Robust model training and generalisation with Studentising flows

Simon Alexanderson [1] **Gustav Eje Henter** [1]

Abstract

Normalising flows are tractable probabilistic models that leverage the power of deep learning to describe a wide parametric family of distributions, all while remaining trainable using maximum likelihood. We discuss how these methods can be further improved based on insights from robust (in particular, resistant) statistics. Specifically, we propose to endow flow-based models with fat-tailed latent distributions such as multivariate Student's t, as a simple drop-in replacement for the Gaussian distribution used by conventional normalising flows. While robustness brings many advantages, this paper explores two of them: 1) We describe how using fatter-tailed base distributions can give benefits similar to gradient clipping, but without compromising the asymptotic consistency of the method. 2) We also discuss how robust ideas lead to models with reduced generalisation gap and improved held-out data likelihood. Experiments on several different datasets confirm the efficacy of the proposed approach in both regards.

1. Introduction

Normalising flows are tractable probabilistic models that leverage the power of deep learning and invertible neural networks to describe highly flexible parametric families of distributions. In a sense, flows combine powerful implicit data-generation architectures (Mohamed & Lakshminarayanan, 2016) of generative adversarial networks (GANs) (Goodfellow et al., 2014) with the tractable inference seen in classical probabilistic models such as mixture densities (Bishop, 1994), essentially giving the best of both worlds.

Much ongoing research into normalising flows strives to devise new invertible neural-network architectures that increase the expressive power of the flow; see Papamakarios et al. (2019) for a review. However, the invertible transformation used is not the only factor that determines the success of a normalising flow in applications. In this paper, we instead turn our attention to the latent (a.k.a. *base*) distribution that flows use. In theory, an infinitely-powerful invertible mapping can turn any continuous distribution into any other, suggesting that the base distribution does not matter. In practice, however, properties of the base distribution can have a decisive effect on the learned models, as this paper aims to show. Based on insights from the field of robust statistics, we propose to replace the conventional standard-normal base distribution with distributions that have fatter tails, such as the Laplace distribution or Student's t. We argue that this simple change brings several advantages, of which this paper focusses on two aspects:

1. It makes training more stable, providing a principled and asymptotically consistent solution to problems normally addressed by heuristics such as gradient clipping.
2. It improves generalisation capabilities of learned models, especially in cases where the training data fails to capture the full diversity of the real-world distribution.

We present several experiments that support these claims. Notably, the gains from robustness evidenced in the experiments do not require that we introduce any additional learned parameters into the model.

2. Background

Normalising flows are nearly exclusively trained using maximum likelihood. We here (Sec. 2.1) review strengths and weaknesses of that training approach; how it may suffer from low statistical robustness and how that affects typical machine-learning pipelines. We then (Sec. 2.2) discuss prior work leveraging robust statistics for deep learning.

2.1. Maximum likelihood and outliers

Maximum likelihood estimation (MLE) is the gold standard for parameter estimation in parametric models, both in discriminative deep learning and for many generative models such as normalising flows. The popularity of MLE is grounded in several appealing theoretical properties. Most importantly, MLE is consistent and asymptotically efficient under mild assumptions (Daniels, 1961). Consistency means that, if the true data-generating distribution is a member of the

[1]Division of Speech, Music and Hearing, KTH Royal Institute of Technology, Stockholm, Sweden. Correspondence to: Gustav Eje Henter <ghe@kth.se>.

Second workshop on *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models* (ICML 2020), Virtual Conference

parametric family we are using, the MLE will converge on that distribution in probability. Asymptotic efficiency adds that, as the amount of data gets large, the statistical uncertainty in the parameter estimate will furthermore be as small as possible; no other consistent estimator can do better.

Unfortunately, MLE can easily get into trouble in the important case of *misspecified models* (when the true data distribution is not part of the parametric family we are fitting). In particular, MLE is not always robust to *outliers*[1]: Since $\ln 0 = -\infty$, outlying datapoints that are not explained well by a model (i.e., have near-zero probability) can have an unbounded effect on the log-likelihood and the parameter estimates found by maximising it. As a result, MLE is sensitive to training and testing data that doesn't fit the model assumptions, and may generalise poorly in these cases.

As misspecification is ubiquitous in practical applications, many steps in traditional machine-learning and data-science pipelines can be seen as workarounds that mitigate the impact of outliers before, during, and after training. For example, careful data gathering and cleaning to prevent and exclude idiosyncratic examples prior to training is considered best practise. Seeing that encountering poorly explained, low-probability datapoints can lead to large gradients that destabilise minibatch optimisation, various forms of gradient clipping are commonplace in practical machine learning. This caps the degree of influence any given example can have on the learned model. The downside is that clipped gradient minimisation is not consistent: Since the true optimum fit sits where the average of the loss-function gradients over the data is zero, changing these gradients means that we will converge on a different optimum in general. Finally, since misspecification tends to inflate the entropy of MLE-fitted probabilistic models (Lucas et al., 2019), it is common practice to artificially reduce the entropy of samples at synthesis time for more subjectively pleasing output; cf. Kingma & Dhariwal (2018); Brock et al. (2019); Henter & Kleijn (2016). The goal of this paper is to describe a more principled approach, rooted in robust statistics, to reducing the sensitivity to outliers in normalising flows.

2.2. Prior work

Robust statistics, and in particular influence functions (Sec. 3) have seen a number of different uses with deep learning, such as explaining neural network decisions (Koh & Liang, 2017) and subsampling large datasets (Ting & Brochu, 2018). In this work, however, we specifically consider statistical robustness in learning probabilistic models, following Hampel et al. (1986); Huber & Ronchetti (2009). This process can be made more robust in two ways: either by changing the parametric family or by changing the fitting principle. Both the first and the second approach have been used in deep learning before. Generative adversarial networks have been adapted to minimise a variety of divergence measures between the model and data distributions (Nowozin et al., 2016; Arjovsky et al., 2017), some of which amount to statistically-robust fitting principles, but they are notoriously fickle to train in practice (Lucic et al., 2018). Henter et al. (2016) instead proposed using the β-divergence to fit models used in speech synthesis, demonstrating a large improvement when training on found data. This approach does not require the use of an adversary. However, the general idea of changing the fitting principle is unattractive with normalising flows, since maximum likelihood is the only strictly proper *local* scoring function (Huszár, 2013, p. 15). This essentially means that all consistent estimation methods not based on MLE take the form of integrals over the observation space. Such integrals are intractable to compute with the normalising flows commonly used today.

The contribution of this paper is instead to robustify flow-based models by changing the parametric family of the distributions we fit to have fatter tails than the conventional Gaussians. Since we still use maximum likelihood for estimation, consistency is assured. This approach has been used to solve inverse problems in stochastic optimisation (Aravkin et al., 2012) and to improve the quality of Google's production text-to-speech systems (Zen et al., 2016). Recently, Jaini et al. (2019) showed that nearly all conventional normalising flows with a Gaussian base are unsuitable for modelling inherently heavy-tailed distributions. However, they do not consider the greater advantages of changing the tail probabilities of the base distribution through the lens of robustness, which extend to data that (like much of the data in our experiments) need not have fat or heavy tails.

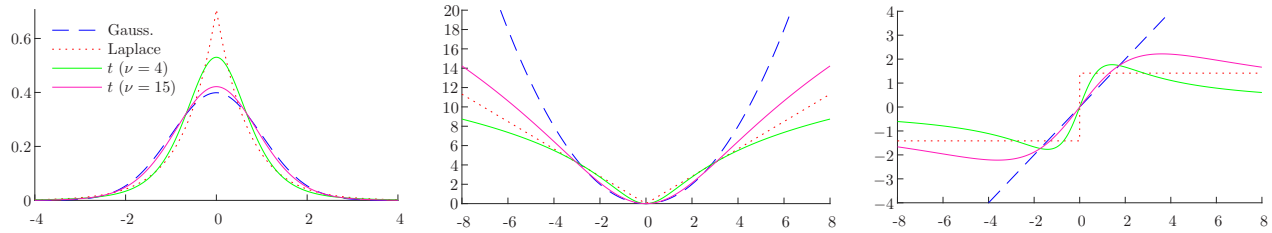
While there are flow-based models with non-Gaussian base distributions such as uniform distributions (Müller et al., 2019) or GMMs (Izmailov et al., 2020; Atanov et al., 2019), these do not have fat tails. To the best of our knowledge, our work represents the first practical exploration of statistical robustness with fat-tailed distributions in normalising flows.

3. Method

This section provides a mathematical analysis of MLE robustness, leading into our proposed solution in Sec. 3.1.

Our overarching goal is to mitigate the impact of outliers in training and test data using robust statistics. We specifically choose to focus on the notion of *resistant statistics*, which are estimators that do not break down under adversarial

[1] While many practitioners informally equate outliers with errors, the treatment in this paper is deliberately agnostic to the origin of these observations. After all, it does not matter whether outliers are simple errors, or represent uncommon but genuine behaviours of the data-generating process, or comprise deliberate corruptions injected by an adversary – as long as the outlying point is in the data, its mathematical effect on our model will be the same.



(a) Probability density functions $p(x)$. (b) Penalty functions $\rho(\varepsilon)$. (c) Influence functions $\psi(\varepsilon)$.

Figure 1: Functions of the normal (dashed), Laplace (dotted), and Student's t distributions (solid) with mean 0 and variance 1.

perturbation of a fraction of the data (arbitrary corruptions only have a bounded effect). For example, among methods for estimating location parameters of distributions, the sample mean is not resistant: By adversarially replacing just a single datapoint in the sample mean, we can make the estimator equal any value we want and make its norm go to infinity. The median, in contrast, is resistant to up to 50% of the data being corrupted.

Informally, being resistant means that we allow the model to "give up" on explaining certain examples, in order to better fit the remainder of the data. This behaviour can be understood through *influence functions* (Hampel et al., 1986). In the special case of maximum-likelihood estimation of location parameters $\boldsymbol{\mu}$, we first define the *penalty function* $\rho(\boldsymbol{\varepsilon})$ as the negative log-likelihood (NLL) loss as a function of $\boldsymbol{\varepsilon} = \boldsymbol{x} - \boldsymbol{\mu}$, offset vertically such that $\rho(\mathbf{0}) = 0$. The influence function $\boldsymbol{\psi}(\boldsymbol{\varepsilon})$ is then just the gradient of ρ with respect to $\boldsymbol{\varepsilon}$. Fig. 1 graphs a number of different distributions in 1D, along with the associated penalty and influence functions. For the Gaussian distribution with fixed scale, the penalty function is the squared error. The resulting $\boldsymbol{\psi}(\boldsymbol{x})$ is a linear function of $\boldsymbol{x}$, as plotted in Fig. 1c, meaning that the extent of the influence of any single outlying datapoint can grow arbitrarily large – the estimator is not resistant. Consequently, using maximum likelihood to fit distributions with Gaussian tails is not statistically robust.

The impact of outliers can be reduced by fitting probability distributions with fatter tails. One example is the Laplace distribution, whose density decays exponentially with the distance from the midpoint $\boldsymbol{\mu}$; see Fig. 1 for plots. The associated penalty is the absolute error $\rho(\boldsymbol{\varepsilon}) = \|\boldsymbol{\varepsilon}\|_2$. This is minimised by the median, which is resistant to adversarial corruptions. The Laplacian influence function in the figure is seen to be a step function and thus remains bounded everywhere, confirming that the median is resistant. This is similar to the effect of gradient clipping in that the influence of outliers can never exceed a certain maximal magnitude.

3.1. Proposed solution

Define a *flow* as a parametric family of densities $\{\boldsymbol{X} = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{Z})\}_{\boldsymbol{\theta}}$, where $\boldsymbol{f}_{\boldsymbol{\theta}}$ is an invertible transformation that depends on the *parameters* $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and $\boldsymbol{Z}$ is a fixed base distribution. Our general proposal is to gain statistical robustness in this model by replacing the traditional multivariate normal base distribution by a distribution with a bounded influence function. Our specific proposal (studied in detail in our experiments) is to replace $\boldsymbol{Z}$ by a multivariate t-distribution, $t_\nu(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, building on Lange et al. (1989). The use of multivariate t-distributions in flows was studied theoretically but not empirically by Jaini et al. (2019)[2] for the special case of triangular flows on inherently heavy-tailed data.

The pdf of the t_ν-distribution in D dimensions is

$$p_t(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \nu) = \Gamma\left(\tfrac{\nu+D}{2}\right)\left(\Gamma\left(\tfrac{\nu}{2}\right)\right)^{-1}|\nu\pi\boldsymbol{\Sigma}|^{-\frac{1}{2}} \cdot \left(1 + \tfrac{1}{\nu}(\boldsymbol{x}-\boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)^{-\frac{\nu+D}{2}}, \quad (1)$$

where the scalar $\nu > 0$ is called the *degrees of freedom*. We see in Fig. 1 that this leads to a nonconvex penalty function and, importantly, to an influence function that approaches zero for large deviations. This is known as a *redescending* influence function, and means that outliers not only have a bounded impact in general (like for the absolute error or gradient clipping), but that gross outliers furthermore will be effectively ignored by the model. Since the density asymptotically decays polynomially (i.e., slower than exponentially), we say that it has *fat tails*. Seeing that the (inverse) transformation $\boldsymbol{f}_{\boldsymbol{\theta}}^{-1}$ now no longer turns the observation distribution $\boldsymbol{X}$ into a normal (Gaussian) distribution, we propose to call these models *Studentising flows*.

As our proposal is based on MLE, we retain both consistency and efficiency in the absence of misspecification. In the face of outlying observations, our approach degrades gracefully, in contrast to distributions having, e.g., Gaussian tails. As we only change the base distribution, our proposal can be combined with any invertible transformation, network architecture, and optimiser to model distributions on $\mathbb{R}^D$. It can also be used with conditional invertible transformations in order to describe conditional probability distributions. Since

[2] Recent follow-up work by Jaini et al. (2020), appearing concurrently with our paper, does contain empirical studies of the effect of t_ν base distributions on the tail properties of flows.

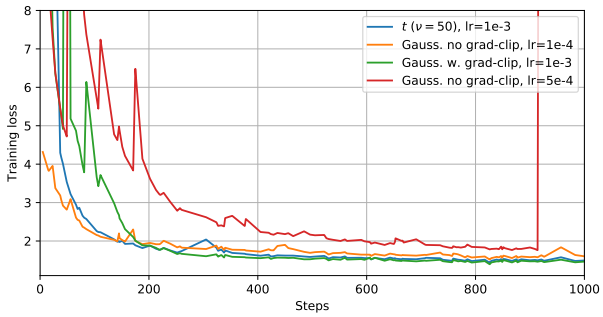


Figure 2: Training loss (NLL) on CelebA data.

the tails of $t_\nu(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ get slimmer as ν increases, we can tune the degree of robustness of the approach by changing this parameter of the distribution.[3] In fact, the distribution converges on the multivariate normal $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ in the limit $\nu \to \infty$. Sampling from the t_ν-distribution can be done by drawing a sample from a multivariate Gaussian and then scaling it on the basis of a sample from the scalar χ^2_ν-distribution; see Kotz & Nadarajah (2004).

4. Experiments

In this section we demonstrate empirically the advantages of fat-tailed base distributions in normalising flows, both in terms of stable training and for improved generalisation.

4.1. Experiments on image data

Our initial experiments considered unconditional models of (uniformly dequantised) image data using Glow (Kingma & Dhariwal, 2018). Specifically, we used the benchmark code from Durkan et al. (2019) trained using Adam (Kingma & Ba, 2015). Implementing t_ν-distributions for the base required just 20 lines of Python code; see Appendix B.

First we investigated training stability on the CelebA faces dataset (Liu et al., 2015). We used the benchmark distributed by Durkan et al. (2019), which considers 64×64 images to reduce computational demands. Our model and training hyperparameters were closely based on those used in the Glow paper, setting $K = 32$ and $L = 3$ like for the smaller architectures in the article. We found that without gradient clipping, training Glow on CelebA required low learning rates to remain stable. As seen in Fig. 2, training with learning rate $\mathrm{lr} = 10^{-4}$ was stable, but training with higher learning rates $\mathrm{lr} \geq 0.5 \cdot 10^{-3}$ did not converge. Clipping the gradient norm at 100, or our more principled approach of changing the base to a multivariate t_ν-distribution (with $\nu = 50$), both enabled successful training at $\mathrm{lr} = 10^{-3}$. We also reached better log-likelihoods on held-out data than the model trained with low learning rate (see Fig. 5 in Appendix A), even though the primary goal of this experiment was not necessarily to demonstrate better generalisation.

[3] It is also possible to treat ν as a learnable model parameter rather than a fixed or hand-tuned hyperparameter, but this procedure is not theoretically robust to gross outliers (Lucas, 1997).

Table 1: Test-set NLL losses on MNIST with and without outliers inserted from CIFAR-10. Δ-values are w.r.t. the corresponding Gaussian alternative ($\nu = \infty$).

	Test	Clean				1% outliers			
Train	$\nu =$	∞	20	50	1000	∞	20	50	1000
Clean	NLL	1.16	1.13	1.13	1.17	1.63	1.27	1.26	1.31
	Δ	0	-0.03	-0.03	0.01	0	-0.36	-0.37	-0.32
1% outliers	NLL	1.17	1.13	1.14	1.18	1.21	1.18	1.19	1.22
	Δ	0	-0.04	-0.03	0.01	0	-0.03	-0.02	0.01

Next we performed experiments on the widely-used MINST dataset (LeCun et al., 1998) to investigate the effect of outliers on generalisation. Since pixel intensities are bounded, image data in general does not have asymptotically fat tails. But while MNIST is considered a quite clean dataset, we can deliberately corrupt training and/or test data by inserting greyscale-converted examples from CIFAR-10 (Krizhevsky, 2009), which contains natural images that are much more diverse than the handwritten digits of MNIST. We randomly partitioned MNIST into training, validation, and test sets (80/10/10 split), and considered four combinations of either clean or corrupted (1% CIFAR) test and/or train+val data. We trained (60k steps) and tested normalising as well as Studentising flows on the four combinations, using the the same learning rate schedule (cosine decay from $\mathrm{lr} = 4 \cdot 10^{-4}$ to 10^{-4}) and hyperparameters ($K = 10$, $L = 3$), and clipping the gradients for the normalising flows only. This produced the negative log-likelihood values listed in Table 1. We see that, for each configuration, the proposed method performed similar to or better than the conventional setup using Gaussian base distributions. The generalisation behaviour of t_ν-distributions was not sensitive to the parameter ν, although very high values ($\nu \approx 1000$ or more) behaved more like the conventional normalising flow, as expected. While in most cases the improvements were relatively minor, Studentising flows generalised much better to the case where the test data displayed behaviours not seen during training.

4.2. Experiments on motion modelling

Last, we studied a domain where normalising flows constitute the current state of the art, namely conditional probabilistic motion modelling as in Henter et al. (2019); Alexanderson et al. (2020). These models resemble the VideoFlow model of Kumar et al. (2020), but also include recurrence and an external control signal. The models give compelling visual results, but have been found to overfit significantly in terms of the log-likelihood on held-out data. This reflects a well-known disagreement between likelihood and subjective impressions; see, e.g., Theis et al. (2016): Humans are much more sensitive to the presence of unnatural output

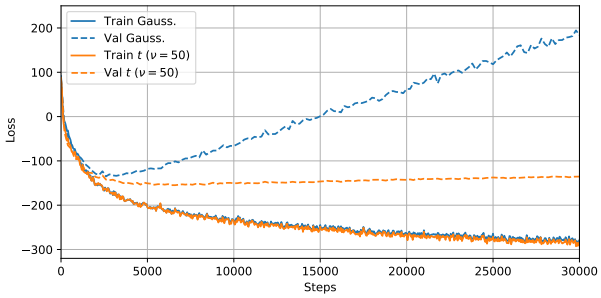


Figure 3: Training and validation losses on locomotion data.

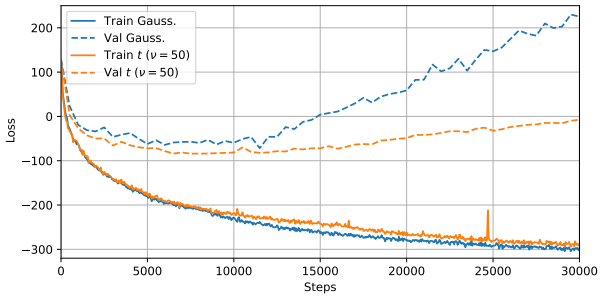


Figure 4: Training and validation losses on gesture data.

examples than they are to *mode dropping*, where models do not represent all possibilities the data can show. Non-robust approaches (which cannot "give up" on explaining even a single observation), on the other hand, suffer significant likelihood penalties upon encountering unexpected examples in held-out data; cf. Table 1. Having methods where generalisation performance better reflects subjective output quality would be beneficial, e.g., when tuning generative models.

We considered two tasks: locomotion generation with path-based control and speech-driven gesture generation. For locomotion synthesis, the input is a sequence of delta translations and headings specifying a motion path along the ground, and the output is a sequence of body poses (3D joint positions) that animate human locomotion along that path. For gesture synthesis, the input is a sequence of speech-derived acoustic features and the output is a sequence of body poses (joint angles) of a character gesticulating and changing stance to the speech. In both cases, the aim is to use motion-capture data to learn to animate plausible motion that agrees with the input signal. See Appendix A for still images and additional information about the data.

For the gesture task we used the same model and parameters as system FB-U in Alexanderson et al. (2020). For the locomotion task, we found that additional tuning of the MG model from Henter et al. (2019) could maintain the same visual quality while reducing training time and improving performance on held-out data. Specifically, we applied a Noam learning-rate scheduler (Vaswani et al., 2017) with peak $\mathrm{lr} = 10^{-3}$ decaying to 10^{-4}, set data dropout to 0.75, and changed the recurrent network from an LSTM to a GRU.

Learning curves for the two tasks are illustrated in Fig. 3 and show similar trends. Under a Gaussian base distribution, the loss on training data decreases, while the NLL on held-out data begins to rise steeply early on during training.[4] This is subjectively misleading, since the perceived quality of randomly-sampled output motion generally keeps improving throughout training. We note that these normalising flows were trained with gradient clipping (both of the norm and individual elements), and the smooth shape of the curves around the local optimum makes it clear that training instability is not a factor in the poor performance.

Using the same models and training setups but with our proposed t_ν-distribution ($\nu = 50$) for the base has essentially no effect on the training loss but brings the validation curves much closer to the training curves. It is also significantly less in disagreement with subjective impressions of the quality of random motion samples with held out control-inputs. While these plots only show the first 30k training steps, the same trends continue over the full 80k+ steps we trained, with normalising flows diverging linearly while the validation losses of Studentising flows quickly saturate; see Fig. 8 in Appendix A.

5. Conclusion

We have proposed fattening the tails of the base (latent) distributions in flow-based models. This leads to a modelling approach that is statistically robust: it remains consistent and efficient in the absence of model misspecification while degrading gracefully when data and model do not match. We have argued that many heuristic steps in standard machine-learning pipelines, including the practice of gradient clipping during optimisation, can be seen as workarounds for core modelling approaches that lack robustness. Our experimental results demonstrate that changing to a fat-tailed base distribution 1) provides a principled way to stabilise training, similar to what gradient clipping does, and 2) improves generalisation, both by reducing the mismatch between training and validation loss and by improving the log-likelihood of held-out data in absolute terms. These improvements are observed for well-tuned models on datasets both with and without obviously extreme observations. We expect the improvements due to increased robustness to be of interest to practitioners in a wide range of applications.

[4] We have been able to replicate similarly-shaped learning curves on CelebA by changing the balance to 20% training data and 80% validation data (see Fig. 6 in Appendix A), suggesting that the root cause of this divergent behaviour is an amount of training data that is too small to adequately sample the full diversity of natural behaviour, leading to a poor model of held-out material. This is despite the fact that the motion databases used for these experiments are among the largest currently available for public use. In classification, Recht et al. (2019) recently highlighted similar issues of poor generalisation on new data from the same source.

Acknowledgement

This work was supported by the Swedish Research Council proj. 2018-05409 (StyleBot) and by the Wallenberg AI, Autonomous Systems and Software Program (WASP) of the Knut and Alice Wallenberg Foundation, Sweden.

References

Alexanderson, S., Henter, G. E., Kucherenko, T., and Beskow, J. Style-controllable speech-driven gesture synthesis using normalising flows. *Comput. Graph. Forum*, 39(2):487–496, 2020. 4, 5

Aravkin, A., Friedlander, M. P., Herrmann, F. J., and van Leeuwen, T. Robust inversion, dimensionality reduction, and randomized sampling. *Math. Program., Ser. B*, 134: 101–125, 2012. 2

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *Proc. ICML*, pp. 214–223, 2017. 2

Atanov, A., Volokhova, A., Ashukha, A., Sosnovik, I., and Vetrov, D. Semi-conditional normalizing flows for semi-supervised learning. In *Proc. INNF*, 2019. 2

Bishop, C. M. Mixture density networks. Technical Report NCRG/94/004, Aston University, Birmingham, UK, 1994. 1

Brock, A., Donahue, J., and Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In *Proc. ICLR*, 2019. 2

CMU Graphics Lab. Carnegie Mellon University motion capture database. http://mocap.cs.cmu.edu/, 2003. 8

Daniels, H. E. The asymptotic efficiency of a maximum likelihood estimator. In *Fourth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pp. 151–163. University of California Press Berkeley, 1961. 1

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. Neural spline flows. *Proc. NeurIPS*, pp. 7509–7520, 2019. https://github.com/bayesiains/nsf. 4

Ferstl, Y. and McDonnell, R. Investigating the use of recurrent motion modelling for speech gesture generation. In *Proc. IVA*, pp. 93–98, 2018. http://trinityspeechgesture.scss.tcd.ie. 8

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Proc. NIPS*, pp. 2672–2680, 2014. 1

Habibie, I., Holden, D., Schwarz, J., Yearsley, J., and Komura, T. A recurrent variational autoencoder for human motion synthesis. In *Proc. BMVC*, pp. 119.1–119.12, 2017. 8

Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. *Robust Statistics: The Approach Based on Influence Functions*. John Wiley & Sons, Inc., 1986. 2, 3

Henter, G. E. and Kleijn, W. B. Minimum entropy rate simplification of stochastic processes. *IEEE T. Pattern Anal.*, 38(12):2487–2500, 2016. 2

Henter, G. E., Ronanki, S., Watts, O., Wester, M., Wu, Z., and King, S. Robust TTS duration modelling using DNNs. In *Proc. ICASSP*, pp. 5130–5134, 2016. 2

Henter, G. E., Alexanderson, S., and Beskow, J. MoGlow: Probabilistic and controllable motion synthesis using normalising flows. *arXiv preprint arXiv:1905.06598*, 2019. 4, 5, 8

Huber, P. J. and Ronchetti, E. M. *Robust Statistics*. John Wiley & Sons, Inc., 2nd edition, 2009. 2

Huszár, F. *Scoring rules, Divergences and Information in Bayesian Machine Learning*. PhD thesis, University of Cambridge, Cambridge, UK, 2013. https://github.com/fhuszar/thesis. 2

Izmailov, P., Kirichenko, P., Finzi, M., and Wilson, A. G. Semi-supervised learning with normalizing flows. In *Proc. ICML*, 2020. 2

Jaini, P., Kobyzev, I., Brubaker, M., and Yu, Y. Tails of triangular flows. *arXiv preprint arXiv:1907.04481v1*, 2019. 2, 3

Jaini, P., Kobyzev, I., Yu, Y., and Brubaker, M. Tails of Lipschitz triangular flows. In *Proc. ICML*, 2020. 3

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015. 4

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Proc. NeurIPS*, pp. 10236–10245, 2018. 2, 4

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *Proc. ICML*, pp. 1885–1894, 2017. 2

Kotz, S. and Nadarajah, S. *Multivariate t Distributions and Their Applications*. Cambridge University Press, 2004. 4

Krizhevsky, A. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, Toronto, ON, Canada, 2009. https://www.cs.toronto.edu/~kriz/cifar.html. 4

Kumar, M., Babaeizadeh, M., Erhan, D., Finn, C., Levine, S., Dinh, L., and Kingma, D. VideoFlow: A conditional flow-based model for stochastic video generation. In *Proc. ICLR*, 2020. 4

Lange, K. L., Little, R. J. A., and Taylor, J. M. G. Robust statistical modeling using the t distribution. *J. Am. Stat. Assoc.*, 84(408):881–896, 1989. 3

LeCun, Y., Cortes, C., and Burges, C. J. C. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998. 4

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proc. ICCV*, 2015. http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html. 4

Lucas, A. Robustness of the Student t based M-estimator. *Commun. Statist.–Theory Meth.*, 26(5):1165–1182, 1997. 4

Lucas, T., Shmelkov, K., Alahari, K., Schmid, C., and Verbeek, J. Adaptive density estimation for generative models. In *Proc. NeurIPS*, pp. 11993–12003, 2019. 2

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. Are GANs created equal? A large-scale study. In *Proc. NeurIPS*, pp. 700–709, 2018. 2

Mohamed, S. and Lakshminarayanan, B. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016. 1

Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., and Weber, A. Documentation mocap database HDM05. Technical Report CG-2007-2, Universität Bonn, Bonn, Germany, 2007. http://resources.mpi-inf.mpg.de/HDM05/. 8

Müller, T., Mcwilliams, B., Rousselle, F., Gross, M., and Novák, J. Neural importance sampling. *ACM Trans. Graph.*, 38(5):145:1–145:19, 2019. 2

Nowozin, S., Cseke, B., and Tomioka, R. f-GAN: Training generative neural samplers using variational divergence minimization. In *Proc. NIPS*, pp. 271–279, 2016. 2

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019. 1

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do ImageNet classifiers generalize to ImageNet? *Proc. ICML*, pp. 5389–5400, 2019. 5

Theis, L., van den Oord, A., and Bethge, M. A note on the evaluation of generative models. *Proc. ICLR*, 2016. 4

Ting, D. and Brochu, E. Optimal subsampling with influence functions. In *Proc. NeurIPS*, pp. 3650–3659, 2018. 2

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Proc. NIPS*, pp. 5998–6008, 2017. 5

Zen, H., Agiomyrgiannakis, Y., Egberts, N., Henderson, F., and Szczepaniak, P. Fast, compact, and high quality LSTM-RNN based statistical parametric speech synthesizers for mobile devices. In *Proc. Interspeech*, pp. 2273–2277, 2016. 2

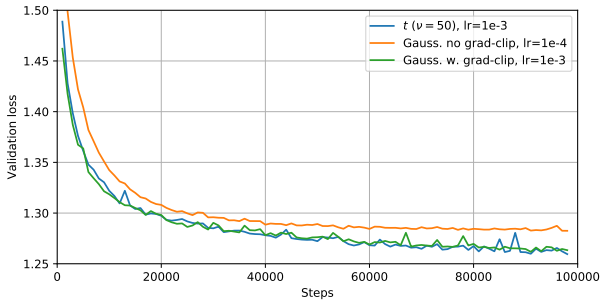


Figure 5: Validation loss on CelebA with different setups.

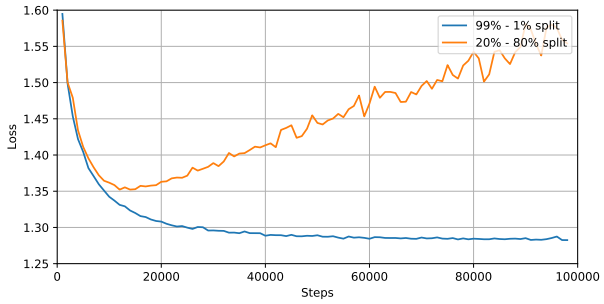


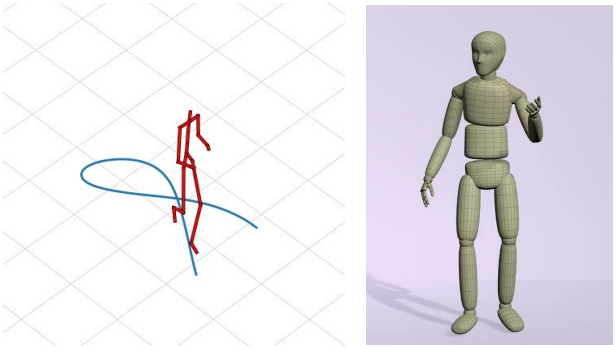
Figure 6: Validation loss on CelebA with different splits.

A. Additional information on data and results

Fig. 5 reports the validation-set performance over 100k steps of training for the three stable systems from Fig. 2. We see that systems trained with the higher learning rate gave noticeably better generalisation performance.

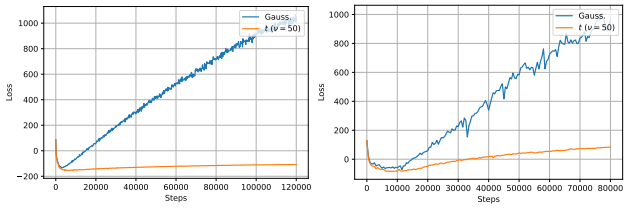
We also performed an experiment on CelebA to see the effect of reduced training-data size on generalisation. In particular, we tried making the training set significantly smaller than before (going from 99% to 20% of the database), while making the validation set much larger (from 1% to 80% of the database) in order to well sample the full diversity of the material. Fig. 6 shows learning curves on the CelebA data with Gaussian base distributions before and after shifting the balance between training and held-out data. We see that, while validation loss originally decreased monotonically, the loss after changing dataset sizes instead reaches an optimum early on in the training and then begins to rise significantly again, reminiscent of the validation curves seen in Sec. 4.2. We conclude that the unusually large generalisation gap on the motion data at least in part can be attributed to the size of the database relative to the complexity of the task.

The two motion-data modelling tasks we considered in Sec. 4.2, namely path-based locomotion control and speech-driven gesture generation, have applications in areas such as animation, computer games, embodied agents, and social robots. For the locomotion data, we used the Edinburgh locomotion MOCAP database (Habibie et al., 2017) pooled with the locomotion trials from the trials from the CMU (CMU Graphics Lab, 2003) and HDM05 (Müller et al., 2007) motion-capture databases. Each frame in the data had an output dimensionality of $D = 63$. Gesture-generation models, meanwhile, were trained on the Trinity Gesture Dataset collected by Ferstl & McDonnell (2018), which is a large database of joint speech and gestures. Each output frame had $D = 65$ dimensions. Fig. 7 shows still images from representative visualisations of the two tasks. Like for image data, the numerical range of these motion datasets is bounded in practice (e.g., by the finite length of human bones coupled with the body-centric coordinate systems used in Henter et al. (2019)), and the data is not known to contain any numerically extreme observations.



(a) Locomotion with control path. (b) Gesticulating avatar.

Figure 7: Snapshots visualising the motion data used.



(a) Locomotion modelling task. (b) Gesture modelling task.

Figure 8: Validation-loss curves of extended training.

Fig. 8 illustrates the point from the end of Sec. 4.2 regarding the growing gap between normalising and Studentising flows over the course of the entire training. We see that the held-out loss of the former diverges essentially linearly, while the proposed method shows saturating behaviour.

B. PyTorch code for the t_ν-distribution

We here reproduce our implementation of log-probability computation and sampling with the multivariate t_ν-distribution:

```
import numpy as np
import scipy.special
import torch

class StudentT():
    def __init__(self, shape, nu=50):
        d = shape[0]
        self._const = scipy.special.loggamma(0.5*(nu+d)) - \
              scipy.special.loggamma(0.5*nu) - 0.5*d*np.log(np.pi*nu)
        self._shape = torch.Size(shape)
        self._nu = nu

    def _log_prob(self, inputs):
        d = self._shape[0]
        input_norms = utils.sum_except_batch(((inputs)**2), num_batch_dims=1)
        likelihood = self._const - \
            0.5*(self._nu+d)*torch.log(1+(1/self._nu)*input_norms)
        return likelihood

    def _sample(self, num_samples):
        d = self._shape[0]
        x = np.random.chisquare(self._nu, num_samples)/self._nu
        x = np.tile(x[:,None], (1,d))
        x = torch.Tensor(x.astype(np.float32))
        z = torch.randn(num_samples, *self._shape)
        return (z/torch.sqrt(x))
```