Semi-Supervised Learning with Normalizing Flows

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

Abstract

We propose a new approach for joint modelling of the labels and data of complex distributions using normalizing flows, Flow Gaussian Mixture Model (FlowGMM). FlowGMM models the data as a mixture of complex distributions, implemented by an invertible transformation of a Gaussian mixture. This hybrid approach is particularly well suited for semi-supervised learning, where FlowGMM learns to push the decision boundary towards a low-density region of the data space. We evaluate the proposed model on a range of semi-supervised image classification problems: MNIST, SVHN, CIFAR-10 datasets. Following semi-supervised learning literature, we propose a modified consistency regularization term for our model which substantially improves performance.

1. Introduction

In many domains unlabeled data is plentiful, while labeled data may be scarce. Semi-supervised learning framework leverages both labeled and unlabeled data reducing the need for expensive manual annotation. Consistency-based methods have shown outstanding performance in semi-supervised image classification (Laine and Aila, 2016; Miyato et al., 2018b; Tarvainen and Valpola, 2017; Athiwaratkun et al., 2019; Verma et al., 2019; Berthelot et al., 2019) and are currently state of the art on challenging datasets like CIFAR-10, CIFAR-100, and Imagenet. However, these methods have not seen much application on domains other than images, where a suitable set of data perturbations to which we want the classifier to be invariant are not known a priori.

Generative models provide a more generally applicable approach to semi-supervised learning. In the work of Kingma et al. (2014), it was shown how the likelihood model of Variational Autoencoder (Kingma and Welling, 2013) could be used for semi-supervised image classification on datasets like MNIST and SVHN. Xu et al. (2017) later extended this framework to semi-supervised text classification.

Generative Adversarial Networks (GANs) have shown remarkable improvement in producing high quality, diverse image samples (Goodfellow et al., 2014; Karras et al., 2017; Miyato et al., 2018a; Brock et al., 2018). While GANs do not have a direct likelihood model, they have been employed for semi-supervised learning through multitask learning objective where the model learns to simultaneously discriminate generated images from real (labeled and unlabeled) images and classify labeled data (Salimans et al., 2016). However, the efficacy of the GAN based SSL methods are not well understood and it has been shown that a GAN with a perfect generator would yield no benefit for classification (Dai et al., 2017); furthermore, training GAN-based methods with more powerful discriminators has proven to be challenging.

Normalizing Flows (NF) (Dinh et al., 2014) present another major class of deep generative models. Unlike GANs and VAEs, normalizing flows can be trained using exact maximum likelihood. NFs admit controllable latent representations and can be sampled efficiently unlike autoregressive models (Papamakarios et al., 2017; Oord et al., 2016). Recent work (Dinh et al., 2016; Kingma and Dhariwal, 2018; Behrmann et al., 2018) demonstrated that normalizing flows can produce high-fidelity samples for natural image datasets.

In this paper, we propose Flow Gaussian Mixture Model (FlowGMM), a simple and natural approach to semi-supervised learning based on normalizing flows. The main idea of FlowGMM is to map each data class to one component in the Gaussian mixture using an invertible transformation (flow). Specifically, in semi-supervised setting the labeled data from a given class is modeled as transformation of the corresponding Gaussian, while unlabeled data is modeled as a transformation of the mixture of the components corresponding to all classes. Due to invertibility, we can compute exact likelihood of the data using the change of variable formula. The model can be trained by maximizing likelihood with respect to the parameters of the flow.

We illustrate FlowGMM on a toy problem in Figure 1. The data distribution is shown in panel (a), the mapping of the data to the latent space is shown in panel (b) and the la-

[*]Equal contribution [1]Cornell University. Correspondence to: Andrew Gordon Wilson <andrew@cornell.edu>.

First workshop on *Invertible Neural Networks and Normalizing Flows* (ICML 2019), Long Beach, CA, USA

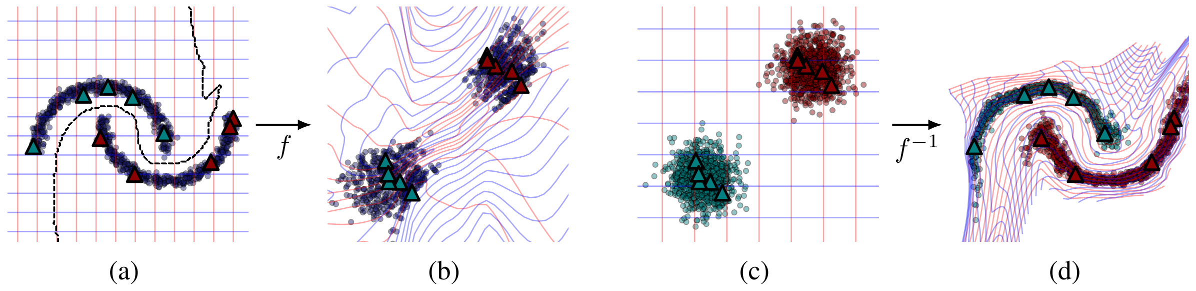

Figure 1. Illustration of semi-supervised learning with Normalizing flows. Labeled data is shown with triangles, colored by the corresponding class label, and blue dots represent unlabeled data.

tent distribution is shown in panel (c) of Figure 1. Test data points are classified to belong to the class represented by the mixture component with the highest density at the corresponding latent points. Panel (a) of Figure 1 shows the decision boundary produced by the classifier.

FlowGMM is based on the *clustering principle*: the decision boundary between classes must lie in the low-density region. Indeed, in the latent space the decision boundary coincides with the perpendicular bisectors of the line segments connecting means of the mixture components (assuming the components are standard normal distributions) and the density of the latent distribution near 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 we can see that the decision boundary indeed lies in a low-density region.

The contributions of this work are as follows:

- We propose FlowGMM, a new classification model based on normalizing flows, that can be naturally applied to semi-supervised learning. We evaluate FlowGMM on a range of consequential semi-supervised image classification benchmarks.
- We propose modified consistency regularization for FlowGMM and empirically demonstrate that it substantially improves performance of the method.
- We conduct a thorough empirical analysis of FlowGMM for supervised and semi-supervised classification. In particular, we show that the predictive uncertainties of the method can be calibrated by scaling the variance of mixture components. We study the ability of FlowGMM to detect out-of-domain data. 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. Finally, we show how our classification model can be used for optimization free feature visualization.

2. 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 $z \sim p_\mathcal{Z}$ often chosen to be Gaussian. The density of the transformed random variable $x = f^{-1}(z)$ is given by the change of variables formula

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

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 (1) of the training data with respect to the parameters θ.

3. Method

3.1. Flow Mixture Model

In our model, we add a discrete latent variable y for the class label, $y \in \{1 \ldots \mathcal{C}\}$. Our latent space distribution, conditioned on a given label k, is Gaussian with mean μ_k and covariance Σ_k:

$$z|y = k \sim \mathcal{N}(\mu_k, \Sigma_k). \quad (2)$$

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

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

Using our flow f, we sample data points conditionally or unconditionally with $x = f^{-1}(z)$. Thus, the likelihood for labeled data is

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

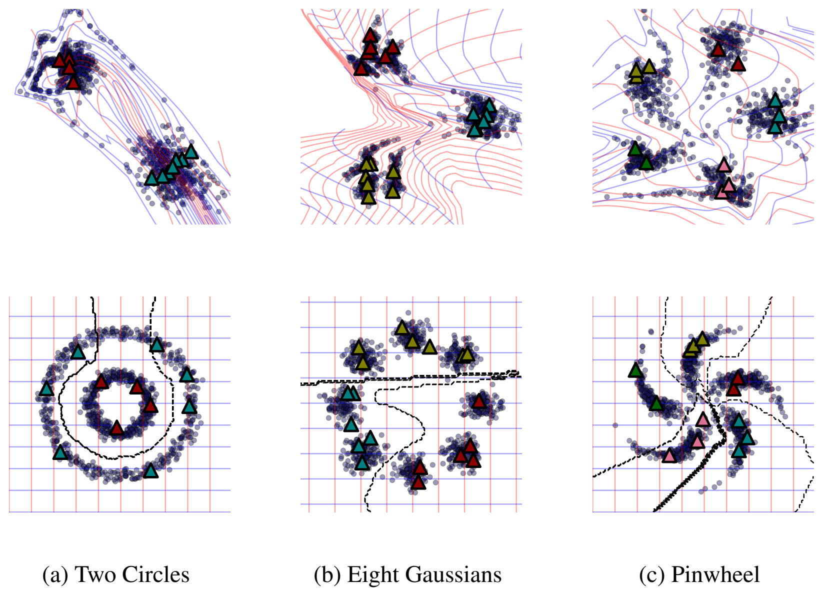
(a) Two Circles (b) Eight Gaussians (c) Pinwheel

Figure 2. Illustration of FlowGMM performance on synthetic datasets. Bottom row shows unlabeled data with blue dots and labeled data with colored triangles; decision boundary is shown with a dashed line. Top row shows the mapping of the data to the latent space.

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

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

over the parameters θ of the bijective function f, which learns both a classifier and a density model. Given a test point x, the model predictive distribution is given by

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

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

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

3.2. Consistency Regularization

Most of the existing state-of-the-art approaches to semi-supervised learning are based on consistency regularization (Laine and Aila, 2016; Miyato et al., 2018b; Tarvainen and Valpola, 2017; Athiwaratkun et al., 2019; Verma et al., 2019). These methods penalize change 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,

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

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 simple consistency regularization term for FlowGMM. Let y'' be the label predicted on image x'' by FlowGMM according to (5). We then define the consistency loss term as the likelihood of the label y'' given the input x'

$$L_{\text{cons}}(x', x'') = \mathcal{N}(f(x')|\mu_{y''}, \Sigma_{y''}). \quad (7)$$

This encourages the model to map small perturbations of the same unlabeled inputs to the same components of the mixture distribution, but not necessarily the same point in contrast to 6. We find that the usual consistency loss degrades sample quality, and this is likely because it conflicts with the models need to preserve intra-class variation like translations. We refer to FlowGMM with consistency term (7) 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).

3.3. Latent Distribution Mean and Covariance Choices

While we can learn both the means μ_i and covariance matrices Σ_i with either directly optimizing likelihood (4), or expectation maximization (see Section B), in practice we didn't find it to help. In our experiments, we set the covariance matrices to identity $\Sigma_i = I$ for all classes. We draw mean vectors μ_i randomly from the standard normal distribution $\mu_i \sim \mathcal{N}(0, I)$ in the latent $\mathcal{Z}$ space and fix them throughout training.

Table 1. Supervised and semi-supervised performance of the proposed model, VAE model (M1+M2 VAE, Kingma et al., 2014) deep invertible generalized linear model (DIGLM, Nalisnick et al., 2019) on MNIST, CIFAR-10 and SVHN. FlowGMM outperforms DIGLM both on MNIST and SVHN. Further, FlowGMM-cons improves upon FlowGMM on all datasets.

Method	MNIST	SVHN	CIFAR-10
	$(n_l = 1k, n_u = 59k)$	$(n_l = 1k, n_u = 72k)$	$(n_l = 4k, n_u = 46k)$
DIGLM Sup ($n_l + n_u$ labels)	99.33	95.74	-
FlowGMM Sup ($n_l + n_u$ labels)	99.63	95.81	88.44
M1+M2 VAE SSL (n_l labels)	97.60	63.98	-
DIGLM SSL (n_l labels)	97.79	-	-
FlowGMM Sup (n_l labels)	97.36	78.26	73.13
FlowGMM (n_l labels)	98.94	82.42	78.24
FlowGMM-cons (n_l labels)	**99.0**	**86.44**	**80.9**

4. Experiments

In this section we present results for FlowGMM on supervised and semi-supervised classification problems. We present a thorough empirical analysis of the model in Section D.

4.1. Synthetic data

We first apply the proposed method to a range of synthetic datasets. We use RealNVP architecture with 6 coupling layers, defined by fully-connected shift and scale networks with 2 hidden layers of size 256 each. The results are shown in Figure 2. The bottom row of the Figure shows the data distribution, labeled data points and the decision boundary. As we can see, even using a small number of labeled data points, FlowGMM puts the decision boundary to a low-density region in the data-space $\mathcal{X}$. On the *two circles* dataset the method struggles; it is generally hard to represent this dataset as an invertible mapping of two Gaussians, as they have different topology. FlowGMM still produces a reasonable decision boundary on this dataset. The top row of the Figure visualizes the mapping to the latent space $\mathcal{Z}$ learned by the flow.

4.2. 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. Please refer to Section C for details on hyper-parameters.

We present the results for FlowGMM and FlowGMM-cons in Table 1. We also provide results for DIGLM (Nalisnick et al., 2019) (which only report semi-supervised performance on MNIST and supervised performance on MNIST and SVHN) and M1+M2 model (Kingma et al., 2014). FlowGMM outperforms both DIGLM and M1+M2 models. Further, FlowGMM-cons improves over FlowGMM on all three datasets, suggesting that consistency regularization is crucial for the proposed model.

We note that the results presented in this section are not competitive with state-of-the-art methods using GANs or consistency regularization (see e.g. Athiwaratkun et al., 2019; Berthelot et al., 2019; Dai et al., 2017); however, the architecture we employ is much less powerful for classification than the ConvNet and ResNet architectures that have been designed for classifcation without the constraint of invertibility. We believe that invertible architectures with better inductive biases for classification (possibly like iResNet (Behrmann et al., 2018)) may help bridge this gap.

EM algorithm Using EM algorithm for optimization (see Appendix B) in the semi-supervised setting on MNIST 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 D.2) which affects the M-step objective and hinders training. We expect mitigating this problem to result in EM achieving faster convergence or improved performance.

5. Conclusion

We proposed FlowGMM, a simple and natural model for semi-supervised learning with normalizing flows. We view our FlowGMM as a new approach for joint classification and density modelling that has a distinct set of strengths and weaknesses compared to other methods. However, the classification performance is not yet competitive with the state-of-the-art approaches (Athiwaratkun et al., 2019; Verma et al., 2019). There are several promising directions that could further improve the performance of FlowGMM. Other types of consistency regularization similar to those proposed in (Miyato et al., 2018b; Verma et al., 2019) could lead to stronger performance. Furthermore, in all experiments we used the RealNVP (Dinh et al., 2016) architecture for the flow. Switching to GLOW (Kingma and Dhariwal, 2018) or an architecture that is designed for classification rather than image generation, such as iResNet (Behrmann et al., 2018), could lead to even better results.

References

Andrei Atanov, Alexandra Volokhova, Arsenii Ashukha, Ivan Sosnovik, and Dmitry Vetrov. Semi-conditional normalizing flows for semi-supervised learning. *arXiv preprint arXiv:1905.00505*, 2019.

Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. 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.

Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. *arXiv preprint arXiv:1811.00995*, 2018.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *Advances in Neural Information Processing Systems 30*, pages 6510–6520. 2017.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Laurent Dinh, Jascha Sohl-Dickstein, Razvan Pascanu, and Hugo Larochelle. A rad approach to deep mixture models. *arXiv preprint arXiv:1903.07714*, 2019.

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

Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. *CoRR*, abs/1706.04599, 2017. URL http://arxiv.org/abs/1706.04599.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

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

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

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

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

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

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018a.

Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018b.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. *arXiv preprint arXiv:1902.02767*, 2019.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, pages 3235–3246, 2018.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

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

Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. *arXiv preprint arXiv:1903.03825*, 2019.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational autoencoder for semi-supervised text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

A. Related work

In general, Normalizing flows (see section 2) learn to transform a unimodal Gaussian distribution to the complex distribution in the data space and substantial work has been invested into improving the expressiveness and inductive biases of this mapping (Dinh et al., 2014; 2016; Kingma and Dhariwal, 2018; Grathwohl et al., 2018; Behrmann et al., 2018). Some normalizing flow papers (such as RealNVP (Dinh et al., 2016)) have used class-conditional sampling, where the transformation is conditioned on the class label. To do so, they pass the class label as an input to coupling layers, conditioning the output of the flow on the class.

The recent work (Dinh et al., 2019) proposed modelling discrete latent variables by layering modules which fold the input for improved unsupervised modeling of discontinuities in data, and apply this method on toy datasets. 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 (5), rather than having 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.

Atanov et al. (2019) simultaneously and independently proposed a semi-supervised learning method based on normalizing flows. Their approach is different in that they focus on using another class-conditional normalizing flow as the latent space distribution $p_{\mathcal{Z}}(\cdot|y)$ while we use simpler Gaussian mixture models.

B. Expectation Maximization

As an alternative to direct optimization of the likelihood (4), we consider Expectation-Maximization algorithm (EM algorithm). 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: $\mathbb{E}_q \log P(X, T|\theta) \rightarrow \max_\theta$. The algorithm can be easily adapted to the semi-supervised setting where a subset of data is labelled 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 labelled data points.

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

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

$$\frac{\mathcal{N}(f(x)|\mu_t, \Sigma_t)}{\sum_{k=1}^{\mathcal{C}} \mathcal{N}(f(x)|\mu_k, \Sigma_k)}$$

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.

C. Hyper-parameters

In image classification experiments, we use the CIFAR-10 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 and 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 the consistency loss term (7), we linearly increase the weight from 0 to 1 for the first 100 epochs following (Athiwaratkun et al., 2019). We use a batch size of 64 and sample 32 labeled and 32 unlabeled data points in each mini-batch. For FlowGMM and FlowGMM-cons, we re-weight the loss on labeled data by $\lambda = 3$ (value tuned on validation (Kingma et al., 2014) on CIFAR-10), as otherwise, we observed that the method underfits the labeled data. The supervised model is trained using the same loss (4), where all the data points are labeled ($n_u = 0$). For FlowGMM and FlowGMM-cons on all the datasets, the mixture components are taken to be unit variance normal distributions with means randomly sampled from the standard normal distribution.

Table 2. Semi-supervised performance of FlowGMM-cons and VAE M1 + M2 model (Kingma et al., 2014) on MNIST for different number of labeled data points. For FlowGMM-cons we report mean and std of test accuracy over 3 runs with different random split of labeled and unlabeled data.

Method	$n_l = 600$	$n_l = 1000$	$n_l = 3000$
M1+M2 VAE SSL (n_l labels)	97.41 ± 0.05	97.60 ± 0.02	97.82 ± 0.04
FlowGMM-cons (n_l labels)	98.1 ± 0.5	99.05 ± 0.1	99.3 ± 0.04

Table 3. Uncertainty calibration for FlowGMM trained on MNIST (1000 objects) and CIFAR-10 in the supervised setting.

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

D. Empirical Analysis

D.1. Dependence on the number of labeled data

Following (Oliver et al., 2018), we evaluate FlowGMM-cons with different number of labeled data. Specifically, we follow the setup of (Kingma et al., 2014) and train FlowGMM-cons on MNIST with 600, 1000 and 3000 labeled data points. We present results in Table 2. FlowGMM-cons outperforms the M1+M2 model of (Kingma et al., 2014) in all the considered settings.

D.2. Uncertainty Representation

For many machine learning applications, it is important to provide well-calibrated predictions. Using FlowGMM for classification task, we would like

$$p(y|x) = \frac{\mathcal{N}(x|\mu_y, \Sigma_y)}{\sum_k \mathcal{N}(x|\mu_k, \Sigma_k)}$$

to represent the probabilities of belonging to a particular class. However, we observe in the experiments that the model performs hard instead of soft label assignment ($p(y|x) = 1$ for exactly one class), and the model is overconfident in its predictions. This occurs due to the high dimensionality of the problem and the way Gaussian mixture parameters are initialized (section 3.3).

We address this problem using temperature scaling (Guo et al. (2017)) which is a simple and popular method for uncertainty calibration. We introduce temperature parameter T and scale all logits $\log \mathcal{N}(x|\mu_k, \Sigma_k)/T$, which corresponds to increasing Gaussian variances by a factor of T. We test calibration on two supervised learning tasks on MNIST dataset with 1000 points (since on the full data set, the model has very strong performance and makes too few mistakes for calibration testing) and CIFAR-10 (the models from section 4.2). For MNIST model, the temperature was tuned on a validation set of 1000 labels from the train set and calibration was tested on the full test set. For CIFAR-10, the test set was split into validation (of size 1000) and test, the temperature was tuned on validation, and calibration was tested on the remaining 9000 objects from the test set. We report negative log likelihood and expected calibration error (see (Guo et al., 2017) for the metric description). The results are presented in Table 3, and we can see that temperature scaling significantly improves both metrics and mitigates overconfidence problem which suggests that the latent space distances learned by the flow model are indeed related to the probabilities of belonging to a class. Since temperature scaling is directly related to variance parameters of the mixture, we hypothesize that variances can be learned in a principled way to directly obtain well-calibrated predictions which we leave as a future work direction.

D.3. Out-of-domain data detection

Density models have shown 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 CIFAR10 (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,

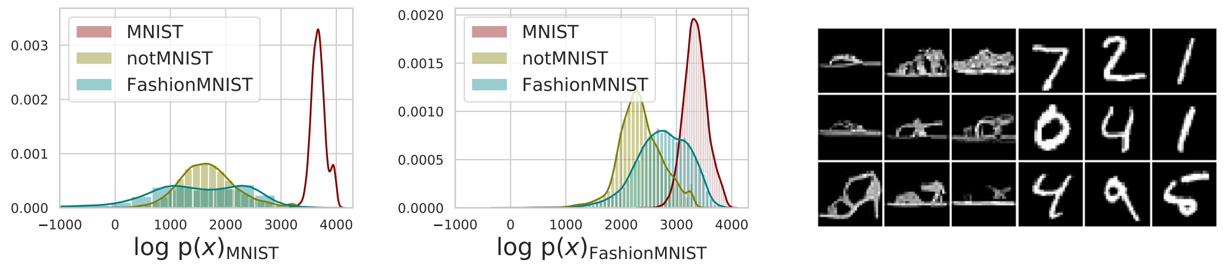

Figure 3. **Left:** Log-likelihoods on in- and out-of-domain data for our model trained on MNIST. **Center:** Log-likelihoods on in- and out-of-domain data for our model trained on FashionMNIST. **Right:** MNIST digits get mapped onto the sandal mode of the FashionMNIST model 75% of the time, often being assigned higher likelihood than elements of the original sandal class. Representative elements are shown above.

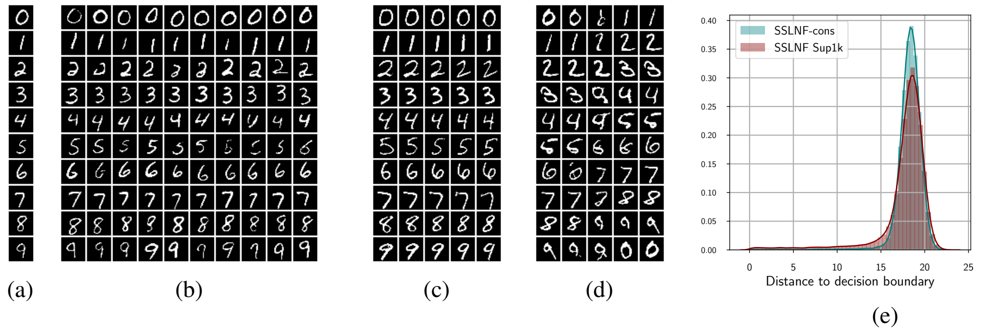

Figure 4. Visualizations of the latent space representations learned by supervised FlowGMM on MNIST. **(a)**: Images corresponding to means of the Gaussians corresponding to different classes. **(b)**: Class-conditional samples from the model at a reduced temperature $T = 0.25$. **(c)**: Latent space interpolations between test images from the same class and **(d)**: from different classes. We interpolate between the original images in leftmost and rightmost positions in each of the rows. Observe that interpolations between objects from different classes pass through low-density regions. **(e)**: 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.

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 $\log p_{\mathcal{X}}(\cdot) = \log p_{\mathcal{Z}}(f(\cdot)) + \log \left|\frac{\partial f}{\partial x}\right|$ on these datasets is shown in Figure 3. For the model trained on MNIST we see that the data from Fashion MNIST and NotMNIST is asigned 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.

D.4. Learned Latent Representations

We next analyze the latent representations learned by FlowGMM-cons. We train the model on MNIST with $1k$ labeled data, same as in section 4.2. In Figure 4a we show the images $f^{-1}(\mu_i)$ corresponding to the means of the Gaussians representing each of the classes. We see that the flow correctly learns to map the means to samples from the corresponding classes. Next, in Figure 4b 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 4b. As we can see, FlowGMM can produce reasonable class-conditional samples simultaneously with achieving a high classification accuracy (99.63%) on the MNIST dataset.

Next, we study interpolations in the latent space for datapoints corresponding to the same class in Figure 4c and different classes in Figure 4d. More precisely, for a pair of images x_1, x_2 we visualize $x(t) = f^{-1}(t \cdot f(x_1) + (1-t) \cdot f(x_2))$. We observe that, as expected, interpolations between objects from different classes typically pass

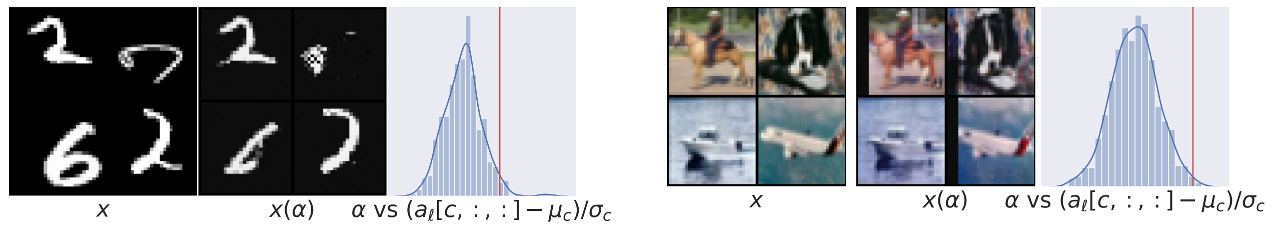

Figure 5. Feature visualization for FlowGMM trained on MNIST (left) and CIFAR10 (right). Each panel shows an original set of four images from the test set, the reconstructions where we have perturbed intermediate level features across spatial locations, and the value of the perturbation α in red vs the whitened distribution of the channel activations. Notice that the channel visualized on the right activates on zeroed out pixels to the left of the image mimicking the random crops applied to the training data, since the model is also a normalizing flow it must preserve information even if not relevant to the classification objective like the random crops.

through low-density regions in the data space, as in many cases samples corresponding to $t \approx 0.5$ are of low fidelity. At the same time, the interpolations between data from the same class appear to stay in high-density region. These observations suggest that, as expected, the model learns to put the decision boundary in the low-density region of the data space.

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 4e. 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. The increased separation from the decision boundary for the unlabeled data suggests that FlowGMM-cons indeed pushes the decision boundary away from the data distribution in agreement with the clustering principle.

D.5. Feature Visualization

Feature visualization has become an important tool for increasing the interpretability of neural networks. 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 and Vedaldi (2015) explore elements in the inverse set of a given set of intermediate features by optimizing over the reconstruction error with a Total Variation regularizer. These methods, while effective, are sensitive to optimization and regularization hyper-parameters and iterative optimization is impractical for real time interactive exploration.

Since our classification model uses a flow which is a sequence of invertible transformations $f(x) = f_{:L}(x) := f_L \circ f_{L-1} \circ \dots f_1(x)$, 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 $a_\ell[c,i,j] = f_{:\ell}(x)_{cij}$ layer with channels c and spatial extent i,j, we may perturb a single neuron with

$$x(\alpha) = f_{:\ell}^{-1}(f_{:\ell}(x) + \alpha\sigma_c\delta_{cij}), \tag{8}$$

where δ_{cij} is a one hot vector at channel c and spatial position i,j; and σ_c is the standard deviation of the activations in channel c over the the training set and spatial locations. Or we can activate a Gaussian centered on i,j: $\delta_{cij} \to \delta_c G_{ij}$, or an entire channel with $\delta_{cij} \to \delta_c$. This can be performed at real time rates to explore the activation parametrized by α and the location cij without any optimization or hyper-parameters. The feature visualization of intermediate layers on test images from MNIST and CIFAR10 are shown in appendix Figure 5.