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%

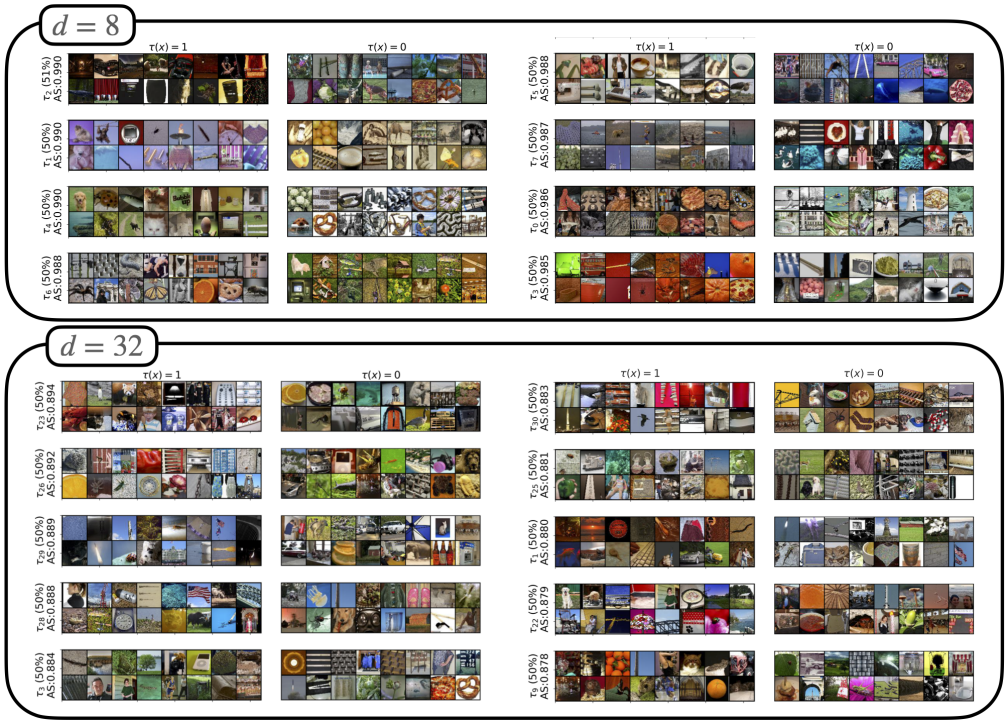

Figure 6: Tasks Visualization for TinyImageNet for $d = \{8, 32\}$. The top 10 tasks for each d, as measured by AS is shown (if $d < 10$, then d tasks are selected). Each column and row shows selected images from a task, for class 1 ($\tau(x) = 1$) and class 0 ($\tau(x) = 0$). In the y-axis, we show 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.

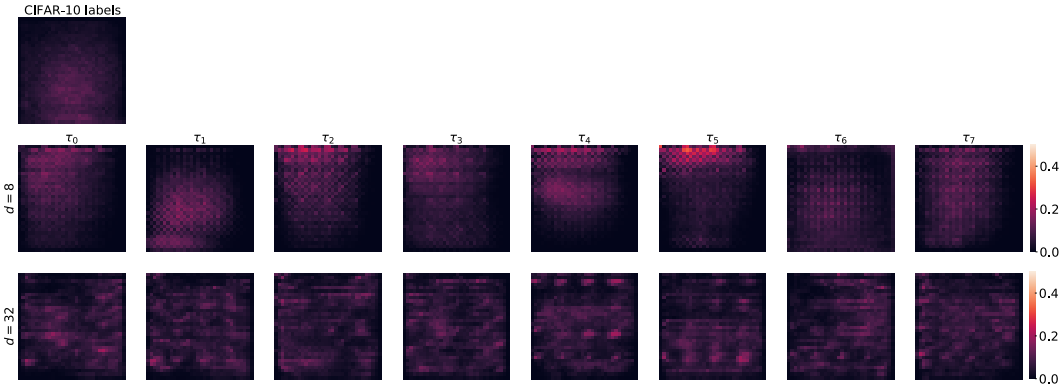

Figure 7: Sufficient input subsets for tasks networks for $d = \{8, 32\}$**.** Each plot shows the heatmap of the frequency that each pixel was selected to be in the sufficient subset. The task network with $d = 8$ seems to attend to more macro features in the image, while with $d = 32$, it seems to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

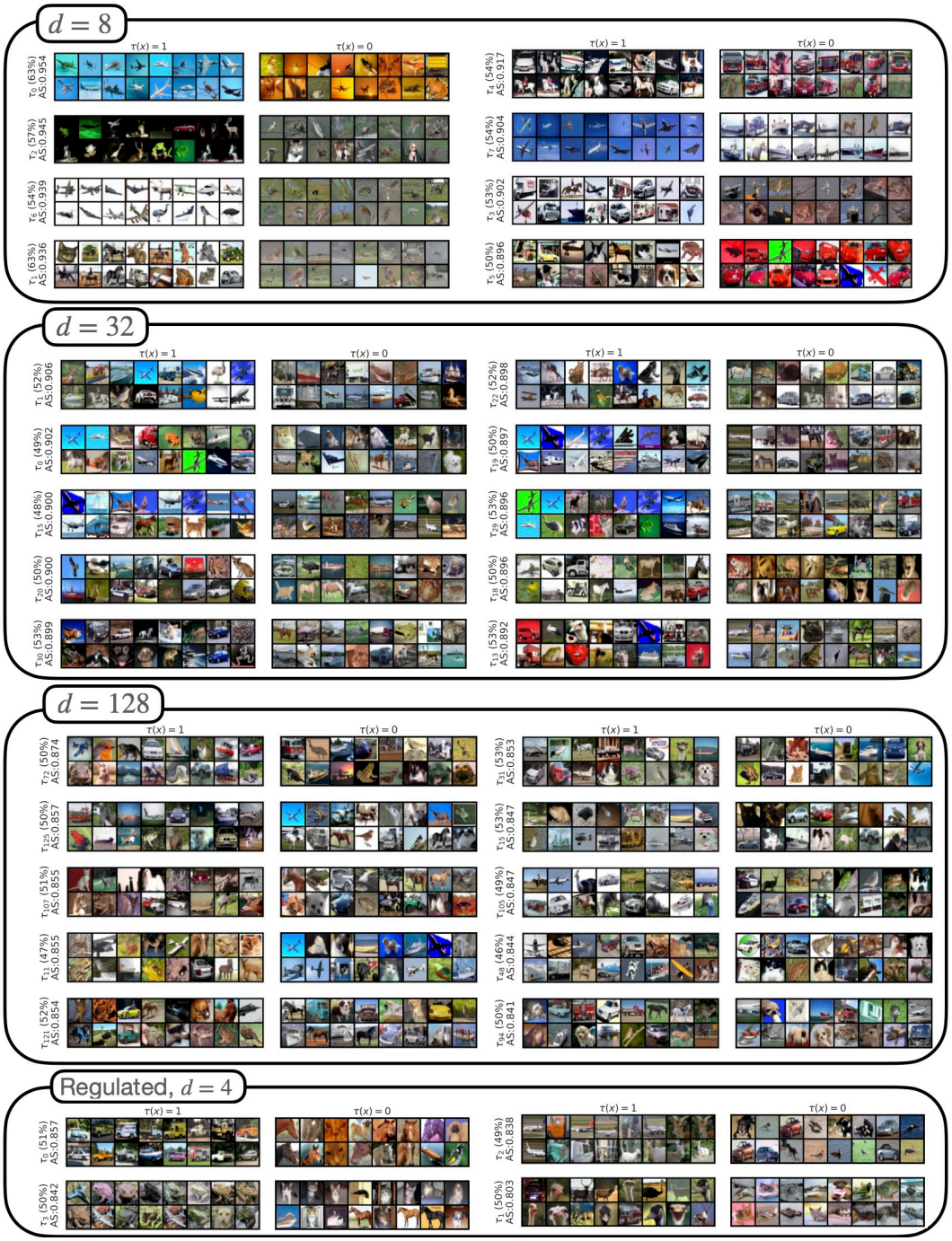

Figure 8: Tasks visualizations for $d = \{8, 32, 128\}$ and the regulated version of task discovery. The top 10 tasks for each d, as measured by AS is shown (if $d < 10$, then d tasks are selected). Each column and row shows selected images from a task, for class 1 ($\tau(x) = 1$) and class 0 ($\tau(x) = 0$). 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.

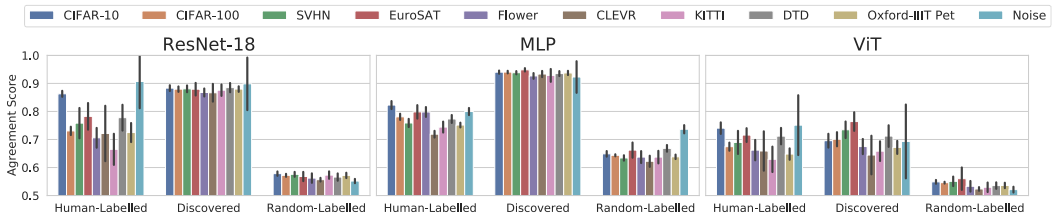

Figure 9: The Dependence of the Agreement Score on the Test Data Domain. We fix training images X_{tr} to be CIFAR-10 and vary X_{te} to measure the AS. For each architecture, we take five tasks from each set of human-labelled, random-labelled and discovered (for the same architecture) tasks and measure 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 stable and stays high across all the domains, whereas the AS of human-labelled tasks is more volatile.

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

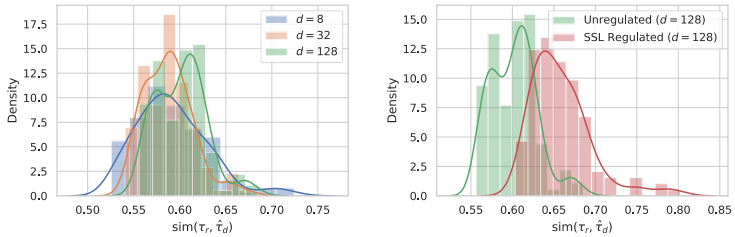

Figure 11: Are human-labelled tasks present in the set of discovered tasks? For each available human-labelled task $\tau \in T_{\mathrm{HL}}$, we find the most similar discovered task $\hat{\tau}_d \in T_{\theta^*_e}$ and plot the corresponding similarities $\mathrm{sim}(\tau_{\mathrm{hl}}, \hat{\tau}_d)$ 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 Appendix G). 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:

$$X^{\mathrm{adv}}_{\mathrm{tr}} = \{x \,|\, \mathbb{I}[\tau(x) \in C_1] = \tau_{\mathrm{d}}(x),\, x \in X\}, \quad X^{\mathrm{adv}}_{\mathrm{te}} = \{x \,|\, \mathbb{I}[\tau(x) \in C_2] = \tau_{\mathrm{d}}(x),\, x \in X\}, \quad (7)$$

In this case, $\tau_{\mathrm{d}}(x)$ does not equate $\tau(x)$ on $X^{\mathrm{adv}}_{\mathrm{tr}}$, 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

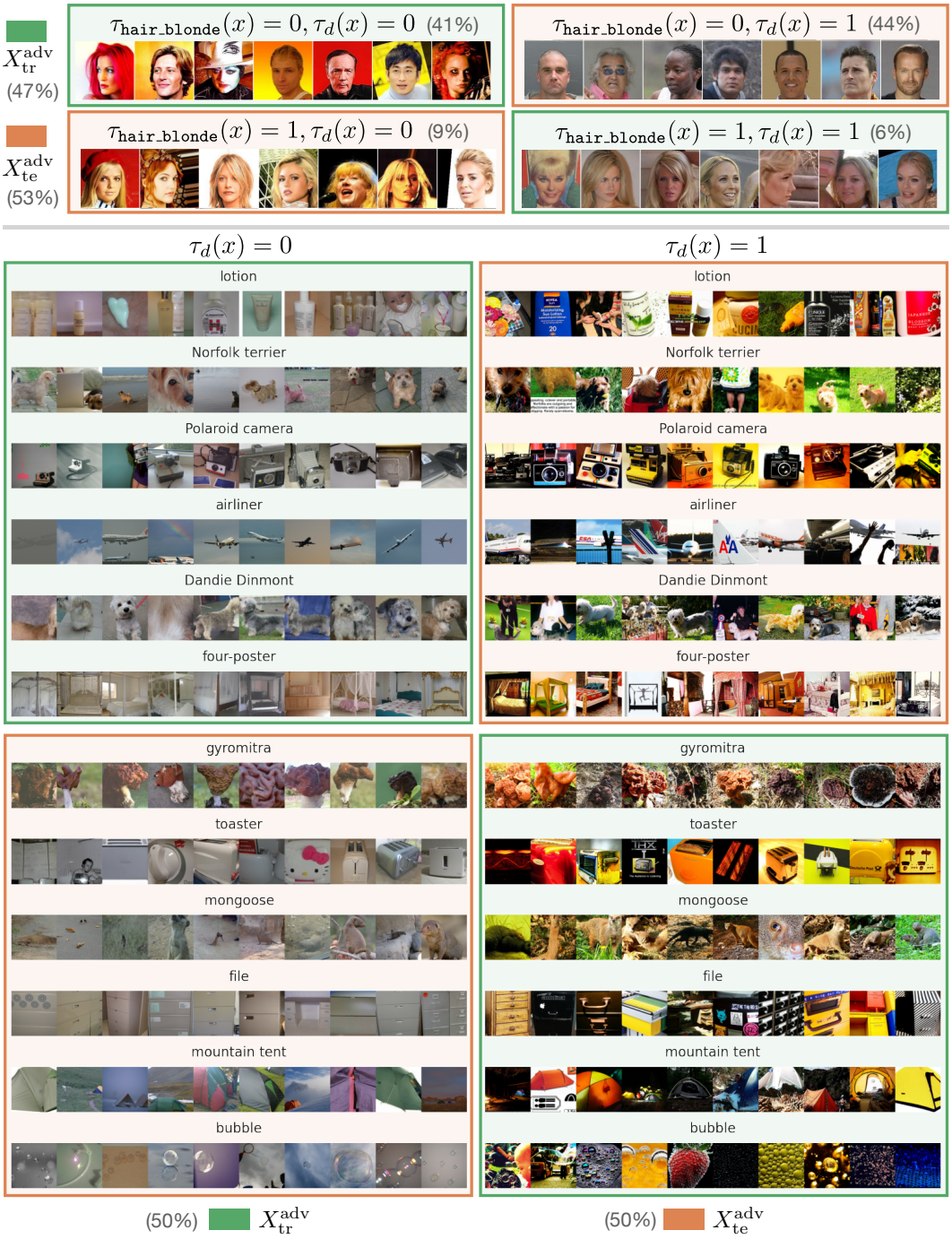

Figure 12: Visualization of adversarial train-test splits for CelebA (top) and ImageNet (bottom) datasets. To construct these splits we use high-AS task corresponding to a randomly initialized network (see Sec. 4.4 and Sec. K).

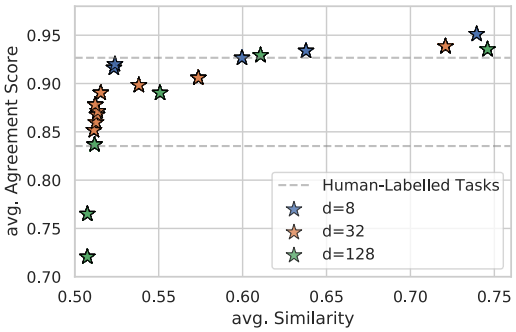

Figure 13: The Trade-off between the Agreement Score and Similarity induced by the λ hyperparameter. Each dot corresponds to a set of task T discovered with the corresponding embedding space dimensionality and parameter λ (dots with higher similarity have higher value of λ). The x coordinate is computed by measuring the maximum similarity for each task in T to the tasks from the same set and averaging these values, i.e., $1/|T| \sum_{t \in T} \max_{t' \in T} \mathrm{sim}(t, t')$. 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.

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.

Fig. 13 shows 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:

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

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 $\mathcal{D}(X_{\mathrm{tr}}^{\mathrm{adv}}, \tau_1)$ 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

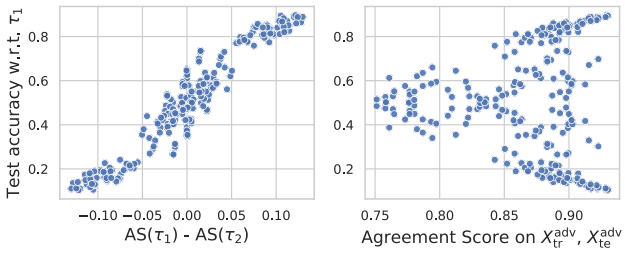

Figure 14: ***Left:*** Each dot corresponds to a pair of tasks (τ_1, τ_2). The *y-axis* is the test accuracy on $\mathcal{D}(X_{\text{te}}^{\text{adv}}, \tau_1)$ after training on the corresponding adversarial split $\mathcal{D}(X_{\text{tr}}^{\text{adv}}, \tau_1)$. The *x-axis* is the difference in 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).

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.

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

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

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

and, therefore:

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

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

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

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 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).

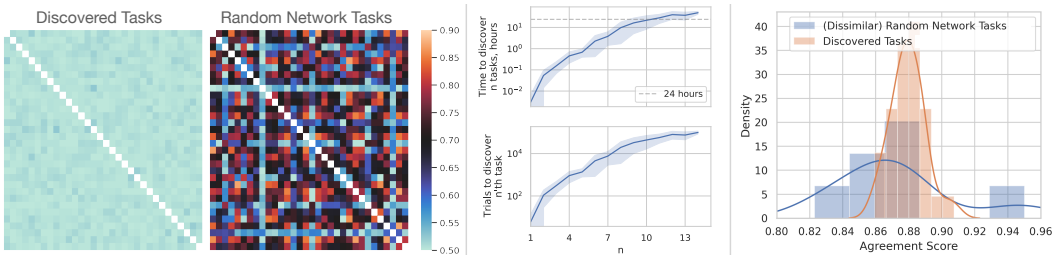

Figure 15: ***Left:*** The pairwise similarity matrix between 32 tasks from the set of discovered and random network tasks. **We see that random network tasks are much more similiar to each other.** ***Center:*** Sampling random networks to discover dissimilar tasks. x-axis: the discovered task number, 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.

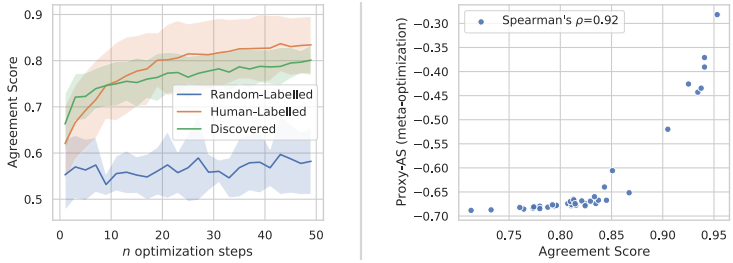

Figure 16: Agreement score approximation for meta-optimization. ***Left:*** The agreement score at every iteration of the inner-loop optimization of two networks for random-labelled, human-labelled and discovered tasks. Standard deviation for each group is over 5 tasks and 3 seeds (accounts for initialization 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.

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

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

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 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.