Semi-Supervised Learning with Normalizing Flows

Pavel Izmailov [*1] **Polina Kirichenko** [*1] **Marc Finzi** [*1] **Andrew Gordon Wilson** [1]

Abstract

Normalizing flows transform a latent distribution through an invertible neural network for a flexible and pleasingly simple approach to generative modelling, while preserving an exact likelihood. We propose FlowGMM, an end-to-end approach to generative semi-supervised learning with normalizing flows, using a latent Gaussian mixture model. FlowGMM is distinct in its simplicity, unified treatment of labeled and unlabeled data with an exact likelihood, interpretability, and broad applicability beyond image data. We show promising results on a wide range of applications, including AG-News and Yahoo Answers text data, tabular data, and semi-supervised image classification. We also show that FlowGMM can discover interpretable structure, provide real-time optimization-free feature visualizations, and specify well calibrated predictive distributions.

1. Introduction

The discriminative approach to classification models the probability of a class label given an input $p(y|x)$ directly. The generative approach, by contrast, models the class conditional density for the data $p(x|y)$, and then uses Bayes rule to find $p(y|x)$. In principle, generative modelling has long been more alluring, for the effort is focused on creating an interpretable object of interest, and "what I cannot create, I do not understand". In practice, discriminative approaches typically outperform generative methods, and thus are far more widely used.

The challenge in generative modelling is that standard approaches to density estimation are often poor descriptions of high-dimensional natural signals. For example, a Gaussian mixture directly over images, while highly flexible for estimating densities, would specify similarities between images as related to Euclidean distances of pixel intensities, which would be a poor inductive bias for handling translations or representing other salient statistical properties. Recently, generative adversarial networks (Goodfellow et al., 2014), variational autoencoders (Kingma & Welling, 2013), and normalizing flows (Dinh et al., 2014), have led to great advances in unsupervised generative modelling, by leveraging the inductive biases of deep convolutional neural networks.

Normalizing flows are a pleasingly simple approach to generative modelling, which work by transforming a distribution through an invertible neural network. Since the transformation is invertible, it is possible to exactly express the likelihood over the observed data, to train the neural network mapping. The network provides both useful inductive biases, and a flexible approach to density estimation. Normalizing flows also admit controllable latent representations and can be sampled efficiently, unlike auto-regressive models (Papamakarios et al., 2017; Oord et al., 2016). Moreover, recent work (Dinh et al., 2016; Kingma & Dhariwal, 2018; Behrmann et al., 2018; Chen et al., 2019; Song et al., 2019) demonstrated that normalizing flows can produce high-fidelity samples for natural image datasets.

Advances in unsupervised generative modelling, such as normalizing flows, are particularly compelling for *semi-supervised* learning, where we wish to share structure over labeled and unlabeled data, to make better predictions of class labels on unseen data. In this paper, we introduce an approach to semi-supervised learning with normalizing flows, by modelling the density in the *latent space* as a Gaussian mixture, with each mixture component corresponding to a class represented in the labeled data. This *Flow Gaussian Mixture Model* (FlowGMM) is to the best of our knowledge the first approach to semi-supervised learning with normalizing flows that provides an exact joint likelihood over both labeled and unlabeled data, for end-to-end training.[*]

We illustrate FlowGMM with a simple example in Figure 1. We are solving a binary semi-supervised classification problem on the dataset shown in panel (a): the labeled data are shown with triangles colored according to their class, and unlabeled data are shown with blue circles. We introduce

[*]Equal contribution [1]New York University. Correspondence to: Pavel Izmailov <pi390@nyu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[*]A short version of this work first appeared at the ICML 2019 Normalizing Flows Workshop (Izmailov et al., 2019). At the same workshop, Atanov et al. (2019) proposed a different approach that uses a class-conditional normalizing flow as the latent distribution.

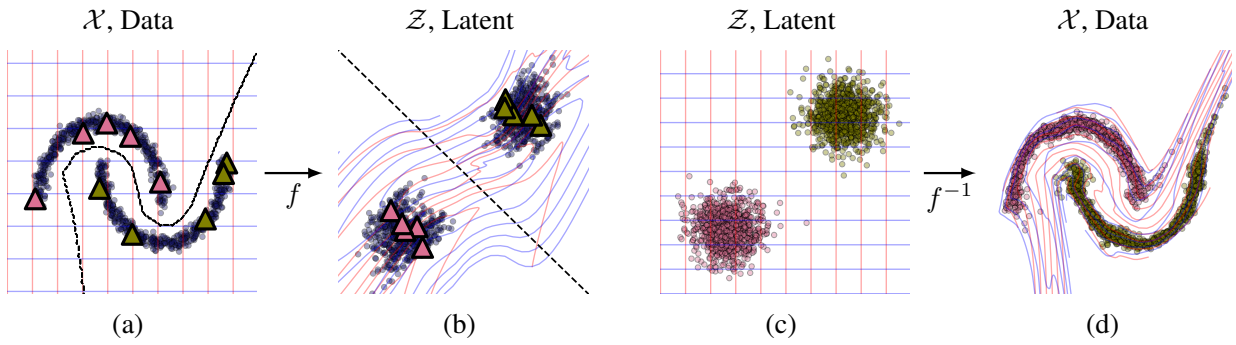


Figure 1. Illustration of semi-supervised learning with FlowGMM on a binary classification problem. Colors represent the two classes or the corresponding Gaussian mixture components. Labeled data are shown with triangles, colored by the corresponding class label, and blue dots represent unlabeled data. **(a):** Data distribution and the classifier decision boundary. **(b):** The learned mapping of the data to the latent space. **(c):** Samples from the Gaussian mixture in the latent space. **(d):** Samples from the model in the data space.

a Gaussian mixture with two components corresponding to each of the classes, shown in panel (c) in the latent space $\mathcal{Z}$ and an invertible transformation f. The transformation f is then trained to map the data distribution in the data space $\mathcal{X}$ to the latent Gaussian mixture in the $\mathcal{Z}$ space, mapping the labeled data to the corresponding mixture component. We visualize the learned transformation in panel (b), showing the positions of the images $f(x)$ for all of the training data points. The inverse f^{-1} of this mapping serves as a class-conditional generative model, that we visualize in panel (d). To classify a data point x in the input space we compute its image $f(x)$ in the latent space, and pick the class corresponding to the Gaussian that is closest to $f(x)$. We visualize the decision boundary of the learned classifier with a dashed line in panel (a).

FlowGMM naturally encodes the *clustering principle*: the decision boundary between classes must lie in the low-density region in the data space. Indeed, in the latent space the decision boundary between two classes coincides with the hyperplane perpendicular to the line segment connecting means of the corresponding mixture components and passing through the midpoint of this line segment (assuming the components are normal distributions with identity covariance matrices); in panel (b) of Figure 1 we show the decision boundary in the latent space with a dashed line. The density of the latent distribution near the decision boundary is low. As the flow is trained to represent data as a transformation of this latent distribution, the density near the decision boundary should also be low. In panel (a) of Figure 1 the decision boundary indeed lies in the low-density region.

The contributions of this work include:

- We propose FlowGMM, a new probabilistic classification model based on normalizing flows that can be naturally applied to semi-supervised learning.
- We show that FlowGMM has good performance on a broad range of semi-supervised tasks, including image, text and tabular data classification.
- We propose a new type of probabilistic consistency regularization that significantly improves FlowGMM on image classification problems.
- To demonstrate the interpretability of FlowGMM, we visualize the learned latent space representations for the proposed semi-supervised model and show that interpolations between data points from different classes pass through low-density regions. We also show how FlowGMM can be used for feature visualization in real-time, without requiring gradients.
- We show that the predictive uncertainties of FlowGMM can be naturally calibrated by scaling the variances of mixture components.

We also provide code at `https://github.com/izmailovpavel/flowgmm`.

2. Related Work

Kingma et al. (2014) represents one of the earliest works on semi-supervised deep generative modelling, demonstrating how the likelihood model of a variational autoencoder (Kingma & Welling, 2013) could be used for semi-supervised image classification. Xu et al. (2017) later extended this framework to semi-supervised text classification.

Many generative models for classification (Salimans et al., 2016; Nalisnick et al., 2019; Chen et al., 2019) have relied upon multitask learning, where a shared latent representation is learned for the generative model and the classifier. With the method of Chen et al. (2019), hybrid modeling is observed to *reduce* performance for both tasks in the

supervised case. While GANs have achieved promising performance on semi-supervised tasks, Dai et al. (2017) showed that classification performance and generative performance are in direct conflict: a perfect generator provides no benefit to classification performance.

Some works on normalizing flows, such as RealNVP (Dinh et al., 2016), have used class-conditional sampling, where the transformation is conditioned on the class label. These approaches pass the class label as an input to coupling layers, conditioning the output of the flow on the class.

Deep Invertible Generalized Linear Model (DIGLM, Nalisnick et al., 2019), most closely related to our work, trains a classifier on the latent representation of a normalizing flow to perform supervised or semi-supervised image classification. Our approach is principally different, as we use a mixture of Gaussians in the latent space $\mathcal{Z}$ and perform classification based on class-conditional likelihoods (see (5)), rather than training a separate classifier. One of the key advantages of our approach is the explicit encoding of clustering principle in the method and a more natural probabilistic interpretation.

Indeed, many approaches to semi-supervised learn from the labeled and unlabeled data using different (and possibly misaligned) objectives, often also involving two step procedures where the unsupervised model is used as pre-processing for a supervised approach (e.g. Nalisnick et al., 2019; Kingma et al., 2014). In general, FlowGMM is distinct in that the generative model is used directly as a Bayes classifier, and in the limit of a perfect generative model the Bayes classifier achieves a provably optimal misclassification rate (see e.g. Mohri et al., 2018). Moreover, approaches to semi-supervised classification, such as consistency regularization (Laine & Aila, 2016; Miyato et al., 2018; Tarvainen & Valpola, 2017; Athiwaratkun et al., 2019; Verma et al., 2019; Berthelot et al., 2019), typically focus on image modelling. We instead focus on showcasing the broad applicability of FlowGMM on text, tabular, and image data, as well as the ability to conveniently discover interpretable structure.

3. Background: Normalizing Flows

The normalizing flow (Dinh et al., 2016) is an unsupervised model for density estimation defined as an invertible mapping $f : \mathcal{X} \to \mathcal{Z}$ from the data space $\mathcal{X}$ to the latent space $\mathcal{Z}$. We can model the data distribution as a transformation $f^{-1} : \mathcal{Z} \to \mathcal{X}$ applied to a random variable from the latent distribution $z \sim p_{\mathcal{Z}}$, which is often chosen to be Gaussian. The density of the transformed random variable $x = f^{-1}(z)$ is given by the change of variables formula

$$p_{\mathcal{X}}(x) = p_{\mathcal{Z}}(f(x)) \cdot \left| \det \left(\frac{\partial f}{\partial x} \right) \right|. \tag{1}$$

The mapping f is implemented as a sequence of invertible functions, parametrized by a neural network with architecture that is designed to ensure invertibility and efficient computation of log-determinants, and a set of parameters θ that can be optimized. The model can be trained by maximizing the likelihood in Equation (1) of the training data with respect to the parameters θ.

4. Flow Gaussian Mixture Model

We introduce the Flow Gaussian Mixture Model (FlowGMM), a probabilistic generative model for semi-supervised learning with normalizing flows. In FlowGMM, we introduce a discrete latent variable y for the class label, $y \in \{1 \dots \mathcal{C}\}$. Our latent space distribution, conditioned on a label k, is Gaussian with mean μ_k and covariance Σ_k:

$$p_{\mathcal{Z}}(z|y = k) = \mathcal{N}(z|\mu_k, \Sigma_k). \tag{2}$$

The marginal distribution of z is then a Gaussian mixture. When the classes are balanced, this distribution is

$$p_{\mathcal{Z}}(z) = \frac{1}{\mathcal{C}} \sum_{k=1}^{\mathcal{C}} \mathcal{N}(z|\mu_k, \Sigma_k). \tag{3}$$

Combining equations (2) and (1), the likelihood for labeled data is

$$p_{\mathcal{X}}(x|y = k) = \mathcal{N}\left(f(x)|\mu_k, \Sigma_k\right) \cdot \left| \det \left(\frac{\partial f}{\partial x} \right) \right|,$$

and the likelihood for data with unknown label is $p_{\mathcal{X}}(x) = \sum_k p_{\mathcal{X}}(x|y = k)p(y = k)$. If we have access to both a labeled dataset $\mathcal{D}_\ell$ and an unlabeled dataset $\mathcal{D}_u$, then we can train our model in a semi-supervised way to maximize the joint likelihood of the labeled and unlabeled data

$$p_{\mathcal{X}}(\mathcal{D}_\ell, \mathcal{D}_u|\theta) = \prod_{(x_i,y_i)\in\mathcal{D}_\ell} p_{\mathcal{X}}(x_i, y_i) \prod_{x_j\in\mathcal{D}_u} p_{\mathcal{X}}(x_j), \tag{4}$$

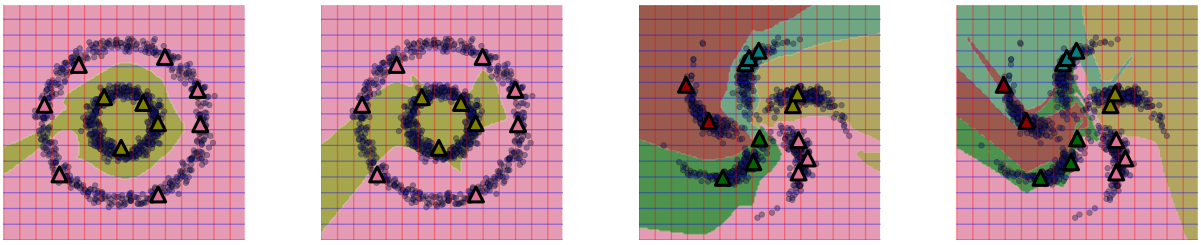
over the parameters θ of the bijective function f, which learns a density model for a Bayes classifier. In particular, given a test point x, the model predictive distribution is given by

$$\begin{aligned} p_{\mathcal{X}}(y|x) &= \frac{p_{\mathcal{X}}(x|y)p(y)}{p(x)} \\ &= \frac{\mathcal{N}\left(f(x)|\mu_y, \Sigma_y\right)}{\sum_{k=1}^{\mathcal{C}} \mathcal{N}(f(x)|\mu_k, \Sigma_k)}. \end{aligned} \tag{5}$$

We can then make predictions for a test point x with the Bayes decision rule

$$y = \arg \max_{i\in\{1,\dots,\mathcal{C}\}} p_{\mathcal{X}}(y = i|x).$$

As an alternative to direct likelihood maximization, we can adapt the Expectation Maximization algorithm for model training as discussed in Appendix A.



(a) Labeled + Unlabeled (b) Labeled Only (c) Labeled + Unlabeled (d) Labeled Only

Figure 2. Illustration of FlowGMM performance on synthetic datasets. Labeled data are shown with colored triangles, and unlabeled data are shown with blue circles. Colors represent different classes. We compare the classifier decision boundaries when only using labeled data (panels b, d) and when using both labeled and unlabeled data (panels a, c) on two circles (panels a, b) and pinwheel (panels c, d) datasets. FlowGMM leverages unlabeled data to push the decision boundary to low-density regions of the space.

4.1. Consistency Regularization

Most of the existing state-of-the-art approaches to semi-supervised learning on image data are based on consistency regularization (Laine & Aila, 2016; Miyato et al., 2018; Tarvainen & Valpola, 2017; Athiwaratkun et al., 2019; Verma et al., 2019; Xie et al., 2020; Berthelot et al., 2020). These methods penalize changes in network predictions with respect to input perturbations, such as random translations and horizontal flips, with an additional loss term that can be computed on unlabeled data,

$$\ell_{cons}(x) = \|g(x') - g(x'')\|^2, \tag{6}$$

where x', x'' are random perturbations of x, and g is the vector of probabilities over the classes.

Motivated by these methods, we introduce a new consistency regularization term for FlowGMM. Let y'' be the label predicted on image x'' by FlowGMM according to (5). We then define our consistency loss as the negative log likelihood of the input x' given the label y'':

$$L_{\text{cons}}(x', x'') = -\log p(x'|y'') = \\ -\log \mathcal{N}(f(x')|\mu_{y''}, \Sigma_{y''}) - \log \left|\det\left(\frac{\partial f}{\partial x'}\right)\right|. \tag{7}$$

This loss term encourages the model to map small perturbations of the same unlabeled inputs to the same components of the Gaussian mixture distribution in the latent space. Unlike the standard consistency loss of (6), the proposed loss in (7) takes values on the same scale as the data log likelihood (4), and indeed we find it to work better in practice. We refer to FlowGMM with the consistency term as FlowGMM-cons. The final loss for FlowGMM-cons is then the weighted sum of the consistency loss (7) and the negative log likelihood of both labeled and unlabeled data (4).

5. Experiments

We evaluate FlowGMM on a wide range of datasets across different application domains including low-dimensional synthetic data (Section 5.1), text and tabular data (Section 5.2), and image data (Sections 5.3, 5.4). In Appendix K we evaluate FlowGMM under class imbalance. We show that FlowGMM outperforms the baselines on tabular and text data. FlowGMM is also state-of-the-art as an *end-to-end generative* approach to semi-supervised image classification, conditioned on architecture. However, FlowGMM is constrained by the RealNVP architecture, and thus does not outperform the most powerful approaches in this setting, which involve discriminative classifiers.

In all experiments, we use the RealNVP normalizing flow architecture. Throughout training, Gaussian mixture parameters are fixed: the means are initialized randomly from the standard normal distribution and the covariances are set to I. See Appendix B for further discussion on GMM initialization and training.

5.1. Synthetic Data

We first apply FlowGMM to a range of two-dimensional synthetic datasets, in order to gain a better visual intuition for the method. We use the RealNVP architecture with 5 coupling layers, defined by fully-connected shift and scale networks, each with 1 hidden layer of size 512. In addition to the semi-supervised setting, we also trained the method only using the labeled data. In Figure 2 we visualize the decision boundaries of the classifier corresponding to FlowGMM for both of these settings on the *two circles* and *pinwheel* datasets. On both datasets, FlowGMM is able to benefit from the unlabeled data to push the decision boundary to a low-density region, as expected. On the two circles problem the method is unable to fit the data perfectly as flows are homeomorphisms, and the disk is topologically distinct from an annulus. However, FlowGMM still pro-

Table 1. Accuracy on BERT embedded text classification datasets and UCI datasets with a small number of labeled examples. The kNN baseline, logistic regression, and the 3-Layer NN + Dropout were trained on the labeled data only. Numbers reported for each method are the median of 4 runs $\pm$ the median average deviation. n_l and n_u are the number of labeled and unlabeled data points.

Method	Dataset (n_l / n_u, classes)			
	AG-News (200 / 200k, 4)	Yahoo Answers (800 / 50k, 10)	Hepmass (20 / 140k, 2)	Miniboone (20 / 65k, 2)
kNN	41.6	19.8	81.2	75.3
Logistic Regression	77.5	54.9	81.2	79.0
3-Layer NN + Dropout	77.5 ± 0.3	55.7 ± 0.3	82.2 ± 0.1	80.4 ± 0.4
RBF Label Spreading	36.1	36.4	84.9	79.3
Π-model	80.2 ± 0.3	56.3 ± 0.04	87.9 ± 0.2	80.8 ± 0.01
FlowGMM	**82.1** ± 1.0	**57.9** ± 0.2	**88.5** ± 0.2	**81.9** ± 0.7

Table 2. Accuracy on semi-supervised classification on CIFAR-10 dataset with features extracted from EfficientNet model pre-trained on ImageNet. We report mean and standard deviation over 3 runs with different labeled data. The kNN baseline and logistic regression were trained on the labeled data only.

Method	$n_l = 250$	$n_l = 1000$	$n_l = 4000$
kNN	84.81 ± 0.17	87.99 ± 0.15	90.41 ± 0.29
Logistic Regression	88.13 ± 0.41	90.40 ± 0.1	92.42 ± 0.17
RBF Label Spreading	88.64 ± 0.67	90.24 ± 0.19	90.99 ± 0.15
kNN Label Spreading	87.20 ± 1.10	89.13 ± 0.15	90.45 ± 0.38
Π-model	89.84 ± 0.25	91.54 ± 0.05	**92.75** ± 0.12
FlowGMM	**90.26** ± 0.45	**91.82** ± 0.09	**92.78** ± 0.03

duces a reasonable decision boundary and improves over the case when only labeled data are available. We provide additional visualizations in Appendix C, Figure 4.

5.2. Text and Tabular Data

FlowGMM can be especially useful for semi-supervised learning on tabular data. Consistency-based semi-supervised methods have mostly been developed for image classification, where the predictions of the method are regularized to be invariant to random flips and translations of the image. On tabular data, desirable invariances are less obvious, finding suitable transformations to apply for consistency-based methods is not trivial. Similarly, approaches based on GANs have mostly been developed for images. We evaluate FlowGMM on the Hepmass and Miniboone UCI classification datasets (previously used in Papamakarios et al. (2017) for density estimation).

Along with standard tabular UCI datasets, we also consider text classification on AG-News and Yahoo Answers datasets. Using the recent advances in transfer learning for NLP, we construct embeddings for input texts using the BERT transformer model (Devlin et al., 2018) trained on a corpus of Wikipedia articles, and then train FlowGMM and other baselines on the embeddings.

We compare FlowGMM to the graph based label spreading method from Zhou et al. (2004), a Π-Model (Laine & Aila, 2016) that uses dropout perturbations, as well as supervised logistic regression, k-nearest neighbors, and a neural network trained on the labeled data only. We report the results in Table 1, where FlowGMM outperforms the alternative semi-supervised learning methods on each of the considered datasets. Implementation details for FlowGMM, the baselines, and data preprocessing details are in Appendix D.

5.3. Transfer Learning on Image Data.

Transfer learning makes use of models pre-trained on large datasets to improve performance on downstream tasks with limited annotated datasets. In practice, models pre-trained on ImageNet (Russakovsky et al., 2015) are often used as feature extractors for computer vision problems. We evaluate FlowGMM in transfer learning setting on CIFAR-10 semi-supervised image classification. We extract image representations for CIFAR-10 using an EfficientNet model (Tan & Le, 2019) pre-trained on ImageNet, and train FlowGMM and baseline models on these features. We report the results in Table 2. In this setting, FlowGMM outperforms Π-Model for 250 and $1k$ labeled examples, and achieves similar performance with $4k$ labels. The implementation details are presented in Appendix E.

Table 3. Accuracy of the FlowGMM, VAE model (M1+M2 VAE, Kingma et al., 2014), DIGLM (Nalisnick et al., 2019) in supervised and semi-supervised settings on MNIST, SVHN, and CIFAR-10. FlowGMM Sup (*All* labels) as well as DIGLM Sup (*All* labels) were trained on full train datasets with all labels to demonstrate general capacity of these models. FlowGMM Sup (n_l labels) was trained on n_l labeled examples (and no unlabeled data). For reference, at the bottom we list the performance of the Π-Model (Laine & Aila, 2016) and BadGAN (Dai et al., 2017) as representative consistency-based and GAN-based state-of-the-art methods. Both of these methods use non-invertible architectures with substantially higher base performance and, thus, are not directly comparable.

Method	Dataset (n_l / n_u)		
	MNIST ($1k/59k$)	SVHN ($1k/72k$)	CIFAR-10 ($4k/46k$)
DIGLM Sup (*All* labels)	99.27	95.74	-
FlowGMM Sup (*All* labels)	99.63	95.81	88.44
M1+M2 VAE SSL	97.60	63.98	-
DIGLM SSL	**99.0**	-	-
FlowGMM Sup (n_l labels)	97.36	78.26	73.13
FlowGMM	98.94	82.42	78.24
FlowGMM-cons	**99.0**	**86.44**	**80.9**
BadGAN	-	95.75	85.59
Π-Model	-	94.57	87.64

Table 4. Semi-supervised classification accuracy for FlowGMM-cons and VAE M1 + M2 model (Kingma et al., 2014) on MNIST for different number of labeled data points n_l.

Method	$n_l = 100$	$n_l = 600$	$n_l = 1000$	$n_l = 3000$
M1+M2 VAE SSL (n_l labels)	96.67	97.41 ± 0.05	97.60 ± 0.02	97.82 ± 0.04
FlowGMM-cons (n_l labels)	98.2	98.7	99	99.2

5.4. Image Classification

We next evaluate the proposed method on semi-supervised image classification benchmarks on CIFAR-10, MNIST and SVHN datasets. For all the datasets, we use the RealNVP (Dinh et al., 2016) architecture. Exact implementation details are listed in the Appendix F. The supervised model is trained using the same loss (4), where all the data points are labeled ($n_u = 0$).

We present the results for FlowGMM and FlowGMM-cons in Table 3. We also report results from DIGLM (Nalisnick et al., 2019), supervised only performance on MNIST and SVHN, and the M1+M2 VAE model (Kingma et al., 2014). FlowGMM outperforms the M1+M2 model and performs better or on par with DIGLM. Furthermore, FlowGMM-cons improves over FlowGMM on all three datasets, suggesting that our proposed consistency regularization is helpful for performance.

Following Oliver et al. (2018), we evaluate FlowGMM-cons varying the number of labeled data points. Specifically, we follow the setup of Kingma et al. (2014) and train FlowGMM-cons on MNIST with 100, 600, 1000 and 3000 labeled data points. We present the results in Table 4. FlowGMM-cons outperforms the M1+M2 model of Kingma et al. (2014) in all the considered settings.[†]

We note that the results presented in this Section are not directly comparable with the state-of-the-art methods using GANs or consistency regularization (see e.g. Laine & Aila, 2016; Dai et al., 2017; Athiwaratkun et al., 2019; Berthelot et al., 2019), as the architecture we employ is much less powerful for classification than the ConvNet and ResNet architectures that have been designed for classification without the constraint of invertibility. We believe that invertible architectures with better inductive biases for classification may help bridge this gap; invertible residual networks (Behrmann et al., 2018; Chen et al., 2019) and invertible CNNs (Finzi et al., 2019) are some of the early examples of this class of architectures.

In general, it is difficult to directly compare FlowGMM with most existing approaches, because the types of architectures available for fully generative normalizing flows are very different than what is available to (partially) discriminative approaches or even other generative methods like VAEs. This difference is due to the invertibility requirement for normalizing flows.

[†]The M1+M2 model has been improved in subsequent work. E.g., ADGM (Maaløe et al., 2016) achieves 99.04% accuracy on MNIST with 100 labels, outperforming FlowGMM-cons.

Table 5. Negative log-likelihood and Expected Calibration Error for supervised FlowGMM trained on MNIST (1k train, 1k validation, 10k test) and CIFAR-10 (50k train, 1k validation, 9k test). FlowGMM-temp stands for tempered FlowGMM where a single scalar parameter σ^2 was learned on a validation set for variances in all components.

	MNIST (test acc. 97.3%)		CIFAR-10 (test acc. 89.3%)	
	FlowGMM	FlowGMM-temp	FlowGMM	FlowGMM-temp
NLL ↓	0.295	0.094	2.98	0.444
ECE ↓	0.024	0.004	0.108	0.038

6. Model Analysis

We empirically analyze different aspects of FlowGMM and highlight some useful features of this model. In Section 6.1 we discuss the calibration of predictive uncertainties produced by the model. In Section 6.2, we study the latent representations learned by FlowGMM. Finally, in Section 6.3, we discuss a feature visualization technique that can be used to interpret the features learned by FlowGMM.

6.1. Uncertainty and Calibration

In many applications, particularly where decision making is involved, it is crucial to have reliable confidences associated with predictions. In classification problems, well-calibrated models are expected to output accurate probabilities of belonging to a particular class. Reliable uncertainty estimation is especially relevant in semi-supervised learning since label information is limited during training. Guo et al. (2017), showed that modern deep learning models are highly overconfident, but could be easily recalibrated with temperature scaling. In this Section, we analyze the predictive uncertainties produced by FlowGMM. In Appendix Section G, we also consider out-of-domain data detection.

When using FlowGMM for classification, the class predictive probabilities are

$$p(y|x) = \frac{\mathcal{N}\left(f(x)|\mu_y, \Sigma_y\right)}{\sum_{k=1}^{\mathcal{C}} \mathcal{N}(f(x)|\mu_k, \Sigma_k)}.$$

Since we initialize Gaussian mixture means randomly from the standard normal distribution and do not train them along with the flow parameters (see Appendix B), FlowGMM predictions become inherently overconfident due to the curse of dimensionality. For example, consider two Gaussians with means sampled independently from the standard normal $\mu_1, \mu_2 \sim \mathcal{N}(0, I)$ in D-dimensional space. If $s_1 \sim \mathcal{N}(\mu_1, I)$ is a sample from the first Gaussian, then its expected squared distances to both mixture means are $\mathbb{E}\left[\|s_1 - \mu_1\|^2\right] = D$ and $\mathbb{E}\left[\|s_1 - \mu_2\|^2\right] = 3D$ (for a detailed derivation see Appendix Section H). In high dimensional spaces, such logits would lead to hard label assignment in FlowGMM ($p(y|x) \approx 1$ for exactly one class). In fact, in the experiments we observe that FlowGMM is overconfident and performs hard label assignment: predicted class probabilities are all close to either 1 or 0.

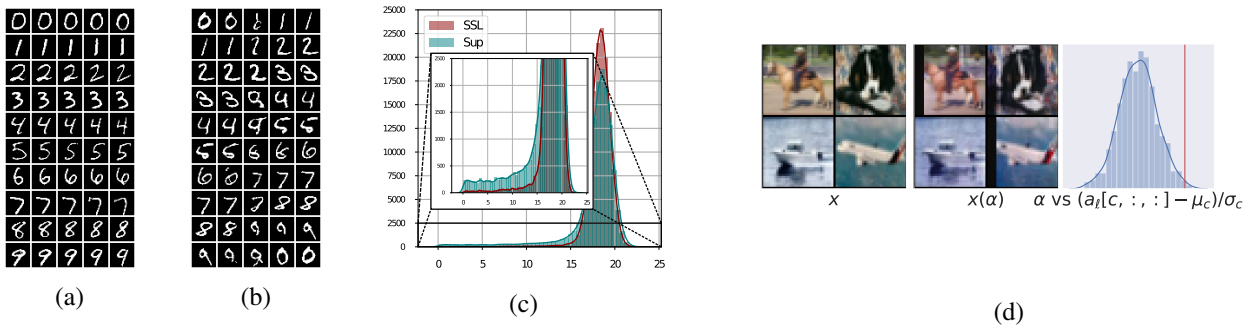
We address this problem by learning a single scalar parameter σ^2 for all components in the Gaussian mixture (the component k will be $\mathcal{N}(\mu_k, \sigma^2 I)$) by minimizing the negative log likelihood on a validation set. This way we can naturally re-calibrate the variance of the latent GMM. This procedure is also equivalent to applying temperature scaling (Guo et al., 2017) to logits $\log \mathcal{N}(x|\mu_k, \Sigma_k)$. We test FlowGMM calibration on MNIST and CIFAR datasets in the supervised setting. On MNIST we restricted the training set size to 1000 objects, since on the full dataset the model makes too few mistakes which makes evaluating calibration harder. In Table 5, we report negative log likelihood and expected calibration error (ECE, see Guo et al. (2017) for a description of this metric). We can see that re-calibrating variances of the Gaussians in the mixture significantly improves both metrics and mitigates overconfidence. The effectiveness of this simple rescaling procedure suggests that the latent space distances learned by the flow model are correlated with the probabilities of belonging to a particular class: the closer a datapoint is to the mean of a Gaussian in the latent space, the more likely it belongs to the corresponding class.

In Appendix J we discuss using mixtures of non-Gaussian latent distributions with heavy tails to address the overconfidence of FlowGMM.

6.2. Learned Latent Representations

We next analyze the latent representation space learned by FlowGMM. We examine latent interpolations between members of the same class in Figure 3(a) and between different classes in Figure 3(b) for our MNIST FlowGMM-cons model trained with $n_\ell = 1k$ labels. As expected, inter-class interpolations pass through regions of low-density, leading to low quality samples but intra-class interpolations do not. These observations suggest that, as expected, the model learns to put the decision boundary in the low-density region of the data space.

In Appendix section I, we present images corresponding to the means of the Gaussian mixture and class-conditional samples from FlowGMM.



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

Figure 3. Visualizations of the latent space representations learned by supervised FlowGMM on MNIST. **(a)**: Latent space interpolations between test images from the same class and **(b)**: from different classes. Observe that interpolations between objects from different classes pass through low-density regions. **(c)**: Histogram of distances from unlabeled data to the decision boundary for FlowGMM-cons trained on $1k$ labeled and $59k$ unlabeled data and FlowGMM Sup trained on $1k$ labeled data only. FlowGMM-cons is able to push the decision boundary away from the data distribution using unlabeled data. **(d)**: Feature visualization for CIFAR-10: four test reconstructions are shown as an intermediate feature is perturbed. The value of the perturbation α is shown in red vs the distribution of the channel activations. Observe that the channel visualized activates on zeroed out pixels to the left of the image mimicking the random translations applied to the training data.

Distance to Decision Boundary To explicitly test this conclusion, we compute the distribution of distances from unlabeled data to the decision boundary for FlowGMM-cons and FlowGMM Sup trained on labeled data only. In order to compute this distance exactly for an image x, we find the two closest means μ', μ'' to the corresponding latent variable $z = f(x)$, and evaluate the expression

$$d(x) = \frac{\left| \|\mu' - f(x)\|^2 - \|\mu'' - f(x)\|^2 \right|}{2\|\mu' - \mu''\|}.$$

We visualize the distributions of the distances for the supervised and semi-supervised method in Figure 3(c). While most of the unlabeled data are far from the decision boundary for both methods, the supervised method puts a substantially larger fraction of data close to the decision boundary. For example, the distance to the decision boundary is smaller than 5 for 1089 unlabeled data points with supervised model, but only 143 data points with FlowGMM-cons. This increased separation suggests that FlowGMM-cons indeed pushes the decision boundary away from the data distribution as would be desired from the clustering principle.

6.3. Feature Visualization

Feature visualization has become an important tool for increasing the interpretability of neural networks in supervised learning. The majority of methods rely on maximizing the activations of a given neuron, channel, or layer over a parametrization of an input image with different kinds of image regularization (Szegedy et al., 2013; Olah et al., 2017; Mahendran & Vedaldi, 2015). These methods, while effective, require iterative optimization too costly for real time interactive exploration. In this Section we discuss a simple and efficient feature visualization technique that leverages the invertibility of FlowGMM. This technique can be used with any invertible model but is especially relevant for FlowGMM, where we can use feature visualization to gain insights into the classification decisions made by the model.

Since our classification model uses a flow which is a sequence of invertible transformations

$$f(x) = f_{:L}(x) := f_L \circ f_{L-1} \circ \cdots \circ f_1(x),$$

intermediate activations can be inverted directly. This means that we can combine the methods of feature inversion and feature maximization directly by feeding in a set of input images, modifying intermediate activations arbitrarily, and inverting the representation. Given a set of activations in the ℓ^{th} layer $a_\ell[c, i, j] = f_{:\ell}(x)_{cij}$ with channels c and spatial extent i, j, we may perturb a single neuron with

$$x(\alpha) = f_{:\ell}^{-1}(f_{:\ell}(x) + \alpha\sigma_c\delta_c),$$

where δ_c is a one hot vector at channel c; and σ_c is the standard deviation of the activations in channel c over the the training set and spatial locations. This procedure can be performed at real-time rates to explore the activation parametrized by α and the location (c, i, j) without any optimization or hyper-parameters. We show the feature visualization for intermediate layers on CIFAR-10 test images in Figure 3(d). The channel being visualized appears to activate on the zeroed pixels from random translations as well as the green channel. Analyzing the features learned by FlowGMM we can gain insight into the workings of the model.

7. Discussion

We proposed a simple and interpretable approach for end-to-end generative semi-supervised prediction with normalizing flows. While FlowGMM does not yet outperform the most powerful discriminative approaches for semi-supervised image classification (Athiwaratkun et al., 2019; Verma et al., 2019), we believe it is a promising step towards making fully generative approaches more practical for semi-supervised tasks. As we develop improved invertible architectures, the performance of FlowGMM will also continue to improve.

Moreover, FlowGMM does outperform graph-based and consistency-based baselines on tabular data including semi-supervised text classification with BERT embeddings. We believe that the results show promise for generative semi-supervised learning based on normalizing flows, especially for tabular tasks where consistency-based methods struggle.

We view interpretability and broad applicability as a strong advantage of FlowGMM. The access to latent space representations and the feature visualization technique discussed in Section 6 as well as the ability to sample from the model can be used to obtain insights into the performance of the model in practical applications.

Acknowledgements

This research is supported by an Amazon Research Award, Facebook Research, Amazon Machine Learning Research Award, NSF I-DISRE 193471, NIH R01 DA048764-01A1, NSF IIS-1910266, and NSF 1922658 NRT-HDR: FUTURE Foundations, Translation, and Responsibility for Data Science.

References

Atanov, A., Volokhova, A., Ashukha, A., Sosnovik, I., and Vetrov, D. Semi-conditional normalizing flows for semi-supervised learning. *arXiv preprint arXiv:1905.00505*, 2019.

Athiwaratkun, B., Finzi, M., Izmailov, P., and Wilson, A. G. There are many consistent explanations of unlabeled data: Why you should average. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rkgKBhA5Y7.

Behrmann, J., Duvenaud, D., and Jacobsen, J.-H. Invertible residual networks. *arXiv preprint arXiv:1811.00995*, 2018.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. MixMatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklkeR4KPB.

Chen, R. T., Behrmann, J., Duvenaud, D., and Jacobsen, J.-H. Residual flows for invertible generative modeling. *arXiv preprint arXiv:1906.02735*, 2019.

Dai, Z., Yang, Z., Yang, F., Cohen, W. W., and Salakhutdinov, R. R. Good semi-supervised learning that requires a bad GAN. In *Advances in Neural Information Processing Systems 30*, pp. 6510–6520, 2017.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Finzi, M., Izmailov, P., Maddox, W., Kirichenko, P., and Wilson, A. G. Invertible convolutional networks. In *Workshop on Invertible Neural Nets and Normalizing Flows, International Conference on Machine Learning*, 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. *CoRR*, abs/1706.04599, 2017. URL http://arxiv.org/abs/1706.04599.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1×1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

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

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pp. 3581–3589, 2014.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

Mahendran, A. and Vedaldi, A. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5188–5196, 2015.

Miyato, T., Maeda, S.-i., Ishii, S., and Koyama, M. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of machine learning*. MIT press, 2018.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. Hybrid models with deep and invertible features. *arXiv preprint arXiv:1902.02767*, 2019.

Olah, C., Mordvintsev, A., and Schubert, L. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Oliver, A., Odena, A., Raffel, C. A., Cubuk, E. D., and Goodfellow, I. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 3235–3246, 2018.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training GANs. In *Advances in neural information processing systems*, pp. 2234–2242, 2016.

Song, Y., Meng, C., and Ermon, S. Mintnet: Building invertible neural networks with masked convolutions. In *Advances in Neural Information Processing Systems*, pp. 11002–11012, 2019.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Tarvainen, A. and Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pp. 1195–1204, 2017.

Verma, V., Lamb, A., Kannala, J., Bengio, Y., and Lopez-Paz, D. Interpolation consistency training for semi-supervised learning. *arXiv preprint arXiv:1903.03825*, 2019.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. Unsupervised data augmentation for consistency training, 2020. URL https://openreview.net/forum?id=ByeL1R4FvS.

Xu, W., Sun, H., Deng, C., and Tan, Y. Variational autoencoder for semi-supervised text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. Learning with local and global consistency. In *Advances in neural information processing systems*, pp. 321–328, 2004.

A. Expectation Maximization

As an alternative to direct optimization of the likelihood (4), we consider Expectation-Maximization algorithm (EM). EM is a popular approach for finding maximum likelihood estimates in mixture models. Suppose $X = \{x_i\}_{i=1}^n$ is the observed dataset, $T = \{t_i\}_{i=1}^n$ are corresponding unobserved latent variables (often denoting the component in mixture model) and θ is a vector of model parameters. EM algorithm consists of the two alternating steps: on E-step, we compute posterior probabilities of latent variables for each data point $q(t_i|x_i) = P(t_i|x_i, \theta)$; and on M-step, we fix q and maximize the expected log likelihood of the data and latent variables with respect to θ: $\mathbb{E}_q \log P(X, T|\theta) \to \max_\theta$. The algorithm can be easily adapted to the semi-supervised setting where a subset of data is labeled with $\{y_i^l\}_{i=1}^{n_l}$: then, on E-step we have hard assignment to the true mixture component $q(t_i|x_i) = I[t_i = y_i^l]$ for labeled data points.

EM is applicable to fitting the transformed mixture of Gaussians. We can perform the exact E-step for unlabeled data in the model since

$$q(t|x) = \frac{p(x|t,\theta)}{p(x|\theta)} = \frac{\mathcal{N}(f(x)|\mu_t, \Sigma_t) \cdot \left|\det\left(\frac{\partial f}{\partial x}\right)\right|}{\sum_{k=1}^{\mathcal{C}} \mathcal{N}(f(x)|\mu_k, \Sigma_k) \cdot \left|\det\left(\frac{\partial f}{\partial x}\right)\right|} = \frac{\mathcal{N}(f(x)|\mu_t, \Sigma_t)}{\sum_{k=1}^{\mathcal{C}} \mathcal{N}(f(x)|\mu_k, \Sigma_k)}$$

which coincides with the E-step of EM algorithm on Gaussian mixture model. On M-step, the objective has the following form:

$$\sum_{i=1}^{n_l} \log\left[\mathcal{N}(f_\theta(x_i^l)|\mu_{y_i^l}, \Sigma_{y_i^l})\left|\frac{\partial f_\theta}{\partial x_i^l}\right|\right] + \sum_{i=1}^{n_u} \mathbb{E}_{q(t_i|x_i^u,\theta)} \log\left[\mathcal{N}(f_\theta(x_i^u)|\mu_{t_i}, \Sigma_{t_i})\left|\frac{\partial f_\theta}{\partial x_i^u}\right|\right].$$

Since the exact solution is not tractable due to complexity of the flow model, we perform a stochastic gradient step to optimize the expected log likelihood with respect to flow parameters θ.

Note that unlike regular EM algorithm for mixture models, we have Gaussian mixture parameters $\{(\mu_k, \Sigma_k)\}_{k=1}^{\mathcal{C}}$ fixed in our experiments, and on M-step the update of θ induces the change of $z_i = f_\theta(x_i)$ latent space representations.

Using EM algorithm for optimization in the semi-supervised setting on MNIST dataset with 1000 labeled images, we obtain 98.97% accuracy which is comparable to the result for FlowGMM with regular SGD training. However, in our experiments, we observed that on E-step, hard label assignment happens for unlabeled points ($q(t|x) \approx 1$ for one of the classes) because of the high dimensionality of the problem (see section 6.1) which affects the M-step objective and hinders training.

B. Latent Distribution Mean and Covariance Choices

Initialization In our experiments, we draw the mean vectors μ_i of Gaussian mixture model randomly from the standard normal distribution $\mu_i \sim \mathcal{N}(0, I)$, and set the covariance matrices to identity $\Sigma_i = I$ for all classes; we fixed GMM parameters throughout training. However, one could potentially benefit from data-dependent placing of means in the latent space. We experimented with different initialization methods, in particular, initializing means using the mean point of latent representations of labeled data in each class: $\mu_i = (1/n_l^i) \sum_{m=1}^{n_l^i} f(x_m^i)$ where x_m^i represents labeled data points from class i and n_l^i is the total number of labeled points in that class. In addition, we can scale all means by a scalar value $\hat{\mu}_i = r\mu_i$ to increase or decrease distances between them. We observed that such initialization leads to much faster convergence of FlowGMM on semi-supervised classification on MNIST dataset, however, the final performance of the model was worse compared to the one with random mean placing. We hypothesize that it becomes easier for the flow model to warm up faster with data-dependent initialization because Gaussian means are closer to the initial latent representations, but afterwards the model gets stuck in a suboptimal solution.

GMM training FlowGMM would become even more flexible and expressive if we could learn Gaussian mixture parameters in a principled way. In the current setup where means are sampled from the standard normal distribution, the distances between mixture components are about $\sqrt{2D}$ where D is the dimensionality of the data (see Appendix H). Thus, classes are quite far apart from each other in the latent space, which, as observed in Section 6.1, leads to model miscalibration. Training GMM parameters can further increase interpretability of the learned latent space representations: we can imagine a scenario in which some of the classes are very similar or even intersecting, and it would be useful to represent it in the latent space. We could train GMM by directly optimizing likelihood (4), or using expectation maximization (see Section A), either jointly with the flow parameters or iteratively switching between training flow parameters with the fixed GMM and training GMM with the fixed flow. In our initial experiments on semi-supervised classification on MNIST, training GMM jointly with the flow parameters did not improve performance or lead to substantial change of the latent representations. Further improvements require careful hyper-parameter choice which we leave for future work.

Table 6. Tuned learning rates for 3-Layer NN + Dropout, Π-model and method on text and tabular tasks. For kNN we report the number of neighbours. All hyper-parameters were tuned via cross-validation.

Method	AG-News	Yahoo Answers	Hepmass	Miniboone
3-Layer NN + Dropout	3e-4	3e-4	3e-4	3e-4
Π-model	1e-3	1e-4	3e-3	1e-4
FlowGMM	3e-4	3e-4	3e-3	3e-4
kNN	$k = 5$	$k = 19$	$k = 9$	$k = 3$

C. Synthetic Experiments

In Figure 4 we visualize the classification decision boundaries of FlowGMM as well as the learned mapping to the latent space and generated samples for three different synthetic datasets.

In Table 7 we compare FlowGMM and SCNF of Atanov et al. (2019) on two synthetic datasets. For the experiments we use 1000 data points with 5 labeled examples per class which we select randomly. To get the error bars we run the experiment 5 times with different labeled data.

D. Tabular data preparation and hyperparameters

The AG-News and Yahoo Answers were constructed by applying BERT embeddings to the text input, yielding a 768 dimensional vector for each data point. AG-News has 4 classes while Yahoo Answers has 10. The UCI datasets Hepmass and Miniboone were constructed using the data preprocessing from Papamakarios et al. (2017), but with the inclusion of the removed background process class so that the two problems can be used for binary classification. We then subsample the fraction of background class examples so that the dataset is balanced. For each of the datasets, a separate validation set of size 5k was used to tune hyperparameters. All neural network models use the ADAM optimizer (Kingma & Ba, 2014).

k-Nearest Neighbors: We tested both using L2 distance and L2 with inputs normalized to unit norm, ($\sin^2$ distance), and the latter performed the best. The value k chosen in the method was found in the range from 1 to 20, and the optimal values for each of the datasets are shown in Table 6.

3 Layer NN + Dropout: The 3-Layer NN + Dropout baseline network has three fully connected hidden layers with inner dimension $k = 512$, ReLU nonlinearities, and dropout with $p = 0.5$. We use the learning rate $3\text{e}{-4}$ for training the supervised baseline across all datasets.

Π-Model: The Π-Model uses the same network architecture, and dropout for the perturbations. The first loss term is the standard cross-entropy for labeled data. The additional consistency loss per unlabeled data point is computed as $L_{\text{Cons}} = ||g(z') - g(z'')||^2$, where g is the the softmax function, and z' and z'' are logit vectors after two evaluations of dropout neural network. We chose the consistency weight $\lambda = 30$ which worked the best across the datasets. The model was trained for 50 epochs with labeled and unlabeled batch size n_ℓ for AG-News and Yahoo Answers, and labeled and unlabeled batch sizes n_ℓ and 2000 for Hepmass and Miniboone.

Label Spreading: We use the local and global consistency method from Zhou et al. (2004), $Y^* = (I - \alpha S)^{-1} Y$ where in our case Y is the matrix of labels for the labeled, unlabeled, and test data but filled with zeros for unlabeled and test. $S = D^{-1/2} W D^{-1/2}$ computed from the affinity matrix $W_{ij} = \exp\left(-\gamma \sin^2(x_i, x_j)\right)$ where $\sin^2(x_i, x_j) := 1 - \frac{\langle x_i, x_j \rangle}{\|x_i\| \|x_j\|}$. This is equivalent to L2 distance on the inputs normalized to unit magnitude. Because the algorithm scales poorly with number of unlabeled points for dense affinity matrices, $O(n_u^3)$, we subsampled the number of unlabeled data points to $10k$ and test data points to $5k$ for this graph method. However, we also evaluate the label spreading algorithm with a sparse kNN affinity matrix on using a larger subset $20k$ of unlabeled data. The two hyperparameters for label spreading (γ/k and α) were tuned by separate grid search for each of the datasets. In both cases, we use the inductive variant of the algorithm where the test data is not included in the unlabeled data.

FlowGMM: We train our FlowGMM model with a RealNVP normalizing flow, similar to the architectures used in Papamakarios et al. (2017). Specifically, the model uses 7 coupling layers, with 1 hidden layer each and 256 hidden units for the UCI datasets but 1024 for text classification. UCI models were trained for 50 epochs of unlabeled data and the text datasets were trained for 200 epochs of unlabeled data. The labeled and unlabeled batch sizes are the same as in the Π-Model.

The tuned learning rates for each of the models that we used for these experiments are shown in Table 6.

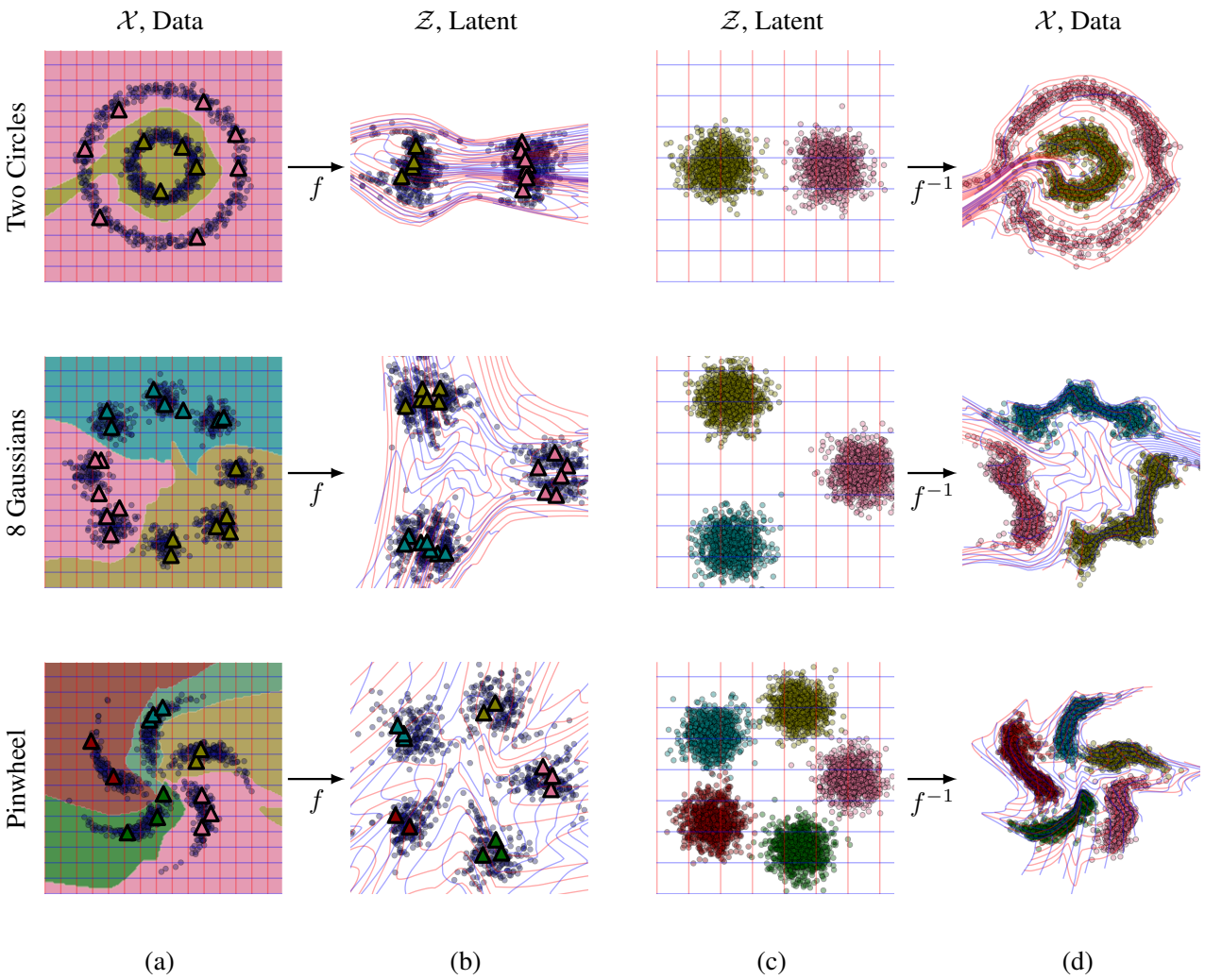


Figure 4. Illustration of FlowGMM on synthetic datasets: two circles (top row), eight Gaussians (middle row) and pinwheel (bottom row). **(a):** Data distribution and classification decision boundaries. Unlabeled data are shown with blue circles and labeled data are shown with colored triangles, where color represents the class. Background color visualizes the classification decision boundaries of FlowGMM. **(b):** Mapping of the data to the latent space. **(c):** Gaussian mixture in the latent space. **(d):** Samples from the learned generative model corresponding to different classes, as shown by their color.

Table 7. Comparison of FlowGMM and SCNF Glow+Glow of Atanov et al. (2019) on synthetic data. For both datasets we use 1000 data points with 5 labeled examples per class. We report mean and standard deviation over 5 runs with different labeled data. FlowGMM achieves similar accuracy and better NLL compared to SCNF.

	FlowGMM		SCNF Glow+Glow	
Data	NLL	Acc (%)	NLL	Acc (%)
Moons	0.82 ± 0.68	99.4 ± 1.1	1.11 ± 0.02	99.7 ± 0.2
Circles	0.83 ± 0.04	97.52 ± 0.5	1.68 ± 0.13	95 ± 1.9

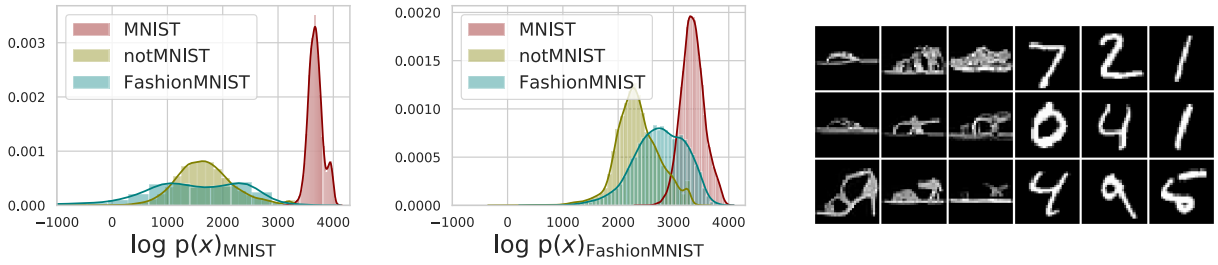


Figure 5. **Left:** Log likelihoods on in- and out-of-domain data for our model trained on MNIST. **Center:** Log likelihoods on in- and out-of-domain data for our model trained on FashionMNIST. **Right:** MNIST digits get mapped onto the sandal mode of the FashionMNIST model 75% of the time, often being assigned higher likelihood than elements of the original sandal class. Representative elements are shown above.

E. Transfer learning

We extracted CIFAR-10 features from EfficientNet (Tan & Le, 2019) pre-trained on ImageNet dataset, which yields 1792 dimensional representations for each image. We test the performance of FlowGMM and baseline models in semi-supervised classification using 250, 1000 and 4000 labeled examples. The results are presented in Table 2. We report mean and standard deviation of 3 runs with different splits on labeled and unlabeled data.

The kNN and logistic regression baselines were trained using labeled data only. For each number of labeled examples for all models, hyperparameters were chosen on a validation data split. In most experiments, we use a setup similar to the experiments on tabular data (see section D).

k-Nearest Neighbours: We tested kNN with L2 distance using unnormalized and normalized features with the value k ranging from 1 to 20.

Π-Model: For the Π-Model, we used a fully-connected neural network with the inner dimension $k = 512$ and dropout with $p = 0.5$. The consistency loss was computed for both labeled and unlabeled data. We perform grid search for the number of hidden layers, learning rate and consistency term weight. We train the model for 30 epochs with the batch size 50 (with 25 labeled and 25 unlabeled examples in each batch).

FlowGMM: We train our FlowGMM model with RealNVP normalizing flow with fully-connected neural networks with 1 hidden layer in coupling layers. We perform the grid search for the number of coupling layers, the number of hidden units in fully-connected networks, and learning rate. The model was trained for 800 epochs of unlabelled data with batch size 50 (25 labeled and 25 unlabeled examples in each batch).

F. Image data preparation and hyperparameters

We use the RealNVP multi-scale architecture with 2 scales, each containing 3 coupling layers defined by 8 residual blocks with 64 feature maps. We use Adam optimizer (Kingma & Ba, 2014) with learning rate 10^{-3} for CIFAR-10 and SVHN and 10^{-4} for MNIST. We train the supervised model for 100 epochs, and semi-supervised models for 1000 passes through the labeled data for CIFAR-10 and SVHN and 3000 passes for MNIST. We use a batch size of 64 and sample 32 labeled and 32 unlabeled data points in each mini-batch. For the consistency loss term (7), we linearly increase the weight from 0 to 1 for the first 100 epochs following Athiwaratkun et al. (2019). For FlowGMM and FlowGMM-cons, we re-weight the loss on labeled data by $\lambda = 3$ (value tuned on validation in Kingma et al. (2014) on CIFAR-10), as otherwise, we observed that the method underfits the labeled data.

G. Out-of-domain data detection

Density models have held promise for being able to detect out-of-domain data, an especially important task for robust machine learning systems (Nalisnick et al., 2019). Recently, it has been shown that existing flow and autoregressive density models are not as apt at this task as previously thought, yielding high likelihood on images coming from other (simpler) distributions. The conclusion put forward is that datasets like SVHN are encompassed by, or have roughly the same mean but lower variance than more complex datasets like CIFAR-10 (Nalisnick et al., 2018). We examine this hypothesis in the context of our flow model which has a multi-modal latent space distribution unlike methods considered in Nalisnick et al. (2018).

Using a fully supervised model trained on MNIST, we evaluate the log likelihood for data points coming from the NotMNIST dataset, consisting of letters instead of digits, and the FashionMNIST dataset. We then train a supervised model on the more complex dataset FashionMNIST and evaluate on MNIST and NotMNIST. The distribution of the log likelihood $\log p_{\mathcal{X}}(\cdot) = \log p_{\mathcal{Z}}(f(\cdot)) + \log \left|\det \left(\frac{\partial f}{\partial x}\right)\right|$ on these datasets is shown in Figure 5. For the model trained on MNIST we see that the data from Fashion MNIST and NotMNIST is assigned lower likelihood, as expected. However, the model trained on FashionMNIST predicts higher likelihoods for MNIST images. The majority ($\approx 75\%$) of the MNIST data points get mapped into the mode of the FashionMNIST model corresponding to sandals, which is the class with the largest fraction of pixels that are zero. Similarly, for the model trained on MNIST the image of all zeros has very high likelihood and gets mapped to the mode corresponding to the digit 1 which has the largest fraction of empty space.

H. Expected Distances between Gaussian Samples

Consider two Gaussians with means sampled independently from the standard normal $\mu_1, \mu_2 \sim \mathcal{N}(0, I)$ in D-dimensional space. If $s_1 \sim \mathcal{N}(\mu_1, I)$ is a sample from the first Gaussian, then its expected squared distances to both mixture means are:

$$\begin{aligned}
\mathbb{E}\left[\|s_1 - \mu_1\|^2\right] &= \mathbb{E}\left[\mathbb{E}\left[\|s_1 - \mu_1\|^2 | \mu_1\right]\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{D} \mathbb{E}\left[(s_{1,i} - \mu_{1,i})^2 | \mu_{1,i}\right]\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{D} \left(\mathbb{E}[s_{1,i}^2] - 2\mu_{1,i}^2 + \mu_{1,i}^2\right)\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{D} \left(1 + \mu_{1,i}^2 - \mu_{1,i}^2\right)\right] = D
\end{aligned}$$

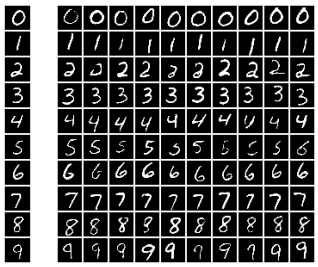
$$\begin{aligned}
\mathbb{E}\left[\|s_1 - \mu_2\|^2\right] &= \mathbb{E}\left[\mathbb{E}\left[\|s_1 - \mu_2\|^2 | \mu_1, \mu_2\right]\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{D} \mathbb{E}\left[(s_{1,i} - \mu_{2,i})^2 | \mu_{1,i}, \mu_{2,i}\right]\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{D} \left(1 + \mu_{1,i}^2 - 2\mu_{1,i}\mu_{2,i} + \mu_{2,i}^2\right)\right] = 3D
\end{aligned}$$

For high-dimensional Gaussians the random variables $\|s_1 - \mu_1\|^2$ and $\|s_1 - \mu_2\|^2$ will be concentrated around their expectations. Since the function $\exp(-x)$ decreases rapidly to zero for positive x, the probability of s_1 belonging to the first Gaussian

$$\begin{aligned}
&\frac{\exp(-\|s_1 - \mu_1\|^2)}{\exp(-\|s_1 - \mu_1\|^2) + \exp(-\|s_1 - \mu_2\|^2)} \approx \\
&\approx \frac{\exp(-D)}{\exp(-D) + \exp(-3D)} = \frac{1}{1 + \exp(-2D)}
\end{aligned}$$

saturates at 1 with the growth of dimensionality D.

I. FlowGMM as generative model



(a) (b)

Figure 6. Visualizations of the latent space representations learned by supervised FlowGMM on MNIST. **(a)**: Images corresponding to means of the Gaussians corresponding to different classes. **(b)**: Class-conditional samples from the model at a reduced temperature $T = 0.25$.

In Figure 6a we show the images $f^{-1}(\mu_i)$ corresponding to the means of the Gaussians representing each class. We see that the flow correctly learns to map the means to samples from the corresponding classes. Next, in Figure 6b we show class-conditional samples from the model. To produce a sample from class i, we first generate $z \sim \mathcal{N}(\mu_i, TI)$, where T is a temperature parameter that controls trade-off between sample quality and diversity; we then compute the samples as $f^{-1}(z)$. We set $T = 0.25^2$ to produce samples in Figure 6b. As we can see, FlowGMM can produce reasonable class-conditional samples simultaneously with achieving a high classification accuracy (99.63%) on the MNIST dataset.

J. Non-Gaussian Latent Distributions

In Section 6.1 we showed that FlowGMM is over-confident: it assigns class probabilities very close to 0 or 1 for all predictions. We showed that the model is over-confident due to the properties of Gaussian distributions in high dimensions (see Appendix H). In this section, we replace the Gaussian base distributions in FlowGMM with a more heavy-tailed Student-t distributions.

We train the model on MNIST with 1000 labeled data points and reuse the hyper-parameters reported in Appendix F. We use a mixture of Student-t distributions with iid components and set the number of degrees of freedom in each component to 5 and the scale to 1. The means are sampled randomly in the same way as in the Gaussian case.

Of the 10000 test images, FlowGMM with Student-t based distributions only assigned confidence less than 0.99 to 24 data points. While the confidences are less extreme compared to FlowGMM with a Gaussian mixture, the model is still severely over-confident. So, unlike temperature scaling, replacing the Gaussian mixture with a mixture of heavier-tailed Student-t distributions does not resolve the issue of over-confidence.

K. FlowGMM under Class Imbalance

In practice, often the classes in the data are not balanced: some classes contain more examples than others. In semi-supervised setting, we may have the same number of labeled examples per class, but the unlabeled data can be unevenly distributed between classes. In this section we evaluate FlowGMM in this setting.

We use the MNIST dataset and drop a subset of data from some of the classes. For the classes $0-2$ we keep all data, for classes $3-5$ we keep 75% of the data, for classes $6-8$ we keep 50% of the data, and in class 9 we only keep 25% of the data. For all classes we use 100 labeled data points per class. We add the class probabilities p_k to the FlowGMM model:

$$p_{\mathcal{Z}}(z) = \sum_{k=1}^{\mathcal{C}} p_k \cdot \mathcal{N}(z|\mu_k, \Sigma_k). \tag{8}$$

We then train FlowGMM as usual, but also optimizing for the class probabilities p_k. The model learned the following class probabilities: $[0.12, 0.13, 0.12, 0.1, 0.1, 0.1, 0.9, 0.9, 0.8, 0.7]$. These probabilities do not exactly reflect the proportions of the data: FlowGMM overestimates the probability of the class 9 that is least represented in the data. However, the probabilities learned by FlowGMM are correlated with the data proportions: the classes that have more unlabeled datapoints get assigned higher probability.