A. Expectation Maximization

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

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

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

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

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

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

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

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

B. Latent Distribution Mean and Covariance Choices

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

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

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

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

C. Synthetic Experiments

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

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

D. Tabular data preparation and hyperparameters

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

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

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

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

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

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

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

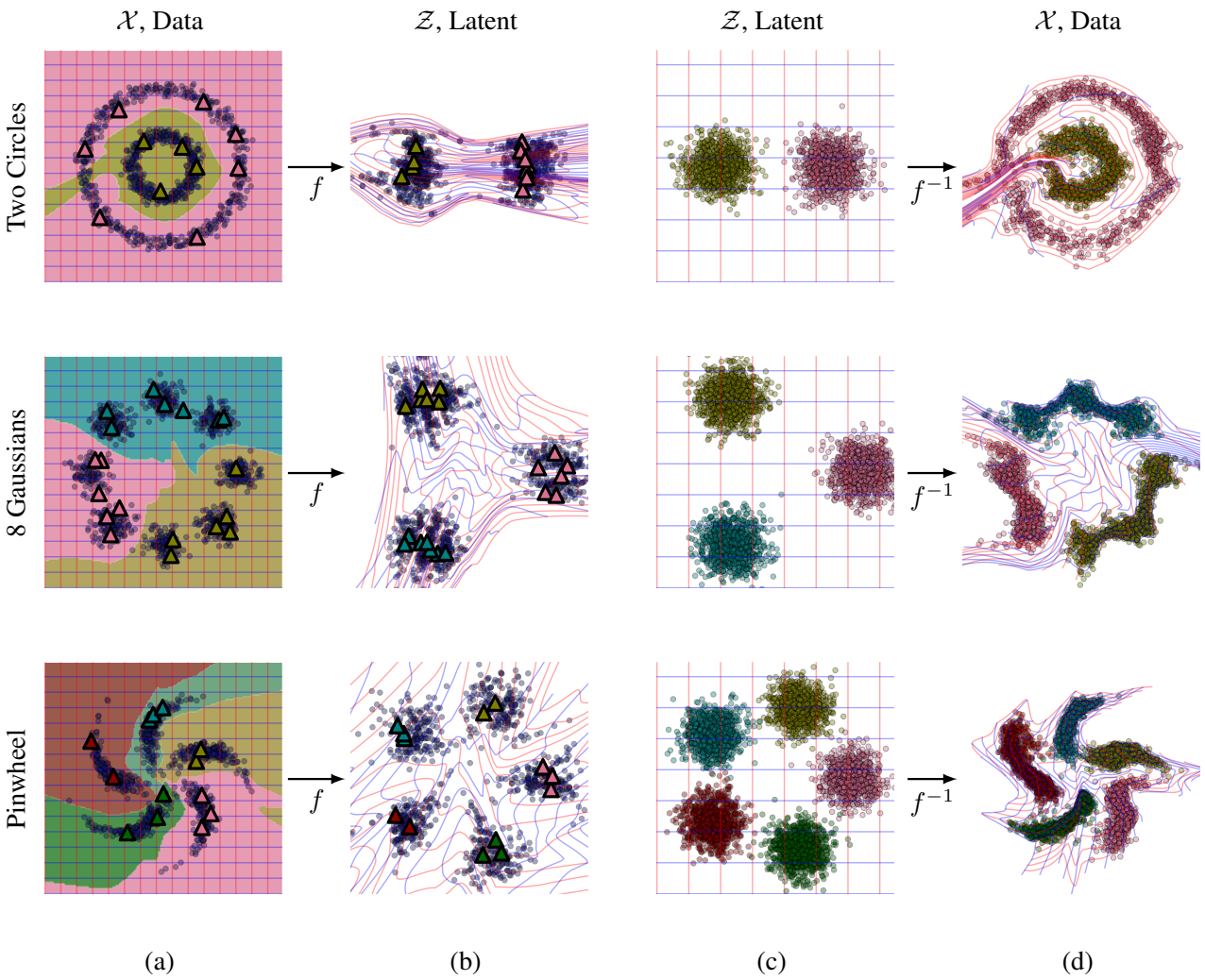


Figure 4. Illustration of FlowGMM on synthetic datasets: two circles (top row), eight Gaussians (middle row) and pinwheel (bottom row). **(a):** Data distribution and classification decision boundaries. Unlabeled data are shown with blue circles and labeled data are shown with colored triangles, where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

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

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

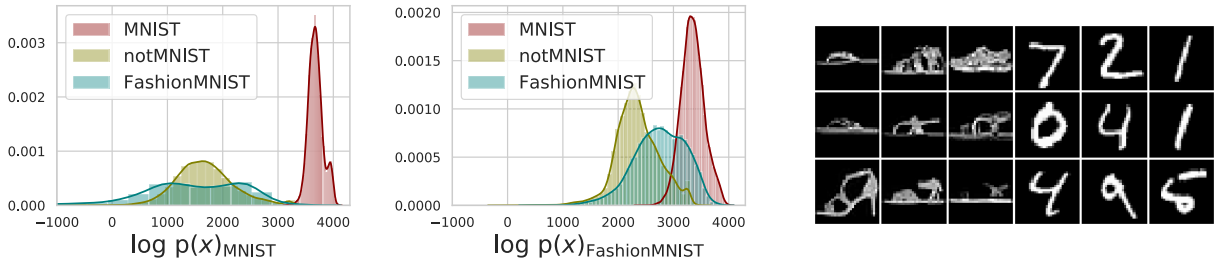


Figure 5. **Left:** Log likelihoods on in- and out-of-domain data for our model trained on MNIST. **Center:** Log likelihoods on in- and out-of-domain data for our model trained on FashionMNIST. **Right:** MNIST digits get mapped onto the sandal mode of the FashionMNIST model 75% of the time, often being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

E. Transfer learning

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

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

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

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

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

F. Image data preparation and hyperparameters

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

G. Out-of-domain data detection

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

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

H. Expected Distances between Gaussian Samples

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

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

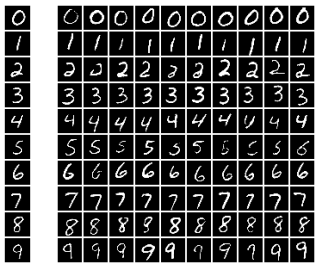
$$\begin{aligned}
\mathbb{E}\left[\|s_1 - \mu_2\|^2\right] &= \mathbb{E}\left[\mathbb{E}\left[\|s_1 - \mu_2\|^2 | \mu_1, \mu_2\right]\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{D} \mathbb{E}\left[(s_{1,i} - \mu_{2,i})^2 | \mu_{1,i}, \mu_{2,i}\right]\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{D} \left(1 +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

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

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

saturates at 1 with the growth of dimensionality D.

I. FlowGMM as generative model



(a) (b)

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

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

J. Non-Gaussian Latent Distributions

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

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

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

K. FlowGMM under Class Imbalance

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

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

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

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