Learning Invariant Weights in Neural Networks

Abstract

Assumptions about invariances or symmetries in data can significantly increase the predictive power of statistical models. Many commonly used models in machine learning are constraint to respect certain symmetries in the data, such as translation equivariance in convolutional neural networks, and incorporation of new symmetry types is actively being studied. Yet, *learning* invariances from the data itself 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 resulting in naturally higher performing models.

1 INTRODUCTION

Intuitively, invariances allow models to extrapolate (or 'generalize') beyond training data (see Figure 1 for an extreme example). 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 was made by Dao et al. [2019]. We show that this invariance can equivalently be described as transformations on the weights, similar to Cohen and Welling [2016] where a neural network is constrained to respect rotational symmetry through rotated weight copies. 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], in order to learn the correct invariance from data. We do what is common in Bayesian model selection and find the invariance by optimizing the marginal likelihood. As the true marginal likelihood is not tractable for commonly used neural networks, we propose a lower bound of the marginal likelihood to allow this in practice. By learning distributions on affine groups, we can select the correct invariance for a particular task, without having to perform cross-validation or even requiring a separate validation set. We succesfully learn the correct invariance on different MNIST and CIFAR-10 tasks leading to better performing models.

2 RELATED WORK

Convolutional neural networks (CNNs) have been successful in a wide range of problems and 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 and 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 and Cesa, 2019, Bekkers, 2019] and recent efforts allow practical equivariance in neural networks for arbitrary symmetry groups [Finzi et al., 2021]. Nevertheless, symmetries are typically fixed and must be known and explicitly specified in advance.

Some studies have proposed to learn invariances through data augmentations [Cubuk et al., 2018, Lorraine et al., 2020], but thus do not embed symmetries in the weights and rely on an additional validation loss. Zhou et al. [2020] does learn parameter sharing of layer weights but requires a meta-learning procedure that also relies on a validation loss. Benton et al. [2020] circumvents the need for valida-

Submitted to the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022). **To be used for reviewing only**.

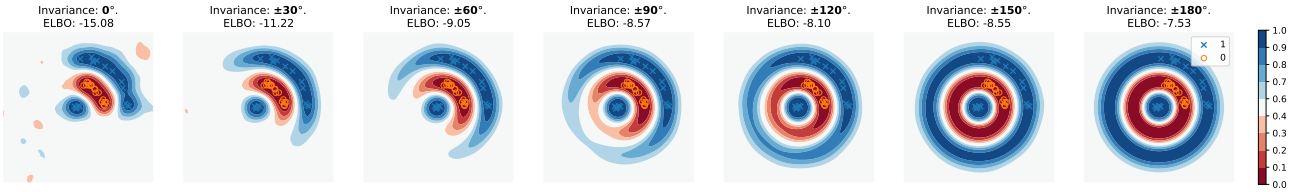


Figure 1: Illustration of extrapolating behaviour further away from toy data for models with no invariance (left), some invariance (middle) up to strict invariance (right). Model prediction plotted as contour and datapoints as $\times$'s and $\bigcirc$'s.

tion data by learning a distribution of input transformations directly on the training loss, but in doing so relies on an additional explicit regularization term that depends on the way invariances are parameterized. Similar to this work, Schwöbel et al. [2020] also proposes to use a lower bound on the marginal likelihood as objective, but again only considers a distribution in the input space rather than on the weights.

We aim to learn invariant weights by optimizing the marginal likelihood: the common method in Bayesian statistics to perform model selection, which is parameterization independent with the aim of being generally applicable to any chosen parameterization of invariance. Interestingly, it has been shown that the marginal likelihood objective coincides with an exhaustive leave-p-out cross-validation averaged over all values of p and held-out test sets [Fong and Holmes, 2020].

Lastly, in Topological VAEs [Keller and Welling, 2021] capsules with 'rolling' feature activations show similarities to the deterministically sampled features obtained from our method, but differ in the reliance on 'temporal coherence'.

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 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 the 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 follow Dao et al. [2019], Raj et al. [2017], van der Wilk et al. [2018], Benton et al. [2020] and construct our invariant function $f_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{\eta})$ from a non-invariant function $g_{\boldsymbol{\theta}}(\boldsymbol{x})$ by summing over the orbit:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{\eta}) = \int g_{\boldsymbol{\theta}}(T(\boldsymbol{x})) p_{\boldsymbol{\eta}}(T) \mathrm{d}T \tag{1}$$

where $p_{\boldsymbol{\eta}}(T)$ denotes a density over the group action transformations parameterized by a vector $\boldsymbol{\eta}$. Through this construction, we hope to induce a relaxed notion of invariance upon the model, sometimes referred to as 'insensitivity' [van der Wilk et al., 2018], 'soft-invariance' [Benton et al., 2020], or 'deformation stability' [Bronstein et al., 2021]. The special case in which the density $p_{\boldsymbol{\eta}}(T)$ is uniformly distributed over the orbit results in the 'Reynolds operator' from Group Theory, which averages functions and thereby induces a 'strict invariance' over the entire group.

3.1 INVARIANT SHALLOW NEURAL NETWORK

We begin by considering a shallow neural network with some imposed constraints and later show that our method also works in the more general case. By introducing some additional constraints (ie. a cosine non-linearity and certain fixed first layer weights), we can guarantee that this network is an approximate (exact in the inifitely wide limit) weight-space equivalent of the Gaussian Process (GP). This is desirable, because GPs have already been proven capable of learning invariance with marginal likelihood in van der Wilk et al. [2018] and we thus expect the lower bound on our marginal likelihood to stay sufficiently tight under these conditions. We construct our invariant function from a non-invariant single-layer neural network:

$$g_{\boldsymbol{\theta}}(T(\boldsymbol{x})) = \sigma\left(\boldsymbol{W}_2 \circ \phi\left(\boldsymbol{W}_1 \circ T \circ \boldsymbol{x}\right)\right) \tag{2}$$

where $\sigma(\cdot)$ is the soft-argmax function, $\boldsymbol{x}$ is the input, and $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are the respective first and second layer weights and biases[1]. We initialize and fix the first layer weights $\boldsymbol{W}_1$ as random Fourier features (RFF) [Rahimi et al., 2007],

[1] In addition to the weights $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, we also consider an additive bias term, which can be implemented by concatenating the input features with an additional 'one'-element but is omitted in notation for clarity.

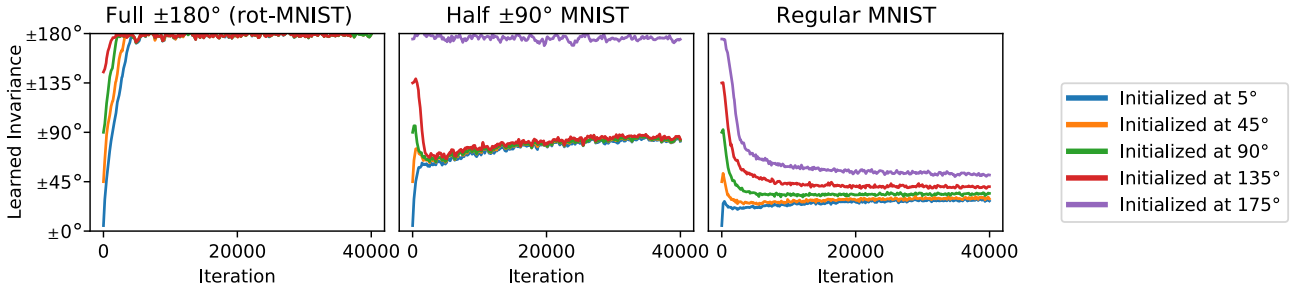


Figure 2: Predicted invariance over training iterations for models initialized with different amounts of invariance when trained on fully rotated MNIST (left), partially rotated MNIST (middle) and regular MNIST (right).

and choose the non-linearity to be the element-wise cosine function $\phi = \cos(\cdot)$. From now on, we will refer to this specific architecture as an RFF neural network. Section 3.7 describes how variational inference is used to learn a variational multivariate Gaussian distribution $q(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with variable mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ over the free parameters in the second layer: $\boldsymbol{\theta} = \text{vec}(\boldsymbol{W}_2)$.

With our particular choice of fixed first layer weights in combination with the cosine activation function, we obtain a weight-space equivalent that is as close as possible to a GP with a radial basis function (RBF) kernel. From van der Wilk et al. [2018], we know that in this case the marginal likelihood is tight and can be used to learn invariance. Later in this study, we loosen these constraints towards models that are more commonly used in DL literature, by considering a network where we choose the non-linearity to be a ReLU activation function $\phi = \max(0, x)$ and learn both the input and the output weights $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$. We refer to this set-up as a ReLU neural network, in which case we find that we are still able to learn invariances from data, indicating that for our purposes the bound on the marginal likelihood also stays sufficiently tight for more general shallow architectures.

3.2 INVARIANCE IN THE WEIGHTS

In Equations 1-2, we showed how we construct an invariant function by integrating or summing over transformed or augmented input samples (ie. '$T(\boldsymbol{x})$'). Yet, instead of explicitly performing these transformations on the input, we can also obtain a mathematically equivalent invariant function by considering transformations on the weights. Note that for the inner term in our neural network definition in Equation 2, we have that $(\boldsymbol{W}_1 \circ T) \circ \boldsymbol{x} = \boldsymbol{W}_1 \circ (T \circ \boldsymbol{x})$ are equal, by associativity of matrix transformations. In other words, first applying transformation T on the weights, similar to the typical construction of equivariant layers, is mathematically equivalent to first applying it to the input, which could be interpret as built-in data augmentation. In practice, however, differences between the two could still arise if applying T requires approximations (e.g. when interpolating rotations on a discretized grid). In our experiments we will consider transforming the weights, thus demonstrating that invariance can be 'built into' the model.

3.3 AFFINE LIE GROUP REPARAMETERIZATION

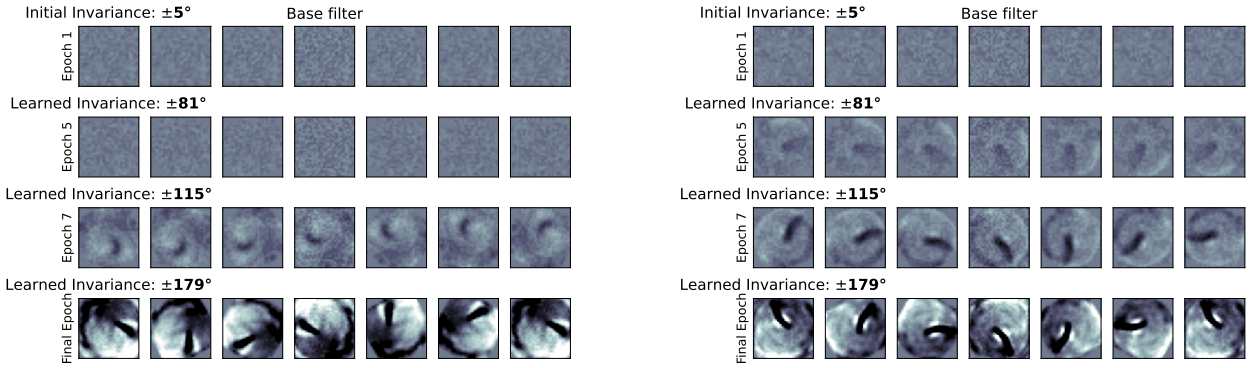
The transformations that are applied on the weights come from a probability distribution $p_{\boldsymbol{\eta}}(T)$ parameterized by $\boldsymbol{\eta}$ over the possible transformations. In order to learn this distribution with back-propagation, we make sure that we can differentiate with respect to $\boldsymbol{\eta}$. For affine transformed weights, we consider a procedure similar to what Benton et al. [2020] used to augment inputs, utilizing the reparameterization trick [Kingma and Welling, 2013] to scale infinitesimal generators with noise from a multi-variate Uniform distribution $\boldsymbol{\epsilon} \sim U[-1, 1]^k$, and similar to the Lie group re-parameterization in Falorsi et al. [2019]. With k=6 generator matrices $\boldsymbol{G}_1, \cdots, \boldsymbol{G}_6$ and learnable parameters $\boldsymbol{\eta} = [\eta_1, \cdots, \eta_6]^T$ we can separately parameterize translation in x, translations in y, rotations, scaling in x, scaling in y, and shearing (see Appendix D). A sample $T \sim p_{\boldsymbol{\eta}}(T)$ can be obtained by transforming noise $\boldsymbol{\epsilon}$ sampled from a k-cubed uniform distribution:

$$T = \exp\left(\sum_i \epsilon_i \eta_i \boldsymbol{G}_i\right), \qquad \boldsymbol{\epsilon} \sim U[-1, 1]^k \tag{3}$$

with matrix exponential $\exp(M) = \sum_{n=0}^{\infty} \frac{1}{n!} M^n$. A distribution over the subgroup of 2d rotations SO(2) can be achieved by only learning the parameter for rotational invariance $\eta_{\text{rot}} = \eta_3$ and fixing $\eta_i = 0$ for all $i \neq 3$. Then,

$$T^{(\text{rot})} = \begin{bmatrix} \cos(\epsilon_3 \eta_{\text{rot}}) & -\sin(\epsilon_3 \eta_{\text{rot}}) & 0 \\ \sin(\epsilon_3 \eta_{\text{rot}}) & \cos(\epsilon_3 \eta_{\text{rot}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

By learning η_{rot}, we can effectively interpolate between no invariance at η_{rot} = 0 to full rotational invariance at $\eta_{\text{rot}} \equiv \pi$.



(a) Feature bank #1 over training iterations.

(b) Feature bank #2 over training iterations.

Figure 3: 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 after training on the fully-rotated MNIST.

Similarly, we can define a distribution over the subgroup of 2d translations $\mathbb{T}(2)$ by fixing $\eta_i = 0$ for all $i > 2$ and learning the translational invariance parameters η_1 and η_2:

$$T^{(\text{trans})} = \begin{bmatrix} 1 & 0 & \epsilon_1 \eta_1 \\ 0 & 1 & \epsilon_2 \eta_2 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

We include full derivations including scaling in Appendix E. In general, $\boldsymbol{\eta} = \mathbf{0}$ corresponds to no invariance and increasing certain scalars in $\boldsymbol{\eta}$ also increases insensitivity to corresponding transformations towards full invariance.

3.4 STOCHASTIC OR DETERMINISTIC SAMPLING

To estimate $f_{\boldsymbol{\theta}}(x; \boldsymbol{\eta})$ from Equation 1, we approximate the integral with a Monte Carlo (MC) estimate:

$$\hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{\eta}) = \frac{1}{S} \sum_{i=1}^{S} g_{\boldsymbol{\theta}}(T_i(\boldsymbol{x})) \tag{6}$$

where S transformations are stochastically sampled from the distribution $T_i \sim p_{\boldsymbol{\eta}}(T)$, in such a way that allows for differentiation w.r.t. $\boldsymbol{\eta}$, possible through the 're-parameterization trick' described in Section 3.3. We know that MC is an unbiased estimator, and thus

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{\eta}) = \mathbb{E}_T \left[\hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{\eta}) \right] \tag{7}$$

with $\mathbb{E}_T := \mathbb{E}_{\prod_{i=1}^{S} p_{\boldsymbol{\eta}}(T_i)}$. Alternatively to stochastic MC sampling, we can obtain a deterministic surrogate of the procedure by replacing the stochastic samples from the noise source $U[-1, 1]^k$ by linearly spaced points along its k-cubed domain. This procedure is similar to quadrature in classical numerical integration, but can also be though of as applying the re-parameterization trick on fixed linspace points instead of uniform noise. A visualization of a discretely sampled filter bank that learns rotational invariance over training iterations is shown in Figure 3. By ensuring sufficient and equally spaced samples, deterministic sampling can be used to ensure reliable and robust inference at test time. Similar to the stochastic sampling, this deterministic procedure is also differentiable and can thus be used during training, but is typically only suited when the number of free parameters in $\boldsymbol{\eta}$ is very small. Nevertheless, deterministic sampling can be theoretically interesting and allow our model to be interpret as a generalization of other architectures. For instance, a single convolutional layer (in which a kernel is discretely and deterministically convolved over the image) followed by spatial pooling, can be interpret as an instance of our invariant MLP with a specific affine invariance transformation in which weights are 'zoomed-in' and deterministically sampled and reapplied over the image plane.

3.5 TRANSFORMATION SAMPLING IN PRACTICE

If $\boldsymbol{\eta}$ comprises several free parameters, discrete sampling suffers from the curse of dimensionality, with the number of required samples growing exponentially with larger K. To illustrate, a sparse 3 quadrature points in each of the $K = 6$ dimensions would already require $3^6 = 729$ samples in case of deterministic sampling. We found that for our model stochastic MC sampling resulted in more robust training behaviour and therefore used this when training the models in the experimental section, except for those in Figure 3 where training procedure of a deterministically sampled learned rotationally invariant filter bank is demonstrated.

(a) Sampled filters of affine model trained on regular mnist.

(b) Sampled filters of affine model trained on rotated mnist.

(c) Sampled filters of affine model trained on scaled mnist.

(d) Sampled filters of affine model trained on translated mnist.

Figure 4: Stochastic samples of learned filter banks of a model capable of learning affine invariances. The same model learns features that are insensitive to different kinds of transformations dependent on the data it was trained on.

3.6 LOWER BOUNDING THE MARGINAL LIKELIHOOD

We have a (typically large) vector $\boldsymbol{\theta}$ containing the model parameters and a (typically small) vector $\boldsymbol{\eta}$ that denotes the invariance parameters. The approach we take in this paper is to perform Bayesian Model Selection and integrate out $\boldsymbol{\theta}$ but find a point-estimate over $\boldsymbol{\eta}$:

$$\hat{\boldsymbol{\eta}} = \arg\max_{\boldsymbol{\eta}} p(\mathcal{D}|\boldsymbol{\eta}) = \arg\max_{\boldsymbol{\eta}} \left[\int p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\eta}) \mathrm{d}\boldsymbol{\theta} \right] \tag{8}$$

where $p(\mathcal{D}|\boldsymbol{\eta})$ is the marginal likelihood [Murphy, 2012] or model evidence: a procedure generally referred to as empirical Bayes or type-II maximum likelihood. The technique has been shown effective in GPs to learn kernels or other hyper-parameters Williams and Rasmussen [2006] and invariances van der Wilk et al. [2018], but is typically intractable for neural networks. We therefore derive a lower bound that allows for optimization of the marginal likelihood in neural networks using stochastic variational inference:

$$\begin{aligned} \log p(\mathcal{D}) &\geq \mathbb{E}_{\boldsymbol{\theta}} \left[\log p(\mathcal{D}|\boldsymbol{\theta}) \right] - \mathrm{KL}(q(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) || p(\boldsymbol{\theta})) \\ &= \mathbb{E}_{\boldsymbol{\theta}} \left[\log p(\boldsymbol{y}|f_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{\eta})) \right] - \mathrm{KL}(q(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) || p(\boldsymbol{\theta})) \\ &= \mathbb{E}_{\boldsymbol{\theta}} \left[\log p \left(\boldsymbol{y} \middle| \mathbb{E}_T \left[\hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{\eta}) \right] \right) \right] - \mathrm{KL}(q(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) || p(\boldsymbol{\theta})) \\ &\geq \mathbb{E}_{\boldsymbol{\theta}} \left[\mathbb{E}_T \left[\log p(\boldsymbol{y}|\hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{\eta})) \right] \right] - \mathrm{KL}(q(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) || p(\boldsymbol{\theta})) \end{aligned} \tag{9}$$

with expectations $\mathbb{E}_{\boldsymbol{\theta}} := \mathbb{E}_{q(\boldsymbol{\theta})}$ and $\mathbb{E}_T := \mathbb{E}_{\prod_{i=1}^{S} p_{\boldsymbol{\eta}}(T_i)}$. We begin Eq 9 with the standard evidence lower bound (ELBO) derived from variational inference. In the second and third line, we expand the likelihood and plug-in Equation 7. In the last line, we use Jensen's inequality together with the fact that our log-likelihood is a convex function. The resulting lower bound comprises an expected log-likelihood term that can be approximated by taking the average cross-entropy on mini-batches of data (see Section 3.7) and a KL-divergence between two multivariate Gaussians which can efficiently be computed in closed-form. Note that we integrate out model parameter vector $\boldsymbol{\theta}$ which is part of the KL-term, while the vector parameterizing the invariances $\boldsymbol{\eta}$ is not. We optimize the derived lower bound w.r.t. both $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ every iteration with stochastic gradient descent.

3.7 VARIATIONAL INFERENCE

To summarize, we propose to learn invariances using stochastic variational inference [Hoffman et al., 2013] and derived a lower bound of the marginal likelihood, or *evidence lower bound* (ELBO) that can be optimized using a stochastic gradient descent method, such as Adam Kingma and Ba [2014]. Variational inference minimizes the KL-divergence between a variational Gaussian distribution $q(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ parameterized by $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and the true posterior on our free model parameters $p(\boldsymbol{\theta}|\mathcal{D})$, where $\boldsymbol{\theta} = \mathrm{vec}(\boldsymbol{W}_2)$. For q we choose a multivariate Gaussian distribution with block-diagonal covariance $\boldsymbol{\Sigma}$ where each block corresponds to an output class c. The covariance is parameterized as a Cholesky decomposition $\boldsymbol{\Sigma} = \boldsymbol{L}^T \boldsymbol{L}$ which is a common trick to maintain computational stability (ensures positive semi-definite $\boldsymbol{\Sigma}$) and does not influence the model. We obtain a differentiable Monte Carlo estimate of $q(\boldsymbol{\theta})$ by sampling L times from the variational distribution, using the reparameterization trick [Kingma and Welling, 2013], and maximize the ELBO:

$$\begin{aligned} \mathcal{L} &= \mathbb{E}_{\boldsymbol{\theta}} \Big[\mathbb{E}_T \Big[\log p(\mathcal{D}|\boldsymbol{\theta}) \Big] \Big] - \mathrm{KL}(q(\boldsymbol{\theta}) || p(\boldsymbol{\theta})) \\ &\approx \frac{1}{L} \sum_{l=1}^{L} \underbrace{\Big[\log p(\boldsymbol{y}| \frac{1}{S} \sum_{i=1}^{S} g_{\boldsymbol{\theta}_l}(T_i(\boldsymbol{x})) \Big]}_{\text{Cross-entropy}} - \underbrace{\mathrm{KL}(q(\boldsymbol{\theta}) || p(\boldsymbol{\theta}))}_{\text{Closed-form KL}} \end{aligned}$$

where we can choose $L = 1$ given a sufficiently large batch size. We obtain a Stochastic Gradient Variational Bayes (SGVB) estimate of the lower bound $\frac{N}{M} \sum_{i=1}^{M} \tilde{\mathcal{L}}(\boldsymbol{\theta}, \{\boldsymbol{x}_i\}, \{y_i\})$ [Kingma and Welling, 2013] to

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

Table 1: ELBO and Test Accuracy scores on MNIST using RFF neural network[2]. For each dataset, we observe that the correct level of invariance for that dataset corresponds with highest ELBO and also correlates with best test accuracy. In addition, we find that automatically learned invariance converges to ELBO and test accuracies similar or beyond the found optimal values from the models with fixed invariance.

allow efficient training on mini batches of data. Full derivations can be found in Appendix A.

4 EXPERIMENTS AND RESULTS

We implemented our method in PyTorch [Paszke et al., 2017], and show results on a toy problem with different degrees of rotational invariance in Figure 1 with 1024 RFF features, $\sigma = 5$, and T applied on the weights.

The rest of the paper contains experiments on MNIST and CIFAR-10 where T is applied on the weights by using the bilinear grid resampling as described in Jaderberg et al. [2015] in combination with small 0.1 sigma Gaussian blur to bandlimit high frequencies. Unless otherwise stated, we used Adam [Kingma and Ba, 2014] for optimization with a learning rate of 0.001 ($\beta_1 = 0.9, \beta_2 = 0.999$) cosine annealed [Loshchilov and Hutter, 2016] to zero. Parameters were initialized as $\boldsymbol{\mu}_c = \mathbf{0}$, $\boldsymbol{L}_c = \boldsymbol{I}$ for all c, $\sigma = 0.3$, and $\alpha = 1.0$. We use $S = 32$ samples from $p_{\boldsymbol{\eta}}(T)$, $L = 1$ and a batch size of 128.

4.1 ON THE NECESSITY OF A BAYESIAN APPROACH

Instead of variational inference, we also tried regular maximum likelihood training of the parameters with Adam and plain cross-entropy, by replacing the variational distribution $q(\boldsymbol{\theta})$ with a point-estimate and omitting the KL-term in the loss function. Interestingly, we found that in this case the model was completely incapable of learning the correct invariance, as can be seen in Figure 5 for rotation. We hypothesize that maximum likelihood alone is not enough to learn invariance, as invariance is a constraint on the weights and thus does not help to fit the data better, whilst marginal likelihood also favours simpler models. 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.

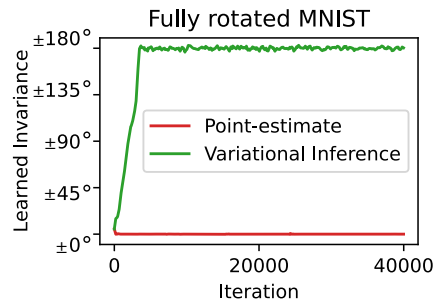


Figure 5: Predicted invariance over training iterations with non-Bayesian point-estimate and approximate Bayesian inference. The model is incapable of learning the correct invariance without VI.

4.2 IDENTIFYING INVARIANCE WITH ELBO

To evaluate whether the ELBO is capable of identifying the apt level of invariance, we consider models with different fixed values of rotational invariance by setting η_{rot}. We then evaluate them on three different versions of MNIST on which we artifically imposed different amounts of rotational invariance by randomly transforming the dataset beforehand: in 'Fully rotated MNIST', we rotate every image with a random uniformly sampled angle in range $[-180°, 180°]$, in 'Partially rotated MNIST' images are rotated with a random angle within $[-90°, 90°]$, and in 'Regular MNIST' we consider the dataset without alterations. In addition, we also consider a model in which η_{rot} is learned, rather than kept fixed.

From Table 1, we observe that for each dataset the model

[2] As explained in Section 3.1, we use the RFF neural network to ensure a tight lower bound and for comparison purposes. Higher accuracies on MNIST and CIFAR-10 were achieved with a ReLU neural network as reported in Table 2 and Table 3.

with the best ELBO corresponds to the model with the right amount of invariance and also correlates with best test accuracy. This finding indicates that the ELBO can correctly identify the required level of invariance, and confirms that choosing the right invariance leads to better generalization on the test set. 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 within 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. Therefore we find that in this case we can use the ELBO to learn invariances in a differentiable manner. Additional results can be found in Appendix C.

4.3 RECOVERING INVARIANCE FROM INITIAL CONDITIONS

To investigate robustness to different initial conditions, we repeat the experiment where we learn invariance parameters $\boldsymbol{\eta}$ during during training on fully-rotated, partially rotated and regular MNIST data but with different initial values, corresponding to rotational invariance of [$\pm 5°$, $\pm 45°$, $\pm 90°$, $\pm 135°$, $\pm 175°$] degrees. Results of this experiments for the RFF neural network are shown in Figure 2, and a similar figure for the ReLU neural network is attached in Appendix C.2. For most initial conditions, we observe that we can succesfully learn and recover the 'correct' amount of invariance for each dataset. One exception being initial 175° degrees on partially rotated dataset, which suggests that training with low initial invariance could be advantageous in practice, for this method. Nevertheless, we conclude that our model 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 output layer - from the hidden units to the output - and keep the first layer - from input to the hidden units - initialized as fixed RFF-features with a particular $\cos(\cdot)$ 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 single hidden layer neural network with ReLU non-linearity $\phi(x) = \max(x, 0)$ with Xavier [Kumar, 2017] initialized weights and 1024 hidden units, where we learn both the input layer and the output layer. We optimize the model using the same variational inference procedure.

We find that we are still able to learn invariances in the setting where parameters of both input and output layer are learned (full comparison in Appendix C). In Figure 3, we plot an illustration of a feature bank (row vector in $\boldsymbol{W}_1$ with 7 samples equally spaced between $-\eta_{\text{rot}}$ and η_{rot} and plotted over training iterations). The top of the figure shows the randomly initialized features without any rotational invariance at the beginning of training. After training on a fully rotated MNIST, the features converge to a particular filter with practically full ±179° rotational invariance, as shown on the bottom of the same figure.

4.5 OTHER TRANSFORMATIONS

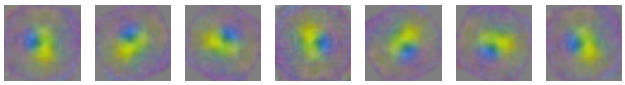
To explore invariance to transformations other than rotation, we repeat the experiment where both layers in a ReLU neural network are learned but allow for different kinds of affine invariance transformations, namely rotation, translation, scale and full affine transformations (see Section 3.3).

In Table 2, we evaluate and compare models that can learn affine invariances with two non-invariant baselines, namely a regular Gaussian Process regression with RBF kernel baseline (SGPR) and a regular shallow neural network baseline (MLP). We use SGPR as a reference, because we know the training procedure is reliable and to ensure enough capacity is given to the single layer MLP. We separately trained the models on fully-rotated, translated, scaled and original versions of MNIST (see Appendix D for details). We find that models with learned invariances (bottom four rows) outperform the model with no invariance (top two rows) in all cases. As expected, a translationally invariant model performs better on a dataset that contains randomly translated examples, and similarly, the rotationally and scale invariant models perform best on respective rotated and scaled versions of MNIST. The model capable of learning affine invariances performs best overall. Moreover, by inspecting the learned coefficients of $\boldsymbol{\eta}$ after training we verified that the learned transformations correspond to the dataset the it was trained on. This can also be observed by inspecting the resulting learned filter banks samples after training on each dataset, shown in Figure 4.

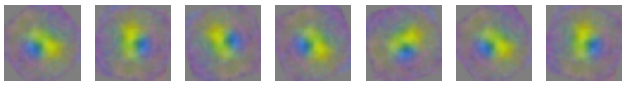
	Test Accuracy			
Model	Fully rotated MNIST	Translated MNIST	Scaled MNIST	Regular MNIST
SGPR	91.19	89.22	72.10	97.52
MLP	90.35	89.34	96.61	98.10
MLP + Rotation (**ours**)	**98.05**	94.08	97.62	98.64
MLP + Translation (**ours**)	93.59	**97.87**	97.98	98.76
MLP + Scale (**ours**)	93.80	94.30	**98.06**	98.35
MLP + Affine (**ours**)	**98.14**	**97.66**	**98.31**	**98.93**

Table 2: Test Accuracy scores for learned invariance using different transformations in a shallow ReLU neural network on the MNIST dataset.

We repeated the same experiment on the CIFAR-10 dataset Krizhevsky et al. [2009] and trained on fully-rotated, translated, scaled version and the original version of the CIFAR-10 dataset and plot test accuracies in Table 3. We



(a) Deterministic samples from learned rotationally invariant filter bank.



(b) Stochastic samples from learned rotationally invariant filter bank.

(c) Deterministic samples from learned rotationally invariant filter bank.

(d) Stochastic samples from learned rotationally invariant filter bank.

Figure 6: Visualization of samples from learned filter banks using discrete sampling learned on different versions of CIFAR-10. The invariant transformations are learned starting from no invariance dependent on the data it was trained on.

consistently find that the best performing model is the one that is parameterized such that it can learn the invariance that corresponds the dataset, typically resulting in several percentage points of improved accuracy compared to the MLP baseline. Furthermore, if we parameterize the MLP with more general affine invariance, capable of expressing rotation, translation and scale invariances, we always achieve similar or improved results compared to the best invariance parameterization. Similar to the MNIST experiments, we find that the MLP with general affine invariances can select the correct invariance based on the used training data. Here we also verified this by inspecting the $\boldsymbol{\theta}$, and found that the learned invariance always matches the invariances that we expect for the corresponding dataset. For example, the model capable of learning affine invariances correctly learned to be rotationally invariant ($\eta_3 \approx \pi$ and $\eta_i \approx 0$ for $i \neq 3$) after training on the fully-rotated CIFAR-10 dataset.

	Test Accuracy			
Model	Fully rotated CIFAR-10	Translated CIFAR-10	Scaled CIFAR-10	Regular CIFAR-10
MLP	41.24	40.75	46.56	54.49
MLP + Rotation (**ours**)	**46.04**	40.71	46.77	54.72
MLP + Translation (**ours**)	40.99	**45.20**	47.44	**55.79**
MLP + Scale (**ours**)	40.92	41.22	**49.28**	54.72
MLP + Affine (**ours**)	**46.12**	**45.77**	**48.81**	**55.44**

Table 3: Test Accuracy scores for learned invariance using different transformations in a shallow ReLU neural network on the CIFAR-10 dataset.

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 [van der Wilk et al., 2018]. We propose a lower bound to allow optimization of the *marginal likelihood* in shallow neural networks and demonstrate on MNIST and CIFAR-10 that, from training data, we can automatically learn weights that are invariant to affine rotations, such as scale, rotation and translation, dependent on the problem at hand. We show that this leads to better generalization and higher predictive test accuracies.

The marginal likelihood is a general model selection method and is parameterization independent. Therefore, we can expect it to work on other invariances and other model architectures. We focussed on affine transformations, but it would be interesting to consider more complex parameterizable transformations over the image space, such as diffeomorphic vector fields [Schwöbel et al., 2020]. We demonstrated that we can learn invariance by sampling a learned compactly supported continuous probability distribution over group actions in common Lie groups. Allowing discrete groups would either require differentiating through a discrete probability distribution, for instance utilizing the Gumbel-Softmax trick [Jang et al., 2016]), or by treating the discrete group as a subgroup of some Lie group and learn to approximately distribute all continuous density $p_{\boldsymbol{\eta}}(T)$ to the group actions of the subgroup. Even in simple cases, this would require more rich and complex probability densities over group actions, such as a mixture distributions or normalizing flows [Rezende and Mohamed, 2015, Tabak and Turner, 2013], capable of expressing multiple modes as in [Falorsi et al., 2019]. Albeit interesting for future work, we did not yet explore practical feasibility of such extensions. We found that we could succesfully learn invariance using marginal likelihood, also referred to as Empirical Bayes or Type-II ML, which could not be learned with regular maximum likelihood (Type-I ML). We did, however, rely on small $\boldsymbol{\eta}$ and learning higher dimensional invariances might require more sophisticated methods which may need additional priors on $\boldsymbol{\eta}$. Lastly, this work focuses on single layer neural networks, and we will consider deeper architectures in future work. For deeper models, we should ask the question whether the bound on the marginal likelihood will stay sufficiently tight [Dutordoir et al., 2021, Ober and Aitchison, 2020, Immer et al., 2021].

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

References

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

Gregory Benton, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. Learning invariances in neural networks. *arXiv preprint arXiv:2010.11882*, 2020.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Tri Dao, Albert Gu, Alexander Ratner, Virginia Smith, Chris De Sa, and Christopher Ré. A kernel theory of modern data augmentation. In *International Conference on Machine Learning*, pages 1528–1537. PMLR, 2019.

Vincent Dutordoir, James Hensman, Mark van der Wilk, Carl Henrik Ek, Zoubin Ghahramani, and Nicolas Durrande. Deep neural networks as point estimates for deep gaussian processes. *arXiv preprint arXiv:2105.04504*, 2021.

Carlos Esteves, Christine Allen-Blanchette, Xiaowei Zhou, and Kostas Daniilidis. Polar transformer networks. *arXiv preprint arXiv:1709.01889*, 2017.

Luca Falorsi, Pim de Haan, Tim R Davidson, and Patrick Forré. Reparameterizing distributions on lie groups. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3244–3253. PMLR, 2019.

Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *arXiv preprint arXiv:2104.09459*, 2021.

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

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Mohammad Emtiyaz Khan. Scalable marginal likelihood estimation for model selection in deep learning. *arXiv preprint arXiv:2104.04975*, 2021.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *arXiv preprint arXiv:1506.02025*, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

T Anderson Keller and Max Welling. Topographic vaes learn equivariant capsules. *arXiv preprint arXiv:2109.01394*, 2021.

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

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

Diederik Pieter Kingma. Variational inference & deep learning: A new synthesis. 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

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

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.

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

Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5048–5057, 2017.

Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review*, 45(1):3–49, 2003.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

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

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer, 2007.

Anant Raj, Abhishek Kumar, Youssef Mroueh, Tom Fletcher, and Bernhard Schölkopf. Local group invariant representations via orbit embeddings. In *Artificial Intelligence and Statistics*, pages 1225–1235. PMLR, 2017.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

Pola Schwöbel, Frederik Warburg, Martin Jørgensen, Kristoffer H Madsen, and Søren Hauberg. Probabilistic spatial transformers for bayesian data augmentation. *arXiv preprint arXiv:2004.03637*, 2020.

Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

Mark van der Wilk, Matthias Bauer, ST John, and James Hensman. Learning invariances using the marginal likelihood. *arXiv preprint arXiv:1808.05563*, 2018.

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

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv preprint arXiv:1807.02547*, 2018.

Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017.

Allan Zhou, Tom Knowles, and Chelsea Finn. Meta-learning symmetries by reparameterization. *arXiv preprint arXiv:2007.02933*, 2020.

APPENDIX A: DETAILED DERIVATION OF VARIATIONAL INVERENCE

Applying Variational Inference (VI) [Hoffman et al., 2013], we maximize the marginal likelihood w.r.t. parameters $\boldsymbol{\theta} = \text{vec}(\boldsymbol{W}_2)$ by minimizing the $D_{\text{KL}}(\cdot||\cdot)$-divergence between approximate posterior $q(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and true posterior distribution of weights $p(\boldsymbol{W}_2|\mathcal{D})$, equivalent to maximizing the evidence lower bound (ELBO) denoted by $\mathcal{L}$:

$$
\begin{aligned}
&\arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} D_{\text{KL}}(q(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma})||p(\boldsymbol{W}_2|\mathcal{D})) \\
&= \arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathbb{E}_{q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma})} \left[\log \frac{q(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma})}{p(\boldsymbol{W}_2|\mathcal{D})}\right] \\
&= \arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathbb{E}_{q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma})} \left[\log \frac{q(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma})}{p(\boldsymbol{W}_2)p(\mathcal{D}|\boldsymbol{W}_2)}\right] + \log p(\mathcal{D}) \\
&= \arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathbb{E}_{q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma})} \left[\log \frac{q(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma})}{p(\boldsymbol{W}_2)p(\mathcal{D}|\boldsymbol{W}_2)}\right] \\
&= \arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathbb{E}_{q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma})} \left[\log p(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma}) - \log p(\boldsymbol{W}_2) - \log p(\mathcal{D}|\boldsymbol{W}_2)\right] \\
&= \arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathbb{E}_{q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma})} \left[\log p(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma}) - \log p(\boldsymbol{W}_2)\right] - \mathbb{E}_{q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma})} \left[\log p(\mathcal{D}|\boldsymbol{W}_2)\right] \\
&= \arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} D_{\text{KL}}(q(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma})||p(\boldsymbol{W}_2)) + \mathbb{E}_{q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma})}[-\log p(\mathcal{D}|\boldsymbol{W}_2)] \\
&= \arg\max_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathbb{E}_{q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma})}[\log p(\mathcal{D}|\boldsymbol{W}_2)] - D_{\text{KL}}(q(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma})||p(\boldsymbol{W}_2)) \\
&= \arg\max_{\boldsymbol{\mu},\boldsymbol{\Sigma}} \mathcal{L}
\end{aligned}
$$

We independently model the weight $\boldsymbol{w}_2^c$ for each class c with a full co-variance multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{w}_2^c|\boldsymbol{\mu}^c, \boldsymbol{\Sigma}^c)$, parameterized by mean vector $\boldsymbol{\mu}^c$ and lower-triangular (Cholesky) decomposition of the co-variance $(\boldsymbol{L}^c)^T\boldsymbol{L}^c = \boldsymbol{\Sigma}^c$ to avoid computational issues, following Kingma [2017]. We can view the variational posterior $q(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as multi-variate Gaussian over all classes with concatenated mean and block-diagonally stacked covariances from which we sample flattened matrix $\boldsymbol{W}_2$ in one go, or -equivalently- sample row vectors $\boldsymbol{w}_2^c$ for each class and concatenate them to obtain matrix $\boldsymbol{W}_2$. By sampling L times from variational approximation $\boldsymbol{W}_2^{(1)}, \boldsymbol{W}_2^{(2)} \dots \boldsymbol{W}_2^{(L)} \sim q(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ we obtain a Monte Carlo estimate of $\mathbb{E}_{\boldsymbol{W}} := \mathbb{E}_{\boldsymbol{W}_2 \sim q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma})}$ required to compute the final ELBO or negative loss $\mathcal{L}(\boldsymbol{\theta}, \mathcal{D})$:

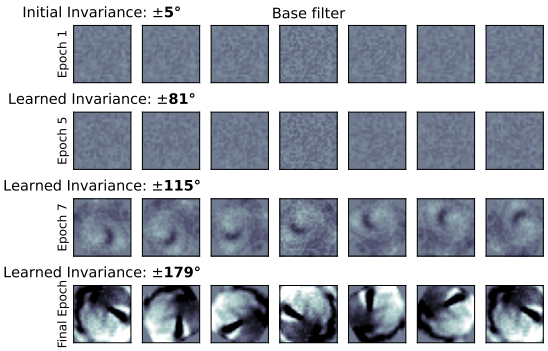
$$
\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) &= \mathbb{E}_{q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma})}[\log p(\mathcal{D}|\boldsymbol{W}_2)] - D_{\text{KL}}(p(\boldsymbol{W}_2|\boldsymbol{\mu}, \boldsymbol{\Sigma})||p(\boldsymbol{W}_2)) \\
&= \mathbb{E}_{q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma})}[\log p(\mathcal{D}|\boldsymbol{W}_2)] - \sum_c D_{\text{KL}}(\mathcal{N}(\boldsymbol{w}_2^c|\boldsymbol{\mu}^c, \boldsymbol{\Sigma}^c)||p(\boldsymbol{w}_2^c)) \\
&= \mathbb{E}_{q(\boldsymbol{W}_2|\boldsymbol{\mu},\boldsymbol{\Sigma}^c)}[\log p(\mathcal{D}|\boldsymbol{W}_2)] - \sum_c D_{\text{KL}}(\mathcal{N}(\boldsymbol{w}_2^c|\boldsymbol{\mu}, \boldsymbol{\Sigma}^c)||\mathcal{N}(\mathbf{0}; \boldsymbol{\Sigma}_p)) \\
&= -\overbrace{\sum_l^L \sum_i^N -\log \sigma_{y_c^{(i)}}\left(\mathbb{E}_{T\sim p_{\boldsymbol{\eta}}(T)}\left[\boldsymbol{W}_2 \circ \phi\left(\boldsymbol{W}_1 \circ T \circ \boldsymbol{x}^{(i)}\right)\right]\right)}^{\text{Regular Average Cross-entropy}} - \overbrace{\sum_c \frac{1}{2}\left[\log \frac{|\boldsymbol{\Sigma}^c|}{|\boldsymbol{\Sigma}_p|} - D + \text{tr}\left\{\boldsymbol{\Sigma}_p\boldsymbol{\Sigma}^c\right\} + \boldsymbol{\mu}^T\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\mu}\right]}^{\text{Closed-form KL Regularizer}}
\end{aligned}
$$

for every input $\boldsymbol{x}^{(i)}$, log soft-argmax output σ_{y_c} for class of corresponding label $y_c^{(i)}$, fixed first layer weights $\boldsymbol{W}_1$, prior weights $\boldsymbol{\Sigma}_p = \boldsymbol{I}\alpha$, input dimensionality D, and trace $\text{tr}(\cdot)$. To allow for mini-batching, we use the Stochastic Variational Bayes Estimate (SGVB) from Kingma and Welling [2013] of the ELBO or negative loss $\tilde{\mathcal{L}}(\boldsymbol{\theta}, \mathcal{D})$:

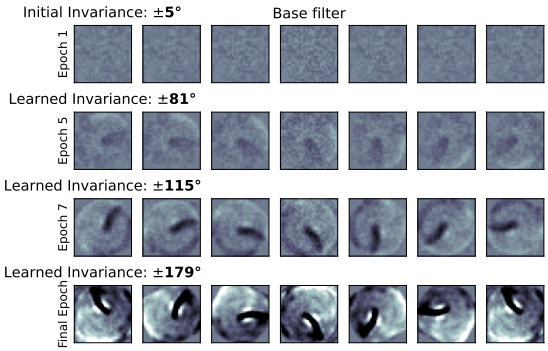
$$
\tilde{\mathcal{L}}(\boldsymbol{\theta}, \mathcal{D}) = -N\overbrace{\frac{1}{M}\sum_l^L \sum_i^M -\log \sigma_{y_c^{(i)}}\left(\mathbb{E}_{T\sim p_{\boldsymbol{\eta}}(T)}\left[\boldsymbol{W}_2 \circ \phi\left(\boldsymbol{W}_1 \circ T \circ \boldsymbol{x}^{(i)}\right)\right]\right)}^{\text{Regular Batch Averaged Cross-entropy}} - \overbrace{\sum_c \frac{1}{2}\left[\log \frac{|\boldsymbol{\Sigma}^c|}{|\boldsymbol{\Sigma}_p|} - D + \text{tr}\left\{\boldsymbol{\Sigma}_p\boldsymbol{\Sigma}^c\right\} + \boldsymbol{\mu}^T\boldsymbol{\Sigma}_p^{-1}\boldsymbol{\mu}\right]}^{\text{Closed-form KL Regularizer}}
$$

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

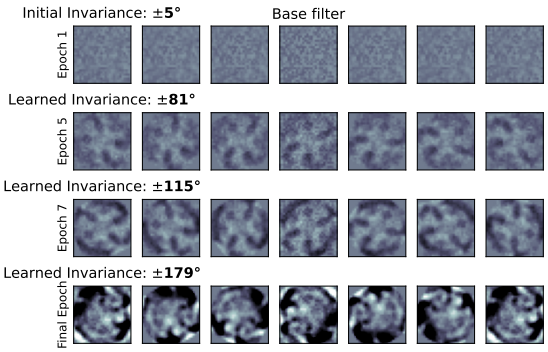
APPENDIX B: WEIGHT VISUALIZATIONS OF LEARNED ROTATIONAL INVARIANCE



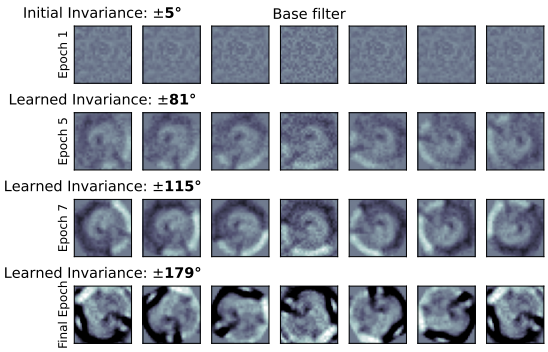
(a) Feature bank #1 over training iterations



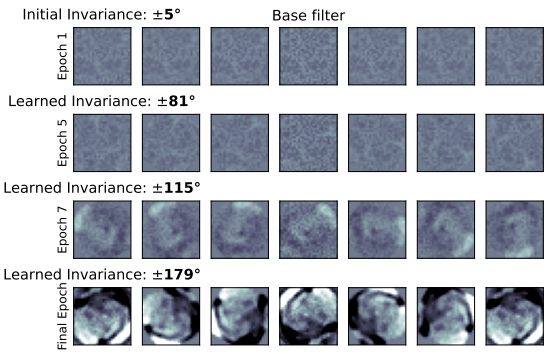
(b) Feature bank #2 over training iterations



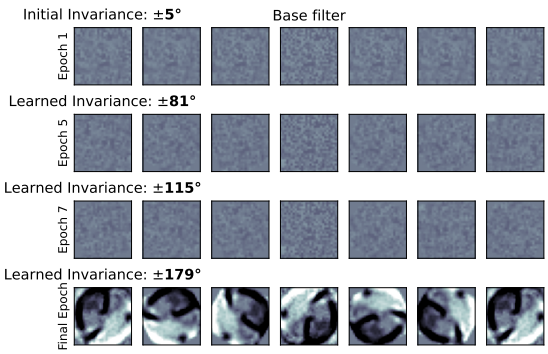
(c) Feature bank #3 over training iterations



(d) Feature bank #4 over training iterations



(e) Feature bank #5 over training iterations



(f) Feature bank #6 over training iterations

Figure 7: 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: ROTATIONAL INVARIANCE IN RFF NEURAL NETWORK

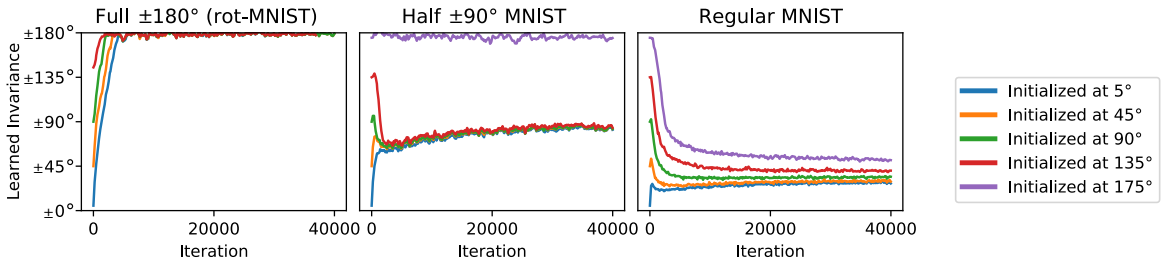


Figure 8: Predicted invariance over training iterations for different initial invariances for RFF neural network.

	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 4: Table containing ELBO and Test Accuracy scores after training for experiments with RFF network.

APPENDIX C.2: ROTATIONAL INVARIANCE IN RELU NEURAL NETWORK

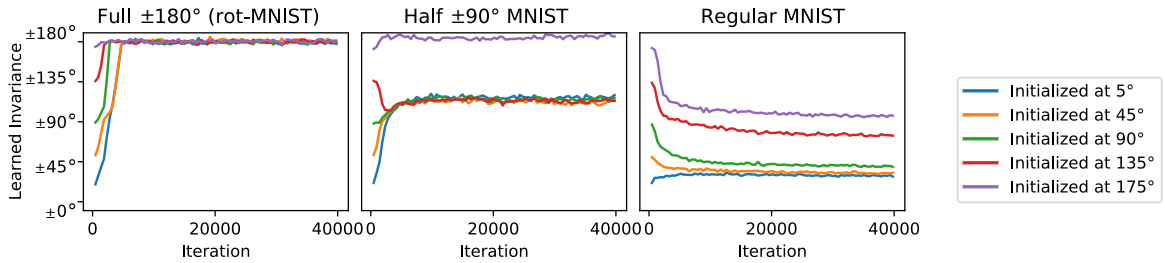


Figure 9: Predicted invariance over training iterations for different initial invariances of ReLU neural network with both input and output layer weights trained.

	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 5: Table containing ELBO and Test Accuracy scores after training for experiments of ReLU neural network with both input and output layer weights trained.

APPENDIX C.3: DIFFERENT TRANSFORMATIONS IN RFF NETWORK

	ELBO				Test Accuracy			
Model	Fully rotated MNIST	Translated MNIST	Scaled MNIST	Regular MNIST	Fully rotated MNIST	Translated MNIST	Scaled MNIST	Regular MNIST
Regular MLP	-1.14	-1.49	-0.69	-0.39	79.29	66.07	89.25	95.16
+ Rotation	**-0.43**	-1.08	-0.62	-0.26	**92.59**	75.06	88.66	96.59
+ Translation	-0.82	**-0.64**	-0.72	-0.24	83.66	**87.81**	86.15	96.78
+ Scale	-0.84	-1.08	**-0.49**	-0.26	82.77	75.48	**91.31**	96.52
+ Affine	**-0.43**	**-0.64**	**-0.54**	**-0.21**	**92.64**	**87.77**	**90.58**	**97.38**

Table 6: ELBO and Test Accuracy scores for learned invariance using different transformations in a shallow RFF neural network.

APPENDIX C.4: DIFFERENT TRANSFORMATION IN RELU NETWORK

	ELBO				Test Accuracy			
Model	Fully rotated MNIST	Translated MNIST	Scaled MNIST	Regular MNIST	Fully rotated MNIST	Translated MNIST	Scaled MNIST	Regular MNIST
Regular MLP	-0.06	-0.06	-0.03	-0.02	90.35	89.34	96.61	98.10
+ Rotation	**-0.05**	-0.06	-0.03	-0.02	**98.05**	94.08	97.62	98.64
+ Translation	-0.09	-0.06	-0.03	-0.02	93.59	**97.87**	97.98	98.76
+ Scale	-0.06	-0.06	-0.03	-0.02	93.80	94.30	**98.06**	98.35
+ Affine	**-0.05**	-0.06	-0.03	-0.02	**98.14**	**97.66**	**98.31**	**98.93**

Table 7: ELBO and Test Accuracy scores for learned invariance using different transformations in a shallow ReLU neural network.

APPENDIX D: DATASET DETAILS

All datasets have 60000 training examples and 10000 test examples and are created by taking regular MNIST and applying random transformations:

Regular MNIST: MNIST handwritten digit database [LeCun et al., 1998].
Partially rotated MNIST: Every sample rotated by radian angle θ, sampled from $\theta \sim U[-\frac{\pi}{2}, \frac{\pi}{2}]$.
Fully rotated MNIST: Every sample rotated by radian angle θ, sampled from $\theta \sim U[-\pi, \pi]$.
Translated MNIST: Translated samples relatively by dx and dy pixels, sampled from $dx, dy \sim U[-8, 8]$.
Scaled MNIST: Every sample scaled around center with $\exp(s)$, sampled from $s \sim U[-\log(2), \log(2)]$.

APPENDIX E: LIE GROUP GENERATORS

We follow Benton et al. [2020] and, similarly, utilize six matrix generators:

$$\boldsymbol{G}_{\text{transx}} = \boldsymbol{G}_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{G}_{\text{transy}} = \boldsymbol{G}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{G}_{\text{rot}} = \boldsymbol{G}_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{G}_{\text{scalex}} = \boldsymbol{G}_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{G}_{\text{scaley}} = \boldsymbol{G}_5 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{G}_{\text{shear}} = \boldsymbol{G}_6 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

To parameterize affine transformations we compute the following matrix exponential [Moler and Van Loan, 2003]:

$$T_{\boldsymbol{\epsilon}} = \exp\left(\sum_i \epsilon_i \eta_i \boldsymbol{G}_i\right), \qquad \boldsymbol{\epsilon} \sim U[-1, 1]^k \tag{10}$$

Optionally, the values of $\boldsymbol{\eta}$ can be constrained to a positive range by passing them through a 'softplus'-function, or in case of $\eta_3 = \eta_{\text{rot}}$ to $[-\pi, \pi]$ using a scaled 'tanh' function, preventing double coverage on the unit circle. In practice, however, we did not find such constraints necessary as long as η_{rot} is reasonably initialized (e.g. $\boldsymbol{\eta} = \mathbf{0}$).

By fixing certain η_i at 0, subsets of the generator matrices parameterize rotation, translation and scaling:

For rotation only:
Learn η_3.
Fix $\eta_i = 0$ for all $i \neq 3$.

$$\begin{aligned} T_{\boldsymbol{\epsilon}}^{(\text{rot})} &= \exp\left(\sum_i \epsilon_i \eta_i \boldsymbol{G}_i\right) \\ &= \exp\left(\epsilon_3 \eta_3 \boldsymbol{G}_3\right) \\ &= \exp\left(\begin{bmatrix} 0 & -\epsilon_3\eta_3 & 0 \\ \epsilon_3\eta_3 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right) \\ &= \begin{bmatrix} \cos(\epsilon_3\eta_3) & -\sin(\epsilon_3\eta_3) & 0 \\ \sin(\epsilon_3\eta_3) & \cos(\epsilon_3\eta_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

For translation only:
Learn η_1 and η_2.
Fix $\eta_i = 0$ for all $i > 2$.

$$\begin{aligned} T_{\boldsymbol{\epsilon}}^{(\text{trans})} &= \exp\left(\sum_i \epsilon_i \eta_i \boldsymbol{G}_i\right) \\ &= \exp\left(\epsilon_1\eta_1\boldsymbol{G}_1 + \epsilon_2\eta_2\boldsymbol{G}_2\right) \\ &= \exp\left(\begin{bmatrix} 0 & 0 & \eta_1 \\ 0 & 0 & \eta_2 \\ 0 & 0 & 0 \end{bmatrix}\right) \\ &= \begin{bmatrix} 1 & 0 & \epsilon_1\eta_1 \\ 0 & 1 & \epsilon_2\eta_2 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

For scaling only:
Learn η_4 and η_5.
Fix $\eta_i = 0$ for all $i \notin \{4, 5\}$.

$$\begin{aligned} T_{\boldsymbol{\epsilon}}^{(\text{scale})} &= \exp\left(\sum_i \epsilon_i \eta_i \boldsymbol{G}_i\right) \\ &= \exp\left(\epsilon_4\eta_4\boldsymbol{G}_4 + \epsilon_5\eta_5\boldsymbol{G}_5\right) \\ &= \exp\left(\begin{bmatrix} \eta_4 & 0 & 0 \\ 0 & \eta_5 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right) \\ &= \begin{bmatrix} \exp(\epsilon_4\eta_4) & 0 & 0 \\ 0 & \exp(\epsilon_5\eta_5) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$