Reliable likelihoods from conditional flow matching generative models in feature space

Anonymous authors
Paper under double-blind review

Abstract

Normalising flows are a flexible class of generative models that provide exact likelihoods, and are often trained through maximum likelihood estimation. Recent work suggests that these models can assign undesirably high likelihood to out-of-distribution image data, questioning their reliability for applications where likelihoods are important (e.g. outlier detection). We show that continuous-time normalising flows trained with the conditional flow matching objective (CFM models) also provide unreliable likelihoods. Motivated by a hypothesis that unreliable likelihoods might be due to image-specific structures in the data, we investigate whether CFM models trained on various feature representations can lead to more reliable likelihoods. We evaluate CFM models trained on (1) the original data; (2) features from a pretrained classifier; (3) features from a pretrained perceptual autoencoder; and (4) features from an autoencoder trained with a simple pixel-based reconstruction loss. We show empirically that representations containing image-specific structure still lead to unreliable likelihoods from CFM models. Our proposed pixel autoencoder representations lead to reliable likelihoods from CFM models on out-of-distribution data, but can yield samples of lower quality, suggesting opportunities for future work.

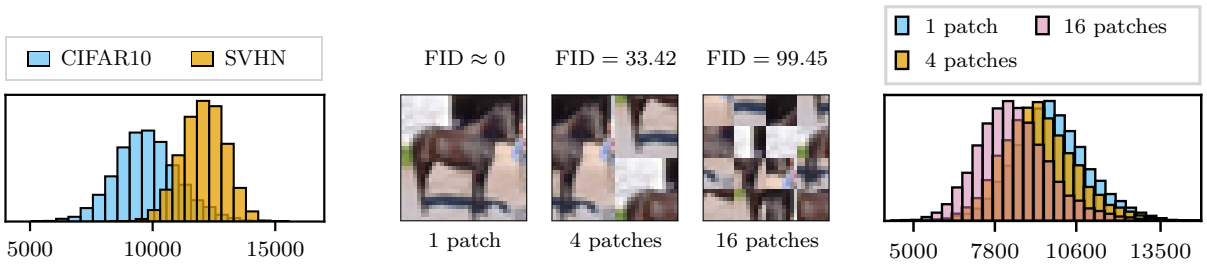
1 Introduction

Normalising flows are generative models that specify a target density through a base distribution and an invertible transformation process, with applications in computer vision (Kingma & Dhariwal, 2018; Dinh et al., 2017; Lipman et al., 2023; Kumar et al., 2020; Müller et al., 2019; Abdelhamed et al., 2019), audio generation (Esling et al., 2019; Kim et al., 2018; Prenger et al., 2019), graph generation (Madhawa et al., 2019), reinforcement learning (Mazoure et al., 2020; Ward et al., 2019; Touati et al., 2019) and physics (Kanwar et al., 2020; Köhler et al., 2019; Noé et al., 2019; Wirnsberger et al., 2020; Wong et al., 2020). They offer exact likelihood evaluation as an advantage over other generative models, enabling, in principle, outlier detection. Of interest to us is the peculiar phenomenon of normalising flows assigning undesirably high likelihoods to out-of-distribution data (Nalisnick et al., 2019a; Kirichenko et al., 2020; Voleti et al., 2024), bringing their reliability for applications into question.

A discrete-step normalising flow specifies a target distribution $p_x(\boldsymbol{x})$ in terms of an easy-to-sample-from base distribution $p_u(\boldsymbol{u})$, and an invertible transformation $\boldsymbol{u} = g(\boldsymbol{x})$ with $\boldsymbol{u} \sim p_u(\boldsymbol{u})$, by employing the change-of-variables formula. $g(\boldsymbol{x})$ is defined as a composite function, usually a neural network whose architecture is restricted for a tractable log-determinant in the change-of-variables formula. The continuous-time variant (Chen et al., 2018; Grathwohl et al., 2019), hereafter referred to as a continuous flow, expresses $\boldsymbol{u} = g(\boldsymbol{x})$ as the solution to an initial value problem (IVP):

$$\frac{d\boldsymbol{z}}{dt} = f_{\boldsymbol{\theta}}(\boldsymbol{z}, t), \qquad t \in [t_1, t_0], \qquad \boldsymbol{z}_0 = \boldsymbol{z}(t_0) = \boldsymbol{u}, \qquad \boldsymbol{z}_1 = \boldsymbol{z}(t_1) = \boldsymbol{x}, \tag{1}$$

and uses a continuous analog of the change-of-variables formula to determine $\log p_x(\boldsymbol{z}_1)$ (Chen et al., 2018). The function $f_{\boldsymbol{\theta}}(\boldsymbol{z}, t)$ defines a time-dependent vector field describing the transformation dynamics, with trainable parameters $\boldsymbol{\theta}$. This formulation circumvents restrictions on $g(\boldsymbol{x})$ for a tractable log-determinant, at the time-cost of simulating solution trajectories for the IVP in Equation 1.



(a) log likelihoods from a CFM model trained on CIFAR10

(b) random patch shuffles of in-distribution test images lead to the histograms of log likelihoods shown on the right

Figure 1: The histograms in (a) indicate that a CFM model trained on CIFAR10 assigns higher likelihood to out-of-distribution data from SVHN, compared to in-distribution CIFAR10 test data. The overlapping log likelihood histograms in (b) indicate that the same model assigns similar likelihoods to various levels of patch-shuffled CIFAR10 test images, despite FID scores that suggest changes in semantic content.

Continuous flows trained with the recently introduced conditional flow matching (Lipman et al., 2023; Tong et al., 2024) objective (CFM models) circumvent maximum likelihood training and the need for simulating solution trajectories. With the bottleneck of simulation removed, continuous flows become more relevant to applications at scale. But it turns out that CFM models also assign unreliable likelihoods to out-of-distribution data, as demonstrated in Figure 1a, where a model trained on the CIFAR10 dataset assigns higher likelihoods to samples from the SVHN dataset. Motivated by previous observations that unreliable likelihoods may stem from the structure of the data (Ren et al., 2019; Serrà et al., 2020; Kirichenko et al., 2020; Zhang et al., 2023), we will explore whether learned representations of the input data can lead to more reliable likelihoods.

The structure hypothesis can be illustrated by applying experiments from Voleti et al. (2024) to CFM models. In Figure 1b we show log likelihood histograms obtained from a CFM model trained on CIFAR10, for patch-shuffled versions of the CIFAR10 test set. The Fréchet inception distance (Heusel et al., 2017) of these datasets increases with the number of shuffled patches, indicating a change in semantic content, yet model likelihoods are affected to a much lesser degree. From this we suspect that CFM model likelihoods may depend on frequently occurring pixels, rather than semantic content.

For discrete-step normalising flows, Kirichenko et al. (2020) managed to improve the reliability of likelihoods by training on feature representations from a classifier pretrained on ImageNet. Such feature representations are convenient but not without limitation. Firstly, the pretrained network may struggle to generalise to data that differs in distribution from ImageNet. Secondly, while a flow model trained in feature space can be used to generate new feature vectors, the absence of a decoder makes them unsuitable for data generation. To address these limitations, we will train and evaluate CFM models on a number of representations: (1) EfficientNet-B4 features, similar to what Kirichenko et al. (2020) did for discrete-step flows; (2) features obtained from a pretrained perceptual autoencoder (Rombach et al., 2022); and (3) features obtained from a pixel autoencoder trained from scratch. Autoencoders provide an ability to decode from feature space back to image space, thereby enabling the generation of image data. Our findings can be summarised as follows:

1. EfficientNet-B4 features lead to CFM models with slightly more reliable likelihoods for out-of-distribution data, but do not resolve the problem entirely and prevent image generation.

2. CFM models trained on perceptual autoencoder features can generate good samples (in a qualitative sense), but do not improve the reliability of likelihoods. We believe this may relate to an observation that image-specific structure is preserved in the feature space of perceptual autoencoders.

3. Finally, a pixel autoencoder that obscures image-specific structure in the feature space offers a substantial improvement in the reliability of likelihoods from CFM models.

2 Related work

2.1 Unreliable likelihoods from normalising flows

Nalisnick et al. (2019a) were the first to observe that flow models can provide high likelihoods to out-of-distribution samples. They show this for discrete-step flow models trained on the FashionMNIST, CIFAR10, CelebA and ImageNet datasets. Moreover, they postulate that the unreliability of likelihoods is due to the structure of the data. For instance, SVHN data has a similar mean to CIFAR10 data, and lower variance, suggesting that the distribution underlying SVHN might be contained within the distribution underlying CIFAR10. Nalisnick et al. (2019b) further relate the unreliability of likelihoods to the mismatch between a model's typical set and its areas of high probability density. Here a model's typical set refers to the set of samples whose entropy is close to the true entropy of the density (Cover & Thomas, 2012). Nalisnick et al. (2019b) also provide a statistical test to determine whether inputs are in the typical set, thereby improving the reliability of anomaly detection. Other works also consider anomaly detection from the perspective of atypicality for both discrete (Høst-Madsen et al., 2019) and continuous data (Sabeti & Høst-Madsen, 2019), but do not consider image datasets and do not provide commentary specifically on flow models. We do not consider statistical tests for anomaly detection, but rather provide commentary on whether unmodified likelihoods from CFM models can be reliable.

Serrà et al. (2020) and Voleti et al. (2024) additionally make observations regarding the structure of the data (or rather, the complexity) and its relation to reliable likelihoods. Serrà et al. (2020) implement a measure of input complexity through a compression algorithm that can be used for a more reliable likelihood score. Voleti et al. (2024) implement multi-resolution in continuous-time normalising flows and show that they, too, exhibit unreliable likelihoods for out-of-distribution data. No solutions are provided for the continuous flows case, making our work the first to do so. Kirichenko et al. (2020) find reliable likelihoods for discrete-step normalising flows by changing the composition of the coupling layers (Dinh et al., 2015), thereby modifying the inductive biases of the flow model. By training the discrete flow model on features from a pretrained classifier, they additionally show that likelihoods become more reliable at the cost of data generation.

There are approaches that are model agnostic, or consider other types of generative models. Ren et al. (2019) observe unreliable likelihoods from autoregressive models, show that it can be attributed to background statistics, and propose a likelihood ratio method to correct for it. Choi et al. (2018) leverage the Watanabe-Akaike information criterion estimated from various generative models, including flow-based variants, to detect anomalous data. Song et al. (2019) leverage an observed difference between the training and evaluation modes of batch normalisation to identify out-of-distribution samples. More recently Zhang et al. (2023) investigate the KL divergence in flow-based models, towards an explanation of unreliable likelihoods. Ultimately, they still leverage local pixel dependencies of representations to perform anomaly detection, indicating the importance of structure.

Our work complements the existing literature on the reliability of out-of-distribution likelihoods by showing that CFM models (trained with the conditional flow matching objective rather than through maximum likelihood) exhibit the same out-of-distribution behaviour. It is worth highlighting the importance of this observation, given that CFM models are more scalable than simulation-based continuous flows. We observe that unreliable likelihoods seem to stem from the structure of the data, and can be mitigated through structural interventions (Ren et al., 2019; Serrà et al., 2020; Kirichenko et al., 2020; Zhang et al., 2023). Following a similar idea, we explore whether autoencoder representations of the input data (that maintain the ability to generate data) are viable solutions to the likelihood reliability problem in CFM models.

2.2 Representation learning

Autoencoders are a form of unsupervised representation learning, and learn feature representations from which the input can be reconstructed. Typically, the feature dimensionality is restricted in order to avoid learning an identity function between the inputs and reconstructions. These models are frequently used in self-supervised representation learning. For instance, a denoising autoencoder (Vincent et al., 2008; 2010) learns to provide clean reconstructions from noisy inputs. Vincent et al. (2008) argue that including robustness to partial destruction of the input in this way leads to learning the structure of the data manifold.

Reconstructing the input under perturbation has been influential in natural language processing too, where a network is tasked to predict masked tokens (Devlin et al., 2019). Masked training objectives can be applied to images, leading to improvements in classification models that are finetuned on labelled dataset (He et al., 2022; Dong et al., 2023). Representations from autoencoders are especially useful for us, since the decoder component provides an ability for data generation from sampled feature vectors.

Other representation learning approaches could be considered when data generation is not important. In the self-supervised literature, representations can be obtained through certain pretext tasks such as context prediction (Doersch et al., 2015), solving jig-saw puzzles (Noroozi & Favaro, 2016) or rotation prediction (Gidaris et al., 2018). There are also contrastive learning approaches where a network learns a representation such that similar inputs are embedded close together, and dissimilar inputs are further apart (Chen et al., 2015; Tian et al., 2020; Khosla et al., 2020).

Our work makes use of classifier- and autoencoder-based representations. We will show in particular that an autoencoder which suppresses image-specific structure in the learned feature space can lead to CFM models that assign reliable likelihoods to out-of-distribution data.

The use of autoencoders in generative modelling is not uncommon. Variational autoencoders (VAEs) (Kingma & Welling, 2013) and vector quantised VAEs (Van Den Oord et al., 2017), for example, are constructed in the autoencoder framework. There have also been various approaches that learn generative models in a feature space with reduced dimensionality. Vahdat et al. (2021) train score-based generative models (SGMs) on features obtained from a VAE, enabling the modelling of non-continuous data and learning of smoother SGMs in a reduced space. Rombach et al. (2022) train a diffusion model also on features from a VAE, and obtain improvements in inpainting and class-conditional image generation. We will consider their pretrained autoencoder, specifically to determine the influence of image-specific structure in the features on likelihood reliability. Dao et al. (2023) use the same autoencoder to learn CFM models in a feature space with reduced dimensionality, with a specific focus on computational efficiency. We will show that this kind of pretrained autoencoder does not assist with the reliability of likelihoods. Instead, the pixel autoencoder we propose has fewer parameters and leads to more reliable likelihoods.

3 Methodology

We are concerned with the problem of unsupervised density estimation. Given a training set $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$ with $\boldsymbol{x} \in \mathbb{R}^d$, and defining the initial condition $\boldsymbol{z}_1 = \psi(\boldsymbol{x})$ as the feature representations of the input data, we construct a continuous flow that computes the likelihood of $p(\boldsymbol{z}_1)$ as

$$\log p(\boldsymbol{z}_1) = \log p(\boldsymbol{z}_0) - \int_{t_1}^{t_0} \operatorname{Tr}\left[\frac{\partial f}{\partial \boldsymbol{z}}\right] dt, \tag{2}$$

where $f_{\boldsymbol{\theta}}(\boldsymbol{z}, t)$ is the dynamics function of a (neural) differential equation defining the transformation between the data and samples from the base distribution. Following Grathwohl et al. (2019), the transformed sample $\boldsymbol{u} = \boldsymbol{z}_0$ and $\log p(\boldsymbol{z}_1)$ are obtained by simultaneously solving Equations 1 and 2 in the torchdiffeq framework (Chen, 2018) for $t \in [t_1, t_0]$. Hutchinson's trace approximation is applied to the Jacobian term for computational efficiency. The conditional flow matching objective is a regression between $f_{\boldsymbol{\theta}}(\boldsymbol{z}, t)$ and a specified conditional vector field that generates probability paths (i.e. how the probability of a sample evolves through time). To train a CFM model, we must specify a parameterisation for the probablity path and the conditional vector field that generates it. Although more general probability paths exist (Tong et al., 2024), we restrict our focus to Gaussian conditional probability paths,

$$p_t(\boldsymbol{z} \mid \boldsymbol{z}_1) = \mathcal{N}(\boldsymbol{z} \mid \boldsymbol{\mu}(t), \sigma^2(t)\boldsymbol{I}), \tag{3}$$

where $\boldsymbol{\mu}(t)$ and $\sigma^2(t)$ describe how the mean and covariance change over time, and with $\boldsymbol{\mu}$ also dependent on $\boldsymbol{z}_1$. Such a probability path follows trajectories between a density concentrated around $\boldsymbol{z}_1$ and the base density, and is specified by a conditional vector field (Lipman et al. (2023), Theorem 3):

$$\boldsymbol{u}_t(\boldsymbol{z} \mid \boldsymbol{z}_1) = \frac{\sigma'(t)}{\sigma(t)} (\boldsymbol{z} - \boldsymbol{\mu}(t)) + \boldsymbol{\mu}'(t), \tag{4}$$

where the prime symbol indicates the derivative with respect to t. By defining $\boldsymbol{\mu}(t) = t\boldsymbol{z}_1$ and $\sigma(t) = 1 - (1 - \sigma_{\min})t$, the target conditional vector field leading to the standard base becomes

$$\boldsymbol{u}_t\left(\boldsymbol{z} \mid \boldsymbol{z}_1\right) = \frac{\boldsymbol{z}_1 - (1 - \sigma_{\min})\boldsymbol{z}}{1 - (1 - \sigma_{\min})t}. \tag{5}$$

The conditional vector field $\boldsymbol{u}_t\left(\boldsymbol{z} \mid \boldsymbol{z}_1\right)$, with $\sigma_{\min}$ set sufficiently small, leads to the conditional flow matching objective (Lipman et al., 2023):

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} ||f_{\boldsymbol{\theta}}(\boldsymbol{z}, t) - \boldsymbol{u}_t(\boldsymbol{z} \mid \boldsymbol{z}_1)||^2, \tag{6}$$

averaged over N samples in a mini-batch, and with probability paths defined over $t \sim \mathcal{U}(0, 1)$. After training, we may compute likelihoods by first obtaining the feature representation $\boldsymbol{z}_1 = \psi(\boldsymbol{x})$, and then solving Equation 2. New samples of images can be generated by the model, if the inverse ψ^{-1} can be evaluated. To do so, we first sample $\boldsymbol{z}_0$ from the base distribution of the trained CFM model, and solve the differential equation in the reverse direction to obtain $\boldsymbol{z}_1$ (i.e. from time t_0 to t_1). The corresponding image sample can then be obtained by computing $\boldsymbol{x} = \psi^{-1}(\boldsymbol{z}_1)$.

Our goal is to determine whether there are parameterisations of $\psi(\boldsymbol{x})$ that may lead to reliable likelihoods. We consider three variants, as described in the sections below. For experimental evaluation we consider the MNIST, FashionMNIST, CIFAR10 and SVHN datasets.

3.1 Features from EfficientNet-B4

As a starting point, we consider the EfficientNet-B4 network pretrained on ImageNet. 1792-dimensional features are extracted from one of the fully connected layers of this model. We thereby reproduce the experimental procedure of Kirichenko et al. (2020), but for continuous flows.

To investigate the impact of presenting this pretrained network with data outside of its training data distribution, we trained a four-class LDA classifier on the EfficientNet-B4 features obtained from the MNIST, FashionMNIST, CIFAR10, and SVHN datasets, where we treat these datasets as the four classes, and achieved a test classification accuracy of 99%. This provides some evidence that the features from EfficientNet-B4 do differentiate at the dataset level, and suggests that a model learning the density over one of these datasets, in the EfficientNet-B4 feature space, may discern feature vectors from the other datasets as out-of-distribution. We also show in Figure 2 that there is a sense of separation between the datasets, even for a 2-dimensional LDA projection of the 1792-dimensional feature vectors.

We note that the conversion from an image to this feature representation is not invertible, preventing the generation of image samples from a CFM model trained in this feature space.

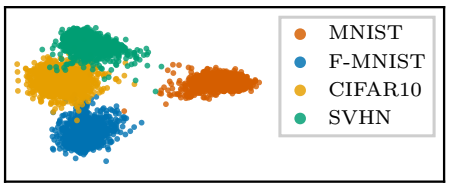


Figure 2: Two-dimensional LDA projections of the EfficientNet-B4 feature representations of training samples from MNIST, FashionMNIST (F-MNIST), CIFAR10 and SVHN.

3.2 Features from a perceptual autoencoder

To retain the ability to generate image samples, we also consider feature vectors obtained from a pretrained autoencoder (Rombach et al., 2022). This autoencoder was trained using a perceptual distance loss (Zhang et al., 2018) that measures the perceptual similarity between two images, and an adversarial objective (Larsen

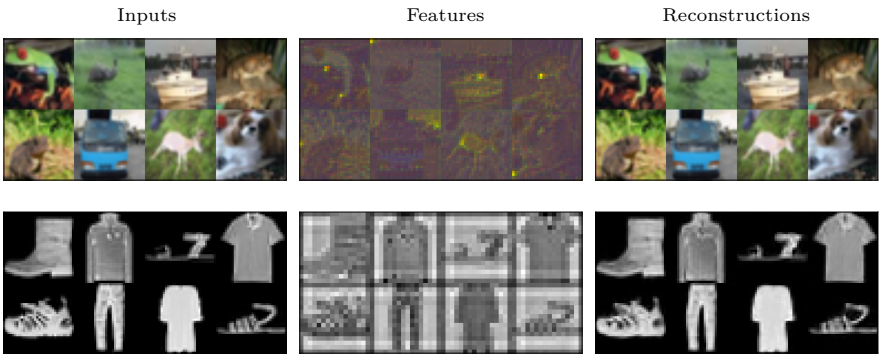


Figure 3: Top row: images from CIFAR10, gamma-corrected feature representations from the pretrained perceptual autoencoder, and decoded reconstructions. Bottom row: images from FashionMNIST, feature vectors from a trained perceptual autoencoder in which the feature dimensionality is smaller than the input data, and decoded reconstructions. We observe that image-specific structure persists in this feature space.

et al., 2016; Isola et al., 2017; Dosovitskiy & Brox, 2016; Esser et al., 2021; Yu et al., 2022) based on image patches (Isola et al., 2017). This combination of objectives is motivated by evidence that an objective based only on pixel reconstruction is inadequate in terms of decoding high-quality images (Wang & Bovik, 2009; Larsen et al., 2016). For instance, Wang & Bovik (2009) show with an example that various image distortions can lead to the same mean squared error when compared to the original image.

The original purpose of this autoencoder was perceptual image compression for computational efficiency in diffusion-based generative models. We instead focus on whether an autoencoder primed for high-quality image reconstruction can assist with reliable likelihoods from CFM models, and provide a means of generating new images of high quality.

We note that feature representations from this perceptual autoencoder have a shape of (28, 28, 4), and is larger than the original image dimensions of our datasets which are either (28, 28, 1) or (32, 32, 3). The pixel autoencoder we describe in Section 3.3 will reduce the input dimensionality.

Figure 3 (top) shows the feature representations and reconstructions of randomly chosen CIFAR10 samples, obtained from the perceptual autoencoder. Examples from the other datasets can be found in Appendix A. It is apparent that the reconstructions for these datasets, on which the autoencoder was not trained, are of high quality. It seems that the autoencoder has learned a general purpose feature representation. That image-specific structure is preserved in the feature representations is striking, and reinforces that the main purpose of this autoencoder is perceptual image compression. A similar observation followed after we trained the autoencoder with a feature dimensionality of (16, 16, 1), as illustrated in the bottom row of Figure 3. We suspect that this might also be due to the fully convolutional nature of the autoencoder, as we found that a fully convolutional autoencoder trained with a pixel-based reconstruction loss also preserves image-specific structure in the feature space.

3.3 Features from a pixel autoencoder

As an alternative to the perceptual autoencoder, which seems to preserve image-specific structure in its features, we consider a simpler autoencoder with a pixel-based, mean squared error reconstruction objective. This autoencoder reduces the dimensionality of the data, which will assist with computational efficiency. It has the advantage of a significantly reduced model capacity compared to the perceptual autoencoder, which can act as a form of regularisation and lead to faster training. Despite its reduced capacity, the autoencoder can offer sharp reconstructions under suitable hyperparameters.

The encoder module consists of 5 convolutional layers, 3 of which are strided, followed by a fully connected layer that controls the dimensionality of the feature space and assists with suppressing image-specific structure in the encoded features. The decoder module consists of a fully connected layer, followed by 5 convolutional layers that attempt to reconstruct the input to the encoder. The autoencoder is trained for

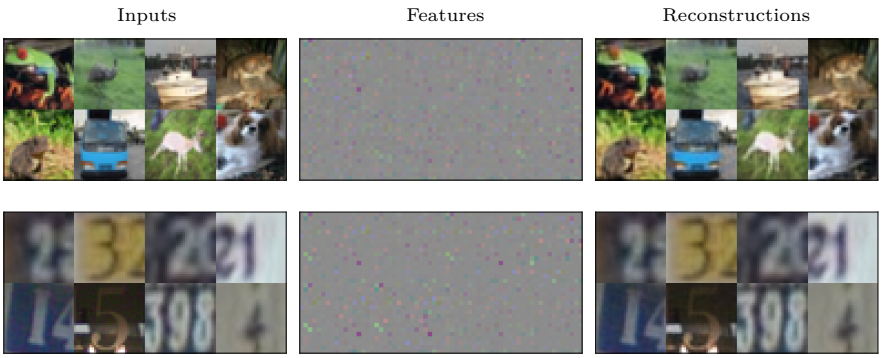


Figure 4: Input images (left), reshaped feature representations (middle), and reconstructions (right) from the pixel autoencoder trained on the CIFAR10 dataset. It seems that image-specific structure is no longer preserved in the features, and that the decoder produces good reconstructions even when presented with inputs from the SVHN dataset (bottom).

500 epochs, with the Adam optimiser (Kingma & Ba, 2015) and a learning rate scheduler that reduces the learning rate from 0.001 when the validation loss plateaus.

Figure 4 shows example feature representations and reconstructions for images from the CIFAR10 and SVHN datasets, obtained from a pixel autoencoder trained on CIFAR10. For display purposes, we reshaped the 768-dimensional feature vectors to images of size (16, 16, 3). We note that image-specific structure does not seem to be preserved in the features. Reshaping the features in this way also allows for a U-Net parameterisation of the vector field (Ho et al., 2020; Lipman et al., 2023), which seems to significantly impact the convergence of CFM model training. We further note that the quality of the reconstructions is high, even for images from datasets other than the autoencoder's training set, indicating that the model has learned a general purpose feature representation. Examples of feature representations and reconstructions for images from the MNIST and FashionMNIST datasets can be found in Appendix B.

3.4 CFM model parameterisation

We train CFM models on each of the respective feature representations presented above, and will compare their performance against baseline CFM models trained in the original image space. Table 1 provides a summary.

The vector field for the CFM model trained on EfficientNet-B4 (ENet) features is parameterised by a fully connected network with skip connections. We experimented with a sinusoidal time embedding and also a hypernetwork for time dependence, but found that adding time as an additional input dimension worked better. The vector field for the baseline, perceptual autoencoder (PercAE) features, and pixel autoencoder (PixAE) features is parameterised using the same time-dependent U-Net with attention, modified according to the input feature dimensionality. Further implementation details are included in Appendix C.

Table 1: Summary of the respective feature representations on which we train CFM models. Greyscale and colour inputs are denoted by (g) and (rgb). The two autoencoders differ in how they capture local pixel correlations: the perceptual autoencoder preserves image-specific structure, while the pixel autoencoder does not. The last column indicates whether or not the trained CFM model can be used to generate images.

Feature space	Description	Pretrained	Feature vector length	Image gen.
Baseline	Original image space	✗	784 (g), 3072 (rgb)	✓
ENet	Features from an EfficientNet-B4 classifier	✓	1792	✗
PercAE	Autoencoder trained with a perceptual loss	✓	3136	✓
PixAE	Autoencoder trained with a pixel-based loss	✗	256 (g), 768 (rgb)	✓

3.5 Evaluation metrics

Likelihood metric. Bits-per-dimension, derived from the average log likelihood, is a common metric to measure generalisation in likelihood-based generative models of discrete image data (Papamakarios et al., 2017). Bits-per-dimension is not applicable to our (continuous) representations of the data, as it would require infinite bits to encode continuous data under the model. Instead we use a signed version of the Bhattacharya distance for out-of-distribution analysis. Given two Gaussian densities $h_1 = \mathcal{N}\left(\mu_1, \sigma_1^2\right)$ and $h_2 = \mathcal{N}\left(\mu_2, \sigma_2^2\right)$, and with $\mu_d = \mu_2 - \mu_1$, the signed distance is defined as

$$D_{\mathrm{SB}}\left(h_1, h_2\right) = \operatorname{sign}(\mu_d)\left[\frac{1}{4}\frac{\left(\mu_2-\mu_1\right)^2}{\sigma_2^2+\sigma_1^2} + \frac{1}{2}\log\left(\frac{\sigma_2^2+\sigma_1^2}{2\sigma_2\sigma_1}\right)\right], \tag{7}$$

where we set $\operatorname{sign}(0) = 1$ to avoid a distance of zero for equal means. $D_{\mathrm{SB}}\left(h_1, h_2\right)$ evaluates to a high value when the means of the h_1 and h_2 are far apart, and their standard deviations are low, thereby indicating less overlap between the densities. $D_{\mathrm{SB}}\left(h_1, h_2\right)$ is positive when $\mu_2 \geq \mu_1$, indicating that h_2 is shifted rightwards from h_1, and it is negative when $\mu_2 < \mu_1$. The signed Bhattacharya distance between two Gaussians with equal parameters is 0. In our evaluations, a distance will be obtained by fitting Gaussian densities to in- and out-of-distribution likelihood histograms, and measuring the magnitude and direction of their overlap. A large magnitude indicates that there is little overlap between the two densities. A positive sign indicates that in-distribution log likelihoods are, on average, higher than out-of-distribution log likelihoods. Large positive distances are therefore indicative of reliable likelihoods.

Sample quality. To evaluate sample quality from a trained CFM model, we will make use of the Fréchet inception distance (FID) (Heusel et al., 2017) between the training set and 50K generated samples. For the CFM models trained on ENet features, the LDA classifier from before is used to inspect samples of generated feature vectors. These generated feature vectors cannot be converted to images, and the classifier scores do not describe sample quality. We consider the LDA classifier merely to verify sensible output from the model.

Quantitative metrics are calculated over multiple runs, and specific hyperparameters and training configurations are provided in Appendix C. We separate our results into three sections, for the three feature spaces, and compare each with a baseline CFM model trained on the original images. Code for reproducing the results will be made available upon acceptance of this paper.

4 Results

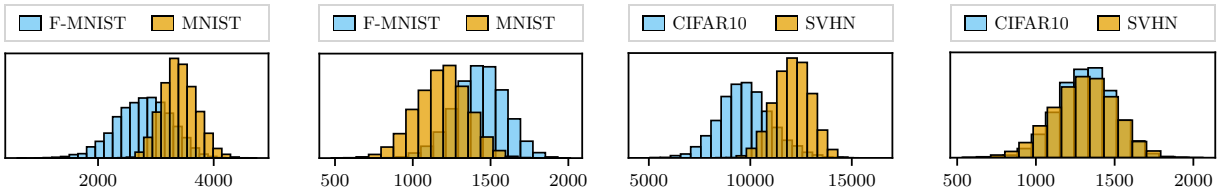
4.1 CFM models trained on ENet features

Table 2 reports the LDA classifier accuracy on feature vector samples generated by CFM models trained on ENet features of MNIST, FashionMNIST, CIFAR10, and SVHN, respectively. This provides some verification that the four trained CFM models can generate feature vectors that are close to their respective training sets. We suspect that the accuracy can be increased through further hyperparameter tuning.

Table 2: LDA classifier accuracy on samples generated by seperate CFM models trained on ENet features of MNIST, FashionMNIST, CIFAR10 and SVHN.

	MNIST	FashionMNIST	CIFAR10	SVHN
LDA accuracy	0.8419	0.8452	0.8586	0.7660

Figures 5a and 5b show log likelihood histograms from baseline and ENet models trained on FashionMNIST. The CFM model trained on ENet features assigns slightly lower likelihoods on average to out-of-distribution data, compared to the baseline. There is still considerable overlap in the histograms of in- and out-of-distribution likelihoods, and room for improvement. Figures 5c and 5d show similar (though somewhat worse) behaviour for a CFM model trained on ENet features of CIFAR10. The impact on likelihood reliability



(a) F-MNIST baseline (b) F-MNIST ENet (c) CIFAR10 baseline (d) CIFAR10 ENet

Figure 5: Log likelihood histograms from baseline CFM models trained on original images, and from CFM models trained on **ENet features** of FashionMNIST and CIFAR10. Blue corresponds to in-distribution test data, and orange to out-of-distribution data.

that we observe here, by training CFM models on ENet feature representations, is not as significant for the CIFAR10 dataset as in the reported results for discrete-step flow models. This may be due to the continuous formulation of our models, or due to the parameterisation of the vector field. Where we use a fully connected network with skip connections, Kirichenko et al. (2020) incorporate their own st-network for the discrete flow. We tried other methods of incorporating time dependence, such as a hypernetwork (Ha et al., 2017), but did not find any improvements.

Table 3 lists the signed Bhattacharya distances between various in- and out-of-distribution histogram distributions. A positive value indicates that the mean of in-distribution likelihoods is higher than the mean of out-of-distribution likelihoods. Models trained on the MNIST and SVHN datasets provide reliable likelihoods, as implied by the positive distances in Table 3. The results in the table also provide quantitative evidence for the improvements seen in Figure 5. We conclude that a CFM model can benefit from training on EfficientNet-B4 feature representations, corroborating what has been shown for discrete flow models. However, since the conversion of images to this kind of feature representation is not invertible, the resulting CFM models cannot be used to generate new samples.

Table 3: Signed Bhattacharya distances between likelihoods of in- and out-of-distribution data, from CFM models trained on original images (the baselines) and **ENet features**, with out-of-distribution sets as shown. Means and standard deviations are measured over multiple training runs.

CFMs trained on MNIST

	Baseline	ENet features
FashionMNIST	2.13 ± 0.06	3.55 ± 0.43

CFMs trained on CIFAR10

	Baseline	ENet features
SVHN	-1.08 ± 0.00	-0.30 ± 0.40

CFMs trained on FashionMNIST

	Baseline	ENet features
MNIST	-0.85 ± 0.01	0.74 ± 0.02

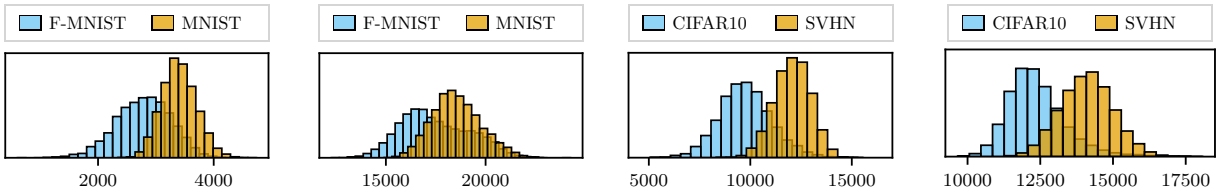
CFMs trained on SVHN

	Baseline	ENet features
CIFAR10	1.72 ± 0.02	2.49 ± 0.51

4.2 CFM models trained on PercAE features

Figures 6a and 6b show log likelihood histograms from baseline and PercAE models trained on FashionMNIST. We note that both models assign higher likelihoods to out-of-distribution data from MNIST, indicating that the perceptual autoencoder does not lead to reliable likelihoods. The same is seen in Figures 6c and 6d, where both models again assign higher likelihoods to out-of-distribution data when trained on CIFAR10. Models trained on the MNIST and SVHN datasets present trends similar to those in Table 3. That is, they correctly assign lower likelihoods to FashionMNIST when trained on MNIST, and to CIFAR10 when trained on SVHN.

ENet features seem to lead to more reliable likelihoods over the baseline, compared to PercAE features. We are doubtful that the increased input dimensionality for models trained on PercAE features are the



(a) F-MNIST baseline (b) F-MNIST PercAE (c) CIFAR10 baseline (d) CIFAR10 PercAE

Figure 6: Log likelihood histograms from baseline CFM models trained on original images, and from CFM models trained on **PercAE features** of FashionMNIST and CIFAR10.

cause of unreliable likelihoods, as ENet features of FashionMNIST were also of higher dimensionality but did lead to more reliable likelihoods. It may instead be due to the class separation we observed in the case of ENet features, and the fact that image-specific structure is preserved in the feature space of the perceptual autoencoder (as demonstrated in Figure 3). If unreliable likelihoods result from the structure of the data, we hypothesise that feature representations which preserve image-specific structure may not resolve the issue.

Figure 7 shows generated features and decoded images from CFM models trained on the PercAE features of FashionMNIST and CIFAR10 which, qualitatively, appear to be quite good.

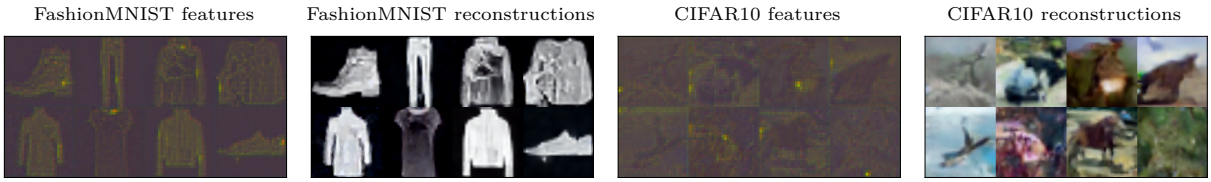


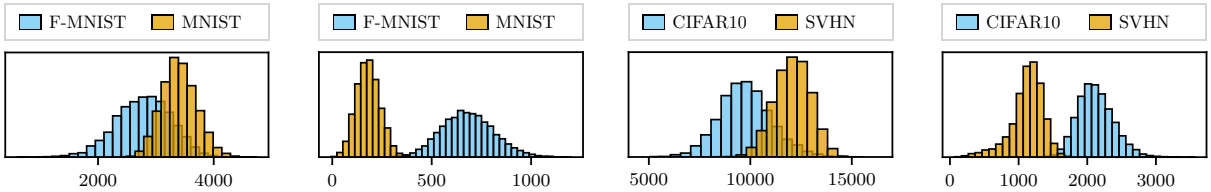
Figure 7: Examples of feature vectors generated by CFM models trained on **PercAE features** of FashionMNIST (left) and CIFAR10 (right), and the decoded reconstructions. We show gamma-corrected versions of the first three (out of four) channels of the feature vectors.

4.3 CFM models trained on PixAE features

Figure 8 shows log likelihood histograms from baseline and PixAE models trained on FashionMNIST and CIFAR10. As before, the baseline CFM models assign higher likelihoods to out-of-distribution data. However, the models trained on PixAE features provide clearly separated likelihoods for in- and out-of-distribution data, with lower likelihoods for the latter. We believe that this may be due to the fact that the pixel autoencoder suppresses image-specific structure, in line with observations from Serrà et al. (2020). This result from training CFM models on PixAE features is a clear improvement over models trained in the other feature spaces considered.

Table 4 shows the signed Bhattacharya distances between in- and out-of-distribution likelihoods, from CFM models trained on PixAE features, again compared to baseline models. The large positive distances for the CFM models trained on PixAE features, across all datasets, again indicate a significant improvement in likelihood reliability, compared to previous approaches. The pixel autoencoder also contains fewer parameters than the alternatives we considered.

We show qualitatively in Figure 9 that features generated from our trained CFM models can successfully be decoded into images, but note that samples from the CIFAR10 and SVHN models are of low quality. There seems to be a trade-off between feature spaces that lead to reliable likelihoods and feature spaces that lead to high-quality samples: PercAE features lead to good samples but unreliable likelihoods, while PixAE features lead to reliable likelihoods but poor samples.



(a) F-MNIST baseline (b) F-MNIST PixAE (c) CIFAR10 baseline (d) CIFAR10 PixAE

Figure 8: Log likelihood histograms from baseline CFM models trained on original images, and from CFM models trained on **PixAE features** of FashionMNIST and CIFAR10.

Table 4: Signed Bhattacharya distances between likelihoods of in- and out-of-distribution data, from CFM models trained on original images (the baselines) and **PixAE features**, with out-of-distribution sets as shown. Means and standard deviations are measured over multiple training runs.

CFMs trained on MNIST

	Baseline	PixAE features
FashionMNIST	2.13 ± 0.06	5.60 ± 0.13

CFMs trained on CIFAR10

	Baseline	PixAE features
SVHN	−1.08 ± 0.00	2.67 ± 0.08

CFMs trained on FashionMNIST

	Baseline	PixAE features
MNIST	−0.85 ± 0.01	3.90 ± 0.19

CFMs trained on SVHN

	Baseline	PixAE features
CIFAR10	1.72 ± 0.02	4.41 ± 0.10

Table 5 confirms our qualitative observations, where we see that FID scores for samples from the PixAE models are relatively high compared to other approaches. For example, Lipman et al. (2023) report FID scores of around 8.0 for generative models trained on CIFAR10. We experimented with training the CFM models for longer, but it did not improve sample quality. The problem persisted for greyscale versions of the CIFAR10 and SVHN datasets, indicating that the complexity through colour channels is not the main cause for the lower quality in generated samples. We also experimented with training CFM models on features obtained from a fully convolutional autoencoder, which yielded samples of higher quality but unreliable likelihoods. Image-specific structure was again recognisable in the feature space of this autoencoder. It is possible that some structure is important for high quality samples, offering avenues for future work.

Table 5: Fréchet inception distances of generated samples from CFM models trained on **PixAE features** of the respective datasets. Means and standard deviations are measured over multiple training runs.

	MNIST	FashionMNIST	CIFAR10	SVHN
FID	31.48 ± 1.41	58.02 ± 4.08	190.30 ± 5.90	104.82 ± 9.12

5 Conclusion

We explored the phenomenon of normalising flow models assigning undesirably high likelihood to out-of-distribution data, specifically in the context of continuous-time flow models trained under the conditional flow matching objective (CFM models), and whether training them in feature space can improve matters. To that end, we considered three different image feature representations. Our results suggest that feature spaces preserving image-specific structure do not solve the problem of unreliable likelihoods, whereas feature representations from more intricate encodings can lead to much more reliable in- and out-of-distribution likelihoods from CFM models.

Our proposed pixel autoencoder features overcome the limitation of features from a pretrained classifier (like EfficientNet-B4) by maintaining the ability to generate image samples. However, the quality of generated

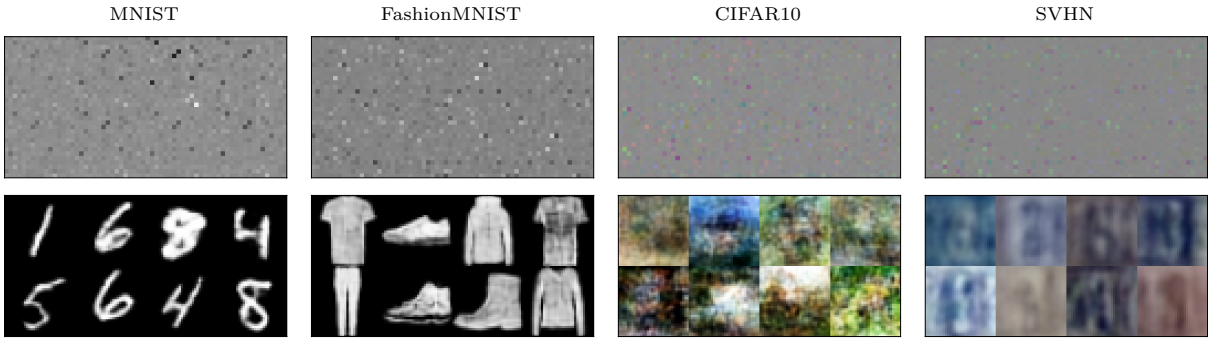


Figure 9: Generated feature vectors (top) from CFM models trained on **PixAE features** of the respective datasets, and the corresponding reconstructions (bottom).

samples can be somewhat worse compared to an autoencoder trained with a perceptual loss (which in turn does not solve the likelihood reliability problem). We hypothesise that sample quality from the pixel autoencoder for datasets like CIFAR10 and SVHN can be improved through additional structural biases in the feature space, possibly obtained from class labels. The models should also be tested on additional datasets, e.g. CIFAR100, CelebA and ImageNet, once image generation capabilities are improved.

In conclusion, our exploration into the use of feature representations for CFM model training contributes to the discourse on reliable likelihoods from scalable continuous flow models.

Broader impact statement

Unreliable likelihoods from flow-based generative models may foster a false sense of confidence in their predictions. This mistaken equivalence between high confidence and model accuracy can result in poor decision making in critical or high-risk fields such as healthcare (e.g. diagnosing medical conditions), finance (e.g. approving loans), and security (e.g. detecting fraud). Asserting the reliability of generative models can accelerate their adoption but also heighten the risk of misuse. Therefore, we advocate for continued research into the reliability of flow-based generative models.

References

Abdelrahman Abdelhamed, Marcus A. Brubaker, and Michael S. Brown. Noise flow: Noise modeling with conditional normalizing flows. *International Conference on Computer Vision*, 2019.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *International Conference on Learning Representations*, 2015.

Ricky T. Q. Chen. torchdiffeq: PyTorch implementation of differentiable ODE solvers, 2018. URL https://github.com/rtqichen/torchdiffeq.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 2018.

Hyunsun Choi, Eric Jang, and Alexander A. Alemi. WAIC, but why? Generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *International Conference on Learning Representations, Workshop Track*, 2015.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *International Conference on Learning Representations*, 2017.

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *International Conference on Computer Vision*, 2015.

Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, Nenghai Yu, and Baining Guo. PeCo: Perceptual codebook for BERT pre-training of vision transformers. *AAAI Conference on Artificial Intelligence*, 2023.

Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *Advances in Neural Information Processing Systems*, 2016.

Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, and Axel Chemla-Romeu-Santos. Universal audio synthesizer control with normalizing flows. *arXiv preprint arXiv:1907.00971*, 2019.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. *Conference on Computer Vision and Pattern Recognition*, 2021.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *International Conference on Learning Representations*, 2018.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*, 2019.

David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. *International Conference on Learning Representations*, 2017.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *Conference on Computer Vision and Pattern Recognition*, 2022.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in Neural Information Processing Systems*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 2020.

Anders Høst-Madsen, Elyas Sabeti, and Chad Walton. Data discovery and anomaly detection using atypicality: Theory. *IEEE Transactions on Information Theory*, 65(9):5302–5322, 2019.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *Conference on Computer Vision and Pattern Recognition*, 2017.

Gurtej Kanwar, Michael S. Albergo, Denis Boyda, Kyle Cranmer, Daniel C. Hackett, Sébastien Racaniere, Danilo Jimenez Rezende, and Phiala E. Shanahan. Equivariant flow-based sampling for lattice gauge theory. *Physical Review Letters*, 125(12):121601, 2020.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 2020.

Sungwon Kim, Sang-gil Lee, Jongyoon Song, Jaehyeon Kim, and Sungroh Yoon. FloWaveNet: A generative flow for raw audio. *International Conference on Machine Learning,*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in Neural Information Processing Systems*, 2018.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Polina Kirichenko, Pavel Izmailov, and Andrew G. Wilson. Why normalizing flows fail to detect out-of-distribution data. *Advances in Neural Information Processing Systems*, 2020.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: Sampling configurations for multi-body systems with symmetric energies. *Machine Learning and the Physical Sciences Workshop at NeurIPS*, 2019.

Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. VideoFlow: A conditional flow-based model for stochastic video generation. *International Conference on Learning Representations*, 2020.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *International Conference on Machine Learning*, 2016.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. *International Conference on Learning Representations*, 2023.

Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. GraphNVP: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

Bogdan Mazoure, Thang Doan, Audrey Durand, Joelle Pineau, and R. Devon Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. *Conference on Robot Learning*, 2020.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Transactions on Graphics*, 38(5):1–19, 2019.

Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *International Conference on Learning Representations*, 2019a.

Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using typicality. *Workshop on Bayesian Deep Learning at NeurIPS*, 2019b.

Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457):11–47, 2019.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. *European Conference on Computer Vision*, 2016.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems*, 2017.

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. WaveGlow: A flow-based generative network for speech synthesis. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. *Advances in Neural Information Processing Systems*, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *Conference on Computer Vision and Pattern Recognition*, 2022.

Elyas Sabeti and Anders Høst-Madsen. Data discovery and anomaly detection using atypicality for real-valued data. *Entropy*, 21(3):219, 2019.

Joan Serrà, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F. Núñez, and Jordi Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. *International Conference on Learning Representations*, 2020.

Jiaming Song, Yang Song, and Stefano Ermon. Unsupervised out-of-distribution detection with batch normalization. *arXiv preprint arXiv:1910.09115*, 2019.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *European Conference on Computer Vision*, 2020.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.

Ahmed Touati, Harsh Satija, Joshua Romoff, Joelle Pineau, and Pascal Vincent. Randomized value functions via multiplicative normalizing flows. *Conference on Uncertainty in Artificial Intelligence*, 2019.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 2021.

Aaron Van Den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 2017.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. *International Conference on Machine Learning*, 2008.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(12), 2010.

Vikram Voleti, Chris Finlay, Adam Oberman, and Christopher Pal. Multi-resolution continuous normalizing flows. *Annals of Mathematics and Artificial Intelligence*, 2024.

Zhou Wang and Alan C. Bovik. Mean squared error: Love it or leave it? A new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009.

Patrick Nadeem Ward, Ariella Smofsky, and Avishek Joey Bose. Improving exploration in soft-actor-critic with normalizing flows policies. *Invertible Neural Networks and Normalizing Flows Worskhop at ICML*, 2019.

Peter Wirnsberger, Andrew J Ballard, George Papamakarios, Stuart Abercrombie, Sébastien Racanière, Alexander Pritzel, Danilo Jimenez Rezende, and Charles Blundell. Targeted free energy estimation via learned mappings. *The Journal of Chemical Physics*, 153(14), 2020.

Kaze WK Wong, Gabriella Contardo, and Shirley Ho. Gravitational-wave population inference with deep flow-based generative network. *Physical Review D*, 101(12):123005, 2020.

Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. *International Conference on Learning Representations*, 2022.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *Conference on Computer Vision and Pattern Recognition*, 2018.

Yufeng Zhang, Jialu Pan, Wanwei Liu, Zhenbang Chen, Kenli Li, Ji Wang, Zhiming Liu, and Hongmei Wei. Kullback-leibler divergence-based out-of-distribution detection with flow-based generative models. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

A Features and reconstructions from the perceptual autoencoder

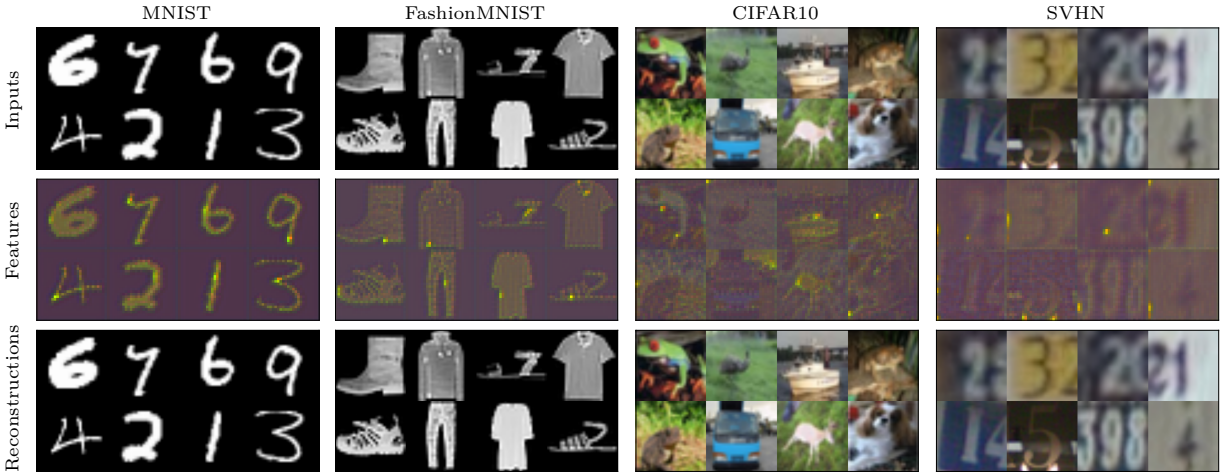


Figure 10: Example images, gamma-corrected feature representations and reconstructions from the pre-trained perceptual autoencoder described in Section 3.2, for the MNIST, FashionMNIST, CIFAR10 and SVHN datasets. Image-specific structures are evident in the features.

B Features and reconstructions from the pixel autoencoder

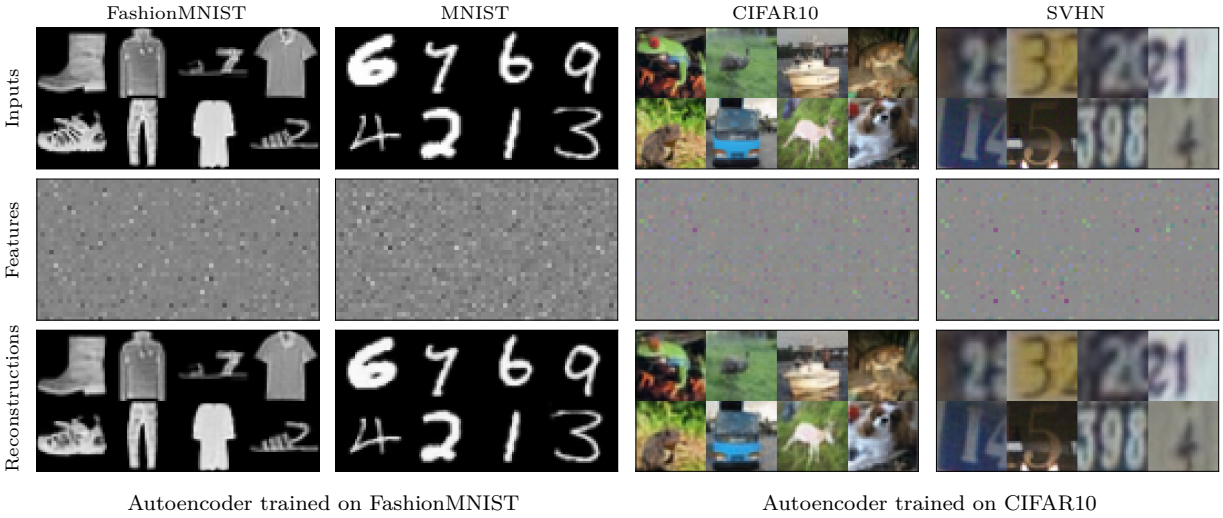


Figure 11: Example images, reshaped feature vectors and reconstructions from two versions of the pixel autoencoder described in Section 3.3. Image-specific structure is no longer preserved in feature space, and the decoder provides good reconstructions even when the autoencoder is presented with input images from a dataset different to its training set.

C Implementation details

Log likelihoods and generated samples from all models are computed using the torchdiffeq framework (Chen, 2018) in PyTorch. All models are trained on a single NVIDIA RTX A6000 GPU. The Adam optimiser (Kingma & Ba, 2015) is used with default values for β_1 and β_2, and its learning rate is warmed up with a linear scheduler. Hyperparameter tuning was performed on the learning rate and number of epochs, to the limits of our compute budget.

The implementation from Tong et al. (2024) is adapted for CFM models trained on the original images, on features from the perceptual autoencoder, and on features from the pixel autoencoder. We refer the reader to the original implementation of Tong et al. (2024) for descriptions of the various hyperparameters. Final hyperparameter values for each CFM model are provided in the following sections.

C.1 CFM models trained on the original images

The vector field for the baseline CFM models trained on original images uses the hyperparameters shown in Table 6.

Table 6: Hyperparameters for the baseline CFM models trained on original image data.

Parameter	MNIST	FashionMNIST	CIFAR10	SVHN
Channels	128	128	128	128
Channels multiple	(1, 2, 2)	(1, 2, 2)	(1, 2, 2, 2)	(1, 2, 2, 2)
Heads	1	1	1	1
Heads channels	1	1	1	1
Attention resolution	16	16	16	16
Dropout	0.0	0.0	0.0	0.0
Batch size	128	256	256	256
Epochs	150	150	150	150
Learning rate (warmed up)	0.0002	0.0002	0.0002	0.0002

C.2 CFM models trained on EfficientNet-B4 features

The vector field for CFM models trained on EfficientNet-B4 features is parameterised by a fully connected neural network with skip connections. We append time as an input, use 5 hidden layers, and skip connections between hidden layers 1 and 5 and hidden layers 2 and 4. The Swish activation function is used (Ramachandran et al., 2017). Table 7 shows additional hyperparameters for these CFM models.

Table 7: Hyperparameters for CFM models trained on features obtained from EfficientNet-B4.

Parameter	MNIST	FashionMNIST	CIFAR10	SVHN
Number of hidden layers	5	5	5	5
Layer width	1610	1610	1610	1610
Batch size	1024	1024	1024	1024
Epochs	300	300	300	300
Learning rate (warmed up)	0.00005	0.00005	0.00005	0.00005

C.3 CFM models trained on perceptual autoencoder features

The vector field for CFM models trained on perceptual autoencoder features uses a modified version of the U-Net parameterisation listed for the baseline, since inputs are of different dimensionality. Table 8 lists the hyperparameters.

Table 8: Hyperparameters for CFM models trained on features from the perceptual autoencoder.

Parameter	MNIST	FashionMNIST	CIFAR10	SVHN
Channels	128	128	128	128
Channels multiple	(1, 2, 2)	(1, 2, 2)	(1, 2, 2, 2)	(1, 2, 2, 2)
Heads	1	1	1	1
Heads channels	1	1	1	1
Attention resolution	16	16	16	16
Dropout	0.0	0.0	0.0	0.0
Batch size	128	128	128	128
Epochs	100	100	100	100
Learning rate (warmed up)	0.0002	0.0002	0.0002	0.0002

C.4 CFM models trained on pixel autoencoder features

The architecture of the pixel autoencoder is described in Section 3.3. A stride length of 2 is used in each of the strided convolutions in the encoder, and in each of the transposed convolutions in the decoder. A kernel size of 3 is used throughout, and the Gaussian error linear unit (GELU) activation function (Hendrycks & Gimpel, 2016) is used. Table 9 lists the modified hyperparameters for CFM models trained on these pixel autoencoder features.

Table 9: Hyperparameters for CFM models trained on features from the pixel autoencoder.

Parameter	MNIST	FashionMNIST	CIFAR10	SVHN
Channels	128	128	128	128
Channels multiple	(1, 2, 2, 2)	(1, 2, 2, 2)	(1, 2, 2, 2)	(1, 2, 2, 2)
Heads	1	1	1	1
Heads channels	1	1	1	1
Attention resolution	16	16	16	16
Dropout	0.0	0.0	0.0	0.0
Batch size	128	128	128	128
Epochs	100	100	100	100
Learning rate (warmed up)	0.0002	0.0002	0.0002	0.0002