This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Do We Always Need the Simplicity Bias? Looking for Optimal Inductive Biases in the Wild

Damien Teney
Idiap Research Institute

Liangze Jiang
EPFL

Florin Gogianu
Bitdefender

Ehsan Abbasnejad
University of Adelaide

Abstract

Neural architectures tend to fit their data with relatively simple functions. This "simplicity bias" is widely regarded as key to their success. This paper explores the limits of this principle. Building on recent findings that the simplicity bias stems from ReLU activations [96], we introduce a method to meta-learn new activation functions and inductive biases better suited to specific tasks.

Findings. *We identify multiple tasks where the simplicity bias is inadequate and ReLUs suboptimal. In these cases, we learn new activation functions that perform better by inducing a prior of higher complexity. Interestingly, these cases correspond to domains where neural networks have historically struggled: tabular data, regression tasks, cases of shortcut learning, and algorithmic grokking tasks. In comparison, the simplicity bias induced by ReLUs proves adequate on image tasks where the best learned activations are nearly identical to ReLUs and GeLUs.*

Implications. *Contrary to popular belief, the simplicity bias of ReLU networks is not universally useful. It is near-optimal for image classification, but other inductive biases are sometimes preferable. We showed that activation functions can control these inductive biases, but future tailored architectures might provide further benefits. Advances are still needed to characterize a model's inductive biases beyond "complexity", and their adequacy with the data.*

1. Introduction

When and why NNs generalize is yet to be understood. Neural networks (NNs) have proven more effective than other machine learning models. However, we still miss a complete explanation of their generalization abilities. A better understanding could help address failures from shortcut learning [29, 93] to distribution shifts [47, 95], biases, and spurious correlations in language models for example [28, 38, 87]. Understanding conditions for generalization would also enable the design of architectures and data preparation from first principles, rather than trial and error.

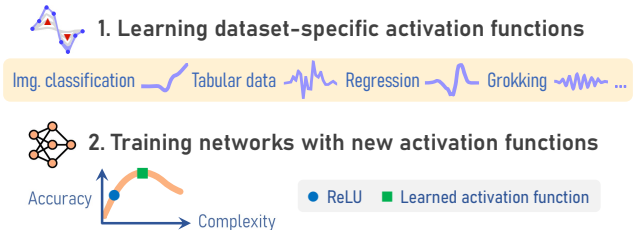

Figure 1. (1) We modulate the inductive bias of neural architectures by learning novel activation functions that improve generalization on specific datasets. (2) With this tool, we study the relation between model accuracy and complexity. We identify tasks where the simplicity bias of ReLU architectures is suboptimal.

This paper studies inductive biases i.e. the assumptions made by learning algorithms to generalize beyond training data [67].[1] A vast literature examines the inductive biases of architectures [14], optimizers [69], losses [42], regularizers [50], etc. The **simplicity bias** is one aspect of the inductive biases of NNs that makes them fit their training data with simple[2] functions [8, 71]. Despite wide belief that the simplicity bias could be due to SGD [7, 40, 60, 92], work on untrained networks showed that it can be explained with architectures alone [15, 31, 64, 99]. ReLUs also seem critical to induce the simplicity bias in typical architectures [96].

Limits of the simplicity bias. The simplicity bias is an intuitive explanation for the ability to generalize on real-world data. It embodies Occam's razor [66] and assumes that data-generating processes in the real world are simple. Additionally, a prior for simplicity is supported by results in algorithmic information theory [18] stating essentially that "*a bias in the distribution of target functions must be towards low complexity*". However, this only means that simplicity is a good prior on average, but not necessarily the best choice on any task or dataset. This matches the no-free lunch theorem [101] according to which no inductive bias is universally useful. Therefore, this paper asks the following.

[1] Inductive biases can formalized as a prior over the space of functions [65].
[2] Simplicity can be formalized using Kolmogorov complexity or its approximations, e.g. frequency, compressibility, sensitivity, etc. [17, 18, 37, 96]

Are there practical applications of machine learning where the simplicity bias is detrimental? In these cases, what do the optimal inductive biases look like?

For example, shortcut learning is one situation where the simplicity bias is already known to be detrimental [85, 93].

Searching for optimal inductive biases by learning activation functions. Prior work [96] showed that ReLU activations are critical to obtain the simplicity bias in typical architectures. Hence we build a new tool to modulate the simplicity bias by learning dataset-specific activation functions. It uses bi-level optimization and a spline parametrization to learn activations free of any prior, such as constraints of smoothness or monotonicity (unlike prior work [2, 6, 82]). This (1) enables the discovery of entirely new activation functions and inductive biases that improve generalization (Figure 1) and (2) highlights the suboptimality of the simplicity bias by comparing the accuracy and complexity of models with ReLUs vs. learned activations.

Findings. We examine four domains that we hypothesized to be impaired by the simplicity bias: tabular data, regression tasks, cases of shortcut learning, and algorithmic tasks. Our intuition is that they require learning functions with high sensitivity or sharp transitions. For each domain, we collect existing datasets then train and analyze models without and with learned activation functions. In all cases, we obtain better generalization with dataset-specific activations, and the improvements are attributable to learning higher-complexity solutions. In comparison, this analysis also shows that classical image datasets (MNIST, CIFAR, FASHION-MNIST, SVHN) are extremely well suited to the inductive biases of ReLUs. The best learned activations are then strikingly similar to variants like GeLUs [39].

Summary of contributions.

- A new method to discover dataset-specific activation functions optimized for generalization.
- An examination of >20 datasets showing that the simplicity bias of ReLU architectures can be suboptimal. (1) For **regression** tasks and **tabular** data, new learned activations greatly improve accuracy by helping learn complex functions. (2) For **image classification**, the process rediscovers smooth variants of ReLUs, suggesting a near-optimal choice for these popular tasks. (3) In cases of **shortcut learning**, we show that different learned activations can steer the learning towards different image features. (4) For **grokking** tasks, new learned activations can eliminate the phenomenon, supporting the explanation as a mismatch between data and architectures. We also measure a positive **transferability** of learned activations across related tasks.
- An analysis showing that improvements with learned activations correlate with the learning of complex functions.

Implications. All cases where the simplicity bias proved suboptimal are in domains where NNs have historically struggled. We now connect them to a common explanation. This implies that architectures tailored to some specific domains may still have a place besides scaling up models and data. Conversely, the suitability of ReLUs to image classification suggests that researchers successfully converged by trial and error to designs well tuned to popular tasks.

2. Methods

This section introduces tools to analyze trained models and to learn new dataset-specific activation functions.

2.1. Visualizing a Model's Function

A neural network implements a function $f_{\boldsymbol{\theta}} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ of parameters $\boldsymbol{\theta}$ (weights and biases) that maps an input $\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}$ to an output $\boldsymbol{y} \in \mathbb{R}^{d_{\text{out}}}$. For a regression task, $\boldsymbol{y} \in \mathbb{R}$ is the predicted value. For a classification task, $\boldsymbol{y}$ is a vector of logits passed through a sigmoid or softmax to obtain class probabilities. Because d_{in} can be large, f can be difficult to visualize and analyze. A workaround is to examine f over 1D or 2D slices of the input space [26, 96]. To obtain a slice in a region of plausible data, we use the training data $\mathcal{T}$. For a 1D slice (linear path), we sample $\boldsymbol{x}_1, \boldsymbol{x}_2 \sim \mathcal{T}$ then define the path $\mathbf{X}_{\boldsymbol{x}_1,\boldsymbol{x}_2} = [\,(1-\lambda)\,\boldsymbol{x}_1 + \lambda\,\boldsymbol{x}_2,\ \lambda \in [0,1]\,]$. We proceed analogously with three points for a 2D slice. We sample λ regularly in [0,1] such that $\mathbf{X}$ is a finite sequence of points. Then f is evaluated on these points to give a 1D sequence or 2D grid of values that are convenient to display and analyze (Figure 7c). When $d_{\text{out}} > 1$ (multi-class task), we examine one random dimension of f's output at a time.

2.2. Measuring a Model's Complexity

We wish to quantify the complexity of the function f implemented by a model trained on data $\mathcal{T}$. Prior work used Fourier decompositions [26, 96] but this requires a delicate implementation. We found a reliable alternative with the total variation (TV) of f averaged over many 1D paths:

$$\text{TV}(f, \mathcal{T}) \;=\; \mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{x}_2 \sim \mathcal{T}} \int_{\boldsymbol{x}_1}^{\boldsymbol{x}_2} \left|f'(\boldsymbol{x})\right| \, d\boldsymbol{x} \,. \tag{1}$$

with f' the first derivative. We estimate (1) using a path as defined in Section 2.1. We name the points in such a path $\mathbf{X}_{\boldsymbol{x}_a,\boldsymbol{x}_z} := [\boldsymbol{x}_a, \boldsymbol{x}_b, \boldsymbol{x}_c, \ldots \boldsymbol{x}_y, \boldsymbol{x}_z]$. We then have:

$$\begin{aligned} \text{TV}(f, \mathcal{T}) \;\approx\; \mathbb{E}_{\boldsymbol{x}_a,\boldsymbol{x}_z \sim \mathcal{T}} \;& \left|f(\boldsymbol{x}_b) - f(\boldsymbol{x}_a)\right| \\ &+ \left|f(\boldsymbol{x}_c) - f(\boldsymbol{x}_b)\right| + \ldots \\ &+ \left|f(\boldsymbol{x}_z) - f(\boldsymbol{x}_y)\right| . \end{aligned} \tag{2}$$

Appendix F shows that (2) correlates closely with a Fourier-based measure of complexity: the higher the TV, the higher the complexity. Yet, it is straightforward to implement and discriminative across small and large values.

2.3. Meta-Learning Activation Functions

Our goal is to optimize the inductive biases of a neural network and recent work [96] showed that the activation functions are the most important component. The typical approach to learn activations [2, 5, 6, 11, 13, 22, 45, 82, 91] (see Related Work) replaces them with a small shared ReLU MLP that implements an $\mathbb{R} \to \mathbb{R}$ function. Its parameters are optimized along the network's. However this cannot discover truly novel activations because the embedded ReLU MLP has itself a simplicity bias and activations are optimized together with the model. We propose instead:

- an **unbiased** parametrization of the activations as splines,
- a bi-level optimization to learn **reusable** activations,
- an episodic training to **optimize for generalization** rather than simply to fit the training data.

Parametrization as splines. We want a space of activation functions free of priors such as the smoothness and monotonicity enforced in prior work [5, 13]. We implement an activation $g_{\boldsymbol{\psi}} : \mathbb{R} \to \mathbb{R}$ as a linear spline with control points defined by $\boldsymbol{\psi}$. We define n_c points spread regularly in an interval $[a, b]$, typically ~ 50 points in $[-5, +5]$. Then g represents piecewise linear segments interpolating values specified in the learned parameters $\boldsymbol{\psi} := [\, g_{\boldsymbol{\psi}}(a), \ldots g_{\boldsymbol{\psi}}(b)) \,] \in \mathbb{R}^{n_c}$. g can represent simple and complex functions, including smooth curves, periodic functions, sharp transitions, etc.

Bi-level optimization & episodic training. Our goal is to get an activation function that can be reused like any other in subsequent training runs. This differs from prior work (e.g. [2]) that continuously updates the activation during training: the final one may not be suitable to start training with. Our solution is a bi-level meta-learning loop. An inner loop trains the model with a fixed activation function. An outer loop trains the activation function to maximize generalization. Each outer step simulates a new learning task or *episode*. This means (1) initializing the model with different weights and (2) using different subsets of data for training and validation. With suitable choices, this can simulate in- or out-of-distribution conditions (see Section 3.4). Without episodes, the learned activation could overfit to a particular model initialization for example, and would not generalize in subsequent training runs. The method is outlined as Algorithm 1. Its implementation is discussed in Appendix C.

> ! *Inductive bias* and *simplicity bias* are not interchangeable. Our method optimizes toward better generalization. Simplicity is only one aspect of the trained models that we analyze post-hoc (e.g. Figure 4).

Algorithm 1 Meta-learning an activation function (AF).

Input: training data $\mathcal{T}$; untrained neural model $f_{\boldsymbol{\theta},\boldsymbol{\psi}}$
Initialize $\boldsymbol{\psi}$ with zeros *Parametrization of AF*
$n_{\text{tr}} \leftarrow 0$ *Number of inner-loop iterations*
while $n_{\text{tr}} < n_{\text{tr}}^{\text{max}}$ ***Outer loop: train AF***
  Increment n_{tr}
  Sample the episode's tr. ($\mathcal{T}'$) and val. ($\mathcal{V}$) sets from $\mathcal{T}$
  Initialize $\boldsymbol{\theta}$ randomly *Model weights and biases*
  for n_{tr} **steps** ***Inner loop: train model with fixed AF***
    Eval. loss on $\mathcal{T}'$: $L \leftarrow \Sigma_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{T}'} \mathcal{L}\big(f_{\boldsymbol{\theta},\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{y})\big)$
    Gradient step on weights/biases: $\boldsymbol{\theta} \leftarrow \text{GD}(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} L)$
  Eval. loss on $\mathcal{V}$: $L \leftarrow \Sigma_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{V}} \mathcal{L}\big(f_{\boldsymbol{\theta},\boldsymbol{\psi}}(\boldsymbol{x}, \boldsymbol{y})\big)$
  Gradient step on AF: $\boldsymbol{\psi} \leftarrow \text{GD}(\boldsymbol{\psi}, \nabla_{\boldsymbol{\psi}} L)$
  if performance on $\mathcal{V}$ worsens **then** break *Early stopping*
Output: optimized AF $\boldsymbol{\psi}$

3. Tasks and Results

We now examine tasks that we hypothesized to be ill-suited to the simplicity bias of ReLU architectures. The intuition is that the target function to learn (e.g. optimal classifier) contains sharp transitions (regression tasks, tabular datasets), or repeating patterns (algorithmic tasks) that contradict the ReLUs' simplicity bias. For each task, we examine existing datasets with the tools from Section 2. In all cases, we find benefits from architectures whose inductive biases favor more complex functions. Additional details and results are provided in Appendix E.

3.1. Image Classification Tasks

Background. We start with classical datasets to validate our methodology: MNIST, FASHION-MNIST, SVHN, CIFAR-10 [49, 52, 68, 102]. They are representative of the vision tasks that guided the development of deep learning. **Our hypothesis** is therefore that the inductive biases of modern architectures and ReLUs are well suited to these datasets.

Setup. For each dataset, we learn activation functions with Algorithm 1. We experiment with two initializations of the spline parameters: as zeros and so as to mimic a ReLU. The goal of the latter is to explore the space of functions similar to ReLUs. Because of the difficulty of the optimization, the algorithm is likely to converge to a *local* optimum similar to ReLUs if there is one. We also experiment with the sharing of the activation function. By default, a single function is shared across the network. Alternatively, we learn a different activation function per layer. This provides more ways to affect the model's inductive biases. Our base architecture is a 3-layer MLP (details in Appendix E).

Results. We compare in Figure 2a the accuracy of models with ReLUs vs. learned activation functions. Differences are small. The learned activations only improve slightly on SVHN and CIFAR. This suggests that the inductive biases of ReLUs are generally well suited to these datasets.

We examine the learned activations in Figure 3a. With

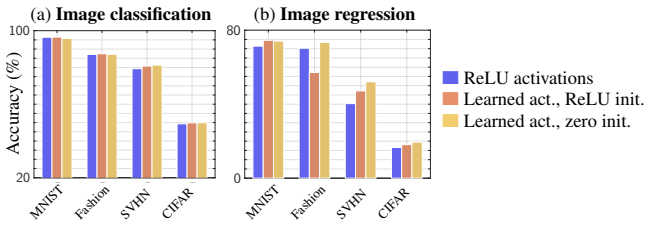

Figure 2. Test accuracy on image datasets. (a) For **classification** tasks, all models perform similarly, suggesting that the inductive biases of ReLUs are well suited to these datasets. (b) For **regression** tasks, models with learned activations perform better, especially from an initialization as zeros, which enables the discovery of completely novel activation functions.

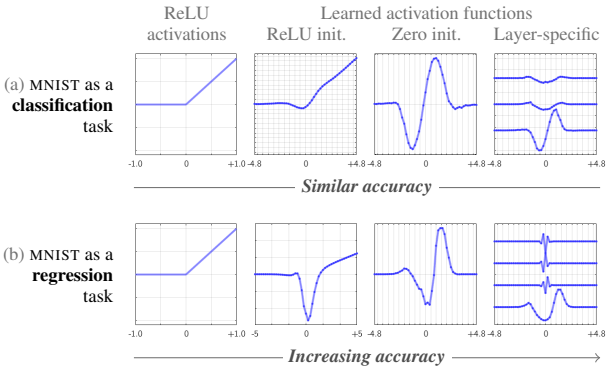

Figure 3. Activation functions learned for MNIST. For a **classification** task, the activation learned from a ReLU resembles the popular GeLUs. For a **regression** task, the learned activations contain irregularities that help a network represent complex functions. See Figure 16 for similar results on other datasets.

an initialization as ReLUs, the optimization converges to a smooth variant remarkably similar to GeLUs [11] which are widely used. This suggests that the research community has empirically converged on a local optimum in the space of activation functions. With an initialization as zeros, we discover wavelets [80] that are unlike common activations but perform as well as ReLUs, i.e. another local optimum.

Take-away: for image classification, learned activations provide very little benefit over ReLUs. Smooth variants of ReLUs are a local optimum in the space of activations. ReLUs' popularity for such tasks could thus be explained with their proximity to this optimum.

3.2. Regression Tasks

Background. Regression tasks are known to be difficult for NNs [90]. They are often turned into a classification through discretization [24, 43]. Existing explanations that invoke implicit biases of gradient descent are clearly incomplete [90]. **Our hypothesis** is that regression is difficult because it often involves irregular decision boundaries [35] in opposition to the typical solutions of ReLU networks [17].

Setup. We use the same setup and image datasets as Section 3.1. The task is now to directly predict class IDs. E.g. for MNIST this means predicting digit values. Models are trained with an MSE loss. To measure accuracy, we discretize the predictions to the nearest class ID.

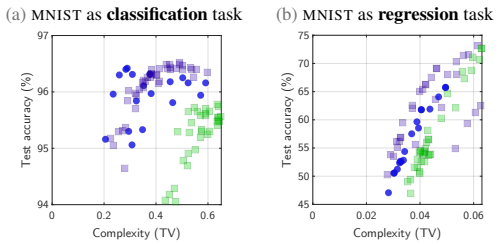

Figure 4. Accuracy vs. complexity on image datasets. Each marker is a model with different hyperparameters and ReLUs (●) or learned activations initialized as ReLUs (■) or as zeros (■). For **classification** (a), ReLUs are close to best. Activations optimized from ReLUs only improve the accuracy slightly, corresponding to the GeLU-like function in Figure 3. For **regression** (b), new activations (learned from zeros) are best. Moreover, accuracy and complexity are clearly correlated only for regression. This supports the hypothesis that regression is more complex than classification and thus benefits from alternatives to the ReLUs' simplicity bias. See Figure 18 for similar results on other datasets.

Results. The first observation from Figure 2b is that regression is clearly more difficult for NNs than classification (lower accuracies) despite the identical underlying task. Importantly, the learned activations now provide clear improvements, especially when learned from scratch (initialization as zeros). This confirms the hypothesis that the inductive biases of ReLUs are not well suited to these tasks.

Figure 3b shows that the learned activations contain more irregularities for regression than classification. Prior work [96] showed that this can help models represent complex functions with sharp transitions. An analysis of the complexity of trained models (Figures 4 and 18) shows that the accuracy is correlated with complexity for regression but not classification. And regression models with learned activations implement functions of higher complexity than with ReLUs. This supports the claim that the improvements arise from overcoming the simplicity bias of ReLUs.

Complexity is only one dimension of the inductive biases. The complexity plots for SVHN (Figure 18) interestingly show that models with ReLUs and learned activations get different accuracies at the same complexity level. This shows that our meta learning approach can search over dimensions of the inductive biases that are not captured by our complexity measure, and are yet to be explicitly studied.

Take-away: regression is more difficult for NNs than classification, and the simplicity bias of ReLUs is partly to blame. Learned activations improve performance by helping networks represent more complex functions.

3.3. Tabular Data

Background. Tabular data is any data with few unstructured dimensions, which often contains low-cardinality variables such as dates or categorical attributes. This contrasts e.g. with images, which contain many correlated, continuous dimensions (pixels). NNs struggle with tabular datasets and are often inferior to decision trees [35, 63]. **Our hypothesis** is that the inductive biases of standard architectures are ill-suited to such data because of the simplicity bias. It makes it difficult to learn functions where small changes in the input (e.g. day of the week) correspond to abrupt changes in the target — the definition of *sensitivity*, a proxy for complexity [17]. This seldom occurs in vision where similar images correspond to similar labels.

Setup. We use 16 real-world classification datasets from Grinsztajn et al. [34, 35]. Baselines include a linear classifier, k-NNs, and boosted decision trees. Our models are MLPs with 1–4 hidden layers (details in Appendix E.4). We compare learned activations functions with ReLUs and TanHs with a global prefactor, $\tanh(\alpha x)$ with $\alpha \in \mathrm{R}^+$ tuned on the validation set. This is a simple option with tunable complexity, albeit with inductive biases of TanHs [45, 96].

We also experiment with learned *input activation functions* (IAFs). The motivation is to learn a different behavior for each input dimension. Since they carry different information, e.g. continuous vs. categorical variables, one could be suited to the simplicity bias while another is not, for example. IAFs are dimension-specific activation functions applied directly on the data before a standard MLP. IAFs are learned like AFs, from an initialization as the identity i.e. no effect by default. They subsume the gated inputs, and Fourier/numerical embeddings from prior work [20, 25, 32].

Results. We compare the accuracy of models on the 16 datasets in Figures 5 and 20. Vanilla MLPs generally perform worse than trees. But adjusting the MLPs' inductive biases with learned prefactors or activations eliminates the gap. IAFs perform best, sometimes even surpassing trees. We analyze below the reasons for these improvements.

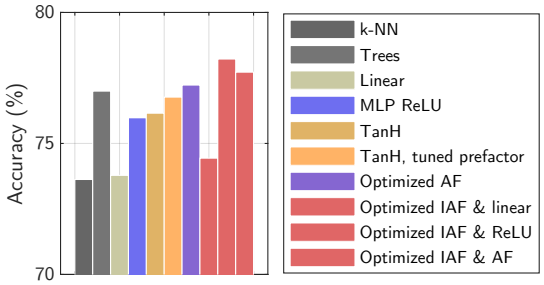

Figure 5. Comparison of model types over 16 tabular datasets. Vanilla MLPs often perform worse than decision trees, but adjusting their inductive biases with learned activation functions (AFs) eliminates this gap. The *input activation functions* (IAFs) enable even better performance. See Figure 20 for results per dataset.

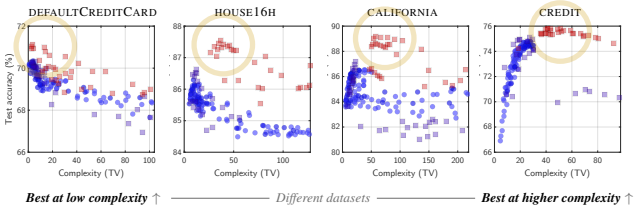

Figure 6. Test accuracy vs. complexity on tabular datasets. Each marker represents a model with different hyperparameters, and ReLUs (●) or learned activations initialized as ReLUs (■) or as zeros (■). The learned activations perform better in all cases, but the accuracy peaks at different complexity levels. For some datasets, a low complexity is best and ReLUs thus perform quite well (leftmost panel, note the smaller Y scale). For other datasets, the opposite is true and the improvements with learned activations is larger.

Learned activation functions close the gap to decision trees by mimicking their inductive bias. We visualize in Figure 7c the functions implemented by different models, plotting their output over slices of the input space (Section 2.1). ReLUs produce the smoothest function while TanHs and learned activations induce sharper patterns. Notably, the IAFs induce sharp axis-aligned decision boundaries that are also characteristic of trees, with which they share a high accuracy. Axis-aligned transitions are the consequence of IAFs applied *independently* to each dimension. Sharp transitions originate from the complex shape of the learned activation function (Figure 7a) which is possible thanks to the unbiased spline parametrization (Section 2.3).

The simplicity/complexity bias is a property of the architecture. We visualize *complexity landscapes* of MLPs in Figure 7b. Similarly to standard *loss* landscapes [54], we plot model complexity over 2D slices of the parameter space. A first global view over a plane aligned with the training trajectory shows that complexity steadily increases during training for all models [48, 75, 104] but does so to the highest level for the best model (IAFs). A second view zooms in on each optimized solution in a random 2D plane. This examines the effect of *arbitrary* perturbations to the parameters.[3] It shows that the ambient complexity of perturbed solutions of the best model is much higher than the solution itself, and than with less accurate models. This means that this architecture is more likely to represent complex functions because they are more abundant in parameter space [65, 84]. **This is why the simplicity bias can be overcome**: it results from architecture choices and not from an inevitable "implicit bias" of SGD [85, 89, 98, 105].

Different tabular datasets require different inductive biases. We examine the relation between accuracy and complexity in Figure 6. The accuracy peaks at different complexity levels for different datasets. For some, a low complexity is best and ReLU MLPs perform well. For oth-

[3] This resembles an analysis of untrained models [15, 64, 96, 99] but focuses on relevant regions on the parameter space, near optimized models.

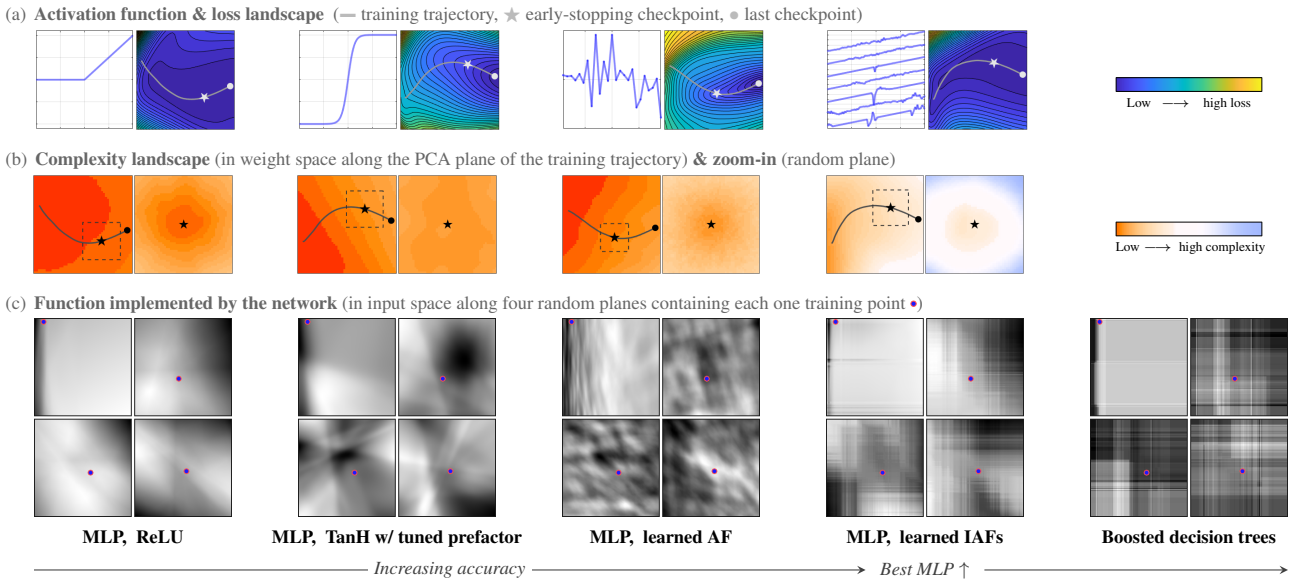

Figure 7. Models trained on the ELECTRICITY [34] tabular dataset. ReLU MLPs perform worst (left). TanHs induce sharper transitions in the **network's function (c)**. So does the learned **activation function (a)** which is itself very irregular. The input activation functions (IAFs) perform best and mimic the axis-aligned boundaries of trees (bottom-right). The **complexity landscapes (b)** show that complexity increases to the highest level in the best model (IAFs). The zoom-in shows that the ambient complexity is also much higher than in other models. This means that it is inherently more likely to represent complex functions since they are more abundant in parameter space.

ers, a higher complexity is best and the improvements with learned activations are larger. This supports the hypothesis that improvements over ReLU MLPs come from overcoming their simplicity bias. The variance across datasets is also unsurprising since they have little in common besides their low dimensionality (full results in Appendix E.4).

Effect of width and depth. We show in Figure 8 that the learned activations can be reused in networks of different widths than they were trained for. The accuracy varies with width similarly as with ReLUs. Teney et al. [96] indeed showed that a model's width affects its capacity but not its inductive biases. Therefore width does not interfere with the effects of the learned activations. Figures 8 and 21 also show that good performance can be achieved with fewer layers than with ReLUs. Learned activations might thus have utility in model compression and distillation.

> **Take-away:** many tabular datasets are ill-suited to ReLU models because they require learning a complex function. Learned activations improve accuracy by implementing sharp axis-aligned decision boundaries that mimic the inductive biases of decision trees.

3.4. Shortcut Learning

Background. Shortcut learning occurs when a model learns spurious features instead of generalizable ones. It is a known consequence of the simplicity bias [85, 93] when the training data contains multiple features of different complexity. **Our hypothesis** is that the preference for some features depends on their alignment with the inductive biases. We will evaluate whether this can be controlled with activation functions.

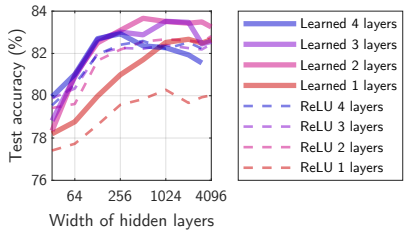

Figure 8. The learned activation functions surpass ReLUs, often with fewer layers. They can also be reused with different network widths (COVERTYPE [34] tabular dataset, see Figure 21 for others).

Setup. We use MNIST/CIFAR collages [85, 93, 94], a classification task over images combining tiles from MNIST and CIFAR-10. The **training set** is ambiguous: both tiles are predictive of the labels. Two unambiguous **test sets** evaluate reliance on either tile: one is predictive, the other contains a random class. We similarly build two **validation sets** to learn **two activation functions** optimized for either tile. We simulate OOD conditions by setting $\mathcal{V}$ in Algorithm 1. The models are the fully-connected MLPs used in [93].

Results. Figure 9 shows that a baseline with ReLUs is prone to shortcut learning. It relies exclusively on MNIST and the accuracy on the CIFAR test set is not better than chance (10%). In comparison, using either learned activation steers the learning towards either tile. The accuracy

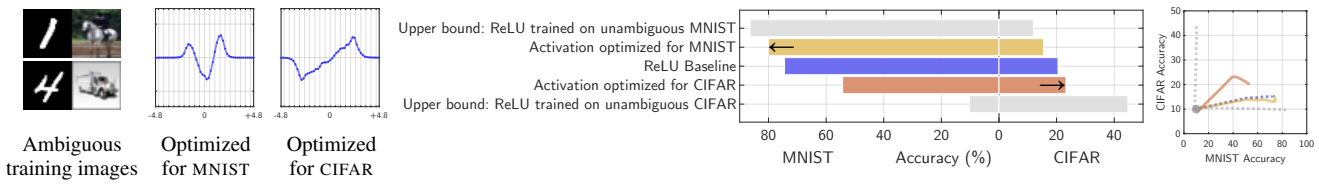

Figure 9. Experiments on shortcut learning with MNIST/CIFAR collages. The ReLU baseline (■) relies mostly on simple MNIST features. We learn two activation functions that shift the preference towards different features (←/→). Training trajectories (right) clearly differ with the activation optimized for CIFAR (—), for MNIST (—), or a ReLU (- -). The model at initialization (random weights) is marked with ● .

shifts towards either of two tiles as the model prioritizes different features, merely with a change of activation function. This shows that the simplicity bias is not an inevitable effect of SGD. Instead, it directly reflects the alignment between the chosen architecture and the data.

Training dynamics. The accuracy on CIFAR remains below a model trained on unambiguous CIFAR data. This is because training dynamics are also important. In Figure 9 (right), we plot the accuracy on the two tiles for the whole training trajectory. The reliance on different features varies, and the model eventually relies primarily on simple ones with enough iterations (i.e. without early stopping). This calls for future work combining our findings with the extensive literature on ID / OOD training dynamics [46, 95, 98].

> **Take-away:** we confirm that shortcut learning is a side effect of the simplicity bias. Different activation functions, while not completely avoiding shortcut learning, can steer the learning towards particular input features.

3.5. Algorithmic Tasks and Grokking

Background. Grokking is a phenomenon where a model first overfits the data (i.e. high training accuracy, low test accuracy) then shifts to high test accuracy after many training steps [73]. This is typically observed on algorithmic tasks and architectures from MLPs to transformers. **Our hypothesis** is that grokking is due to a mismatch between the target function and the model's inductive biases. Indeed, typical architectures were not developed for the algorithmic tasks where grokking is typically observed. To verify this hypothesis, we will show that endowing an architecture with the right inductive biases, using learned activation functions, can eliminate the phenomenon. Supporting this hypothesis, Zhou et al. [110] proposed that grokking comes from the frequency principle (i.e. low frequencies learned first by SGD), and Kumar et al. [51] showed that it correlates with a misalignment between features at initialization and the target function.

Setup. Following [36, 51, 58] we train 1-hidden layer MLPs on algorithmic tasks, defined each by one binary operation (Figures 10 and 27). e.g. $y = (x_1 + x_2) \bmod 13$. The operands are passed as one-hot vectors and the task is a classification over possible outputs. Details in Appendix E.6.

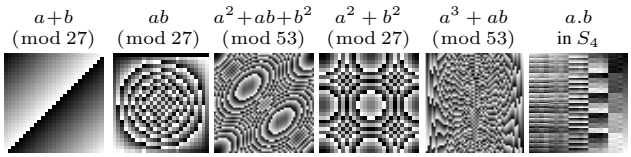

Figure 10. Target functions used to investigate grokking [73] (details in Appendix, Figure 27). These patterns are very different from the tasks for which typical architectures were developed.

Results. We compare models with ReLU vs. learned activations across various tasks, network widths, and fractions of training data. We find that the learned, task-specific activations lead to faster convergence and/or higher test accuracy (Figures 11–13). On modular addition (a common task in the grokking literature) the learned-activation model converges $\sim 10\times$ faster than ReLUs. Curiously, some models with learned activations also end up overfitting (decreasing test accuracy) with prolonged training. In contrast, ReLU networks either never generalize (test accuracy ~ 0) or *grok* and keep a high accuracy indefinitely. Further investigation is needed to explain this difference. We examine learned activation functions in Figure 12. See Figure 28 in the Appendix for results on other algorithmic tasks [73].

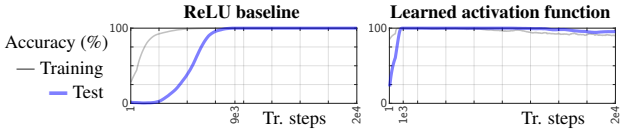

Figure 11. The learned activations essentially eliminate grokking (delayed convergence). On the above task (addition mod 27), our model converges $\sim 10\times$ faster than ReLUs.

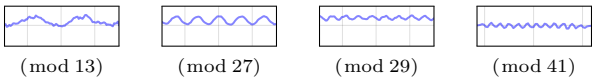

Figure 12. Activations learned for modular addition. The frequency of the sine-like function varies across versions of the task.

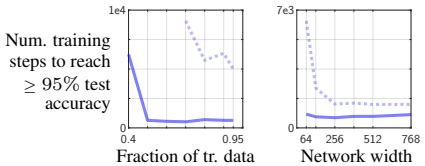

Figure 13. Models with learned activations (—) converge faster than ReLUs (- -) across a variety of settings (addition mod 27).

Take-away: learned activations eliminate grokking in all our cases, suggesting, as a cause, the mismatch between the data and the architectures' inductive biases.

4. Do the Activations Transfer Across Tasks?

So far, we used dataset-specific activation functions and found that there exist better alternatives to ReLUs. A practical application would be the learning of activation functions suitable to a broad task, or range of related datasets.

As a first step, we study the specialization of the activations functions (AFs) learned for the 22 algorithmic tasks from Figure 27 [73]. We evaluate every task/activation combination, yielding the 22×22 matrix of Figure 14. The learned activations do transfer, with improvements in accuracy and convergence shared across tasks. We also evaluate an activation learned learned on *all tasks* simultaneously. The accuracy across tasks (i.e. per-column average) reaches 61.5% vs. only 19.9% for ReLUs. and 54.0% on average for tasks-specific solutions. This procedure can thus improve performance on a *range* of related tasks. Future work could leverage it to discover activation functions that improve performance in other specific domains. See Appendix E.3 for other transfer experiments using image regression tasks.

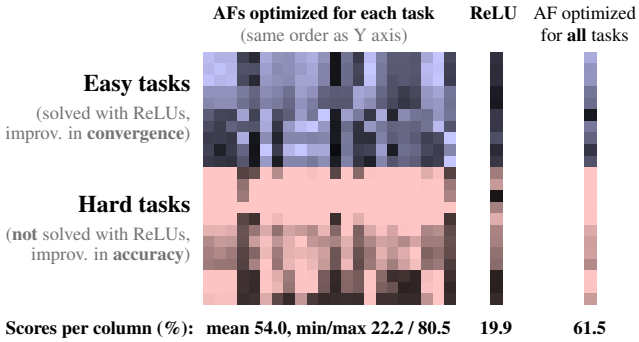

Figure 14. Transfer of AFs (columns) across algorithmic tasks (rows). Colors represent the fraction of the best convergence speed or accuracy per task (brighter is better). If the activations were over-specialized, the matrix would be diagonal. On the contrary, it is densely filled, indicating positive transfer across many tasks.

5. Discussion

We used activation functions as a tool to show that there exist a variety of inductive biases that are useful across applications of NNs. The impossibility of *universal* inductive biases is well known [101] but a strong argument has also been made that deep learning research is converging towards few architectures with wide applicability [31]. This argument rests on the NNs' simplicity bias being a good match for real-world data [12, 18, 56]. Our results do not invalidate these assumptions: NNs are widely applicable and their simplicity bias is evidently very effective *on average*. Our results show instead the following.

1. There exist **real-world tasks where the inductive biases of typical architectures' are suboptimal**. This explanation connects four domains where NNs historically struggled.
2. The **simplicity bias in modern NNs depends on particular design choices**, the activation functions in particular. Research has converged on these choices by trial and error, in large part by optimizing performance on vision tasks. Therefore the adequacy of ReLUs for image classification (Section 3.1) is not accidental.

Relevance to transformers and language models. The simplicity bias exists in transformers [10, 109] and language models [4, 31, 96]. Their embedding layer resembles *input activations functions* (Section 3.3). Could this explain the transformers' remarkable flexibility? I.e. a simplicity bias on an initial mapping of arbitrary complexity. Zhong and Andreas [108] indeed trained embeddings alone in a random-weight transformer and could learn complex tasks.

Limitations and open questions. We prioritized breadth by establishing a new connection across multiple disparate topics in machine learning. Each section could expand into its own paper with additional models, datasets, comparisons, etc. Our findings on shortcut learning for example (Section 3.4) could yield new methods to address distribution shifts, though no such claim is made here. Here are the most promising follow-up questions opened by this paper.

- **How to fully characterize inductive biases?** We focused on simplicity for its prevalence in AI [31], philosophy [72], and the natural sciences [12]. But it is only one dimension among many to characterize inductive biases.
- **Can we improve state-of-the-art architectures?** We used simple MLPs to isolate the effects of activations functions since they are central to the simplicity bias [96]. But other existing mechanisms (architectural, optimization) may already tweak or attenuate the simplicity bias.
- **Can we learn transferable activation functions for other domains?** We examined transferability in Sections 4 and E.3. The results suggest the possibility of better architectures optimized for specific domains. Predicting the suitability of an architecture/dataset pair ex ante (prior to training) would be extremely useful. This may follow from advances on the first open question above.
- **Are there other detrimental effects of the simplicity bias?** Any learning algorithm needs inductive biases to "fill the gaps" between training examples. The better they are, the fewer examples are needed. Researching what inductive biases are most useful on real-world tasks might thus hold the key for machine learning to become as data-efficient as humans. More speculatively, high-level cognition has been argued to require postulating explanations beyond the data [16, 23]. In this regard, simplicity-biased architectures might also hold us back.

References

[1] Sravanti Addepalli, Anshul Nasery, Venkatesh Babu Radhakrishnan, Praneeth Netrapalli, and Prateek Jain. Feature reconstruction from outputs can mitigate simplicity bias in neural networks. In *ICLR*, 2022. 1

[2] Konstantinos Panagiotis Alexandridis, Jiankang Deng, Anh Nguyen, and Shan Luo. Adaptive parametric activation. *arXiv preprint arXiv:2407.08567*, 2024. 2, 3, 1

[3] Konstantinos Panagiotis Alexandridis, Jiankang Deng, Anh Nguyen, and Shan Luo. Adaptive parametric activation. In *ECCV*, 2025. 2

[4] Badr AlKhamissi, Greta Tuckute, Antoine Bosselut, and Martin Schrimpf. Brain-like language processing via a shallow untrained multihead attention network. *arXiv preprint arXiv:2406.15109*, 2024. 8

[5] Andrea Apicella, Francesco Isgro, and Roberto Prevete. A simple and efficient architecture for trainable activation functions. *Neurocomputing*, 2019. 3, 1, 2

[6] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 2021. 2, 3, 1

[7] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *NeurIPS*, 2019. 1

[8] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *ICML*. PMLR, 2017. 1

[9] Samuel James Bell and Levent Sagun. Simplicity bias leads to amplified performance disparities. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 355–369, 2023. 1

[10] Satwik Bhattamishra, Arkil Patel, Varun Kanade, and Phil Blunsom. Simplicity bias in transformers and their ability to learn sparse boolean functions. *arXiv preprint arXiv:2211.12316*, 2022. 8

[11] Garrett Bingham, William Macke, and Risto Miikkulainen. Evolutionary optimization of deep learning activation functions. In *Genetic and Evolutionary Computation Conference*, 2020. 3, 4, 1, 2

[12] Mark Buchanan. A natural bias for simplicity. *Nature Physics*, 2018. 8

[13] Irit Chelly, Shahaf E Finder, Shira Ifergane, and Oren Freifeld. Trainable highly-expressive activation functions. *arXiv preprint arXiv:2407.07564*, 2024. 3, 1, 2

[14] Nadav Cohen and Amnon Shashua. Inductive bias of deep convolutional networks through pooling geometry. *arXiv preprint arXiv:1605.06743*, 2016. 1

[15] Giacomo De Palma, Bobak Kiani, and Seth Lloyd. Random deep neural networks are biased towards simple functions. *NeurIPS*, 2019. 1, 5

[16] David Deutsch. *The beginning of infinity: Explanations that transform the world*. Penguin, 2011. 8

[17] Benoit Dherin, Michael Munn, Mihaela Rosca, and David Barrett. Why neural networks find simple solutions: The many regularizers of geometric complexity. *NeurIPS*, 35, 2022. 1, 4, 5

[18] Kamaludin Dingle, Chico Q Camargo, and Ard A Louis. Input–output maps are strongly biased towards simple outputs. *Nature communications*, 2018. 1, 8

[19] Pedro Domingos. The role of occam's razor in knowledge discovery. *Data mining and knowledge discovery*, 1999. 1

[20] Marius Dragoi, Florin Gogianu, and Elena Burceanu. Closing the gap on tabular data with fourier and implicit categorical features. *Submission to ICLR (available on OpenReview)*, 2024. 5, 1

[21] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 2022. 1

[22] Stanislas Ducotterd, Alexis Goujon, Pakshal Bohra, Dimitris Perdios, Sebastian Neumayer, and Michael Unser. Improving lipschitz-constrained neural networks by learning activation functions. *Journal of Machine Learning Research*, 2024. 3, 1, 2

[23] Daniel C Elton. Applying deutsch's concept of good explanations to artificial intelligence and neuroscience–an initial exploration. *Cognitive Systems Research*, 2021. 8

[24] Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024. 4

[25] James Fiedler. Simple modifications to improve tabular neural networks. *CoRR*, abs/2108.03214, 2021. 5

[26] Sara Fridovich-Keil, Raphael Gontijo Lopes, and Rebecca Roelofs. Spectral bias in practice: The role of function frequency in generalization. *NeurIPS*, 2022. 2

[27] Gallant. There exists a neural network that does not make avoidable mistakes. In *IEEE International Conference on Neural Networks*. IEEE, 1988. 1

[28] Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. Bias and fairness in large language models: A survey. *Computational Linguistics*, 2024. 1

[29] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2020. 1

[30] Florin Gogianu, Tudor Berariu, Mihaela C Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: an optimisation perspective. In *ICML*, 2021. 11

[31] Micah Goldblum, Marc Finzi, Keefer Rowan, and Andrew Gordon Wilson. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. *arXiv preprint arXiv:2304.05366*, 2023. 1, 8

[32] Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. *NeurIPS*, 2022. 5, 1, 9

[33] Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *Proceedings of the Royal Society A*, 2022. 1

[34] Léo Grinsztajn. Tabular data learning benchmark. *https://github.com/LeoGrin/tabular-benchmark*, 2022. 5, 6, 9, 10, 14

[35] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *NeurIPS*, 2022. 4, 5, 1

[36] Andrey Gromov. Grokking modular arithmetic. *arXiv preprint arXiv:2301.02679*, 2023. 7

[37] Michael Hahn, Dan Jurafsky, and Richard Futrell. Sensitivity as a complexity measure for sequence classification tasks. *Transactions of the ACL*, 2021. 1

[38] Hrayr Harutyunyan, Rafayel Darbinyan, Samvel Karapetyan, and Hrant Khachatrian. In-context learning in presence of spurious correlations. *arXiv preprint arXiv:2410.03140*, 2024. 1

[39] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GeLUs). *arXiv preprint arXiv:1606.08415*, 2016. 2, 1

[40] Katherine L Hermann and Andrew K Lampinen. What shapes feature representations? exploring datasets, architectures, and training. *arXiv preprint arXiv:2006.12433*, 2020. 1

[41] Hossein Hosseini, Baicen Xiao, Mayoore Jaiswal, and Radha Poovendran. Assessing shape bias property of convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018. 1

[42] Like Hui and Mikhail Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. *arXiv preprint arXiv:2006.07322*, 2020. 1

[43] Ehsan Imani, Kai Luedemann, Sam Scholnick-Hughes, Esraa Elelimy, and Martha White. Investigating the histogram loss in regression. *arXiv preprint arXiv:2402.13425*, 2024. 4

[44] Ameya D Jagtap and George Em Karniadakis. How important are activation functions in regression and classification? a survey, performance comparison, and future directions. *Journal of Machine Learning for Modeling and Computing*, 4(1), 2023. 1

[45] Ameya D Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 2020. 3, 5, 1, 2

[46] Anchit Jain, Rozhin Nobahari, Aristide Baratin, and Stefano Sarao Mannelli. Bias in motion: Theoretical insights into the dynamics of bias in sgd training. *arXiv preprint arXiv:2405.18296*, 2024. 7

[47] Liangze Jiang and Damien Teney. OOD-chameleon: Is algorithm selection for ood generalization learnable? *arXiv preprint arXiv:2410.02735*, 2024. 1

[48] Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. SGD on neural networks learns functions of increasing complexity. *NeurIPS*, 2019. 5, 1

[49] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 3

[50] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, 2017. 1

[51] Tanishq Kumar, Blake Bordelon, Samuel J Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. *arXiv preprint arXiv:2310.06110*, 2023. 7

[52] 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. 3

[53] Alexander Li and Deepak Pathak. Functional regularization for reinforcement learning via learned fourier features. *NeurIPS*, 2021. 1

[54] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *NeurIPS*, 2018. 5

[55] Xuan Li, Yun Wang, and Bo Li. Tree-regularized tabular embeddings. *arXiv preprint arXiv:2403.00963*, 2024. 1

[56] Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 2017. 8

[57] Ziqi Liu, Wei Cai, and Zhi-Qin John Xu. Multi-scale deep neural network (mscalednn) for solving poisson-boltzmann equation in complex domains. *arXiv preprint arXiv:2007.11207*, 2020. 1

[58] Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. *NeurIPS*, 2022. 7

[59] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024. 2

[60] Kaifeng Lyu, Zhiyuan Li, Runzhe Wang, and Sanjeev Arora. Gradient descent on two-layer nets: Margin maximization and simplicity bias. *NeurIPS*, 2021. 1

[61] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. 1

[62] Augustine N Mavor-Parker, Matthew J Sargent, Caswell Barry, Lewis Griffin, and Clare Lyle. Frequency and generalisation of periodic activation functions in reinforcement learning. *arXiv preprint arXiv:2407.06756*, 2024. 1

[63] Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? *NeurIPS*, 2024. 5

[64] Chris Mingard, Joar Skalse, Guillermo Valle-Pérez, David Martínez-Rubio, Vladimir Mikulik, and Ard A Louis. Neural networks are a priori biased towards boolean functions with low entropy. *arXiv preprint arXiv:1909.11522*, 2019. 1, 5

[65] Chris Mingard, Guillermo Valle-Pérez, Joar Skalse, and Ard A Louis. Is SGD a bayesian sampler? well, almost. *Journal of Machine Learning Research*, 2021. 1, 5

[66] Chris Mingard, Henry Rees, Guillermo Valle-Pérez, and Ard A Louis. Do deep neural networks have an inbuilt occam's razor? *arXiv preprint arXiv:2304.06670*, 2023. 1

[67] Tom M Mitchell. The need for biases in learning generalizations. *Rutgers University CS tech report CBM-TR-117*, 1980. 1

[68] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 3

[69] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014. 1

[70] Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. *NeurIPS*, 2021. 1

[71] Tomaso Poggio, Kenji Kawaguchi, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Xavier Boix, Jack Hidary, and Hrushikesh Mhaskar. Theory of deep learning III: the non-overfitting puzzle. *CBMM Memo*, 2018. 1

[72] Karl Popper. *"7. Simplicity". The logic of scientific discovery*. Routledge, 1959. 8

[73] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022. 7, 8, 12

[74] Aahlad Manas Puli, Lily Zhang, Yoav Wald, and Rajesh Ranganath. Don't blame dataset shift! shortcut learning due to gradients and cross entropy. *NeurIPS*, 2023. 1

[75] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *ICML*. PMLR, 2019. 5, 1

[76] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 2017. 1

[77] Sameera Ramasinghe and Simon Lucey. Beyond periodicity: Towards a unifying framework for activations in coordinate-MLPs. In *ECCV*. Springer, 2022. 1

[78] Sameera Ramasinghe, Lachlan E MacDonald, and Simon Lucey. On the frequency-bias of coordinate-MLPs. *NeurIPS*, 2022. 1

[79] Mihaela Rosca, Theophane Weber, Arthur Gretton, and Shakir Mohamed. A case for new neural network smoothness constraints. *I Can't Believe It's Not Better (ICBINB) Workshop at NeurIPS*, 2020. 11

[80] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. Wire: Wavelet implicit neural representations. In *CVPR*, 2023. 4, 1

[81] Simone Scardapane, Michele Scarpiniti, Danilo Comminiello, and Aurelio Uncini. Learning activation functions from data using cubic spline interpolation. *Neural Advances in Processing Nonlinear Dynamic Signals*, 2019. 2

[82] Simone Scardapane, Steven Van Vaerenbergh, Simone Totaro, and Aurelio Uncini. Kafnets: Kernel-based non-parametric activation functions for neural networks. *Neural Networks*, 2019. 2, 3, 1

[83] Jürgen Schmidhuber. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Networks*, 1997. 1

[84] Luca Scimeca, Seong Joon Oh, Sanghyuk Chun, Michael Poli, and Sangdoo Yun. Which shortcut cues will DNNs choose? a study from the parameter-space perspective. *arXiv preprint arXiv:2110.03095*, 2021. 5

[85] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *NeurIPS*, 2020. 2, 5, 6, 11

[86] Kexuan Shi, Xingyu Zhou, and Shuhang Gu. Improved implicit neural representation with fourier reparameterized training. In *CVPR*, 2024. 1

[87] Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. A long way to go: Investigating length correlations in rlhf. *arXiv preprint arXiv:2310.03716*, 2023. 1

[88] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 2020. 1

[89] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 2018. 5

[90] Lawrence Stewart, Francis Bach, Quentin Berthet, and Jean-Philippe Vert. Regression as classification: Influence of task formulation on neural network features. In *ICML*. PMLR, 2023. 4

[91] Leon René Sütfeld, Flemming Brieger, Holger Finger, Sonja Füllhase, and Gordon Pipa. Adaptive blending units: Trainable activation functions for deep neural networks. In *Intelligent Computing: Proceedings of the Computing Conference*. Springer, 2020. 3, 1, 2

[92] Remi Tachet, Mohammad Pezeshki, Samira Shabanian, Aaron Courville, and Yoshua Bengio. On the learning dynamics of deep neural networks. *arXiv preprint arXiv:1809.06848*, 2018. 1

[93] Damien Teney, Ehsan Abbasnejad, Simon Lucey, and Anton van den Hengel. Evading the simplicity bias: Training a diverse set of models discovers solutions with superior ood generalization. *arXiv preprint arXiv:2105.05612*, 2021. 1, 2, 6, 11

[94] Damien Teney, Maxime Peyrard, and Ehsan Abbasnejad. Predicting is not understanding: Recognizing and addressing underspecification in machine learning. In *ECCV*. Springer, 2022. 6, 1

[95] Damien Teney, Yong Lin, Seong Joon Oh, and Ehsan Abbasnejad. ID and OOD performance are sometimes in-

versely correlated on real-world datasets. *NeurIPS*, 2024. 1, 7

[96] Damien Teney, Armand Mihai Nicolicioiu, Valentin Hartmann, and Ehsan Abbasnejad. Neural redshift: Random networks are not random functions. In *CVPR*, 2024. 1, 2, 3, 4, 5, 6, 8, 11, 14

[97] Tijmen Tieleman. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26, 2012. 8

[98] Nikita Tsoy and Nikola Konstantinov. Simplicity bias of two-layer networks beyond linearly separable data. *arXiv preprint arXiv:2405.17299*, 2024. 5, 7

[99] Guillermo Valle-Perez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2018. 1, 5

[100] Colin White, Mahmoud Safari, Rhea Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. Neural architecture search: Insights from 1000 papers. *arXiv preprint arXiv:2301.08727*, 2023. 2

[101] David H Wolpert. The supervised learning no-free-lunch theorems. *Soft computing and industry: Recent applications*, 2002. 1, 8

[102] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 3

[103] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*. Wiley Online Library, 2022. 1

[104] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019. 5, 1

[105] Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. In *ICONIP*. Springer, 2019. 5

[106] Zhi-Qin John Xu, Yaoyu Zhang, and Tao Luo. Overview frequency principle/spectral bias in deep learning. *Communications on Applied Mathematics and Computation*, 2024. 1

[107] Ge Yang, Anurag Ajay, and Pulkit Agrawal. Overcoming the spectral bias of neural value approximation. *arXiv preprint arXiv:2206.04672*, 2022. 1

[108] Ziqian Zhong and Jacob Andreas. Algorithmic capabilities of random transformers. *arXiv preprint arXiv:2410.04368*, 2024. 8

[109] Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Josh Susskind, Samy Bengio, and Preetum Nakkiran. What algorithms can transformers learn? a study in length generalization. *arXiv preprint arXiv:2310.16028*, 2023. 8

[110] Zhangchen Zhou, Yaoyu Zhang, and Zhi-Qin John Xu. A rationale from frequency perspective for grokking in training neural network. *arXiv preprint arXiv:2405.17479*, 2024. 7