Learning Invariant Weights in Neural Networks

Tycho F.A. van der Ouderaa [1] **Mark van der Wilk** [1]

Abstract

It is well-known that assumptions about invariances or symmetries in data can seriously increase the predictive power of statistical models. Many commonly used models in machine learning have certain symmetries built-in, such as translation equivariance in convolutional neural networks, and incorporation of new symmetry types are actively being studied. Yet, efforts to *learn* such invariances from the data itself are in their early days and inferring invariances remains an open research problem. It has been shown that marginal likelihood offers a principled way to learn invariances in Gaussian Processes. We propose a weight-space equivalent to this approach, by minimizing a lower bound on the marginal likelihood to learn invariances in neural networks.

1. Introduction

Intuitively, invariances allow models to extrapolate (or 'generalize') beyond training data (see Figure 1). An invariant model does not change in output when the input is changed by transformations to which it is deemed invariant. The most straightforward way to achieve this is perhaps by enlarging the dataset with transformed examples: a process known as data augmentation. A link between invariance and data augmentation in kernel space has been made by Dao et al. (2019). In this study, we show that this invariance can exactly be described as transformations on the weights, similar to Cohen & Welling (2016). We then continue and adopt the principled method of marginal likelihood that has proven effective in learning invariance in Gaussian Processes (GPs) (van der Wilk et al., 2018). To allow invariance learning in neural networks, we propose to optimize a lower bound on the marginal likelihood, and succesfully learn weights with the correct amount of rotational invariance when trained on rotated versions of MNIST.

[1]Department of Computation, Imperial College London, London, United Kingdom. Correspondence to: Tycho F.A. van der Ouderaa <tycho.vanderouderaa@imperial.ac.uk>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning., Copyright 2021 by the author(s).

2. Related Work

Convolutional neural networks (CNNs) have been successful on wide range of problems and have played a key role in the success of Deep Learning (LeCun et al., 2015). It is commonly understood that the translational symmetries that arise from effective weight-sharing in CNNs is an important driver for its outstanding performance on many tasks.

In Cohen & Welling (2016), a group-theoretical framework was proposed that extends CNNs beyond translational symmetries, and demonstrated this for discrete group actions. Many studies since have proposed ways to incorporate other symmetries, such as continuous rotation, scale and translation, into the weights of neural networks (Worrall et al., 2017; Weiler et al., 2018; Marcos et al., 2017; Esteves et al., 2017; Weiler & Cesa, 2019; Bekkers, 2019) and recent efforts allow practical equivariance in neural networks for arbitrary symmetry groups (Finzi et al., 2021).

Although incorporating the right symmetries in the weights often results in better performing models, the invariances (or 'equivariances') are typically fixed and must be specified beforehand. In this study, we aim to learn invariances from the data itself by optimizing the marginal likelihood: the common method in Bayesian statistics to perform model selection, which can be interpret as exhaustive leave-p-out cross-validation averaged over all values of p and held-out test sets (Fong & Holmes, 2020).

Some other works have considered learning data augmentations (Jaderberg et al., 2015; Cubuk et al., 2018), but without utilizing marginal likelihood (or lower bound thereof) and require additional validation data. In Schwöbel et al. (2020) and Benton et al. (2020) a lower bound is considered. Nevertheless, these approaches learn invariance through transformations on the input - thus a type of data augmentation - rather than through transformations on the weights.

3. On Invariant Modelling

A model $f(\cdot)$ is deemed 'strictly invariant' when the output is unaffected by a set of transformations: $f(T_g \circ \boldsymbol{x}) = f(\boldsymbol{x}), \forall g \in G, \boldsymbol{x} \in \mathcal{X}$ of which each transformation T_g is governed by a group action $g \in G$ forming a group. We can obtain an invariant model by averaging the outputs over every transformation T_g. Although group theory introduces

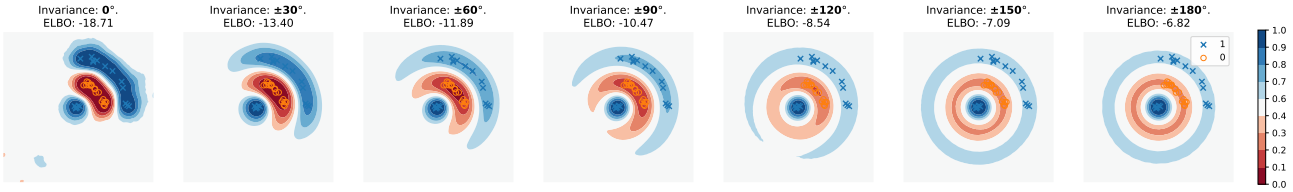


Figure 1. Illustration of different extrapolating behaviour further away from 2d toy data for models with no invariance (left), some invariance (middle) up to strict invariance (right). Contour plot shows model prediction, datapoints are shown with ×'s and ○'s.

a rigid mathematical framework that is often used to describe and incorporate symmetries in statistical and machine learning models, it is restricted in the sense that the set of transformations that generate a group is always closed, by definition of a group. To illustrate, imagine the classic MNIST image recognition problem (LeCun et al., 1998): here invariance to rotations up to a certain angle allows for better extrapolation to tilded versions of fitted digits and thus more robust predictions and increased sample efficiency. However, invariance to full 360 degree rotations (all SO(2) group actions) may prohibit us from differentiating between a '6' and a '9'. In an effort to overcome this issue, we construct our invariant function $f\ (\boldsymbol{x})$ from non-invariant function $g\ (\boldsymbol{x})$ by summing over the orbit with a compactly supported probability distribution $p(T)$ over invariant transformations:

$$f\ (\boldsymbol{x}) = \int g\ (T(\boldsymbol{x}))p(T)\mathrm{d}T \tag{1}$$

Hereby, we hope to induce a relaxed notion of invariance on the model, sometimes referred to as 'insensitivity' (van der Wilk et al., 2018) or 'soft-invariance' (Benton et al., 2020) upon the model, of which 'strict invariance' is the special case where we take the orbit over the entire group.

3.1. Invariant Neural Network Set-up

To ensure we parameterize invariance such that it can automatically be learned from data, we start by considering a two-layer neural network that is an approximate (exact in the inifitely wide limit) weight-space equivalent of the Gaussian Process that has proven capable of learning invariance with marginal likelihood in (van der Wilk et al., 2018). In this case, our neural network would look like:

$$\begin{aligned} f\ (\boldsymbol{x}) &= \mathsf{E}_{T\ \ p(T)}\left[\boldsymbol{W}_2 \circ \phi\left(\boldsymbol{W}_1 \circ T \circ \boldsymbol{x}\right)\right] \\ p(y|\boldsymbol{\theta}, \mathcal{D}) &= \sigma\ \ \mathsf{E}_{T\ \ p(T)}\left[\boldsymbol{W}_2 \circ \phi\left(\boldsymbol{W}_1 \circ T \circ \boldsymbol{x}\right)\right] \end{aligned} \tag{2}$$

where $\sigma(\cdot)$ is the soft-argmax function, $\boldsymbol{x}$ is the input, first layer weights $\boldsymbol{W}_1$ and bias[1] are initialized as random Fourier features (RFF) (Rahimi et al., 2007), non-linearity $\phi = \cos(\cdot)$ is chosen to be element-wise cosine function, and we learn a Bayesian posterior estimate of weight matrix $\boldsymbol{W}_2$ with bias[1]. Later in the study, we also try learning invariances in a more common neural network with a ReLU activation function $\phi = \max(0, x)$ where both layers are learned and observe that the constraints may not always be necessary to obtain a tight enough bound on the marginal likelihood to learn invariance.

Note that in this set-up $(\boldsymbol{W}_1^T \circ T) \circ \boldsymbol{x} = \boldsymbol{W}_1^T \circ (T \circ \boldsymbol{x})$ are equal, by associtivity of matrix transformations. In other words, first applying transformation T on the weights, similar to the typical construction of equivariant layers, and first applying them to the input, which could be interpret as built-in data augmentation, are mathematically equivalent. In practice, however, differences between the two could still arise from approximations required when applying T (e.g. interpolation for rotation in discrete grid-space). In our experiments we will consider transforming the weights to demonstrate that invariance can be 'built into' the model.

3.2. Variational Inference

Calculating the exact marginal likelihood is often intractable for neural networks. Therefore, we utilize stochastic variational inference (Hoffman et al., 2013) to derive and optimize an *evidence lower bound* of the marginal likelihood and minimize the D_{KL}-divergence between a full-covariance variational Gaussian distribution $q\ (\boldsymbol{W}_2|\boldsymbol{\mu},\ \ )$ and the true posterior. We assume a Gaussian prior $\mathcal{N}(\boldsymbol{w}_2^c|\mathbf{0}, \boldsymbol{I}\alpha)$ on $\boldsymbol{w}_2^c$ for each class c. We obtain a Monte Carlo estimate by sampling L times from the variational distribution, utilizing the reparameterization trick (Kingma & Welling, 2013), and minimize the following loss function:

$$\mathcal{L} = \underbrace{D_{\mathsf{KL}}(q\ (\boldsymbol{W}_2|\boldsymbol{\mu},\ \ )||p(\boldsymbol{W}_2))}_{\text{Closed-form KL}} + \frac{1}{L}\sum_{l}\underbrace{-\log p(\mathcal{D}|w)}_{\text{Cross-entropy}}$$

where we can choose $L = 1$ given a sufficiently large batch size with Stochastic Gradient Variational Bayes (SGVB) loss estimate $\frac{N}{M}\sum_{i=1}^{M}\tilde{\mathcal{L}}(\boldsymbol{\theta}, \{\boldsymbol{x}_i\}, \{y_i\})$ (Kingma & Welling, 2013). Full derivations in Appendix A.

[1]In our experiments, we include bias weights, which can be implemented by concatenating the input features with an additional 'one'-element and is omitted in notation for clarity.

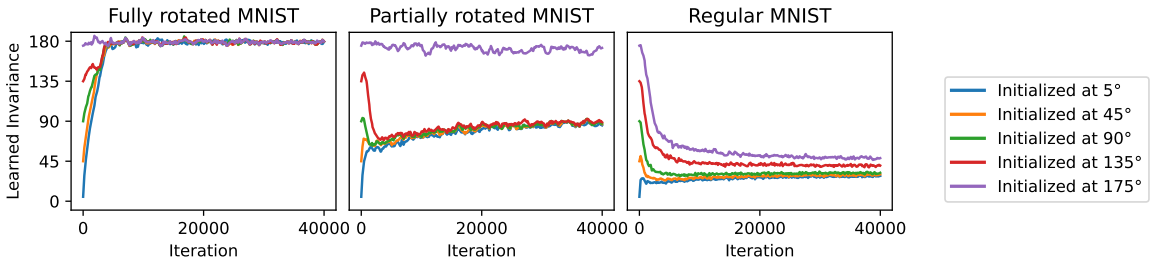


Figure 2. Predicted invariance over training iterations for models with degrees of invariance at initialization trained on variations of MNIST that have been randomly rotated to varying degree: full rotations (left), half rotations (middle) and original dataset (right).

3.3. Lie Group Reparameterization

We utilize the re-parameterization trick (Kingma & Welling, 2013) to parameterize a Lie Group, by scaling its infitesimal generator $A(t)$ with a Uniform distribution $U[-v, v]$, similar to re-parameterizations of Lie Groups in (Falorsi et al., 2019). Hereby, we obtain a distribution over invariant transformations $p_v(T)$ parameterized by scalar v from which we can obtain N differentiable samples:

$$\{e^{A(t_1)}, e^{A(t_2)}, \cdots e^{A(t_N)}\}, \qquad t_i \sim U[-v, v] \tag{3}$$

We also consider deterministically sampling t_i from linearly spaced grid $[-v, v; N]$ of N values between $-v$ and v:

$$\{e^{A(t_1)}, e^{A(t_2)}, \cdots e^{A(t_N)}\}, \qquad t_i \in [-v, v; N] \tag{4}$$

This study focuses on a particular reparameterization of the Lie Group of 2d rotations SO(2), defined by the scaled infinitesimal generator and corresponding Lie exponential:

$$A(t) = \begin{matrix} 0 & -t\pi \\ t\pi & 0 \end{matrix}, \quad e^{A(t)} = \begin{matrix} \cos(t\pi) & -\sin(t\pi) \\ \sin(t\pi) & \cos(t\pi) \end{matrix} \tag{5}$$

By learning v, we can effectively interpolate from no-invariance for $v = 0$ to full rotational invariance at $v = 1$.

4. Experiments and Results

We implemented our method in PyTorch (Paszke et al., 2017), and show results on toy problem with different degrees of rotational invariance in Figure 1 with 1024 RFF features and $\sigma = 5.0$, and T applied on the weights.

In the rest of the paper, we will perform experiments on MNIST by using bilinear grid resampling as described in Jaderberg et al. (2015) in combination with small 0.1 sigma Gaussian blur to limit high frequencies to apply transformations T on the weights. For optimization, we used Adam (Kingma & Ba, 2014) with a learning rate of 0.001 ($\beta_1 = 0.9, \beta_2 = 0.999$) cosine annealed (Loshchilov & Hutter, 2016) to zero. We initialize parameters $\boldsymbol{\mu}_c = \mathbf{0}$, $\boldsymbol{L}_c = \boldsymbol{I}$ for all c, $\sigma = 0.3$, and $\alpha = 1.0$. Furthermore, we use 32 samples from $p(T)$, $L = 1$ and a batch size of 128.

4.1. On the Necessity of a Bayesian Approach

Lastly, we try training the parameters directly with standard SGD and cross-entropy, opposed to our VI approach, by replacing the variational distribution q_θ with a point-estimate and omitting the D_{KL}-term in the loss function. As can be seen in Figure 3, we found that the resulting model was completely incapable of learning the correct invariance. This result substantiates the use of marginal likelihood (or a lower bound thereof) for hyper-parameter selection for neural networks, and invariance learning in particular. More broadly speaking, it proves a convincing case for probabilistic machine learning models, such as Bayesian neural networks, beyond their oft-cited use for uncertainty estimation.

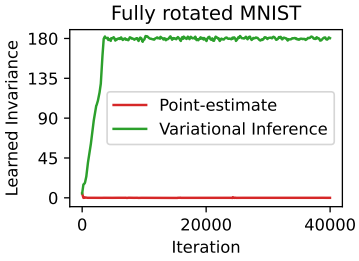


Figure 3. Predicted invariance over training iterations with non-Bayesian point-estimate and approximate Bayesian inference. The model is incapable of learning the correct invariance without VI.

4.2. Identifying Invariance with ELBO

In our first experiment, we evaluate whether the ELBO is capable of identifying the apt level of invariance, as specified by parameter v. We train our model on three versions of MNIST on which we artifically imposed different amounts of invariance by rotating images: In 'Fully rotated MNIST', we rotate every image with a random uniformly sampled angle $\theta \in [-180, 180]$, in 'Partially rotated MNIST' images are rotated with a random angle of $\theta \in [-90, 90]$, and in 'Regular MNIST' we consider the dataset without alterations. In addition, we evaluate a model where the invariance parameter v is learned, rather than kept fixed.

From Table 1, we observe that models with the best ELBO and test accuracy correspond with the invariance that we imposed on the dataset, indicating that the ELBO can correctly identify the required level of invariance with best

(a) Feature bank #1 over training iterations.

(b) Feature bank #2 over training iterations.

Figure 4. Illustration of converging filter banks of two features. Features are initialized randomly with almost no invariance and converge to particular filters with practically full (±179°) rotational invariance when trained on fully rotated MNIST training data.

test generalization. On regular MNIST we observe that a small amount of invariance yields better ELBO than no invariance, which could be explained by some intrinsic rotational variation in the dataset. Furthermore, we find that the ELBO obtained after learning invariance corresponds with the optimal ELBO in the set of models with fixed invariance. Additional results can be found in Appendix C.

	ELBO			Test Accuracy		
Model	Fully rotated MNIST	Partially rotated MNIST	Regular MNIST	Fully rotated MNIST	Partially rotated MNIST	Regular MNIST
Fixed 5°	-1.07	-0.80	-0.36	79.29	86.71	96.00
Fixed 45°	-0.63	-0.49	-0.26	87.35	91.13	95.93
Fixed 90°	-0.52	-0.44	-0.30	90.33	91.69	94.69
Fixed 135°	-0.45	-0.45	-0.36	91.19	91.04	92.13
Fixed 175°	-0.43	-0.47	-0.45	91.57	90.47	90.97
Learned	-0.43	-0.42	-0.26	91.72	92.34	96.40

Table 1. Table containing ELBO and Test Accuracy scores after training for experiments with RFF neural network.

4.3. Recovering Invariance from Initial Conditions

In Figure 2, we repeat the last experiment where we automatically learn invariances for different initial invariances $\phi_1 \in [5°, 45°, 90°, 135°, 175°]$. For most initial conditions we find that the model was able to succesfully recover the amount of invariance imposed on the dataset. One exception being initial 175° degrees on partially rotated dataset, which suggests that training with low initial invariance could be advantageous in practice. Nevertheless, we conlude that we can recover invariance relatively robustly independent of initial conditions.

4.4. Learning Invariance in ReLU Network

So far, we have only considered the set-up where we learn the second layer and keep the first layer initialized as fixed RFF-features with a particular $\cos()$ activation function. We chose this fixed basis function model to ensure a sufficiently tight bound on marginal likelihood where the only source of looseness is the non-Gaussian likelihood. Now, we will let loose of these constraints and consider a general two-layer neural network with ReLU non-linearity $\sigma(x) = \max(x; 0)$ with Xavier (Kumar, 2017) initialized weights and 1024 hidden layers, where we learn both two layers. We optimize the model using the same variational inference procedure.

We find that we are also able to learn invariance in this setting (full comparison in Appendix C). In Figure 4, we show an illustration of a feature bank (row vector W_{h_1} for range$[-v; v; 7]$) over training iterations. As can be seen, the features in the experiment are randomly initialized with almost no rotational invariance. After training on a fully rotated MNIST dataset, the features converge to a particular filter with practically full ±179° rotational invariance.

5. Discussion and Conclusion

In this paper, we propose a method to learn invariant weights in neural networks from data itself. We follow what is common in Bayesian statistics and optimize the marginal likelihood to perform Bayesian model selection: a method that has been proven capable to learn invariances in GPs. We propose a lower bound to allow optimization of the marginal likelihood in neural networks and demonstrate the approach by automatically learning rotationally invariant weights in a two-layer neural network on rotated versions of MNIST.

The marginal likelihood is a general way of doing model selection and is parameterization independent. Therefore, we can expect it to work on other invariances and other model architectures. For deeper models, however, we should ask the question whether the bound on the marginal likelihood will stay sufficiently tight (Dutordoir et al., 2021; Ober & Aitchison, 2020; Immer et al., 2021).

To conclude, we hope our findings inspire other works to allow neural networks that automatically learn invariances from data.

References

Bekkers, E. J. B-spline cnns on lie groups. *arXiv preprint arXiv:1909.12057*, 2019.

Benton, G., Finzi, M., Izmailov, P., and Wilson, A. G. Learning invariances in neural networks. *arXiv preprint arXiv:2010.11882*, 2020.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Dao, T., Gu, A., Ratner, A., Smith, V., De Sa, C., and Ré, C. A kernel theory of modern data augmentation. In *International Conference on Machine Learning*, pp. 1528–1537. PMLR, 2019.

Dutordoir, V., Hensman, J., van der Wilk, M., Ek, C. H., Ghahramani, Z., and Durrande, N. Deep neural networks as point estimates for deep gaussian processes. *arXiv preprint arXiv:2105.04504*, 2021.

Esteves, C., Allen-Blanchette, C., Zhou, X., and Daniilidis, K. Polar transformer networks. *arXiv preprint arXiv:1709.01889*, 2017.

Falorsi, L., de Haan, P., Davidson, T. R., and Forré, P. Reparameterizing distributions on lie groups. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3244–3253. PMLR, 2019.

Finzi, M., Welling, M., and Wilson, A. G. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *arXiv preprint arXiv:2104.09459*, 2021.

Fong, E. and Holmes, C. On the marginal likelihood and cross-validation. *Biometrika*, 107(2):489–496, 2020.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

Immer, A., Bauer, M., Fortuin, V., Rätsch, G., and Khan, M. E. Scalable marginal likelihood estimation for model selection in deep learning. *arXiv preprint arXiv:2104.04975*, 2021.

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. Spatial transformer networks. *arXiv preprint arXiv:1506.02025*, 2015.

Kingma, D. P. Variational inference & deep learning: A new synthesis. 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kumar, S. K. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*, 2017.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Marcos, D., Volpi, M., Komodakis, N., and Tuia, D. Rotation equivariant vector field networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5048–5057, 2017.

Ober, S. W. and Aitchison, L. Global inducing point variational posteriors for bayesian neural networks and deep gaussian processes. *arXiv preprint arXiv:2005.08140*, 2020.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Rahimi, A., Recht, B., et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, pp. 5. Citeseer, 2007.

Schwöbel, P., Warburg, F., Jørgensen, M., Madsen, K. H., and Hauberg, S. Probabilistic spatial transformers for bayesian data augmentation. *arXiv preprint arXiv:2004.03637*, 2020.

van der Wilk, M., Bauer, M., John, S., and Hensman, J. Learning invariances using the marginal likelihood. *arXiv preprint arXiv:1808.05563*, 2018.

Weiler, M. and Cesa, G. General e(2)-equivariant steerable cnns. *arXiv preprint arXiv:1911.08251*, 2019.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv preprint arXiv:1807.02547*, 2018.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5028–5037, 2017.

Appendix A: Detailed Derivation of Variational Inverence

Applying Variational Inference (VI) (Hoffman et al., 2013), we maximize the marginal likelihood w.r.t. parameters θ by minimizing the $D_{KL}(\cdot\|\cdot)$-divergence between approximate posterior $q(W_2|\phi)$ and true posterior distribution of weights $p(W_2|D)$, by minimizing the negative evidence lower bound (ELBO) denoted $\mathcal{L}$:

$$
\begin{aligned}
&\arg\min D_{KL}(q(W_2|\phi)\|p(W_2|D)) \\
&= \arg\min E_{q(W_2|\phi)}\left[\log \frac{p(W_2|\phi)}{p(W_2|D)}\right] \\
&= \arg\min E_{q(W_2|\phi)}\left[\log \frac{p(W_2|\phi)}{p(W_2)p(D|w)}\right] + \log p(D) \\
&= \arg\min E_{q(W_2|\phi)}\left[\log \frac{p(W_2|\phi)}{p(W_2)p(D|w)}\right] \\
&= \arg\min E_{q(W_2|\phi)}[\log p(W_2|\phi) - \log p(W_2) - \log p(D|W_2)] \\
&= \arg\min E_{q(W_2|\phi)}[\log p(W_2|\phi) - \log p(W_2)] - E_{q(W_2|\phi)}[\log p(D|W_2)] \\
&= \arg\min D_{KL}(p(W_2|\phi)\|p(W_2)) + E_{q(W_2|\phi)}[-\log p(D|W_2)] \\
&= \arg\min \sum_c D_{KL}(\mathcal{N}(w_2^c|\mu^c,\sigma^c)\|p(w_2^c)) + E_{q(W_2|\phi)}[-\log p(D|W_2)] \\
&= \arg\min \mathcal{L}
\end{aligned}
$$

We independently model the weights w_2^c for each class c with a full co-variance multivariate Gaussian distribution $\mathcal{N}(w_2^c|\mu^c,\Sigma^c)$, parameterized by mean vector μ^c and lower-triangular (Cholesky) decomposition of the co-variance $L^c = \text{chol}(\Sigma^c)$ to avoid computational issues, similar to Kingma (2017). We can view the variational posterior $q(W_2|\phi)$ as multi-variate Gaussian over all classes with concatenated mean and block-diagonally stacked covariances from which we sample flattened matrix W_2 in one go, or -equivalently- sample row vectors w_2^c for each class separately and concatenate them to obtain matrix W_2. By sampling L times from variational approximation $W_2^{(1)}, W_2^{(2)}, \ldots, W_2^{(L)} \sim q(W_2|\phi)$ we obtain a Monte Carlo estimate of $E_{q(W_2|\phi)}$ yielding the final negative ELBO loss $L(\theta, D)$:

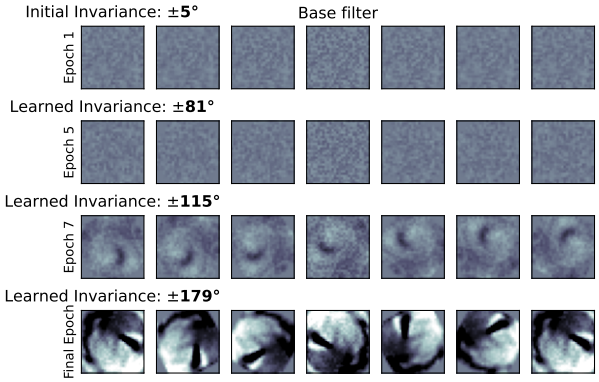
$$
\begin{aligned}
L(\theta, D) &= E_{q(W_2|\phi,\Sigma^c)}[-\log p(D|W_2)] + \sum_c D_{KL}(\mathcal{N}(w_2^c|\mu^c,\Sigma^c)\|p(w_2^c)) \\
&= E_{q(W_2|\phi,\Sigma^c)}[-\log p(D|W_2)] + \sum_c^C D_{KL}(\mathcal{N}(w_2^c|\mu^c,\Sigma^c)\|\mathcal{N}(0,\Sigma_p)) \\
&= \underbrace{\sum_l^L \sum_i^N -\log_{y_c^{(i)}} E_{T\sim p(T)}\left[W_2 \cdot (W_1 \cdot T \cdot x^{(i)})\right]}_{\text{Regular Average Cross-entropy}} + \underbrace{\sum_c \frac{1}{2}\left[\log\frac{|\Sigma_p^c|}{|\Sigma^c|} - D + \text{tr}\{\Sigma_p^{-c}\Sigma^c\} + \mu^T \Sigma_p^{-1}\right]}_{\text{Closed-form KL Regularizer}}
\end{aligned}
$$

for every input $x^{(i)}$, log soft-argmax output y_c for class of corresponding label $y_c^{(i)}$, fixed first layer weights W_1, prior weights $\Sigma_p = I$, input dimensionality D, and trace $\text{tr}(\cdot)$. To allow for mini-batching, we use the Stochastic Variational Bayes Estimate (SGVB) from Kingma & Welling (2013) of the negative ELBO loss $\tilde{\mathcal{L}}(\theta, D)$:

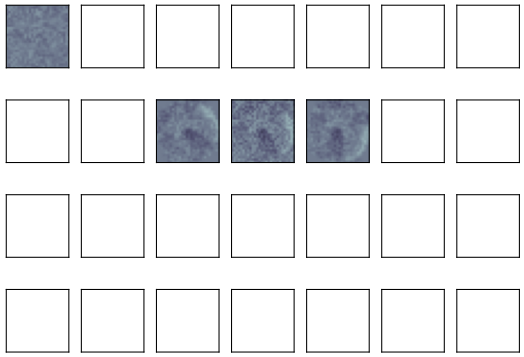
$$
\tilde{\mathcal{L}}(\theta, D) = \underbrace{\frac{N}{M}\sum_l^L \sum_i^M -\log_{y_c^{(i)}} E_{T\sim p(T)}\left[W_2 \cdot (W_1 \cdot T \cdot x^{(i)})\right]}_{\text{Regular Batch Averaged Cross-entropy}} + \underbrace{\sum_c \frac{1}{2}\left[\log\frac{|\Sigma_p^c|}{|\Sigma^c|} - D + \text{tr}\{\Sigma_p^{-c}\Sigma^c\} + \mu^T \Sigma_p^{-1}\right]}_{\text{Closed-form KL Regularizer}}
$$

where we can choose $L = 1$ if we use a sufficiently large batch size.

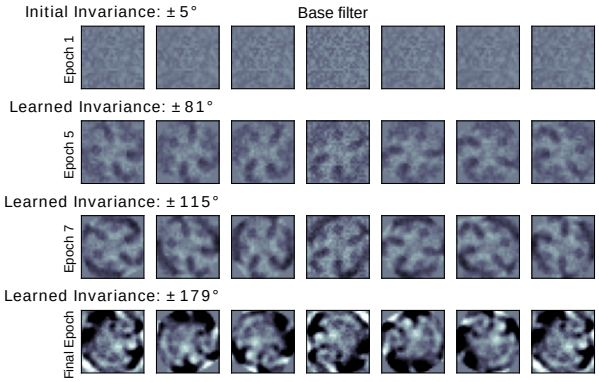
Appendix B: Additional Weight Visualizations



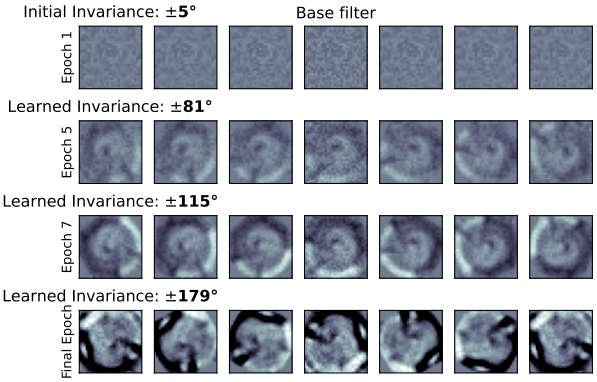
(a) Feature #1 over training iterations



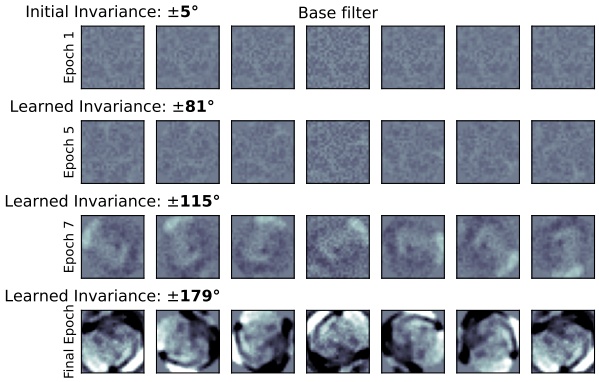
(b) Feature #2 over training iterations



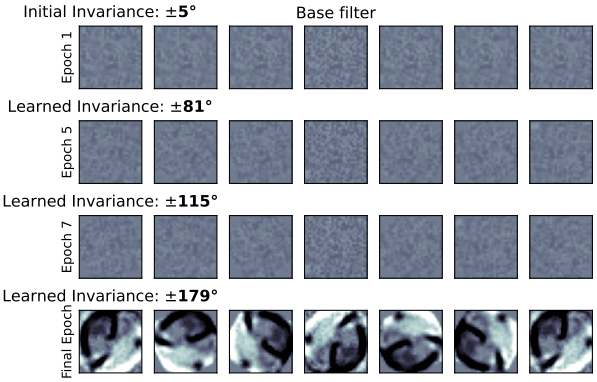
(c) Feature #3 over training iterations



(d) Feature #4 over training iterations



(e) Feature #5 over training iterations



(f) Feature #6 over training iterations

Figure 5. Illustration of the features banks over training iterations. Features are randomly initialized with almost no rotational invariance and converge to particular filters with full rotational invariance when trained on fully rotated MNIST data.

Appendix C.1: Additional Results for RFF Neural Network

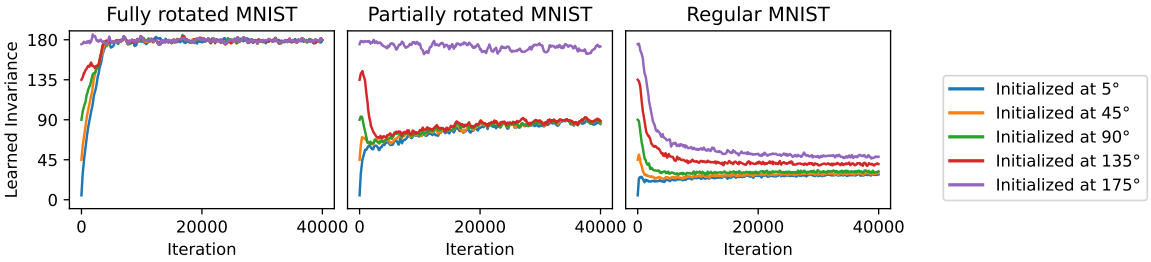


Figure 6. Predicted invariance over training iterations for different initial invariances for RFF neural network.

	ELBO			Test Accuracy		
Model	Fully rotated MNIST	Partially rotated MNIST	Regular MNIST	Fully rotated MNIST	Partially rotated MNIST	Regular MNIST
Fixed 5°	-1.07	-0.80	-0.36	79.29	86.71	**96.00**
Fixed 45°	-0.63	-0.49	**-0.26**	87.35	91.13	95.93
Fixed 90°	-0.52	**-0.44**	-0.30	90.33	**91.69**	94.69
Fixed 135°	-0.45	-0.45	-0.36	91.19	91.04	92.13
Fixed 175°	**-0.43**	-0.47	-0.45	**91.57**	90.47	90.97
Learned (5° Init)	**-0.43**	**-0.42**	**-0.26**	**91.72**	**92.34**	**96.40**
Learned (45° Init)	**-0.43**	**-0.42**	**-0.26**	**91.65**	**92.31**	**96.42**
Learned (90° Init)	**-0.43**	**-0.42**	**-0.26**	**91.65**	**92.37**	**96.40**
Learned (135° Init)	**-0.43**	**-0.42**	**-0.26**	**91.66**	**92.37**	**96.10**
Learned (175° Init)	**-0.43**	**-0.43**	**-0.26**	**91.68**	**91.69**	95.64

Table 2. Table containing ELBO and Test Accuracy scores after training for experiments with RFF neural network.

Appendix C.2: Additional Results for ReLU Neural Network

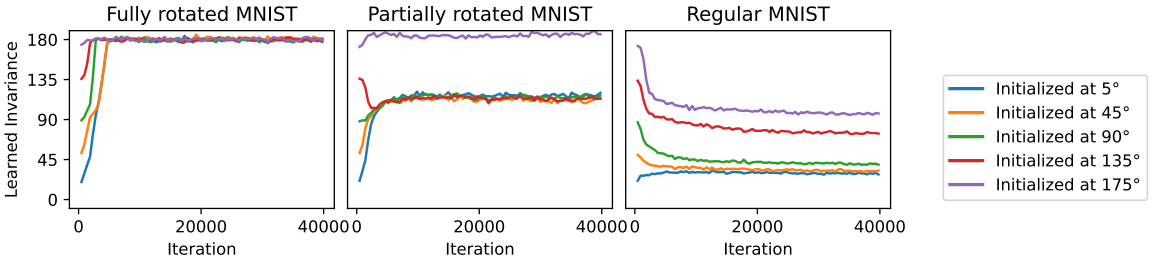


Figure 7. Predicted invariance over training iterations for different initial invariances of ReLU neural network with both layers trained.

	ELBO			Test Accuracy		
Model	Fully rotated MNIST	Partially rotated MNIST	Regular MNIST	Fully rotated MNIST	Partially rotated MNIST	Regular MNIST
Fixed 5°	-0.28	-0.20	**-0.02**	87.21	90.68	96.76
Fixed 45°	-0.09	**-0.06**	-0.02	95.24	96.46	98.13
Fixed 90°	-0.07	-0.06	-0.03	96.50	97.11	**98.14**
Fixed 135°	**-0.06**	-0.06	-0.04	97.15	**97.31**	97.79
Fixed 175°	-0.07	-0.06	-0.06	**97.53**	97.30	97.15
Learned (0° Init)	-0.07	**-0.06**	**-0.02**	97.34	97.13	**98.40**
Learned (45° Init)	-0.07	**-0.05**	**-0.02**	97.23	**97.36**	**98.27**
Learned (90° Init)	-0.07	**-0.06**	**-0.02**	97.28	**97.22**	**98.19**
Learned (135° Init)	**-0.06**	**-0.05**	**-0.02**	97.45	**97.29**	**98.33**
Learned (175° Init)	**-0.06**	**-0.06**	**-0.03**	97.23	**97.23**	**98.03**

Table 3. Table containing ELBO and Test Accuracy scores after training for experiments of ReLU neural network with both layers trained.