Task Discovery: Finding the Tasks that Neural Networks Generalize on

Anonymous Author(s)
Affiliation
Address
email

Abstract

When developing deep learning models, we usually decide what task we want to solve then search in the space of models in order to design one that generalizes well on this task. An intriguing question would be: *what if, instead of fixing the task and searching in the model space, we fix the model and search in the task space?* Can we find tasks that the model generalizes on? How do they look, or do they indicate anything?

This is the question we address in this paper. We propose a *task discovery* framework that automatically finds examples of such tasks via optimizing a generalization-based quantity called *agreement score*. With this framework, we demonstrate that the same set of images can give rise to many tasks on which neural networks generalize well. The understandings from task discovery can also provide a tool to shed more light on deep learning and its failure modes: as an example, we show that the discovered tasks can be used to generate *"adversarial train-test splits"* which make a model fail at test time, without changing the pixels or labels, but by only selecting how the datapoints should be split between training and testing.

1 Introduction

Deep learning models are found to generalize well, i.e. exhibit low test error, when trained on human-labelled tasks (e.g., the "real" human-labelled task in Fig. 1). This can be seen as a consequence of the models' inductive biases that favour solutions with low test error over those which also have zero training loss but higher test error. In this paper, we aim to find what are examples of other tasks[1] that are preferred by neural networks, i.e., on which they generalize well.

We define a quantity called *agreement score* to measure how well a network generalizes on a task. It indicates whether two networks trained on the same task with different training stochasticity (e.g., initialization) make similar predictions on new test images. We show that it, indeed, makes for a useful metric and differentiates between human- and random-labelled tasks (see Fig. 1). Intuitively, a high agreement score can be seen as a necessary condition for generalization, as there cannot be generalization if the agreement score is low and networks converge to different solutions. On the other hand, if the agreement is high, then there is a stable solution that different networks converge to, and, therefore, generalization is possible.

Based on the agreement score as a prerequisite of generalization, we develop a *task discovery* framework that optimizes it and finds new tasks on which neural networks generalize well. Experimentally,

[1] In the context of this paper, generally a "task" is a labeling of a dataset. Any label set defines a task.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

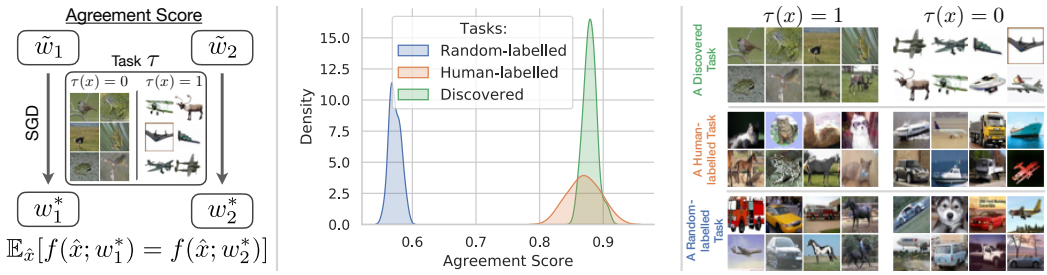

Figure 1: ***Left:*** The agreement score measures weather two networks trained on the task τ with different optimization stochasticity (e.g., initialization) make the same prediction on a test image $\hat{x}$. ***Center:*** The agreement score differentiates between real human-labelled tasks based on CIFAR-10 original labels and random-labelled tasks. The task discovery framework finds novel tasks with AS similar to real ones. ***Right:*** Examples of a discovered, real (animals vs. non-animals) and random-labelled tasks on CIFAR-10. The discovered task looks visually distinct and seems to be based on the image background.

we found that the same images can allow for many different tasks on which different network architectures generalize well (see an example in Fig. 1).

Finally, we show how the proposed framework can help us understand deep learning better by demonstrating its failure modes. We use the discovered tasks to split a dataset into a train-test split in an *adversarial* way. After training on this train set, the network fails to make correct predictions on the test set. Unlike manually created benchmarks that reveals similar failing behaviour [54, 31, 65] or pixel-level adversarial attacks [58, 26, 38], the proposed approach finds the adversarial split automatically and does not need to change image pixels.

2 Related Work

Deep learning generalization. It is not well understood yet why deep neural networks generalize well in practice while being overparameterized and having large complexity [17, 14, 46, 67, 62]. A line of work approach this question by investigating the role of the different components of deep learning [1, 44, 2, 1, 40, 69, 59, 13] and developing novel complexity measures that could predict neural networks' generalization [45, 20, 30, 41, 56]. The proposed task discovery framework can serve as an experimental tool to shed more light on when and how deep learning models generalize.

Similarity measures between networks. Measuring how similar two networks are is challenging and depends on a particular application. Two common approaches are to measure the similarity between hidden representations [29, 24, 27, 53, 28, 40] and the predictions made on unseen data [57]. Similar to our work, [13, 18, 51] also use the AS measure. In contrast to these works, which mainly use the AS as a metric, we turn it into an optimization objective to find new tasks with the desired property of good generalization.

Bias-variance decomposition. The test, which is commonly used to measure generalization, can be decomposed into bias and variance, known as the bias-variance decomposition [12, 5, 60, 9]. Recent works investigate how both terms behave for modern deep learning models [4, 42, 66, 43]. The agreement score used in this work can be seen as a measure of the variance in this decomposition. We, however, characterize its dependence on the task being learned instead of the model's complexity and find the tasks that maximizes it.

Meta-optimization. Meta-optimization methods seek to optimize the parameters of another optimization method. They are commonly used for hyper-parameter tuning [37, 32, 50, 3] and gradient-based meta-learning [11, 64, 48]. We use meta-optimization to find the task parameters that maximize the agreement score, computation of which involves training two networks on this task. Meta-optimization methods are memory and computationally expensive, and multiple solutions were developed [34, 36, 47] to amortize these costs, which can also be applied to improve the efficiency of the proposed task discovery framework.

Creating distribution shifts. Distributions shifts can result from e.g. spurious correlations, under-sampling [65] or adversarial attacks [58, 26, 38]. To create such shifts to, e.g., study the failure modes of networks, one needs to manually define them [54, 31]. We show that it is possible to create many such shifts *automatically* on *a given dataset* using the discovered tasks. The network's accuracy drops significantly (from 0.8 to 0.2) under these shifts (see Sec. 5.3, Fig. 5).

3 Agreement Score: Measuring Consistency of Labeling Unseen Data

In this section, we introduce the basis for the task discovery framework: a task definition and a measure of a task generalizability, the agreement score. We, further, provide an empirical analysis of the agreement score and demonstrate that it differentiates two types of tasks: human-labelled real and random-labelled tasks.

Notation and Definitions. Given a set of N images $X = \{x_i\}_{i=1}^N$, $x \in \mathcal{X}$, we define a *task* as a binary labelling of this set $\tau : X \to \{0, 1\}$ (a multi-class extension is straightforward). We denote the corresponding labelled dataset as $\mathcal{D}(X, \tau) = \{(x, \tau(x)) | x \in X\}$ and the set of all possible tasks on X as $\mathcal{T}_X$. As a learning algorithm $\mathcal{A}$, we consider a neural network $f(\cdot; w) : \mathcal{X} \to [0, 1]$ with weights w trained by gradient optimization on a dataset $\mathcal{D}(X, \tau)$ with cross-entropy loss. Due to the inherent stochasticity, such as random initialization and mini-batching, this learning algorithm induces a distribution over the weights given a dataset $w \sim \mathcal{A}(\mathcal{D}(X, \tau))$.

3.1 Agreement Score as a Measure of Generalization

A standard approach to measure how well a learning algorithm $\mathcal{A}$ generalizes when given a training dataset $\mathcal{D}(X_{\mathrm{tr}}, \tau)$ is to measure the test error on $\mathcal{D}(X_{\mathrm{te}}, \tau)$. It can be decomposed into bias and variance terms [5, 60, 9], where, in our case, the stochasticity is w.r.t. $\mathcal{A}$ and the *train-test split is fixed*. Most works study how this decomposition depends on the choice of $\mathcal{A}$ [43, 66], whereas we examine its dependence on τ for a fixed $\mathcal{A}$. The bias term measures how much the average prediction deviates from τ and can be simply set to zero by only changing the task's test labels on X_{te}. The variance term captures how predictions of different models agree with each other and does not depend on task's test labels, but only on training labels through $\mathcal{D}(X_{\mathrm{tr}}, \tau)$. We, therefore, suggest to measure the generalizability of a task τ with an *agreement score*, as defined below.

For given train-test splits $X_{\mathrm{tr}}, X_{\mathrm{te}}$ and a task τ, we define the agreement score (AS) as:

$$\mathrm{AS}(\tau; X_{\mathrm{tr}}, X_{\mathrm{te}}) = \mathbb{E}_{w_1, w_2 \sim \mathcal{A}(\mathcal{D}(X_{\mathrm{tr}}, \tau))} \mathbb{E}_{x \sim X_{\mathrm{te}}} \mathbb{I}[f(x; w_1) = f(x; w_2)], \tag{1}$$

where the first expectation is over different models trained on $\mathcal{D}(X_{\mathrm{tr}}, \tau)$ and the inner expectation is averaging over the test set. Practically, this corresponds to training two networks from different initializations on the training dataset labelled by τ and measuring the agreement between these two models' predictions on the test set (see the inner-loop in Fig. 3).

The agreement score (Eq. 1) depends only on the task's training labels and no test labels are needed. It, however, is tightly connected to the test error, and having a high-AS task τ that labels training data, one can construct a task with high test accuracy (see Appendix A). Note, however, that when the task is given, the high-AS provides only a necessary but not a sufficient condition for high test accuracy as test labels can take any values in general (e.g., be adversarial as in Sec. 5), and, thus, it cannot be used to predict the test accuracy in this case [18, 23].

3.2 Agreement Score Behaviour for Random and Real Tasks

Here we demonstrate that the agreement score exhibits the desired behaviour of differentiating human-labelled real from random-labelled tasks. We take the set of images X from CIFAR-10 dataset [25] and split the original training set into 45K images for X_{tr} and 5K for X_{te}. We split 10 original classes differently into two sets of 5 classes to construct 5-vs-5 binary classification tasks τ_{r}, out of all $\binom{10}{5}$ tasks we randomly sub-sample 20 ones to form the set of *real tasks* $T_{\mathrm{Real}} = \{\tau_{\mathrm{r}}\}$. We construct the set of 20 *random tasks* T_{Rnd}, by generating a binary label for each image randomly and fixing them throughout the training, similar to [68]. We use ResNet-18 [16] architecture and Adam [22] optimizer as the learning algorithm $\mathcal{A}$, unless otherwise specified. We measure the AS by training two networks for 100 epochs, which is enough to achieve zero training error for all considered tasks.

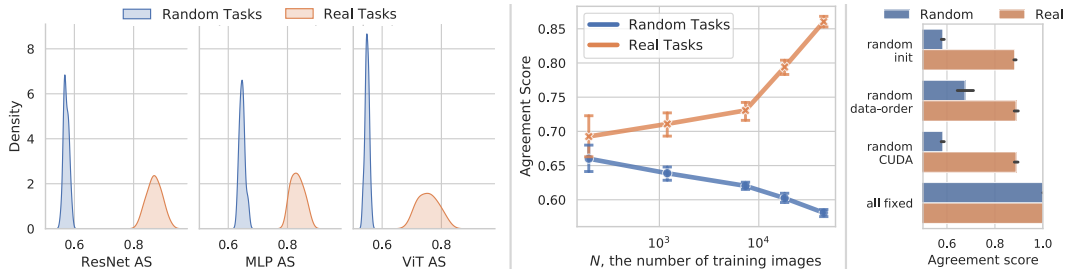

Figure 2: Agreement score for real and random tasks. ***Left:*** AS measured on three architectures: ResNet-18, MLP and ViT. ***Center:*** AS measured on ResNet-18 for different numbers of training images N. The standard deviation is over 4 tasks and 3 different data splits. ***Right:*** Ablating the sources of stochasticity present in $\mathcal{A}$. Each row shows when one of the 1) initialization, 2) data-order or 3) CUDA (see Sec. 3.2) is stochastic and the other two sources are fixed, and the bottom is when all the sources are fixed. The differentiation between real and random tasks persists across different architectures, data-sizes and sources of randomness.

AS differentiates real tasks from random. Fig. 1 shows that real tasks have higher AS than random ones. This coincides with our intuition that one should not expect generalization on a random task, for which AS is close to the chance level of 0.5. Note, however, that random tasks are just samples from $\mathcal{T}_{X_{\text{tr}}}$ (which also includes real ones) and the corresponding empirical distribution of AS (Fig. 2) is an unbiased estimation of the AS distribution over $\mathcal{T}_{X_{\text{tr}}}$, suggesting that high-AS tasks do not make for a large fraction of all tasks. A high AS of real tasks is also consistent with the understanding that a network is able to generalize when trained on CIFAR-10 labels.

How does the AS differ across architectures? In addition to ResNet, we measure the agreement score for real and random tasks using MLP [57] and ViT architectures [10] (see Appendix B). Fig. 2 shows that the AS for all the architectures differentiates real and random tasks well. MLP and ViT have lower AS than ResNet-18, aligned with our understanding that convolutional networks should exhibit better generalization on this small dataset due to their architectural inductive biases. Similar to the architectural analysis, we provide the analysis on the data-dependence of the AS in Appendix H.

How does the AS depend on the training size? When too little data is available for training, one could expect the stochasticity in $\mathcal{A}$ to play a bigger role than data, i.e., the agreement score decreases with less data. This intuition is consistent with empirical evidences on real tasks as can be seen in Fig. 2. However, for random tasks, the AS increases with less data. One possible conjecture is that when training dataset is too small, two networks deviate less from their initializations, which give rise to similar tasks resulting in a high AS.

Any stochasticity is enough to provide differentiation. We ablate the three sources of stochasticity in $\mathcal{A}$: 1) initialization, 2) data-order and 3) CUDA stochasticity [19]. With all three sources fixed, the system is fully deterministic and AS=1. Interestingly, we found that when *any* of these variables change between two runs, the AS drops creating the same separation between real and random tasks.

Thse emperical observations show that the AS differentiate between real and random tasks across multiple architectures, dataset sizes and the sources of stochasticity.

4 Task Discovery via Meta-Optimization of the Agreement Score

As we saw in the previous section, AS provides a useful measure of how well a network can generalize on a given task. A natural question that arises then: *are there high-AS tasks other than real (i.e., human-labelled) ones and what are they?* In this section, we establish a *task discovery* framework to study this question computationally.

4.1 Task Space parametrization and Agreement Score Meta-Optimization

Our goal is to find a task τ that maximizes $\text{AS}(\tau)$, which is a high-dimensional discrete optimization problem. In order to make it differentiable and utilize first-order optimization methods, we, first,

substitute the 0-1 loss in Eq. 1 with the cross-entropy loss l_{ce}. Then, we parameterize the space of tasks with a task-network $t_\theta : \mathcal{X} \to \{0, 1\}$ and treat the AS as a function of its parameters θ. Given all the components are differentiable, we can now evaluate the gradient of the AS w.r.t. task parameters $\nabla_\theta AS(t_\theta)$ and use it for a gradient-based optimization. This results in a meta-optimization problem where the inner-loop optimization is over parameters w_1, w_2 and the outer-loop optimization is over task parameters θ (see Fig. 3).

Evaluating the meta-gradient w.r.t. θ has high memory and computational costs, as the inner-loop incorporates training two networks from scratch. In practice, we found that even after a single epoch there is a separation in AS between random and real tasks, which provides a useful learning signal (see Appendix B). We additionally use gradient checkpointing [7] after every inner-loop step in order to avoid the linear memory consumption in the number of steps. All this together allows us to run the discovery process for the ResNet-18 model on a single 40GB A100.

4.2 Discovering a Set of Dissimilar Tasks with High Agreement Score

The aforementioned AS meta-optimization results in only a single high-AS task, whereas there are potentially many tasks with high AS. We, therefore, would like to discover and inspect a set of tasks $T = \{t_{\theta_1}, \dots, t_{\theta_K}\}$ instead of a single one. A naive approach to find such a set would be to run the meta-optimization from K different intializations of task-parameters θ. However, this results in the set of similar (or even the same) tasks, as we also observed in our preliminary experiments. We, therefore, aim to discover *dissimilar* tasks to better represent the set of all tasks with high AS. We measure the similarity between two tasks on a set X as follows:

$$\text{sim}_X(\tau_1, \tau_2) = \max \left\{ \mathbb{E}_{x \sim X} \mathbb{I}[\tau_1(x) = \tau_2(x)], \mathbb{E}_{x \sim X} \mathbb{I}[\tau_1(x) = 1 - \tau_2(x)] \right\}, \tag{2}$$

where the maximum accounts for the label flipping. Since this metric is not differentiable, we, instead, use a differentiable loss L_{sim} to minimize the similarity (defined later in Sec. 4.3, Eq. 5). Finally, we formulate the task discovery framework as the following optimization problem over T:

$$\underset{T=\{t_{\theta_1}, \dots, t_{\theta_K}\}}{\arg\max} \quad \mathbb{E}_{t_\theta \sim T} \text{AS}(t_\theta) - \lambda \cdot L_{\text{sim}}(T). \tag{3}$$

Note, that we naturally avoid discovering trivial solutions with high AS that are highly imbalanced (e.g., labelling all objects with class 0) due to the similarity loss, as these task will be similar to each other, and a set T that includes them will be penalized.

In practice, we could solve this optimization sequentially, i.e., first find t_{θ_1}, freeze it and add it to T, then optimize for t_{θ_2}, and so on. However, we found this to be slow, so in the next section 4.3 we provide a solution that is more efficient, i.e. can find more tasks with less compute.

Regulated Task discovery. The task discovery formulation above only concerns with finding high-AS tasks that are dissimilar. One can introduce additional constraints to *regulate* the discovery process, e.g., in the form of a regularizer or the task-network architecture. This approach can also allow for a guided task discovery to favour the discovery of one task over the others. We provide an example in Sec. 5.1 by using self-supervised pre-trained embeddings as the input to the task-network.

4.3 Modelling The Set of Tasks with an Embedding Space

Modeling every task by a separate network increases both the memory and computational costs. In order to amortize these costs, we adopt an approach popular in multi-task learning and model task-networks with a shared encoder $e(\cdot; \theta_e) : \mathcal{X} \to \mathbb{R}^d$ and a task-specific linear head $\theta_l \in \mathbb{R}^d$, so $t_\theta(x) = e(x; \theta_e)^\top \theta_l$. See Fig. 3 for visualization. Then, instead of learning a fixed set of different task-specific heads, we aim to learn an embedding space where *any* linear hyperplane gives rise to a high-AS task. Thus, an encoder with parameters θ_e defines the following set of tasks:

$$T_{\theta_e} = \{t_\theta \,|\, \theta = (\theta_e, \theta_l),\ \theta_l \in \mathbb{R}^d\}. \tag{4}$$

This set is not infinite as it might seem at first, since many linear layers will correspond to the same shattering of X. The size of T_{θ_e} as measured by the number of unique tasks on X is only limited by

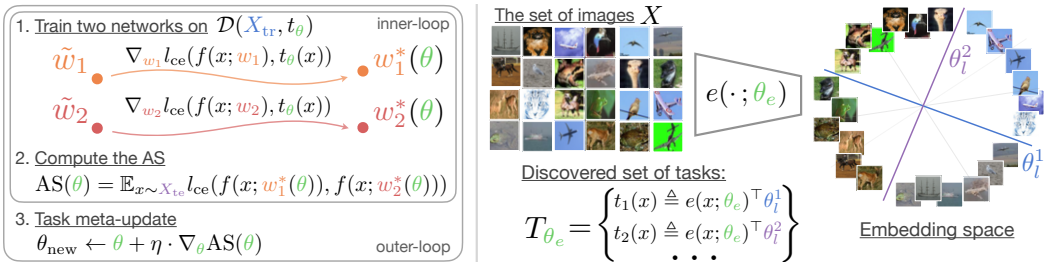

Figure 3: ***Left:*** Agreement score meta-optimization w.r.t. the task. *Inner-loop:* given the task $t_\theta \in T$, two networks w_1, w_2 optimize the cross-entropy loss l_{ce} on the training set labelled by the task (Sec. 4.1). After training, the agreement between two networks is calculated on the test set. *Outer-loop:* the task parameters θ are updated with the meta-gradient of the AS. ***Right:*** the task-space paramertrization with the shared encoder (Sec. 4.3). The encoder $e(\cdot\,; \theta_2)$ (after projection) distributes images uniformly on the sphere in the embedding space. Different tasks are then formed by applying a linear classifier with weights, e.g., θ_l^1, θ_l^2 passing through the origin. The corresponding set of tasks T_{θ_e} consists of all such linear classifiers in this space. This results in a more efficient task discovery framework than modelling each task with a separate network.

the dimensionality of the embedding space d and the encoder's expressivity. Potentially, it can be as big as the set of all shatterings when $d = |X| - 1$ and for an appropriate encoder e [61].

We adopt a uniformity loss over the embedding space [63] to induce dissimilarity between tasks:

$$L_{\text{sim}}(T_{\theta_e}) = L_{\text{unif}}(\theta_e) = \log \mathbb{E}_{x_1, x_2} \exp \left\{ \alpha \cdot \frac{e(x_1; \theta_e)^\top e(x_2; \theta_e)}{\|e(x_1; \theta_e)\| \cdot \|e(x_2; \theta_e)\|} \right\}, \tag{5}$$

where the expectation is taken w.r.t. pairs of objects randomly sampled from the training set. It favours the embeddings of images to be uniformly distributed on a sphere, and, if the encoder satisfies this property, any two orthogonal hyperplanes $\theta_l^1 \perp \theta_l^2$ (passing through origin) give rise to two tasks with similarity 0.5 as measured by Eq. 2 (see Fig. 3).

In order to optimize the AS averaged over T_{θ_e} in Eq. 3 , we use a Monte Carlo gradient estimator by sampling one hyperplane θ_l at each step. To sample dissimilar tasks, we sample orthogonal hyperplanes, which result in dissimilar tasks due to uniformity. See Appendix B for more details. As a result of running the task discovery framework, we find the encoder parameters θ_e^* that optimize the objective Eq. 3 and gives rise to the corresponding set of tasks $T_{\theta_e^*}$ (see Eq. 4 and Fig. 3).

4.4 Discovering high-AS tasks on CIFAR-10 dataset

In this section, we demonstrate that the proposed framework successfully finds dissimilar tasks with high AS. We consider the same setting as in Sec. 3.2. We use the encoder-based discovery described in Sec. 4.3 and use the same ResNet-18 architecture for $e(\cdot; \theta_e)$ by only changing the last classification layer with a linear projection to $\mathbb{R}^d$ with $d = 32$. We use Adam as the meta-optimizer and SGD optimizer for the inner-loop. Refer to Appendix B for more details.

We optimize Eq. 3 to find θ_e^* and sample 32 tasks from $T_{\theta_e^*}$ by taking d orthogonal hyperplanes θ_l. We refer to this set of 32 tasks as T_{ResNet}. The average similarity Eq. 2 between all pairs of tasks from this set is 0.51, close to the minimum 0.5 level. For each discovered task, we evaluate its AS in the same way as in Sec. 3.2 (according to Eq. 1). Fig. 1 demonstrates that the proposed task discovery framework successfully finds tasks with high AS similar to real tasks. See more visualizations and analyses on these discovered tasks in Sec. 5 and the Appendix C.

Random Networks are high-AS tasks. Interestingly, in our experiments we found that if we simply initialize the task-network randomly, the corresponding task (after applying softmax) will have high AS, on par with real and discovered ones (≈ 0.85). Different random initializations, however, give rise to very similar tasks (e.g., 32 randomly sampled network have the average similarity of 0.68 against 0.51 for the task discovery framework, see Appendix D for more detailed comparison). Therefore, this naive approach does not result in an efficient task discovery framework, as one needs to draw probibitely many initializations to get sufficiently dissimilar tasks.

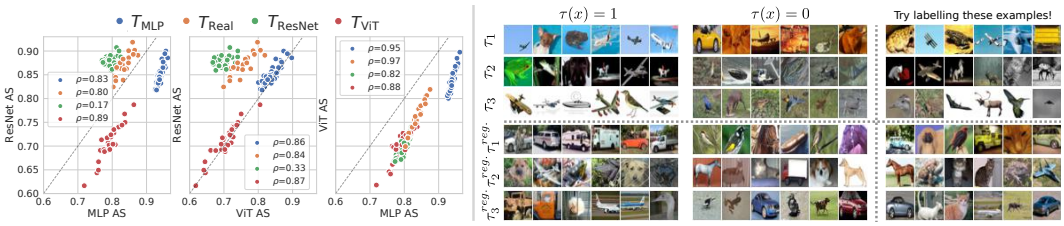

Figure 4: ***Left:*** The AS for tasks from $T_{\text{Real}}, T_{\text{ResNet}}, T_{\text{MLP}}, T_{\text{ViT}}$ measured on all three architectures (x and y axis). ***Right:*** The images for each class of our discovered tasks are shown in the first two columns. We selected some unlabelled examples in the third column for the reader to guess. The answers are in the Appendix C. These images have been sampled to be the most discriminative, as measured by the network predicted probability. ***Right*** (top): Discovered tasks from the unregulated version of task discovery seem to be based on color patterns e.g. $\tau_1(x) = 1$ are images with similar blue color, which makes sense as a learnable task. ***Right*** (bottom): From a regulated version where the encoder was pretrained with SimCLR. The tasks seems to correspond to semantic tasks e.g. $\tau_1^{reg.}(x) = 1$ are images of vehicles with different backgrounds. As the pretraining encourages the encoder to be invariant to color, it seems to be biased towards semantic information. Thus, the framework is able to pick up on the inductive biases from SimCLR pretraining, via the discovered tasks.

5 An Empirical Study of the Discovered Tasks

In this section, we first perform a qualitative analysis on the discovered tasks in Sec. 5.1. Second, we analyze how discovered tasks differ for three different architectures in Sec. 5.2. Finally, in Sec. 5.3, we present how the discovered tasks can be used to construct an *adversarial attack* for a network by simply curating a train-test split.

5.1 Qualitative analyses of the discovered tasks

Here we attempt to analyze the discovered tasks. Fig. 4 (top right) shows examples of images for three discovered tasks found in Sec. 4.4. For ease of visualization, we selected the most discriminative images from each class as measured by the network's predicted probability. Some tasks seems to be based on color, e.g. τ_1 class 1 seems to includes images with blue (sky or water) color, and class 0 images with orange color. In the interest of space, the visuals for all tasks are in the Appendix C, alongside some post-hoc interpretability analysis.

Regulated task discovery with contrastive pre-training described in Sec. 4.2 allows us to incorporate additional information to favour tasks based on high-level concepts. We use augmentation-based contrastive pre-training that learns embeddings that are invariant to color information [8, 15, 6, 39]. Specifically, we use embeddings of the ResNet-50 [16] trained with SimCLR [7] as the input to the encoder e instead of raw images, which in this case is a 2-layer fully-connected network.

Fig. 4 (bottom right) shows the resulting tasks. As the encoder is now more invariant to color, the discovered tasks seems to be more aligned with CIFAR-10 classes e.g. samples from τ_1 class 1 show vehicles against different backgrounds. Thus, the framework is able to represent different types of inductive biases, in this case, the effect of SSL pretraining, via the discovered tasks.

Do we discover human-labelled tasks? A natural question to ask is whether the set of discovered tasks contains real tasks corresponding to human-labelled semantic classes. To answer this, for each real task $\tau_{\text{r}} \in T_{\text{real}}$, we find the most similar discovered task $\hat{\tau}_d \in T_{\theta_e^*}$ for the encoder θ_e^* from Sec. 4.4 (see Appendix E for more details). We observe that no real task is contained in the set of discovered set as all the similarities $\text{sim}(\tau_{\text{r}}, \hat{\tau}_d)$ are below 0.7. We also found that simply increasing the dimensionality of the embedding space does not help significantly. Please see the Appendix E for more discussions.

5.2 High-AS tasks for different architectures

In this section, we study whether the AS measured on different architectures correlate well on different groups tasks discovered for each of these architectures. We consider the same architectures as in Sec. 4.4: ResNet-18, MLP, and ViT. We change both f and e to be one of MLP or ViT and run

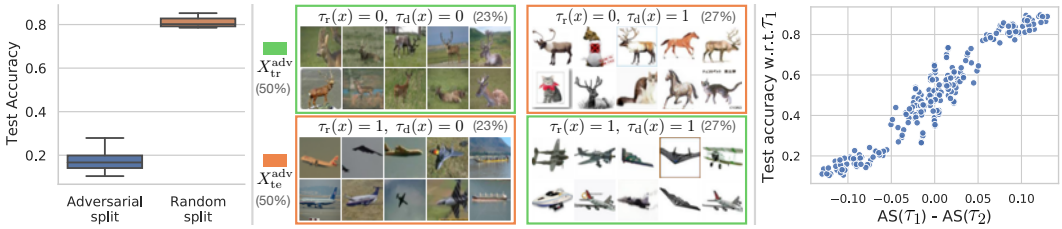

Figure 5: Adversarial splits on CIFAR-10. ***Left:*** Test accuracy for real tasks τ_{r} on two types of splits: random and adversarial based on a τ_{d} (see the center image and Eq. 6). The boxplot distribution is over four τ_{r} tasks for the random split and 24 $(\tau_{\text{r}}, \tau_{\text{d}})$ pairs for the adversarial one. Training with the adversarial split results in a *significant* drop in test performance. ***Center:*** Example of an adversarial split, constructed such that train and test data have the opposite correlation between τ_{r} and τ_{d} (the percentage of images in each group is shown in brackets). The model seems to make its predictions based on τ_{d}, hence, the significant test accuracy drop. ***Right:*** Each dot corresponds to a pair of tasks (τ_1, τ_2), the *y-axis* is the test accuracy after training on the corresponding adversarial split, and the *x-axis* is the difference in the agreement score of two tasks averaged over random train-test splits. The plot suggests that the network is *biased towards the task with a higher AS*: the higher the AS of τ_1 is compared to τ_2, the higher the accuracy w.r.t. τ_1 is.

the same discovery process as in Sec. 4.4. As a result, we obtain three sets: $T_{\text{ResNet}}, T_{\text{MLP}}, T_{\text{ViT}}$, each of 32 tasks. For each task, we evaluate its AS on all three architectures. Fig. 4 shows that high-AS tasks for one architecture do not necessary have similar AS when measured on another architecture (e.g., T_{ResNet} on MLP and ViT). For MLP and ViT architectures, we find that the AS groups correlate well for all groups, which is aligned with the understanding that these architectures are more similar to each other than to ResNet-18.

More importantly, we note that it is not enough to compare architectures on any single group of tasks. For example, if comparing the AS for ResNet-18 and MLP only on real tasks T_{Real}, one might conclude that they correlate well ($\rho = 0.8$), suggesting they generalize similarly. However, when the set T_{ResNet} is taken into consideration, the conclusion changes ($\rho = 0.17$). Thus, it is important to analyse the different architectures on a broader set of tasks not to bias our understanding, and the proposed task discovery framework allows for more complete analyses.

5.3 Adversarial Splits: creating adversarial train-test dataset partition using discovered tasks

For a given real task τ_{r}, let us consider the standard procedure of training a network on $\mathcal{D}(X_{\text{tr}}, \tau_{\text{r}})$ and testing it on $\mathcal{D}(X_{\text{te}}, \tau_{\text{r}})$. The network achieves high test accuracy on average when the train-test split is sampled randomly. Using discovered tasks, we show how to construct *adversarial* splits on which the test accuracy drops significantly. To do that, we consider a task τ_{d} and construct the split, s.t. two tasks $\tau_{\text{r}}, \tau_{\text{d}}$ have the same labels on the training set $X_{\text{tr}}^{\text{adv}}$ and the opposite ones on the test set $X_{\text{te}}^{\text{adv}}$ (see Fig. 5):

$$X_{\text{tr}}^{\text{adv}} = \{x \,|\, \tau_{\text{r}}(x) = \tau_{\text{d}}(x),\ x \in X\}, \quad X_{\text{te}}^{\text{adv}} = \{x \,|\, \tau_{\text{r}}(x) \neq \tau_{\text{d}}(x),\ x \in X\}. \tag{6}$$

Fig. 5 shows that for different pairs of $(\tau_{\text{r}}, \tau_{\text{d}})$ after training the the network on $\mathcal{D}(X_{\text{tr}}^{\text{adv}}, \tau_{\text{r}})$, the test accuracy on $\mathcal{D}(X_{\text{te}}^{\text{adv}}, \tau_{\text{r}})$ drops significantly from 0.8 to 0.2, suggesting the network predicts $1 - \tau_{\text{r}}$ on $X_{\text{te}}^{\text{adv}}$, which coincides with τ_{d}. We note that we keep the class-balance and sizes of the random and adversarial splits approximately the same (see Fig. 5).

The discovered task τ_{d}, in this case, can be seen as a spurious feature [55, 21] and the adversarial split creates a spurious correlation between τ_{r} and τ_{d} on $X_{\text{tr}}^{\text{adv}}$, that fools the network. Similar behaviour was observed before on datasets where spurious correlations were curated manually [54, 31, 65]. In contrast, the described approach using the discovered tasks allows us to find such spurious features, to which networks are vulnerable, *automatically*. It can potentially find spurious correlations on datasets where none was known to exist or find new ones on existing benchmarks as we show below.

Neural networks learn the task with a higher AS. The empirical observation above demonstrates that when a network is given a training dataset where τ_{r} and τ_{d} coincide, favours predicting the latter. While theoretical analysis is needed to answer and understand this phenomena, here, we provide an empirical clue towards its understanding.

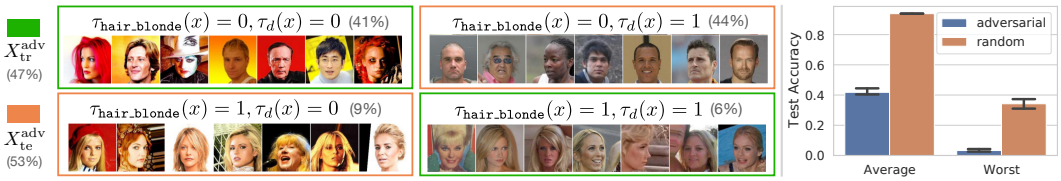

Figure 6: Adversarial splits on CelebA. ***Left:*** Sample images for a adversarial split (the percentage of images in each group is shown in brackets). ***Right:*** Test accuracy for real tasks τ_{r} on two types of splits: random and adversarial, based on a τ_{d}. On the x-axis, "average" corresponds to the average over all the test data, and the "worst" group is over the (`hair_blonde`=1, `male`=1) images. Training on the adversarial split results in a significant drop in performance indicating that the network again learnt to classify based on τ_d instead of τ_r.

We consider a set of 10 discovered tasks (5 from each of the discovery runs with $d \in \{8, 32\}$) and 4 real tasks. For each pair of tasks (τ_1, τ_2), we construct the adversarial split according to Eq. 6, train a network on $\mathcal{D}(X_{\text{tr}}^{\text{adv}}, \tau_1)$ and test it on $\mathcal{D}(X_{\text{te}}^{\text{adv}}, \tau_1)$. Fig. 5 shows the test accuracy against the difference in the agreement score of these two tasks $\text{AS}(\tau_1) - \text{AS}(\tau_2)$. We find that the test accuracy w.r.t. τ_1 well correlates with the difference in the agreement: when $\text{AS}(\tau_1)$ is sufficiently larger than $\text{AS}(\tau_2)$, the network makes predictions according to τ_1, and vice-versa. When agreement scores are similar, the network makes test predictions according to neither of them (but the AS on the adversarial split mostly remains high, see Appendix G).

Adversarial split for CelebA via random network task. Hair color prediction on the CelebA dataset [33] is a popular benchmark for research on spurious correlations due to the naturally occurring correlation between hair color (`hair_blonde`$\in \{0, 1\}$) and gender (`male`$\in \{0, 1\}$): images with (`hair_blonde`=1, `male`=0) occurs more often than (`hair_blonde`=1, `male`=1). We show the effectiveness of our proposed approach and automatically create an adversarial split for this dataset.

We use the original set of train and val images from CelebA as X and the `hair_blonde` attribute as τ_{r}. To get a high-AS task, we utilize the phenomena observed in Sec. 4.4 and use a randomly initialized network as τ_{d}. Fig. 6 shows images from the corresponding adversarial split and the test accuracy for both splits. We measure the test accuracy on all test images (average) and (`hair_blonde`=1, `male`=1) images (the worst group). The adversarial split fools the network significantly in this case as well according to both average and worst-case accuracy.

6 Conclusion and Limitations

In this work, we introduce task discovery, a framework that finds examples of tasks on which a given network architecture generalizes well. It uses the agreement score as the measure of generalization and optimizes it over the space of tasks. We show the effectiveness of this approach and demonstrate multiple examples of such tasks, other than original human-labelled ones. This framework provides an empirical tool to analyze neural networks through the lens of the tasks they generalize on and can potentially help us to better understand how deep learning works. We, however, only scratched the surface of this more general research direction, and there are several limitations to our work.

Completeness: the set of discovered tasks is not complete and does not necessarily include all the tasks with high agreement score (e.g., human-labelled tasks). Novel optimization methods that better traverse different optima, e.g. [49, 35], and further scaling are needed to address this problem. Also, while the proposed encoder-based parametrization results in an efficient task discovery, there might not be an encoder such that the corresponding set of tasks T_{θ_e} contains all and only high-AS ones.

Scalability: the task discovery framework relies on an expensive meta-optimization, which limits its applicability to large-scale detests. This problem can be addressed in future works with recent advances in efficient meta-optimization methods [34, 52].

Interpretability: Analysis of discovered high-AS tasks can shed more light on how deep learning works (e.g., Sec. 5.3). However, not all of these tasks might be easily interpreted and understood (see Appendix E for more detailed discussion).

References

[1] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019.

[2] Devansh Arpit, Stanisław Jastrzębski, 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 *International conference on machine learning*, pages 233–242. PMLR, 2017.

[3] Harkirat Singh Behl, Atılım Güneş Baydin, and Philip HS Torr. Alpha maml: Adaptive model-agnostic meta-learning. *arXiv preprint arXiv:1905.07435*, 2019.

[4] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[5] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

[7] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[9] Pedro Domingos. A unified bias-variance decomposition. In *Proceedings of 17th international conference on machine learning*, pages 231–238. Morgan Kaufmann Stanford, 2000.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[12] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[13] Guy Hacohen, Leshem Choshen, and Daphna Weinshall. Let's agree to agree: Neural networks share classification order on real datasets. In *International Conference on Machine Learning*, pages 3950–3960. PMLR, 2020.

[14] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[18] Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J Zico Kolter. Assessing generalization of sgd via disagreement. *arXiv preprint arXiv:2106.13799*, 2021.

[19] Mohammad Hadi Jooybar. *Deterministic execution on GPU Architectures*. PhD thesis, University of British Columbia, 2013.

[20] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[21] Fereshte Khani and Percy Liang. Removing spurious features can hurt accuracy and affect groups disproportionately. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 196–205, 2021.

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

[23] Andreas Kirsch and Yarin Gal. A note on" assessing generalization of sgd via disagreement". *arXiv preprint arXiv:2202.01851*, 2022.

[24] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

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

[26] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[27] Aarre Laakso and Garrison Cottrell. Content and cluster analysis: assessing representational similarity in neural systems. *Philosophical psychology*, 13(1):47–76, 2000.

[28] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

[29] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, John E Hopcroft, et al. Convergent learning: Do different neural networks learn the same representations? In *FE@ NIPS*, pages 196–212, 2015.

[30] Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. In *The 22nd international conference on artificial intelligence and statistics*, pages 888–896. PMLR, 2019.

[31] Weixin Liang and James Zou. Metashift: A dataset of datasets for evaluating contextual distribution shifts and training conflicts. *arXiv preprint arXiv:2202.06523*, 2022.

[32] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[33] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

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

[35] Jonathan Lorraine, Paul Vicol, Jack Parker-Holder, Tal Kachman, Luke Metz, and Jakob Foerster. Lyapunov exponents for diversity in differentiable games. *arXiv preprint arXiv:2112.14570*, 2021.

[36] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International conference on machine learning*, pages 2952–2960. PMLR, 2016.

[37] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015.

[38] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[39] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[40] Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems*, 31, 2018.

[41] Vaishnavh Nagarajan and J Zico Kolter. Generalization in deep networks: The role of distance from initialization. *arXiv preprint arXiv:1901.01672*, 2019.

[42] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[43] Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018.

[44] 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.

[45] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401. PMLR, 2015.

[46] Quynh Nguyen and Matthias Hein. Optimization landscape and expressivity of deep cnns. In *International conference on machine learning*, pages 3730–3739. PMLR, 2018.

[47] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[48] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.

[49] Jack Parker-Holder, Luke Metz, Cinjon Resnick, Hengyuan Hu, Adam Lerer, Alistair Letcher, Alexander Peysakhovich, Aldo Pacchiano, and Jakob Foerster. Ridge rider: Finding diverse solutions by following eigenvectors of the hessian. *Advances in Neural Information Processing Systems*, 33:753–765, 2020.

[50] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International conference on machine learning*, pages 737–746. PMLR, 2016.

[51] Iuliia Pliushch, Martin Mundt, Nicolas Lupp, and Visvanathan Ramesh. When deep classifiers agree: Analyzing correlations between learning order and image statistics. *arXiv preprint arXiv:2105.08997*, 2021.

[52] Aniruddh Raghu, Jonathan Lorraine, Simon Kornblith, Matthew McDermott, and David K Duvenaud. Meta-learning to improve pre-training. *Advances in Neural Information Processing Systems*, 34, 2021.

[53] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

[54] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

[55] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning*, pages 8346–8356. PMLR, 2020.

[56] Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

[57] Gowthami Somepalli, Liam Fowl, Arpit Bansal, Ping Yeh-Chiang, Yehuda Dar, Richard Baraniuk, Micah Goldblum, and Tom Goldstein. Can neural nets learn the same model twice? investigating reproducibility and double descent from the decision boundary perspective. *arXiv preprint arXiv:2203.08124*, 2022.

[58] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[59] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

[60] Giorgio Valentini and Thomas G Dietterich. Bias-variance analysis of support vector machines for the development of svm-based ensemble methods. *Journal of Machine Learning Research*, 5(Jul):725–775, 2004.

[61] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.

[62] Roman Vershynin. Memory capacity of neural networks with threshold and relu activations. *arXiv preprint arXiv:2001.06938*, 2020.

[63] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

[64] Lilian Weng. Meta-learning: Learning to learn fast. *lilianweng.github.io*, 2018.

[65] Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre Alvise-Rebuffi, Ira Ktena, Taylan Cemgil, et al. A fine-grained analysis on distribution shift. *arXiv preprint arXiv:2110.11328*, 2021.

[66] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking bias-variance trade-off for generalization of neural networks. In *International Conference on Machine Learning*, pages 10767–10777. PMLR, 2020.

[67] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Small relu networks are powerful memorizers: a tight analysis of memorization capacity. *Advances in Neural Information Processing Systems*, 32, 2019.

[68] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[69] Shengjia Zhao, Hongyu Ren, Arianna Yuan, Jiaming Song, Noah Goodman, and Stefano Ermon. Bias and generalization in deep generative models: An empirical study. *Advances in Neural Information Processing Systems*, 31, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]