HyperInvariances: Amortizing Invariance Learning

Ruchika Chavhan [1] Henry Gouk [1] Jan Stühmer [2] Timothy Hospedales [1 2]

Abstract

Providing invariances in a given learning task conveys a key inductive bias that can lead to sample-efficient learning and good generalisation, if correctly specified. However, the ideal invariances for many problems of interest are often not known, which has led both to a body of engineering lore as well as attempts to provide frameworks for invariance learning. However, invariance learning is expensive and data intensive for popular neural architectures. We introduce the notion of amortizing invariance learning. In an up-front learning phase, we learn a low-dimensional manifold of feature extractors spanning invariance to different transformations using a hyper-network. Then, for any problem of interest, both model and invariance learning are rapid and efficient by fitting a low-dimensional invariance descriptor an output head. Empirically, this framework can identify appropriate invariances in different downstream tasks and lead to comparable or better test performance than conventional approaches. Our HyperInvariance framework is also theoretically appealing as it enables generalisation-bounds that provide an interesting new operating point in the trade-off between model fit and complexity.

1. Introduction

Exploiting symmetries is crucial to the success of many intelligent systems. Convolutional neural networks most famously provide automatic translation equivariance, which improves learning for a large set of problems where this is an appropriate assumption. Symmetries can be seen as an example of inductive bias, which constrains the search space, or hypothesis class, of a learner. When well matched to the problem at hand, this leads to improvements in learning speed and generalisation and robustness to spurious correlations (Geirhos et al., 2020).

The importance of symmetries and invariances has led to research on architectures to enforce specific inductive biases', such as rotation and scale equivariance (Worrall et al., 2017; Worrall & Welling, 2019) (cf: translation). However, for many problems the ideal invariances are not known a-priori. This has led to a growing body of work on providing the ability to learn invariances, for example, by MAP learning (Benton et al., 2020), marginal-likelihood learning (Immer et al., 2022) and meta-learning (Zhou et al., 2021) (finding a common invariance that works well for a given problem family). Perhaps the most popular way to imbue general architectures with particular invariances is using data augmentation. Optimising for the same output under a given transformation leads to invariance to that transformation (Benton et al., 2020). However, while flexible and simple to implement, this is extremely costly and data intensive.

In a distinct body of work, research has identified a set of augmentations (and hence implicitly invariances) which are widely useful in representation learning for many popular tasks in computer vision (Cubuk et al., 2019). In the explosively growing field of self-supervised learning, this is exemplified by the suite of augmentations used by popular learners such as SimCLR (Chen et al., 2020), which have been analysed in terms of the resulting invariances (Ericsson et al., 2021). However, while these augmentation suites are effective for many problems, they are not ideal for all problems (Ericsson et al., 2021; Xiao et al., 2021). For example, while object recognition may prefer rotation invariance, dense prediction may prefer rotation equivariance.

In this paper we explore the idea of *amortising* invariance learning. In an up-front step, we aim to train a feature extractor that compactly encodes a range of possible invariances. More specifically, we parameterize the feature extractor in terms of a hypernetwork (Ha et al., 2017), which is conditioned on an invariance descriptor. This HyperInvariance architecture thus defines a low-dimensional manifold of feature extractors defined by hypernetwork inputs, with points on the manifold corresponding to different invariance properties. In a task-specific step, we take this frozen (hyper) feature extractor and learn a new prediction head as well as the hypernetwork inputs, which correspond to choice of

[*] Equal contribution [1] School of Informatics, University of Edinburgh [2] Samsung AI Research, Cambridge. Correspondence to: Ruchika Chavhan <r.chavhan@sms.ed.ac.uk>.

Published at the ICML 2022 Workshop on Spurious Correlations, Invariance, and Stability. Baltimore, Maryland, USA. Copyright 2022 by the author(s).

invariance for this task. In practice this framework means that new tasks can be solved – including both model and invariance learning – quickly, data-efficiently, and with few learnable parameters. Because invariances are selected out of a low-dimensional set, they can be chosen very efficiently compared to existing approaches that train general neural architectures for invariances from scratch (Raghu et al., 2021; Immer et al., 2022; Zhou et al., 2021).

From a learning-theoretic point of view, HyperInvariance is also appealing. Generalisation bounds guarantee testing error in terms of empirical risk (training error) plus model complexity. Standard frameworks provide different trade offs between these terms. Flexible deep models provide a good train fit, but poor model complexity, while shallow models provide a weaker train fit, but limited model complexity. HyperInvariance provides an interesting new operating point. By squeezing a rich and relevant variety of feature extractors into a low-dimensional space of hypernetwork inputs, we can provide an improved training fit with limited additional model complexity.

2. Related Work

Invariance Learning Invariances have been created by mean embeddings (Lyle et al., 2020) and learned by MAP (Benton et al., 2020), marginal likelihood (Immer et al., 2022), or meta learning (Zhou et al., 2021; Cubuk et al., 2019; Raghu et al., 2021) – where gradients from the validation set are backpropagated to update the invariances or augmentation distributions. These mostly aim to learn task-specific symmetries, except (Zhou et al., 2021) who learn symmetries for a given family of tasks. All these approaches are highly data and compute intensive due to the substantial effort required to train such symmetries into general purpose architectures. Our HyperInvariance framework amortises the cost of invariance learning so that it is quick and cheap to learn task-specific invariances downstream.

Invariances in Self Supervision Self-supervised methods (Jing & Tian, 2021; Ericsson et al., 2022) often rely on contrastive augmentations (Chen et al., 2020). Their success has been interpreted as engendering invariances (Ericsson et al., 2021; Wang & Isola; Purushwalkam Shiva Prakash & Gupta, 2020) through these augmentations, which in turn provide good inductive bias for downstream tasks. While self-supervision sometimes aspires to providing a single general purpose feature suited for all tasks in the guise of foundation models (Bommasani et al., 2021), studies have shown that different augmentations (invariances) are suited for different downstream tasks, with no single feature being optimal for all tasks. This leads to the tedious need to produce and combine an ensemble of features (Xiao et al., 2021; Ericsson et al., 2021), or task-specific self-supervised pre-training (Raghu et al., 2021), which is extremely costly. Our HyperInvariance breaths new life into the notion of general purpose features by defining a parametric feature extractor that spans an easily accessible range of invariances.

3. Methodology

Pre-train We begin by considering a set of pre-training tasks $\mathcal{T}_{\text{train}}$ with dissimilar invariance requirements. Let the dataset corresponding to task t be $\mathcal{D}^t = \{x_i^t, y_i^t\}_{i=1}^{n_t}$, where n_t is the number of samples per-task. Let f_θ and $\Phi_{\text{train}} = \{\phi^t\}_{t\in\mathcal{T}_{\text{train}}}$ be the shared feature extractor and task-specific decoder weights for training tasks respectively. Predictions for a task t is given by $\hat{y}^t = \langle \phi^t, f_\theta(x^t)\rangle$.

During pre-training the invariance descriptor i^t for each task is assumed observed. This is a vector $i \in [0,1]^K$ over K possible transformations, where $i_k = 1$ and $i_k = 0$ indicate invariance and sensitivity to the kth factor respectively.

We employ hypernetworks to generate weights for the given encoder given the desired invariance descriptor i^t. In this setup, we use a hypernetwork which is shared between all training tasks to generate weights for the encoder to satisfy task-specific invariance requirements.

The hypernetwork h, parametrized by W generates weights for the encoder given a invariance hyper-parameter i_t for task t as $\theta = h_W(i_t)$. Thus, predictions for task t are

$$\hat{y}^t = \langle \phi^t, f_{h(i_t)}(x^t)\rangle \tag{1}$$

In the pre-training stage, the hypernetwork and the corresponding task-specific parameters are updated to optimize the loss of the training tasks.

$$W^\star, \Phi^\star_{\text{train}} = \arg\min_{W,\Phi_{\text{train}}} \frac{1}{|\mathcal{T}_{\text{train}}|} \sum_{t\in\mathcal{T}_{\text{train}}} \frac{1}{n_t} \sum_{j=1}^{n_t} \mathcal{L}(\hat{y}_j^t, y_j^t) \tag{2}$$

Downstream We next consider a set of downstream tasks $\mathcal{T}_{\text{test}}$, that can be solved optimally with different invariance requirements which are unknown to us. We denote the training data available for a downstream task $t' \in \mathcal{T}_{\text{test}}$ as $\mathcal{D}^{t'} = \{x_i^{t'}, y_i^{t'}\}_{i=1}^{n_{t'}}$. In the downstream task training stage, we employ the optimal hypernetwork learned from the pre-training stage to make predictions for the test-tasks. For a downstream task t', let $\hat{y}^{t'} = \langle \phi_t, f_{h^\star(i_t)}(x^{t'})\rangle$ be the predicted output given a invariance hyper-parameter i_t. Subsequently, we evaluate the optimal invariance hyper-parameters and prediction heads Φ_{test} by minimzing the task-specific loss of the training set,

$$i_t^\star, \Phi^\star_{\text{test}} = \arg\min_{i_t,\Phi_{\text{test}}} \frac{1}{|\mathcal{T}_{\text{test}}|} \sum_{t'\in\mathcal{T}_{\text{test}}} \frac{1}{n_{t'}} \sum_{j=1}^{n_{t'}} \mathcal{L}(\hat{y}_j^{t'}, y_j^{t'}). \tag{3}$$

Continuous vs Discrete Invariances Estimates This setup provides the option of (i) using continuous invariance estimates i_t^* and Φ_t^* directly, or (ii) discretizing invariance estimates, $\tilde{i}^* = \text{round}(i_t^*)$, before re-learning Φ. The latter may provide an advantage if i_t^* can be stuck in a local minima, and also provides a learning theoretic advantage.

Theoretical Analysis We provide some theoretical analysis into the potential of our approach to overfit when applied to novel downstream tasks not seen during pre-training in the theorem below.

Theorem 3.1. *For 1-Lipschitz loss function, $\mathcal{L}$, ff for all ϕ we have that $\|\phi\| \leq B$ and $\|f_\phi(x)\| \leq X$, the following holds with probability $1-\delta$*

$$\mathbb{E}_{x^t,y^t}[\mathcal{L}(\hat{y}^t, y^t)] \leq \frac{1}{n_t}\sum_{j=1}^{n_t}\mathcal{L}(\hat{y}_j^t, y_j^t) + \frac{2XB}{\sqrt{n_t}} + 3\sqrt{\frac{\ln(|I|/\delta)}{2n_t}},$$

where $I = \{0,1\}^d$ is the space of possible invariance hyperparameters.

The proof is in the appendix. This theorem demonstrates that the overfitting behaviour of our approach when applied to novel tasks scales similarly to conventional linear models (i.e., $\frac{2XB}{\sqrt{n}}$), but due to the more flexible feature extractor we have a potential to obtain a much better fit on the training data, and therefore a better guarantee on test error. In contrast, formal analysis of the overfitting behaviour of deep neural networks cannot obtain as tight bounds, instead depending exponentially on the depth of the network due to the product of norms of the weights in each layer (Bartlett et al., 2017; Golowich et al., 2018; Long & Sedghi, 2020).

4. Experiments and Results

We conduct synthetic and real-world learning experiments.

Synthetic experiments: Firstly, we perform a set of simple experiments consisting of hand-designed tasks with distinct invariance requirements trained jointly in a multi-task learning setup. *Pre-Train:* We consider color and rotation invariance on a Colored-Rotated MNIST dataset, which is generated by randomly coloring one of the RGB channels of grayscale Rotated MNIST images. In our experiments, Rotated MNIST is generated by rotating MNIST images by randomly selecting an angle from the set $\{-90, -60, -30, 0, 30, 60, 90\}$. We consider three tasks: (i) digit prediction, (ii) rotation prediction and (iii) color prediction. In this setup, digit prediction is ideally color and rotation invariant, while rotation angle prediction is rotation-sensitive and color-invariant and color prediction is rotation-invariant and color-sensitive. Following the convention of assigning invariance descriptors, we assign the invariance hyper-parameters to be $i = [1, 1]$, $i = [1, 0]$ and $i = [0, 1]$ for digit, color, and rotation prediction respectively. We use Augerino (Benton et al., 2020) augmentations to teach the invariances corresponding to each hyper-parameter setting.

	Digit Prediction				Rotation Prediction			
N	$i_\star$	$A_{I_\star}$	$A_{i_\star}$	A_{MTL}	$i_\star$	$A_{I_\star}$	$A_{i_\star}$	A_{MTL}
10	[61, 65]	27.5	**33.3**	22.8	[35, 75]	52.9	**55.8**	45.7
20	[60, 74]	41.9	**42.1**	35.2	[6, 88]	72.1	**74.4**	70.0
50	[63, 80]	47.9	**49.2**	46.7	[2, 93]	77.8	**81.1**	74.1
100	[65, 86]	51.3	**52.5**	47.0	[0 , 86]	79.7	**84.4**	76.3
200	[72, 91]	51.5	**54.2**	48.0	[0, 90]	85.3	**86.9**	83.0

Table 1. Digit and Rotation prediction on ColoredRotated-KMNIST. i^*: estimated invariance strength (%) to (rotation, color). A_{I_*}, A_{i_*}: HyperInvariance accuracy (%) with binarized and continous invariance respectively. A_{MTL} Multi-task baseline accuracy.

Task	HyperSimCLR			Competitors		
	$i^\star$	$A_{I_\star}$	$A_{i_\star}$	SimCLR	Ventral-SimCLR	Dorsal-SimCLR
300W	[46, 59]	74.0	**84.0**	45.0	51.0	82.0
CIFAR10	[99, 96]	76.9	78.4	**78.6**	68.4	50.5

Table 2. Quantitative results for two downstream tasks after SimCLR-ResNet18 training on STL10: CIFAR recognition (accuracy, %) and 300W landmark detection (R^2, %). Our HyperSimCLR outperforms regular SimCLR (Chen et al., 2020) on 300W, and Dorsal/Ventral SimCLRs (Ericsson et al., 2021) on both.

With these tasks, we learn a hypernet derived feature encoder and distinct readout heads (Eqs. 1 and 2). *Downstream:* As a downstream task, we consider digit prediction and rotation prediction on a new domain of Colored-Rotated KMNIST, generated in the same way as the pre-training dataset to examine if the HyperInvariance model can learn appropriate invariances with limited data. Here the (hyper) feature extractor is frozen, and a new readout head and hyperparameters (Eq 3) are learned. *Competitor:* For a baseline we do multi-task learning (MTL) among source tasks with a conventional shared feature extractor, and learn a new head for the downstream task.

Contrastive Learning experiment: We next evaluate our HyperInvariance framework with a real-world contrastive learning experiment, taking SimCLR (Chen et al., 2020) as a representative state of the art learner to build upon. To define a set of invariances of interest, we borrow from Ericsson et al. (2021) who extract two subsets of SimCLR augmentations denoted *dorsal* and *ventral*. Ericsson et al. (2021) showed that contrastive models trained to maximize similarity between images and their dorsal/ventral augmented counterparts learn representations invariant to corresponding transformations. *Pre-train:* We instantiate our framework with SimCLR (hereafter referred to as HyperSimCLR) and a ResNet18 architecture and train on STL-10 (Coates et al., 2011). During training, we assign invariance hyperparameters as $i = [1, 1]$, $i = [1, 0]$ and $i = [0, 1]$ for default, ventral and dorsal augmentations respectively. For every i, the hypernetwork is trained to optimize the contrastive loss for an image and the counterpart with augmentation corresponding to i. *Downstream* Given the learned frozen HyperSimCLR-ResNet18 feature, we train linear readout

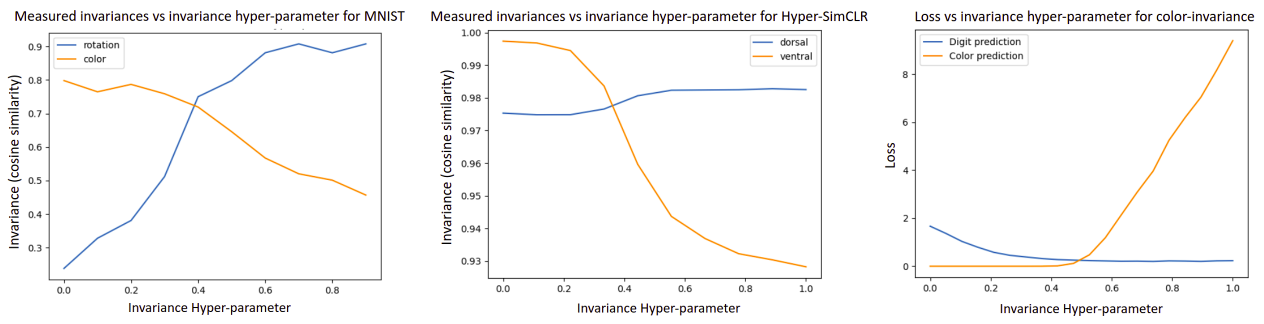

Figure 1. Left: Measured invariances (to rotation and color) of hypernet features as a function of invariance hyperparameter - MNIST experiment. Middle: Measured invariance (to dorsal and ventral transforms) of HyperSimCLR-ResNet18 feature extractor as a function of invariance hyperparameter - STL10. Right: Train loss for digit and rotation prediction vs invariance hyperparameter - STL10/HyperSimCLR.

and hyperparameters for new tasks. For recognition we evaluate CIFAR10 classification, and for regression, we evaluate 300W facial interest point detection (Sagonas et al., 2016).

4.1. Results

With the experiment setup above, we aim to answer the following three research questions:

Does the HyperNetwork learn to generate invariant features? To examine if the hypernetwork has learned features that are provide a specified invariance, we measure invariance by evaluating the cosine similarity between images and their augmented versions when providing different hypernetwork inputs. Figure 1(left) reports the invariance to rotation and color in the synthetic multi-task experiment. At each x-coordinate we pass $[i, 1-i]$ to the hypernetwork to interpolate between rotation and color invariance. The trend shows that invariances broadly vary between minimum and maximum as a function of hyperparameter i. Figure 1(middle) performs the corresponding evaluation for dorsal and ventral augmentations in the SimCLR-ResNet18 experiment. Again we see that measured invariance is a near monotonic function of the hyperparameter. These results show that the hypernet can synthesize features with a desired invariance for both shallow and deep CNNs, and that both supervised multi-task and self-supervised pre-training can be used.

Can the preferred invariance of a downstream task be identified? Given our pre-trained hypernet (on MNIST and STL10 respectively) we study downstream tasks (KMNIST and CIFAR, 300W respectively) by learning a linear readout and invariance hyperparameters. First, for the synthetic experiment, we solve downstream KMNIST with a variety of invariance parameters, interpolating along $[i, 1]$ (color invariance). We report the training loss for digit prediction and color prediction. The results in Figure 1(right) show that loss is a clean monotonic function of the corresponding invariance: Digit prediction is best with maximum color invariance; color prediction is obviously best with minimum color invariance. Both curves are quite smooth, so we expect optimising loss wrt the invariance parameter (x-axis) will discover a good invariance parameter.

Next we check if we can optimise downstream tasks wrt preferred invariance parameter (Eq. 3). The results for synthetic digit prediction and color prediction are shown in Table 1 for N training samples per class. Inspecting Table 1 for digit prediction, we see that invariances are tending towards $[1, 1]$ (and always round to $[1, 1]$) correctly identifying that digit prediction should be both color and rotation invariant. For rotation prediction in Table 1, invariances are tending towards $[0, 1]$ (and always round to $[0, 1]$) confirming that color prediction should be only rotation invariant.

Quantitative results: For the synthetic experiment results in Table 1 shows that (i) HyperInvariance outperforms the multi-task baseline in accuracy, especially at low samples per class – demonstrating the benefit of the correctly discovered invariance as an inductive bias. (ii) Discretizing the invariances usually performs only slightly worse (A_{I_*} vs A_{i_*}), while providing theoretical benefits.

For the HyperSimCLR experiment in Table 2, we can see that for CIFAR-10: (i) Our HyperSimCLR selects both dorsal and ventral invariance (ii) it performs similarly to regular SimCLR baseline and both outperform the dorsal- and ventral-alone baselines. Meanwhile for 300W: (i) HyperSimCLR clearly outperforms regular SimCLR thanks to the ability to tune invariances to a moderate amount with a small preference for Dorsal invariances, and even slightly outperforms DorsalSimCLR which was previously best at this task. Overall, this solves the problem in (Ericsson et al., 2021) where specific choice of dorsal, ventral, or regular SimCLR had to be made on a per-problem basis. A single HyperSimCLR feature can support different tasks through amortised invariance learning.

4.2. Discussion

We have shown that multiple invariances can be pre-learned by a single (hyper) feature extractor. This enables downstream tasks to easily select a useful invariance. This provides an interesting new avenue of study for general purpose features suited for diverse downstream tasks.

Acknowledgements

R.C. was supported by Samsung AI Research, Cambridge.

References

Bartlett, P. L. and Mendelson, S. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 2002.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 2017.

Benton, G. W., Finzi, M., Izmailov, P., and Wilson, A. G. Learning invariances in neural networks. In *Advances in Neural Information Processing Systems*, 2020.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. 2021.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *EEE / CVF Computer Vision and Pattern Recognition Conference*, 2019.

Ericsson, L., Gouk, H., and Hospedales, T. M. Why do self-supervised models transfer? investigating the impact of invariance on downstream tasks. *CoRR*, 2021.

Ericsson, L., Gouk, H., Loy, C. C., and Hospedales, T. M. Self-supervised representation learning: Introduction, advances and challenges. *IEEE Signal Processing Magazine*, 2022.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2020.

Golowich, N., Rakhlin, A., and Shamir, O. Size-independent sample complexity of neural networks. In *Conference On Learning Theory*, pp. 297–299. PMLR, 2018.

Ha, D., Dai, A. M., and Le, Q. V. Hypernetworks. In *International Conference on Learning Representations*, 2017.

Immer, A., van der Ouderaa, T. F., Fortuin, V., Rätsch, G., and van der Wilk, M. Invariance learning in deep neural networks with differentiable laplace approximations. *arXiv*, 2022.

Jing, L. and Tian, Y. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 2021.

Long, P. M. and Sedghi, H. Generalization bounds for deep convolutional neural networks. In *International Conference on Learning Representations*, 2020.

Lyle, C., van der Wilk, M., Kwiatkowska, M., Gal, Y., and Bloem-Reddy, B. On the benefits of invariance in neural networks. *arXiv*, 2020.

Purushwalkam Shiva Prakash, S. and Gupta, A. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. *Advances in Neural Information Processing Systems*, 2020.

Raghu, A., Lorraine, J., Kornblith, S., McDermott, M., and Duvenaud, D. K. Meta-learning to improve pre-training. *Advances in Neural Information Processing Systems*, 34, 2021.

Sagonas, C., Antonakos, E., Tzimiropoulos, G., Zafeiriou, S., and Pantic, M. 300 faces in-the-wild challenge: database and results. *Image and Vision Computing*, 2016.

Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere supplementary material.

Worrall, D. and Welling, M. Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems*, 2019.

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

Xiao, T., Wang, X., Efros, A. A., and Darrell, T. What should not be contrastive in contrastive learning. In *International Conference on Learning Representations*, 2021.

Zhou, A., Knowles, T., and Finn, C. Meta-learning symmetries by reparameterization. In *International Conference on Learning Representations*, 2021.

A. Additional Discussion On Theory

We can contrast the result in Theorem 3.1 with the corresponding result for standard approaches.

The corresponding result for the popular protocol of linear readout from a fixed feature would reduce the third term in the right-hand side of the bound by eliminating the dependence on $|I|$. However, it would also worsen the empirical risk (first term, RHS). This is illustrated in Table 3 where we also show the train performance for HyperInvariance and the MTL baseline – HyperInvariance provides a clearly better training fit.

Meanwhile, the corresponding result for the other popular protocol of fine-tuning the whole feature extractor would improve the first empirical risk term in the bound, but introduce an exploding complexity term in the bound. In particular, the second term of the bound would be proportional to a product of norms of the weight matrices in each layer, thus scaling exponentially with the depth of the network. See, e.g., Golowich et al. (2018) for a demonstration of this in the framework of Rademacher complexity-based analysis.

This is why we describe HyperInvariance as providing an interesting new theoretical operating point between two popular protocols.

Connection to Empirical Results With regard to the discretization operation, our current experiments report binary discretization of the continuously estimated invariance parameter, which works well in the synthetic experiments and CIFAR-10. We remark that our framework and Theorem are compatible with any quantization strength, for example ternary or higher. This may provide a better tradeoff between empirical performance + theoretical guarantees for benchmarks like 300W, where the strong results for our continuous model (A_{i_*}) in Tab 2 suggest that a moderate amount of invariance is preferred.

A.1. Proof of Theorem 3.1

Proof. There is a one-to-one mapping between elements of the finite set of invariance hyperparameters and potential feature extractors, implying that there is also a finite number, $|I|$, of potential feature extractors for a novel task. For a linear model coupled with a pre-selected feature extractor, one can apply the standard generalisation bound for linear models based on (empirical) Rademacher complexity (Bartlett & Mendelson, 2002) to obtain, with probability $1-\delta$,

$$\mathbb{E}_{x^t,y^t}[\mathcal{L}(\hat{y}^t, y^t)] \leq \frac{1}{n_t}\sum_{j=1}^{n_t}\mathcal{L}(\hat{y}^t_j, y^t_j) + \frac{2XB}{\sqrt{n_t}} + 3\sqrt{\frac{\ln(1/\delta)}{2n_t}}.$$

Our result follows from using the union bound to optimise over the choice of feature extractor. □

B. Additional Details

B.1. Synthetic experiments

Hypernetwork: For the multi-task learning experiments, we fix f_θ to be a single convolution layer with 16 filters of kernel size 5×5. This convolution is performed with a stride of 2 and is followed by batch normalization and ReLU before passing the features to the task-specific heads. The hypernetwork is designed as a two-layer network generates weights for the convolution layer given binary invariance descriptors . We design the hypernetwork such that $w_1 \in \mathbb{R}^{k\times d_\text{h}}$, $b_1 \in \mathbb{R}^{d_h}$, $w_2 \in \mathbb{R}^{d_\text{h}\times d_\text{out}}$ and $b_2 \in \mathbb{R}^{d_\text{out}}$. Here, we choose $d_\text{h} = 40$ and subsequently $d_\text{out} = 1200$ which is equal to the total number of weights of the convolution layer. For colored and rotated MNIST images, the output of the hypernetwork is further resized as $3 \times 16 \times 5 \times 5$.

Training Details: For digit prediction, color predictions, and rotation prediction the output cardinality of the task-specific prediction heads is 10, 3, and 7 respectively. During the pre-training stage, the hypernetwork and the task-specific weights are trained for 200 epochs with the Adam optimizer with a learning rate of $5e^{-4}$ along with a cosine annealing learning rate scheduler. During the downstream task learning stage, the invariance hyper-parameters and task-specific decoders are trained with the Adam optimizer with the same learning rate as above.

	Digit Prediction					Rotation Prediction				
N	$i_\star$	$A_{i^\star}^{\text{train}}$	$A_{i_\star}$	$A_{\text{MTL}}^{\text{train}}$	A_{MTL}	$i_\star$	$A_{i_\star}^{\text{train}}$	$A_{i_\star}$	$A_{\text{MTL}}^{\text{train}}$	A_{MTL}
10	[61, 65]	98.2	33.3	59.8	22.8	[35, 75]	100.0	55.8	55.8	45.7
20	[60, 74]	98.1	42.1	71.6	35.2	[6, 88]	100.0	74.3	74.4	70.0
50	[63, 80]	95.7	49.2	73.2	46.7	[2, 93]	97.1	81.1	81.1	74.1
100	[65, 86]	87.7	52.5	64.5	47.0	[0 , 86]	84.4	79.7	84.4	76.3
200	[72, 91]	82.3	54.2	72.9	48.0	[0, 90]	92.5	86.9	86.9	83.0

Table 3. Digit and Rotation prediction on ColoredRotated-KMNIST. i^*: estimated % invariance to (rotation, color). $A_{i^\star}^{\text{train}}$: HyperInvariance train accuracy with continuous invariance, A_{i_*}: HyperInvariance test accuracy with continuous invariance. $A_{\text{MTL}}^{\text{train}}$: Multi-task baseline train accuracy. A_{MTL} Multi-task baseline test accuracy.

B.2. Hyper-SimCLR

Hypernetwork: For contrastive learning experiments, the hypernetwork is trained to generate the parameters of the convolution layers of the Resnet18 model. Unlike the synthetic experiments, the hypernetwork generates kernels for multiple convolution layers of the Resnet18 model. For SimCLR experiments with two types of invariances ($k = 2$), the hypernetwork is designed as a two layer-network with and $d_{\text{h}} = 64$ such that $w_1 \in \mathbb{R}^{2\times 64}$ and $b_1 \in \mathbb{R}^{64}$. However, to generate a convolution layer with parameters denoted by $\{\theta_l\}_{l=1}^{18}$ of the Resnet18 architecture, we learn a separate set of $\{w_2^l, b_2^l\}_{l=1}^{18}$ such that $\theta_l = {w_2^l}^T(\sigma(w_1^T i) + b_1) + b_2^l$. We direct the readers to (Ha et al., 2017) for more details on the hypernetwork architecture that generates all convolution kernels for Resnet18.

Augmentations: In this section of the appendix, we provide details about the augmentation policies used to train Hyper-SimCLR (Ericsson et al., 2021). Ventral augmentation is a set of spatial transformations including random resized crop and random horizonal flip. Dorsal augmentation is a set of appearance changing augmentations consisting random grayscaling, random color jitter and gaussian blurring. Finally, the default augmentations is a combination of both ventral and dorsal augmentations.

Datasets: We pretrain the Hyper-SimCLR model on the unlabeled split of the STL-10 dataset consisting of 100000 images with each image is of size 96×96. We resize these images to 224×224 to match the image size of the downstream tasks. To evaluate the Hyper-SimCLR model on downstream tasks, we perform experiments on 300W landmark detection and CIFAR-10 image classification. On 300W use only the indoor sets where we use 40% of the images to form a test set. For CIFAR-10, 80 % of the labeled data available is used for learning invariances.

Training details: During the pre-trainig stage, the hypernetwork is trained for 200 epochs with the Adam optimizer with a learning rate of $3e^{-4}$, and weight decay coefficient of $1e^{-4}$ along with a cosine annealing learning rate scheduler. During the fine-tuning stage, the invariance hyper-parameters and corresponding task-specific prediction heads are trained for 100 epochs with the Adam optimizer using a learning rate of $3e^{-4}$ with a multi-step learning rate scheduler that decays the learning rate of each parameter group by $\gamma = 0.1$ after every 10 epochs. In this stage, we apply weight decay coefficient of $8e^{-4}$ only to the parameters of the linear readout layers.