Task Discovery: Finding the Tasks that Neural Networks Generalize on

Andrei Atanov **Andrei Filatov** **Teresa Yeo** **Ajay Sohmshetty** **Amir Zamir**

Swiss Federal Institute of Technology (EPFL)

https://taskdiscovery.epfl.ch

Abstract

When developing deep learning models, we usually decide what task we want to solve then search for a model that generalizes well on the 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? These are the questions 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*. We demonstrate that one set of images can give rise to many tasks on which neural networks generalize well. These tasks are a reflection of the *inductive biases* of the learning framework and the *statistical patterns present in the data*, thus they can make a useful tool for analysing the neural networks and their biases. As an example, we show that the discovered tasks can be used to automatically create *"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 the train and test sets. We end with a discussion on human-interpretability of the discovered tasks.

1 Introduction

Deep learning models are found to generalize well, i.e., exhibit low test error when trained on human-labelled tasks. This can be seen as a consequence of the models' inductive biases that favor solutions with low test error over those which also have low training loss but higher test error. In this paper, we aim to find what are examples of other tasks[1] that are favored by neural networks, i.e., on which they generalize well. We will also discuss some of the consequences of such findings.

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

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

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

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

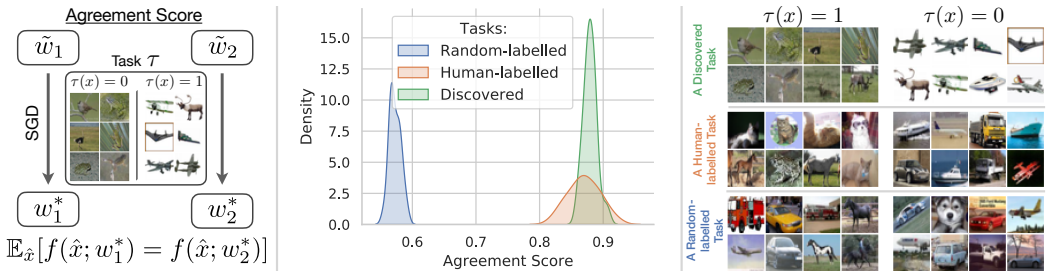

Figure 1: ***Left:*** The *agreement score* (AS) measures whether two networks trained on the task τ with different optimization stochasticity (e.g., initialization) make the same prediction on a test image $\hat{x}$. $f(\hat{x}, w)$ denotes a network with weights w applied to an input $\hat{x}$. $\tilde{w}$ and w^* denote initial and converged weights, respectively. ***Center:*** The agreement score successfully differentiates between human-labelled tasks based on CIFAR-10 original labels and random-labelled tasks. The *task discovery* framework finds novel tasks with high agreement scores. ***Right:*** Examples of a discovered, human-labelled (animals vs. non-animals) and random-labelled tasks on CIFAR-10. This particular discovered task looks visually distinct and seems to be based on the image background.

Finally, we discuss how the proposed framework can help us understand deep learning better, for example, by demonstrating its biases and failure modes. We use the discovered tasks to split a dataset into train test sets in an *adversarial* way instead of random splitting. After training on the train set, the network fails to make correct predictions on the test set. The adversarial splits can be seen as using the "spurious" correlations that exist in the datasets via discovered tasks. We conjecture that these tasks provide strong adversarial splits since task discovery finds tasks that are "favored" by the network the most. Unlike manually curated benchmarks that reveal similar failing behaviours [67, 47, 77], or pixel-level adversarial attacks [71, 40, 51], the proposed approach finds the adversarial split automatically and *does not need to change any pixels or labels* or collect new difficult or adversarial images.

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 [26, 21, 59, 80, 75]. A line of work approaches this question by investigating the role of the different components of deep learning [1, 57, 2, 1, 53, 82, 72, 19] and developing novel complexity measures that could explain neural networks' generalization [58, 32, 46, 54, 69]. The proposed task discovery framework can serve as an experimental tool to shed more light on when and how deep learning models generalize. Also, compared to existing quantities for studying generalization, namely [81], which is based on the *training process only*, our agreement score is directly *based on test data and generalization*.

Similarity measures between networks. Measuring the similar between two networks is challenging and depends on a particular application. Two common approaches are to measure the similarity between hidden representations [45, 38, 41, 66, 44, 53] and the predictions made on unseen data [70]. Similar to our work, [19, 29, 64] also use an agreement score 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 AS used in this work can be seen as a measure of the variance term in the bias-variance decomposition of the test error [18, 4, 73, 14]. Recent works investigate how this term behaves in deep learning models [3, 55, 79, 56]. In contrast, we characterize its dependence on the task being learned instead of the model's complexity and find those that maximize the AS.

Creating distribution shifts. Distributions shifts can result from e.g. spurious correlations, undersampling [77] or adversarial attacks [71, 40, 51]. To create such shifts to study the failure modes of networks, one needs to define them manually [67, 47]. We show that it is possible to *automatically* create *many* such shifts that lead to a significant accuracy drop on a given dataset using the discovered tasks (see Sec. 6 and Fig. 5-left). In contrast to other automatic methods to find such shifts, the proposed approach allows to creates many of them for a learning algorithm rather than for a single trained model [16], does not require additional prior knowledge [12] or adversarial optimization [42], and does not change pixel values [40] or labels.

Data-centric analyses of learning. Multiple works study how training data influences the final model. Common approaches are to measure the effect of removing, adding or mislabelling individual data points or a group of them [36, 37, 27, 28]. Instead of choosing which objects to train on, in task discovery, we choose how to label a fixed set of objects (i.e., tasks), s.t. a network generalizes well.

3 Agreement Score: Measuring Consistency of Labeling Unseen Data

In this section, we introduce the basis for the task discovery framework: the definition of a task and the agreement score, a measure of the task's generalizability. We then provide an empirical analysis of this score and demonstrate that it differentiates human- 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 (i.e., label sets) on X as $\mathcal{T}_X$. We consider a learning algorithm $\mathcal{A}$ to be a neural network $f(\cdot; w) : \mathcal{X} \to [0, 1]$ with weights w trained by SGD 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 the generalization of a learning algorithm $\mathcal{A}$ trained on $\mathcal{D}(X_{\text{tr}}, \tau)$ is to measure the test error on $\mathcal{D}(X_{\text{te}}, \tau)$. The test error can be decomposed into bias and variance terms [4, 73, 14], where, in our case, the stochasticity is due to $\mathcal{A}$ while the *train-test split is fixed*. We now examine how this decomposition depends on the task τ. The bias term measures how much the average prediction deviates from τ and mostly depends on what are the test labels on X_{te}. The variance term captures how predictions of different models agree with each other and does not depend on the task's test labels but only on training labels through $\mathcal{D}(X_{\text{tr}}, \tau)$. We suggest measuring the generalizability of a task τ with an *agreement score*, as defined below.

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

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

where the first expectation is over different models trained on $\mathcal{D}(X_{\text{tr}}, \tau)$ 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 Fig. 1-left and the inner-loop in Fig. 3-left).

The AS depends only on the task's training labels, thus, test labels are not required. However, it is tightly connected to the test error, as having a high-AS task τ that labels training data, one can construct a task with a high test accuracy. We note, however, that when the task is given and fixed, the high-AS provides only a necessary but not a sufficient condition for a high test accuracy as test labels can take any values in general (e.g., be adversarial as in Sec. 5), thus, it cannot be used to predict the test accuracy, in this case [29, 35]. Refer to Appendix J for a more in-depth discussion.

3.2 Agreement Score Behaviour for Random- and Human-Labelled Tasks

Here, we demonstrate that the AS exhibits the desired behaviour of differentiating human-labelled from random-labelled tasks. We take the set of images X from the CIFAR-10 dataset [39] 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_{hl}. Out of all $\binom{10}{5}$ tasks, we randomly sub-sample 20 ones to form the set of *human-labelled tasks* T_{HL}. We construct the set of 20 *random-labelled tasks* T_{RL} by generating binary labels for all images randomly and fixing them throughout the training, similar to [81]. We use ResNet-18 [24] architecture and Adam [34] 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.

AS differentiates human- from random-labelled tasks. Fig. 1-center shows that human-labelled tasks have a higher AS than random-labelled 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, that the empirical distribution of the AS for random-labelled tasks (Fig. 2-left) is an unbiased estimation of the AS distribution over all possible tasks, as T_{RL} are uniform samples from $\mathcal{T}_{X_{\text{tr}}}$. This suggests that high-AS tasks do not make for a large fraction of all tasks. A high AS of human-labelled tasks

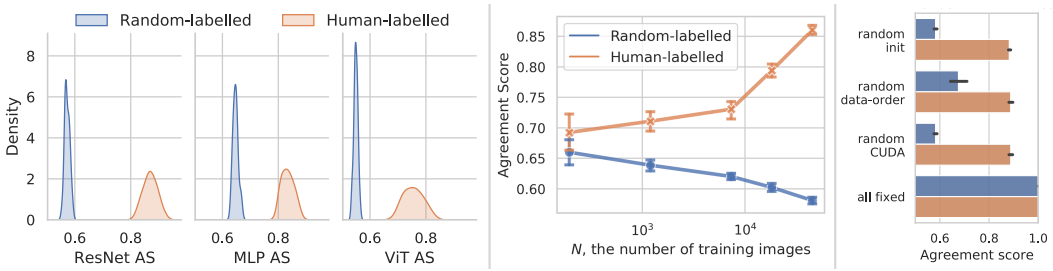

Figure 2: Agreement score for human- and random-labelled tasks. ***Left:*** AS measured on three architectures: ResNet-18, MLP and ViT. ***Center:*** AS measured on ResNet-18 for different numbers of training images N. The standard deviation is over four random tasks and three data splits. ***Right:***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

is also consistent with the understanding that a network is able to generalize when trained on the original CIFAR-10 labels.

How does the AS differ across architectures? In addition to ResNet-18, we measure the AS using MLP [70] and ViT architectures [15] (see Appendix L.3). Fig. 2-left shows that the AS for all the architectures distinguishes tasks from T_{RL} and T_{HL} 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-dependency of the AS in Appendix C.

How does the AS depend on the training size? When only 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 evidence for human-labelled tasks, as seen in Fig. 2-center. However, for random-labelled tasks, the AS increases with less data (but they still remain distinguishable from human-labelled tasks). One possible reason is that when the training dataset is very small, and the labels are random, the two networks do not deviate much from their initializations. This results in similar networks and consequently a high AS, but basically irrelevant to the data and uninformative.

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 [31]. The system is fully deterministic with all three sources fixed, thus, AS=1. Interestingly, when *any* of these variables change between two runs, the AS drops, creating separation between human- and random-labelled tasks.

These empirical observations show that the AS well differentiates between human- and random-labelled tasks across multiple architectures, dataset sizes and sources of stochasticity.

4 Task Discovery via Meta-Optimization of the Agreement Score

As the previous section shows, AS is a useful measure of network's generalization on a given task. A natural question then arises: *are there high-AS tasks other than human-labelled ones, and what are they?* In this section, we establish a *task discovery* framework to automatically search for these tasks and study this question computationally.

4.1 Task Space Parametrization and Agreement Score Meta-Optimization

Our goal is to find a task τ that maximizes $\text{AS}(\tau)$. It is a high-dimensional discrete optimization problem which is computationally hard to solve. In order to make it differentiable and utilize more efficient 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. This basically means that we view the labelled training dataset $\mathcal{D}(X_{\text{tr}}, \tau)$ and a network t_θ with the same labels on X_{tr} as being equivalent, as one can always train a network to fit the training dataset.

Given that all the components are now differentiable, we can calculate the gradient of the AS w.r.t. task parameters $\nabla_\theta AS(t_\theta)$ by unrolling the inner-loop optimization and use it for gradient-based

optimization over the task parameters θ. 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 θ (see Fig. 3-left).

Evaluating the meta-gradient w.r.t. θ has high memory and computational costs, as we need to train two networks from scratch in the inner-loop. To make it feasible, we limit the number of inner-loop optimization steps to 50, which we found to be enough to separate the AS between random and human-labelled tasks and provide a useful learning signal (see Appendix L.1). We additionally use gradient checkpointing [8] after every inner-loop step to avoid linear memory consumption in the number of steps. This allows us to run the discovery process for the ResNet-18 model on the CIFAR-10 dataset using a single 40GB A100. See Appendix L.2 for more details.

4.2 Discovering *a Set* of Dissimilar Tasks with High Agreement Scores

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

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

where the maximum accounts for the labels' 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:

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

We show the influence of λ on task discovery in Appendix H. Note that this naturally avoids discovering trivial solutions that are highly imbalanced (e.g., labelling all objects with class 0) due to the similarity loss, as these tasks are 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. In the next Sec. 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 – which is the minimum requirement for having generalizable tasks. One can introduce additional constraints to *regulate* the discovery process, e.g., by adding a regularizer to Eq. 3 or via the task network's architectural biases. This approach can allow for a guided task discovery to favor the discovery of particular tasks. We provide an example of a regulated discovery 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

Modelling every task by a separate network increases 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 that $t_\theta(x) = e(x;\theta_e)^\top \theta_l$. See Fig. 3-right 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:

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

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 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 the encoder e is flexible enough [74].

We adopt a uniformity loss over the embedding space [76] to induce dissimilarity between tasks:

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

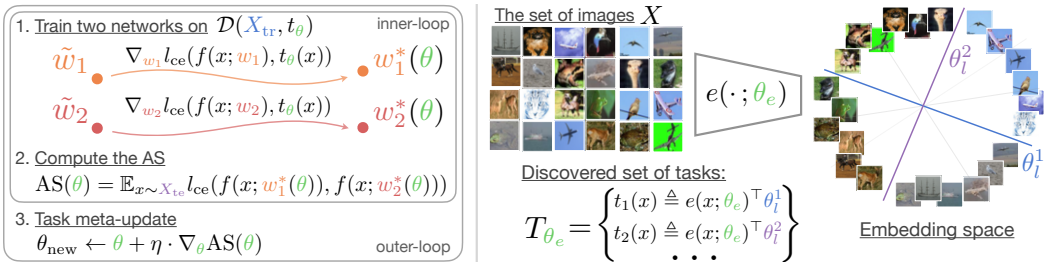

Figure 3: ***Left:*** Agreement score meta-optimization. *Inner-loop:* given the task $t_\theta \in T$, two networks w_1, w_2 optimize the cross-entropy loss l_{ce} on the training set labelled by the task t_θ (Sec. 4.1). After training, the agreement between two networks is calculated on the test set. *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 and 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.

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

In order to optimize the AS averaged over T_{θ_e} in Eq. 3, we can use a Monte-Carlo gradient estimator and sample one hyper-plane θ_l at each step, e.g., w.r.t. an isotropic Gaussian distribution, which, on average, results in dissimilar tasks given that the uniformity is high. 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-right).

Note that the framework outlined above can be straightforwardly extended to the case of discovering multi-way classification tasks as we show in Appendix D, where instead of sampling a hyperplane that divides the embedding space into two parts, we sample K hyperplanes that divide the embedding space into K regions and give rise to K classes.

Towards *the space* of high-AS tasks. Instead of creating *a set* of tasks, one could seek to define *a space* of high-AS tasks. That is to define *a basis set* of tasks and a binary operation on a set of tasks that constructs a new task and preserves the AS. The proposed formulation with a shared embedding space can be seen as a step toward this direction. Indeed, in this case, the structure over $T_{\theta_e^*}$ is imposed implicitly by the space of liner hyperplanes $\theta_l \in \mathbb{R}^d$, each of which gives rise to a high-AS task.

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 ResNet-18 architecture for $e(\cdot; \theta_e)$ with a linear layer mapping to $\mathbb{R}^d$ (with $d = 32$) instead of the last classification layer. We use Adam as the meta-optimizer and SGD optimizer for the inner-loop optimization. Please refer to Appendix L.2 for more details.

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

Random networks give rise to high-AS tasks. Interestingly, in our experiments, we found that if we initialize a task network randomly (standard uniform initialization from PyTorch [63]), the corresponding task (after applying softmax) will have a high AS, on par with human-labelled and discovered ones (≈ 0.85). Different random initializations, however, give rise to very similar tasks (e.g., 32 randomly sampled networks have an average similarity of 0.68 compared to 0.51 for the discovered tasks. See Appendix K for a more detailed comparison). Therefore, a naive approach of sampling random networks does not result in an efficient task discovery framework, as one needs to

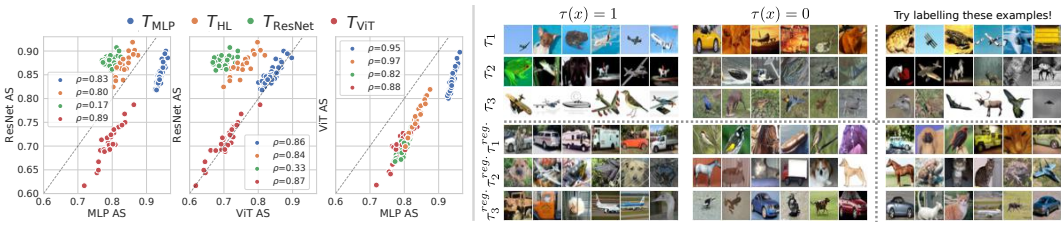

Figure 4: ***Left:*** The AS for tasks from T_{HL}, T_{ResNet}, T_{MLP}, T_{ViT} measured on all three architectures (x and y axis). ***Right:*** Sample images for each class of our discovered tasks are shown in the first two columns. We show some unlabelled examples in the third column for the reader to guess the label. The answers are in the Appendix B. These images have been sampled to be the most discriminative, as measured by the network predicted probability. ***Right*** (top): Some of the 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 make sense as a learnable task and are expected (see Sec. 5.1). ***Right*** (bottom): The same, but for a regulated version where the encoder was pretrained with SimCLR. The tasks seems to correspond more 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.

draw prohibitively many of them to get sufficiently dissimilar tasks. Note, that this result is specific to the initialization scheme used and not *any* instantiation of the network's weights necessarily results in a high-AS task.

5 Empirical Study of the Discovered Tasks

In this section, we first perform a qualitative analysis of the discovered tasks in Sec. 5.1. Second in Sec. 5.2, we discuss the human-interpretability of the discovered tasks and whether human-labelled tasks should be expected to be discovered. Then, we analyze how discovered tasks differ for three different architectures in Sec. 5.3. We also study the dependency of the AS on the test data domain and show the results in Appendix C in the interest of space.

5.1 Qualitative Analyses of the Discovered Tasks

Here we attempt to analyze the tasks discovered by the proposed discovery framework. Fig. 4-top-right shows examples of images for three sample 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. In the interest of space, the visuals for all tasks are in the Appendix B, alongside some post hoc interpretability analysis.

Some of these tasks seem to be based on color, e.g. the class 1 of τ_1 includes images with blue (sky or water) color, and the class 0 includes images with orange color. Other tasks seem to pick up other cues. These are basically reflections of the *statistical patterns present in the data* and the *inductive biases of the learning architecture*.

Regulated task discovery described in Sec. 4.2 allows us to incorporate additional information to favor the discovery of specific tasks, e.g., ones based on more high-level concepts. As an example, we use self-supervised contrastive pre-training that learns embeddings invariant to the employed set of augmentations [20, 22, 5, 52]. Specifically, we use embeddings of the ResNet-50 [24] trained with SimCLR [9] as the input to the encoder e instead of raw images, which in this case is a 2-layer fully-connected network. Note that the AS is still computed using the raw pixels.

Fig. 4-bottom-right shows the resulting tasks. Since the encoder is now more invariant to colour information due to the colour jittering augmentation employed during pre-training [9], the discovered tasks seem to be more aligned with CIFAR-10 classes; e.g. samples from τ_1^{reg}'s class 1 show vehicles against different backgrounds. Note that task discovery regulated by contrastive pre-training only provides a tool to discover tasks invariant to the set of employed augmentations. The choice of augmentations, however, depends on the set of tasks one wants to discover. For example, one should not employ a rotation augmentation if they need to discover view-dependent tasks [78].

5.2 On Human-Interpretability and Human-Labelled Tasks

In this section, we discuss *i)* if the discovered tasks should be visually interpretable to humans, *ii)* if one should expect them to contain human-labelled tasks and if they do in practice, and *iii)* an example of how the discovery of human-labelled tasks can be promoted.

Should the discovered high-AS tasks be human-interpretable? In this work, by "human-interpretable tasks", we generally refer to those classifications that humans can visually understand or learn sufficiently conveniently. Human-labelled tasks are examples of such tasks. While we found in Sec. 3.2 that such interpretable tasks have a high AS, the opposite is not true, i.e., not *all* high-AS tasks should be visually interpretable by a human – as they reflect the inductive biases of the particular learning algorithm used, which are not necessarily aligned with human perception. The following "green pixel task" is an example of such a high-AS task that is learnable and generalizable in a statistical learning sense but not convenient for humans to learn visually.

The *"green pixel task"*. Consider the following simple task: the label for x is 1 if the pixel p at a fixed position has the green channel intensity above a certain threshold and 0 otherwise. The threshold is chosen to split images evenly into two classes. This simple task has the AS of approximately 0.98, and a network trained to solve it has a high test accuracy of 0.96. Moreover, the network indeed seems to make the predictions based on this rule rather than other cues: the accuracy remains almost the same when we set all pixels but p to random noise and, on the flip side, drops to the chance level of 0.5 when we randomly sample p and keep the rest of the image untouched. This suggests that the network captured the underlying rule for the task and generalizes well in the statistical learning sense. However, it would be hard for a human to infer this pattern by only looking at examples of images from both classes, and consequently, it would appear like an uninterpretable/random task to human eyes.

It is sensible that many of the discovered tasks belong to such a category. This indicates that finding more human-interpretable tasks would essentially require *bringing in additional constraints and biases* that the current neural network architectures do not include. We provide an example of such a promotion using the SSL regulated task discovery results below.

Do discovered tasks contain human-labelled tasks? We observe that the similarity between the discovered and most human-labelled tasks is relatively low (see Fig. 11-left and Appendix E for more details). As mentioned above, human-labelled tasks make up only a small subset of all tasks with a high AS. The task discovery framework aims to find different (i.e., dissimilar) tasks from this set and not necessarily *all* of them. In other words, there are many tasks with a high AS other than human-labelled ones, which the proposed discovery framework successfully finds.

As mentioned above, introducing additional inductive biases would be necessary to promote finding more human-labelled tasks. We demonstrate this by using the tasks discovered with the SimCLR pre-trained encoder (see Sec. 5.1). Fig. 11-right shows that the recall of human-labelled tasks among the discovered ones increases notably due to the inductive biases that SimCLR data augmentations bring in.

5.3 The Dependency of the Agreement Score and Discovered Tasks on the Architecture

In this section, we study how the AS of a task depends on the neural network architecture used for measuring the AS. We include human-labelled tasks as well as a set of tasks discovered using different architectures in this study. We consider the same architectures as in Sec. 3.2: ResNet-18, MLP, and ViT. We change both f and e to be one of MLP or ViT and run 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 with 32 tasks. For each task, we evaluate its AS on all three architectures. Fig. 4-left shows that high-AS tasks for one architecture do not necessary have similar AS when measured on another architecture (e.g., T_{ResNet} on ViT). For MLP and ViT architectures, we find that ASs correlate well for all groups for tasks, which is aligned with the understanding that these architectures are more similar to each other than to ResNet-18.

More importantly, we note that comparing architectures on any single group of tasks is not enough. For example, if comparing the AS for ResNet-18 and MLP only on human-labelled tasks T_{HL}, one might conclude that they correlate well ($\rho = 0.8$), suggesting they generalize on similar tasks. However, when the set T_{ResNet} is taken into account, 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.

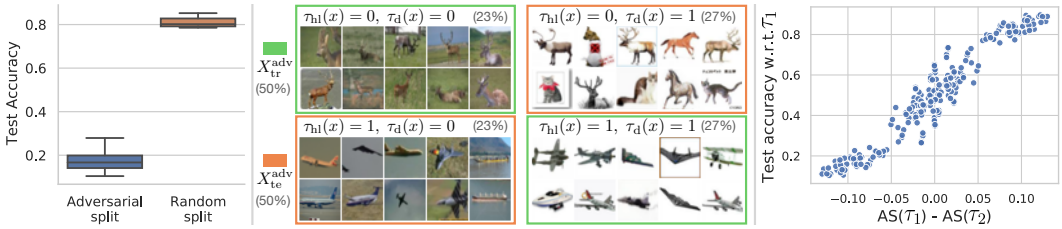

Figure 5: Adversarial splits on CIFAR-10. ***Left:*** Test accuracy for human-labelled tasks τ_{hl} on random and adversarial splits. The boxplot distribution is over four τ_{hl} tasks for the random split and 24 $(\tau_{\text{hl}}, \tau_{\text{d}})$ pairs for the adversarial one. Training with the adversarial split results in a significant test performance drop. ***Center:*** Example of an adversarial split, constructed such that images from train and test sets have the opposite correlation between τ_{hl} and τ_{d} (the percentage of images in each group is shown in brackets). The model seems to learn to make its predictions based on τ_{d} instead of τ_{hl} (hence, the significant test accuracy drop). ***Right:*** Each dot corresponds to a pair of tasks (τ_1, τ_2), where the *y-axis* is the test accuracy after training on the corresponding adversarial split to predict τ_1, and the *x-axis* is the difference in the AS of these two tasks. The plot suggests that the network *favors learning the task with a higher AS*: the higher the AS of τ_1 is, the higher the accuracy w.r.t. τ_1 is. Thus, for an adversarial split to be successful, the AS of the target task τ_1 should be lower than the AS of τ_2.

6 Adversarial Dataset Splits

In this section, we demonstrate how the discovered tasks can be used to reveal biases and failure modes of a given network architecture (and training pipeline). In particular, we introduce an *adversarial split* – which is a train-test dataset split such that the test accuracy of the yielded network significantly drops, compared to a random train-test split. This is because the AS can reveal what tasks a network *favors* to learn, and thus, the discovered tasks can be used to construct such adversarial splits.

6.1 Creating Adversarial Train-Test Dataset Partitions Using Discovered High-AS Tasks

For a given human-labelled task τ_{hl}, let us consider the standard procedure of training a network on $\mathcal{D}(X_{\text{tr}}, \tau_{\text{hl}})$ and testing it on $\mathcal{D}(X_{\text{te}}, \tau_{\text{hl}})$. The network usually achieves a high test accuracy when the dataset is split into train and test sets randomly. Using discovered tasks, we show how to construct *adversarial* splits on which the test accuracy drops significantly after training. To do that, we take a discovered task τ_{d} with a high AS and construct the split, s.t. τ_{hl} and τ_{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-center):

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

Fig. 5-left shows that for various pairs of $(\tau_{\text{hl}}, \tau_{\text{d}})$, the test accuracy on $\mathcal{D}(X_{\text{te}}^{\text{adv}}, \tau_{\text{hl}})$ drops significantly after training on $\mathcal{D}(X_{\text{tr}}^{\text{adv}}, \tau_{\text{hl}})$. This suggests the network chooses to learn the cue in τ_{d}, rather than τ_{hl}, as it predicts $1 - \tau_{\text{hl}}$ 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-center).

A discovered task τ_{d}, in this case, can be seen as a spurious feature [68, 33] and the adversarial split creates a spurious correlation between τ_{hl} 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 [67, 47, 77]. 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 shown below.

In Appendix F, we show how to extend this approach to multi-class classification, and provide results for the results for the original 10-way classification task of the CIFAR-10 dataset. We also construct and present adversarial adversarial splits for ImageNet and CelebA datasets in Appendix G.

6.2 Neural Networks Favor Learning the Task with a Higher AS.

The empirical observation made in the previous section demonstrates that a network trained on a dataset where τ_{hl} and τ_{d} coincide predicts the latter. While theoretical analysis is needed to understand the cause of this phenomenon, here, we provide an empirical clue towards its understanding.

We consider a set of 10 discovered tasks and 4 human-labelled tasks. For all pairs of tasks (τ_1, τ_2) from this set, 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-right shows the test accuracy against the difference in the agreement

score of these two tasks $\mathrm{AS}(\tau_1) - \mathrm{AS}(\tau_2)$. We find that the test accuracy w.r.t. τ_1 correlates well with the difference in the agreement: when $\mathrm{AS}(\tau_1)$ is sufficiently larger than $\mathrm{AS}(\tau_2)$, the network makes predictions according to τ_1, and vice-versa. When the agreement scores are similar, the network makes test predictions according to neither of them (see Appendix I for further discussion). This observation suggests that an adversarial split is successful, i.e., causes the network to fail, if the AS of the target task τ_1 is lower than that of the task τ_2 used to create the split.

7 Conclusion

In this work, we introduce *task discovery*, a framework that finds tasks on which a given network architecture generalizes well. It uses the Agreement Score (AS) as the measure of generalization and optimizes it over the space of tasks to find generalizable ones. We show the effectiveness of this approach and demonstrate multiple examples of such generalizable tasks. We find that these tasks are not limited to human-labelled ones and can be based on other patterns in the data. This framework provides an empirical tool to analyze neural networks through the lens of the tasks they generalize on and can potentially help us better understand deep learning. Below we outline a few research directions that can benefit from the proposed task discovery framework.

Understanding neural networks' inductive biases. Discovered tasks can be seen as a reflection of the inductive biases of a learning algorithm (network architectures, optimization with SGD, etc.), i.e., a set of preferences that allow them to generalize on these tasks. Therefore, having access to these biases in the form of concrete tasks could help us understand them better and guide the development of deep learning frameworks.

Understanding data. As discussed and shown, task discovery depends on, not only the learning model, but also the data in hand. Through the analysis of discovered tasks, one can, for example, find patterns in data that interact with a model's inductive biases and affect its performance, thus use the insights to guide dataset collection.

Generalization under distribution shifts. The AS turned out to be predictive of the cues/tasks a network favors in learning. The consequent adversarial splits (Sec. 6) provide a tool for studying the biases and generalization of neural networks under distribution shifts. They can be constructed automatically for datasets where no adverse distribution shifts are known and help us to build more broad-scale benchmarks and more robust models.

8 Limitations

The proposed instantiation of a more general task discovery framework has several limitations, which we outline below, along with potential approaches to address them.

Completeness: the set of discovered tasks does not necessarily include *all* of the tasks with a high AS. Novel optimization methods that better traverse different optima of the optimization objective Eq. 3, e.g., [61, 50], and further scaling are needed to address this aspect. Also, while the proposed encoder-based parametrization yields an efficient task discovery method, it imposes some constraints on the discovered tasks, as there might not exist 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 datasets. This problem can be potentially addressed in future works with recent advances in efficient meta-optimization methods [49, 65] as well as employing effective approximations of the current processes.

Interpretability: As we discussed and demonstrated, analysis of discovered high-AS tasks can shed more light on how the neural networks work. However, it is the expected behavior that not all of these tasks may be easily interpretable or "useful" in the conventional sense (e.g. as an unsupervised pre-training task). This is more of a consequence of the learning model under discovery, rather than the task discovery framework itself. Having discovered tasks that exhibit such properties requires additional inductive biases to be incorporated in the learning model. This was discussed in Sec. 5.2.

Acknowledgment: We thank Alexander Sax, Pang Wei Koh, Roman Bachmann, David Mizrahi and Oğuzhan Fatih Kar for the feedback on earlier drafts of this manuscript. We also thank Onur Beker and Polina Kirichenko for the helpful discussions.

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

[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. 2

[3] 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. 2

[4] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006. 2, 3

[5] 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

[6] Brandon Carter, Siddhartha Jain, Jonas W Mueller, and David Gifford. Overinterpretation reveals image classification model pathologies. *Advances in Neural Information Processing Systems*, 34, 2021. 19

[7] Brandon Carter, Jonas Mueller, Siddhartha Jain, and David Gifford. What made you do this? understanding black-box decisions with sufficient input subsets. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 567–576. PMLR, 2019. 17

[8] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. 5

[9] 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. 7, 17

[10] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017. 17

[11] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014. 19

[12] Elliot Creager, Joern-Henrik Jacobsen, and Richard Zemel. Environment inference for invariant learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2189–2200. PMLR, 18–24 Jul 2021. 2

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 22

[14] Pedro Domingos. A unified bias-variance decomposition. In *Proceedings of 17th international conference on machine learning*, pages 231–238. Morgan Kaufmann Stanford, 2000. 2, 3

[15] 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. 4

[16] Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. Domino: Discovering systematic errors with cross-modal embeddings. *arXiv preprint arXiv:2203.14960*, 2022. 2

[17] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 19

[18] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992. 2

[19] 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. 2

[20] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006. 7

[21] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016. 2

[22] 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. 7

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 29

[24] 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. 3, 7, 29

[25] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019. 19

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

[27] Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*, 2022. 3

[28] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J. Spanos. Towards efficient data valuation based on the shapley value. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1167–1176. PMLR, 16–18 Apr 2019. 3

[29] Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J Zico Kolter. Assessing generalization of sgd via disagreement. *arXiv preprint arXiv:2106.13799*, 2021. 2, 3, 26

[30] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017. 19

[31] Mohammad Hadi Jooybar. *Deterministic execution on GPU Architectures*. PhD thesis, University of British Columbia, 2013. 4

[32] 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. 2

[33] 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. 9

[34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[35] Andreas Kirsch and Yarin Gal. A note on" assessing generalization of sgd via disagreement". *arXiv preprint arXiv:2202.01851*, 2022. 3, 26

[36] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017. 3

[37] Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 32, 2019. 3

[38] 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. 2

[39] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 3, 19

[40] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016. 2

[41] Aarre Laakso and Garrison Cottrell. Content and cluster analysis: assessing representational similarity in neural systems. *Philosophical psychology*, 13(1):47–76, 2000. 2

[42] Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed Chi. Fairness without demographics through adversarially reweighted learning. *Advances in neural information processing systems*, 33:728–740, 2020. 2

[43] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–104. IEEE, 2004. 19

[44] 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. 2

[45] 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. 2

[46] 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. 2

[47] Weixin Liang and James Zou. Metashift: A dataset of datasets for evaluating contextual distribution shifts and training conflicts. *arXiv preprint arXiv:2202.06523*, 2022. 2, 9

[48] 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. 22

[49] 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. 10

[50] 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. 10

[51] 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. 2

[52] 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. 7

[53] 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. 2

[54] Vaishnavh Nagarajan and J Zico Kolter. Generalization in deep networks: The role of distance from initialization. *arXiv preprint arXiv:1901.01672*, 2019. 2

[55] 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. 2

[56] 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. 2

[57] 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. 2

[58] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401. PMLR, 2015. 2

[59] Quynh Nguyen and Matthias Hein. Optimization landscape and expressivity of deep cnns. In *International conference on machine learning*, pages 3730–3739. PMLR, 2018. 2

[60] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. 19

[61] 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. 10

[62] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012. 19

[63] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 6, 29

[64] 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. 2

[65] 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. 10

[66] 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. 2

[67] 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. 2, 9

[68] 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. 9

[69] Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017. 2

[70] 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. 2, 4

[71] 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. 2

[72] 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. 2

[73] 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. 2, 3

[74] 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. 5

[75] Roman Vershynin. Memory capacity of neural networks with threshold and relu activations. *arXiv preprint arXiv:2001.06938*, 2020. 2

[76] 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. 5

[77] 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. 2, 9

[78] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. *arXiv preprint arXiv:2008.05659*, 2020. 7

[79] 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. 2

[80] 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. 2

[81] 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. 2, 3

[82] 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. 2

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]

Appendix Outline

You can find additional visualizations in the supplementary materials file. This appendix contains the following sections:

- In Appendix A we provide additional results on task discovery on TinyImageNet.
- In Appendix B we provide **vizualizations for more discovered tasks** and provide the answers for the quiz in Fig. 4 of the main paper.
- In Appendix C we present the results of **measuring the AS on test data from different image domains and datasets**.
- In Appendix D we describe how to extend the proposed task discovery method to the case of multi-class tasks and provide results of the discovery.
- In Appendix E we provide the experiment setup description and the quantitative results for the discussion on human-interpretability of the discovered tasks from Sec. 5.2.
- In Appendix F we **show how to extend the proposed adversarial splits to a general K-way classification task and provide results for CIFAR-10 and ImageNet**
- In Appendix G we present adversarial splits for Imagenet and CelebA datasets constructed using randomly initialized networks.
- In Appendix H we show how does the λ hyperparameter influence task discovery.
- In Appendix I we provide additional discussion on the adversarial splits' results from Sec. 5.3.
- In Appendix J, **we show the connection between the AS and test accuracy.**
- In Appendix K we provide more details on the observation made in Sec. 4.4 that **randomly initialized networks give rise to high-AS tasks**.
- In Appendix L we describe **the settings for the task-discovery in more details**, including the discussion on the modified differentiable version of the AS used for meta-optimization.

A Task Discovery on Tiny ImageNet

We extend our experimental setting by scaling it along the dataset size axis and run the proposed task discovery framework on the TinyImageNet dataset [10]. We use the same framework with the shared embedding space formulation with $d \in \{8, 32\}$ and ResNet18 with global pooling after the last convolutional layer. Fig. 6shows examples of discovered tasks. All discovered tasks have AS above 0.8, whereas human-labelled binary tasks (constructed using original classes in the same way as for CIFAR-10 as described in Sec. 3.2) have AS below 0.65. We assume that this may be due to the fact that TinyImageNet contains semantically close classes. Binary task can assign different labels for these classes, which forces the agreement to be lower because it is more difficult for model learn on how to discriminate these classes.

B Vizualization of the Tasks Discovered on CIFAR-10

We provide a complete set of visuals from task networks with different embedding space dimensions ($d = \{8, 32, 128\}$). The top 10 tasks for each d, as measured by AS, are shown in Fig. 8 and the full set of tasks can be found in the `supmat_taskviz` folder in the supplementary material. Visuals for the regulated version of task discovery, with an encoder pre-trained with SimCLR [9], is also shown in Fig. 8. The task network used for visualizations in Fig. 4 in the main paper has $d = 8$. As in the main paper, the images for each class have been selected to be the most discriminative, as measured by the network's predicted probabilities. As d increases, the less interpretable the tasks seem to be. With lower values of d, the tasks seem to be based on colour patterns. Refer to Sec. 5.2 for a discussion as to whether it is reasonable to expect these tasks to be interpretable.

We also attempt to analyze how the task network makes its decisions. To do this, we use a post hoc interpretability method, Sufficient Input Subsets (SIS) [7]. SIS aims to find the minimal subset of the input, i.e. pixels, whose values suffice for the model to make the same decision as on the original input. Running SIS gives us a ranking of each pixel in the image. We assign the value 1 to the top 5%

Figure 6: Tasks Visualization for TinyImageNet for $d = \{8, 32\}$. The top 10 tasks for each d, as measured by AS is shown (if $d < 10$, then d tasks are selected). Each column and row shows selected images from a task, for class 1 ($\tau(x) = 1$) and class 0 ($\tau(x) = 0$). In the y-axis, we show the fraction of images in class 1 in brackets and the AS for that task. The images for each class have been selected to be the most discriminative, as measured by the network's predicted probabilities.

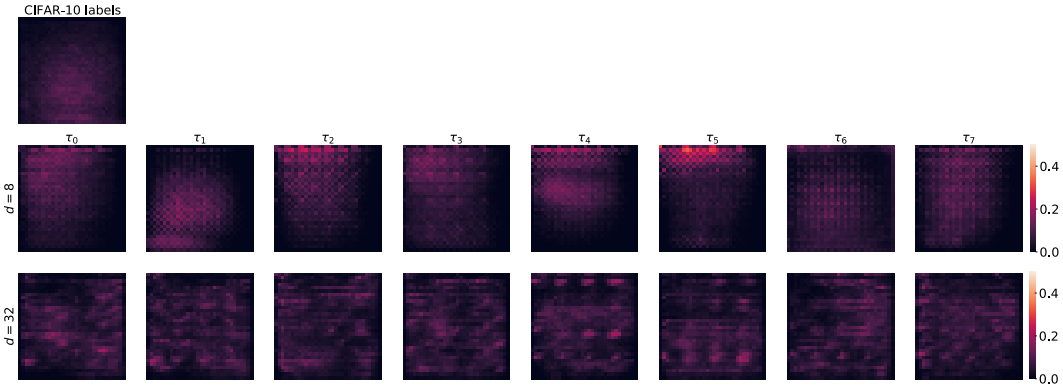

Figure 7: Sufficient input subsets for tasks networks for $d = \{8, 32\}$. Each plot shows the heatmap of the frequency that each pixel was selected to be in the sufficient subset. The task network with $d = 8$ seems to attend to more macro features in the image, while with $d = 32$, it seems to use cues for generalization that are scatter over the image.

of the pixels and 0 to all the others. Thus, each image results in a binary mask. Similar to [6], we aggregate these binary masks for each task for tasks networks with $d = \{8, 32\}$ to get the heatmaps shown in Fig. 7. This allows us to see if different tasks are biased towards different areas of the image. Note that this only tells us where the network may be looking at in the image, but not how it is using that information. For comparison, we also show the SIS map from training on the CIFAR-10 original classification task and a random-labelled task. As expected, the heatmap for CIFAR-10 is roughly centred, as objects in the images are usually in the centre. Furthermore, increasing d results in heatmaps that are more scattered. This seems to suggest that task networks discovered with a smaller embedding size use cues in the image that are more localized.

The labels of the images in Fig. 4 (last column) of the main paper are as follows: $\tau_1 = [0, 1, 0, 1, 1, 0]$, $\tau_2 = [1, 1, 1, 0, 0, 0]$, $\tau_3 = [0, 0, 1, 1, 1, 0]$, $\tau_1^{reg.} = [1, 0, 0, 1, 0, 1]$, $\tau_2^{reg.} = [0, 1, 0, 0, 1, 1]$, $\tau_3^{reg.} = [0, 1, 1, 1, 0, 0]$.

C Dependence of the Agreement Score on the Test Data Domain

In this section, we study how the AS of different tasks depends on the test data used to measure the agreement between two networks. We fix X_{tr} to be images from the CIFAR-10 dataset and take test images X_{te} from the following datasets: CIFAR-100 [39], CLEVR[30], Describable Textures Dataset [11], EuroSAT [25], KITTI [17], 102 Category Flower Dataset [60], Oxford-IIIT Pet Dataset [62], SVHN [43]. We also include the AS measured on noise images sampled from a standard normal distribution. Before measuring the AS, we standardise images channel-wise for each dataset to have zero mean and unit variance similar to training images.

The results for all test datasets and architectures are shown in Fig. 9. First, we can see that the differentiation between human-labelled and discovered tasks and random-labelled tasks successfully persists across all the datasets. For ResNet-18 and MLP, we can see that the AS of discovered tasks stays almost the same for all considered domains. This might suggest that the patterns the discovered tasks are based on are less domain-dependent and provide reliable cues across different image domains. On the other hand, human-labelled tasks are based on high-level semantic patterns corresponding to original CIFAR-10 classes, which change drastically across different domains (e.g., there are no animals or vehicles in CLEVR images), and, as a consequence, their AS is more domain-dependent and volatile.

D Discovering Multi-Class Classification Tasks

In this section, we show how the task discovery framework with a shared embedding space proposed in Sec. 4.3 can be extended to the case of multi-class classification tasks. That is, instead of discovering high-AS binary tasks, we are interested in finding a K-way classification task $\tau : \mathcal{X} \to \{0, \dots, K\}$. Note that the formulation of the agreement score (Eq.1 in the main paper) and its differentiable approximation (see Sec. 4.1 and Sec. L.2) remain almost the same. To model the set of tasks T, we use the same shared encoder formulation as in Sec. 4.3, but with the linear layer predicting K logits, i.e., $\theta_l \in \mathbb{R}^{d \times K}$ and optimize the same loss with the uniformity regularizer. In order to sample tasks where classes are balanced (a similar number of images in each class), we construct a linear layer $\theta_l = [\theta_l^1, \dots, \theta_l^K]$, s.t. $\angle(\theta_l^i, \theta_l^{i+1}) = 2\pi/K$, by randomly sampling θ_l^1 and a direction along which we rotate it. Note, that if the uniformity constraint is satisfied, then the fraction of images corresponding to each class will be $\angle(\theta_l^i, \theta_l^{i+1})/2\pi = 1/K$, i.e., classes will be balanced.

E Do Discovered Tasks Contain Human-Labelled Tasks?

Here, we describe in more detail how we answer this question studied in Sec. 5.2. In order to answer it, for each human-labelled task $\tau_{\text{hl}} \in T_{\text{HL}}$, we find the most similar discovered task $\hat{\tau}_d \in T_{\theta_e^*}$ for the encoder $e(\cdot\,; \theta_e^*)$ from Sec. 4.4. We, however, cannot simply list all the discovered tasks from this set and compare them against all human-labelled ones since this set is virtually infinitely large (any hyperplane is supposed to give rise to a high-AS task). Therefore, for each task τ_{hl}, we fit a linear classifier θ_l^r using embeddings as inputs and τ_{hl} as the target. We use the corresponding task $\hat{\tau}_d(x) = \theta_l^{r\top} e(x; \theta_e^*)$ as (an approximation of) the most similar task from the set of discovered tasks $T_{\theta_e^*}$. We run task discovery with different embedding space size to study whether an embedding space with higher dimensionality give rise to discovered tasks that are more similar to the human-labelled

Figure 8: Tasks visualizations for $d = \{8, 32, 128\}$ and the regulated version of task discovery. The top 10 tasks for each d, as measured by AS is shown (if $d < 10$, then d tasks are selected). Each column and row shows selected images from a task, for class 1 ($\tau(x) = 1$) and class 0 ($\tau(x) = 0$). In the y-axis, we show the fraction of images in class 1 in brackets and the AS for that task. The images for each class have been selected to be the most discriminative, as measured by the network's predicted probabilities. For unregulated tasks, the higher d is, the less (immediately) interpretable the tasks seem to be. For low d, the tasks seem to be based on colour patterns. On the other hand, the regulated tasks seem to be based on semantic information. Refer to Sec. 5.2 for a discussion as to whether it is reasonable to expect these tasks to be interpretable.

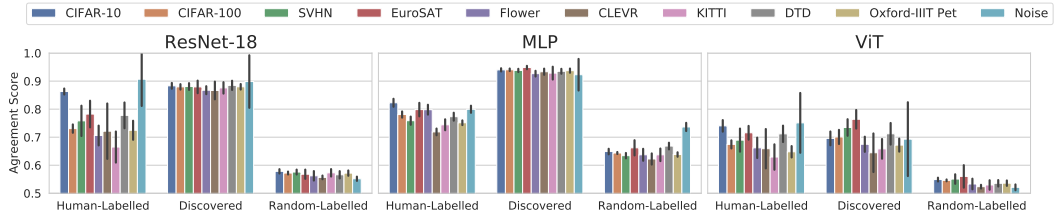

Figure 9: The Dependence of the Agreement Score on the Test Data Domain. We fix training images X_{tr} to be CIFAR-10 and vary X_{te} to measure the AS. For each architecture, we take five tasks from each set of human-labelled, random-labelled and discovered (for the same architecture) tasks and measure their AS on different test data domains. The standard deviation is over five tasks and three random seeds for each pair of a task type and test domain. We see that the differentiation between human-labelled and discovered tasks and random-labelled tasks persists across multiple domains. We also find that the AS of discovered tasks is more sta

Figure 10: Visualisation of 10-way Task Discovery on CIFAR-10. Each column and row shows selected images from a task, for class $i = \{0, \ldots 9\}$ ($\tau(x) = i$). The images for each class have been selected to be the most discriminative, as measured by the network's predicted probabilities.

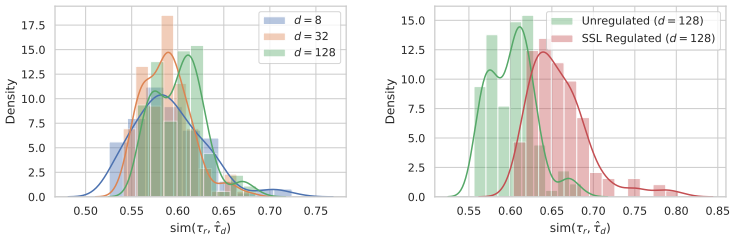

Figure 11: Are human-labelled tasks present in the set of discovered tasks? For each available human-labelled task $\tau \in T_{\text{HL}}$, we find the most similar discovered task $\hat{\tau}_d \in T_{\theta^*_e}$ and plot the corresponding similarities $\text{sim}(\tau_{\text{hl}}, \hat{\tau}_d)$ as a distribution. ***Left:*** Unregulated task discovery with varying embedding space dimensionalities d. The similarity stays relatively low for most human-labelled tasks suggesting they are not "fully" included in the set of discovered tasks at the scale with which we experimented. ***Right:*** Unregulated and SSL regulated versions of task discovery with $d = 128$. As expected, the tasks discovered by the regulated version are more similar to human-labelled tasks due to the additional inductive biases.

Split	CIFAR-10	CelebA	ImageNet, top-1
Random	0.78 ± 0.04	0.94 ± 0.00	0.59 ± 0.01
Adversarial	0.41 ± 0.10	0.42 ± 0.02	0.29 ± 0.00

Table 1: The test set accuracy of networks trained on the original multi-class tasks for CIFAR-10 and ImageNet, and on `hair_blond` attribute for CelebA. To create adversarial splits, we used discovered tasks for the CIFAR-10 dataset (see Appendix F) and tasks corresponding to randomly-initialized networks for ImageNet and CelebA (see Sec. **??**). The test performance drops significantly when using adversarial splits.

tasks. Fig. 11-left shows that the similarity increases with scaling the embedding size but remains relatively low for most human-labelled tasks for the scale at which we ran our experiments.

F Adversarial Splits for Multi-Class Classification.

In this section we show one way to extend the adversarial splits proposed in Sec. 6 to a multi-way classification target task. Let us consider a multi-class classification task $\tau : X \to C$, where $C = \{1, \dots, K\}$, for which we want to construct an adversarial split. We cannot create a "complete" correlation between τ and a binary discovered task similar to Eq. 6 as they have different numbers of classes. On the other hand, using a K-way discovered task will result in having too few images in the corresponding train set (where $\tau_{\mathrm{d}}(x) = \tau(x)$). Instead, we suggest creating only a partial correlation between τ and a binary discovered task τ_{d}. To do that, we split the set of all classes C into two disjoint sets C_1, C_2 and create the an adversarial split in the following way:

$$X_{\mathrm{tr}}^{\mathrm{adv}} = \{x \,|\, \mathbb{I}[\tau(x) \in C_{\mathbf{1}}] = \tau_{\mathrm{d}}(x),\, x \in X\}, \;\; X_{\mathrm{te}}^{\mathrm{adv}} = \{x \,|\, \mathbb{I}[\tau(x) \in C_{\mathbf{2}}] = \tau_{\mathrm{d}}(x),\, x \in X\}, \quad (7)$$

In this case, $\tau_{\mathrm{d}}(x)$ does not equate $\tau(x)$ on $X_{\mathrm{tr}}^{\mathrm{adv}}$, but it is predictive of whether $\tau(x) \in C_1$ or $\tau(x) \in C_2$, creating a partial correlation. Intuitively, if τ_{d} provides a good learning signal, the network can learn to distinguish between C_1 and C_2 first and then perform classification inside each set of classes. In this the case, the test set will fool the network as it has the correlation opposite to the train set.

Tab. 1 shows results of attacking the original 10-way semantic classification task from CIFAR-10. We can see that the proposed adversarial splits based on the discovered high-AS tasks generalize well to the multi-class classification case, i.e., the accuracy drops noticeably compared to random splits. Note that the network, in this case, is trained on only half the CIFAR-10 original training set. Hence, the accuracy is 0.8 for a random split as opposed to around 0.9 when trained on the full dataset.

G Adversarial Splits for ImageNet and CelebA via Random-Network Task.

We demonstrate the effectiveness of the proposed approach and create adversarial splits for ImageNet [13] and CelebA [48], a popular dataset for research on spurious correlations. To get a high-AS task, we utilize the phenomenon observed in Sec. 4.4 and use a randomly initialized network as τ_{d}. We use ResNet-50 for ImageNet and ResNet-18 for CelebA. We do not use augmentations for training. Both adversarial and random splits partition the original dataset equally into train and test, and, therefore, the networks are trained using half of the original training set. Tab. 1 shows that adversarial splits created with the task corresponding to a randomly initialized network lead to a significant accuracy drop compared to random train-test splits. See Appendix **??** for visualization.

H λ Hyperparameter: Trade-off between the Agreement Score and Similarity

The task discovery optimization objective (Eq. 3) has two terms: 1) average agreement score over the set of discovered tasks T and 2) similarity loss, measuring how similar tasks are to each other. The hyperparameter λ controls the balance between these two losses. If $\lambda = 0$, a trivial solution would be to have all the tasks the same with the highest possible AS. On the other hand, when $\lambda \to \infty$, tasks in T will be dissimilar to each other with lower average AS (since the number of high-AS tasks is limited). One can tune the hyperparameter to fit a specific goal based on whether many different tasks are needed or a few but with the highest agreement score.

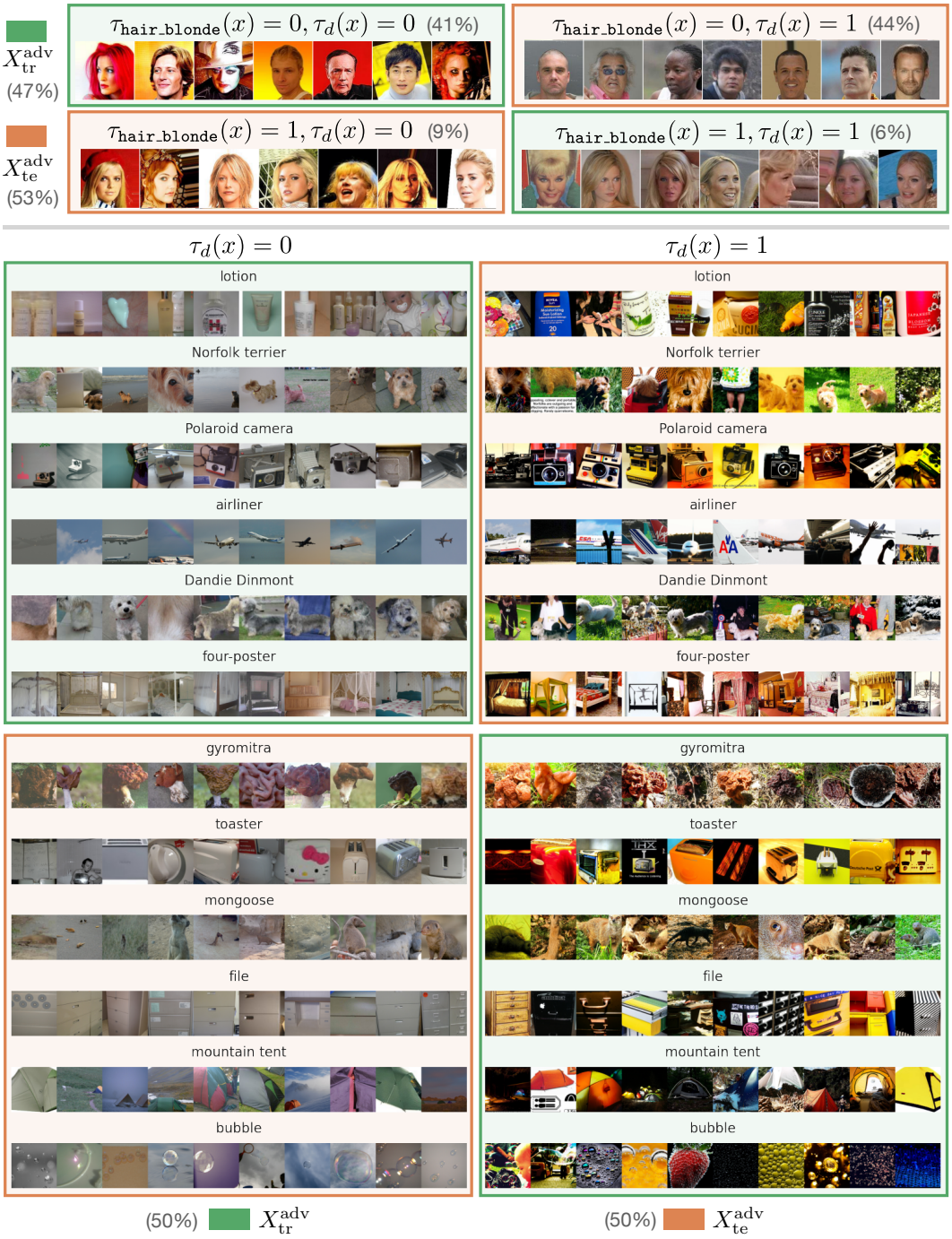

Figure 12: Visualization of adversarial train-test splits for CelebA (top) and ImageNet (bottom) datasets. To construct these splits we use high-AS task corresponding to a randomly initialized network (see Sec. 4.4 and Sec. K).

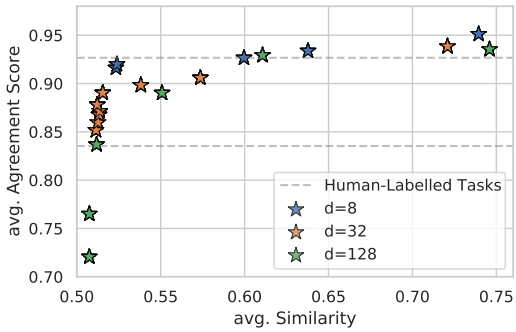

Figure 13: The Trade-off between the Agreement Score and Similarity induced by the λ hyperparameter. Each dot corresponds to a set of task T discovered with the corresponding embedding space dimensionality and parameter λ (dots with higher similarity have higher value of λ). The x coordinate is computed by measuring the maximum similarity for each task in T to the tasks from the same set and averaging these values, i.e., $1/|T| \sum_{t \in T} \max_{t' \in T} \text{sim}(t, t')$. The y coordinate is the average AS over tasks from T. We can see that the method is not very sensitive to λ (for the considered dimensionalities) and most runs have high average AS even when similarity is low.

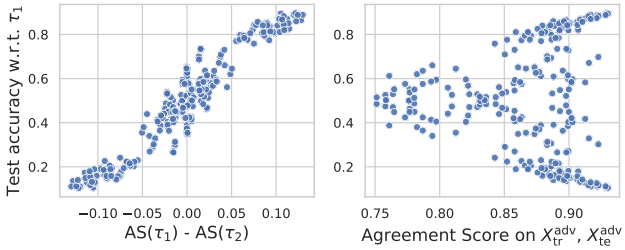

Figure 14: ***Left:*** Each dot corresponds to a pair of tasks (τ_1, τ_2). The *y-axis* is the test accuracy on $\mathcal{D}(X_{\text{te}}^{\text{adv}}, \tau_1)$ after training on the corresponding adversarial split $\mathcal{D}(X_{\text{tr}}^{\text{adv}}, \tau_1)$. The *x-axis* is the difference in the agreement score of two tasks averaged over random train-test splits. This is the same plot as in Fig. 5-right. ***Center:*** The same data but with *x-axis* being the AS measured on the adversarial split, i.e., $\text{AS}(\tau_1; X_{\text{tr}}^{\text{adv}}, X_{\text{te}}^{\text{adv}})$ $(= \text{AS}(\tau_2; X_{\text{tr}}^{\text{adv}}, X_{\text{te}}^{\text{adv}}))$. The symmetry on the plots is due to the interchangeable role of τ_1 and τ_2 as they are binary tasks, which only changes the value of the test accuracy ($\text{acc} \to 1 - \text{acc}$). We see that even when the test accuracy is close to 0.5 w.r.t. to both τ_1 and τ_2, the AS stays high in most cases, suggesting that networks generalize well when trained on $\mathcal{D}(X_{\text{tr}}^{\text{adv}}, \tau_1)$ but w.r.t. other labels on test images from $X_{\text{te}}^{\text{adv}}$ (see Proposition 1).

Fig. 13shows how such a trade-off looks in practice for the shared embedding formulation with different dimensionalities of the embedding space. We can see that, in fact, the proposed method for task discovery is not very sensitive to the choice of λ as the AS mostly stays high even when the similarity is close to its minimum value.

I Can Networks Trained on Adversarial Splits Generalize?

In Sec. 6 of the main paper, we introduce adversarial train-test splits that "fool" a network, significantly reducing the test accuracy compared to randomly sampled splits. However, it remains unclear whether networks can potentially exhibit generalization when trained on these adversarial splits w.r.t. other test labels or if the behaviour is similar to training on a random-labelled task, i.e., different networks converge to different solutions. In order to rule out the latter hypothesis, we measure the AS of a task on the adversarial train-test split and show that it stays high, i.e., the task stays generalizable (in the view of Proposition 1).

Recall that for a pair of tasks (τ_1, τ_2), the adversarial split is defined as follows:

$$X_{\text{tr}}^{\text{adv}} = \{x \,|\, \tau_1(x) = \tau_2(x),\, x \in X\}, \quad X_{\text{te}}^{\text{adv}} = \{x \,|\, \tau_1(x) \neq \tau_2(x),\, x \in X\}, \tag{8}$$

the only difference with the definition from Sec. 6, Eq. 6, is that here we consider a general case with τ_1, τ_2 being any two tasks not only a human-labelled or discovered. In Sec. 6, we show that the test accuracy on $\mathcal{D}(X_{\text{tr}}^{\text{adv}}, \tau_1)$ seems to correlate well with the difference in agreement scores between two tasks (see Fig. 14). In particular, when the agreement scores are close to each other, i.e., the difference is close to zero, the test accuracy w.r.t. both tasks is close to 0.5. We further investigate if networks, in this case, can generalize well w.r.t. some other labelling of the test set $X_{\text{te}}^{\text{adv}}$, i.e., whether the agreement score on this split is high.

Fig. 14-right shows that even when the test accuracy is close to 0.5, the AS measured on the adversarial split, i.e., $\text{AS}(\tau_1; X_{\text{tr}}^{\text{adv}}, X_{\text{te}}^{\text{adv}})$, stays high in most cases. This observation means that networks trained on $\mathcal{D}(X_{\text{tr}}^{\text{adv}}, \tau_1)$ do generalize well in the sense of Proposition 1. That is, they converge to a consistent solution, which, however, differs from both τ_1 and τ_2.

J The Connection between the Agreement Score and Test Accuracy

In this section, we establish the connection between the test accuracy and the agreement score for the fixed train-test split $X_{\text{tr}}, X_{\text{te}}$, mentioned in Sec. 3.1. We show that the agreement score provides a lower bound on the "best-case" test accuracy but cannot predict test accuracy in general.

For a task τ and a train-test split $X_{\text{tr}}, X_{\text{te}}$, we consider the test accuracy to be the accuracy on the test set $\mathcal{D}(X_{\text{te}}, \tau)$ averaged over multiple models trained on the same training dataset $\mathcal{D}(X_{\text{tr}}, \tau)$:

$$\text{ACC}(\tau; X_{\text{tr}}, X_{\text{te}}) = \mathbb{E}_{w \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} \mathbb{E}_{x \sim X_{\text{te}}} [f(x; w) = \tau(x)], \tag{9}$$

where $\mathbb{E}_{x \sim X_{\text{te}}}$ stands for averaging over the test set. We will further omit $X_{\text{tr}}, X_{\text{te}}$ and simply use $\text{ACC}(\tau)$ since the split remains fixed. Recall that for a learning algorithm $\mathcal{A}$, we define the agreement score (AS) as follows (see Sec. 3.1):

$$\text{AS}(\tau; X_{\text{tr}}, X_{\text{te}}) = \mathbb{E}_{w_1, w_2 \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} \mathbb{E}_{x \sim X_{\text{te}}} [f(x; w_1) = f(x; w_2)]. \tag{10}$$

Intuitively, when the agreement score is high, models trained on $\mathcal{D}(X_{\text{tr}}, \tau)$ make similar predictions, and, therefore, there should be some labelling of test data X_{te}, such that the accuracy w.r.t. these labels is high. We formalize this intuition in the following proposition.

Proposition 1. *For a given train-test split $X_{\text{tr}}, X_{\text{te}}$, a task defined on the training set $\tau : X_{\text{tr}} \to \{0, 1\}$ and a learning algorithm $\mathcal{A}$, the following inequalities hold for any $h : X_{\text{te}} \to \{0, 1\}$:*

$$\text{ACC}(\hat{\tau}) \geq \text{AS}(\tau) \geq 2\text{ACC}(\tilde{\tau}) - 1, \tag{11}$$

where

$$\hat{\tau} = \begin{cases} \tau(x), & x \in X_{\text{tr}} \\ \arg\max_c \mathbb{E}_{w \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} [f(x; w) = c], & x \in X_{\text{te}} \end{cases},$$

and

$$\tilde{\tau} = \begin{cases} \tau(x), & x \in X_{\text{tr}} \\ h(x), & x \in X_{\text{te}} \end{cases}.$$

Proof. 1) Let us, first, show that the first inequality holds true.

$$\begin{aligned} \text{ACC}(\hat{\tau}) &= \mathbb{E}_{w \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} \mathbb{E}_{x \sim X_{\text{te}}} [f(x; w) = \hat{\tau}(x)] \\ &= \mathbb{E}_{w \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} \mathbb{E}_{x \sim X_{\text{te}}} [f(x; w) = \arg\max_c \mathbb{E}_{\bar{w} \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} [f(x; \bar{w}) = c]] \\ &= \mathbb{E}_{x \sim X_{\text{te}}} \mathbb{E}_{w \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} [f(x; w) = \arg\max_c \mathbb{E}_{\bar{w} \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} [f(x; \bar{w}) = c]] \\ &= \mathbb{E}_{x \sim X_{\text{te}}} \max_c \mathbb{E}_{w \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} [f(x; w) = c] \end{aligned}$$

Due to the maximum, for any $h : X_{\text{te}} \to \{0, 1\}$, the following holds for any $x \in X_{\text{te}}$:

$$\max_c \mathbb{E}_{w \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} [f(x; w) = c] \geq \mathbb{E}_{w \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} [f(x; w) = h(x)],$$

and, therefore:

$$\mathbb{E}_{x \sim X_{\text{te}}} \max_c \mathbb{E}_{w \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} [f(x; w) = c] \geq \mathbb{E}_{x \sim X_{\text{te}}} \mathbb{E}_{w \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))} [f(x; w) = h(x)]. \tag{12}$$

In particular, Eq. 12 holds for $f(\cdot\,; w_2)$ for any $w_2 \sim \mathcal{A}(\mathcal{D}(X_{\text{tr}}, \tau))$, and, therefore:

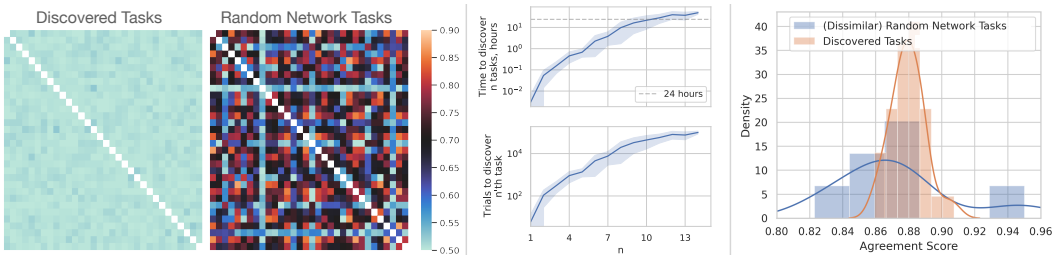

Figure 15: ***Left:*** The pairwise similarity matrix between 32 tasks from the set of discovered and random network tasks. **We see that random network tasks are much more similiar to each other.** ***Center:*** Sampling random networks to discover dissimilar tasks. x-axis: the discovered task number, y-axis: hours to discover n tasks (top) and the number of generated random networks to obtain nth dissimilar task (bottom). We use the similarity threshold 0.55 for this experiment. The standard deviation is over 5 runs with different initial seeds. ***Right:*** The distribution of the AS for the set of the found (dissimilar) random network tasks and discovered tasks from Sec. 4.4. We see that while random networks give rise to high-AS tasks, we cannot control for their similarity and, hence, it is and inefficient task discovery framework. Also, the AS drops a bit compared to tasks discovered with the proposed task discovery framework.

$$
\begin{aligned}
\mathrm{ACC}(\hat{\tau}) &\geq \mathbb{E}_{x\sim X_{\mathrm{te}}}\mathbb{E}_{w\sim\mathcal{A}(\mathcal{D}(X_{\mathrm{tr}},\tau))}[f(x;w)=f(x;w_2)] \\
\mathbb{E}_{w_2\sim\mathcal{A}(\mathcal{D}(X_{\mathrm{tr}},\tau))}\mathrm{ACC}(\hat{\tau}) &\geq \mathbb{E}_{w_2\sim\mathcal{A}(\mathcal{D}(X_{\mathrm{tr}},\tau))}\mathbb{E}_{x\sim X_{\mathrm{te}}}\mathbb{E}_{w\sim\mathcal{A}(\mathcal{D}(X_{\mathrm{tr}},\tau))}[f(x;w)=f(x;w_2)] \\
\mathrm{ACC}(\hat{\tau}) &\geq \mathbb{E}_{w,w_2\sim\mathcal{A}(\mathcal{D}(X_{\mathrm{tr}},\tau))}\mathbb{E}_{x\sim X_{\mathrm{te}}}[f(x;w)=f(x;w_2)] \\
\mathrm{ACC}(\hat{\tau}) &\geq \mathrm{AS}(\tau)\ [=\mathrm{AS}(\hat{\tau})]
\end{aligned}
$$

2) Note that $\forall a,b,c\in\{0,1\}$ holds $[a=b]\geq[a=c]+[b=c]-1$. Then:

$$
\begin{aligned}
\mathrm{AS}(\tau) &= \mathbb{E}_{w_1,w_2\sim\mathcal{A}(\mathcal{D}(X_{\mathrm{tr}},\tau))}\mathbb{E}_{x\sim X_{\mathrm{te}}}[f(x;w_1)=f(x;w_2)] \\
&\geq \mathbb{E}_{w_1,w_2\sim\mathcal{A}(\mathcal{D}(X_{\mathrm{tr}},\tau))}\mathbb{E}_{x\sim X_{\mathrm{te}}}[f(x;w_1)=\tilde{\tau}]+[\tilde{\tau}=f(x;w_2)]-1 \\
&\geq \mathbb{E}_{x\sim X_{\mathrm{te}}}\left(\mathbb{E}_{w_1\sim\mathcal{A}(\mathcal{D}(X_{\mathrm{tr}},\tau))}[f(x;w_1)=\tilde{\tau}]+\mathbb{E}_{w_2\sim\mathcal{A}(\mathcal{D}(X_{\mathrm{tr}},\tau))}[\tilde{\tau}=f(x;w_2)]\right)-1 \\
&= \mathrm{ACC}(\tilde{\tau})+\mathrm{ACC}(\tilde{\tau})-1 \\
&= 2\mathrm{ACC}(\tilde{\tau})-1
\end{aligned}
$$

□

Proposition 1 establishes a connection between the AS and the test accuracy measured on the same test set X_{te}. It can be easily seen from the first inequality in Eq. 11 that the AS provides only a *necessary* condition for high test accuracy, and, therefore, AS cannot be predictive of the test accuracy in general [29] as was also noted in [35]. Indeed, test accuracy will be high if test labels are in accordance with $\hat{\tau}$ and low if, for example, they correspond to $1-\hat{\tau}$.

As can also be seen from the second inequality in Eq. 11, one can directly optimize the $\mathrm{ACC}(\tilde{\tau})$ w.r.t. $\tilde{\tau}$ (both the training and test labels) to find a generalizable task. However, compared to optimizing the AS, this would additionally require modelling test labels of the task on X_{te}, whereas AS only requires training labels on X_{tr} and, hence, has fewer parameters to optimize.

K On Random Networks as high-AS Tasks

In Sec. 4.4 of the main paper, we found that the AS of a randomly initialized network is high. This section provides more details about this experiment and additional discussion of the results.

K.1 Constructing the Task Corresponding to a Randomly Initialized Network

When initializing the network randomly and taking the labels after the softmax layer, the corresponding task is usually (as found empirically) unbalanced with all labels being either 0 or 1, which trivially is a high-AS task but not of interest. We, therefore, apply a slightly different procedure to construct a

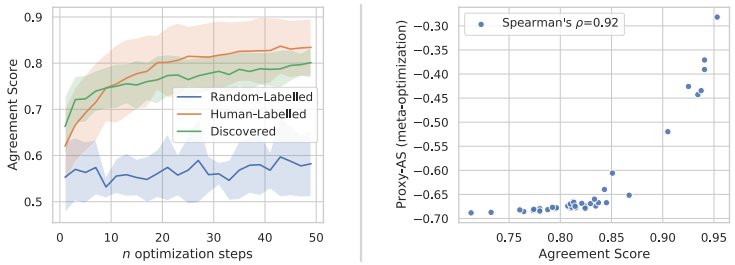

Figure 16: Agreement score approximation for meta-optimization. ***Left:*** The agreement score at every iteration of the inner-loop optimization of two networks for random-labelled, human-labelled and discovered tasks. Standard deviation for each group is over 5 tasks and 3 seeds (accounts for initialization and data-order). We can see that the differentiation between high-AS (human-labelled and discovered) tasks and random-labelled occurs early in training and the AS after only 50 steps can be reliably used for task discovery. ***Right:*** The dependence between the proxy-AS (50 steps/soft labels/(negative)cross-entropy agreement/SGD) and the original Agreement Score (100 epochs/hard labels/0-1 agreement/Adam). We see that the two correlate well with Spearman's rank correlation of 0.92 making it a good optimization proxy.

task corresponding to a randomly initialized network. First, we collect the logits of the randomly initialized network for each image and sort them. Then, we split images evenly into two classes according to this ordering, which results in the corresponding labelling from a *random network task*. Fig. 15-right shows the AS for a set of such tasks (we discuss how we construct this set in the next Sec. K.2). We can see that tasks corresponding to randomly initialized networks indeed have high AS. We use the same ResNet-18 architecture with the same initialization scheme as for measuring the AS (see Sec. L).

K.2 Drawing Random Networks as a Naive Task Discovery Method

Based on the observation made in Sec. K.1, one could suggest drawing random networks as a naive method for task discovery. The major problem, however, is that two random network tasks are likely to have high similarity in terms of their labels (see Fig. 15-left), whereas in task discovery we want to find a set of more diverse tasks. A simple modification to do so is to sequentially generate random network tasks as described in Sec. K.1, and keep the task if its maximum similarity to previously saved tasks is below a certain threshold. We perform this procedure with the threshold of 0.55. Fig. 15-center shows that this naive discovery method is inefficient. For comparison, our proposed task discovery from Sec. 4.4 takes approximately 24 hours to train. Also, we can see that the AS of (dissimilar) random network tasks seems to deteriorate compared to the set of tasks discovered via meta-optimization.

L Experimental Setup For Task Discovery

In this section, we provide more technical details on the proposed task discovery framework, including a discussion on a differentiable version of the AS in Sec. L.1, meta-optimization setting in Sec. L.2 and the description of architectures in Sec. L.3.

L.1 Modifying the AS for Meta-Optimization

In order to be able to optimize the AS defined in Sec. 3.1 we need to make it differentiable and feasible, i.e., reduce its computational and memory costs. We achieve this by applying the following changes to the original AS:

- We use (negative) cross-entropy loss instead of the 0-1 loss to measure the agreement between two networks after training.
- To train two networks in the inner-loop, we use "soft" labels of the task-network t_θ, i.e., probabilities, instead of "hard" $\{0, 1\}$ labels. Since the task networks influence the inner-loop through the training labels, we can backpropagate through these "soft" labels to the parameters θ.

- We use SGD instead of Adam optimizer in the inner-loop as it does not require storing additional momentum parameters and, hence, has lower memory cost and is more stable in the meta-optimization setting.
- We limit the number of inner-loop optimization steps to 50.

We refer to the corresponding AS with these modifications as *proxy-AS*. In this section, we validate whether the proxy-AS is a good objective to optimize the original AS.

First, we verify that 50 steps are enough to provide the differentiation between human-labelled and random-labelled tasks. Fig. 16-left shows that the differentiation between high-AS human-labelled and discovered tasks and random-labelled tasks occurs early in training, and 50 steps are enough to capture the difference. The results of task discovery also support this as the found tasks have high AS when two networks are trained till convergence for 100 epochs (see Fig. 1of the main paper). Further, Fig. 16-right shows that the proxy-AS (with all modifications applied) correlates well with the original AS and, therefore, makes for a good optimization proxy.

L.2 Task Discovery Meta-Optimization Details

Algorithm 1 Pseudo-Code for Task Discovery with Shared Embedding Space

initialize the task-encoder weights θ_e^0
initialize orthogonal task-specific linear heads $\{\theta_l^i\}_{i=1}^d$
for t in $0, \dots, T-1$ **do** ▷ outer-loop
  $\theta_l \sim \{\theta_l^i\}_{i=1}^d$ ▷ sample a task-specific head
  $w_1^0, w_2^0 \leftarrow p(w)$ ▷ initialize two networks weights randomly
  $L_{\text{AS}}(\theta_e^t), L_{\text{unif}}(\theta_e^t) \leftarrow 0, 0$
  for k in $0, \dots, K-1$ **do** ▷ inner-loop (assume less than 1 epoch)
    sample a training batch $[x_1, \dots, x_M]$
    $L_{\text{AS}}(\theta_e^t) \leftarrow L_{\text{AS}}(\theta_e^t) + \frac{1}{M}\sum_{i=1}^M l_{\text{ce}}(f(x_i; w_1^k), f(x_i; w_2^k))$ ▷ x_is are novel for fs
    $h_i \leftarrow e(x_i;\, \theta_e^t),\ i = 1, \dots, M$
    $L_{\text{unif}}(\theta_e^t) \leftarrow L_{\text{unif}}(\theta_e^t) + \mathcal{L}_{\text{unif}}(h)$
    $t_i \leftarrow \sigma(\theta_l^\top h_i),\ i = 1, \dots, M$ ▷ get task "soft" labels
    $w_1^{k+1} \leftarrow w_1^k - \alpha \cdot \nabla_w \frac{1}{M}\sum_{i=1}^M l_{\text{ce}}(f(x_i; w_1^k), t_i)$
    $w_2^{k+1} \leftarrow w_2^k - \alpha \cdot \nabla_w \frac{1}{M}\sum_{i=1}^M l_{\text{ce}}(f(x_i; w_2^k), t_i)$
  end for
  $\theta_e^{t+1} \leftarrow \theta_e^t - \eta \cdot \nabla_{\theta_e}(L_{\text{AS}}(\theta_e^t) + L_{\text{unif}}(\theta_e^t))$ ▷ use accumulated AS and uniformity losses
end for
return $\theta_e^T, \{\theta_l^i\}_{i=1}^d$

For all task discovery experiments, we use the following setup. We use the same architecture as the backbone for both the networks $f(\cdot\,; w)$ in the inner-loop and the task encoder $e(\cdot\,; \theta_e)$, which has an additional linear projection layer to $\mathbb{R}^d$. To model tasks, we use d predefined orthogonal hyper-planes $\{\theta_l^i\}_{i=1}^d$, which remain fixed throughout the training. Then at each meta-optimization step, we sample a task corresponding to one of these d hyper-planes θ_l and update the encoder by using the gradient of the AS w.r.t. encoder parameters: $\nabla_{\theta_e}\text{AS}(t_{(\theta_e,\theta_l)})$. We found it to be enough to optimize the AS w.r.t. these tasks corresponding to the d predefined hyper-planes to train the encoder for which *any* randomly sampled hyper-plane $\tilde{\theta}_l$ gives rise to a task with high AS.

We accumulate the uniformity loss (Sec. 4.3, Eq. 5) for embeddings of all images passed through the encoder during the inner-loop and backpropagate through it once at each meta-optimization step. We set the weight λ of the uniformity loss for the main setting with $d = 32$ to 0.66 for ResNet-18, 5 for MLP and 100 for ViT. These weights were chosen by trying multiple values and ensuring that similarities between 32 discovered tasks for each architecture are approximately the same between 0.5 and 0.55. We use the SGD optimizer with `lr=1e-2` and batch size 512 for the inner-loop and Adam with `lr=1e-3` as the meta-optimizer for the outer-loop.

To further speed up the task discovery framework, we early-stop the inner-loop optimization when the AS between two networks is above 0.6 for more than three consecutive steps. Also, since we limit the number of steps to 50, every training batch in the inner-loop is novel for two networks and can be

Module name	Module	in	out
Layer 1	Linear($3072, 1024$), ReLU, BN(1024)	3072	1024
Layer 2	Linear($1024, 512$), ReLU, BN(512)	1024	512
Layer 3	Linear($512, 256$), ReLU, BN(256)	512	256
Layer 4	Linear($256, 64$), ReLU	256	64
Classifier	Linear($64, 2$)	64	2

Table 2: MLP Architecture

seen as validation data. We utilize this and accumulate the proxy-AS between two networks on these training batches (before applying the gradient update on them) and use it as the final proxy-AS for backpropagation and updating the encoder. See Algorithm 1.

L.3 Architectural Details

In all cases, we initialize models' weights w_1, w_2 with Kaiming uniform initialization [23], which is the default initialization in PyTorch [63].

For the MLP, we use the 4-layer perceptron with batch normalization (see Tab. 2).

We use standard ResNet18 [24] architecture adapted for CIFAR10 dataset: the first 7x7 conv with stride=2 is replaced by 3x3 conv with stride=1.

We use ViT architecture in the following way. We split an image onto 4x4 patches. We remove dropout and replace all LayerNorm layers with BatchNorm to achieve faster convergence. We reduce the embedding dimensionality and the MLP hidden layers' dimensionality from 512 to 256 and utilize only six transformer blocks to fit the model into memory for task discovery. After the transformer, we feed class embedding to a linear classifier.