On Mixed Conditional FFJORD with Large-Batch Training

Tan M. Nguyen [1 2] **Animesh Garg** [2] **Richard G. Baraniuk** [1] **Anima Anandkumar** [2 3]

Abstract

We introduce a new conditioning method for the Continuous Normalizing Flows and their improved version, the Free-form Jacobian of Reversible Dynamics (FFJORD) model. We name our new conditional model, the Mixed Conditional FFJORD (MC-FFJORD). In the MC-FFJORD, the latent code is partitioned into the supervised and unsupervised codes. The supervised code is used for conditioning the model while the unsupervised code accounts for other latent variations in the data. Our partitioning approach fosters the learning of disentangled representation by the model. Our empirical study corroborates the advantage of the MC-FFJORD over the baseline conditional FFJORD (C-FFJORD) in both classification and density estimation. We show that the MC-FFJORD requires less number of function evaluations and is more efficient than the C-FFJORD when training with large batches. We also explore the possibility of scaling up the MC-FFJORD using large-batch training with large learning rates and high error tolerances for the ODE solvers in the model. While the MC-FFJORD trained with this approach outperforms the MC-FFJORD trained with large batches only, its performance still lags behind that of the same model trained with small batches.

1. Introduction

Reversible models like the flow-based generative models are desirable in many downstream tasks including robust control and active learning due to their exact latent-variable inference and likelihood estimation. When conditioning these models on supervised signals, their reverse inference passes result in predictive models whose predictions are equipped with exact uncertainty estimation. Such uncertainty estimation is calculated from the likelihoods provided by the model and of importance for many real world applications including molecular modelling and robot control. However, computing the likelihood in the flow-based models is expensive and usually requires strict restrictions on the architectures in order to reduce the cost of computation. Recently, (Chen et al., 2018) introduced a new type of reversible models, named the Continuous Normalizing Flows (CNFs), which employ ordinary differential equations (ODEs) to transform the latent variables into data. Using the Hutchinsons trace estimator (Hutchinson, 1989), the Free-form Jacobian of Reversible Dynamics (FFJORD) models (Grathwohl et al., 2018) improve over the CNFs by providing an unbiased estimation of the likelihood at the very low cost of $\mathcal{O}(D)$ where D is the number of hidden units.

A conditioning method has been proposed for flow-based models and is used in the conditional GLOW (Kingma & Dhariwal, 2018). This method models the latent code with a mixture distribution, such as the mixture of Gaussians, whose parameters are functions of the conditioning supervised signals. An auxiliary classifier is usually applied on the latent code to improve the performance of the model. We find that such conditioning approach is inefficient, and from our experiments, we observe that it does not work well with the FFJORD. First, this conditioning method requires a large amount of extra parameters due to the large size of the latent code in FFJORD. Second, some of the extra parameters are unnecessary since different classes might share some common features. Learning those unnecessary parameters might hinder the learning of other useful features.

To tackle the aforementioned problems, we propose a simple yet efficient conditioning approach for the FFJORD. We name our method the Mixed Conditional FFJORD (MC-FFJORD). In the MC-FFJORD, we partition the latent code into two contiguous and non-overlapping part: the supervised and unsupervised codes (see Figure 1). We only use the supervised code to condition the model by employing the same mixture distribution and auxiliary classifier approach as in the traditional conditioning method mentioned above. We model the unsupervised code with a normal distribution. The unsupervised code captures other latent variations in the data. Our partitioning strategy encourages the model to learn better disentangled representation of the data. We empirically demonstrate that the MC-FFJORD outperforms the traditional conditional FFJORD (C-FFJORD) – which uses

[1]Rice University [2]NVIDIA [3]California Institute of Technology. Correspondence to: Tan M. Nguyen <mn15@rice.edu>.

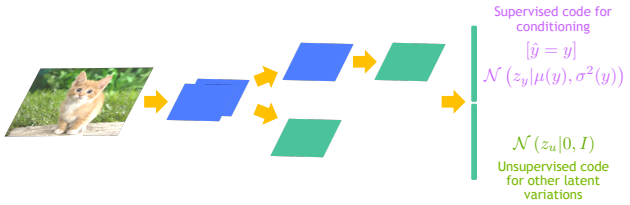


Figure 1. Mixed Conditional FFJORD (MC-FFJORD) model.The latent code is split into the supervised and unsupervised codes. The supervised code is used for conditioning the model and for classification. The unsupervised code captures other latent variations in the data.

the full latent code for conditioning – in both classification and density estimation while requiring smaller number of function evaluations when training with large batches.

In this paper, we also explore the possibility of scaling the MC-FFJORD with large-batch training in order to take advantage of the memory efficiency of the FFJORD-based models. In order to improve the generalization of the model which suffers from large-batch training, we employ the large learning rate method (Goyal et al., 2017; Smith et al., 2018) and propose to set high error tolerance for the ODE solvers in the MC-FFJORD. We observe that while the MC-FFJORD trained with large batches, large learning rate and high error tolerance improves over the MC-FFJORD trained with large batches only, its performance still lags behind the same model trained with small batches.

2. Mixed Conditional FFJORD

The MC-FFJORD only uses a portion of the latent code for conditioning on the supervised signals. In particular, let z be the latent code of the image x after the forward step in the FFJORD. We split z into two contiguous and non-overlapping parts – the supervised latent code z_y and the unsupervised latent code z_u – such that $z = [z_y, z_u]$. The latent code z_y is then utilized for conditioning the model on the supervised signals such as the object categories and for auxiliary supervised learning tasks including object classification and regression. Like in the conditional GLOWs, $p(z_y)$ is modeled by a mixture of Gaussian $\mathcal{N}(z_y|\mu(y), \Sigma(y))$ whose mean vectors $\mu(y)$ and covariance matrices $\Sigma(y)$ are functions of the supervised signal y. To ease the computation, we assume that the covariance matrices $\Sigma(y)$ are diagonal. The unsupervised latent code z_u follows a normal distribution $\mathcal{N}(z_u|0, I)$ and accounts for other latent variations in the data.

Compared to the C-FFJORD in which the full latent code is used for conditioning, the MC-FFJORD needs much less parameters since the size of the supervised code z_y is much smaller than the size of z. For example, in our experiments, z_y is only half the size of z. Furthermore, the partitioning of the latent code z into the supervised and unsupervised codes encourages the model to learn better disentangled representations of the input images and, therefore, facilitates the learning. As later discussed in Section 4.1, our experiments on CIFAR10 suggest that the MC-FFJORD requires significantly less number of function evaluations (NFEs) by the ODE solvers than the C-FFJORD. This evidence indicates that the partition strategy in the MC-FFJORD indeed helps alleviate the difficulty during the training and aids the learning of the model.

Learning: We learn the MC-FFJORD by optimizing the supervised loss and the negative log-likelihood of the model. The log-likelihood $\log p(x)$ is computed from $\log p(z)$ using the Change of Variables theorems as in the FFJORD. The Jacobian trace is approximated by the Hutchinson's trace estimator. The log-likelihood $\log p(z)$ of the latent code is the sum of $\log p(z_y)$ and $\log p(z_u)$.

3. Large-Batch Training for Mixed Conditional FFJORD

Large-batch training has been explored in the context of convolutional neural networks (CNNs) in order to scale up stochastic gradient-based methods. While large-batch training makes efficient use of the computational power of each GPU and reduces the training time significantly, it suffers from poor generalization since the training tends to converge to sharp minimizers of the loss function (Keskar et al., 2017). We observe that when training using large batches, the FFJORD model suffers from the same issue. In this section, we explored two potential solutions to improve the generalization of the MC-FFJORD and C-FFJORD trained with large batches. The first solution is to increase the learning rate proportional to the batch size. This approach has been studied in (Goyal et al., 2017; Smith et al., 2018) for large-batch training of CNNs. The second solution is to increase the error tolerance of the ODE solvers in the FFJORD-based models.

Using large learning rate: Large-batch training yields less noisy estimation for gradients. This lack of noise in stochastic gradients hurts the generalization of the model as suggested in (Keskar et al., 2017). Intuitively, large learning rate introduces more noise into the training and thus helps improve the generalization. When training the MC-FFJORD and C-FFJORD with large batches, we start with large learning rate proportional to the batch size, and then decay the learning rate during the training. We also follow the suggestion in (Goyal et al., 2017) to start the training with a warm up phase in which learning rate gradually increase from 0 to the chosen value.

Using large error tolerance of the ODE solvers: In addition to utilizing large learning rate, we also set larger error tolerance for the ODE solvers when training with large batches. The advantage of this approach is two-fold. First, it reduces the number of function evaluations by the ODE solvers and, therefore, speed up the training. Second, it introduces more noise into the gradients, especially when we increase the error tolerance of the reverse-time ODE solvers used in the adjoint method to estimate the gradients. From our experimental study, we observe that the first continuous normalizing flow (CNF) in each stacked CNF of the FFJORD models cannot tolerate a lot of error. Otherwise, the training quickly becomes unstable. For other CNFs in each stack, we can use high error tolerance.

4. Empirical Results

4.1. Mixed Conditional FFJORD vs. Conditional FFJORD

In this section, we empirically demonstrate the advantage of the MC-FFJORD over the baseline C-FFJORD when training on CIFAR10.

4.1.1. Implementation Details

Datasets: We validate the effectiveness of the MC-FFJORD on the CIFAR10 dataset. CIFAR10 contains 50,000 training images and 10,000 test images with classes from 10 different object categories. Uniform noise is added to the images, and during training, the data are augmented by random horizontal flip.

Networks: We experiment with the multiscale architecture composed of multiple flows as in (Grathwohl et al., 2018). The network uses 4 scale blocks, each of which contain 2 flows, a "squeeze" operator, and then other 2 flows. Each flow is made of 3 convolutional layers with 64 filters whose kernel size is 3. The squeeze operators are applied between flows to down-sample the spatial resolution of the images while increasing the number of channels. We apply the softplus nonlinearity at each layer.

When conditioning the model, we use linear networks for the classifier and the condition encoder. The parameters of the networks are initialized to zeros. In MC-FFJORD, we use half of the latent code for conditioning.

Training: We train both the C-FFJORD and MC-FFJORD with the Adam optimizer (Kingma & Ba, 2015). When using the batch size of 900, we train for 500 epochs with learning rate of 0.001 which was decayed to 0.0001 after 250 epochs. When using the batch size of 8,000, we train for 500 epochs with learning rate of 0.01 which was decayed to 0.001 and then to 0.0001 at epoch 125 and 400 respectively.

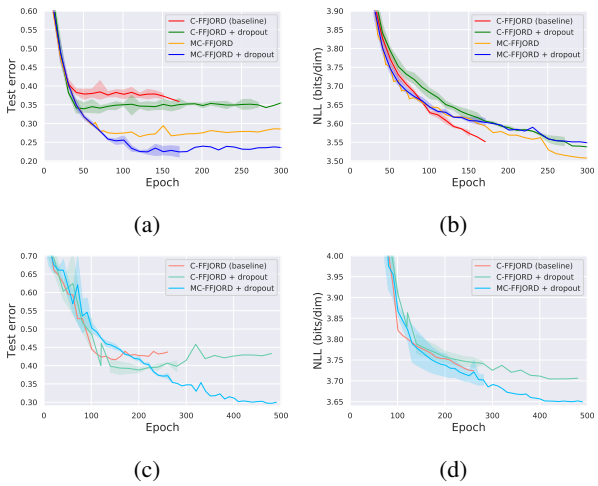


Figure 2. Evolution of classification errors and NLLs of the test data during the training of conditional FFJORD on CIFAR10 using small batch size (top row) and large batch size (bottom row).

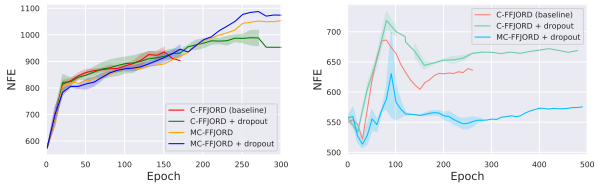
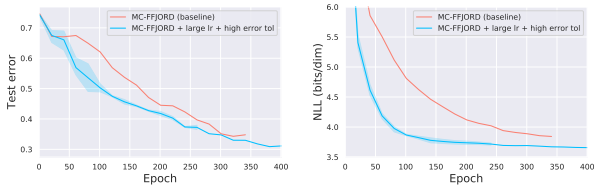


Figure 3. Evolution of the number of function evaluations during the training of conditional FFJORD on CIFAR10 using small batch size (left) and large batch size (right).

4.1.2. Quantitative Comparisons

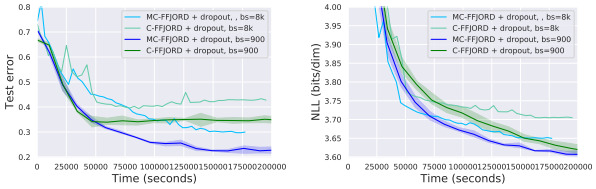
Classification error: Figure 2a shows that compared to the C-FFJORD, the MC-FFJORD improves the test classification error on CIFAR10 by 8.21%. The similar improvement (11.57%) has also been observed when we apply dropout (Srivastava et al., 2014) on both models in order to improve the classification results. Interestingly, the advantage of the MC-FFJORD over the C-FFJORD still holds when we train the models with large batch size of 8k (10.04% improvement in test classification error, see Figure 2c). Training the MC-FFJORD and the C-FFJORD with large batch size are quite unstable, and we need to apply dropout to help the training converge. Therefore, we only compare the MC-FFJORD and C-FFJORD with dropout when training with large batch size.

Likelihood: Not only does the MC-FFJORD facilitate supervised classification but it also enhances the negative log-likelihood scores compared to the C-FFJORD. Figure 2b reveals a slight improvement in negative log-likelihood of MC-FFJORD over C-FFJORD when training using small



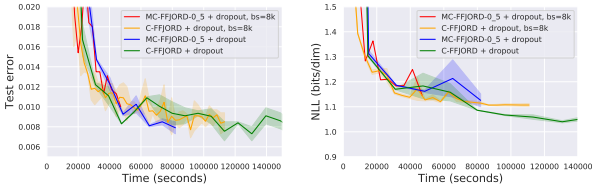
(a) (b)

Figure 4. (a) Test errors and (b) test NLLs for MC-FFJORD trained on CIFAR10 using large batch size with and without large learning rate and high error tolerance.



(a) (b)

Figure 5. (a) Test errors and (b) test NLLs versus training time for conditional FFJORD trained on CIFAR10 using small and large batch size.



(a) (b)

Figure 6. (a) Test errors and (b) test NLLs versus training time for conditional FFJORD trained on MNIST using small and large batch size.

batches with and without dropout. The improvement is more consistent and visible in Figure 2d when we train both models using large batches (around 0.056% improvement in NLL). When training with small batches, we also observe that using dropout slightly hinders the learning for density estimation and increase the negative log-likelihood by a small amount.

Number of Function Evaluations: Given the promising advantage of the MC-FFJORD over the C-FFJORD in classification and density estimation tasks, we would like to compare the efficiency of these models. On one hand, one would expect that the MC-FFJORD would be more efficient than the C-FFJORD since the MC-FFJORD has less parameters in its classifier and condition encoder. On another hand, using smaller-size latent codes for conditioning might make classification and density estimation tasks more challenging for the model, thereby increasing the number of function evaluations in the ODE solvers and then the training time. We measure the the number of function evaluations (NFEs) in the ODE solvers during the training of both the MC-FFJORD and C-FFJORD to shed light on their comparative efficiency. Figure 3 shows the evolution of NFEs during training of MC-FFJORD and C-FFJORD with small batch size of 900 (3a) and large batch size of 8k (3b). While the MC-FFJORD is as efficient as the C-FFJORD in the small batch size realm, it requires much less NFEs than the C-FFJORD when training with large batch size (around 100 NFEs less). This result together with the advantage of the MC-FFJORD and the C-FFJORD supports our hypothesis that not all activations in the latent code are needed to condition the FFJORD model.

4.2. Comparison between Large-Batch and Small-Batch Trainings

Training neural networks using large batches is much faster than small batches, but suffers from poor generalization. Using large learning rate and fine-tuning error tolerance for the ODE solvers have helped improve the training speed and generalization of the MC-FFJORD trained with large batch size (see Figure 4). Given promising results from training MC-FFJORD with both large and small batches in section 4.1, we would like to compare their performance in term of training speed. Currently, the MC-FFJORD trained with small batches yields better test classification errors and negative log-likelihoods on CIFAR10 than the MC-FFJORD trained with large batches. However, we would like to explore if the MC-FFJORD trained with large batches can reach certain test classification errors and negative log-likelihoods faster than the MC-FFJORD trained with small batches. In this section, we also extend the comparison to C-FFJORD trained with small and large batches.

Figure 5a shows that in spite of using both large learning rate and fine-tuned error tolerance, the MC-FFJORD and C-FFJORD trained with large batches still lag behind their counterparts trained with small batches in term of test classification error. However, in term of negative log-likelihood score, Figure 5b demonstrates that the MC-FFJORD and C-FFJORD trained with large batches reach a certain value of negative log-likelihood faster than with small batches.

Results on MNIST: We evaluate our proposed methods on the MNIST dataset. Figure 6a and b shows that when training with small batches, the MC-FFJORD is still better than the C-FFJORD in term of test classification error, but not the negative log-likelihood. When training with large batches, we do not observe much advantage of the MC-FFJORD over the C-FFJORD even though large-batch training reaches certain values of test error and test negative log-likelihood slightly faster than small-batch training.

References

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pp. 6571–6583, 2018.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2018.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines (corr: V19 p433-450). *Communications in Statistics: Simulation and Computation*, 18:1059–1076, 1989.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

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

Smith, S. L., Kindermans, P.-J., and Le, Q. V. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1Yy1BxCZ.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.