymbol{\theta}$.

In the context of above discussion on transfer learning, however, we take a different view. We wish to transfer knowledge from a trained generative model f, to some new task. To be concrete, assume we have a generative model that will output an arbitrary celebrity face when given a latent vector $\mathbf{z} \sim \mathcal{N}(\vec{0}, I)$ as input. Can we use the same generative model to only output celebrities with red hair? Can we output a celebrity which looks like an anime character? Can we even classify handwritten digits?

We find that all of the above and more is possible, with the condition that our generative model f is *invertible*. That is, we require an inverse function $\mathbf{z} = f^{-1}(\mathbf{x})$ which takes a data point (*e.g.*, an image) as input, and outputs the corresponding latent vector. Normalizing flow models [21] are the most natural class of such generative models, as they are by nature invertible, unlike other architectures such as GANs which can be inverted only under specific circumstances [5].

Our method works by taking the flow model and its parameters as fixed, and instead warping the latent distribution so that we can control the latent vectors that we sample. Specifically, we treat the latent distribution $q(\mathbf{z})$ as a prior distribution in a Bayesian inference setting where we update it to a posterior $q(\mathbf{z}|\boldsymbol{\zeta}_{1:m})$ conditioned on some observed data samples $\mathbf{x}_{1:m}$ mapped to corresponding latent vectors using $\boldsymbol{\zeta}_i = f^{-1}(\mathbf{x}_i), i = 1, \ldots, m$. We call our method Transflow Learning, as it uses flow models to perform tasks for which they were not originally trained.

In Section 2 we cover some essential concepts, followed by Section 3 where we describe the mechanism by which we warp $q(\mathbf{z})$: Bayesian inference in the latent space. Section 4 covers related work. In Section 5, we provide example use cases in which knowledge can be transferred, specifically modeling distributions other than the training data, and solving downstream tasks. Section 6 discusses future work followed by conclusions in Section 7.

2. Background

2.1. Normalizing Flow Models

A normalizing flow is a series of learned invertible transformations which can transform one probability distribution into another. If random variable $\mathbf{z}_0$ with associated probability density $q_0(\mathbf{z}_0)$ is put through a series of invertible transformations $\{f_1, ..., f_K\}$ so that

$$\mathbf{z}_K = f_K \circ ... \circ f_1(\mathbf{z}_0) = f(\mathbf{z}_0) \tag{1}$$

then we have

$$q_K(\mathbf{z}_K) = q_0(\mathbf{z}_0) \prod_{k=1}^{K} \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|^{-1} . \tag{2}$$

Each f_k contains learnable parameters, which are typically learned by maximum likelihood. The flow is "normalizing" because each $q_k(\mathbf{z}_k)$ defines a valid probability distribution [22].

In this paper we designate $\mathbf{z} = \mathbf{z}_0$ as the latent vector and $\mathbf{x} = \mathbf{z}_K$ as the data point produced by the transformation $\mathbf{x} = f(\mathbf{z})$. We require two properties of flow models: that f is invertible (which is true as it is the composition of invertible functions), and that vectors around $f^{-1}(\mathbf{x})$ correspond to data close to $\mathbf{x}$, even for $\mathbf{x}$ that the flow model had not seen during training (which holds empirically as long as $\mathbf{x}$ is not extremely unnatural).

2.2. Posterior Inference

Bayesian inference is a powerful tool for reasoning about probability distributions [7]. Core to the idea of Bayesian inference are the concepts of the *prior*, which is a probability distribution $p(z)$ that we assume exists before having seen any data, and the *posterior*, which is a probability distribution $p(z|x)$ that we obtain after having observed some data (or evidence) x with a *likelihood* $p(x|z)$. These are related with the relationship $p(z|x) = p(x|z)\, p(z)/p(x)$. The likelihood is essentially a weighting that tells us to what degree the prior must be moved after having observed some evidence.

Also important is the concept of a *conjugate prior* [23], which means that with certain choices of prior and likelihood distributions, we can ensure that we have a closed-form analytical expression for the posterior distribution. In particular, we will need to use the fact that if we have a prior which is a multivariate Gaussian and a likelihood function which is a multivariate Gaussian with a known covariance matrix, then the posterior is also guaranteed to be a multivariate Gaussian.

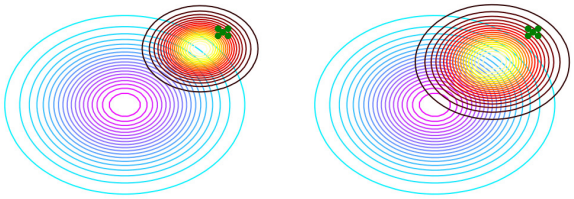
Using knowledge of conjugate distributions is particularly attractive, as it will allow us to solve for the posterior parameters analytically. Without a certain specification of likelihood function, we would need to resort to sampling-based methods such as stochastic variational inference [10] in order to obtain an approximation to the posterior.

3. Algorithm

Our algorithm is detailed in Algorithm 1. The key insight is to treat the underlying latent distribution $q(\mathbf{z})$ of a flow

Algorithm 1 Transflow Learning

Input: Trained flow model $\mathbf{x} = f(\mathbf{z})$ with inverse $\mathbf{z} = f^{-1}(\mathbf{x})$, where $\mathbf{x}$ are data and $\mathbf{z}$ are latents
For each set of evidence data $\mathbf{x}_{1:m}$, find corresponding latents $\{\boldsymbol{\zeta}_1, ..., \boldsymbol{\zeta}_m\} = \{f^{-1}(\mathbf{x}_1), ..., f^{-1}(\mathbf{x}_m)\}$
Construct posterior $q(\mathbf{z}|\boldsymbol{\zeta}_{1:m}) = \mathcal{N}(\mu_p, \boldsymbol{\Sigma}_p)$ by computing $\boldsymbol{\mu}_p$ and $\boldsymbol{\Sigma}_p$ analytically
Obtain samples from the data posterior $p(\mathbf{x}|\mathbf{x}_{1:m})$ by computing $\mathbf{x} = f(\mathbf{z})$ where $\mathbf{z} \sim q(\mathbf{z}|\boldsymbol{\zeta}_{1:m})$



$\lambda = 0.3m$ $\lambda = 0.7m$

Figure 2: Transflow Learning finds a posterior (top right) in between the prior (bottom left) and the evidence (cross mark). We see that as λ becomes larger, the mean of the posterior becomes closer to the mean of the prior, and the covariance of the posterior becomes larger, but in both cases the evidence can be sampled with relatively high probability.

model as a prior, where usually $q(\mathbf{z}) = \mathcal{N}(\vec{0}, I)$. We are then interested in obtaining a posterior over the latent space of the flow model $q(\mathbf{z}|\boldsymbol{\zeta}_{1:m})$, conditioned on $\boldsymbol{\zeta}_i$, which are some data observations $\mathbf{x}_i$ mapped to the latent space using $\boldsymbol{\zeta}_i = f^{-1}(\mathbf{x}_i), i = 1, \ldots, m$.

In other words, we provide evidence in the form of new latent vectors and, conditioned on this evidence, we find a posterior distribution over the flow model's latent variables. This effectively gives us a new generative model from which we can sample data resembling the evidence. In order to accomplish this, we require our generative model f to be invertible.

3.1. Computing the Posterior over Latent Vectors

As most implementations of normalizing flow models use a multivariate Gaussian to model $q(\mathbf{z})$, with an appropriate choice of likelihood function we can compute the posterior analytically. Assume we are given a trained flow model, and that during training the model was shown latent vectors from $\mathbf{z} \sim \mathcal{N}(\vec{0}, I)$. If we use a multivariate Gaussian likelihood function with covariance matrix Σ, then the posterior over the latent vectors is also a multivariate Gaussian, so that $q(\mathbf{z}|\boldsymbol{\zeta}_{1:m}) = \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$, and its parameters are given by the formulae:

$$\boldsymbol{\mu}_p = (\boldsymbol{\Sigma}_0^{-1} + m\boldsymbol{\Sigma}^{-1})^{-1}(\boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0 + m\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{\zeta}}) \quad (3)$$

$$\boldsymbol{\Sigma}_p = (\boldsymbol{\Sigma}_0^{-1} + m\boldsymbol{\Sigma}^{-1})^{-1} \quad (4)$$

where m is the number of observed data points, $\bar{\boldsymbol{\zeta}}$ is the mean of observed latent vectors $\boldsymbol{\zeta}_{1:m}$, $\boldsymbol{\mu}_0$ is the prior mean and $\boldsymbol{\Sigma}_0$ is the prior covariance matrix. As we know that $\boldsymbol{\mu}_0$ is equal to $\vec{0}$ and $\boldsymbol{\Sigma}_0$ is the identity matrix, we can further simplify these formulae:

$$\boldsymbol{\mu}_p = (I + m\boldsymbol{\Sigma}^{-1})^{-1}(m\boldsymbol{\Sigma}^{-1}\bar{\boldsymbol{\zeta}}) \quad (5)$$

$$\boldsymbol{\Sigma}_p = (I + m\boldsymbol{\Sigma}^{-1})^{-1} \quad (6)$$

The choice of $\boldsymbol{\Sigma}$ here serves as a hyperparameter. One natural choice would be to use a scalar matrix, which implies that we would like to keep the latent vectors uncorrelated and weighted identically. With likelihood covariance $\boldsymbol{\Sigma}$ being set to λI, with λ a scaling hyperparameter, the posterior parameters become simple to compute:

$$\boldsymbol{\mu}_p = (I + \frac{m}{\lambda}I)^{-1}(\frac{m}{\lambda}I\bar{\boldsymbol{\zeta}}) = \frac{\frac{m}{\lambda}\bar{\boldsymbol{\zeta}}}{\frac{m}{\lambda} + 1} \quad (7)$$

$$\boldsymbol{\Sigma}_p = (I + \frac{m}{\lambda}I)^{-1} = \frac{1}{\frac{m}{\lambda} + 1}I \quad (8)$$

There are three special cases that we can examine:

1. If we let $\lambda = c$, where c is a small constant relative to m, we can see that the mean of the posterior will be close to the sample mean, and the covariance of the posterior will be very small.

2. If we let $\lambda = m$, we can see that the mean of the posterior will be locked onto $0.5\bar{\boldsymbol{\zeta}}$, and the covariance will remain constant at $0.5I$, regardless the value of m.

3. If we let λ become arbitrarily large, then the posterior mean will be close to $\vec{0}$ and the posterior covariance will be close to I, *i.e.*, all conditioning is completely ignored and we get back the original flow model.

In other words, low values of λ relative to m will give a posterior that is close to the sample mean, and with very low covariance, whereas high values of λ will become more and more like the original flow model.

3.2. Computing the Posterior Predictive Distribution

The posterior predictive distribution evaluates the probability of a possible unobserved value $\boldsymbol{\zeta}$ of latent vectors conditioned on the observed values $\boldsymbol{\zeta}_{1:m}$. It is obtained by marginalizing the distribution of $\boldsymbol{\zeta}$ over the posterior $q(\mathbf{z}|\boldsymbol{\zeta}_{1:m})$:

$$q(\boldsymbol{\zeta}|\boldsymbol{\zeta}_{1:m}) = \int p(\boldsymbol{\zeta}|\mathbf{z}, \boldsymbol{\zeta}_{1:m})\, q(\mathbf{z}|\boldsymbol{\zeta}_{1:m})\, \mathrm{d}z \quad (9)$$

In the setting of multivariate Gaussian conjugate priors that we described in the previous section, its form is known and easily calculated:

$$\boldsymbol{\mu}_{pp} = \boldsymbol{\mu}_p = \frac{\frac{m}{\lambda}\bar{\boldsymbol{\zeta}}}{\frac{m}{\lambda}+1} \tag{10}$$

$$\boldsymbol{\Sigma}_{pp} = \boldsymbol{\Sigma}_p + \boldsymbol{\Sigma} = \frac{1}{\frac{m}{\lambda}+1} I + \lambda I \tag{11}$$

Its mean is exactly the same as that of the posterior, and its covariance is also identical, save for the extra λI term, which accounts for uncertainty in the parameters. We can use the posterior predictive distribution for a wide variety of tasks unrelated to sampling, and in Section 5.3 we will show how to do MNIST [15] classification without training.

3.3. Properly Setting λ

The hyperparameter λ determines the variance of the likelihood that is used in the conditioning on observed data points. This is similar to the use of approximate Bayesian computation [19, 26] likelihoods in Bayesian inverse graphics [18], where the variance of the likelihood plays the role of a "tolerance" in judging how closely an image generated by the generative model matches an observed image. High tolerances admit generation of images that do not closely match the observation, whereas low tolerances push inference towards closely mimicking the observation while reducing the sample efficiency in complex image settings.

From the perspective of the analytical posteriors introduced in Section 3.1, while setting λ to a high value may seem like a mistake due to the behaviour of the posterior as λ grows larger, we argue that low values of λ are even more dangerous. As flow models learn invertible maps, the dimensionality of the latent vectors must be equal to that of the output. For example, if we wish to output full-colour, 256 by 256 images, then the dimensionality of the latent space is $3 \times 256 \times 256 = 196,608$. In contrast, the dimensionality of the latent space for a typical GAN [8] or VAE [12], which do not have this restriction, is around 100.

The high dimensionality of flow model latent vectors implies that vectors which should be "close" in that they share similar features in image space will be very far in the L_2 sense. This has implications for the sample mean of latent vectors, $\bar{\boldsymbol{\zeta}}$, which will have a smaller L_2 norm as more observed data points are averaged, due to the curse of dimensionality spreading supposedly "similar" vectors in different directions relative to the origin. As vectors with smaller norm are closer to the mean of Gaussian on which the flow model was trained, $\vec{0}$, this has the effect that conditioning on many data points will give a posterior mean which is very "generic," as it has unreasonably high probability under the original model (*i.e.*, much higher probability than a vector randomly sampled from $\mathcal{N}(\vec{0}, I)$). While this is not bad in and of itself (after all, the mean of the original distribution, $\vec{0}$, is the most generic image possible), this issue is further compounded by the covariance of the posterior shrinking as more data points are added, making it so that if λ is set too low, every sample from the distribution is extremely generic. Even though the mean of the posterior distribution has even smaller norm with larger values of λ, the larger covariance makes up for it.

In practice we find that a large range of settings of λ work, depending on the nature of the conditioning, but in general values which are in the range $[0.3m, 0.7m]$ are preferable. We explore the consequences of different choices of λ in Section 5.

4. Related Work

Image2StyleGAN [1] explored interpolations and embeddings of real images into the latent space of a GAN [8]. They found that while able to embed natural images almost perfectly, including those out of the distribution on which the GAN was trained, they were unable to do sensible latent space interpolations. Our method is able to do sensible interpolations between out-of-distribution datasets by interpolating the mean and covariance of their posterior distributions as we show in Figure 3. This method can be thought of as first projecting the out-of-distribution images onto the flow model manifold before interpolating.

Neural Style Transfer [6] is a method for re-rendering images with a different style, while also keeping the content similar. Our method can be seen as similar to Neural Style Transfer methods, with the "content" being provided by the flow model and the "style" being provided by the evidence. Unlike previous Neural Style Transfer methods which work on a single content image, we learn an entire distribution from which we can sample.

CycleGAN [27] and Few-Shot Unsupervised Image-to-Image Translation [16] are also methods for blending two unpaired datasets, but the aim is slightly different. Whereas these methods are capable of turning one specific image into that resembling a different class, we are able to generate many diverse samples of the class given as evidence. At the same time, our method would be unable to modify a single image in a meaningful way.

Glow [11] explored the use of manipulation vectors in order to induce specific attributes in images. Whereas their simple algebraic method required both positive and negative examples of the attribute they wished to express, we require only positive examples. We also obtain a full posterior distribution from which we can sample many diverse images, unlike their method which can only transform a single image.

The common thread between our work and previous works is that while many previous works showed manipulations of individual images using trained generative models, we are the first to combine pre-trained generative models

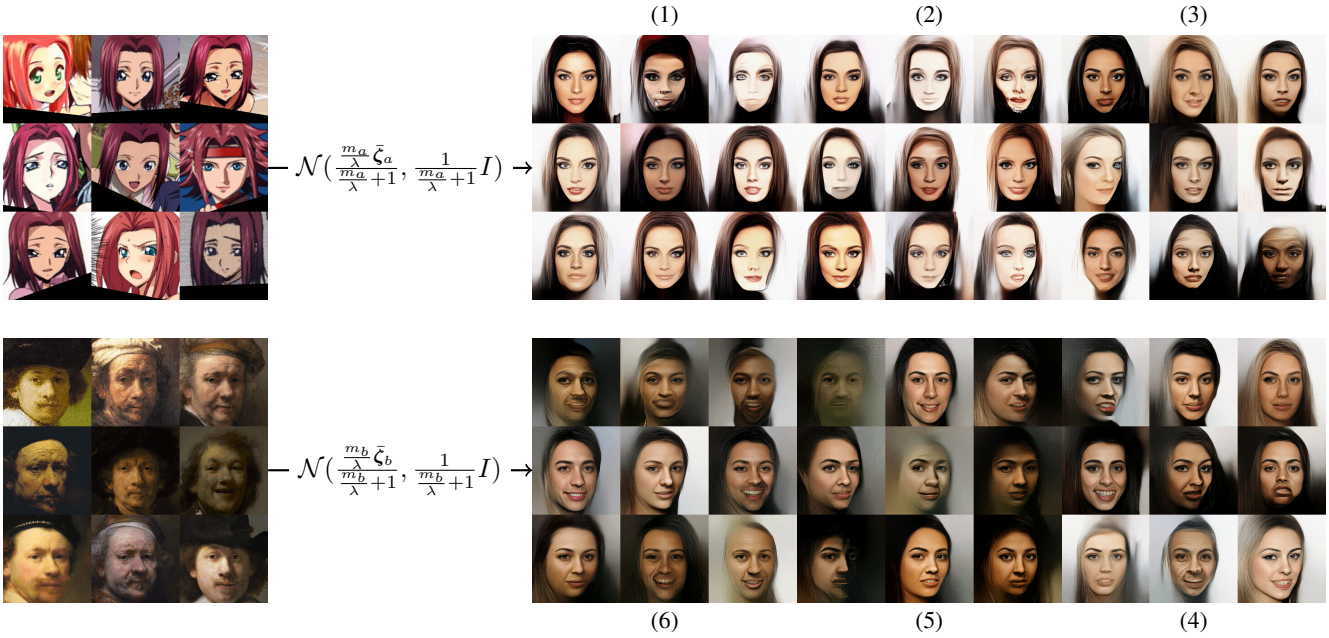


Figure 3: Interpolation between two sets of images far outside the training distribution, by first projecting onto the manifold of human faces, and then interpolating the parameters of the posterior distributions. Note that as the distribution gets closer to that of Rembrandt's self-portraits, the colours in the image get darker, men are sampled much more frequently, the hair is often gone from the samples (as Rembrandt often wore a hat which blended in with the background), and the sampled faces are more tilted towards the right. (View in numerical or reverse numerical order)

Figure 4: Direct interpolation between two images from the same dataset as in Figure 3. Note that many intermediate images are not faces.

with new data to create an entirely new generative model.

5. Experiments

While there are many different types of flow models such as NICE [3], Real NVP [4], and Flow++ [9], we chose to use Glow [11] for all of our experiments. This is an arbitrary choice mainly influenced by the public availability of a pre-trained model for Glow, trained on the CelebA dataset [17]. Flow models are currently prohibitively difficult to train, both in terms of time and compute requirements, and this study is solely interested in exploring transfer learning using existing models.

5.1. In-distribution Conditioning

A simple experiment to demonstrate the capabilities of Transflow Learning is to sample from some coherent subset of data within the CelebA dataset, such as people with red hair, people with glasses, or individual people. We found that for categories which are strict subsets of the training data, such as people with red hair, we could create a reasonable posterior distribution with both a low amount of data and a wide range of λ. In Figure 5 we show results from Transflow Learning given 5 images of people with red hair, a distribution which is wholly a subset of CelebA, and 21 images of greyscale human faces, a distribution which is not represented in the CelebA training set, but is also not too far off.

We also attempted to condition on natural faces with a large occlusion, and were surprised by the results. Figure 6 shows results when attemping to condition Glow on 25 images of President Obama with a large occlusion over his eyes. Transflow Learning was shockingly able to gener-

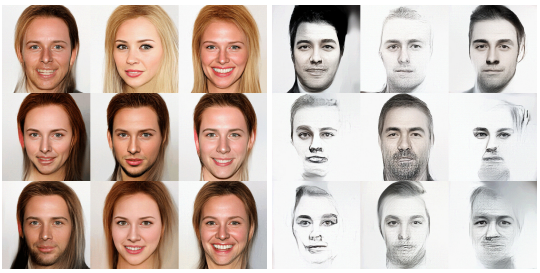


Figure 5: A flow model trained on CelebA, conditioned on relatively natural images. The images of people with red hair (left) are sampled from a posterior using only five images as evidence, showing that our model is very sample-efficient for strict subsets of the CelebA distribution. Greyscale images (right), despite not appearing in the CelebA training set, were also successfully captured by a Transflow Learning posterior.

ate images of men with a neon-green occlusion over their eyes, despite similar images clearly not being located in the CelebA training set. It is important to reemphasize at this point that Transflow Learning in no way modifies the flow model—amazingly, there was simply a region in the Glow latent space in which latent vectors corresponding to these images exist, and Transflow Learning was able to find a Gaussian covering this space. As the posterior contains elements of both the prior and the evidence, we expected the posterior to perform similar to inpainting, and were surprised to learn that the latent space of Glow was rich enough to be able to generate these images which were far outside of the training set. We only observed an inpainting-like effect for relatively high values of λ (*i.e.*, values close to m), but at that point the model had forgotten to also generate President Obama, and was generating seemingly random samples with a faint, translucent occlusion around the eyes.

Even more surprisingly, when we changed the occlusion so that it would be made up of random pixels as opposed to one solid colour, Transflow Learning was no longer able to generate human faces, even for relatively high values of λ. We believe that this effect is due to how unlikely the noisy occlusion is compared to the monochromatic occlusion. We found that latent vectors corresponding to a real image of President Obama, the same image of President Obama with a monochromatic occlusion, and an image of an anime character have the log-likelihoods of -284,462, -281,377, and -285,610 respectively. Notably, these are all contained in roughly the same range, and the image of President Obama with a monochromatic occlusion was actually more likely than the image without the occlusion. Conversely, the latent vector corresponding to an image of President Obama with a noisy occlusion has a log-likelihood of -333,436, a number which is completely off the charts. This effect pushes the posterior too far out, to the point that samples around the posterior mean no longer correspond to meaningful images. Indeed, the pattern around the eyes in the posterior samples also resemble patterns that appear for any image corresponding to a latent vector with extremely high magnitude, which are guaranteed to be unlikely.

5.2. Out-of-distribution Conditioning

We also found that Transflow Learning could generate samples of many types of images which are not strictly human faces. While generated images were often nonsensical when conditioned on images which could not be interpreted in any way as a face, we found that a wide variety of images, such as cartoon faces or paintings of faces, gave interesting results. In Figure 8 we show two examples of such a conditioning, on self-portraits of Rembrandt and images of an anime character.[1]

We found that the setting of λ was much more difficult in out-of-distribution scenarios. While with in-distribution conditioning we could freely set λ to any reasonable value and achieve sensible (although different) results, many settings of λ for out-of-distribution conditioning created distributions that were either too narrow or too much like the original flow model.

The CelebA dataset [17] is also strongly aligned, which created difficulty in conditioning on out-of-distribution datasets. We found that even for datasets that could be interpreted as human faces, sample quality decreased sharply in the presence of poorly aligned inputs. This posed particular difficulty when conditioning on anime faces, as the facial keypoint detector trained on human faces frequently mistook anime mouths for noses and chins for mouths, or more often failed to find a face at all.

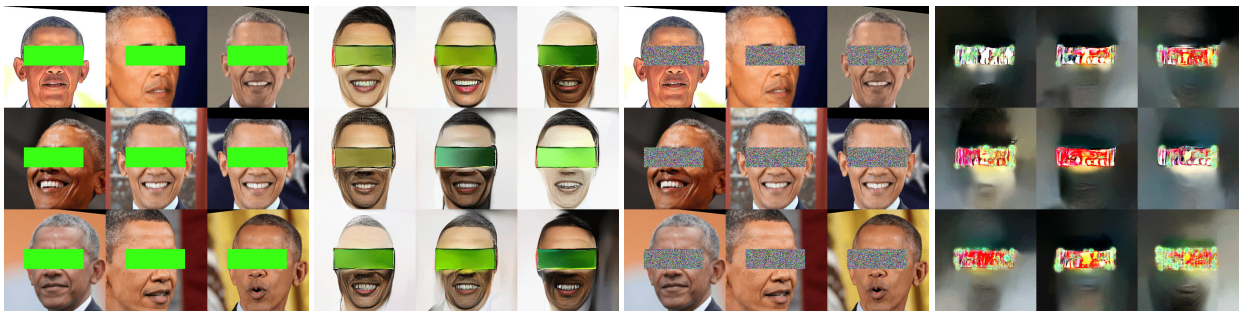
While samples from the flow model are visually meaningless when evidence cannot be interpreted as a human face, the learned posteriors can still be used for downstream tasks. In the next section, we will show that Transflow Learning can use a flow model trained on the CelebA dataset to do MNIST classification in a low-shot setting.

5.3. MNIST Classification

In order to classify MNIST digits through transfer learning with a pre-trained flow model, we must use the posterior predictive distribution given in Section 3.2. The workflow is as follows:

1. Take a flow model pre-trained on any dataset
2. Compute posterior predictive distributions conditioned on a number of observations from each class in MNIST, obtaining ten separate distributions

[1] Anime character images taken from the Anime Face Character Dataset: http://www.nurs.or.jp/~nagadomi/animeface-character-dataset/



(a) (b) (c) (d)

Figure 6: Even when providing Transflow learning with evidence that is far outside of the distribution on which the flow model is originally trained (a), we are able to learn a sensible posterior distribution (b). Evidence that is so unlikely that it could not have come from a natural image (c), however, causes the posterior mean to be too far from the mean of the original distribution, and output samples (d) are no longer meaningful, even for high values of λ.

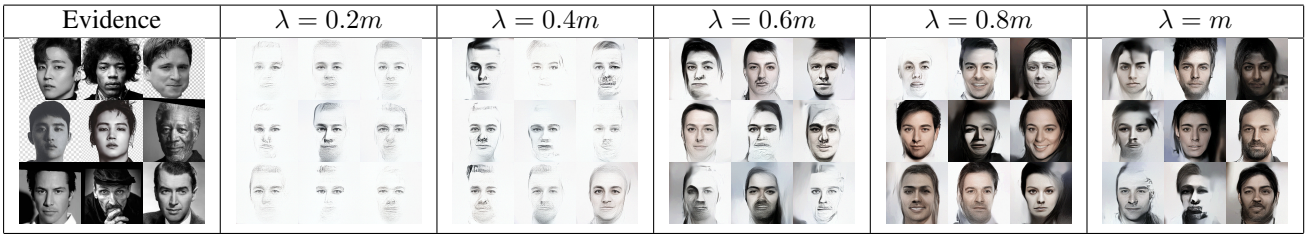


Figure 7: Varying the λ hyperparameter for a greyscale dataset. λ that is low creates images resembling pencil sketches, whereas λ that is high creates images with very subdued colors.

3. When given an image of a new digit $\mathbf{x}$, compute the probability of $f^{-1}(\mathbf{x})$ under each of the ten posterior predictive distributions
4. The new image $\mathbf{x}$ is classified as having come from the posterior predictive distribution under which it was the most likely

We compared Transflow Learning to k-nearest neighbors in both pixel and the flow model latent space on the task of m-shot MNIST classification. Our results are located in Table 1. We wish to emphasize that unlike previous methods using generative models for few-shot learning, we did *not* pre-train our flow model on the MNIST training set. For each experiment, we used the same implementation of Glow trained on CelebA and then showed each algorithm only m labeled images from each class in the MNIST training set.

It is also important to emphasize that no “training" in the traditional sense is done here whatsoever. In the Transflow Learning experiments, the labeled MNIST images are simply used to warp the latent distribution of the CelebA flow model (Figure 9). This has implications for transfer learning using large datasets as conditioning, as for a dataset of size n, we would only require n evaluations of the function

m	Ours	Pixel k-NN	Latent k-NN
1	**30.39%**	14.99%	27.78%
5	**46.39%**	32.99%	35.10%
10	**58.73%**	19.50%	40.40%
20	**65.35%**	22.83%	44.12%
30	**69.52%**	20.55%	47.58%

Table 1: Accuracy on single-shot and few-shot MNIST classification, given a flow model trained on CelebA. For all k-Nearest Neighbors experiments, k was set to 3 when possible, otherwise 1.

from data to latent variables, f^{-1}, in order to obtain a new classifier. Compared to the common practice of gradient-based fine-tuning of models with new training data, which requires several epochs of both costly forward and backward propagations, our method is exceptionally cheap in terms of number of function evaluations required.

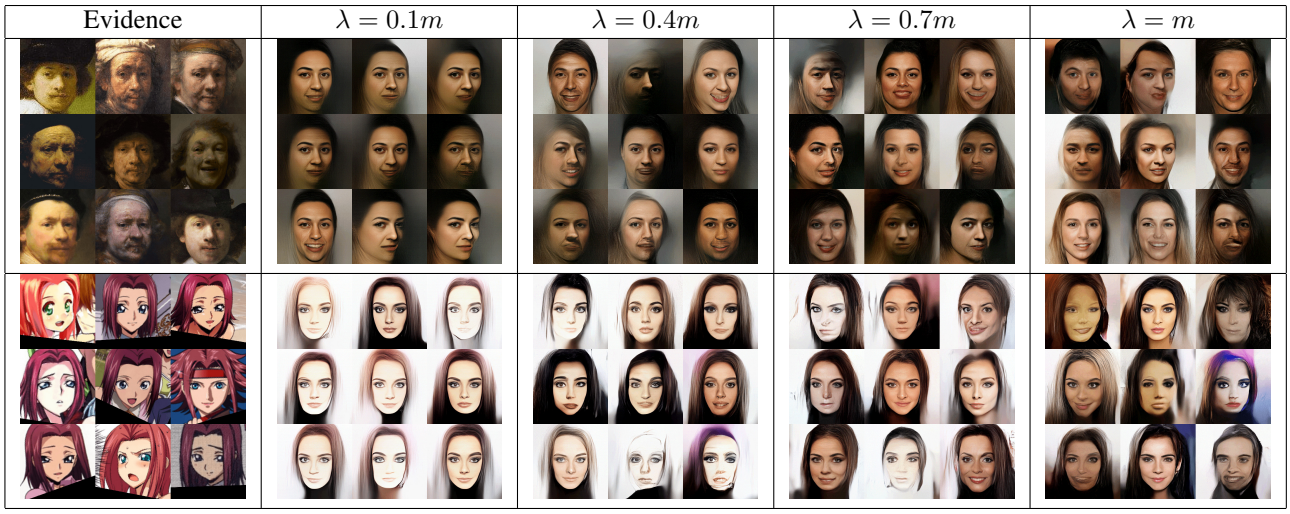


Figure 8: Varying the λ hyperparameter for different out-of-distribution conditioning datasets. λ that is too low creates samples too close to the sample mean, whereas λ that is too high creates samples too close to those from the original distribution.

Figure 9: Samples from the posterior of a CelebA flow model conditioned on MNIST, for m equal to 1, 5, and 30 respectively. For m equal to 1, the samples look very similar to the evidence. As m is increased, sample quality decreases due to the sample means becoming closer to $\vec{0}$ (and therefore becoming more "humanlike"), but prediction accuracy increases greatly.

6. Future Work

While flow models are the most natural choice to study invertible generative models, it is also possible, albeit more unwieldy, to find a mapping from data to latent vectors in other generative models. One such example is the BiGAN [5], which adds an extra term to the GAN objective in order to learn this mapping. As our methods are not specific to flow models in particular and only require the model to be invertible, it would also be possible to do posterior inference in the BiGAN latent space. As this latent space is several orders of magnitude smaller than the flow model latent space, it is very likely that posterior inference in a GAN's latent space would allow for a lower setting of the λ hyperparameter and more finely-grained results. For instance, as the sample mean would be much closer to that of a natural image than the sample mean under a flow model, perhaps it would be possible to give multiple images of a specific person as conditioning, and generate new images of that person. At the same time, perhaps the size of the flow model latent space is contributing to the richness of samples that we are able to generate, which is a question we would like to investigate in future work.

Training generative models on very multimodal datasets, such as videos complete with sound, is currently not feasible. If, however, it were, we could use partial data (such as only sound) as conditioning and then perform posterior inference in the latent space. In this scenario, the flow model would then possibly be able to generate plausible video that goes with the sound given as conditioning. Given our experiments with occluded faces, however, making this work may not be a trivial task.

7. Conclusions

We have introduced Transflow Learning, a simple method for doing transfer learning with invertible generative models. We demonstrated the capabilities of our algorithm on several generative modeling tasks, and even on downstream tasks such as handwritten digit classification.

We look forward to future research developments in invertible generative models, in particular developments in making flow models less difficult to train. Such developments would be a boon to the applicability of Transflow Learning, especially when being used for downstream tasks.

Acknowledgements

We would like to thank Alyosha Efros, Jonathan Ho, and Jay Whang for comments on an early version of this manuscript. This work was supported by the ERC grant ERC-2012-AdG 321162-HELIOS, EPSRC grant Seebibyte EP/M013774/1 and EPSRC/MURI grant EP/N019474/1. We would also like to acknowledge the Royal Academy of Engineering and FiveAI.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4432–4441, 2019.

[2] S. R. K. Branavan, David Silver, and Regina Barzilay. Learning to Win by Reading Manuals in a Monte-Carlo Framework. *Journal Of Artificial Intelligence Research*, 43:661–704, 2012.

[3] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[4] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[5] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial Feature Learning. *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[6] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A Neural Algorithm of Artistic Style. *arXiv preprint arXiv:1508.06576*, 2015.

[7] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[9] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design. In *International Conference on Machine Learning*, pages 2722–2730, 2019.

[10] Matt Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[11] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[12] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[13] Gregory Koch. *Siamese Neural Networks for One-Shot Image Recognition*. PhD thesis, University of Toronto, 2015.

[14] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

[15] LeCun Yann, Cortes Corinna, and Burges Christopher. THE MNIST DATABASE of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

[16] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-Shot Unsupervised Image-to-Image Translation. *arXiv preprint arXiv:1905.01723*, 2019.

[17] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Attributes in the Wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[18] Vikash K Mansinghka, Tejas D Kulkarni, Yura N Perov, and Josh Tenenbaum. Approximate bayesian image interpretation using generative probabilistic graphics programs. In *Advances in Neural Information Processing Systems*, pages 1520–1528, 2013.

[19] Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

[20] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[21] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

[22] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

[23] Robert Schlaifer and Howard Raiffa. *Applied statistical decision theory*. 1961.

[24] Pedro A Tsividis, Thomas Pouncy, Jacqueline L Xu, Joshua B Tenenbaum, and Samuel J Gershman. Human Learning in Atari. *2017 AAAI Spring Symposium Series*, 2017.

[25] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu.

Conditional Image Generation with PixelCNN Decoders. In *Advances in neural information processing systems*, pages 4790–4798, 2016.

[26] Richard David Wilkinson. Approximate bayesian computation (abc) gives exact results under the assumption of model error. *Statistical applications in genetics and molecular biology*, 12(2):129–141.

[27] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.