Self-Supervised Task Discovery

Elias Wang
Department of Electrical Engineering
Stanford University
elias.wang@stanford.edu

Abstract

While large datasets, such as ImageNet, have propelled the performance of deep neural networks, they are expensive and time-consuming to curate. This has motivated the interest in low-cost self-supervised tasks that can generate general-purpose features, e.g. that support ImageNet classification, for the case of vision, without requiring manually labeled data. We propose a framework for automatically discovering a self-supervised task that results in powerful visual features. Our approach uses a model, the proposer, to generate the parameters that define the task. Training a network, the solver, to solve this task results in metrics about how good the task is. Using this metric as a reward signal, we can then update the proposer to generate better and better tasks. We apply our method to rotation prediction, where the task is to recognize which rotation angle, from a set, was applied to an image, and demonstrate that the algorithm is able to find a set of angles that achieve comparable performance to state-of-the-art self-supervised methods.

1. Introduction

In recent years, there is rapid progress in computer vision with the use of deep convolutional neural networks [18] (CNNs) for learning powerful image representations. These CNNs are typically trained on a massive amount of manually labeled data in order to learn high-level visual features that transfer to a variety of tasks, such as object detection [9], semantic segmentation [20], or image captioning [13]. However, this fully supervised method is limited by the reliance on manually labeled data, which is expensive and time-consuming to collect at scale.

This problem has sparked increased interest in developing methods to learn high-level visual features without manual annotation of data. Some examples of these unsupervised methods include, clustering based methods ([8], [19]), the reconstruction based methods ([4], [21]), and generative probabilistic models ([10], [7]). Among them, is the self-supervised learning paradigm that defines a pretext task, which only uses the visual information present in the images to provide a surrogate supervision signal for feature learning. For example, some self-supervised tasks include, colorization ([28], [17]), relative position ([5], [23]), egomotion [2], and rotation prediction [3]. The reasoning is that solving these tasks will constrain the CNN to learn high-level image representations that can be useful for other visual tasks. However, designing powerful self-supervised tasks still largely relies on intuition and requires ample time. Additionally, recent work has shown that combining multiple self-supervised tasks almost always leads to improved performance ([6], [22]), which suggests that the space of good self-supervised tasks may be relatively large and optimal points may be unintuitive.

We present a general framework for self-supervised task discovery. Most self-supervised methods can be parameterized to encompass a broader range of tasks than the one originally proposed. For example, for colorization, instead of *a priori* deciding to use the *Lab* colorspace, we could learn a (parameterized) function that takes the original (RGB) image and outputs an image such that trying to predict the last two channels from the first will produce useful visual features. It is possible that, within this space, there exists a function that will provide better results than RGB-to-*Lab*. Ultimately, we would like a general function that encapsulates all the current self-supervised task and more, Fig. 1. Our approach uses a model, the proposer, to generate the parameters of that function, which defines a task. Training a network, the solver, to solve this task results in metrics about how good the task is. Using this metric as a reward signal, we can then update the proposer to generate better and better tasks.

We apply our method to rotation prediction [3], where the task is to recognize which rotation angle, from a set, was applied to an image. Therefore, the goal is to find a set of angles for the rotation task that produce general visual features. We choose the rotation task since the search space (finite number of continuous angles) is large enough to potentially generalize, but small enough so that experiments

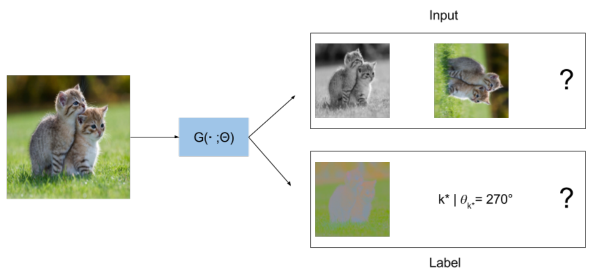


Figure 1: Tasks generated by a general function $G(\cdot;\Theta)$ with parameters Θ that takes in raw, unlabeled images. Specific task inputs and labels are shown for the case of colorization and rotation. The question marks represent the unknown inputs and labels that produce the best self-supervised task.

could be run within reasonable compute and time limitations. Our results demonstrate that the algorithm is able to find a set of angles that satisfy the same properties as described in the original paper [3] and achieve comparable performance to state-of-the-art self-supervised methods.

2. Related Work

The problem of learning to discover an optimal state is a standard formulation in reinforcement learning. In particular, Neural Architecture Search (NAS) [29] uses reinforcement learning to discover optimal hyperparameters for neural networks. The work encodes the structure and connectivity of a neural network in a variable-length string. It is therefore possible to use a recurrent neural network to generate such a string. Training the network specified by the string on the task of interest will result in an given validation accuracy. Using this accuracy as the reward signal, they compute the policy gradient to update the controller. As a result, in the next iteration, the controller will give higher probabilities to architectures that receive high accuracies, allowing the controller will learn to improve its search over time. The idea of using some internal metric as a reward has also been explored in various other contexts ([24], [12], [16]). Our method is similar to the structure of NAS, but is applied in a novel context. Instead of optimizing architectural hyperparameters of a neural network, we aim to optimize for the parameters that define a (self-supervised) task.

The search for a good self-supervised task is also related to the concept of active learning, where an agent is optimized to obtain optimal data. Traditionally, this is in the context where there is abundant unlabeled data, limited labeled data, and the ability to label data. Then obtaining optimal data means choosing maximally informative unlabeled data that should be labeled [26]. From a developmental behavior point of view, it has been shown that babies act on the world, generating data, in a way that enhances their understanding of how it works ([11], [27]). A connection exists due to the nature of how a self-supervised task also defines the data that the network sees.

3. Methodology

Fig. 2 shows the framework for discovering an optimal task. There is a continuous interaction between two components, the proposer and the solver. The solver calculates certain metrics for a given task and the proposer uses the metrics to select better tasks. There are many choices for possible metrics, but the metric used should be informative of how well the task supports learning high-level visual features.

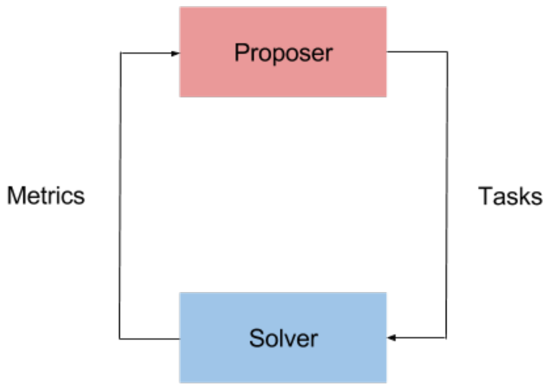


Figure 2: Overview of the framework for discovering self-supervised tasks

3.1. Solver

The input to the solver is a finite set of angles that define a given rotation prediction task. The solver then trains a CNN to solve the given rotation task, thereby computing the desired metrics.

Fig. 3 shows an overview of the rotation prediction task. A given task is defined by a set of K angles, $\Theta = \{\theta_k|_{k=1}^{K}\}$, with corresponding geometric transformations $R(\cdot|\theta_k)$ that results in a rotated image $X^k = R(X|\theta_k)$, when applied to image X. The CNN model $F(\cdot)$ is given a transformed image X^{k^*} as input and assigns a probability distribution over the possible transformations:

$$F(X^{k^*};\phi) = F^k(X^{k^*};\phi)|_{k=1}^{K} \tag{1}$$

where $F^k(X^{k^*};\phi)$ is the predicted probability for rotation by angle θ_k and ϕ are the learnable parameters of the model $F(\cdot)$.

Finally, given a dataset of N training images $D = X_i|_{i=1}^{N}$ we learn the CNN parameters ϕ by solving:

$$\min_{\phi} loss(D, \phi) \tag{2}$$

where $loss(\cdot)$ is defined as:

$$loss(D, \phi) = -\frac{1}{N}\sum_{i=1}^{N} log(F^{k^*}(X_i^{k^*}; \phi)) \tag{3}$$

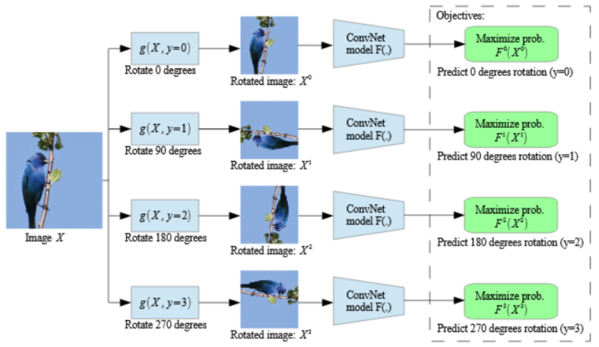


Figure 3: Overview of the rotation prediction task, from [3]

The optimization procedure produces a sequence of data on which the model was trained $D_{0,...,t}$ and model parameters $\phi_{0,...,t}$. With these, we introduce a metric function $M_t(D_{0,...,t}, \phi_{0,...,t}; \Omega)$ that quantifies aspects of the task at each time step t, where Ω represents possible parameters or additional inputs. Some examples metrics include, rotation prediction accuracy, training loss value, and area under the curve (AUC) of the loss. While any of these could potentially be used, we decided on the AUC metric since it captures information about the whole training trajectory. Therefore, our metric function takes the following form:

$$M_t(D_{0,...,t}, \phi_{0,...,t}; \omega) = \sum_{\tau=1}^{t} \frac{\hat{L}_{\tau-1} + \hat{L}_{\tau}}{2} \tag{4}$$

where $\hat{L} = S(L; \omega)$ is the smoothed loss with a smoothing function $S(\cdot; \omega)$ and $L_t = loss(D_t, \phi_t)$ is the loss at time t.

Therefore, the solver can be described as a mapping from Θ to $M_t^{\Theta} = M_t(D_{0,...,t}, \phi^{\Theta}_{0,...,t}; \omega)$ where ϕ^{Θ} are the parameters learned solving the task defined by Θ.

3.2. Proposer

The proposer takes a task (e.g. a set of angles) with the corresponding metric values (e.g. AUC) and outputs new tasks. Since the metric values serve as a proxy for the power of the task, the proposer aims to propose tasks that optimize the metrics.

In this work, we accomplish this in a two steps. First we build a model $P(\cdot; \psi)$, with parameters ψ, to predict the metric M_t^{Θ}. Next, we define a policy π_P that samples new tasks explicitly from the metric model as follows:

$$\pi_P(\Theta) \sim exp(\beta P(\Theta; \psi)) \tag{5}$$

where the hyperparameter β, or inverse temperature, is used to control the exploration-exploitation trade-off.

An alternative to the two step approach would be to build a single model that directly generates new tasks, e.g. with policy gradients. We do not adopt this method for simplicity and it is left for future work.

4. Experiments & Results

We provide an overview of the rotation task landscape by analyzing the relationship between the choice of angles to predict and the resulting representations that are learned. First, we start by reproducing the RotNet results. Then we perform a series of experiments by solving rotation prediction with different sets of angles. We demonstrate that metrics obtained from the training, such as AUC, correlate with the performance of the learned visual features on ImageNet classification. Finally, we apply our problem search framework to discover effective rotation tasks.

4.1. Rotation Prediction

4.1.1 RotNet Implementation

We use the RotNet architecture described in [3], which is based on AlexNet[15] with minor modifications. The final fully connected layer modified to produce the same number of outputs as angles in the task, there is no local response normalization, and there is batch normalization after every linear layer. We train the network using SGD with momentum 0.9, and weight decay $5e-4$ for 30 epochs with an initial learning rate of 0.01 and decayed by a factor of 10 after 10 and 20 epochs. We also feed all rotations of the image simultaneously within the same mini-batch, which was found to improve performance in [3]. This means that the size of each mini-batch is actually a factor of the number of angles in the task larger than the number of unique unrotated images. For the original 4 angle case with batch size 192, this would be $768 = 4 \times 192$. We keep this number constant in our experiments with different number of angles. Unlike the original paper, since we rotate images by angles that are not multiples of 90 degrees, there will be empty spaces (filled with zeros) in addition to the low level interpolation artifacts. We want to focus on the visual artifacts so a circular mask was applied to the images.

To evaluate the generalization and power of the self-supervised task, we train a classifier to perform the ImageNet classification task using the learned features. Following the method used in [23], we freeze the weights in conv1-4 and retrain the following layers, i.e. conv5, fc6, fc7, and fc8. We use the same hyperparameters as in the rotation task, except we use a batch size of 256 for this case. The top-1 ImageNet classification accuracy is computed on the held-out validation set. We verified that we were able to reproduce the original result of 50.0% top-1 accuracy (we obtained 51.9%).

4.1.2 Rotation Artifacts

To understand the nature of the effect of the rotation artifacts, we try various sets of angles with different levels of artifacts present. We plot the confusion matrix to visualize the errors the models are making. The general result of these experiments can be summarize by a couple examples. The first is the case where the set of angles (in degrees) is $\{x + 90n \mid x \in \{0, 45\}, n \in \{0, 1, 2, 3\}\}$. We see that the two values for x create two subsets, each of which contains angles that are multiples of 90 degrees apart from the others. Within each subset there are the same artifacts, but these differ between the subsets. We see this phenomenon reflected in the block diagonal structure of the confusion matrix in Fig. 4. Another example is the case where the angles are $\{0, 36, 108, 216\}$. Based on the result of the previous example, we would expect that the rotations by 36 and 216 degrees to be confused with each other while the other rotations can be easily distinguished from the others using the visual artifacts. As shown in Fig. 5, this is exactly the result we obtain. There is almost perfect accuracy in classifying rotations by 0 and 108 degrees with accuracies over 99.9%.

4.2. Choosing a Metric

We then ask whether performance on the rotation task predicts the performance of the learned representations on ImageNet classification. Another way to think about this is that using artifacts allows the network to perform better in rotation classification without actually learning powerful visual representations. To do this, we train RotNet on a variety of tasks ranging from 2 to 8 angles, including random sets of angles and the original set, with different levels of artifacts present in each. We compare how well different metrics for performance correlate to the ability of the learned representations to transfer to ImageNet classification. We first looked at the final validation rotation classification accuracy as the metric, Fig. 6a, which is able to separate out the harder problems (i.e. lower accuracy, further to the left), but falls short for the easier problems. This ceiling effect where multiple tasks reach the same, almost perfect accuracy, but result in different performances on ImageNet is clearly shown by the points in the bottom right of Fig. 6a. Therefore, we propose to use the area under curve (AUC) of the loss as an alternative. The AUC is calculated on the smoothed loss, and we use a moving average filter

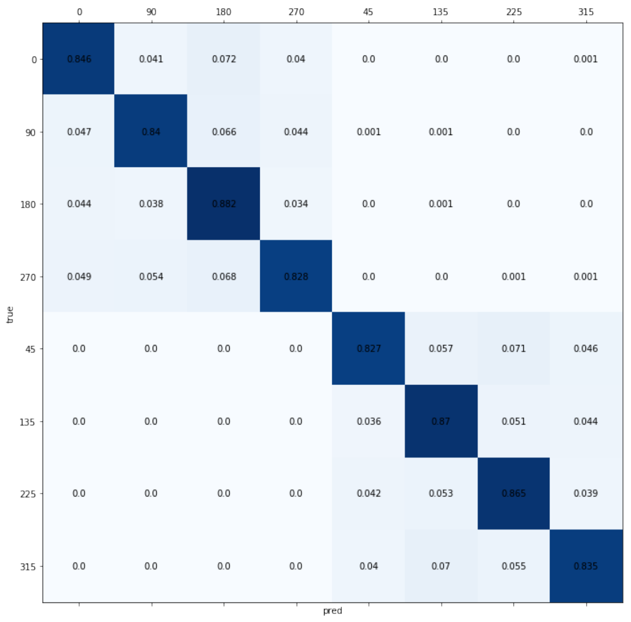


Figure 4: Confusion matrix for the 8-way rotation classification task with angles $\{x + 90n \mid x \in \{0, 45\}, n \in \{0, 1, 2, 3\}\}$ on validation images.

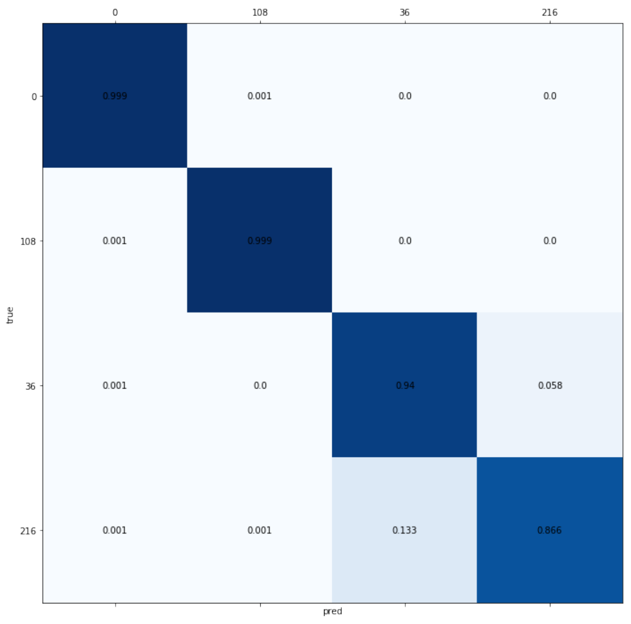
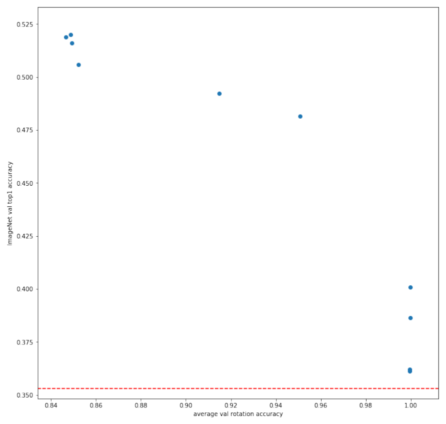
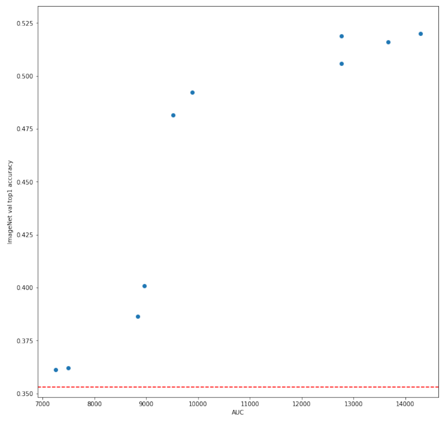


Figure 5: Confusion matrix for the 4-way rotation classification task with angles $\{0, 36, 108, 216\}$ on validation images.

with kernel length 100 for all the experiments. The rationale is that since the AUC is computed from the whole loss trajectory it may provide finer resolution than the rotation accuracy. Fig. 6b shows that this is indeed the case and there is a nice correlation between the AUC and ImageNet transfer performance at both ends of the scale.



(a) Rotation Accuracy



(b) AUC

Figure 6: Each point represents a different set of angles used in the rotation classification task. The x-axis is the final validation rotation classification accuracy (**top**) or the AUC at 6000 steps (**bottom**). The y-axis is the best ImageNet top-1 accuracy when the layers after conv4 are trained. The dotted red line denotes baseline performance when conv1-4 are initialized with random weights.

Since the framework relies on doing multiple passes of training on the rotation task, it is critical that the metric can be quickly determined during training. To investigate what a sufficient number of steps to train, we plot the correlation between AUC and ImageNet top-1 accuracy as a function of the number of steps. We see the results in Fig. 7 and the number of steps can be chosen to reach the desired trade-off between accuracy and speed. While there is a sharp increase in correlation up until about 6000 steps, it is worth noting that the correlation is still quite good for fewer steps.

One thing to note is that we do not take into account the degenerate cases when using AUC as the metric. An exam-

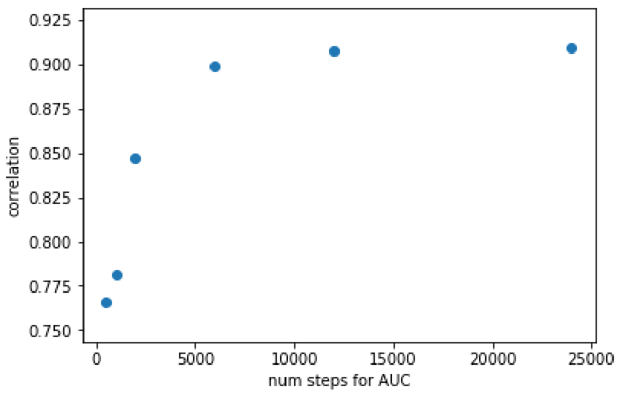


Figure 7: Plot of correlation between AUC and ImageNet top-1 accuracy versus number of steps for AUC.

ple of a degenerate case for the rotation task is when two angles are the same, which results in an unlearnable problem with a high AUC. However, even though the AUC is high, these are regions we want to avoid. A possible remedy would be to combine the rotation accuracy and AUC together in the metric, since the rotation accuracy will be at chance for these cases. A more general approach is to have a model of the inverse problem, who's loss would give an estimate of how degenerate the problem is. This is left for future work.

4.3. Learning the Metric Map

Here we investigate the ability of different classes of models to learn the metric map for the rotation task with two angles. To generate the training data, we sampled 800 points in $[0, 360]^2$ and computed the AUC at 6000 steps for each.

The initial model was a simple multi-layer perceptron (MLP). The input is a vector of the two angles in the task and the label was the AUC. We use Adam [14] to minimize the l_2 loss between our model's outputs and the labels. A thorough hyperparameter search was conducted over the learning rate, batch size, model depth, and hidden dimensions. However, no model was able to successfully capture the data. The sparse manifold on which the hard problems lie is reflected in the significant class imbalance in our dataset. To alleviate this we also tried including an additional term in the loss that would weight examples with high labels more, which did not lead to significant improvements.

In search of a better model class for learning this landscape, we chose Gaussian processes (GP). The GP with RBF kernel was trained using the scikit-learn Python package [25]. We can see in Fig. 8 that the GP is able to capture all the significant peaks. The strong peaks along the diagonal correspond to degenerate problems where both angles are the same. As expected, there are also other slightly lower peaks that correspond to problems where the two angles are multiple of 90 degrees apart. The slight checker pattern is a result of the details in TensorFlow's [1] rotate function. For angles close to multiple of 90 degrees, there is a larger deviation up to which a rotation by that amount will still result in the same image.

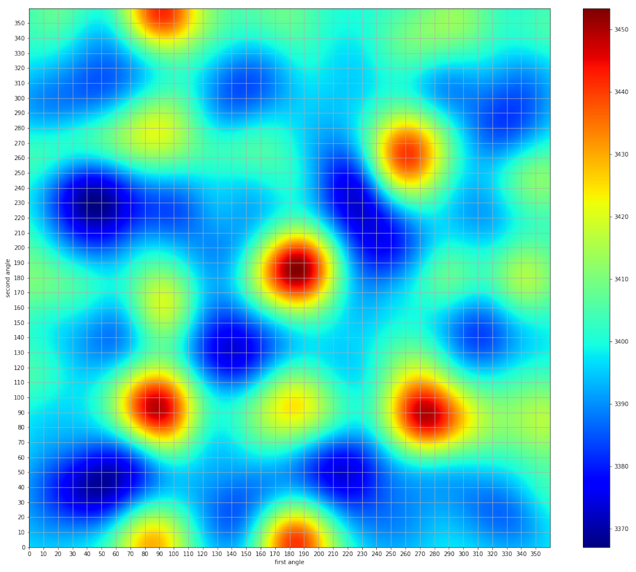


Figure 8: Plot of metric map learned by GP. Each axis represents one of the two angles in the task and the color corresponds to the predicted AUC.

4.4. Task Discovery

We apply our task discovery framework to the subset of problems contained in the rotation task. Specifically, our task search space is a pair of angles (i.e. $K = 2$), where each angle can take a value from the *continuous* range $[0, 360]$.

4.4.1 Implementation Details

Our implementation made use of distributed TensorFlow in order to parallelize the tasks across GPUs on a cluster. The tasks were divided into three jobs: parameter server, proposer, and solver. The parameter server is responsible for maintaining the two FIFO queues for proposer inputs (metrics queue) and solver inputs (task angles queue). The proposer is composed of two tasks, one for each step described in 3.2. The first task trains the GP model described in 4.3 by dequeuing from the metrics queue, with a batch size of 2. It adds these values to its history of inputs and retrains the model with the full history. The second task performs sampling based on Eq. 5, where β follows an exponential decay with a decay rate of 0.97 and initial value of 1 and the model P is the one model recently trained. The sampled tasks are continuously fed into the task angles queue, removing the oldest element in the queue when necessary. A solver task obtains the most recent problems from the task angles queue and uses the AlexNet based architecture and training procedure described in 4.1 to compute the AUC. When the training is complete, the solver enqueues the task angles with its corresponding AUC into the metrics queue before repeating. The solver task is the most computationally intensive part of the procedure so several choices were made to improve efficiency. The first is using 64x64 images instead of the standard 256x256 size, which improved training speeds by about 4x normally but only 2x in the distributed implementation. We also only train for 500 steps, which we found to be adequate. The main benefit of the distributed implementation is the ability to scale the number of solver tasks to the amount of GPU resources available easily. For our experiments we used 5 solver tasks across 3 GPUs, trained overnight.

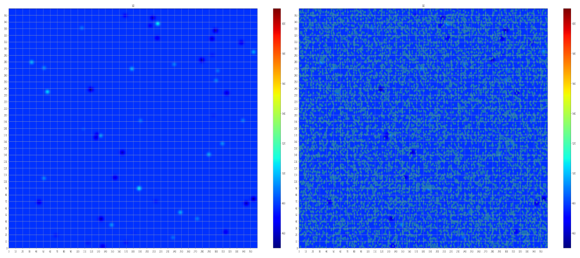
4.4.2 Results

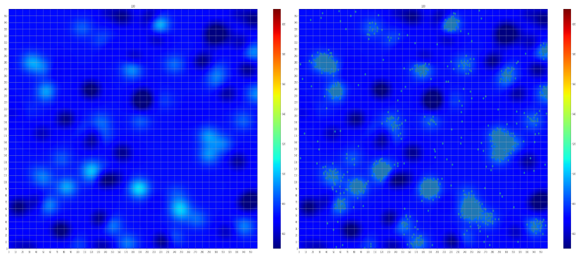
Our method is able to converge on a set of angles 90 degrees apart, which, as demonstrated in 4.1.2 and 4.2, results in a strong self-supervised task. Fig. 9 shows the trajectory of metric maps learned by the GP. In Fig. 9a we see a mostly flat posterior since only 10 points have been sampled and none of them have a comparable AUC to the hardest problems. We see a gradual increase in the number of peaks as well as the appearance of several more significant peaks. For example, in Fig. 9c, there are major peaks around (55, 235) and (180, 270). By the end, after 200 steps, the algorithm converges to a single task, $\Theta = \{179.7, 270.1\}$. The right-hand side of Fig. 9 illustrates the effect of the temperature decay in our sampling policy. The blue dots represent the samples that are chosen at the given time step. The relatively high temperature in the beginning allows for adequate exploration. By 50 steps, Fig. 9b, we can see that the samples are already starting to concentrate around the peaks. At step 100, the sample are concentrated at a single peak and even more so at step 200.

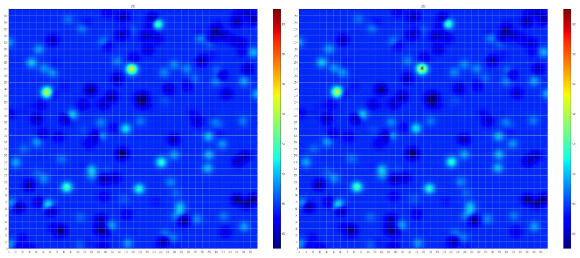
We then train the self-supervised rotation task with these angles and evaluate the learned representation on ImageNet classification, which obtains a top-1 accuracy of about 49% on ImageNet, demonstrating the effectiveness of these angles for the self-supervised rotation task. This is not unexpected since we also verified that images rotated by 179.7 and 270.1 degrees are the same images rotated by 180 and 270 degrees, respectively.

5. Conclusion

Just as the difficulty of designing powerful neural network architectures inspired the development of Neural Ar-


(a) Step 25 (50 samples)


(b) Step 50 (100 samples)


(c) Step 100 (200 samples)

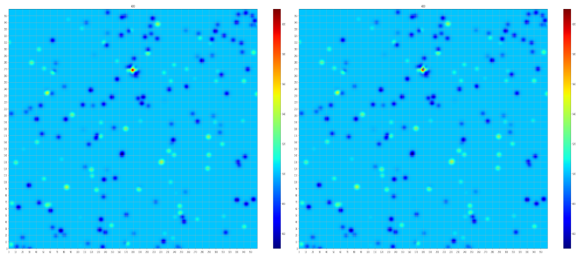

(d) Step 200 (400 samples)

Figure 9: Plot of metric map learned at different points in time (**left**) with samples at the corresponding temperature overlaid as blue dots (**right**). Each axis represents one of the two angles in the task and the color corresponds to the predicted AUC.

chitecture Search, the success of the multitude of self-supervised tasks motivates the utility of a problem search algorithm. In this work, we present a method for searching a task space and demonstrate its effectiveness in the case of a two angle rotation prediction task. However, our method is fully general given any reasonable parameterization of the task space. This suggests that given a sufficiently large task space, our method should be able to find a self-supervised

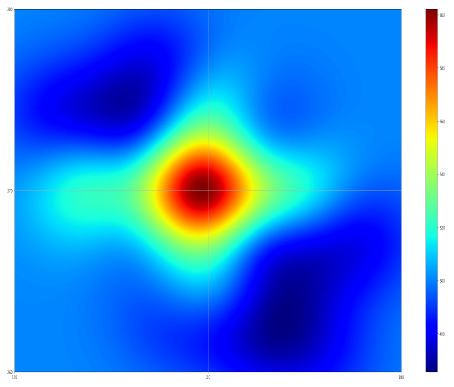


Figure 10: Plot of metric map learned at step 200 zoomed in to highest peak at roughly (180, 270).

task that performs better than any current method. The same framework can also be viewed from the perspective of active and continual learning. So far, we only consider the task defined by the final set of parameters our algorithm converges on. However, we can also take the task to be a sequence of tasks corresponding to the parameters generated during the search trajectory. In this case, our method becomes a general approach for automated curriculum learning. These directions are left for future work.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[2] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. *CoRR*, abs/1505.01596, 2015.

[3] Anonymous. Unsupervised representation learning by predicting image rotations. *International Conference on Learning Representations*, 2018.

[4] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[5] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. *CoRR*, abs/1505.05192, 2015.

[6] C. Doersch and A. Zisserman. Multi-task self-supervised visual learning. *CoRR*, abs/1708.07860, 2017.

[7] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016.

[8] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 766–774, 2014.

[9] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

[11] A. Gopnik, A. Meltzoff, and P. Kuhl. *The Scientist In The Crib: Minds, Brains, And How Children Learn*. Harper-Collins, 2009.

[12] R. Houthooft, X. Chen, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: Variational information maximizing exploration. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1109–1117. Curran Associates, Inc., 2016.

[13] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *CoRR*, abs/1604.06057, 2016.

[17] G. Larsson, M. Maire, and G. Shakhnarovich. Colorization as a proxy task for visual understanding. *CoRR*, abs/1703.04044, 2017.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] R. Liao, A. Schwing, R. Zemel, and R. Urtasun. Learning deep parsimonious representations. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 5076–5084. Curran Associates, Inc., 2016.

[20] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[21] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59, 2011.

[22] T. N. Mundhenk, D. Ho, and B. Y. Chen. Improvements to context based self-supervised learning. *arXiv preprint arXiv:1711.06379*, 2017.

[23] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[24] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. *arXiv preprint arXiv:1705.05363*, 2017.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] B. Settles. *Active Learning*, volume 18. Morgan & Claypool Publishers, 2011.

[27] A. E. Stahl and L. Feigenson. Cognitive development. Observing the unexpected enhances infants' learning and exploration. *Science*, 348(6230):91–94, Apr 2015.

[28] R. Zhang, P. Isola, and A. A. Efros. *Colorful Image Colorization*, pages 649–666. Springer International Publishing, Cham, 2016.

[29] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016.