A Bayesian Approach to Invariant Deep Neural Networks

Nikolaos Mourdoukoutas [1] **Marco Federici** [2] **Georges Pantalos** [1] **Mark van der Wilk** [3] **Vincent Fortuin** [1]

Abstract

We propose a novel Bayesian neural network architecture that can learn invariances from data alone by inferring a posterior distribution over different weight-sharing schemes. We show that our model outperforms other non-invariant architectures, when trained on datasets that contain specific invariances. The same holds true when no data augmentation is performed.

1. Introduction

Deep learning models interpolate famously well on data that are generated from the training distribution. Nevertheless, when it comes to generalizing to out-of-distribution examples (e.g., transformed inputs), their predictive potential is more restricted. For example, a classifier might be able to correctly predict the label of a handwritten digit, but could easily fail when the digit is rotated.

A typical approach to solve this problem is to perform data augmentation (DA), where one includes transformed inputs in the training set and consequently boosts the performance of the learned model when presented with similar examples. However, this method is not guaranteed to be successful (Lyle et al., 2020). This is no surprise, as learning to label an image that is rotated by $90°$, for instance, does not imply generalization to different degrees of rotation.

A different class of methods emerges from the fact that for a linear map to be invariant under some transformation, there must be a specific weight-sharing scheme. This is a consequence of Schur's lemma (Kondor & Trivedi, 2018). To that end, if one aims to be robust against a specific type of transformation, it is sufficient to know the corresponding way the parameters of the networks should be shared.

[1]ETH Zürich, Zürich, Switzerland [2]University of Amsterdam, Amsterdam, Netherlands [3]Imperial College London, London, United Kingdom. Correspondence to: Nikolaos Mourdoukoutas <nmourdou@ethz.ch>.

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

1.1. Contributions

We propose a method to learn such weight-sharing schemes from data. As a proof of concept, we focus on being invariant to two types of transformations applied on images, namely rotations and flips. However, our algorithm can be applied to any other choice of symmetry, as long as the corresponding weight-sharing scheme is available. Apart from achieving good performance during inference, our model is able to learn such invariances from data. This is achieved by specifying a probability distribution over the weight-sharing schemes for the input layer, thus formulating a Bayesian neural network. We specify a prior over the parameters of this distribution and then perform MAP inference. We empirically verify and illustrate the capabilities of our model on the MNIST dataset.

2. Background

In order to construct weight-sharing schemes, upon which we will later define a distribution, we will use the concept of the so-called Reynolds' operator, which has been used for similar reasons in previous work (Yarotsky, 2018; van der Pol et al., 2021; Mouli & Ribeiro, 2021). While our notation will be partly adapted from Mouli & Ribeiro (2021), we would like to highlight that our approach is quite different. We do not employ any causal mechanisms nor do we make any assumptions on the data-generating process, as our method does not entail any data augmentation.

Suppose we wish to perform classification of images of shape $3 \times n \times n$. We flatten the images and consider them as vectors in the input space $\mathcal{X} = \mathbb{R}^{3n^2}$. A function $f : \mathcal{X} \to \mathcal{X}$ is invariant to a linear transformation, represented by a matrix A, if the following equality holds for all $x \in \mathcal{X}$: $f(Ax) = f(x)$. In this case, we call A a symmetry of $\mathcal{X}$. If we have another symmetry B, then it easily follows that AB is a symmetry as well. This leads us to consider groups of transformations, with the function composition being the group multiplication.

Definition 1. *Let G be a (finite) group of linear automorphisms from $\mathcal{X}$ to itself. A transformation $\overline{T}$ is called G-invariant, if it holds that $\overline{T}(Tx) = \overline{T}(x)$, for all $x \in \mathcal{X}$ and $T \in G$.*

An element of the group will act on $\mathcal{X}$ via its matrix representation. Explicitly, we can express an arbitrary $T \in G$ with $T(x) = Ax$, for some $A \in \mathbb{R}^{3n^2 \times 3n^2}$ and $x \in \mathcal{X}$.

We wish to specify a weight-sharing scheme, which will enable our neural network to generalize well when it is presented with examples that are generated by the aforementioned action on the input space. Formally, a layer with weights w, bias b, and activation function σ is G-invariant if $\sigma\left(w^\top x + b\right) = \sigma\left(w^\top Tx + b\right)$ for all $T \in G$.

This will be achieved through the Reynolds operator, which we now introduce.

Definition 2. *Let G be a (finite) group of linear automorphisms. The mapping* $\overline{T} : x \mapsto \frac{1}{|G|} \sum_{T \in G} T(x)$ *is the Reynolds operator of the group.*

It can be shown that the Reynolds operator is G-invariant and a projection, that is, it holds that $\overline{T}^2 = \overline{T}$. An immediate consequence of the latter is that all its eigenvalues are either 0 or 1. This fact enables the characterization of G-invariant layers: a layer has this property if and only if its weights are in the span of the eigenvectors of the Reynolds operator with eigenvalue 1. Proofs of these claims can be found in Appendix A.

To summarize, suppose we wish to be invariant to a particular group G. We then proceed by computing the matrix representation of the group's Reynolds operator; computing the eigenvectors $(v_i)_{i=1}^d$ that correspond to the eigenvalue 1; expressing every neuron in the first hidden layer of the network as $w = \sum_{i=1}^d a_i v_i$; and learning the parameters $(a_i)_{i=1}^d$ for every neuron. We now introduce our method.

3. Method

Consider a collection of groups $(G_k)_{k=1}^q$ of linear automorphisms, under which we wish to be invariant. Let $(V_k)_{k=1}^q$ be the matrices that have as columns the eigenvectors which span the eigenspace of the eigenvalue 1 of the corresponding Reynolds operators, where $V_k \in \mathbb{R}^{d_k \times 3n^2}$ for $k = 1, ..., q$. For an illustration of how to compute these matrices, we refer to Appendix A.

We aim to learn whether any of these invariances are present in the data (or could potentially be during testing). To this end, we consider a Categorical distribution over the possible symmetries. The corresponding probabilities are annotated by $(p_k)_{k=1}^{q+1}$, where p_{q+1} is the probability that an observation is not indicative of any symmetry. Of course, it holds that $\sum_{k=1}^{q+1} p_k = 1$. Let $\boldsymbol{\pi} = (p_1, ..., p_{q+1})$. We assume that $\boldsymbol{\pi} \sim \mathrm{Dir}(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_{p+1})$ is the concentration parameter of the Dirichlet distribution and $\alpha_k > 0$. We note that if we are interested in only one group, that is, $q = 1$, then we suppose that $\boldsymbol{\pi} = p_1 \sim \mathrm{Beta}(\alpha_1, \alpha_2)$. We will explain how we choose α in both cases later. By specifying a prior over the parameters of interest, we enable maximum a posteriori (MAP) estimation.

We proceed with the construction of our probabilistic invariant input layer. Fix $k \in \{1, ..., q + 1\}$ and consider the mapping $F : \mathbb{R}^{3n^2} \to \mathbb{R}^m$, which maps the input to the first hidden representation. We construct its weight matrix as follows: Every neuron in the hidden layer has the option to be invariant with respect to the group G_k, with probability p_k. To achieve this, a neuron has to lie in the respective eigenspace. Hence, we first concatenate the matrices $(V_k)_{k=1}^q$, which are the bases of these spaces, into $V = (p_1 V_1, ..., p_q V_q)^\top \in \mathbb{R}^{d \times 3n^2}$, where $d = \sum_{i=1}^q d_i$. Then, for every $j \in \{1, \dots, m\}$, we consider the learnable parameters $(a_{ji}^k)_{i=1}^{d_k}$, or equivalently $a^j = ((a_{ji}^1)_{i=1}^{d_1}, ..., (a_{ji}^q)_{i=1}^{d_q})$. This results in a learnable matrix $A \in \mathbb{R}^{d \times m}$. Let $I \in \mathbb{R}^{d \times 3n^2}$ be a generalized identity matrix with ones on the main diagonal and zeros everywhere else. The weight matrix for the first hidden layer is then $W = A^\top (V + I p_{q+1}) \in \mathbb{R}^{m \times 3n^2}$.

Essentially, we have defined a Categorical posterior distribution over the weights of our network. This construction allows the network to choose which kind of transformations it should be invariant to, according to the data it is presented. While we treat the probabilities $\boldsymbol{\pi}$ in a Bayesian way, all the other parameters can be treated deterministically.

Our goal is to perform classification. Let $\mathcal{Y} = \{1, \dots, m\}$ be the output space, where m is the number of classes in the dataset. Then, we wish to minimize the negative log-posterior over the training data points $(x_i, y_i)_{i=1}^n \subseteq \mathcal{X} \times \mathcal{Y}$, that is, we solve

$$\arg\min_{\boldsymbol{\pi}, A, W} -\log p(\boldsymbol{\pi} \mid \boldsymbol{\alpha}) - \sum_{i=1}^n \log p\left(y_i \mid x_i, \boldsymbol{\pi}, A, W\right),$$

where W are the learnable parameters for the rest of the layers and the likelihood of the labels is Categorical. To estimate gradients for the parameters $\boldsymbol{\pi}$, we use the Gumbel-Softmax reparameterization trick (Jang et al., 2017).

The regularization term of the objective function depends on the concentration hyperparameter $\boldsymbol{\alpha}$. It should be chosen to yield an uninformative prior, in order to be as objective as possible when it comes to learning any potential invariances from the data. In the simplest scenario where $q = 1$, one can choose a Jeffreys prior (Jeffreys, 1946), that is, $(\alpha_1, \alpha_2) = \left(\frac{1}{2}, \frac{1}{2}\right)$. When $q \geq 2$, a uniform choice over the simplex with $\alpha_i \equiv \alpha$, for some $\alpha > 0$ works in a similar way.

The second term tries to learn the right parameter-sharing scheme from the available candidates, while at the same time aiming to correctly classify all the training examples. We hypothesize that the model should be able to detect whether there are symmetries in our data in terms of the final MAP estimate of $\boldsymbol{\pi}$. Consequently, it should closely compete with its frequentist counterparts with the proper weight-sharing schemes in each case.

On the other hand, when trained on non-image data, the estimator should converge to $\hat{\boldsymbol{\pi}}_{MAP} = (0, ..., 0, 1)$, that is, the network should prefer to have a regular multi-layer perceptron (MLP) network structure.

4. Related Work

(Frequentist) Invariant Neural Networks There have been many works that build invariant networks by finding a proper weight-sharing scheme. This is achieved by designing special kinds of layers, which make the resulting models invariant or equivariant to certain types of symmetries, like rotations and reflections (Cohen & Welling, 2016; Ravanbakhsh et al., 2017; Zhang et al., 2018). As our goal was also to learn invariances, we were inspired by Zhou et al. (2021) and Mouli & Ribeiro (2021). In contrast to our method, none of them employed the Bayesian paradigm.

Bayesian neural networks Bayesian neural networks (MacKay, 1992; Neal, 1992) have gained a lot of popularity recently (Blundell et al., 2015; Hernández-Lobato & Adams, 2015; Wenzel et al., 2020; Fortuin et al., 2021a;b; Fortuin, 2021; Izmailov et al., 2021; Immer et al., 2021a;b; D'Angelo et al., 2021; D'Angelo & Fortuin, 2021). However, to the best of our knowledge, none of these works dealt explicitly with inferring posteriors over weight-sharing schemes to learn invariances from data.

Probabilistic approaches to invariance Gaussian processes (GPs) are another probabilistic model that has been used to model invariant functions. In Ginsbourger et al. (2013), it is shown that a GP has invariant trajectories if and only if its kernel is argument-wise invariant. This specific idea is further developed in Ginsbourger et al. (2016) and Ginsbourger et al. (2012). A similar method is developed in van der Wilk et al. (2018), with the difference that there, invariances are incorporated into the structure of the model. This makes it possible to learn symmetries by maximizing the marginal likelihood of the model. These works mainly concern regression and low-dimensional tasks. However, in van der Wilk et al. (2017), a novel algorithm is developed that is suited for classification tasks with high-dimensional data like images.

5. Experiments

We conducted experiments on the MNIST dataset (LeCun, 1998). Hence, our (flattened) input space is $\mathcal{X} = \mathbb{R}^{784}$, since the images are grayscale and of size 28×28. We focus on two groups of symmetries: Rotations of $k \times 90$ degrees, for $k \in \{0, 1, 2, 3\}$, as well as vertical and horizontal flips.

We first carry out a sanity check for our model, by performing data augmentation. Here, we assess whether our model can detect if these invariances are present in the data. Furthermore, we compare its test accuracy with that of other frequentist networks: two invariant architectures as well as one multi-layer perceptron (MLP). Afterwards, we examine how ours and other non-invariant models (with respect to flips and rotations) perform when they are trained without data augmentation, but tested on transformed inputs.

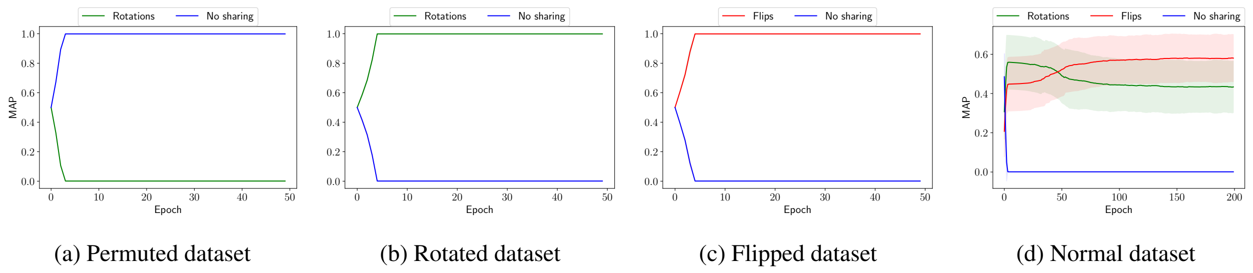
5.1. Validation of our model

We used three different versions of MNIST. Firstly, one where all the pixels were randomly permuted according to a fixed permutation of size 784. This allows the network to still recognize patterns in the images and perform its task, but also creates a situation where there is no symmetry present in the data. For the last two variations, we respectively flip and rotate some of the images. In both cases, every digit was transformed in exactly one way during training. This enables us to challenge the models regarding their extrapolating capacity to other configurations of the inputs. To this end, during testing, we randomly transform all the points. In addition, when it comes to rotations, all the images labeled with 6 or 9 were removed, as they are identical when rotated. An illustration of these datasets can be found in Figure 3.

As mentioned above, we trained three models on these datasets: Ours, which we call InvariantNet, as well as its two frequentist analogues, that is, multi-layer perceptron (MLP) networks with the proper weight-sharing scheme between their input and first hidden layers. These two are called FlipNet and RotNet. Lastly, we used a standard MLP for comparison. We note that in this simple experiment, we are in the case where $q = 1$. Hence, InvariantNet has two different versions, one for rotations and one for flips.

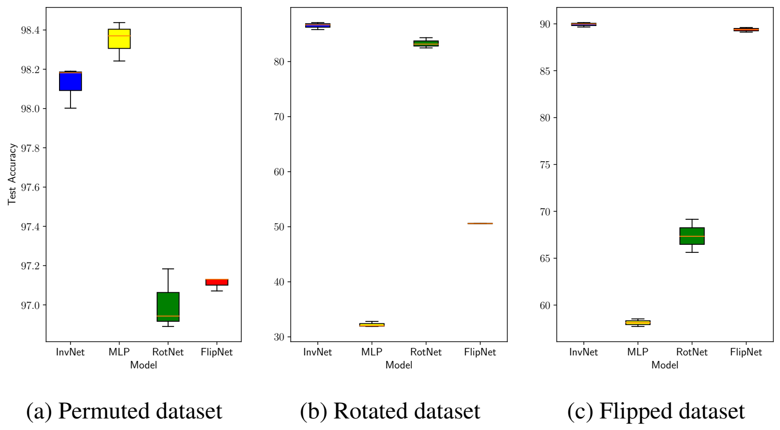
More details on the architectures of the networks and their training are available in Appendix B.

As expected, InvariantNet learns any invariance that was present in the data. This is confirmed by Figures 1a, 1b, 1c, where one can observe the trajectories of the MAP estimates of $\boldsymbol{\pi}$ during training for all three versions of MNIST. Regarding the performances of the different models, which can be inspected in Figure 2, we note that in the transformed cases, our model performed competitively with (and even slightly surpassed) the manually engineered invariant networks, namely RotNet and FlipNet. This is probably explained by the prior-regularization term. Moreover, we see that it was also second-best on the permuted dataset, closely approaching the performance of the MLP.



(a) Permuted dataset (b) Rotated dataset (c) Flipped dataset (d) Normal dataset

Figure 1. Means of the MAP estimates for $\boldsymbol{\pi}$. We see that our network converges to the correct invariances in each case.



(a) Permuted dataset (b) Rotated dataset (c) Flipped dataset

Figure 2. Test accuracies across different models and datasets. When symmetry is present in the data, our model performs competitively with the "correct" hand-engineered models, while discovering the respective invariances directly from the data.

Table 1. Classification accuracies for different models trained on plain MNIST and tested on transformed data. Our model outperforms all the baselines on the transformed test sets.

MODEL	PLAIN	ROTATED	FLIPPED
INVNET	94.2 ± 0.2	$\mathbf{58.2 \pm 1.3}$	$\mathbf{72.3 \pm 1.5}$
CNN	$\mathbf{99.0 \pm 0.2}$	47.2 ± 1.1	63.9 ± 1.0
MLP	98.3 ± 0.3	43.3 ± 0.6	64.1 ± 0.4
BMLP	97.4 ± 0.2	43.1 ± 0.9	63.5 ± 0.8
BCNN	98.6 ± 0.1	47.0 ± 0.5	63.1 ± 0.4

5.2. Out-of-distribution extrapolation

We now focus on the normal MNIST, where we did not synthetically build in any symmetries and study the case $q = 2$, that is, we consider being invariant to both groups simultaneously. Again, by making an uninformative choice for the hyperparameter of the Dirichlet distribution, we encourage the model to take into account all possible choices, namely being invariant to rotations, flips, or having no sharing at all. According to Figure 1d, the model believes that it is almost equally probable that rotations or flips could be present. To test whether this would lead to better generalization, we trained our model and three baselines on the plain dataset without augmentation. Specifically, we chose frequentist and Bayesian MLPs and CNNs (Shridhar et al., 2019). For the Bayesian models, we used a standard Gaussian prior and approximated the posterior distribution with Bayes-by-Backprop, using the local reparametrization trick (Kingma et al., 2015).

We see in Table 1 that InvariantNet clearly outperforms all the non-invariant models when it is asked to infer the classes of transformed examples. This comes at the price of a slightly lower accuracy on plain images, which nonetheless remains quite competitive. This is because the network sacrifices some of its flexibility, which would make it perform better on normal examples, in order to be invariant to flips and rotations during the inference stage.

6. Conclusion

In this paper, we presented a novel probabilistic approach to performing (approximately) invariant classification. Our method is general and can be applied to an arbitrary number of groups, as long as matrix representations of the desired symmetries are available. The proposed model is able to learn invariances of interest from augmented data, as well as to generalize well to examples that were not included during the augmentation. Moreover, we have shown that our network extrapolates well to out-of-distribution samples when trained without data augmentation, compared to other non-invariant models.

An avenue for future work could be to move from MAP estimation to approximation of the full posterior distribution over the parameters of the Categorical distribution. What is more, to obtain uncertainty estimates of the predictions, one could treat all the weights of the network stochastically, thus turning it into a fully Bayesian neural network. Finally, one could try to be completely agnostic and also learn the weight-sharing scheme from the data, by defining a proper priors over the symmetry matrices.

References

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/cohenc16.html.

D'Angelo, F. and Fortuin, V. Repulsive deep ensembles are bayesian. *arXiv preprint arXiv:2106.11642*, 2021.

D'Angelo, F., Fortuin, V., and Wenzel, F. On stein variational neural network ensembles. *arXiv preprint arXiv:2106.10760*, 2021.

Fortuin, V. Priors in bayesian deep learning: A review. *arXiv preprint arXiv:2105.06868*, 2021.

Fortuin, V., Garriga-Alonso, A., van der Wilk, M., and Aitchison, L. Bnnpriors: A library for bayesian neural network inference with different prior distributions. *Software Impacts*, pp. 100079, 2021a.

Fortuin, V., Garriga-Alonso, A., Wenzel, F., Rätsch, G., Turner, R., van der Wilk, M., and Aitchison, L. Bayesian neural network priors revisited. *arXiv preprint arXiv:2102.06571*, 2021b.

Ginsbourger, D., Bay, X., Roustant, O., and Carraro, L. Argumentwise invariant kernels for the approximation of invariant functions. *Annales de la Faculté des sciences de Toulouse : Mathématiques*, Ser. 6, 21(3):501–527, 2012. doi: 10.5802/afst.1343. URL https://afst.centre-mersenne.org/articles/10.5802/afst.1343/.

Ginsbourger, D., Durrande, N., and Roustant, O. Kernels and designs for modelling invariant functions: From group invariance to additivity. In Ucinski, D., Atkinson, A. C., and Patan, M. (eds.), *mODa 10 – Advances in Model-Oriented Design and Analysis*, pp. 107–115, Heidelberg, 2013. Springer International Publishing. ISBN 978-3-319-00218-7.

Ginsbourger, D., Roustant, O., and Durrande, N. On degeneracy and invariances of random fields paths with applications in gaussian process modelling. *Journal of Statistical Planning and Inference*, 170:117–128, 2016. doi: https://doi.org/10.1016/j.jspi.2015.10.002. URL https://www.sciencedirect.com/science/article/pii/S0378375815001640.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869. PMLR, 2015.

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*, 2021a.

Immer, A., Korzepa, M., and Bauer, M. Improving predictions of bayesian neural nets via local linearization. In *International Conference on Artificial Intelligence and Statistics*, pp. 703–711. PMLR, 2021b.

Izmailov, P., Vikram, S., Hoffman, M. D., and Wilson, A. G. What are bayesian neural network posteriors really like? *arXiv preprint arXiv:2104.14421*, 2021.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax, 2017.

Jeffreys, H. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 186(1007):453–461, 1946. ISSN 00804630. URL http://www.jstor.org/stable/97883.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick, 2015.

Kondor, R. and Trivedi, S. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2747–2755. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/kondor18a.html.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Lyle, C., van der Wilk, M., Kwiatkowska, M., Gal, Y., and Bloem-Reddy, B. On the benefits of invariance in neural networks, 2020. URL https://arxiv.org/abs/2005.00178.

MacKay, D. J. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Mouli, S. C. and Ribeiro, B. Neural networks for learning counterfactual g-invariances from single environments. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=7t1FcJUWhi3.

Neal, R. M. Bayesian training of backpropagation networks by the Hybrid Monte Carlo method. Technical report, University of Toronto, 1992.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Equivariance through parameter-sharing, 2017.

Shridhar, K., Laumann, F., and Liwicki, M. A comprehensive guide to bayesian convolutional neural network with variational inference, 2019.

van der Pol, E., Worrall, D. E., van Hoof, H., Oliehoek, F. A., and Welling, M. Mdp homomorphic networks: Group symmetries in reinforcement learning, 2021.

van der Wilk, M., Rasmussen, C. E., and Hensman, J. Convolutional gaussian processes, 2017.

van der Wilk, M., Bauer, M., John, S., and Hensman, J. Learning invariances using the marginal likelihood. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31 (NeurIPS)*, pp. 9960–9970. Curran Associates, Inc., 2018.

Wenzel, F., Roth, K., Veeling, B. S., Świ atkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. How good is the Bayes posterior in deep neural networks really? In *International Conference on Machine Learning*, 2020.

Yarotsky, D. Universal approximations of invariant maps by neural networks, 2018.

Zhang, D., Wang, H., Figueiredo, M., and Balzano, L. Learning to share: Simultaneous parameter tying and sparsification in deep learning,. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rypT3fb0b.

Zhou, A., Knowles, T., and Finn, C. Meta-learning symmetries by reparameterization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=-QxT4mJdijq.

A. Derivations

We first introduce in more detail some algebraic concepts that are used in our method. Then, we will see how one can construct a weight-sharing scheme with respect to a given group. We follow Mouli & Ribeiro (2021).

Let G be a finite group of linear automorphisms over $\mathbb{R}^d$. As a reminder, an automorphism is a linear transformation from $\mathbb{R}^d$ to itself, which is also a bijection.

We now prove the following two lemmas regarding the invariance of the Reynolds operator and the construction of invariant neurons.

Lemma 3. *Suppose $\overline{T}$ is the Reynolds operator of the group G. Then $\overline{T}$ is G-invariant, that is, it holds that: $\overline{T}(Tx) = \overline{T}(x)$, for all $x \in \mathbb{R}^d$ and $T \in G$. Furthermore, $\overline{T}$ is a projection, that is, $\overline{T}^2 = T$.*

Proof. Fix $F \in G$. Then we compute:

$$\overline{T} \circ F = \frac{1}{|G|} \sum_{T \in G} T \circ F = \frac{1}{|G|} \sum_{T' \in G'} T',$$

where $G' = \{T \circ F : \forall T \in G\}$. If we had that $G' = G$, then we would be done. Indeed, as groups are closed under multiplication, it holds that $T \circ F \in G$ for an arbitrary $T \in G$ and as a consequence, $G' \subseteq G$. Now, as F is a bijection, by multiplying with F^{-1} from the right, we get that $T_1 \circ F = T_2 \circ F$ if and only if $T_1 = T_2$, where T_1, T_2 are arbitrary elements of G. Thus, $|G| = |G'|$ and the result follows.

Now, we will show that $\overline{T}$ is a projection. Observe that:

$$\overline{T}^2 = \overline{T}\left(\frac{1}{|G|} \sum_{T \in G} T\right) = \frac{1}{|G|} \sum_{T \in G} \overline{T} \circ T = \overline{T},$$

where in the second equality we used the linearity of $\overline{T}$ and its G-invariance. □

It is easy to verify that the eigenvalues of a projection are either 0 or 1. To see that, fix a non-zero eigenvector w of $\overline{T}$. Then, $w^\top \overline{T}^2 = (w^\top \overline{T})\overline{T} = \lambda w^\top \overline{T} = \lambda^2 w^\top$. On the other hand, $w^\top \overline{T}^2 = w^\top \overline{T} = \lambda w^\top$. Hence, we must have that $\lambda = 0$ or 1.

Let $\Lambda = \left\{w \in \mathbb{R}^d : w^\top \overline{T} = w^\top\right\}$ be the eigenspace of $\overline{T}$, which corresponds to the eigenvalue 1. Then we have the following result that characterizes G-invariant neurons:

Lemma 4. *Let $b \in \mathbb{R}$, $w \in \mathbb{R}^d$ and $f(x) = w^\top x + b$, for $x \in \mathbb{R}^d$. Then, for an arbitrary $T \in G$, we have $f(Tx) = f(x)$ if and only if $w \in \Lambda$.*

Proof. We only prove the first direction. Consider the orthogonal basis $(w_i)_{i=1}^m$ of Λ, which is constituted of the eigenvectors that correspond to the eigenvalue 1.

Fix $0 \neq w \in \Lambda$. Then, we can find $(c_i)_{i=1}^m \subseteq \mathbb{R}$, such that $w^\top = \sum_{i=1}^m c_i w_i^\top = \sum_{i=1}^m c_i w_i^\top \overline{T}$, since $w_i^\top \overline{T} = w_i^\top$ for all $i = 1, ..., m$. Now, for $x \in \mathbb{R}^d$ and $T \in G$, we have that:

$$\begin{aligned} f(Tx) &= w^\top (Tx) + b \\ &= \sum_{i=1}^m w_i^T c_i \overline{T}(Tx) + b \\ &= \sum_{i=1}^m w_i^T c_i \overline{T} x + b \\ &= w^\top \overline{T} x + b \\ &= w^\top x + b \\ &= f(x), \end{aligned}$$

where in the third equality we used the invariance of $\overline{T}$ and in the fifth one that $w \in \Lambda$. □

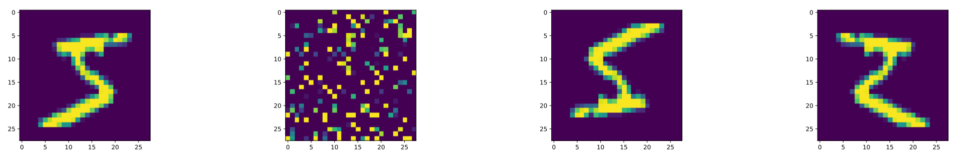
Because of the above, one way to construct a G-invariant neuron is to constraint it to lie in Λ. Hence, to do so, we proceed as follows: We compute $\overline{T}$; we find the basis of Λ; finally, we learn the coefficients $(c_i)_{i=1}^m$.

We give a specific example on how to compute the Reynolds operator for the the following group of rotations: $G = \{Id, T_{90}, T_{180}, T_{270}\}$. For simplicity, we consider that we have inputs of size 2×2, and consequently our (flattened) input domain is $\mathcal{X} = \mathbb{R}^4$. Constructions for inputs of higher dimensions are similar. The corresponding matrix representations of G are the following:

$$\left\{ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \right.$$

$$\left. \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \right\}$$

If we sum these matrices and divide by the cardinality of G, which is 4, we get the representation for the Reynolds operator: $\overline{T} = \frac{1}{4}\mathbf{I}_{4\times4}$, where $\mathbf{I}_{4\times4}$ is the 4×4 square matrix whose entries are all 1. One can easily check that it is G-invariant, that is, it holds that $\overline{T}T_k = \overline{T}$, for $k \in \{0, 90, 180, 270\}$. Lastly, in this case, $\Lambda = \mathit{span}\left\{(1,1,1,1)^\top\right\}$, hence the parameter-sharing scheme is simply $V = (1,1,1,1)^\top$.



(a) Normal dataset (b) Permuted dataset (c) Rotated dataset (d) Flipped dataset

Figure 3. Examples for the four datasets.

B. Implementation Details

For the first experiment, we used three variants of the MNIST dataset (LeCun, 1998): One where we permuted all the pixels, and two augmented versions with flips and rotations. The digits 6 and 9 were removed for rotations. Examples can be found in Figure 3.

All neural networks architectures had one hidden layer of width 100 and were trained until convergence, using early stopping.

Finally, for the second experiment, we gave more capacity to InvariantNet, by using a hidden layer of size 200. For its training, we used $\mathrm{Dir}(2, 2, 2)$ as a prior and temperature 1.