Bayesian inference for semi-supervised learning with normalizing flows

Apoorv Vikram Singh APOORV.SINGH@NYU.EDU
Net ID: avs8673

Phat V Nguyen PVN2005@NYU.EDU
Net ID: pvn2005

Abstract

Normalizing flows work by transforming a latent distribution through an invertible neural network for an approach to generative modelling. Izmailov et al. (2019) proposed FlowGMM, an approach to generative semi-supervised learning using normalizing flows, using a latent Gaussian mixture model. This project builds on top of their work by doing a Bayesian inference for the FlowGMM model using techniques suggested by Wilson & Izmailov (2020). We first show the need for Bayesian inference using synthetic data, and how labels can affect the quality of the solution given by FlowGMM. Then, we show results on real-world text data-sets: AG-News and Yahoo Answers. The results show significant improvement compared to the FlowGMM model, showing that Bayesian model averaging can give much better performance when compared to point estimates.

1. Introduction

Generative Modelling In generative approach to classification it is assumed that the underlying distribution over the data has a specific parametric form and our goal is to estimate the parameters of the model. If we succeed in learning the parameters, we can learn the optimal classifier called the Bayes optimal classifier. It is usually more difficult to learn the underlying distribution than to learn an accurate predictor directly.

Normalizing Flows Neural networks have been successfully been used for generative modelling applications, e.g., GANs (Goodfellow et al., 2014), normalizing flows (Dinh et al., 2017). In this project, we focus on normalizing flows. Normalizing flows work by transforming a simple distribution to build more complicated ones through an invertible function. The invertible function used is a specific type of neural-network called the "invertible neural network". The neural network can be chosen for a specific problems at hand, e.g., if the convolutional neural networks (CNN) provide suitable inductive biases for image data, then one could work with an "invertible CNN". Since the neural network is invertible, one can express the exact likelihood over the data which is used to train the neural network.

Semi-Supervised Learning In semi-supervised learning framework, we are given both labelled and unlabelled data and the aim is to use and the aim is to leverage both types of data to make predictions of class labels on unseen data. Perhaps surprisingly, unlabelled data can help with making better predictions (Göpfert et al., 2019). Intuitively, unlabelled data close to the labelled points help drive better predictions, compared to labelled-only data.

FlowGMM Izmailov et al. (2019) introduced an approach to semi-supervised learning with normalizing flows called as *Flow Gaussian Mixture Model* (FlowGMM). They model the density in the "latent" space by a mixture of Gaussians, where each mixture component corresponds to a class represented in the labelled data.

The FlowGMM model is best illustrated with an example given by Izmailov et al. (2019). We reproduce their example of binary classification, for the sake of completeness, in Figure 1. The data is shown in the panel (a), where the triangles correspond to the labeled examples coloured according to their class label. The blue circles correspond to unlabeled data. Izmailov et al. (2019) introduce a Gaussian mixture with two components corresponding to the two classes, in panel (c). Note that mixture of Gaussians are in the *latent space* $\mathcal{Z}$. The invertible neural network f is trained to map the data distribution in the data space $\mathcal{X}$ to the Gaussian mixtures in the latent space $\mathcal{Z}$. The mapping is shown in the panel (a) and (b) in Figure 1. Since the neural network in invertible, we can talk about f^{-1}. The f^{-1}

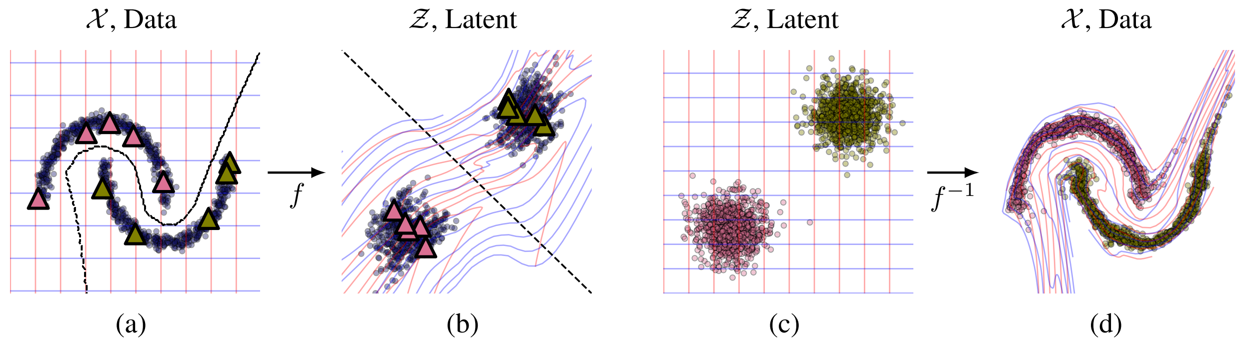


Figure 1. Izmailov et al. (2019, Figure 1). Illustration of semi-supervised learning with FlowGMM on a binary classification problem. Colors represent the two classes or the corresponding Gaussian mixture components. Labeled data are shown with triangles, colored by the corresponding class label, and 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.

serves as a class-conditional generative model visualized in panel (d). To classify a data point $x \in \mathcal{X}$, first compute its image $f(x) \in \mathcal{Z}$, and pick the class corresponding to the Gaussian that is closest to $f(x)$. Since we pick the Gaussian component closest to $f(x)$, the decision boundary in the latent space is the hyperplane (shown in panel (b)) which is the perpendicular bisector of the line joining the means of the two Gaussians (assuming the Gaussian components are distributed with identity covariance matrices). Since the flow is trained to represent data as the transformation the latent distribution, the density near the decision boundary should also be low, as visualized in panel (a). Therefore, Izmailov et al. (2019) claim that the FlowGMM encodes the "*clustering principle*: The decision boundary between classes lie in the low-density region in the data space."

Izmailov et al. (2019) use likelihood maximization to learn the parameters of the neural networks. Using a point estimator for the neural network's parameters will increase the estimation bias by ignoring epistemic uncertainty.

Bayesian Inference In this work, our approach is to perform Bayesian model averaging over high performing models to improve generalization. We first motivate with visual examples for why using the maximum likelihood solution can yield to bad solutions. The FlowGMM can be sensitive to the locations of the labelled examples. For instance, in Figure 2, we chose the labels (represented by triangles) in the two classes carefully. We believe that this could correspond to real-world settings where obtaining labelled examples are hard to obtain, and even if we do obtain them, they need not be iid samples from the data-generating distributions. In such a case, the panel (a) of Figure 2 using the FlowGMM show that the classifier obtained is not desirable. Since there are many possible classification boundaries possible, we believe that Bayesian model averaging might give a more desirable result.

Next, we explore the techniques discussed in Wilson & Izmailov (2020) to perform Bayesian model averaging. We first look at a technique proposed by Lakshminarayanan et al. (2017) called the deep-ensembles, which as suggested by Wilson & Izmailov (2020) can be viewed as a compelling mechanism for Bayesian marginalization. We show in panel (b) of Figure 2 that deep ensembles (using 10 ensembles) indeed give a more desirable boundary. This example gives a clear evidence that Bayesian inference can be used to obtain better classifiers. We also verify this on real-world data and show our results in the experiments section.

Wilson & Izmailov (2020) also propose a method for Bayesian marginalization called MultiSWAG which obtain better resultEXs than deep ensembles. We had initially planned to work implement it but due to inferior results of a single run of SWAG, we decided to not go ahead with this and work on improving calibration of our existing approaches.

In the experiments, we noticed that our model was overconfident. Izmailov et al. (2019) improve their model calibration by learning the variance scaling of the Gaussians in the latent space, which they claim is equivalent to temperature-scaling proposed by Guo et al. (2017). The accuracy vs confidence curve post scaling improves (see the Experiments section).

Both deep ensembles and multi-SWAG train the neural networks multiple times. Since training neural networks multiple times can be computationally prohibitive, we try the performance of stochastic weight averaging (SWA) proposed by Izmailov et al. (2018) in hope to improve the classification in real-world data. However, we found no significant improvement when using SWA as opposed to

FlowGMM. We shall discuss about this in the final version of the report. We also implemented the SWA-Gaussian (SWAG) approach proposed in (Maddox et al., 2019), where one can sample from the posterior distribution of parameters at the SWA solution. Both SWA and SWAG in our case performed inferior to the original FlowGMM approach. We believe that it could be due to the difficulty training the model, i.e., choosing the optimal parameters for the learning rate. We show performance of deep-ensembles on text data-sets and discuss the results in the Experiments section.

Finally, we introduce something called the *flow ensembles*. This is also an ensembling method, where we exploit the fact that we have access to the exact log-likelihood of the labels given the data. We provide intuition about it in the Methodology section, but the interesting part is that it consistently outperforms deep-ensembles in all our experiments.

2. Related Works

Normalizing Flows were popularised by Rezende & Mohamed (2015) in the context of variational inference and by Dinh et al. (2014) for density estimation. Our work builds on top of (Izmailov et al., 2019). Izmailov et al. (2019) propose FlowGMM, an approach to generative semi-supervised learning with normalizing flows, using a latent Gaussian mixture model. They show good performance on broad range of semi-supervised learning tasks, on image, text, and tabular data classification. Nalisnick et al. (2019) trains a classifier on the latent representation to perform semi-supervised learning on image classification. This is different from (Izmailov et al., 2019) because they use class-conditional likehihood to perform classification. A nice theoretical property of FlowGMM is the generative modelling is used directly as a Bayes classifier, and if the generative model obtained is exact, then the Bayes classifier achieves provably optimal mis-clssification rate (Mohri et al., 2012).

Dinh et al. (2017) introduced a real-valued non-volume preserving (real NVP) neural-networks, a set of powerful, stably invertible, and learnable neural-networks, resulting in an unsupervised learning algorithm with exact log-likelihood computation to tackle the problem of learning highly nonlinear models in high-dimensional continuous spaces through maximum likelihood. We use the realNVP architecture in all our experiments.

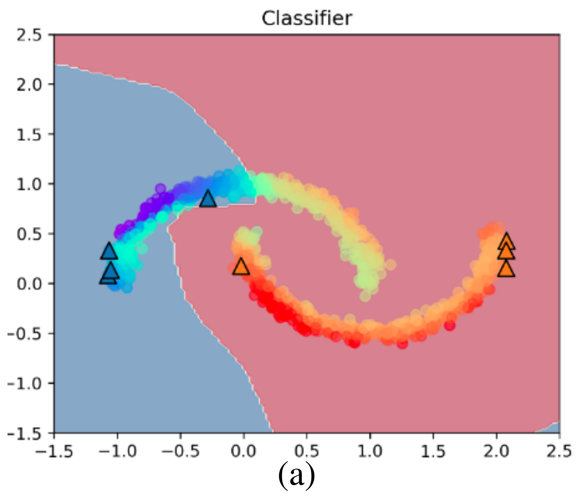
In this work we use Bayesian inference for FlowGMM model. There have been many works on Bayesian neural networks and we refer the interested reader to (Wilson & Izmailov, 2020) to know about some earlier works on Bayesian neural networks and existing approaches to Bayesian inference. Here, we mostly focus on deep ensembles, SWA, SWAG, and multi-SWAG. Deep ensembles, proposed by Lakshminarayanan et al. (2017), train an ensemble of neural networks by initializing at M different values and repeating the minimization multiple times which could lead to M different solutions, if the loss is non-convex. The final prediction is given by the average of the M predictions. Wilson & Izmailov (2020) argue that deep ensembles give an approximation to the BMA integral (Equation (6)). This is because deep ensembles are formed by maximum likelihood retraining of the same neural network architecture multiple times, leading to different basins of attraction. Stochastic weight averaging (SWA) introduced by Izmailov et al. (2018) performs an equally weighted average of the points traversed by stochastic gradient descent (SGD) with a cyclical or high constant learning rate. Izmailov et al. (2018) argue that SWA can improve generalization of the neural network by finding parameters in flat regions that tend to perform well. Maddox et al. (2019) introduced SWA-Gaussian (SWAG), which builds on SWA. SWAG additionally computes a low-rank plus diagonal approximation to the covariance of the iterates, which is used together with the SWA mean, to define a Gaussian posterior approximation over neural network weights. One can then sample from this Gaussian distribution to perform Bayesian model averaging. We refer the readers to (Izmailov et al., 2018; Maddox et al., 2019) for further discussion on generalization. Inspired by deep ensembles Wilson & Izmailov (2020) introduce multi-SWAG. Instead of focusing SWAG approximation on a single basin, which may contain a lot of redundancy in function space, making a relatively minimal contribution to computing the Bayesian predictive distribution, one can represent multiple basins of attraction by using ensembles of SWAG.

3. Methodology

Background 1: Normalizing Flows Normalizing flow (Dinh et al., 2017) is an unsupervised model for density estimation. It is defined as an invertible map $f : \mathcal{X} \to \mathcal{Z}$ (an invertible neural network), which maps from data space $\mathcal{X}$ to latent space $\mathcal{Z}$. The data distribution can be modelled as a transformation $f^{-1} : \mathcal{Z} \to \mathcal{X}$, applied to a random variable from latent distribution $z \sim p_{\mathcal{Z}}$. The density of the transformed random variable x can then be calculated using 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 model can be trained by maximizing likelihood in Equation (1) of training data with respect to parameters θ of the function f (i.e., the weights of the invertible neural network).



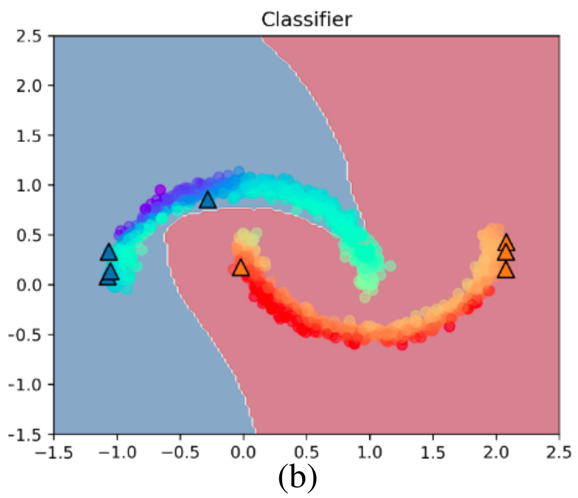


Figure 2. Classification boundary of FlowGMM (Izmailov et al., 2019) and FlowGMM with deep ensembles when the labelled data are not uniformly available over the data-distribution. We created "biased" labels, meaning, labels which do not give information near the decision boundary, on a 2-dimensional data-set. **(a):** Shows the the classification boundary using FlowGMM, which is not well represented in the areas where the labelled examples are lacking. **(b):** Shows that classification boundary using FlowGMM with deep ensemble (10 ensembles). We see that the decision boundary has significantly improved. This example provides a good visualization for why Bayesian model averaging is necessary.

Background 2: FlowGMM We follow the exposition of Izmailov et al. (2019). The number of class labels are assumed to be known before-hand based on the labelled data. First, a latent variable $y \in \{1, \dots, \mathcal{C}\}$ for the class label. The latent space distribution, condition on a label k, is a 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 is then a mixture of Gaussians. When the classes are balanced, then the 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 (1) and (2), the likehihood for the labelled data is

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

and the likelihood of the unlabelled data is $p_{\mathcal{X}}(x) = \sum_k p_{\mathcal{X}}(x|y=k)\; p(y=k)$. The joint likehihood of the labelled ($\mathcal{D}_l$) and the unlabelled ($\mathcal{D}_u$) data-set can be written as

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

Izmailov et al. (2019) train the model in semi-supervised way by maximizing the joint likelihood of the labelled and unlabelled data over the parameters θ of the function f. This can be used to learn a density model for a Bayes classifier. For a given test point x_*, the predictive distribution is given by

$$\begin{aligned} p_{\mathcal{X}}(y|x_*, \theta_{mle}) &= \frac{p_{\mathcal{X}}(x_*|y, \theta_{mle})\; p(y)}{p(x_*|\theta_{mle})} \\ &= \frac{\mathcal{N}(f_{\theta_{mle}}(x_*)|\mu_y, \Sigma_y)}{\sum_{k=1}^{\mathcal{C}} \mathcal{N}(f_{\theta_{mle}}(x_*)|\mu_y, \Sigma_y)}. \end{aligned} \tag{5}$$

Then, the prediction for a test point x_* with Bayes decision rule

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

Bayesian Model Averaging (BMA) Recall that in Equation (5) we assumed the parameter $\theta = \theta_{mle}$ to be fixed, which was learnt via maximum likehihood of the data (see Equation (4)) with respect to parameter θ. Instead, we want to use all settings of parameters θ, weighted by their posterior probabilities. In this work we look at the Bayesian model average with respect to θ. Let $\mathcal{D}$ denote the set of all data, i.e., $\mathcal{D} = \mathcal{D}_u \cup \mathcal{D}_l$. Concretely, given a test point x_*, the predictive distribution is given by

$$\begin{aligned} p_{\mathcal{X}}(y|x_*, \mathcal{D}) &= \int p_{\mathcal{X}}(y|x_*, \theta)\ p(\theta|\mathcal{D})\ \mathrm{d}\theta \\ &= \int \frac{p_{\mathcal{X}}(x_*|y, \theta)\ p(y)}{p_{\mathcal{X}}(x_*|\theta)}\ p(\theta|\mathcal{D})\ \mathrm{d}\theta \qquad (6) \\ &= \int \frac{\mathcal{N}(f_\theta(x_*)|\mu_y, \Sigma_y)}{\sum_{k=1}^{\mathcal{C}} \mathcal{N}(f_\theta(x_*)|\mu_y, \Sigma_y)}\ p(\theta|\mathcal{D})\ \mathrm{d}\theta. \end{aligned}$$

We recommend the readers to contrast this with Equation (5). We can then make predictions for a test point x_* with the Bayes decision rule

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

Computing the integral in the Equation (6) is difficult. Fortunately, there are some known approximation/heuristics to compute the above integral. We find the work of Wilson & Izmailov (2020) most relevant to compute the approximation to the integral, where they propose that deep-ensembles and multi-SWAG can act as a "good" approximation to the integral. Please refer the related works section for a more detailed discussion on this.

Deep Ensembles and Flow Ensembles We find that a new ensembling method, which we call the *flow ensemble*, consistently performs better than deep-ensembling and multi-SWAG in our experiments. To explain that, we first look at what deep-ensembles precisely does. Wilson & Izmailov (2020) propose that deep ensembles give a approximation to the BMA integral (Equation (6)). That is,

$$p(y|x_*, \mathcal{D}) \approx \frac{1}{M} \sum_{j=1}^{M} p(y|x_*, \theta_j),$$

where $\theta_1, \ldots, \theta_M$ are the minimized values of the neural network, starting from M-different initialization. With normalizing flows, we have access to exact log-likelihoods. Instead of taking average of output probabilities as in deep ensembles, we take average of log-likelihood of labels, given the test data. We call this the *flow ensemble*, i.e.,

$$\begin{aligned} p(y|x_*, \mathcal{D}) &\approx \frac{\exp\left(\frac{1}{M}\sum_{j=1}^{M} \log p_{\mathcal{Z}}(f_{\theta_j}(x_*)|\mu_y, \Sigma_y)\right)}{\sum_{k=1}^{\mathcal{C}} \exp\left(\frac{1}{M}\sum_{j=1}^{M} p_{\mathcal{Z}}(f_{\theta_j}(x_*)|\mu_k, \Sigma_k)\right)} \\ &= \frac{\left(\prod_{j=1}^{M} p_{\mathcal{Z}}(f_{\theta_j}(x_*)|\mu_y, \Sigma_y)\right)^{1/M}}{\sum_{k=1}^{\mathcal{C}} \left(\prod_{j=1}^{M} p_{\mathcal{Z}}(f_{\theta_j}(x_*)|\mu_k, \Sigma_k)\right)^{1/M}}. \end{aligned} \qquad (7)$$

where $\theta_1, \ldots, \theta_M$ are as before. Intuitively, this method chooses a label which has consistently-high likelihood across all the M-runs. As we shall see in the experiments section, this method consistently achieves better accuracy than deep ensembles.

Table 1. Accuracy on AG-News and Yahoo Answers datasets processed with BERT with a small number of labeled examples. The Deep Ensembles-3 and Flow Ensembles-3 models are combined from 3 indepently trained FlowGMM models. The FlowGMM model is chosen from the three models in the ensembles, based on the performance on the validation set. n_l and n_u are the number of labeled and unlabeled data points.

	Dataset (n_l / n_u, classes)	
Method	AG-News (200 / 120k, 4)	Yahoo Answers (800 / 50k, 10)
FlowGMM	82.6	59.0
Deep Ensembles-3	83.3	62.2
Flow Ensembles-3	**83.8**	**63.2**

4. Experiments

Text data We evaluated our ensemble models for semi-supervised text classification tasks on AG-News and Yahoo Answers datasets. The datasets were preprocessed using the BERT transformer model (Devlin et al., 2019) trained on a corpus of Wikipedia articles (Izmailov et al., 2019). We trained FlowGMM on the embeddings of each dataset using the RealNVP architecture with 7 coupling layers, each with 1 hidden layer of size 1024. The Deep Ensembles a Flow Ensembles model is combined from three separately trained FlowGMM models. We compare the results of FlowGMM with Deep Ensembles models on the two datasets in Table 1.

The class predictive probabilities of FlowGMM in Equation (5) can be expressed by taking the softmax of the log likelihoods, called the logits, of test data in the latent space. The prediction probabilities of the Deep Ensembles model are then obtained by taking the mean of prediction probabilities of individual ensemble models. As expected, the Deep Ensembles model outperforms the individual FlowGMM models on both AG-News and Yahoo Answers dataset. The Flow Ensembles differs from Deep Ensembles by computing the prediction probabilities by taking the softmax function of the averaged logits of individual models instead. The results in Table 1 includes the performance of our newly introduced model, which further improved accuracy compared to Deep Ensembles.

Deep Ensembles and Flow Ensembles models In this section, we look into the quality of the predictive distribution of Deep Ensembles and Flow Ensembles on the AG-News and Yahoo Answers datasets. The mean vectors μ_i are randomized, and the covariance matrix is set to be identity for all classes (Izmailov et al., 2019). They are kept constant throughout the whole training procedure. We attempted to specify the parameters based on data, and learn such parameters by optimizing the likelihood. Neither of

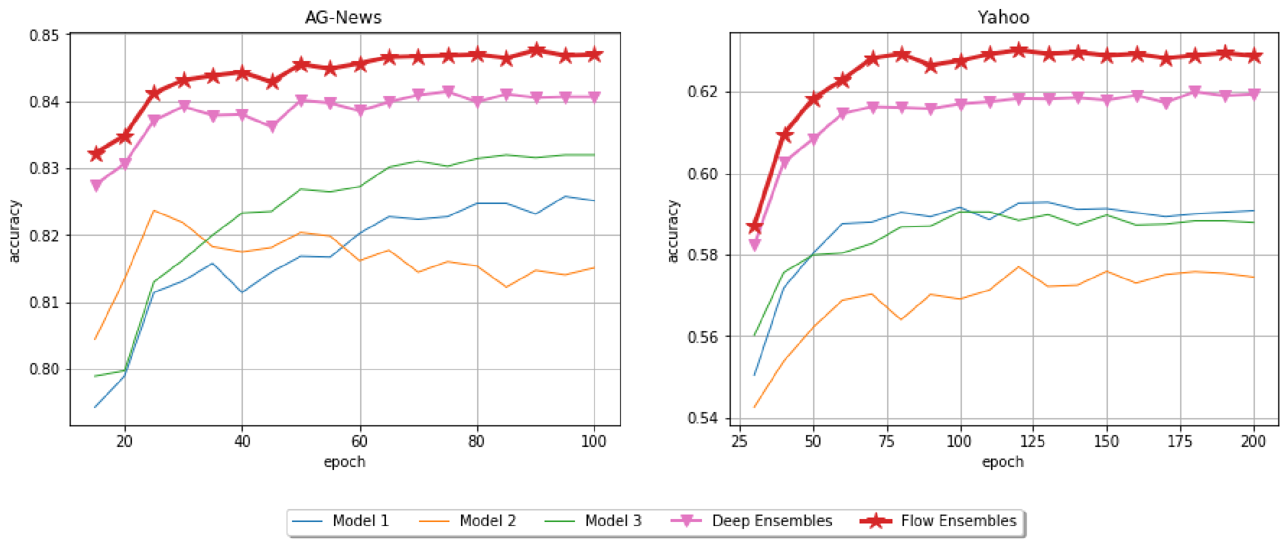


Figure 3. Test accuracy for FlowGMM, Deep Ensembles and Flow Ensembles models on AG-News and Yahoo News datasets over various training epochs. The Flow Ensembles and Deep Ensembles models combine three indepedently trained FlowGMM models (Model 1, Model 2, and Model 3) with randomized means for the Gaussian mixture components. The Flow Ensembles model constantly provides improvement in accuracy over the Deep Ensembles model, and significantly outperforms the individual FlowGMM models.

such methods provided a significant improvement over performance of the FlowGMM model, even though the model was able to achieve 100% accuracy on the labeled training data. We believe that the invertible neural network is flexible enough to map the labeled data, along with nearby unlabeled data, to be close to the mean of the corresponding Gaussian mixture component, under any settings of the parameters.

Rather than betting on a single setting of the model parameters, which include the parameters of the invertible flow and the Gaussian mixture components, we could benefit by performing Bayesian Model Averaging over parameters (Wilson & Izmailov, 2020). The Deep Ensembles model allows us to represent uncertainty of parameters by combining different flows with different settings of the Gaussian means. One can think of combining FlowGMM models with different Gaussian parameters as exploring different basins of attraction. By taking the average over several models' predictions, Deep Ensembles will take into account the uncertainty in class predictions over all models and make better predictions.

Accuracy and Calibration The predictive probabilities of FlowGMM are mostly overconfident. Due to the curse of dimensionality, the long distance between Gaussian mixture components will lead to hard label assignment ($p(y|x) \approx 1$ for exactly one class). The predictive probabilities of Deep Ensembles over highly confident predictions will be spread out evenly over several classes and can lead to random guessing predictions. The intuition for Flow Ensembles is from the uncertainty of test data in latent space.

The Flow Ensembles model computes the prediction probabilities over the average of the logits, which in Equation (7) is equivalent to setting the likelihood to be the geometric mean of the likelihood of individual models. We can see that the Flow Ensembles model will be more likely to predict a class with consistently high likelihood between models. Flow Ensembles will also reduce the effect of the curse of dimensionality by scaling down the likelihood of overconfident predictions. Figure 3 show the accuracy of the Deep Ensembles model, the Flow Ensembles model and the three individual FlowGMM models over different training epochs.

Another quality of Deep Ensembles is improving model calibration and produce reliable predictive probabilities. Since the FlowGMM predictions are overconfident, directly averaging over several models' predictions significantly improves the quality of predictive probabilities. Our Flow Ensembles model instead compute the probabilities over the averaged logits in latent space. In our experiments, the improve in Expected Calibration Error for the Flow Ensembles over its individual models is insignificant. This is because the predictions of Flow Ensembles are averaged over latent space, which encourages overconfidence due to the curse of dimensionality.

Temperature Scaling Izmailov et al. (2019) proposes a method to improve calibration by introducing the covariance scale parameter σ^2 for all components in the Gaus-

Table 2. Negative log-likelihood and Expected Calibration Error for semi-supervised FlowGMM trained on AG News (200 labeled, 120k unlabeled, 100 validation, 7600 test) and Yahoo Answers (800 labeled, 50k unlabeled, 400 validation, 10k test). FlowGMM-temp and FE-3-temp stands for tempered FlowGMM and Flow Ensembles over three FlowGMM models where the temperature parameter was learned on a validation set.

	AG News		Yahoo Answers	
	NLL ↓	ECE ↓	NLL ↓	ECE ↓
FlowGMM	9.034	0.181	18.139	0.399
FlowGMM-temp	0.826	0.122	1.381	0.036
FE-3	8.157	0.167	16.695	0.385
FE-3-temp	0.773	0.112	1.332	0.039

sian mixture (the component k will be $\mathcal{N}(\mu_k, \sigma^2 I)$). The parameter is learned by minimizing the negative log likelihood on a validation set. Re-calibrating the covariances of the Gaussian components was shown to improve negative log likelihood (NLL) loss and expected calibration error (ECE) supervised on supervised classification tasks on MNIST and CIFAR datasets. This re-calibrating procedure is equivalent to applying temperature scaling (Guo et al., 2017) to logits $\log \mathcal{N}(x|\mu_k, \Sigma_k)$. We refer to this procedure as tempering. We can see that the logits averaging procedure for Flow Ensembles is similar to temperature scaling, but it rescales the logits on the training set instead to improve predictive performance. We can combine Flow Ensembles with temperature scaling by dividing the averaged logits from Flow Ensembles by a learned temperature parameter over a validation set.

In Table 2, we report the improvement in NLL and ECE metrics in semi-supervised Flow Ensembles after temperature scaling. In Figure 4, we plot the accuracy as a function of confidence of the Flow Ensembles and the individual FlowGMM models, trained on the AG News and Yahoo Answers dataset. By confidence we mean that if a class has high probability during prediction then it has high confidence because it shows how much the algorithm is confident for that class.

Stochastic Weight Averaging Stochastic weight averaging (SWA) is a method proposed by Izmailov et al. (2018) to find flat optima that generalizes well. The procedure encourages optimization with a high constant learning rate to explore flat regions and averages over the weights. Using the SWA weights, we can sample and perform Bayesian Model Averaging by approximating a Gaussian distribution.

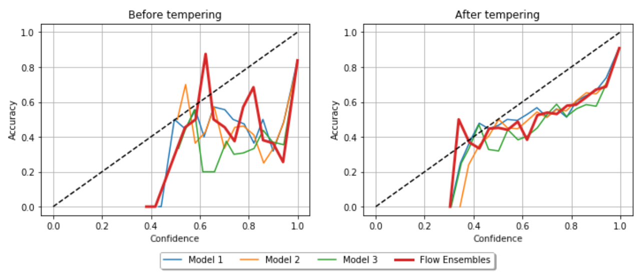
In our experiments, we used a decaying learning rate for the first 75% of training time, and a high constant learning rate for the remaining 25% of the time. We calculate the SWAG modes every 1% of training time. Overall, we didn't find good solutions either using SWA or averaging over SWAG samples. Optimizing the parameters of the flow using a low decaying learning rate discovers much better local minimum solutions than using a high constant learning rate. This, however, will result in finding solutions in regions with low flatness, and therefore doesn't generalize well. Optimizing using a high constant learning rate can find flat regions, but the predictive performance benefits of either the SWA solution, or performing Bayesian Model Averaging over the SWAG samples, is still inferior to using a low decaying learning rate. Due to difficulties in finding the optima flat regions, we believe that multi-SWAG will not provide improvement in predictive accuracy and calibration, compared to Deep Ensembles and Flow Ensembles.

5. Discussion

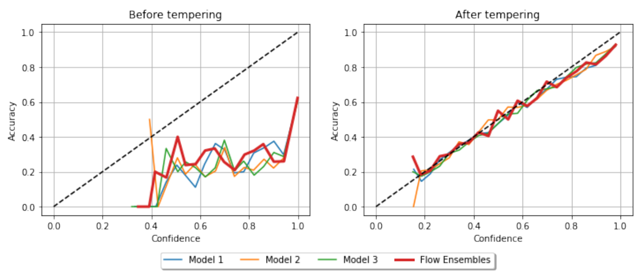
In this project we explore Bayesian inference for the FlowGMM model. While there are many classical methods for Bayesian model averaging, we consider the methods suggested by Wilson & Izmailov (2020), and propose our own ensembling method. Hence, we only consider deep ensembles, SWA, SWAG, flow-ensembles. To visualize the importance of the need of Bayesian model averaging, we chose labels carefully in Figure 2, panel (a) such that the classification boundary given by FlowGMM is not good. We then show in Figure 2, panel (b) that using 10 ensembles the classification boundary improved significantly!

Moreover, we experimented with real-world data to showcase that the deep ensemble heuristic to Bayesian model averaging indeed gives an improved performance. On AG-News data-set deep ensembles gave an improved accuracy of 0.7% and on Yahoo answers data-set, an improved accuracy of 3.2% (see Table 1). We also propose a method called Flow Ensembles, which instead of taking average of output probabilities as in deep ensembles, takes average of log-likelihood of labels, given the test data. We see in Figure 3 that Flow Ensembles consistently performs better than Deep Ensembles in terms of accuracy.

While Flow Ensembles provides improvement over Deep ensembles in predictive accuracy, it does not seem to improve calibration compared to the latter. This is because Flow Ensembles computes the predictive probabilities by averaging the log likelihood of data in latent space, which encourages overconfidence due to the curse of dimensionality. To improve calibration for Flow Ensembles, we performed temperature scaling over the logits by minimizing the negative log likelihood on a validation set. We believe that a more studied settings of the Gaussian mixture parameters will help reducing the overconfidence nature of the FlowGMM models and further improve calibration for Flow Ensembles



(a)



(b)

Figure 4. Accuracy over confidence of Flow Ensembles and FlowGMM models in semi-supervised classification before and after temperature scaling on **(a):** AG News and **(b):** Yahoo Answers dataset. The settings for temperature scaling are described in Table 2. Tempering significantly improves the calibration of the model using a relatively small validation set size.

We also experimented with Stochastic Weight Averaging parameter settings to find the flat minima for better generalization. Parameter optimization using a high constant learning rate resulted in difficulties in finding flat regions that perform well in predictive accuracy. More advanced learning rate schedulers that encourages exploring local minima such as cyclical learning rate may be promising in improving generalization for FlowGMM models.

References

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

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

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

Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'14, Cambridge, MA, USA, 2014. MIT Press.

Göpfert, Christina, Ben-David, Shai, Bousquet, Olivier, Gelly, Sylvain, Tolstikhin, Ilya, and Urner, Ruth. When can unlabeled data improve the learning rate? In *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pp. 1500–1518, Phoenix, USA, 25–28 Jun 2019. PMLR.

Guo, Chuan, Pleiss, Geoff, Sun, Yu, and Weinberger, Kilian Q. On calibration of modern neural networks. In Precup, Doina and Teh, Yee Whye (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/guo17a.html.

Iwata, Tomoharu, Duvenaud, David, and Ghahramani, Zoubin. Warped mixtures for nonparametric cluster shapes. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pp. 311–320. AUAI Press, 2013.

Izmailov, Pavel, Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018.

Izmailov, Pavel, Kirichenko, Polina, Finzi, Marc, and Wilson, Andrew Gordon. Semi-supervised learning with normalizing flows. *arXiv preprint arXiv:1912.13025*, 2019.

Lakshminarayanan, Balaji, Pritzel, Alexander, and Blundell, Charles. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Maddox, Wesley J, Izmailov, Pavel, Garipov, Timur, Vetrov, Dmitry P, and Wilson, Andrew Gordon. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, volume 32, pp. 13153–13164. Curran Associates, Inc., 2019.

Mohri, Mehryar, Rostamizadeh, Afshin, and Talwalkar, Ameet. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X.

Nalisnick, Eric, Matsukawa, Akihiro, Teh, Yee Whye, Gorur, Dilan, and Lakshminarayanan, Balaji. Hybrid models with deep and invertible features. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4723–4732, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Rezende, Danilo and Mohamed, Shakir. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

Wilson, Andrew Gordon and Izmailov, Pavel. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.