META-LEARNING THE INDUCTIVE BIASES OF SIMPLE NEURAL CIRCUITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Animals receive noisy and incomplete information, from which we must learn how to react in novel situations. A fundamental problem is that training data is always finite, making it unclear how to generalise to unseen data. But, animals do react appropriately to unseen data, wielding Occam's razor to select a parsimonious explanation of the observations. How they do this is called their inductive bias, and it is implicitly built into the operation of animals' neural circuits. This relationship between an observed circuit and its inductive bias is a useful explanatory window for neuroscience, allowing design choices to be understood normatively. However, it is generally very difficult to map circuit structure to inductive bias. In this work we present a neural network tool to bridge this gap. The tool allows us to meta-learn the inductive bias of neural circuits by learning functions that a neural circuit finds easy to generalise, since easy-to-generalise functions are exactly those the circuit chooses to explain incomplete data. We show that in systems where the inductive bias is known analytically, i.e. linear and kernel regression, our tool recovers it. Then, we show it is able to flexibly extract inductive biases from differentiable circuits, including spiking neural networks. This illustrates the intended use case of our tool: understanding the role of otherwise opaque pieces of neural functionality, such as non-linearities, learning rules, or connectomic data, through the inductive bias they induce.

1 INTRODUCTION

Generalising to unseen data is a fundamental problem for animals and machines: you receive a set of noisy training data, say an assignment of valence to the activity of a sensory neuron, and must fill in the gaps to predict the valence at other activity values, figure 1A. This is hard because, without prior assumptions, it is completely underconstrained. Many explanations or hypotheses can perfectly fit any set of data (Hume, 1748), Fig 1A, but different choices will lead to wildly different outcomes. Further, the training data that you receive is likely noisy; the vital task of sifting the signal from the noise in a training set can heavily influence generalisation to unseen data, figure 1B.

Generalising requires prior assumptions about the likely explanations of the data. For example, prior belief that small changes in the neural activity should lead to correspondingly small changes in valence would bias you towards smoother explanations of the data, breaking the tie between options 1 and 2 in figure 1A. It is a circuit's inductive bias that leads it to choose certain, otherwise similarly well-fitting, explanations of the data, over others.

The inductive bias of a learning algorithm, such as a neural network, can be a powerful route to understanding in both Machine Learning and Neuroscience. Classically, the success of convolutional neural networks can be attributed to their explicit inductive bias towards translation-invariant classifications (LeCun et al., 1998), and these ideas have since been very successfully extended to networks with a range of structural biases (Bronstein et al., 2021). Further, a wide variety of network features have been linked to implicit regularisation of the network, such as: the stochasticity of SGD (Mandt et al., 2017), parameter initialisation (Glorot & Bengio, 2010), early stopping (Hardt et al., 2016), low rank biases of gradient descent Gunasekar et al. (2017), or the discretisation of gradient steps (Barrett & Dherin, 2020).

In neuroscience, the inductive bias has been used to give normative explanations for representational or structural choices via their effect on generalisation. For example, there has been recent work on

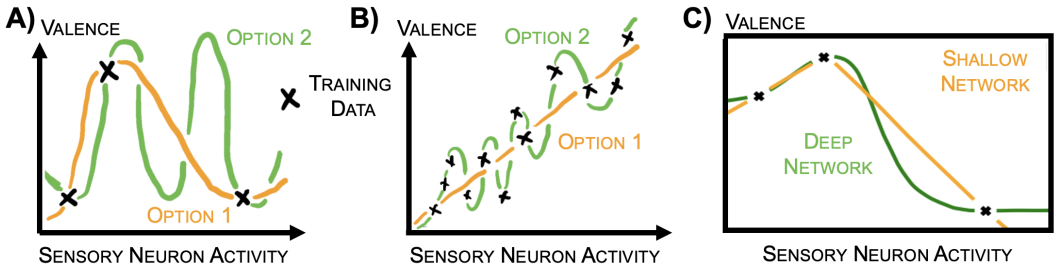

Figure 1: **Classification Requires Prior Assumptions. A:** The same set of data can be perfectly fit by many functions. **B:** Different assumptions about the signal quality of the data leads to very different fittings. **C:** Training a 2 (shallow) or 8 (deep) layer ReLU network in the same way on the same dataset leads to very different generalisation properties for non-obvious reasons.

neural network models of the cerebellum that has shown that the threshold of the network's non-linearity has a strong effect on its ability to generalise functions with different frequency content (Xie et al., 2022). It is experimentally observed that these network properties vary across the cerebellum, and hence this work suggests that each part of the cerebellum may be tuned to tasks with particular smoothness properties. This is exemplary of a spate of recent papers that have applied similar techniques to visual representations (Bordelon et al., 2020; Pandey et al., 2021), mechanosensory representations (Pandey et al., 2021), and the olfactory system (Harris, 2019).

Despite the potential utility of inductive bias as a way to understand neural circuits, the approach is limited, since mapping between learning algorithms and their inductive bias is highly non-trivial. Numerous aspects of a circuit (learning rules, architecture, non-linearities, etc.) will influence the way it chooses to generalise. For example, training two simple ReLU networks of different depth to classify three data points leads to fundamentally different generalisations for non-obvious reasons, figure 1C. In a few constrained cases analytic bridges have been built that map a learning algorithm to its inductive bias. Most notably, work has characterised the inductive bias of kernel regression, an algorithm that maps data points to a feature space in which linear regression to labels is then performed (Sollich, 1998; Bordelon et al., 2020; Simon et al., 2021). This approach has been influential; all the cited examples of understanding neuroscientific findings via their induced inductive bias have used this analytic bridge. However, it severely limits the applicability of this approach: most biological circuits cannot be well approximated as performing a fixed feature map then linearly regressing to labels!

Here, we develop a flexible neural network approach that is able to meta-learn the inductive bias of essentially any differentiable supervised learning algorithm. It follows a meta-learning framework (Vanschoren, 2019): an outer neural network (the meta-learner) assigns labels to a dataset, this labelled dataset is then used in the inner optimisation to train the inner neural network (the learner). The meta-learner is then trained on a meta-loss which measures the generalisation error of the learner to unseen data. Through gradient descent on the meta-loss, the meta-learner meta-learns to label data in a way that the learner finds easy to generalise. These easy-to-generalise functions form a description of inductive bias. If a function is easy to generalise that implies that when the network receives a few training points labelled according to this function, it chooses to generalise them in the appropriate manner. They are therefore functions the network would regularly use to explain finite datasets, i.e. if function is easy-to-generalise for a given network then the network is inductively biased towards that function.

To our knowledge, the most related work is Li et al. (2021). Li et al. create a set of neural networks, trained or untrained, then attempt to summarise this set via a distribution over labels as a function of input. To do this, they meta-learn the parameters of a gaussian process, which assigns a distribution of labels to each input. This provides an interpretable way to understand the behaviour of sets of networks, rather than our work which characterises the inductive bias of learning algorithms through the functions they find easy to generalise. We recommend potential users to consider both approaches depending on their application.

In the following sections we describe our scheme, and verify that it works by comparing to the known inductive biases of linear and kernel regression. We then extend it in several ways. First, networks are inductively biased towards areas of function space, not single functions. To explore this, we learn a set of orthogonal functions that a given network finds easy to learn, hence providing a richer characterisation of the inductive bias. Second, we introduce a setup for gaining insight into the way a given design choice (architecture, learning rule, non-linearity) effects the inductive bias. We do this by assembling two networks that differ only by the design choice in question; then we meta-learn a function that one network finds much easier to generalise than the other, which might allow a user to answer why a particular feature of the circuit is present. We will again verify both of these techniques against algorithms with known inductive biases. Finally we will apply our approach to more adventurous settings. Simple networks trained by gradient descent; and spiking neural networks, trained using the surrogate gradient method (Neftci et al., 2019; Zenke & Vogels, 2021), a recent major advance in gradient-based training of spiking networks. In both cases we show our tool can be used to examine the networks' inductive biases.

2 A Neural Network to Meta-Learn Inductive Bias

Our main contribution is a meta-learning framework for extracting the inductive bias of differentiable learning algorithms, figure 2A, that we will describe and test in this section. The outer-loop is a neural network, the meta-learner, that assigns labels to inputs sampled from some distribution, hence creating the real-world function that our circuit of interest will try to learn. The inner-loop learning algorithm, the learner, is the circuit whose inductive bias we would like to extract, for example a model of a biological sensory processing circuit that assigns valences to inputs. When provided with a training dataset of inputs and labels the learner adjusts its parameters according to its internal learning rules. Then the generalisation error of the trained learner is measured on a held-out test set, and this is used as the meta-loss to train the meta-learner. This process repeats, retraining the learner at every iteration and iteratively developing the weights of the meta-learner, until the meta-learner is performing a labelling for which the learner has a low generalisation error after training on a small number of datapoints (we used around 30). Thus, the meta-learner has extracted a function towards which the learner is inductively biased.

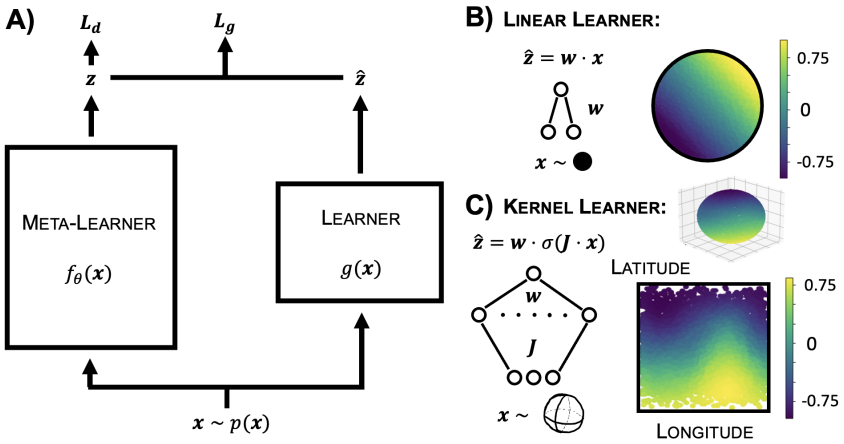

Figure 2: **Meta-Learning the Inductive Bias. A: Schematic** Meta-learner labels a dataset which is used to train the learner. Then gradient descent is performed on the generalisation error of the learner to unseen data ($\mathcal{L}_g$), and the Sinkhorn divergence between the meta-learner's label distribution and a target distribution ($\mathcal{L}_d$), which we choose to be uniform between -1 and 1. **B: Linear** The meta-learner learns to label data sampled from a circle in a linearly separable way when the learner is ridge regression. **C: Kernel** For a kernel regression learner labelling data sampled from the surface of a sphere the meta-learner labels the data with a function that is very close to the predicted first order spherical harmonic (99% of norm within first order spherical harmonics). The labelling on the sphere is shown, and then shown projected onto a clearer 2D plot that we will use henceforth.

As just outlined, the meta-learner will find the function that the learner finds easiest to generalise, which in most cases is the trivial function where all inputs are assigned the same label (similar to GAN mode collapse (Lala et al., 2018)). To avoid this, we will introduce a term in the meta-loss that forces the distribution of labels to take a particular (non-constant) form. Specifically, we introduce a term in the meta-loss that measures the Sinkhorn divergence between the assigned label distribution and a uniform distribution between -1 and 1. The full algorithm pseudocode is shown in Algorithm 1.

Algorithm 1: Pseudocode for Meta-Learning the Learner's Inductive Bias

```
Initialise metalearner: f_θ(x)
while Step count < Total do
    Generate dataset from input distribution: x ~ p(x)
    Label using metalearner: z = f_θ(x)
    Split inputs and labels into test and train datasets: D_Tr & D_Te
    Train leaner using D_Tr giving trained learner network: g(x)
    Label D_Te using trained learner: ẑ = g(x)
    Compute the generalisation error of the leaner: L_g = Σ_i (z_i − ẑ_i)^2
    Compute the Sinkhorn Divergence of metalearner's labels from uniform [−1, 1]: L_d
    Take θ gradient step on meta-loss: L = L_g + L_d
end
```

We now perform a couple of tests to show this scheme works by meta-learning sensible functions for linear and kernel regression, whose inductive biases are understood. First, for ridge regression on data sampled from a 2D circle the meta-learner assigns linearly separable labels to the data, figure 2B; exactly the labelling to which the simple linear circuit can easily generalise.

Next, we meta-learn kernel ridge regression's bias. Kernel regression involves projecting the input data through some fixed mapping to a feature space (e.g. the last hidden layer of a fixed neural network) and performing linear regression to the labels from the feature space. Bordelon et al. (2020) have shown that the inductive bias of kernel regression algorithms can be understood through the kernel eigenfunctions ($\{v_i(\boldsymbol{x})\}$ with eigenvalue $\{\lambda_i\}$). These are defined on an input distribution $p(\boldsymbol{x})$ via a kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ that measures similarity in features space as:

$$\int k(\boldsymbol{x}, \boldsymbol{x}') v_i(\boldsymbol{x}') dp(\boldsymbol{x}') = \lambda_i v_i(\boldsymbol{x}). \tag{1}$$

The higher the eigenvalue the more inductively biased the kernel regression algorithm is towards this eigenfunction, i.e. the fewer training points needed to reach a given generalisation error when trained to fit this eigenfunction. More general functions can be understood by how much they project onto high and low eigenvalue eigenfunctions. This tells us that our meta-learner, in searching for the non-constant function that kernel regression finds easiest to generalise, should pick out the eigenfunction with highest eigenvalue.

To test this we meta-learn the inductive bias of a two-layer neural network with fixed first layer weights. We sample data uniformly from the sphere and randomly connect a large hidden layer of ReLU neurons to the three input neurons. The elements of this random weight matrix are drawn iid from a standard normal, and the learning algorithm then performs ridge regression on the hidden layer activities. Previous work has analytically derived the kernel for this network and it's eigenfunctions (Cho & Saul, 2009; Mairal & Vert, 2018), which are spherical harmonics. The higher the frequency of the spherical harmonics the lower its eigenvalue. Matching this, our network meta-learns one of the set of lowest frequency spherical harmonics, figure 2C.

3 Meta-Learning multiple functions

Having shown our tool works on some simple test cases, we now extend it to find a richer characterisation of the inductive bias. A given learning algorithm is inductively biased towards areas of function space, not just one particular function. In order to gain access to this larger space we

learn a series of meta-learners. The first of these is exactly as described above, it is trained with a meta-loss that measures the generalisation error and the sinkhorn divergence of the label distribution from uniform. Then we iteratively introduce additional meta-learners. To ensure each meta-learner learns a new aspect of the inductive bias we add a term to the meta-loss that penalises the square of the dot product between the current meta-learner's labelling and that of all the previously trained meta-learners, figure 3A. I.e. for the orthogonality loss between $f_\theta(\boldsymbol{x})$ and a previously trained meta-learner $h(\boldsymbol{x})$ we sample a set of data and compute,

$$\mathcal{L}_{\text{Orthog}} = \left(\sum_i h(\boldsymbol{x}_i) f_\theta(\boldsymbol{x}_i) \right)^2 \tag{2}$$

This forces each subsequent learner to discover a function that is orthogonal to all previous functions, but is still relatively easy for the learner to generalise.

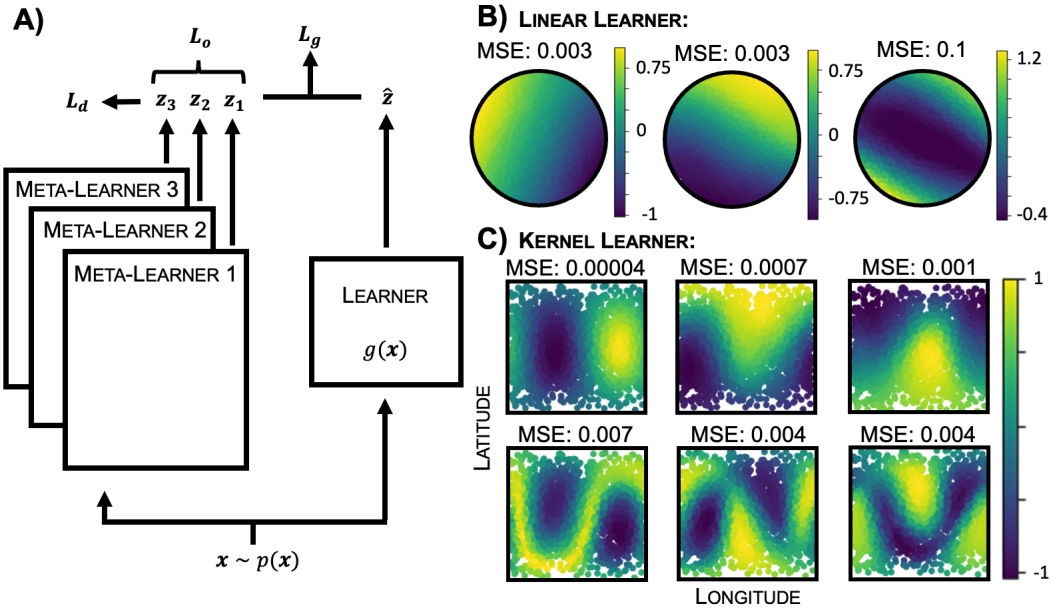

Figure 3: **Meta-Learning many functions. A: Schematic** We learn many meta-learners, each of which has to learn a labelling that is additionally orthogonal to all previous meta-learners. **B: Linear** The meta-learners learn two orthogonal linear functions and an orthogonal but hard to learn third function. **C: Kernel** We learn 6 meta-learners that first learn three first order spherical harmonics, then three linear combinations of the five second order spherical harmonics, as predicted.

We again check that this scheme is working by comparing to learning algorithms with known inductive biases. When tested on linear regression of 2D data the meta-learners learn two orthogonal linearly separable sets of labels, then finds a third orthogonal function that the learner struggles to generalise, as expected, figure 3B. We then test on the same kernel regression network we described previously. Theory predicts that the meta-learners should learn the eigenfunctions in decreasing order of their eigenvalue, and we find that this is true to a good approximation, figure 3C, learning approximations to the three first order spherical harmonics, and then three approximations to second order spherical harmonics.

For linear classifications (e.g. linear and kernel regression) the full set of orthogonal functions explains the entire inductive bias, since the average generalisation error of any target function is the sum of the generalisation errors of each the kernel eigenfunctions weighted by projection of the target function onto that eigenfunction. This will not in general be true, since for a non-linear classifier the generalisation error on a function $f(\boldsymbol{x}) = f_a(\boldsymbol{x}) + f_b(\boldsymbol{x})$ does not equal the generalisation error on $f_a(\boldsymbol{x})$ plus that on $f_b(\boldsymbol{x})$. Nonetheless, we expect the set of orthogonal functions will still be a helpful guide to a network's inductive bias, even for non-linear classifiers.

4 Finding the Effect of Design Choices on the Inductive Bias

Our work is motivated by the desire to understand how design choices in learning algorithms - such as architecture, learning rule, and non-linearities - lead to downstream generalisation effects, particularly in biological neural networks. The ideas we have presented are already capable of answering many of these questions. However, one additional setting which we have found useful is to compare two networks with some architectural difference between them, and learn functions that one of the networks finds much easier to generalise than the other. In this way, you can build intuition for the impact of design features on the inductive bias of the network. For example, you may find a biological network with a particular type of non-linearity not seen elsewhere in the brain. By finding a function that a network with this non-linearity learns much better than one without, you may gain insight into the role of the non-linearity.

To examine this we again create a meta-learner that labels data, but this time the labels are used to train two learners. We then train the meta-learner so that one learner (the chosen student) is much better at generalising than the other (neglected student). This is done by minimising the generalisation errors of the chosen student minus the neglected student, figure 4A. We again test this approach on well understood algorithms, and show that it can find functions that a kernel regression algorithm is able to learn better than linear regression, figure 4B, i.e. a higher frequency, non-linearly seperable function.

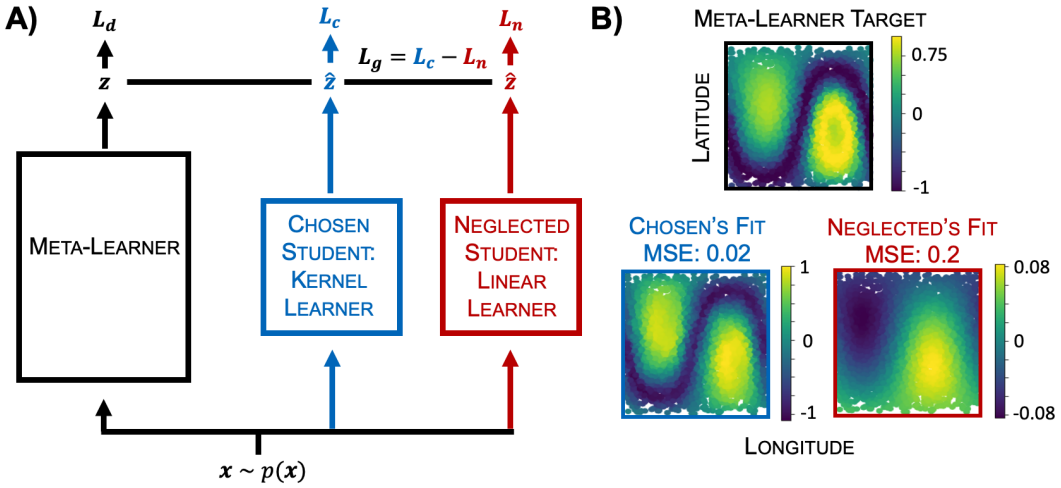

Figure 4: **Meta-learning design choice impact. A: Schematic** Labellings are learnt that a chosen student generalises much better than a neglected one. **B: Kernel vs. Linear** When this meta-learner is trained to label in a way that kernel but not linear regression is able to fit it learns a non-linearly separable function that the kernel fits and order of magnitude better than linear regression.

This illustrates some of the games that could be played in this setting. For example, you could play a co-operative game, in which you meta-learn a function that a set of learners all find easy to generalise, and each learner could have different connectivity matrices to match the distribution in real animals, to ensure the tool does not over-fit to some specific details. However as the losses become more complex training becomes harder, for example this adversarial setting between chosen and neglected student is hard to make robust if the two learners are relatively similar.

5 Meta-Learning Applied to More Complex Learning Algorithms

In the previous sections we have developed and tested a suite of ways to meta-learn the inductive bias of simple learning algorithms. We now apply our tools to networks whose inductive bias cannot be understood analytically. By taking gradients through the training procedure we can meta-learn inductive biases for networks trained using PyTorch, which includes standard networks trained via backpropagation, or, thanks to recent advances in the surrogate gradient method (Neftci et al., 2019),

spiking neural networks. We will provide code on github that produces all our figures, which should be easily adapted to networks of interest.

5.1 META-LEARNING FOR BACKPROP TRAINED NETWORKS

We begin by testing our framework on a simple feedfoward ReLU network with 2 hidden layers, learnt using gradient descent. While the functions this network finds easy to generalise cannot be extracted analytically, our tool finds that, unsurprisingly, these networks are also inductively biased towards smooth explanations of the data, learning six smooth orthogonal classifications that increase in frequency and mean squared error 5B. We include this example as the simplest PyTorch implementation of learning the inductive bias of a network trained by gradient descent, in the hope that the code can be easily adapted for future use.

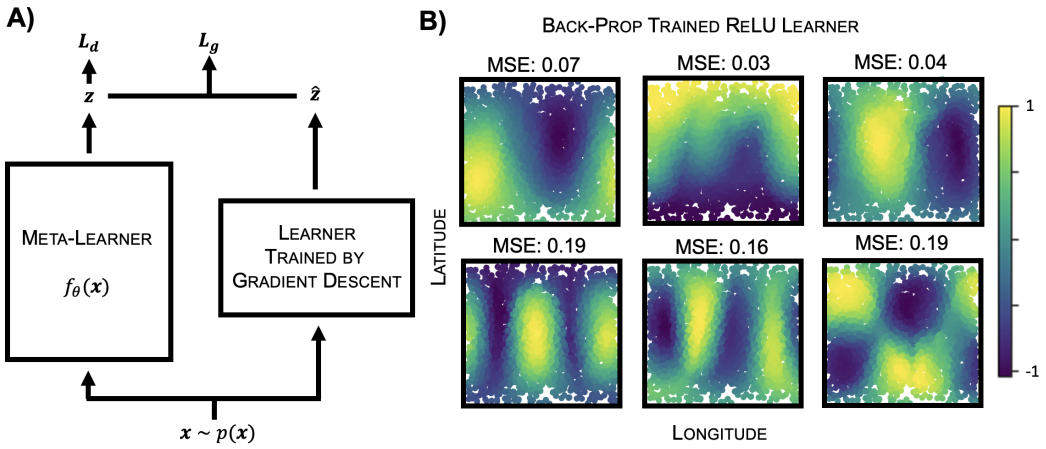

Figure 5: **Meta-learning a Backprop Trained Network A: Schematic** We meta-learn the functions a 2-layer ReLU network trained using backprop finds easy to generalise. **B: Low Frequency Bias**

5.2 SPIKING NEURAL NETWORK

Surrogate gradient methods allow gradient based training of spiking neural networks, (Neftci et al., 2019; Zenke & Vogels, 2021). The key difficult of gradient based approaches to training spiking networks is that the gradient is discontinuous: 0 at all times until a spike arrives, then infinite. Surrogate gradient methods circumvent this by smoothing the gradient, providing a signal for training.

We make use of this development to meta-learn functions that a spiking network finds easy to generalise. The network we study is a modification of a model developed for a recent tutorial (Goodman et al., 2022; Zenke, 2019), which is trained to assign a label to an incoming spike train. The network is a model of an interaural phase difference detection circuit. The input spike train is parameterised by a phase difference, $\Delta\phi$, that generates two sets of spike trains, one in each ear, figure 6A. These spikes are then processed through a hidden layer of linear-integrate-and-fire neurons (LIF), before reaching a classification layer. The valence assigned by the circuit is a real number, calculated by summing the output neurons activity over the length of the trial. Then the meta-learning framework is as before: the meta-learner assigns valences to input phase differences, these labels are used to train the spiking neural network by surrogate gradient descent, then the generalisation error is computed on heldout data and the meta-learner is trained to minimise this error and a distribution loss.

Our resuilts, figure 6B, show that the inductive bias is relatively simple. We learn the three orthogonal easiest-to-generalise labellings, using the same method as in section 3, and find that they express a simple smoothness prior: low frequency labellings are easy to learn.

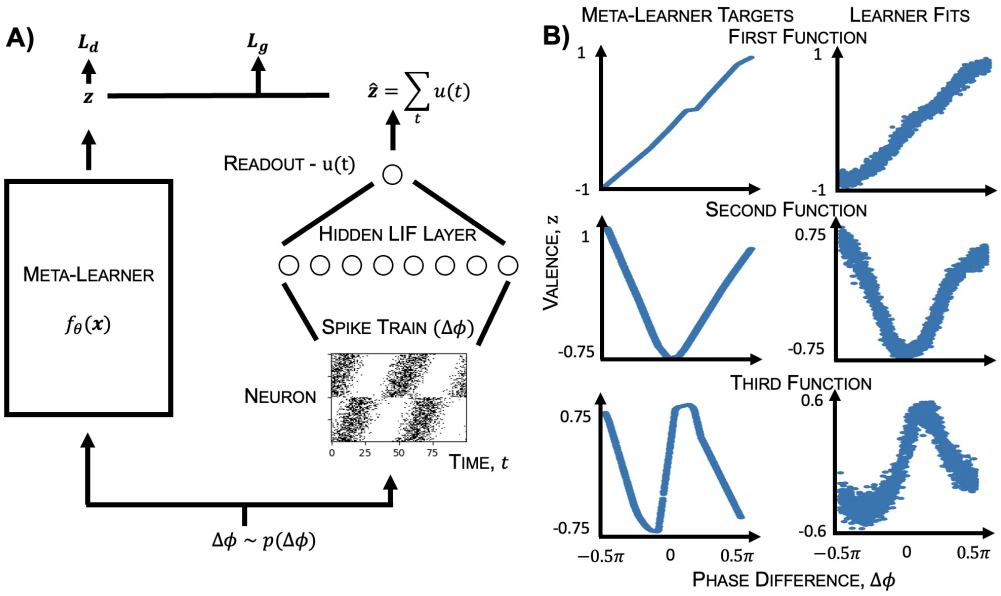

Figure 6: **Meta-learning through a spiking neural network. A: Schematic** Labellings are learnt that the spiking network, with weights trained via surrogate gradient descent, finds easy to generalise. Phase differences, $\Delta\phi$, are sampled uniformly and used to generate spike train by sampling from a poisson process with the following rates: for half the neurons $r_n = \frac{r_{\max}}{2}(1 + \sin(t + \theta_n))^2$, where n is a neuron index and θ_n are uniformly sampled offsets; for the other half there is an additional phase shift: $r_n = \frac{r_{\max}}{2}(1 + \sin(t + \theta_n + \Delta\phi))^2$. These two populations represent sensory neurons in the two ears, and $\Delta\phi$ is the phase difference between them. This activity feeds forward into a population of linear-integrate-and-fire neurons, and onwards to a readout linear-integrate neuron. The valence assigned is the sum of the readout's activity over time. **B: Inductively Biased towards Smoothness** The spiking network finds it easiest to learn low frequency functions, as evidenced by the three orthogonal functions we meta-learn that have increasingly high frequency components. On the left is the meta-learner's target function, the right shows the spiking network's labelling. As can be seen, the spiking network captures the main behaviour, but increasingly poorly at higher frequencies.

6 Discussion

Here we have presented a meta-learning approach to extract the inductive bias of differentiable supervised learning algorithms, which we hope will be useful in normatively interpreting the effect of particular features of biological networks. This approach required few assumptions on the biological network beyond those that make the inductive bias an interesting way to conceptualise the circuit in the first place. We required, first, the circuit must be interpretable as performing some supervised learning mapping. Second, you must specify the input data. And, third, you must specify, and be able to take gradients through, the way in which the circuit learns to classify data. We will discuss each of these three limitations and ways in which they could be relaxed. Regardless, it is heartening that any circuit satisfying these will, in principle, suffice. The previously discussed analytic bridge between kernel regression and its inductive bias (Bordelon et al., 2020; Simon et al., 2021) has already found multiple uses in biology in just a few years (Bordelon & Pehlevan, 2021; Pandey et al., 2021; Harris, 2019; Xie et al., 2022), despite its stringent assumptions. We hope that by relaxing many of those assumptions we can offer a route to allow these ideas to be applied more broadly.

Despite our optimism for this broad approach, there remain many challenges. Most fundamentally, sets of functions that a given circuit finds easy to learn from few training examples are still relatively hard to interpret. In this work we have stuck to low dimensional spaces in which the resulting functions can be interpreted by eye. In this vein, building low-dimensional toy models of biological systems from which inductive biases can be extracted does seem a plausible way to make progress in some cases, but will not be universal. To make the concept of inductive bias more helpful, more tools

are needed to interpret the resulting functions. Studying the inducitve bias on established datasets, such as MNIST, seems productive, since there are some ground truth labellings to compare to. Other simple ideas like looking at the frequency spectrum, or projecting onto some other function basis, might show some details, but not all. Our framework maps algorithms to their inductive bias, more ideas are needed to turn those extracted biases into interpretable findings.

The first requirement for our tool to be applicable was an input data distribution, which often will not be easy to guess. This can be avoided by, for example, using real neural data as the input. Alternatively, if there is a limited set of neural data, generative modelling techniques could be used to fit the distribution of neural data and new samples drawn from that distribution. Finally, you could imagine a single meta-learner that creates not only the label, but also the datapoint that a learner finds easy to fit. I.e. the meta-learner could generate the entire dataset by transforming a noise sample into an input-output pair. This would have to be carefully regularised in order to avoid trivial input distributions, but could in principle learn the input statistics that particular networks are tuned to process.

Next, we could imagine relaxing our second assumption, that the network performs simple supervised learning. Extracting a circuit from the brain and conceptualising it as performing supervised learning will always be an approximation. Most simply, there will almost always be effects elsewhere in the brain, like attentional focus, that will have an unmodelled impact on the function of the circuit. As an illustration of how these could be included in our framework, attention is often thought to happen through neuromodulation (Eldar et al., 2013). There exist models of how neuromodulation influences circuit function, for example by sharpening neural non-linearities (Ferguson & Cardin, 2020; Aston-Jones & Cohen, 2005), and these could be included in the learner model. The meta-learner could then meta-learn two outputs, one label and one neuromodulator. It's goal would be to meta-learn functions that the learner finds easy to learn with limited quantities of neuromodulatory attention applied to specific, exemplar, training points.

Finally, in a slightly cooky way, we could avoid even needing to specify the form of the learning algorithm, by interfacing with an animal directly! Animals' inductive biases are objects of interest in their own right, but could also give insight into the underlying neural processing. These biases could be studied by replacing the inner learner with a real animal that is trained on a labelled dataset from the meta-learner, then tested on new datapoints. Since you cannot compute gradients through the computations of a living animal (I think...), the meta-learner would have to use signals computed using only meta-loss evaluations. This could be done using black-box optimisation procedures, like the Nelder-Mead method (Singer & Nelder, 2009). The meta-learner would then meta-learn labellings that the real animal is proficient at generalising!

7 Conclusion

The inductive bias is a promising angle from which to understand pattern classifiers. Analytic bridges between circuit design and inductive bias 'explain' the presence of aspects of the circuit through their effect on the network's generalisation properties. This has aided understanding in both artificial (Canatar et al., 2021; Bahri et al., 2021) and biological (Bordelon & Pehlevan, 2021; Pandey et al., 2021; Harris, 2019; Xie et al., 2022) networks, however these techniques require very constraining assumptions. Here we have presented and tested a meta-learning approach to flexibly extract the inductive bias from neural circuits. We have shown its broad applicability by applying it to simple networks trained by backprop and spiking neural networks. As new and large neuroscientific datasets mature, such as connectomics (Zheng et al., 2018), methods to interpret the results will increasingly be needed. By building a link between structural and functional domains, we hope this tool will allow otherwise opaque features of biological neural networks to be interpreted and understood, through their impact on the network's inductive bias.

References

Gary Aston-Jones and Jonathan D Cohen. An integrative theory of locus coeruleus-norepinephrine function: adaptive gain and optimal performance. *Annu. Rev. Neurosci.*, 28:403–450, 2005.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021.

David GT Barrett and Benoit Dherin. Implicit gradient regularization. *arXiv preprint arXiv:2009.11162*, 2020.

Blake Bordelon and Cengiz Pehlevan. Population codes enable learning from few examples by shaping inductive bias. *BioRxiv*, 2021.

Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In *International Conference on Machine Learning*, pp. 1024–1034. PMLR, 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.

Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature communications*, 12(1):1–12, 2021.

Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. *Advances in neural information processing systems*, 22, 2009.

Eran Eldar, Jonathan D Cohen, and Yael Niv. The effects of neural gain on attention and learning. *Nature neuroscience*, 16(8):1146–1153, 2013.

Katie A Ferguson and Jessica A Cardin. Mechanisms underlying gain modulation in the cortex. *Nature Reviews Neuroscience*, 21(2):80–92, 2020.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Dan Goodman, Tomas Fiers, Richard Gao, Marcus Ghosh, and Nicholas Perez. Spiking neural network models in neuroscience - cosyne tutorial 2022. https://zenodo.org/record/7044500#.Yy7crezML9s, 2022.

Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. *Advances in Neural Information Processing Systems*, 30, 2017.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pp. 1225–1234. PMLR, 2016.

Kameron Decker Harris. Additive function approximation in the brain. *arXiv preprint arXiv:1909.02603*, 2019.

David Hume. *An enquiry concerning human understanding and other writings*. 1748.

Sayeri Lala, Maha Shady, Anastasiya Belyaeva, and Molei Liu. Evaluation of mode collapse in generative adversarial networks. *High Performance Extreme Computing*, 2018.

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.

Michael Y Li, Erin Grant, and Thomas L Griffiths. Meta-learning inductive biases of learning systems with gaussian processes. In *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2021.

Julien Mairal and Jean-Philippe Vert. Machine learning with kernel methods. *Lecture Notes, January*, 10, 2018.

Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

Biraj Pandey, Marius Pachitariu, Bingni W Brunton, and Kameron Decker Harris. Structured random receptive fields enable informative sensory encodings. *bioRxiv*, 2021.

James B Simon, Madeline Dickens, and Michael R DeWeese. Neural tangent kernel eigenvalues accurately predict generalization. *arXiv preprint arXiv:2110.03922*, 2021.

Saša Singer and John Nelder. Nelder-mead algorithm. *Scholarpedia*, 4(7):2928, 2009.

Peter Sollich. Learning curves for gaussian processes. *Advances in neural information processing systems*, 11, 1998.

Joaquin Vanschoren. Meta-learning. In *Automated machine learning*, pp. 35–61. Springer, Cham, 2019.

Marjorie Xie, Samuel Muscinelli, Kameron Decker Harris, and Ashok Litwin-Kumar. Task-dependent optimal representations for cerebellar learning. *bioRxiv*, 2022.

Freidemann Zenke. Spytorch. https://zenodo.org/record/3724018#.Yy7coOzML9t, 2019.

Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4):899–925, 2021.

Zhihao Zheng, J Scott Lauritzen, Eric Perlman, Camenzind G Robinson, Matthew Nichols, Daniel Milkie, Omar Torrens, John Price, Corey B Fisher, Nadiya Sharifi, et al. A complete electron microscopy volume of the brain of adult drosophila melanogaster. *Cell*, 174(3):730–743, 2018.