# Supplementary Materials: Hybrid Models with Deep and Invertible Features

**Eric Nalisnick** [*1] **Akihiro Matsukawa** [*1] **Yee Whye Teh** [1] **Dilan Gorur** [1] **Balaji Lakshminarayanan** [1]

## A. Background on Affine Coupling Layers

One ACL performs the following operations (Dinh et al., 2017):

1. *Splitting*: $\boldsymbol{x}$ split it at dimension $d$ into two separate vectors $\boldsymbol{x}_{:d}$ and $\boldsymbol{x}_{d:}$ (using Python list syntax).
2. *Identity and Affine Transformations*: Given the split $\{\boldsymbol{x}_{:d}, \boldsymbol{x}_{d:}\}$, each undergoes a separate operation:

$$\texttt{identity} : \boldsymbol{h}_{:d} = \boldsymbol{x}_{:d} \tag{1}$$
$$\texttt{affine} : \boldsymbol{h}_{d:} = t(\boldsymbol{x}_{:d}; \boldsymbol{\phi}_t) + \boldsymbol{x}_{d:} \odot \exp\{s(\boldsymbol{x}_{:d}; \boldsymbol{\phi}_s)\}$$

   where $t(\cdot)$ and $s(\cdot)$ are translation and scaling operations with no restrictions on their functional form. We can compute them with neural networks that take as input $\boldsymbol{x}_{:d}$, the other half of the original vector, and since $\boldsymbol{x}_{:d}$ has been copied forward by the first operation, no information is lost that would jeopardize invertibility.
3. *Permutation*: Lastly, the new representation $\boldsymbol{h} = \{\mathbf{h}_{:d}, \mathbf{h}_{d:}\}$ is ready to be either treated as output or fed into another ACL. If the latter, then the elements should be modified so that $\mathbf{h}_{:d}$ is not again copied but rather subject to the affine transformation. Dinh et al. (2017) simply exchange the components (i.e. $\{\mathbf{h}_{d:}, \mathbf{h}_{:d}\}$) whereas Kingma & Dhariwal (2018) apply a $1 \times 1$ convolution, which can be thought of as a continuous generalization of a permutation.

Several ACLs are composed to create the final form of $f(\boldsymbol{x}; \boldsymbol{\phi})$, which is called a *normalizing flow* (Rezende & Mohamed, 2015). Crucially, the Jacobian of these operations is efficient to compute, simplifying to the sum of all the scale transformations:

$$\log \left| \frac{\partial \boldsymbol{f}_{\boldsymbol{\phi}}}{\partial \boldsymbol{x}} \right| = \log \exp \left\{ \sum_{l=1}^{L} s_l(\boldsymbol{x}_{:d}; \boldsymbol{\phi}_{s,l}) \right\}$$
$$= \sum_{l=1}^{L} s_l(\boldsymbol{x}_{:d}; \boldsymbol{\phi}_{s,l})$$

where $l$ is an index over ACLs. The Jacobian of a $1 \times 1$ convolution does not have as simple of an expression, but Kingma & Dhariwal (2018) describe ways to reduce computation.

[*]Equal contribution [1]DeepMind. Correspondence to: Balaji Lakshminarayanan <balajiln@google.com>.

*Proceedings of the 36$^{th}$ International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019. 

## B. Additional Semi-Supervised Results

### B.1. Hyperparameters for Semi-supervised learning

The hyperparameters are described in Table S1.

| Hyper-parameter | Grid values |
|---|---|
| Dropout rate | 0, 0.2, 0.5 |
| $\epsilon_{\mathsf{VAT}}$ for Virtual Adversarial Training | 1, 5 |
| $\lambda_{\mathsf{EM}}$ for Entropy Minimization loss | 0, 0.3, 1, 3 |
| $\lambda_{\mathsf{VAT}}$ for Virtual Adversarial Training loss | 0, 0.3, 1, 3 |

*Table S1.* Hyperparameters for the semi-supervised learning experiments.

### B.2. Semi-supervised learning on SVHN

The results are shown in Table S2. Similar to the semi-supervised results on MNIST, we observe that our model can effectively leverage unlabeled data.

| Model | SVHN-Error ↓ | SVHN-NLL ↓ |
|---|---|---|
| 1000 labels only | 19.26% | 0.78 |
| 1000 labels + unlabeled | 5.90% | 0.38 |
| All labeled | 4.86% | 0.26 |
| All labeled + unlabeled | **2.80%** | **0.17** |

*Table S2.* Results of hybrid model for semi-supervised learning on SVHN. Arrows indicate which direction is better.

## C. Extensions

### C.1. Mixed Effects Model

To model heteroscedastic noise, we can also add "random effects" to the model at the latent level. The model is then:

$$\mathbb{E}[y_n|\mathbf{x}_n] = g^{-1}\left(\boldsymbol{\beta}^T f(\mathbf{x}_n; \boldsymbol{\phi}) + \mathbf{u}^T \mathbf{a}_n\right),$$
$$\mathbf{u} \sim p(\mathbf{u}), \quad \mathbf{a}_n \sim p(\mathbf{a})$$

where $\mathbf{a}_n$ is a vector of random effects associated with the fixed effects $\mathbf{x}_n$, and $\mathbf{u}$ are the corresponding parameters in the GLM. Note that Depeweg et al. (2018) also use random effects model to handle heteroscedastic noise, but add them at the input level.

## References

Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning. In *ICML*, 2018.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density Estimation Using Real NVP. In *International Conference on Learning Representations (ICLR)*, 2017.

Kingma, D. P. and Dhariwal, P. Glow: Generative Flow with Invertible 1x1 Convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Rezende, D. and Mohamed, S. Variational Inference with Normalizing Flows. In *International Conference on Machine Learning (ICML)*, 2015.