Semi-Conditional Normalizing Flows for Semi-Supervised Learning

Andrei Atanov [1 2] **Alexandra Volokhova** [3 4] **Arsenii Ashukha** [5] **Ivan Sosnovik** [6] **Dmitry Vetrov** [5 1]

Abstract

This paper proposes a semi-conditional normalizing flow model for semi-supervised learning. The model uses both labelled and unlabeled data to learn an explicit model of joint distribution over objects and labels. Semi-conditional architecture of the model allows us to efficiently compute a value and gradients of the marginal likelihood for unlabeled objects. The conditional part of the model is based on a proposed *conditional coupling layer*. We demonstrate a performance of the model for semi-supervised classification problem on different datasets. The model outperforms the baseline approach based on variational auto-encoders on MNIST dataset.

1. Introduction

Modern supervised machine learning approaches require large amount of labelled data. However, large labelled datasets are not always available and cannot be collected without additional effort. In many fields, on the other hand, unlabelled data is accessible. This makes semi-supervised learning an important field of machine learning research. Semi-supervised methods aim to employ unlabelled data to achieve a competitive performance of supervised models while using only a few labelled examples. Previous work on this topic includes Joachims (1999); Blum et al. (2004); Rosenberg et al. (2005). These methods, however, poorly scale on huge unlabelled datasets and high dimensional data Zhu (2005); Kingma et al. (2014).

Recent advances in unsupervised learning, specifically deep generative models (Kingma & Welling, 2013; Goodfellow et al., 2014; Dinh et al., 2014; Van den Oord et al., 2016), allow to model complex data distribution $p_\theta(x)$. Modern methods for semi-supervised learning (Kingma et al., 2014; Springenberg, 2015; Salimans et al., 2016) actively employ deep generative models to learn from unlabeled data. The common approach suggests to model *joint* distribution $p_\theta(x, y) = p_\theta(x|y)p(y)$ over objects x and labels y, where $p_\theta(x|y)$ is a conditional generative model. The joint distribution allows inferring class-label distribution $p_\theta(y|x)$ that is used to make a prediction for a supervised problem.

Kingma et al. (2014) proposed a semi-supervised learning method that is based on variational auto-encoders (Kingma & Welling, 2013). This approach scales well on large datasets and provides flexible generative models for high-dimensional data. However, direct likelihood optimization in this case is intractable, and it is usually tackled by methods for approximate variational inference (Paisley et al., 2012; Hoffman et al., 2013; Ranganath et al., 2014).

Recently a new family of deep generative models called normalizing flows has been proposed (Rezende & Mohamed, 2015). Normalizing flows model data by a deterministic invertible transformation of a simple random variable, e.g. standard normal. The framework provides a tractable likelihood function and allows its direct optimization. Data distribution in this case can be inferred using change of variables formula by computing Jacobian determinant of the transformation. Recent advances in this field (Kingma et al., 2016; Dinh et al., 2016; Kingma & Dhariwal, 2018; Grathwohl et al., 2018; Berg et al., 2018) provide a number of flexible architectures with tractable and efficient determinant computation.

It has been shown that dimension reduction is crucial for semi-supervised methods (Kingma et al., 2014). Unfortunately, normalizing flows have a latent representations of the same dimension as input data. However, it was shown that a multi-scale architecture (Dinh et al., 2016) produces latent representation, in which only a small part of components stores consistent semantic information about input data. That opens a possibility to use this kind of architecture for dimension reduction.

In this paper, we propose *Semi-Conditional Normalizing*

[*]Equal contribution [1]Samsung-HSE Laboratory, National Research University Higher School of Economics [2]Skolkovo Institute of Science and Technology [3]Moscow Institute of Physics and Technology [4]Yandex School of Data Analysis [5]Samsung AI Center Moscow [6]UvA-Bosch Delta Lab, University of Amsterdam, Netherlands. Correspondence to: Andrei Atanov <ai.atanow@gmail.com>, Alexandra Volokhova <s-volohova@yandex.ru>, Arsenii Ashukha <ars.ashuha@gmail.com>.

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

Flow — a new normalizing flow model that is suitable for semi-supervised classification problem. The proposed model provides a tractable likelihood estimation for both labeled and unlabeled data. The semi-conditional architecture (Section 3.3) allows us to effectively compute value and gradients of a marginal likelihood $p_\theta(x)$. The conditional distribution $p_\theta(x|y)$ is modelled by a proposed *conditional coupling layer*, that is built on the top of the original coupling layer (Dinh et al., 2016). For dimension reduction, we adapt a multi-scale architecture, and we find it to be a crucial component of our model. In experiments, we demonstrate the empirical performance of the model for semi-supervised classification problem. We also find that the proposed model incorporates a data obfuscation mechanism by design, that may be useful for semi-supervised fair learning.

2. Semi-Supervised Learning with Deep Generative Models

We consider semi-supervised classification problem with partly labelled dataset. Let us denote an object as $x_i \in \mathbb{R}^d$ and a corresponding class label as $y_i \in \{1, \dots, K\}$. We denote the set of labelled and unlabelled objects as $\mathcal{L}$ and $\mathcal{U}$ respectively. Generative model with parameters θ defines joint likelihood $p_\theta(x, y)$ of an object and its label. The objective function for this model is a log-likelihood of the dataset:

$$L(\theta) = \sum_{(x_i,y_i)\in\mathcal{L}} \log p_\theta(x_i, y_i) + \sum_{x_j\in\mathcal{U}} \log p_\theta(x_j), \quad (1)$$

where $p_\theta(x_j) = \sum_{k=1}^{K} p_\theta(x_j, y{=}k)$ is a marginal likelihood for unlabelled object. The joint likelihood $p_\theta(x, y)$ is often parameterized as $p_\theta(x, y) = p_\theta(x|y)p(y)$, where a prior $p(y)$ is a uniform categorical distribution and a conditional distribution $p_\theta(x|y)$ is defined by a conditional generative model. On the test stage, the prediction of the model – a posterior distribution over labels $p_\theta(y|x)$ – can be found as $p_\theta(y|x) = \frac{p_\theta(x,y)}{p_\theta(x)}$.

Kingma et al. (2014) proposed to use variational autoencoders (Kingma & Welling, 2013) as the conditional generative model $p_\theta(x|y)$. However, such parametrization, does not allow to maximize the objective (1) directly and obliges to use stochastic optimization of a variational lower bound, that does not guarantee convergence to the maximum of (1). Additionally, the posterior predictive distribution $p_\theta(y|x)$ cannot be computed in a closed form and needs a time-consuming sample-based estimation.

We present a new way to model the conditional distribution $p_\theta(x|y)$ using normalizing flows (Rezende & Mohamed, 2015). In our approach we have access to all required distributions to maximize log-likelihood of the data (1) directly and to compute exact posterior $p_\theta(y|x)$ see (Section 3.3).

3. Semi-Conditional Normalizing Flows

3.1. Normalizing Flows

Normalizing flows model data as a deterministic and invertible transformation $x = g_\theta(z)$ of a simple random variable z, e.g. standard normal. We denote an inverse transformation g_θ^{-1} as f_θ, i.e. this inverse function maps data x to latent representation z. Log-density of this model can be calculated using change of variable formula:

$$\log p_\theta(x) = \log \left| \frac{\partial f_\theta(x)}{\partial x^T} \right| + \log p(z), \quad (2)$$

where $\frac{\partial f_\theta(x)}{\partial x^T}$ is a Jacobian of the transformation f_θ and prior $p(z)$ is a standard normal distribution $\mathcal{N}(z|0, I)$. The transformation f_θ has to be bijective and has a tractable Jacobian determinant. A number of flow architectures (Dinh et al., 2014; Kingma et al., 2016; Kingma & Dhariwal, 2018) have been proposed recently. These architectures are based on operations with simple form determinant of Jacobian. For example, Real NVP architecture proposed by Dinh et al. (2016) is based on an affine coupling layer. This layer splits an input variable x into two non-overlapping parts x_1, x_2 and applies affine transformation based on the first part x_1 to the other x_2 in the following way:

$$z_1 = x_1, \quad z_2 = x_2 \odot \exp(s(x_1)) + t(x_1),$$

where s, t are arbitrary neural networks. This transformation has a simple triangular Jacobian, and the use of neural networks allows to model complex non-linear dependencies.

3.2. Dimension Reduction with Normalizing Flows

It has been shown that performance of a semi-supervised model benefits from using low-dimensional data representation (Kingma et al., 2014). We also observe this effect in our experiments (Section 4.3). Unfortunately, normalizing flows preserve a dimensionality of z equal to x one. However, modern normalizing flow models have multiscale structure, where differnet parts of the latent representation z pass different number of transformations. The part that passes all transformations contains most of semantic information about an input object (Dinh et al., 2016), we will refer to this part as z_f and to the other components as z_{aux}.

3.3. Semi-Conditional Normalizing Flow

Semi-supervised methods based on generative models require conditional density estimation (Section 2). In order to model conditional distribution $p_\theta(x|y)$ rather then just $p_\theta(x)$ we have to condition the transformation f_θ on y. Perhaps, the simplest way would be to use different flows for each class (Trippe & Turner, 2018). Nevertheless, this approach has high memory cost and does not share weights between classes. In (Kingma et al., 2016) authors proposed

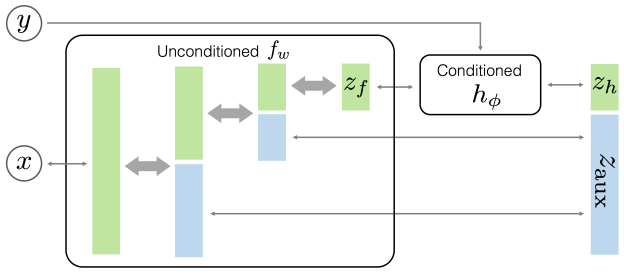


Figure 1: The proposed semi-conditional architecture consists of two parts: a large unconditional flow $f_w(x)$, and a relatively small conditional flow $h_\phi(z_f; y)$. The unconditional flow $f_w(x)$ is based on a multi-scale architecture and maps an input x into a low-dimensional z_f and an auxiliary vector z_{aux}. The conditional flow $h_\phi(z_f; y)$ maps the low-dimensional vector z_f to $z_h = h_\phi(z_f; y)$. The architecture allows to compute $p_\theta(x) = \mathbb{E}_y p_\theta(x, y)$ with a single forward pass of the computationally expensive flow f_θ and one pass of the inexpensive flow h_ϕ for every class label y.

a memory efficient conditional autoregressive architecture. Unfortunately, sequential structure of autoregressive models leads to high computational cost. Therefore, we adapt a more memory and computationally efficient affine coupling layer and propose a *conditional affine coupling layer* defined as follows:

$$z_1 = x_1, \quad z_2 = x_2 \odot \exp(s(x_1, y)) + t(x_1, y),$$

where neural networks s, t have a class variable as an additional input. This allows to model complex dependencies on the class variable and at the same time keeps determinant of Jacobian easy to compute.

Semi-Conditional Architecture During the optimisation of the objective (1), we need to evaluate marginal log-likelihood for unlabelled data $p_\theta(x) = \sum_{y=1}^{K} p_\theta(x|y)p(y)$. However, this requires K forward passes. We address this issue with a proposed *semi-conditional* architecture where only a small number of deep layers are conditioned on y. First, we map x to z_f and z_{aux} with an unconditional flow $f_w(x)$ and then map the deepest components z_f to z_h with a conditional $h_\phi(z_f; y)$. In this case, the Jacobian of the unconditional flow f_w is independent of y, and we can pull it out of the sum and compute only once for all classes:

$$\log p_\theta(x) = \log \left| \frac{\partial f_w(x)}{\partial x^T} \right| + \log \mathcal{N}(z_{\text{aux}}|0, I) \qquad (3)$$
$$+ \sum_{y=1}^{K} \log \left| \frac{\partial h_\phi(z_f; y)}{\partial z_f^T} \right| + \log \mathcal{N}(z_h|0, I) + \log p(y)$$

Note, that we pass only the deepest components z_f to the conditional flow h_ϕ, instead of the hole vector $[z_f, z_{\text{aux}}]$. In our experiments (Section 4.3) we found it to be an essential part of our model.

f_w	h_ϕ	Moons		Circles	
		Error,%	NLL	Error,%	NLL
GLOW	GLOW	0.6 ± 0.4	1.11 ± 0.02	5.0 ± 1.9	1.68 ± 0.13
	GMM	1.2 ± 1.2	1.15 ± 0.01	14.2 ± 9.0	1.28 ± 0.15
FFJORD	GLOW	0.3 ± 0.2	1.12 ± 0.03	6.2 ± 2.2	1.7 ± 0.2
	GMM	5.3 ± 6.3	1.15 ± 0.06	25 ± 2	0.97 ± 0.05

Table 1: Test accuracy and negative log-likelihood (NLL) for different models on two toy datasets (see visualization at Appendix. D). f_θ corresponds to the unconditional part and h_ϕ to the conditional part of SCNF model (Fig. 1). GMM stands for Gaussian mixture model, and Glow is a flow-based model. Glow conditional flow h_ϕ in terms of test error outperforms GMM for different types of unconditional flow on both datasets. Taking into account values of uncertainties, we see that unconditional FFJORD gives roughly the same test error and negative log-likelihood as unconditional Glow.

3.4. Learning of Semi-Conditional Normalizing Flows

The parameters $\theta = \{w, \phi\}$ of the model (4) are estimated via maximum likelihood approach (1). Normalizing flows provide us with tractable log-likelihood function $\log p_\theta(x, y)$ (4) along with marginal log-likelihood $\log p_\theta(x)$. Therefore, we can compute a gradient $\nabla_\theta \mathcal{L}(\theta)$ of the objective (1) and use a stochastic gradient optimization.

$$\log p_\theta(x, y) = \log \left| \frac{\partial f_w(x)}{\partial x^T} \right| + \log \left| \frac{\partial h_\phi(z_f; y)}{\partial z_f^T} \right| \qquad (4)$$
$$+ \log \mathcal{N}(z_h|0, I) + \log \mathcal{N}(z_{\text{aux}}|0, I) + \log p(y)$$

Connection to NF with Learnable Prior. We can treat the second normalizing flow h_ϕ as a conditional prior distribution for the first unconditional flow f_w. A simple example of such a learnable prior is a Gaussian mixture model (GMM) where each of mixture components is a Gaussian distribution $\mathcal{N}(z_f|\mu_y, \Sigma_y)$ corresponded to one of the class label y. We compare a performance of the proposed conditional normalizing flow and the Gaussian model as the conditional part (Section 4). A gold standard to find parameters of GMM is an expectation maximization algorithm (MacKay, 2003). We also adapt this algorithm for our model (see Appendix B) and compare it with direct optimization (Section 4.2). We, however, did not find a significant difference between them.

4. Experiments

4.1. Toy Semi-Supervised Classification

We train proposed Semi-Conditional Normalizing Flow on toy 2-dimensional problems: moons and concentric circles. For each problem the training dataset consists of 1000 objects and only 10 of them are labeled. We consider Gaussian mixture model and Glow as the conditional flow h_ϕ. For the unconditional flow f_θ we take Glow and recently proposed FFJORD (Grathwohl et al., 2018) models. We do

Model	Optimisation	L_{clf}	Error, %	Bits/dim
Kingma et al. (2014)	VI	✓	3.3 ± 0.1	-
SCNF-GLOW (Ours)	SGD	✗	1.9 ± 0.3	1.145 ± 0.004
	EM-SGD	✓	2.0 ± 0.1	1.151 ± 0.010
		✗	1.9 ± 0.0	1.146 ± 0.002
SCNF-GMM (Ours)	SGD	✗	14.2 ± 2.4	1.143 ± 0.011
	EM-SGD	✓	16.9 ± 5.3	1.141 ± 0.006
		✗	13.4 ± 2.8	1.145 ± 0.005

Table 2: Test error and bits per dimension on MNIST dataset (lower better). We use 100 labelled objects to train the models (averaging done over 3 different splits). SCNF stands for Semi-Conditional Normalizing Flows with the same unconditional flow and different conditional parts. SGD is a direct gradient optimisation of the objective, EM-SGD is an expectation maximisation algorithm, and VI is a variational inference. L_{clf} is an additional classification loss. We found that the proposed SCNF-GLOW model outperforms VAE-based approach (Kingma et al., 2014).

not use multiscale architecture with dimension reduction as we have only 2-dimensional input. We observe that models with Glow conditioning archive lower test error in comparison with Gaussian Mixture Model conditioning. To make the fair comparison, we use roughly the same number of parameters for each SCNF model. Quantitative results can be seen at Tab. 1 and visualization at Appendix. D.

4.2. Semi-Supervised Classification on MNIST

We demonstrate performance of the proposed model on MNIST dataset (LeCun et al., 1998). The standard protocol (Kingma et al., 2014) was used to model semi-supervised setting. We split the training set of size 60,000 into labelled and unlabelled parts. The size of labelled part equals to 100. The algorithm performance was averaged on 3 different random splits of the dataset. We use Glow architecture (Kingma & Dhariwal, 2018) for an unconditional part (see Appendix E). We reduce the size of an input with a multi-scale architecture from 784 for x to 196 for z_f. For a conditional part h_ϕ we compare Glow architecture with conditional affine coupling layers (SCNF-GLOW) and Gaussian mixture model (SCNF-GMM). We compare our Semi-Conditional Normalizing Flow (SCNF) model with VAE-based approach (Kingma et al., 2014), which also uses architecture with dimension reduction.

The results can be seen at Tab. 2. We observe that the proposed SCNF model with Glow-based conditional part outperforms VAE-based model. The GMM conditioning seems to be not sufficiently expressive for this problem and shows much poorer performance. We also did not find any difference between the performance of SGD and EM-SGD optimization algorithms.

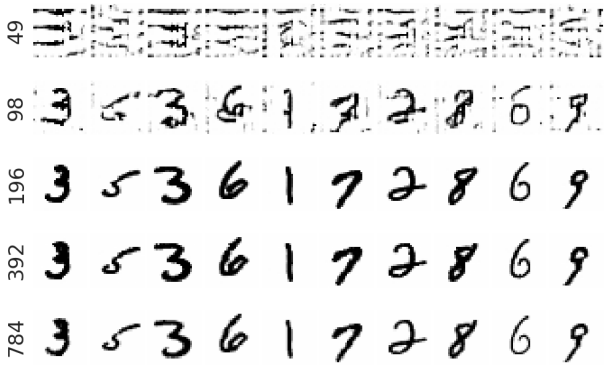


Figure 2: Reconstructions of images using different number of deepest hidden components z_f. From top to bottom: 49, 98, 196, 392, 784. The latter one corresponds to real images as this is the hole representation and the architecture is invertible. We zeroed the auxiliary components z_{aux} when perform the reconstruction. We found that the deepest 196 components provide quite accurate reconstructions.

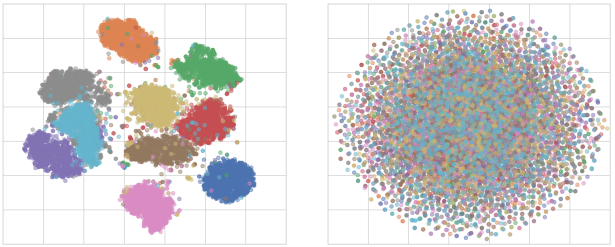
Dimension	Test Error, %	Train Error, %
48	2.6	0
98	2.0	0
196	**1.9**	0
392	61.4	0
784	91.1	0

Table 3: Test and train erros on MNIST dataset for different dimensions the deepest components z_f that we pass to conditional model $h_\phi(z_f; y)$ (see Fig.1).

4.3. Dimension Reduction

Training classification model in a high-dimensional space with only a few labelled examples may lead to overfitting. In Kingma et al. (2014) authors showed that semi-supervised classification methods benefits from using low-dimensional representation of objects. In Section 3.2 we proposed a natural dimension reduction technique for our model. In this Section we examined an impact of the proposed technique.

We use the SCNF-GLOW architecture from the previous experiment (Section 4.2) and vary a dimension of the latent representation z_f. The results can bee seen at Tab. 3. We find that with the dimension growth the model is prone to overfitting and results in nearly random guess test performance. To demonstrate an information that remains in the latent representation we reconstruct an image using z_f and zeroing the auxiliary components z_{aux}. The corresponding reconstructions can be found at Fig. 2. Surprisingly, reconstructions from 49 deepest components do not look like original images, while conditional flow is still able to achieve low test error on test set.



(a) t-SNE Embeddings of z_f (b) t-SNE Embeddings of z_h

Figure 3: We embed low-dimension representations of MNIST digits into two dimensional space using t-SNE method. z_f are from an unconditional part f_θ and z_h are from a conditional part h_ϕ. Different colors correspond to different classes. We found that embeddings z_f from the unconditional model form clusters for different classes, and embeddings z_h seem to be independent of class variable.

4.4. Semi-Supervised Data Obfuscation

Modern machine learning algorithms have become widely used for decisions that impact real lives. These algorithm have to be non-discriminative and does not rely on such features as gender and race. One way to reduce the discrimination is to construct object representations, that are independent of these discrimination-inducing features. We consider a class label to be such discriminative information, and focus on finding a representation of x that is independent of y. The latent representation z_f contains almost all semantic information about an object x, as x can be quite accurate restored from z_f when its dimension equals to 196 (see Figure 2) and we assume that $x = f_w^{-1}(z_f)$. However, it means that z_f contains information about y and, therefore, does not suitable. On the other hand, we found that the representation z_h from a conditional flow is independent of y, but still contains all the other information about z_f and therefor x, since $z_f = h_\phi^{-1}(z_h; y)$. Therefore, z_h can be used as a non-dicriminative representation of an object x.

The intuition, why latent representations z_h are not dependent of class variable y, can be explained by contradiction with log-likelihood (4) maximization. The likelihood, as a function of h_ϕ consists of two terms: a determinant of Jacobian $\partial h_\phi(z_f; y)/\partial z_f^T$ and a prior density $\mathcal{N}(z_h|0, I)$. The Jacobian term encourage the transformation h_ϕ to increase a volume, while the prior term prevent it from covering the whole space. The conditional flow h_ϕ models different transformations for each class label y. Therefore, it is able to fill the same high-density area in the latent space for objects from different classes and increase the determinant of Jacobian while preserving the prior density the same. This leads to a higher likelihood and reduces the information about class label y.

In order to provide an evidence for this intuition, we trained two fully-connected classifiers of the same size in a space of z_f and z_h. For both classifiers we were able to fit training data perfectly. However, the classifier that was trained with representations z_h has nearly random accuracy on test data, while the other one achieves almost perfect performance. To illustrate that z_f depends on y and z_h does not, we embed these latent representations using t-SNE (Maaten & Hinton, 2008) into 2-dimensional space (Fig. 3). In Appendix A we also explore what type of information contains in z_h.

5. Conclusion & Discussion

This work investigates the idea of using deep generative models for semi-supervised learning, as well as, expands an area of normalizing flows applications. We propose *Semi-Conditional Normalizing Flows* – a new architecture for semi-supervised learning that provides a computationally efficient way to learn from unlabeled data. Although the model does not directly force the low-dimensional representations to be semantically meaningful, we always observe it in practice. This effect is crucial for our model and needs an additional investigation.

In experiments, we show that the model outperforms semi-supervised variational auto-encoders (Kingma et al., 2014). We refer the improvement to end-to-end learning and a tractable likelihood, which optimization does not require to use approximate variational methods. We also show, that the proposed model has a notable effect of data obfuscation (Section 4.4), that is provided by the model design and properties of normalizing flows. This effect can be also used for semi-supervised domain adaptation as has been proposed recently by (Ilse et al., 2019).

On more complex datasets the flows tend to ignore a class label y, that in case of Gaussian Mixture prior is caused by similar mixture components for different classes. As proposed in Appendix D of (Kingma & Dhariwal, 2018), the effect can be cured by using classification loss on the hidden components of the flow. We found that classification loss improves the performance of our model on more complex datasets, however, recent methods based on data augmentation (Berthelot et al., 2019; Xie et al., 2019) tend to show a better performance.

Independently, (Izmailov et al., 2019) also proposed to use normalizing flows with GMM priors for semi-supervised learning.

Acknowledgements

This research is in part based on the work supported by Samsung Research, Samsung Electronics.

References

Berg, R. v. d., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. *arXiv:1803.05649*, 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.

Blum, A., Lafferty, J., Rwebangira, M. R., and Reddy, R. Semi-supervised learning using randomized mincuts. In *ICML*, pp. 13. ACM, 2004.

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

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv:1605.08803*, 2016.

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.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv:1810.01367*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Ilse, M., Tomczak, J. M., Louizos, C., and Welling, M. Diva: Domain invariant variational autoencoders. *arXiv preprint arXiv:1905.10427*, 2019.

Izmailov, P., Kirichenko, P., Finzi, M., and Wilson, A. G. Semi-supervised learning with normalizing flows. *ICML Workshop on Invertible Neural Nets and Normalizing Flows*, 2019.

Joachims, T. Transductive inference for text classification using support vector machines. In *Icml*, volume 99, pp. 200–209, 1999.

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

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *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.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

MacKay, D. J. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Paisley, J., Blei, D., and Jordan, M. Variational bayesian inference with stochastic search. *arXiv:1206.6430*, 2012.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial Intelligence and Statistics*, pp. 814–822, 2014.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *arXiv:1505.05770*, 2015.

Rosenberg, C., Hebert, M., and Schneiderman, H. Semi-supervised self-training of object detection models. 2005.

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.

Springenberg, J. T. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv:1511.06390*, 2015.

Trippe, B. L. and Turner, R. E. Conditional density estimation with bayesian normalising flows. *arXiv:1802.04908*, 2018.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pp. 4790–4798, 2016.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. Unsupervised data augmentation, 2019.

Zhu, X. J. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

Figure 4: Reconstructions from the SCNF-GLOW model with different labels. Top row corresponds to source images, that are encoded into latent space with the conditional flow using true labels. Then, we reconstruct this latent representations using different class-labels. We can see that the latent representation define a style but not a particular class.

A. Conditional Samples from Semi-Conditional Normalizing Flows

The proposed SCNF model, as have been argued in Section 4.4, removes the information about a class label y from the latent representation z_h. To demonstrate what type of information contains in z_h, we choose 10 different digit images and encode them with their true class labels. Then, for each image we fix its latent representation z_h and decode it using different class labels y. We zeroed z_{aux} to not use information from it. The results can be seen at Fig. 4. We found that z_h seems to contain information about form and style of a digit.

B. Expectation Maximization algorithm for Semi-Conditional Normalizing Flow

Parameters of the proposed model are found via maximum likelihood approach (Section 2, 3.4). For labelled data we directly maximize a joint log-likelihood $\log p_\theta(x, y)$ (4). For unlabelled objects we have to maximize a marginal log-likelihood (3):

$$\log p_\theta(x) = \log \sum_{y=1}^{K} p_\theta(x|y)p(y).$$

It can be seen as a mixture model with components $p_\theta(x|y)$ for different y. The class variable y in this case plays a role of a hidden variable. A gold standard for training models with hidden variables is an Expectation Maximization algorithm.

Let us introduce an auxiliary distribution $q(y)$ over the latent variable y and define the following decomposition of the marginal log-likelihood:

$$\log p_\theta(x) = \underbrace{\mathbb{E}_{q(y)} \log p_\theta(x, y)}_{\mathcal{L}(q,\theta)} + \mathrm{KL}(q(y)\|p_\theta(y|x)),$$

where the second term is a Kullback-Leibler divergence between true posterior over y and its approximation $q(y)$. The Expectation Maximization algorithm optimizes the marginal log-likelihood with an iterative process of two subsequent steps:

$$\text{E-step:} \quad q^*(y) = \arg\min_q \ \mathrm{KL}(q(y)\|p_{\theta_{old}}(y|x))$$
$$\text{M-step:} \quad \theta_{new} = \arg\max_\theta \ \mathcal{L}(q^*, \theta).$$

The first step can be done analytically since the KL-divergence is non-negative and equals to 0 iff $q(y) = p_{\theta_{old}}(y|x) = \frac{p_{\theta_{old}}(x,y)}{p_{\theta_{old}}(x)}$. This step ensures that the lower bound is equal to the true log-likelihood $\mathcal{L}(q^*, \theta_{old}) = \log p_\theta(x)$ for the previous parameters θ_{old}. Then, during M-step we optimize the lower bound and, therefor, the marginal likelihood. Since in the proposed model the true posterior is tractable we can use this analytic solution. The optimization problem on the M-step is intractable, unfortunately. However, we can use a gradient method for an approximate optimization. We use one gradient step from previous parameters θ_{old} on the M-step. The final EM algorithm for the proposed model looks as follows:

$$\text{E-step:} \quad q^*(y) = p_{\theta_{old}}(y|x)$$
$$\text{M-step:} \quad \theta_{new} = \theta_{old} + \nabla_\theta \mathcal{L}(q^*, \theta)\big|_{\theta=\theta_{old}}.$$

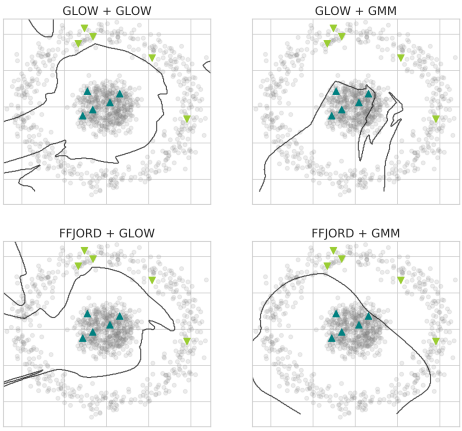
We refer to this algorithm as EM-SGD.

C. Conditional FFJORD

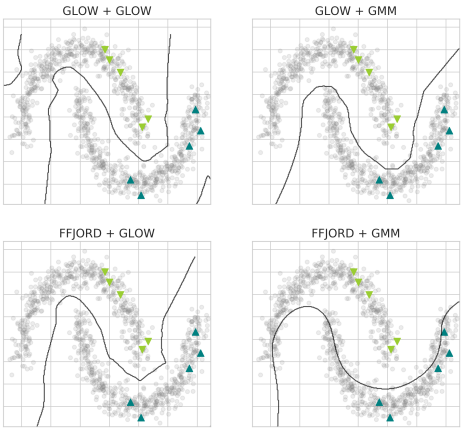
Our model consists of unconditional and conditional flows. We showed how GLOW (Kingma & Dhariwal, 2018) can be efficiently conditioned and used it in experiments (Table 1). Moreover, we propose a natural way to condition FFJORD (Grathwohl et al., 2018). This flow maps objects to latent space by solving the following ordinary differential equation:

$$\frac{dz(t)}{dt} = f_\theta(z(t), t),$$

where $z(t)$ is a trajectory that draws latent representation $z(0)$ to objects $x = z(T)$. Dynamics function f is a neural



(a) Circles



(b) Moons

Figure 5: Visualization of classification curves for different SCNF models on toy datasets. Gray objects are unlabelled and colored are labelled points. Black line corresponds to a decision boundary of the corresponding classification rule. On Circles problem, we found that FFJORD with GMM is prone to overfit labeled data, that leads to poor test accuracy. We also did not find that regularization helps. This effect, however, disappears since we use more labeled data.

network that takes $z(t)$ and t input and returns $\frac{dz(t)}{dt}$. We propose to feed also a class label y into f to make conditional FFJORD. So, we consider to solve the following equation:

$$\frac{dz(t)}{dt} = f_\theta(z(t), t, y)$$

We assume that conditional FFJORD will perform better than conditional GLOW because as it was shown (Grathwohl et al., 2018) FFJORD is more expressive than GLOW. But our current results didn't give significant improvements, so we are going to investigate this problem in the future.

D. Toy Experiments Visualization

We plot decision boundaries for models trained on toy datasets (Section 4.1) at Fig. 5.

E. Normalizing Flow Architecture for MNIST

For experiments on MNIST dataset we use Glow-based architecture for both unconditional and conditional parts. For coupling layers use two types of masks: checkerboard and channel-wise. We use residual architecture (He et al., 2016) with 4 residual blocks with hidden size 64 as a neural network in all coupling layers.

E.1. The Unconditional Flow f_θ

We also transform pixels to logits as have been proposed in (Dinh et al., 2016). The sequence of transformations in the unconditional flow f_θ looks as follows:

```
ToLogits()
InvertibleConv2d(1)
MaskedCouplingLayer(mask=checkerboard)
ActNorm(1)
InvertibleConv2d(1)
MaskedCouplingLayer(mask=checkerboard)
ActNorm(1)
InvertibleConv2d(1)
MaskedCouplingLayer(mask=checkerboard)
ActNorm(1)
SpaceToDepth(2x2)
InvertibleConv2d(4)
CouplingLayer(mask=channel)
ActNorm(4)
InvertibleConv2d(4)
CouplingLayer(mask=channel)
ActNorm(4)
FactorOut([4, 14, 14] -> [2, 14, 14])
InvertibleConv2d(2)
MaskedCouplingLayer(mask=checkerboard)
ActNorm(2)
InvertibleConv2d(2)
MaskedCouplingLayer(mask=checkerboard)
ActNorm(2)
InvertibleConv2d(2)
MaskedCouplingLayer(mask=checkerboard)
ActNorm(2)
SpaceToDepth(2x2)
InvertibleConv2d(8)
CouplingLayer(mask=channel)
ActNorm(8)
InvertibleConv2d(8)
CouplingLayer(mask=channel)
ActNorm(8)
FactorOut([8, 7, 7] -> [4, 7, 7])
InvertibleConv2d(4)
MaskedCouplingLayer(mask=checkerboard)
ActNorm(4)
InvertibleConv2d(4)
MaskedCouplingLayer(mask=checkerboard)
ActNorm(4)
InvertibleConv2d(4)
MaskedCouplingLayer(mask=checkerboard)
ActNorm(4)
InvertibleConv2d(4)
CouplingLayer(mask=channel)
ActNorm(4)
```

```
InvertibleConv2d(4)
CouplingLayer(mask=channel)
ActNorm(4)
```

The deepest 196 components of the output of this model then pass to the conditional flow h_θ.

E.2. The Conditional Flow h_θ

The sequence of transformations in the conditional flow looks as follows:

```
ActNorm(196)
InvertibleConv2d(196)
ConditionalCouplingLayer(mask=channel)
ActNorm(196)
InvertibleConv2d(196)
ConditionalCouplingLayer(mask=channel)
ActNorm(196)
InvertibleConv2d(196)
ConditionalCouplingLayer(mask=channel)
ActNorm(196)
InvertibleConv2d(196)
ConditionalCouplingLayer(mask=channel)
FactorOut([196, 1, 1] -> [98, 1, 1])
ActNorm(98)
InvertibleConv2d(98)
ConditionalCouplingLayer(mask=channel)
ActNorm(98)
InvertibleConv2d(98)
ConditionalCouplingLayer(mask=channel)
ActNorm(98)
InvertibleConv2d(98)
ConditionalCouplingLayer(mask=channel)
ActNorm(98)
InvertibleConv2d(98)
ConditionalCouplingLayer(mask=channel)
```

As a prior we use Gaussian distribution with zero mean and learnable diagonal covariance matrix.