



AFRL-RI-RS-TR-2023-181

AUTOMATED MULTISOURCE ADAPTATION VIA ZERO-SHOT INFORMATION GENERATION (AMAZING)

HRL LABORATORIES, LLC

OCTOBER 2023

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC)(<http://www.dtic.mil>).

AFRL-RI-RS-TR-2023-181 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

NANCY SEPULVEDA
Work Unit Manager

/ S /

MATTHEW KOCHAN
Technical Advisor
Intelligence Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE		2. REPORT TYPE		3. DATES COVERED	
OCTOBER 2023		FINAL TECHNICAL REPORT		START DATE	END DATE
				AUGUST 2019	MAY 2023
4. TITLE AND SUBTITLE Automated Multisource Adaptation via Zero-shot INformation Generation (AMAZING)					
5a. CONTRACT NUMBER		5b. GRANT NUMBER		5c. PROGRAM ELEMENT NUMBER	
FA8750-19-C-0098		N/A		61101E	
5d. PROJECT NUMBER		5e. TASK NUMBER		5f. WORK UNIT NUMBER	
				R2TK	
6. AUTHOR(S) Amir Rahimi					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) COLUMBIA UNIVERSITY SPONSORED PROJECTS ADMINISTRATION 116TH AND BDWY NEW YORK NY 10027					8.PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIEA 525 Brooks Road Rome NY 13441-4505				10.SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI	11.SPONSOR/MONITOR'SREPORT NUMBER(S) AFRL-RI-RS-TR-2023-181
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT During this project, HRL Laboratories LLC, in collaboration with a world-class team from Johns Hopkins University (JHU, Prof. Philip Koehn) and the University of California Davis (UC Davis, Prof. Hamed Pirsiavash), made significant advancements in the fields of image classification, self-supervised learning, and machine translation, particularly in learning from limited data and adapting to novel environments. Their efforts under the Learn with Less Labels (LwLL) program were aimed at innovating systems that can learn from data even when labels are scarce or entirely absent. For self-supervised learning, they proposed a novel mean-shift algorithm that learns representations by grouping images together without contrasting between them or adopting much of prior on the structure or number of the clusters and created methods to incorporate constraints to the mean-shift algorithm, enabling us to leverage additional knowledge sources. They also developed robust defense mechanisms against sophisticated backdoor attacks tailored for transformer-based algorithms, thus bolstering the robustness of self-supervised learning.					
15. SUBJECT TERMS Image classification, self-supervised learning, video recognition, zero-shot learning, machine translation					
16. SECURITY CLASSIFICATION OF:				17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	c. THIS PAGE		SAR	83
U	U	U			
19a. NAME OF RESPONSIBLE PERSON NANCY A.SEPULVEDA					19b. PHONE NUMBER (Include area code) N/A

TABLE OF CONTENTS

List of Figures	ii
List of Tables	v
1.0 SUMMARY	1
2.0 INTRODUCTION	2
3.0 METHODS, ASSUMPTIONS, PROCEDURES; RESULTS AND DISCUSSION.....	5
3.1 Image Classification	5
3.1.1 Generalized Sliced-Wasserstein Embeddings, Pooling by Sliced Wasserstein Embedding	5
3.1.2 Part-Based Learning.....	11
3.1.3 Deep Embedded Clustering in Few-shot Representations (DECiFR)	12
3.2 Self-supervised learning	22
3.2.1 Mean Shift for Self-Supervised Learning	22
3.2.2 Backdoor Attacks on Self-Supervised Learning	29
3.2.3 Consistent Explanations by Contrastive Learning	37
3.3 Machine translation	42
3.3.1 Embedding Space Mapping for Bilingual Lexicon Induction	42
3.3.2 Graph Matching Methods for Bilingual Lexicon Induction	47
3.3.3 Bilingual Lexicon Induction for Low-Resource Languages using Graph Matching via Optimal Transport (GOAT)	51
3.4 JPL Evaluations.....	54
4.0 CONCLUSIONS.....	56
5.0 REFERENCES	58
APPENDIX A – Publications and Presentations	72
LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS.....	73

LIST OF FIGURES

<i>Figure 1: Our AMAZING six-step process enables learning with a fraction of labels per class in a new target domain. Steps 1 to 5 use zero labels from the target domain.</i>	4
<i>Figure 2: The GSWE framework for learning from sets computes transport maps to reference sets</i>	7
<i>Figure 3: Our proposed graph embedding framework, WEGL, combines node embedding methods with the linear Wasserstein embedding framework.</i>	8
<i>Figure 4: Self-Supervised Learning with GSWE outperforms PMA on the Point Cloud MNIST dataset</i>	9
<i>Figure 5: We evaluated the performance of PSWE and other baseline pooling methods on (a) 3D point cloud classification from the ModelNet40 dataset, (b) TUD graph classification datasets, and (c) image recognition on the NWPU-RESISC45 and Places-Extra69 datasets.</i>	10
<i>Figure 6: Our architecture for part-based few-shot learning</i>	11
<i>Figure 7: Our part-based-learning approach outperformed the baseline for 5 or more labels per class.....</i>	12
<i>Figure 8: 5-way few-shot accuracy using DECFR compared with SOTA on the CUB200 dataset. Our method, DECFR, is shown in orange.</i>	13
<i>Figure 9: DECFR architecture consists of three components: 1) Feature Learning, 2) Label Propagation, and 3) Classification. Given a mixture of backbones, 1a) Useful backbones are pre-selected based on their corresponding DUNN indexes. 1b) The flat versions of selected backbones are used to learn a more discriminative representation by simultaneous mapping and clustering features using DEC. 2) The optimized features are used in the cKNN module and more labeled samples are added in the feature space. 3) KPCA is applied to obtain the final classifications.</i>	14
<i>Figure 10: Accuracy gain by using DECFR along with cKNN compared to KNN.....</i>	18
<i>Figure 11: 5-way 5-shot few-shot accuracy of DECFR using all the different combinations of 4 a) selected backbones by the pre-processing module (DUNN-index), b)randomly selected backbones on the CUB200 dataset.</i>	19
<i>Figure 11 shows 5-way 5-shot few-shot accuracy of DECFR using all the different combinations of 4 a) selected backbones by the pre-processing module (DUNN-index), b)randomly selected backbones on the CUB200 dataset.</i>	19
<i>Figure 12: MSF method: Similar to BYOL, we maintain two encoders (“target” and “online”). The online encoder is updated with gradient descent while the target encoder is the moving average of the online encoder. We augment an image twice and feed to both encoders. We add the target embedding to the memory bank and look for its nearest neighbors in the memory bank. Obviously target embedding itself will be the first nearest neighbor. We want</i>	

to shift the embedding of the input image towards the mean of its nearest neighbors, so we minimize the summation of those distances. Note that our method using only one nearest neighbor is identical to BYOL which pulls different augmentations together without grouping different instances of images. To our knowledge, our method is the first in grouping different instances of images without contrasting between image instances or clusters. 22

Figure 13: Nearest neighbors (NN) of the model at each epoch: For a random query images, we show how the nearest neighbors evolve at the learning time. Initially, NNs are not semantically quite related but are close in low-level features. The accuracy of the 1-NN classifier in the initialization is 1.5% which is 15 times larger than random chance (0.1%). This little signal is bootstrapped in our learning method and results in NNs of the late epochs which are mostly semantically related to the query image. More visualizations can be found in the appendix. 23

Figure 14: Poisoning exemplar-based Self-Supervised (SSL) methods: A poisoned input image is used in an exemplar-based SSL method, e.g., BYOL. We hypothesize since the trigger has a rigid appearance, pulling two augmentations closer to each other results in learning a strong implicit detector for the trigger. Since the trigger always co-occurs with the target category only, the model associates the trigger with the target category 30

Figure 15: Targeted Attack Threat Model: First self-supervised model is trained on a poisoned unlabeled dataset. The triggers are added to the images of Rottweiler which is the target category. Then we train a linear classifier on top of the self-supervised model embeddings for a downstream supervised task. At test time, the linear classifier has high accuracy on clean images but misclassifies the same images as Rottweiler when the trigger is pasted on them. 30

Figure 16: Our method significantly improves the consistency of Grad-CAM explanation heatmaps under data augmentation. For both the RV and the Puma, our method highlights the same portions of the image in both the original and augmented versions. 37

Figure 17: The block diagram of our method. Our method consists of both cross-entropy loss (LCE) and contrastive Grad-CAM consistency loss (LCGC). We load a batch of random images, and consider one to be the query image. We feed the query image to the network and calculate LCE. We calculate Grad-CAM for this image on the top predicted category and then augment the heatmap. We then augment all the images in the batch, feed them to our model, and calculate the Grad-CAM heatmap for the top predicted category. The top category is chosen using the original image and not the augmented one. (Note that all random images are also augmented with independently sampled parameters. We do not show this to reduce the clutter.) The heatmap from the augmented query image is considered to be the positive example. The heatmaps from the other random images in the batch are the negative examples. The augmented query heatmap, the positive heatmap, and the negative heatmaps are all fed into our contrastive comparison function to produce our LCGC term. 38

Figure 18: Grad-CAM visualization results for images from ImageNet validation set using ResNet50. We observe that the model trained using our method focuses on the most discriminative regions of the object instead of the background pixels. In the 6th column above,

we compute the Grad-CAM explanation for the category “volleyball” and we see that our model no longer focuses on the players and correctly highlights the volleyball as the cause of the target category. The last column contains a failure example for the category “Dogsled”. The baseline model correctly highlights the sled whereas our model incorrectly highlights the dog..... 42

Figure 19: Unsupervised machine translation based on a common cross-lingual embedding space 43

Figure 20: Comparing Euclidean vs. Graph views. (a) Euclidean view assumes common embedding space and computes costs based on pairs of vectors. (b) Graph-based view assumes graph structure and computes cost based on pairs of edges. Both exploit within-language neighborhood info but in different ways. 47

Figure 21: IterSGM and IterProc. Run SGM (or Procrustes) in forward and reverse directions. Combine hypotheses and pass them as seeds to SGM (Procrustes). Pull final translations on the last iteration from the forward run. 48

Figure 22: Optimization step of FAQ vs. GOAT. FAQ arbitrarily chooses the direction of a permutation matrix. GOAT averages permutation matrices to take a smoother path..... 52

Figure 23: Visualization of our results for select languages. Procrustes (dashed) vs. SGM (dotted) vs. GOAT (solid). X-axis: number of seeds (log scale). Y-axis: Precision@1 (higher is better). GOAT is typically best. 53

Figure 24: Combination system: Iterative Procrustes (IterProc) and GOAT. (1) run GOAT in forward/reverse directions (2) intersect hypotheses, pass to IterProc, (3) run IterProc forward and reverse (4) intersect hypotheses, pass to Step (1). Repeat N cycles. (End) Get final translations from the forward run of GOAT or IterProc..... 53

Figure 25: Evaluation Performance Low-label Regime. DECFR vs Baseline for tasks 1, 2, and 3 for few-shot checkpoints (1-4). 55

Figure 26: Evaluation Performance Low label Regime. DECFR vs Kprop for tasks 1, 2 and 3 for few-shot checkpoints (1-4). 55

LIST OF TABLES

<i>Table 1: Test classification accuracy (%) of our PSWE method and the baseline pooling mechanisms on the ModelNet40 point cloud dataset using multi-layer perceptron (MLP) and induced set attention block (ISAB) backbones.</i>	<i>10</i>
<i>Table 2: Comparing PSWE with GAP, PMA, and FSPool for various backbone networks on various datasets from TUD Benchmark. Bold numbers are the best for each row.</i>	<i>11</i>
<i>Table 3: 5-way 5-shot few-shot accuracy using DECFR with selected backbone architectures by DUNN-index pre-processing compared 714 to using best and worst performing single backbones.....</i>	<i>18</i>
<i>Table 4: 5-way few-shot accuracy using DECFR compared with SOTA on the CUB200 dataset.</i>	<i>20</i>
<i>Table 5: 5-way few-shot accuracy using DECFR compared with SOTA on RESISC45 dataset. 20</i>	
<i>Table 6: 5-way few-shot accuracy using DECFR compared with SOTA on EuroSat dataset. ...</i>	<i>21</i>
<i>Table 7: 5-way few-shot accuracy using DECFR compared with SOTA on the Fewshot-CIFAR100 dataset. The top two accuracies are highlighted in bold.....</i>	<i>21</i>
<i>Table 8: 5-way few-shot accuracy using DECFR compared with SOTA on the miniImageNet dataset. The top two accuracies are highlighted in bold.....</i>	<i>21</i>
<i>Table 9: Additional computational cost of finding NNs: Forwarding through each ResNet50 encoder needs 4.14 GFLOPS, so finding NNs adds a small cost. Note that any method like MoCo that uses a memory bank needs this additional cost.</i>	<i>25</i>
<i>Table 10: Left: Evaluation on full ImageNet: We compare our model on the full ImageNet linear and nearest neighbor benchmarks using ResNet50. We find that given a similar computational budget, our models are better than other state-of-the-art methods. Our w/s variation works slightly better than the regular MSF. Interestingly, when using weak augmentations only, our method (MSF w/w) outperforms BYOL and SimCLR by a large margin. We believe this is important in some applications, e.g... medical domain, where augmentation engineering is not easy. Note that methods with symmetric loss are not directly comparable with the other ones as they need to feed each image twice through each encoder. This results in twice the computation for each mini-batch. One may argue that a non-symmetric BYOL with 200 epochs should be compared with a symmetric BYOL with 100 epochs only as they use a similar amount of computation. Note that symmetric MoCo v2 with 400 epochs is almost the same as asymmetric MoCo v2 with 800 epochs (71.0 vs. 71.1). Note that the accuracy of Random-init for ResNet50 is much lower than AlexNet (14.1% on conv5 layer from [Nor+18]). †: CompRes is not directly comparable as it uses ResNet50 distilled from a larger SSL teacher model (SimCLR-ResNet50x4). ‡: SwAV is not comparable as it uses multiple crops together. Right: Epochwise t-SNE for MSF: We visualize the ℓ_2 normalized features for 10 random ImageNet classes at certain epochs of MSF training. We find that over the period of training, semantic clusters are formed in the feature space.</i>	<i>27</i>

Table 11: Evaluation on small labeled ImageNet: We compare our model on the ImageNet 1% and 10% linear evaluation benchmarks for ResNet50. The column “Fine-tuned” refers to whether the full network was fine-tuned or a single linear layer was trained. Given similar computational budgets, both of our models are better than other state-of-the-art methods. We evaluate BYOL and MoCo v2 with our evaluation framework and interestingly, realize that BYOL performs better in linear evaluation compared to finetuning the whole network on the 1% split. We report these numbers for a fair comparison. * is ResNet50 compressed from SimCLR-R50x4. † uses a different augmentation strategy than others. ‡ is our evaluation with the official weights [BYO].	28
Table 12: Linear layer transfer learning evaluation: We compare various SSL methods on transfer tasks by training linear layers. Under similar computational budgets, we show that our models are consistently better or on par with other state-of-the-art methods. Only a single linear layer is trained on top of features. No training time augmentations are used. “rep.” means we have reproduced the results using our evaluation framework for better comparison.	28
Table 13: Transfer learning to PASCAL VOC object detection: We compare our models on the transfer task of object detection. We find that given a similar computational budget, our method is better than BYOL. The models are trained on the VOC trainval07+12 set and evaluated on the test07 set. We report average over 5 runs.	29
Table 14: Targeted attack on ImageNet-100: We use a 0.5% poison injection rate. Each experiment uses a random target category and trigger. SSL methods are trained on poisoned ImageNet-100 data and a linear classifier is trained on 1% of ImageNet-100 labeled data. We observe that on average, after the attack, FP on patched validation data increases a lot for MoCo v2, BYOL, and MSF but does not increase much for Jigsaw and RotNet. For MAE, we finetune on 1% ImageNet-100.	35
Table 15: Targeted attack on ImageNet-100: We use 0.5% poison injection rate. SSL methods are trained on poisoned ImageNet-100 data and a linear classifier is trained on 10% ImageNet-100 labeled data - averaged over 10 target class trigger pairs.	36
Table 16: Untargeted attack on ImageNet-100: We poison 5% random images of ImageNet-100 training set. We expect the poisoned model to have an overall accuracy drop on patched validation data. The targeted attack contributes to a 5-point decrease in accuracy. The linear classifier is trained on 1% of ImageNet-100.	36
Table 17: Classification accuracy along with the Content Heatmap (CH) and CGC Loss explanation metrics on the ImageNet validation set. Note that lower is better for CGC Loss.	41
Table 18: The Grad-CAM explanation maps generated by our ResNet50 model outperform the baseline ResNet50 model on the IAUC metric [PDS18] using the ImageNet validation set.	41
Table 19: Classification accuracy and Content Heatmap (CH) evaluation on the 28 UnRel categories [Pey+17] that overlap with ImageNet.	41

<i>Table 20: On the 1% ImageNet limited-label setting, our CGC method leverages unlabeled data for the LCGC loss term and is able to improve the classification accuracy and explanations when evaluated on ImageNet validation data. Both models are initialized from a ResNet50 model trained in an unsupervised manner using SwaV [Car+20].</i>	42
<i>Table 21: UN=United Nations (parallel data extracted from the United Nations website), CC=Common Crawl (parallel data extracted from a general web crawl), News=News Crawl (monolingual news content).</i>	46
<i>Table 22: Unsupervised MT performance on a single run using the United Nations (UN) dataset. “Diff. Dom.” uses UN data as source and Common Crawl (*) as target. “Sup.” is supervised with UN parallel data. We report BLEU scores (higher is better). A / A refers to UN training dataset A used on the source and target sides, for example.</i>	46
<i>Table 23: Unsupervised MT performance for truly low-resource language pairs (Sinhala-English and Nepali-English).</i>	47
<i>Table 24: Precision@1 results for the graph matching method SGM vs. the baseline Procrustes method on German-English and Russian-English. Except for a moderate amount of seed bilingual entries (1000 and 2000-word translations) for German-English, our SGM method outperforms the standard approach.</i>	49
<i>Table 25: Precision@1 of IterProc vs. IterSGM. Add-All runs for 10 iterations, seeding subsequent iterations with the intersection of hypotheses from forward and reverse directions. For Stochastic-Add, seeds are fed in up to 100 at a time until all are used. In parentheses is the improvement over Add-All.</i>	50
<i>Table 26: P@1 for combined iterative method. One may begin from either IterProc (“Start: Procrustes”) or SGM (“Start: SGM”) and may pull final hypotheses from either Procrustes (“-PullProc”) or SGM (“-PullSGM”). “Prev. Best” is the best result from previous experiments (excluding active learning). Bold is best overall.</i>	51
<i>Table 27: P@1 of Combination Exps. -EP starts with GOAT, and ends with IterProc. -EG: IterProc, ends with GOAT. Prev is the previous best of prior experiments.</i>	54

1.0 SUMMARY

During this project, HRL Laboratories LLC, in collaboration with a world-class team from Johns Hopkins University (JHU, Prof. Philip Koehn) and the University of California Davis (UC Davis, Prof. Hamed Pirsiavash), made significant advancements in the fields of image classification, self-supervised learning, and machine translation, particularly in learning from limited data and adapting to novel environments. Our efforts under the Learn with Less Labels (LwLL) program are aimed at innovating systems that can learn from data even when labels are scarce or entirely absent. For self-supervised learning, we proposed a novel mean-shift algorithm that learns representations by grouping images together without contrasting between them or adopting much of prior on the structure or number of the clusters and created methods to incorporate constraints to the mean-shift algorithm, enabling us to leverage additional knowledge sources. We also developed robust defense mechanisms against sophisticated backdoor attacks tailored for transformer-based algorithms, thus bolstering the robustness of self-supervised learning. Further, we've introduced methods to enhance deep model interpretation, such as enforcing consistency and a contrastive learning method, which improves explanation heatmaps and boosts classification accuracy, particularly on fine-grained datasets, underscoring the importance of quality explanations alongside accuracy in model evaluation. Our image classification research comprises innovative works on Generalized Sliced-Wasserstein Embeddings (GSWE), Part-Based Learning, and Deep Embedded Clustering in Few-shot Representations (DECiFR). We proposed the GSWE for learning from set-structured data which led to the creation of a unique pooling layer for deep neural networks, our novel pooling layer, PSWE, does not require a significant increase in the number of parameters compared to existing methods, providing an efficient solution. Additionally, we developed the DECiFR algorithm, which enhances feature embeddings and boosts few-shot classification accuracy. DECiFR achieves this by leveraging Deep Embedded Clustering to abstract discriminative information from different backbones and implementing a contrastive version of KNN. For machine translation, we introduced ground-breaking graph-matching techniques utilizing optimal transport, substantially improving performance in low-data environments, and developed pioneering methods for inducing bilingual lexicons from monolingually trained word embedding spaces. These advancements, showcased in this report, have not only resulted in superior outcomes and performance improvements but have also laid a solid foundation for future research and development. Our solutions have consistently surpassed existing methods, achieving favorable results in Hackathon evaluations for machine translation, video recognition, and image classification.

2.0 INTRODUCTION

Our innovative efforts, as part of the LwLL program, aimed to design systems capable of learning from data when few or no labels are available. Throughout this program, we have successfully proposed and developed a variety of solutions addressing image classification, self-supervised learning, and machine translation. This report showcases these diverse solutions, along with their respective results and discussions, structured independently, as a consequence some mathematical notations will differ among these works. The main solutions will be discussed and summarized in a dedicated subsection under Section 3 in which the methods and results are combined, further details can be found in the referenced papers.

We begin by introducing the problem and highlighting the primary solutions. The subsequent sections delve into the details, summarizing our efforts and advancements achieved under the LwLL program.

Problem Description:

The state-of-the-art for learning fewer labeled samples is to use a deep neural network that is pre-trained on a different but similar dataset with lots of labeled samples and, then, re-tune the final layer(s) of the network to classify the new target dataset. This approach has two weaknesses: 1) it assumes common features between datasets without enforcing commonality, leading to errors; and 2) it neglects the abundance of unlabeled data, limiting its performance (label reduction is limited to about 100x before dramatically losing accuracy). For learning without labels, state-of-the-art zero-shot learning (ZSL) approaches struggle with two things: 1) defining semantically meaningful attributes, which often come from human annotation or paired textual domains, and 2) not knowing if the input samples belong to seen or unseen classes of data (i.e., generalized-ZSL versus classic ZSL), leading to performance much lower than for supervised learning (8x higher prediction error).

As part of the LwLL program, our innovations were driven by our goal to construct a system that can learn from a new unlabeled target dataset and predict labels in the target domain without loss of accuracy compared to fully-supervised learning. To get us near this perfect performance, we propose a new six-step process that consists of 1) creating a similarity graph between source models, 2) learning optimal pseudo tasks for each source model, 3) extracting components from source models, 4) building a dictionary of abstract attributes from the components, 5) aligning target data with model data to map the target data onto abstract attributes, and 6) active learning for data for which we do not have a good match with existing attributes (Figure 1). The benefit of this approach is that for a large chunk of data (we estimate 90% maybe more depending on the dataset), we can leverage previously labeled data to learn with *zero new labels* because the dissection into components increases the chance to leverage old data. For the remaining data, we can still learn efficiently with only 1 label per class, resulting in 0.1 labels per class in total.

Step 1. Source-similarity-graph creation: We leverage a multitude of pre-existing data sources. For LwLL, DARPA is expected to provide a corpus of about 400 machine-learning datasets and associated models. We propose a new similarity measure between these source models and create a source similarity graph (SSG) linking the models based on their nearest neighbors. The benefit of the graph is to quickly find a small set of models that are closest to the target data.

Step 2. Pseudo-tasks optimization: Optimizing a pseudo-task for self-supervised learning is new. First, we find the best pseudo task for each source model. Here, the optimization

criterion is to create networks that allow a discriminative component extraction *comparable to* what we get from a network trained in a supervised way (since we know the source labels). Then, on the target data, we again carry out SSL by combining the best pseudo tasks from the closest source models. The result is an initial target network that maps the input onto a latent feature space.

Step 3. Unsupervised data decomposition: Given the trained networks from the sources, we build upon HRL’s unique prior work [Kolouri et al. 2017] to extract salient components from input data (e.g., extracting the cropped image of a car tire). Different from our prior work, we propose new ideas that include building a hierarchy of components and extracting salient words or phrases.

Step 4. Attribute-dictionary creation: We map all components across input data and sources onto abstract attributes. In prior work, we have demonstrated that this mapping can be carried out through unsupervised clustering, resulting in semantically meaningful clusters, e.g., a cluster of bike wheels [Kolouri et al., 2017]. Different from our previous work, here, we propose to combine components from multiple sources because we want to obtain unique attributes across models. Given the mined attributes, we build a dictionary of sets of attributes. For each set, we have the corresponding labels from the sources. So, for objects and actions, we can learn a mapping from the attributes onto labels. For language, we learn a recurrent neural network that generates a sentence given a set of attributes.

Step 5. Zero-shot attribute distillation: Given our attributes from the sources, we need to process the unlabeled target data so that they map onto the right set of attributes. To achieve this mapping, we first select the closest source model or models based on our similarity graph and then use a new method (zero-shot attribute distillation) that aligns the sources with the target. Alignment happens by learning a shared latent space for the source and target models that is predictive of the attributes of the source data and simultaneously generative of the target data. As a result, we have a network that maps target data onto a probability distribution over the abstract attributes in our dictionary. We can then map this distribution onto a label or sentence. In all these five steps, we did not use a single training label to assign labels to data from a new domain.

Step 6. Active learning: For some data, we may not find a good attribute representation, where goodness is, e.g., measured by having high values in the attribute probability distribution or high confidence in these values. For data without a good representation, we propose an active learning method. To carry out this learning in a label-efficient way, we propose to first collect all data for which active learning is required and then carry out label propagation in the latent feature space of the data.

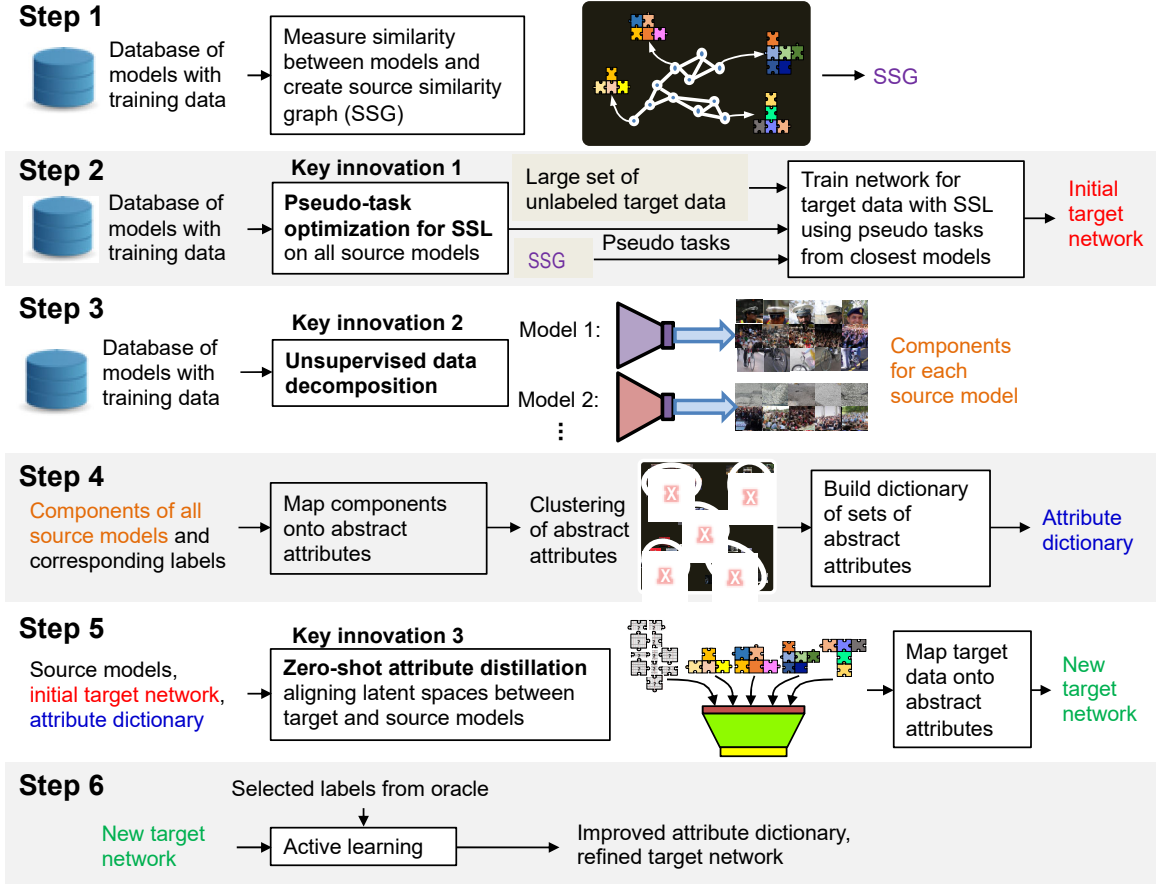


Figure 1: Our AMAZING six-step process enables learning with a fraction of labels per class in a new target domain. Steps 1 to 5 use zero labels from the target domain.

The output of these six steps is a trained network that returns labels for a target domain. These steps describe the process for training a new network from scratch. If only adapting a network to a new sensor or different viewpoint or lighting, we can simplify the above process. Here, we use a trained model from the six-step process as a starting point, which becomes the new source model. In prior work, we demonstrated domain adaptation with zero labels and minimal loss compared to supervised learning for handwritten digits, but a larger loss (12.6%) for more complex input (road scenes). We aim to drop this loss for complex input below 1% by adapting the target network with self-supervised learning using the optimal pseudo-task from the source model.

Throughout the program, we have actively contributed to the various evaluations and meetings. Our code, designed for image classification, video recognition, and machine translation, can be accessed at <https://gitlab.lollll.com/hrl/>. We have built this code to operate within a single Docker container. The container interfaces with the JPL LwLL API (https://gitlab.lollll.com/lwll/lwll_api), allowing it to receive essential information about data, checkpoints, and evaluation metrics, among other parameters. Detailed instructions and explanations are provided in the associated readme files.

3.0 METHODS, ASSUMPTIONS, PROCEDURES; RESULTS AND DISCUSSION

This final report highlights some of our methods and solutions implemented during the program. For a more comprehensive understanding and additional details, we encourage referring to the related research papers. For a deeper understanding and further details, we recommend consulting the associated research papers. In the sections that follow, we provide a summary of our endeavors in image classification, self-supervised learning, and machine translation, emphasizing the results and discussions for each topic.

3.1 Image Classification

Regarding image classification research we present a summary of work in this section and discuss Generalized Sliced-Wasserstein Embeddings (Section 3.1.1), Part-Based Learning (Section 3.1.2), and Deep Embedded Clustering in Few-shot Representations (Section 3.1.3). The Generalized Sliced-Wasserstein Embeddings (GSWE) and Pooling by Sliced Wasserstein Embedding (PSWE) leverage the Generalized Sliced Wasserstein distance, forming a framework for unsupervised and supervised learning on set-structured data, and developing a new pooling layer respectively. These techniques have shown efficacy in various domains including 3D point clouds, graph learning, and image classification. Meanwhile, the Wasserstein Embedding for Graph Learning (WEGL) offers a novel and fast framework for embedding entire graphs for various graph-level prediction tasks. Our part-based learning architecture exploits attribute histograms for efficient few-shot learning. We additionally propose Deep Embedded Clustering in Few-shot Representations (DECiFR), a model-agnostic framework for Few-Shot Learning (FSL). Unlike previous FSL approaches, our DECiFR framework does not require prior knowledge of backbone performance to determine which backbone is more suitable for a specific FSL task. The pipeline consists of three major components: feature learning, label propagation, and classification. Utilizing an ensemble of pre-trained model architectures, our proposed approach leverages a feature preprocessing stage with the DUNN index and a Deep Embedded Clustering (DEC) approach to achieve better clustering representation. This methodology resulted in superior few-shot classification results compared to several state-of-the-art methods.

3.1.1 Generalized Sliced-Wasserstein Embeddings, Pooling by Sliced Wasserstein Embedding

3.1.1.1 Leveraging Wasserstein distances for forming and comparing deep representations of images

An increasing number of machine learning tasks deal with learning representations from set-structured data. Solutions to these problems involve the composition of permutation equivariant modules (e.g., self-attention, or individual processing via feed-forward neural networks) and permutation-invariant modules (e.g., global average pooling, or pooling by multi-head attention).

Many traditional machine learning architectures, such as feed-forward neural networks, operate on constant-size inputs. Each sample fed into such an architecture consists of a list of features, whose size is kept fixed throughout the training/testing process. On the other hand,

architectures, such as convolutional and recurrent neural networks (CNNs and RNNs, respectively), allow the size of each input sample to be arbitrary, but they still carry a notion of, e.g., spatial or temporal, ordering among the input features.

Nevertheless, there exist many problems in which each sample consists of an unordered set of elements. 3-D point cloud classification, sequence order, and even problems as simple as finding the maximum/minimum element of a set are examples of problems in which the size of each input sample can be different and the ordering of the input elements is unimportant. Such a set learning phenomenon also arises at the output of graph neural networks (GNNs) and CNNs, where a backbone is applied on the graph nodes or, respectively, the original input image pixels, leading to an unordered set of node embeddings (resp., super-pixel features). This intermediate set of embeddings is then mapped to a constant-size embedding that represents the entire input graph/image through a pooling method, such as average/max pooling, which is insensitive to the size and order of the embedding set.

Such problems have motivated general-purpose set embedding methods that provide a parametric mapping of sets to a fixed-dimensional embedding space by means of a permutation-invariant function. In (Zaheer et al., 2017), the authors introduced the notion of Deep Sets, where each element of a set first undergoes a backbone, and the resulting embeddings are then aggregated via a simple mean/sum pooling method. The work (J. Lee et al., 2019) proposed Set Transformers, where a permutation-equivariant self-attention mechanism is used to perform message-passing among the set elements, followed by a cross-attention module with set seed elements to derive a permutation-invariant embedding for the whole set. Moreover, in (Skianis et al., 2020), a network-flow-based approach is introduced, where the relationship between each set and multiple hidden sets are used to derive the set embeddings for any given set.

On a different, but related note, in their pioneering work, (Kusner et al., 2015) viewed the elements of a set as samples of an underlying probability measure and leveraged 1-Wasserstein distance (i.e., the earth mover's distance) to compare sets with one another. The work of (Kusner et al., 2015) and its extension to supervised learning (Huang et al., 2016) show that comparing the underlying probability measures is a powerful idea and leads to excellent performance. The Wasserstein distances and their variations have become increasingly popular in machine learning and computer vision. In our work, we leverage the Generalized Sliced Wasserstein (GSW) distance (Kolouri et al., 2019) and propose a geometrically-interpretable framework for learning from set-structured data. We make the following contributions:

- 1 Proposing a Euclidean embedding for the generalized sliced-Wasserstein distance, which we refer to as Generalized Sliced-Wasserstein Embedding (GSWE)
- 2 Leveraging GSWE to devise a new framework for unsupervised and supervised learning on set structured data; and,
- 3 Demonstrating the effectiveness of GSWE in learning from sets and comparing it to state-of-the-art approaches (J. Lee et al., 2019)(Skianis et al., 2020).

An overview of the proposed GSWE framework is shown in Figure 2. Each d -dimensional element in a given input set Z_i , as well as each reference set $Z_0^k, k \in \{1, \dots, K\}$ is passed

through multiple slicers $\{g_{\theta_l}\}_{l=1}^L$. For each slicer, we then perform interpolation on the slicer outputs and derive the optimal transport maps that push the slicer output distributions of a given set to the slicer output distributions of each of the reference sets. The resultant transport maps are then concatenated across all slices and reference sets to derive the final set of embeddings. The slicer operations on the input set elements are permutation-equivariant. Moreover, in practice, the slicers may be implemented using multi-layer perceptrons (MLPs), which share the parameters of all layers except for the last layer. In other words, the set of L slicers can be viewed as a unified MLP, represented by a mapping $g: \mathbb{R}^d \rightarrow \mathbb{R}^L$.

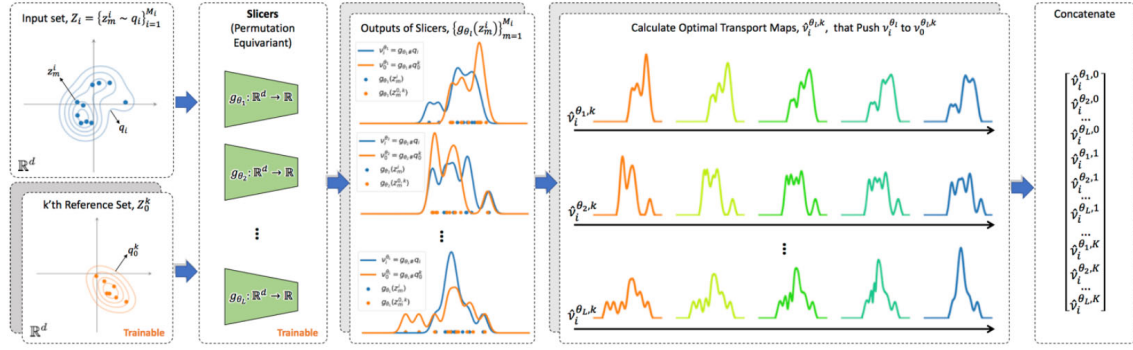


Figure 2: The GSWE framework for learning from sets computes transport maps to reference sets

3.1.1.2 Generalized Sliced-Wasserstein Embeddings, Pooling by Sliced Wasserstein Embedding

As presented, we introduced the idea of Generalized Sliced-Wasserstein Embeddings [2] for learning from sets. We used this idea to develop a new pooling layer and carried out rigorous head-to-head comparisons with the existing methods on various domains including 3D point clouds, graph learning, and image classification. Pooling plays a critical role in aggregating features extracted by convolutional layers in modern deep neural networks. Global average/max pooling captures only a limited representation of the distribution of convolutional features. In contrast, a more holistic comparison between the distributions of convolutional features is beneficial [3], especially in low-label regimes. Here, we describe a novel pooling mechanism, which is rooted in the generalized sliced-Wasserstein distances. Our proposed pooling mechanism allows for a holistic comparison of the convolutional features while not requiring a significant increase in the number of parameters needed for pooling, unlike attention-based pooling mechanisms. We demonstrate that our proposed pooling generalizes global average pooling, and it has an elegant geometric interpretation [2]. Notably, our pooling provides a Euclidean embedding for the generalized sliced-Wasserstein distance between the distribution of convolutional features and trainable reference sets. We showed that our proposed approach generalizes the recently proposed pooling approach, Feature Sort Pooling (FSPool), and provides a theoretical foundation for this otherwise heuristic approach. We denote our pooling by Pooling by Sliced Wasserstein Embedding (PSWE).

3.1.1.3 Wasserstein Embedding for Graph Learning

We recognize the need to adapt machine learning architectures to various input sizes. Traditional structures, such as feed-forward neural networks, handle constant-size inputs, while

others, like convolutional and recurrent neural networks (CNNs and RNNs), handle arbitrarily sized inputs, maintaining an inherent spatial or temporal order. To bridge this gap, we present Wasserstein Embedding for Graph Learning (WEGL) a novel and fast framework for embedding entire graphs in a vector space, in which various machine learning models are applicable for graph-level prediction tasks. We leverage new insights on defining similarity between graphs as a function of the similarity between their node embedding distributions. Specifically, we use the Wasserstein distance to measure the dissimilarity between node embeddings of different graphs. Different from prior work, we avoid pairwise calculation of distances between graphs and reduce the computational complexity from quadratic to linear in the number of graphs. WEGL calculates Monge maps from a reference distribution to each node embedding and, based on these maps, creates a fixed-sized vector representation of the graph. We evaluate our new graph embedding approach on various benchmark graph-property prediction tasks, showing state-of-the-art classification performance while having superior computational efficiency. Our proposed graph embedding framework, WEGL, is presented in Figure 3.

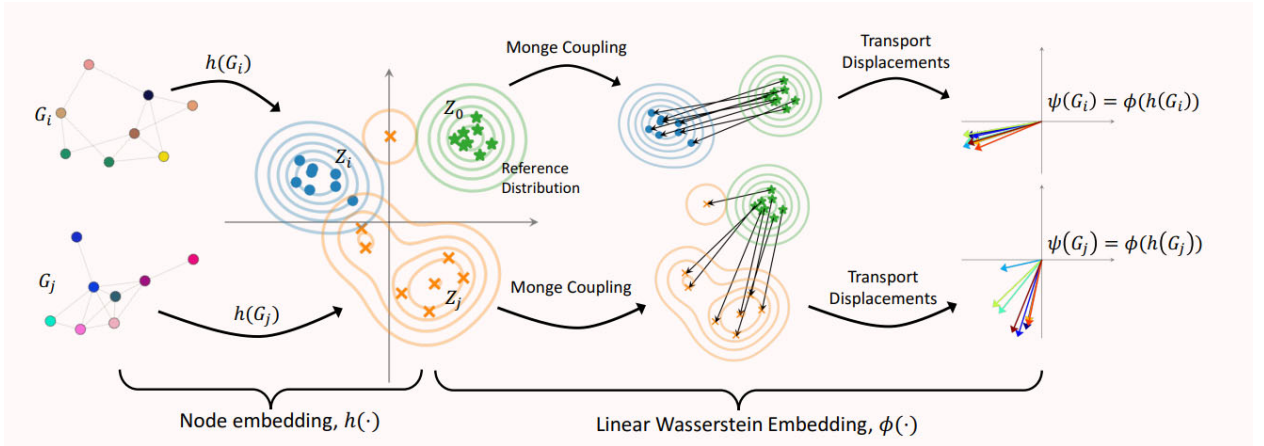


Figure 3: Our proposed graph embedding framework, WEGL, combines node embedding methods with the linear Wasserstein embedding framework.

3.1.1.4 Results and Discussion

We use the GSWE framework for learning on sets in supervised and self-supervised settings. Here we report the results of our self-supervised learning experiments. We first focused on unsupervised learning of set embeddings using the self-supervised approaches. We used the following two datasets:

- **Set-Circles:** We generate a simple two-dimensional dataset called, Set-Circles (see Figure 2, Top Left). Each sample is a set with a random number of elements generated on a random arc on a circle with additive noise. There are two classes, in the dataset which are identified by the radius of the circle the samples live on. The average norm of the set elements is an ideal feature for discriminating the classes.
- **Point Cloud MNIST:** We also consider the two-dimensional point cloud MNIST dataset (LeCun et al., 1998), where each sample consists of a set of points in the XY-plane converted from the pixels of a corresponding 0-9 digit image (see Figure 2, Bottom Left).

We carried out self-supervised learning on these datasets using two losses, namely, SimCLR and SimSiam. After training the networks, we performed nearest neighbor retrieval for the test sets and measured the label agreement between the input and the retrieved set. We emphasize that for the Set-Circles dataset to avoid a trivial solution, all methods use a backbone that maps the set elements to a one-dimensional embedding (otherwise the problem becomes trivial to solve).

Figure 4 shows the performance of GSWE as compared with the Set Transformer architecture (denoted by PMA, referring to the pooling with multi-head attention module) for different cardinalities of the reference set. As the figure shows, on both datasets and using both loss functions, our proposed approach either performs similarly to or outperforms the Set Transformer method for reference sets with more than one element. For one element in the reference set, our proposed GSWE method effectively reduces to global average pooling, while PMA can be viewed as weighted global average pooling. That explains the performance gain achieved by PMA for reference sets of size 1. However, as soon as an additional element is added to the reference set, GSWE performs significantly better than PMA. Moreover, for the same backbone, GSWE has far fewer parameters than PMA, due to the absence of multi-head attention in the pooling module in GSWE, which helps explain the superiority of GSWE as compared to PMA in terms of nearest neighbor retrieval accuracy for sufficiently complex datasets.

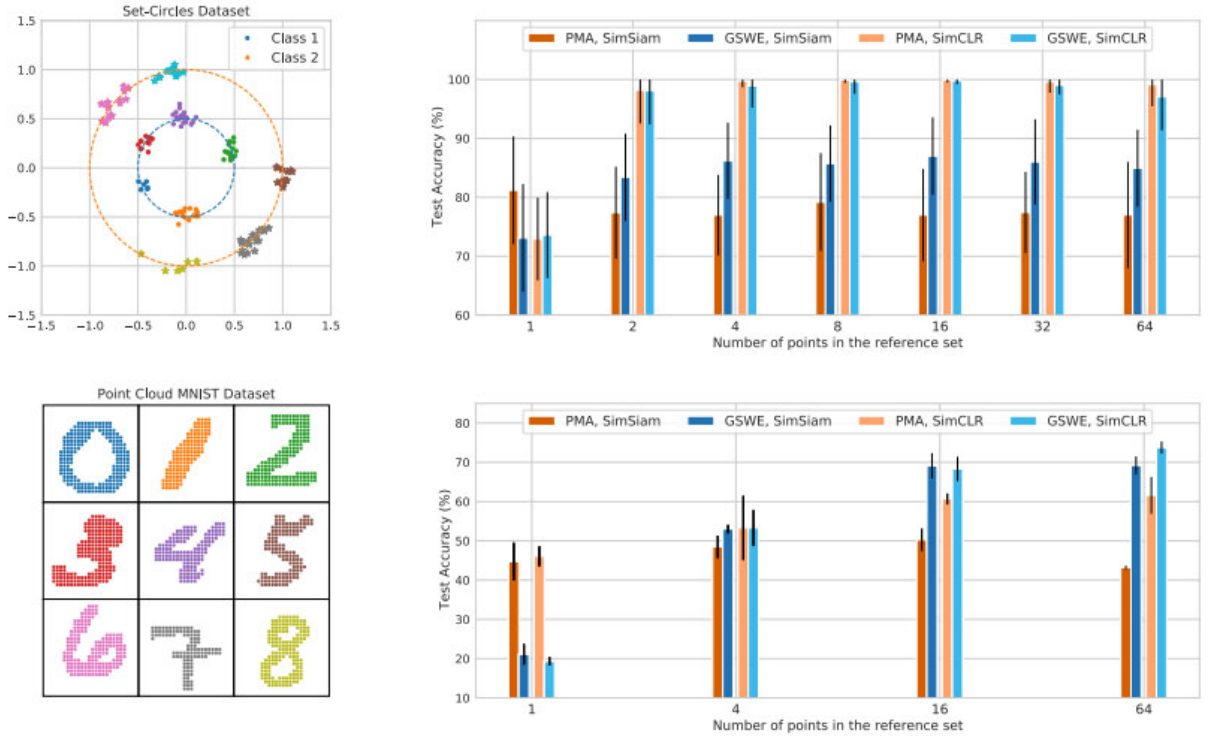


Figure 4: Self-Supervised Learning with GSWE outperforms PMA on the Point Cloud MNIST dataset

We evaluated our PSWE method on a variety of point cloud, graph, and image datasets as depicted in Figure 5. For comparison, we used three different pooling methods: Global average pooling (GAP), Pooling by Multi-head Attention (PMA) [4], and Featurewise Sort Pooling (FSPool) [5]. Below, we include our results on different datasets.

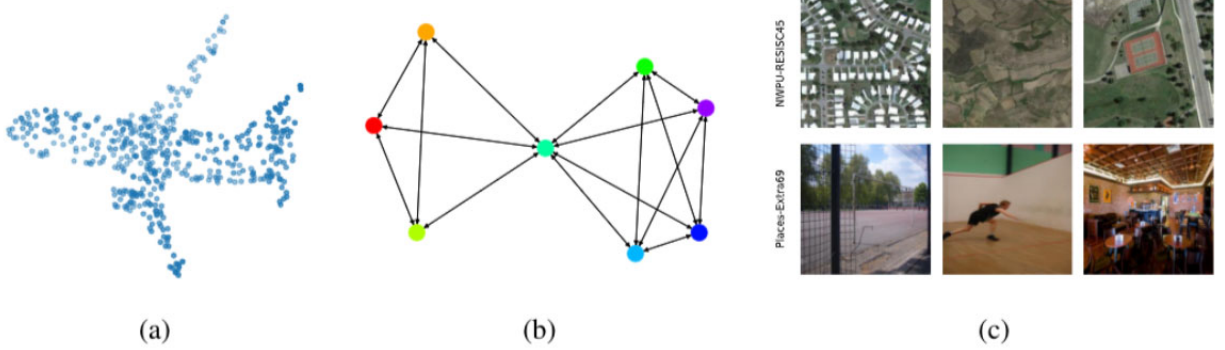


Figure 5: We evaluated the performance of PSWE and other baseline pooling methods on (a) 3D point cloud classification from the ModelNet40 dataset, (b) TUD graph classification datasets, and (c) image recognition on the NWPU-RESISC45 and Places-Extra69 datasets.

Point Cloud Classification: We used the ModelNet40 dataset [6], consisting of 3-dimensional point clouds derived from triangular meshes of 12,311 CAD models belonging to 40 object categories. We sampled 1024 points uniformly at random from each object as in [7] and used the official split, with 9,843 training samples and 2,468 test samples. We used two different backbones, namely multi-layer perceptron (MLP) and induced set attention block (ISAB) from the Set Transformer architecture [4]. Table 1 shows our results on this dataset. We improved the state-of-the-art results on ModelNet for the MLP backbone and provided comparable results for the induced set attention block.

Table 1: Test classification accuracy (%) of our PSWE method and the baseline pooling mechanisms on the ModelNet40 point cloud dataset using multi-layer perceptron (MLP) and induced set attention block (ISAB) backbones.

Backbone	GAP	PMA	FSPool	PSWE					
				$L = 1$	$L = 4$	$L = 16$	$L = 64$	$L = 256$	$L = 1024$
MLP	57.8 ± 0.5	86.6 ± 0.6	85.8 ± 0.5	14.9 ± 1.0	52.9 ± 2.1	77.4 ± 0.4	83.9 ± 0.6	86.5 ± 0.5	86.9 ± 0.3
ISAB	86.6 ± 0.5	87.6 ± 0.6	87.3 ± 0.5	32.4 ± 3.6	83.9 ± 0.6	86.2 ± 0.5	86.9 ± 0.3	87.3 ± 0.4	87.6 ± 0.4

Graph Classification: Next, we use the prominent TUD benchmark [8] and evaluated the performance of our method on five graph classification datasets, consisting of social network (IMDB-B, IMDB-M, REDDIT-B) and bio-informatics (ENZYMES, PROTEINS) datasets. For the former group of datasets, we used one-hot encoded degrees as initial node features, while for the latter group, we used the provided node labels as initial node features. We then passed the features, alongside the adjacency matrices, to three popular graphs neural network (GNN) backbones, namely Graph Convolutional Network (GCN) [9], Graph Attention Network (GAT) [10], and Graph Isomorphism Network (GIN) [11]. Upon deriving the final node embeddings of a given graph from a GNN backbone, we treat them as elements of a

set and apply PSWE and the baseline pooling methods to derive a fixed-size graph-level representation that is fed to a linear classifier. Table 2 shows our results on graph datasets.

Table 2: Comparing PSWE with GAP, PMA, and FSPool for various backbone networks on various datasets from TUD Benchmark. Bold numbers are the best for each row.

	Backbone	GAP	PMA	FSPool	PSWE					
					$L = 1$	$L = 4$	$L = 16$	$L = 64$	$L = 256$	$L =$
IMDB-B	GCN	69.6 $\pm$ 3.9	74.1 $\pm$ 5.3	75.5 $\pm$ 3.7	72.6 $\pm$ 7.6	74.6 $\pm$ 5.8	77.3 $\pm$ 5.1	73.0 $\pm$ 7.5	72.7 $\pm$ 5.1	73.5
	GAT	73.4 $\pm$ 3.5	70.5 $\pm$ 7.6	72.4 $\pm$ 6.9	71.3 $\pm$ 7.2	74.4 $\pm$ 5.8	74.0 $\pm$ 5.7	70.9 $\pm$ 7.6	73.0 $\pm$ 3.6	73.4
	GIN	73.0 $\pm$ 5.8	70.0 $\pm$ 8.0	73.4 $\pm$ 6.1	73.8 $\pm$ 6.5	72.5 $\pm$ 6.0	72.0 $\pm$ 4.3	72.3 $\pm$ 8.1	74.6 $\pm$ 7.0	68.8
IMDB-M	GCN	51.8 $\pm$ 4.2	50.1 $\pm$ 2.8	51.1 $\pm$ 5.4	44.2 $\pm$ 4.9	50.7 $\pm$ 4.6	50.8 $\pm$ 3.8	49.6 $\pm$ 3.6	51.4 $\pm$ 4.2	50.2
	GAT	49.7 $\pm$ 3.4	49.6 $\pm$ 4.9	50.2 $\pm$ 3.9	44.3 $\pm$ 4.2	49.2 $\pm$ 4.1	50.2 $\pm$ 4.5	48.2 $\pm$ 4.6	47.9 $\pm$ 4.3	49.4
	GIN	49.7 $\pm$ 2.9	50.2 $\pm$ 3.0	50.8 $\pm$ 5.3	44.6 $\pm$ 4.4	49.1 $\pm$ 2.7	48.0 $\pm$ 6.4	50.6 $\pm$ 2.6	50.2 $\pm$ 3.4	49.5
RDT-B	GCN	81.9 $\pm$ 2.6	82.2 $\pm$ 2.5	84.0 $\pm$ 2.8	77.5 $\pm$ 4.5	80.4 $\pm$ 3.3	81.5 $\pm$ 2.1	82.1 $\pm$ 3.2	81.9 $\pm$ 3.2	81.7
	GAT	75.8 $\pm$ 3.3	76.0 $\pm$ 3.5	84.7 $\pm$ 3.3	78.7 $\pm$ 2.5	82.0 $\pm$ 3.2	82.1 $\pm$ 3.1	81.7 $\pm$ 3.2	83.0 $\pm$ 3.3	81.7
	GIN	81.2 $\pm$ 3.1	77.6 $\pm$ 7.9	84.4 $\pm$ 2.9	83.2 $\pm$ 3.3	83.1 $\pm$ 2.8	83.8 $\pm$ 2.8	84.6 $\pm$ 2.3	83.9 $\pm$ 2.8	83.7
PROTEINS	GCN	69.1 $\pm$ 5.2	72.4 $\pm$ 5.9	74.9 $\pm$ 5.4	72.5 $\pm$ 3.9	73.3 $\pm$ 5.3	73.3 $\pm$ 5.6	73.2 $\pm$ 6.1	72.8 $\pm$ 6.0	73.9
	GAT	69.7 $\pm$ 4.4	72.4 $\pm$ 6.1	73.0 $\pm$ 5.1	72.9 $\pm$ 4.6	73.1 $\pm$ 4.5	72.8 $\pm$ 4.4	73.9 $\pm$ 4.6	74.4 $\pm$ 4.4	73.7
	GIN	69.8 $\pm$ 6.5	72.3 $\pm$ 4.7	72.6 $\pm$ 4.5	71.3 $\pm$ 4.9	72.4 $\pm$ 6.0	73.4 $\pm$ 4.8	73.5 $\pm$ 4.4	73.0 $\pm$ 5.0	74.9
ENZYMES	GCN	25.0 $\pm$ 5.1	32.1 $\pm$ 4.5	33.5 $\pm$ 4.2	20.0 $\pm$ 3.9	24.9 $\pm$ 6.5	31.8 $\pm$ 5.1	32.5 $\pm$ 3.0	37.8 $\pm$ 4.9	33.7
	GAT	24.2 $\pm$ 5.3	28.8 $\pm$ 3.9	34.2 $\pm$ 6.7	22.3 $\pm$ 4.0	26.3 $\pm$ 5.6	30.6 $\pm$ 4.7	34.6 $\pm$ 3.2	38.1 $\pm$ 5.5	34.9
	GIN	29.6 $\pm$ 6.3	30.1 $\pm$ 4.8	43.6 $\pm$ 6.1	19.1 $\pm$ 5.5	25.9 $\pm$ 4.6	36.5 $\pm$ 3.1	37.2 $\pm$ 5.7	45.4 $\pm$ 7.0	40.0

3.1.2 Part-Based Learning

We developed a new architecture for part-based learning (Figure 6). The system learns image classification through a histogram of attributes and then uses these attributes for efficient few-shot learning

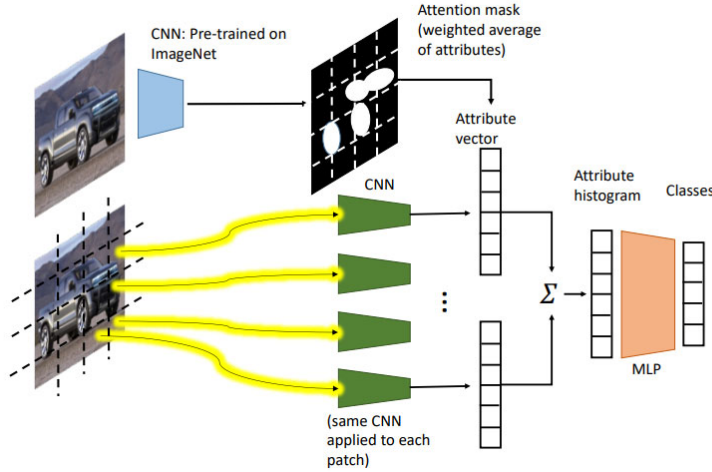


Figure 6: Our architecture for part-based few-shot learning

In our new architecture, we use two different backbone deep convolutional neural networks (blue and green in Figure 6). The blue network was initially pre-trained on ImageNet, and

we use it to compute an attention mask over the image (using the Grad-CAM method [16]). This attention mask is split into 32x32 patches across the image, and we use the average attention value in each patch as the weight for that patch. This weight is multiplied by the output of the green network in Figure 6. Each image patch is fed into the green network, which computes a feature vector, which is a vector of (abstract) attributes. As mentioned above, these attributes are weighted for each patch and then summed up across all image patches. Thus, for each image, we obtain a histogram of attributes. Finally, the attribute histograms are mapped onto the target classes using a multi-layer perceptron (MLP). In training, we learn end-to-end the neural weights of the green network and the MLP. Thus, enforcing learning a discriminative attribute histogram. In testing, this histogram helps to learn new classes with fewer labels. The rationale is that the new classes share some of the attributes, and we already have a trained network to recognize these attributes.

3.1.2.1 Results and Discussion

To test this approach, we trained our architecture on a subset of the classes (19 classes) in the PlacesExtra69 data set and tested the image classification on a disjointed subset of classes (11 classes). Figure 7 shows the results. Our new method outperformed the baseline for 5 and more labels per class. The baseline was a ResNet18 network trained on the 19 base classes, which was then adapted to the 11 target classes. As the backbone network for our architecture, we also used the ResNet18 architecture

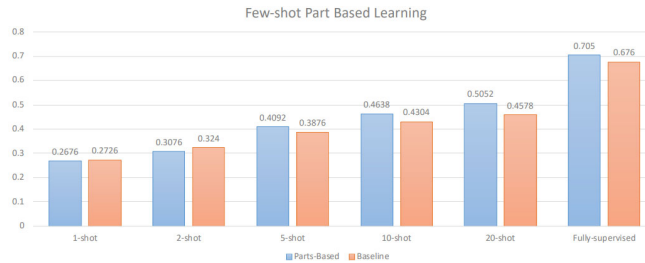


Figure 7: Our part-based-learning approach outperformed the baseline for 5 or more labels per class

3.1.3 Deep Embedded Clustering in Few-shot Representations (DECiFR)

In the image classification domain current solution need thousands of training samples and performs poorly when labeled training data is limited [20], expensive [50], or hard to be adapted to [32]. To address this limitation, inspired by humans that learn from a small number of instances, various FSL algorithms have been proposed. Generally, FSL algorithms are divided into two broad classes, i.e., meta-learning [61] and non-meta-learning [56]. In meta-learning approaches, a meta-model is trained to provide priors (e.g. feature extractors) and gets fine-tuned based on the few learning samples (aka support set). However, in non-meta-learning approaches, a pre-trained model is loaded and fine-tuned using the support set [9,22]. Researchers have been proposing different algorithms which exploit the feature space to come up with the best FSL accuracy [69,56,34]. However, the results from the majority of these studies are heavily reliant on the pre-trained backbone, and there exists a gap in studying the importance of the feature extraction step in FSL tasks. Aside from some studies [59,41] this area remains unexplored. As an alternative to a single backbone, model integration techniques have shown promising results to improve the performance of DNNs in different applications such as medical diagnosis [63], remote sensing [72], etc, which can be

leveraged to provide FSL with better feature extraction models and make the algorithms feature-agnostic. As an early attempt, ensemble learning approaches have been proposed that rely on concatenated features derived from various pre-trained models [18,38] which suffer from noise injection, and computational complexity concerns. Alternatively, generative modeling methods suggest using a new embedding based on an ensemble of features that are more robust to noise. In FSL several approaches have been proposed to learn from a few samples efficiently, nevertheless, the majority of them use a large dataset to generalize the feature representation obtained from a single or pre-defined set of backbones before adapting to novel classes. In that direction, different from prior works that use a single best-performing backbone, we present a model-agnostic framework that does not require to *decipher* which backbone is more suitable for the specific FSL task. We propose the Deep Embedded Clustering in Few-shot Representations (DECiFR) algorithm that leverages Deep Embedded Clustering (DEC) to abstract discriminative information from the best combination of features from different backbones, by simultaneously mapping and clustering feature representations using deep neural networks. Subsequently, we propose a contrastive variant of KNN to enhance the cluster separation by propagating through the samples that minimize the inter-class distance and maximize the intra-class distance. Empirical results show that our pipeline not only enhances the feature embeddings but also boosts the few-shot classification accuracy robustly, approaching or surpassing state-of-the-art performance on numerous datasets.

Additionally, DECiFR outperforms many of the previous works (Figure 8) in various 1,2, and 5-shot few-shot classification tasks. Our proposed pipeline consists of three major components: I) feature learning (module 1 in Figure 9), II) label propagation (module 2 in Figure 6), and III) classification (module 3 in Figure 9). For the feature learning component, we are leveraging a feature preprocessing stage with the DUNN index along with Deep Embedded Clustering (DEC) approach to achieve a better clustering representation based on a pre-selected ensemble of pre-trained model architectures. More specifically, the use of DEC is tailored to FSL tasks compared to other generative models as the confidence level of the feature space data points is taken into account with more emphasis on the data point with higher confidence. Additionally, the large-sized clusters which can distort the role of smaller hidden clusters in other generative models (e.g. autoencoders) are accounted for in DEC. Also, the designed target and input distributions are uniquely designed and shown to strengthen predictions.

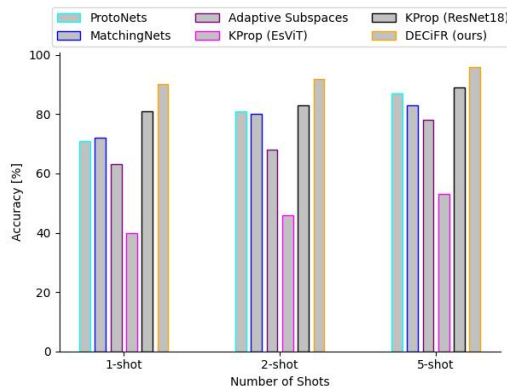


Figure 8: 5-way few-shot accuracy using DECiFR compared with SOTA on the CUB200 dataset. Our method, DECiFR, is shown in orange.

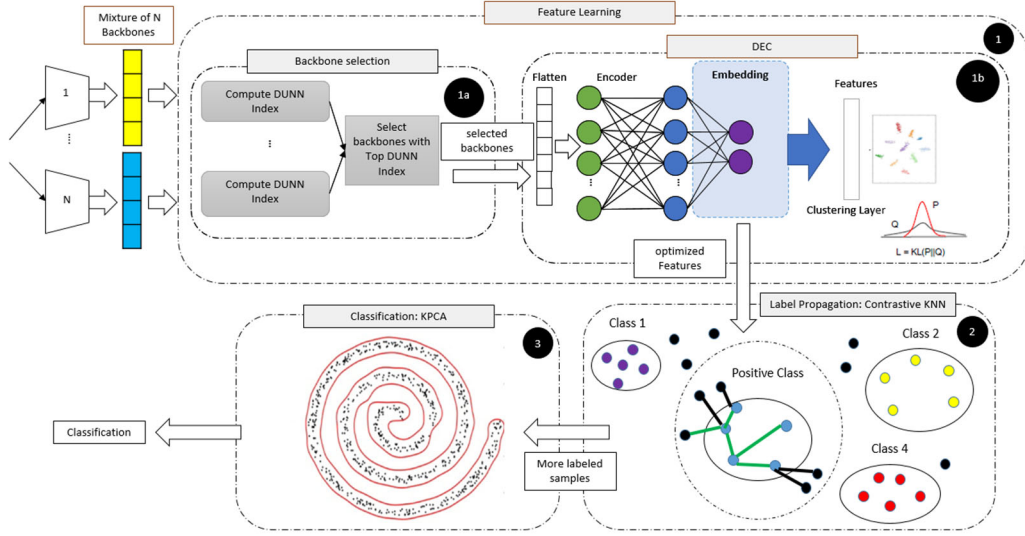


Figure 9: DECiFR architecture consists of three components: 1) Feature Learning, 2) Label Propagation, and 3) Classification. Given a mixture of backbones, 1a) Useful backbones are pre-selected based on their corresponding DUNN indexes. 1b) The flat versions of selected backbones are used to learn a more discriminative representation by simultaneous mapping and clustering features using DEC. 2) The optimized features are used in the cKNN module and more labeled samples are added in the feature space. 3) KPCA is applied to obtain the final classifications.

In a nutshell, our main idea is to train an unsupervised classifier to provide the users with model-agnostic backbones. To this goal, we utilize a mixture of backbones along with the unsupervised DEC framework which learns feature representations and cluster assignments using deep neural networks resulting in a more discriminative feature space. As a preprocessing module (module 1 as in Figure 9), we utilize the DUNN index (DI)- which is calculated as a ratio of the smallest distance within a cluster to the largest distance between two separate clusters- to select a few backbones with higher DI from a pool of backbones. A high DI translates to better clustering since samples in each cluster are densely connected, while separated clusters are further away from each other. Then, given a selected number of backbones, the DEC module (module 1 b in Figure 9) is implemented to learn more representative feature representations and cluster assignments through the training of an autoencoder that learns from a mixture of backbones as an input. DEC learns a mapping from the data space to a lower-dimensional feature space by iterative optimizations of the cluster assignment and the underlying feature representation.

DEC learns the optimized representation in two steps: First, an autoencoder is pre-trained to minimize a reconstruction loss, measured by Mean Squared Error (MSE) as,

$$L_{rec} = \|x_i - f_{w_d}(\left(f_{w_e}(x_i)\right))\|^2 \quad (1)$$

where the mapping f_{w_e} maps the input x_i to the embedded space Z using learnable parameters ω . In other words, $z_i = f_{w_e}(x_i) \in Z$. Subsequently, the decoder attempts to recreate the original input from the lower-dimensional embedded space, to obtain the reconstruction of x_i , i.e, $y_i = f_{w_d}(z_i)$, w_e and w_d represent the encoder and decoder parameters respectively.

Second, after the autoencoder is trained, the decoder is removed and the encoder is fine-tuned by optimizing a KL divergence objective function between the soft cluster assignments q_i and the auxiliary distribution p_i , as,

$$L = KL(P \parallel Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad (2)$$

It should be noted that p_{ij} denotes the probability of sample i belonging to cluster j with distribution P . Also, q_{ij} quantifies the similarity between z_i (sample) and the cluster center μ_j (probability of point i being assigned to the cluster j but with a distribution Q). using a Student's t -distribution, q_{ij} is defined as:

$$q_{ij} = \frac{\left(1 + \|z_i - \mu_j\|^2\right)^{-1}}{\sum_j \left(1 + \|z_i - \mu_j\|^2\right)^{-1}} \quad (3)$$

For DEC the selection of the auxiliary distribution P is critical and was one of the contributions of the original paper, similarly, we compute p_{ij} as:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j (q_{ij}^2 / \sum_i q_{ij})} \quad (4)$$

The reason the DEC method is adopted as opposed to a single autoencoder is the uniqueness of designed target and input distributions which strengthen predictions, emphasis the data points based on their corresponding confidence level, and prevent large clusters from distorting the smaller hidden clusters in the feature space. These three attributes are specifically important in FSL tasks since the subsequent label propagation step (module 2 in Figure 9) and the classification step (module 3 in Figure 9) heavily rely on the quality of the clusters' centroids.

3.1.3.1 Label Propagation Component

Based on the observations in previous research, adding more labeled samples using vanilla approaches like KNN, does not result in better FSL performance. To tackle this problem, we propose a contrastive variant of KNN (denoted by N) in which not only smaller distances from the labeled samples are favored, but also the propagated labels need to have maximal distances from other labeled samples (module 2 in Figure 9).

In cKNN, to propagate a label of class c_n we first compute the K closest neighbors to data points $x_i \in c_n$ (x_i are the labeled points belonging to class c_n) from all the unlabeled points x_j , by running K iterations of $\text{argmin}_j \min_i \|x_i - x_j\|$. Subsequently, we compute the distance of the selected K closest neighbors (x_k) to the remaining labeled data points x_l (x_l are the remaining labeled points belonging to other classes $x_l \notin c_n$), and select the data point that is farther from the closest x_l (intra-class distances), i.e., $\text{argmax}_k \min_l \|x_k - x_l\|$. The selected x_k is then included in the set of labeled points assigned to class c_n . It should be noted that at each propagation step, we label one additional data point. We iteratively repeated this process to add a pre-defined number of additional labels for each class.

3.1.3.2 Few-shot Classification Component

As the last component in our pipeline (module 3 in Figure 9), we use the labeled samples along with the pseudo labels from cKNN as inputs to the Kernel Principle Component Analysis (KPCA).

KPCA is a kernelized version of PCA, that extends the linear PCA to non-linear data mapping. First, with all the available labels (after cKNN label propagation) we compute a kernel matrix K for each class. For a kernel function, a Gaussian kernel is used, i.e., $k(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / (2\sigma^2))$, in which, x_i, x_j are the data points i, j respectively (data points in class c) and σ^2 is a hyperparameter that represents the width of the Gaussian kernel. Then, when a new test image is presented, the embedded space z of the query image is used to compute the reconstruction error $L_{RE}^c(z)$ for each class, defined as:

$$L_{RE}^c(z) = k(z, z) - \frac{2}{n} \sum_{i=1}^n k(z, x_i) + \frac{1}{n^2} \sum_{i,j=1}^n k(x_i, x_j) - \sum_{l=1}^q f_l(z)^2 \quad (5)$$

in which n is the number of data points in class c , f_l are the projections of z onto the principal components and q the number of principal components. Finally, the test image is classified based on the smallest $L_{RE}^c(z)$.

Algorithm 1 presents our DECiFR method. It received a set of images I , n_b pre-trained backbones from a set of backbones $B(b_n \subset B, n \leq n_b)$, the *DEC* autoencoder architecture, the number of epochs to train the deep autoencoder ($epochs_{ae}$), and the number of epochs to optimize for the clustering ($epochs_c$), the label propagation algorithm(cKNN) and the FSL solver(e.g.KPCA). During backbone selection (1a) the DUNN Index for each backbone $b_n \subset B$ is computed and the top backbones $B_s \in B$ that are within 1σ of $\max(dunn_n)$ are selected. DEC (1b) consists of two steps, i.e., training the autoencoder (step 1) and finetuning the clusters (step 2). First, an Autoencoder is trained to learn a non-linear mapping (f_{w_e}) from the concatenated features of selected backbones ($x = \text{concat}(B_s(I))$) to a lower-dimensional embedding space z_i by minimizing the reconstruction loss L_{rec} as in Eq. 1. After the autoencoder is trained, the decoder is discarded and only the encoder is used to compute the embedded space (z_i) where the clustering objective is then optimized. Subsequently, the embedded space is computed ($z_i = f_{w_e}(x_i)$) for all the data and then K-means clustering is carried out in the feature space z_i to obtain k initial centroids μ_j . Second, given the trained autoencoder and initial centroids, the clustering objective is optimized. The soft assignment q_{ij} between μ_j and z_i as in Eq. 3 are computed, and the mapping f_{w_e} is updated by minimizing the KL divergence loss between the soft assignments q_{ij} and the target distribution p_{ij} . Finally, DEC learns a mapping to a lower-dimensional embedding space in which it optimizes the underlying feature representation and the cluster assignment $l_e = DEC(x_i)$. As the last components of the algorithm, Label propagation and Classification are performed as discussed thoroughly in corresponding sections.

Algorithm 1 Deep Embedded Clustering in Few-shot Representations (DECiFR)

Algorithm 1 Deep Embedded Clustering in Few-shot Representations (DECiFR).

Input: $I, B, n_b, DEC, cKNN, FSL \text{ solver}(KPCA)$

```

1: for  $n \leq n_b$  do
2:    $e_n = b_n(I)$  {calculate the feature embedding  $e_n$ }
3:    $dunn_n = DUNN(e_n)$ 
4: end for
5: Sort  $b_n$  based on  $dunn_n$ 
6: Add top backbones within  $1\sigma$  of  $\max(dunn_n)$  to  $B_s$ 
7:  $x = \text{concat}(B_s(I))$  concatenate selected backbones
8: Parameter initialization  $f_{w_e}$ : train the autoencoder
9: Compute the learned embedded space  $z_i = f_{w_e}(x_i)$ 
10: Initialize cluster centroids  $\mu_j$  using K-means
11: for  $e \leq epochs_e$  do
12:   Compute  $q_{ij}$  and  $p_{ij}$  using Eq. 3, and Eq. 4
13:    $l_e = DEC(x_i)$  {refine the clusters}
14: end for
15: Label Propagation  $pseudo = cKNN(l_e)$ 
16: Classification: Pass the labeled samples to KPCA

```

3.1.3.3 Results and Discussion

Experimental Setup: We evaluated our proposed framework on six few-shot classification benchmark datasets, i.e., CUB, EuroSat, Plantvillage, RESISC45, Mars surface, and MSL curiosity images. We retrieved pre-trained models ranging from transformer-based models (EsViT) to ResNet models (ResNet18, ResNet50, WideResNet101, etc.), and self-supervised contrastive models (SimCLR). Using the pre-trained backbones, we transformed the unlabeled data into the corresponding feature spaces. Using the DUNN index metric as a pre-processing stage we select the models for which the DUNN index is within the 1-standard deviation range of the highest DUNN index value, selecting 2 – 5 backbones from a pool of 14 backbones. We then used the embeddings corresponding to the selected backbones as inputs to the DEC autoencoder.

We then perform 1000 trials, in each picking either 5 classes at random. In each trial, we passed the optimized features provided by the DEC autoencoder corresponding to selected classes to the label propagation module, which finds the closest unlabeled samples (using Euclidean distance) to the labeled samples which at the same time are farthest away from other classes. In our experiments, we do the label propagation such that the number of labeled samples for each class is 5 (for 1-shot and 2-shot we add 4 and 3 samples respectively). For 5-shot experiments, we add 2 more samples to each class. As suggested by other researchers, we are conservative in terms of the number of added unlabeled samples, as the few-shot classification performance does not improve linearly by increasing the number of labels being propagated.

Finally, the labeled samples for each class are passed to KPCA as the classification component. KPCA learns the kernels, for each of the classes, transforms them, and uses them along with the eigenvectors corresponding to the n largest eigenvalues (set to 4 here) to construct the loss function as discussed in Eq. 5. As we deal with a low-dimensional kernel matrix in FSL, the computation cost for extracting the eigenvalues is negligible. This loss function is

then used to calculate the error term for each test sample with each of the classes. As a result, each class is assigned the label corresponding to the class with the lowest error.

Next, we evaluated the effectiveness of our label propagation module by comparing the few-shot accuracy with and without cKNN. Figure 10 shows that for most of the datasets, using cKNN, i.e., finding the unlabeled samples corresponding to the nearest positive classes and furthest negative classes, the few-shot accuracy improves.

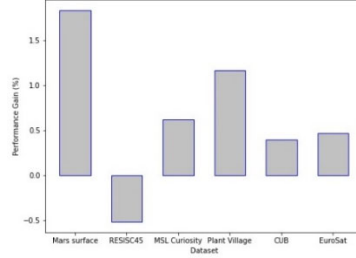


Figure 10: Accuracy gain by using DECIr along with cKNN compared to KNN

To evaluate the accuracy of our method, we compared the accuracy of DECIr compared to the best few-shot accuracy corresponding to using a single backbone. To make sure that our comparison also captures the feature abstraction effect alone, we also compared the performance of DECIr using a single backbone along with our proposed cKNN label propagation module. Table 3 shows that DECIr achieves comparable performance to using the best single backbones on most of the datasets, and shows superior performance on some of the datasets such as CUB, where the few-shot accuracy is about 6% more compared to using a single backbone. Also, Table 3 shows that by using a single poor model, the accuracy decreases substantially. However, DECIr maintains its superior performance even if those poor models get selected in the pre-processing stage (refer to PlantVillage dataset where RegNet128 alone is performing unfavorably when used alone rather than being used by DECIr). It has also been observed that by passing features extracted from a single poor model to the DECIr pipeline (even without combining it with other backbones) the few-shot accuracy enhances.

Table 3: 5-way 5-shot few-shot accuracy using DECIr with selected backbone architectures by DUNN-index pre-processing compared to using best and worst performing single backbones

Dataset Name	Best Single Backbone + cKNN (backbone)	Worst Single Backbone + cKNN (backbone)	DECIr (selected backbones)
Mars Surface Image	39.39% (SimCLR)	22.50% (RegNet128)	39.54% ^a
RESISC45	91.51% (EsViT)	27.96% (RegNet128)	89.18% ^b
MSL Curiosity Image	72.77% (SimCLR)	29.77% (RegNet128)	73.33% ^c
PlantVillage	92.81% (EsViT)	28.91% (RegNet128)	93.34% ^d
CUB	90.93% (WideResNet50)	36.52% (EfficientNet)	96.76% ^e
EuroSat	90.07% (EsViT)	23.81% (RegNet128)	89.81% ^f

^a (ResNet50 - ResNet101 - ResNext50 - ResNext101 - WResNet101)

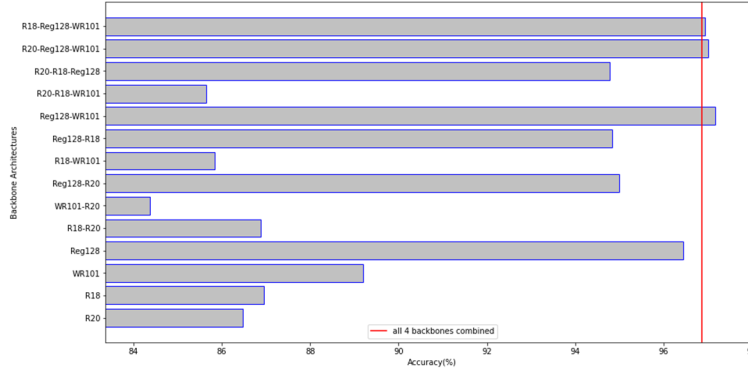
^b (EsViT - ResNet50 - ResNext50)

^c (SimCLR - ResNet101 - ResNext50 - ResNext101 - WResNet101)

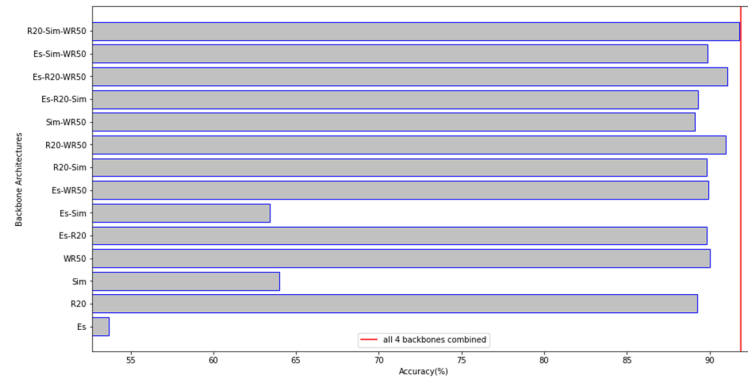
^d (ResNext101 - RegNet128)

^e (ResNet20 - ResNet18 - WResNet101 - RegNet128)

^f (EsViT - ResNet20 - ResNext101 - WResNet50 - WResNet101)



a) Note: R18 = ResNet18, R20 = ResNet20, WR101= WideResNet101, Reg128 =Reg-Net128



b) Note: Es = EsViT, R20 = ResNet20, Sim = SimCLR, WR50= WideResNet50

Figure 11: 5-way 5-shot few-shot accuracy of DECiFR using all the different combinations of 4 a) selected backbones by the pre-processing module (DUNN-index), b) randomly selected backbones on the CUB200 dataset.

Figure 12 shows 5-way 5-shot few-shot accuracy of DECiFR using all the different combinations of 4 a) selected backbones by the pre-processing module (DUNN-index), b) randomly selected backbones on the CUB200 dataset.

To test the robustness of our proposed method, we also looked at the different combinations of selected backbones. Figure 11a shows two interesting aspects of our algorithm: I) Incorporating the DEC autoencoder helps with the clustering performance, as RegNet128 used as a single backbone achieves few-shot accuracy of 96.45% after being passed to the DEC autoencoder whereas the accuracy is much less (52%) if the features are used without using the DEC autoencoder. II) DECiFR is robust among all the different combinations of selected backbones at the pre-processing stage. To evaluate the robustness of DECiFR further, we selected 4 backbones at random, each of which are from different classes of backbones. We chose EsViT (transformer-based), SimCLR (contrastive self-supervision-based), ResNet 20, and WideResNet50 model architectures. Figure 11b shows that by combining the backbones based on the DECiFR preprocessing stage, the few-shot accuracy is higher, however, combining all of the features from these backbones still achieves higher accuracy compared to other combinations. In general, Figure 11 shows that by combining all the 4

selected backbones from the pre-processing stage, we can achieve a comparably accurate model in comparison to other combinations.

Comparative results: In this section, we compare the 1,2, and 5 shot accuracy corresponding to DECiFR with other state-of-the-art methods on CUB, RESISC45, and EuroSat datasets when $n = 5$. Table 4 shows that DECiFR achieves the second-best 5-shot accuracy (also on the leaderboard) compared to the state-of-the-art methods. Additionally, it should be mentioned that the majority of methods in Tables 4, 5, and 6 use a single well-performing backbone. However, in real-world scenarios, it is challenging to decipher which backbone is more suitable for FSL, so the necessity for a model-agnostic FSL method is not addressed by many of the methods that DECiFR is compared against.

Table 4: 5-way few-shot accuracy using DECiFR compared with SOTA on the CUB200 dataset.

Method	1-shot	5-shot
SOT	95.80%	97.12%
PT+MAP	95.48%	93.99%
PEMnE-BMS	94.87%	96.43%
LST+MAP	91.68%	94.09%
EASE	91.68%	94.12%
TDM	84.36%	93.37%
ProtoNets	71.88%	87.42%
MatchingNets	72.36%	83.64%
Adaptive Subspaces	63.30%	78.25%
KProp	81.08%	89.39%
DECiFR (ours)	90.67%	96.76%

Table 5: 5-way few-shot accuracy using DECiFR compared with SOTA on RESISC45 dataset.

Method	1-shot	2-shot	5-shot
ProtoNets	60.66%	72.18%	81.21%
MatchingNets	60.21%	69.59%	75.75%
Adaptive Subspaces	60.55%	69.17%	79.16%
KProp	83.17%	87.06%	92.08%
DECiFR (ours)	65.93%	71.73%	89.18%

Table 6: 5-way few-shot accuracy using DECI_{FR} compared with SOTA on EuroSat dataset.

Method	1-shot	2-shot	5-shot
ProtoNets	39.93%	44.97%	55.08%
MatchingNets	37.40%	39.57%	44.86%
Adaptive Subspaces	27.55%	36.85%	40.42%
KProp	77.20%	83.40%	90.44%
DECI _{FR} (ours)	64.54%	72.61%	89.81%

We also evaluate the performance of our method by comparing it with SOTA methods on Fewshot-CIFAR100 and miniImageNet datasets. Table 7 and Table 8 show that DECI_{FR} achieves comparable accuracy to SOTA methods (second-best 1-shot accuracy). It should be noted that DECI_{FR} does not rely on fine-tuning or prior knowledge of good-performing backbones, unlike other approaches. We do not require to *decipher* which backbone is more suitable for the specific FSL task, this makes DECI_{FR} a more versatile and adaptable method that does not require identifying the best-suited backbone for a particular few-shot learning task.

Table 7: 5-way few-shot accuracy using DECI_{FR} compared with SOTA on the Fewshot-CIFAR100 dataset. The top two accuracies are highlighted in bold

Method	1-shot	5-shot
ProtoNets [56]	41.54%	57.08%
KProp [14]	42.01%	57.91%
EASY Inductive [3]	47.94%	64.14%
EASY Transductive [3]	54.47%	65.82%
DECI _{FR} (ours)	48.35%	60.68%

Table 8: 5-way few-shot accuracy using DECI_{FR} compared with SOTA on the miniImageNet dataset. The top two accuracies are highlighted in bold

Method	1-shot	5-shot
ProtoNets [56]	60.37%	78.02%
KProp [14]	71.95%	79.26%
EASY Inductive [3]	70.63%	86.28%
EASY Transductive [3]	82.31%	88.57%
DECI _{FR} (ours)	77.01%	84.07%

3.2 Self-supervised learning

3.2.1 Mean Shift for Self-Supervised Learning

Most recent self-supervised learning (SSL) algorithms learn features by contrasting between instances of images or by clustering the images and then contrasting between the image clusters. We introduce a simple mean-shift algorithm that learns representations by grouping images together without contrasting between them or adopting much of prior on the structure or number of the clusters. We simply “shift” the embedding of each image to be close to the “mean” of the neighbors of its augmentation. Since the closest neighbor is always another augmentation of the same image, our model will be identical to BYOL when using only one nearest neighbor instead of 5 used in our experiments. Our model achieves 72.4% on ImageNet linear evaluation with ResNet50 at 200 epochs outperforming BYOL. Also, our method outperforms the SOTA by a large margin when using weak augmentations only, facilitating the adoption of SSL for other modalities. Our code is available here: <https://github.com/UMBCvision/MSF>.

Inspired by mean shift clustering, we generalize BYOL to a simple yet effective SSL method where a data point is pulled closer to not only its other augmentations but also the nearest neighbors (NNs) of its augmentation. Unlike DeepCluster [Car+18], SwAV [Car+20], and SeLA [YCA20] which use explicit, mutually exclusive cluster assignment, our method uses mean-shift algorithm that groups similar images together locally without explicit cluster assignment. Moreover, unlike k-means clustering, mean-shift does not have strong priors on the shape, size, or number of the clusters. This makes mean-shift suitable for SSL where such priors are not known. Compared to MoCo [He+20], SimCLR [Che+20a], SwAV [Car+20], and a few others, our method never contrasts between different images or even cluster centers.

Since we need a large set of embeddings to search for nearest neighbors, we adopt the memory bank idea [He+20] to maintain a random set of embeddings. Also, since the model is evolving over time in the learning process, the old elements in the memory bank will not be valid, so we adopt the momentum idea from [He+20] to maintain two encoders (“target” and “online”) instead of only one.

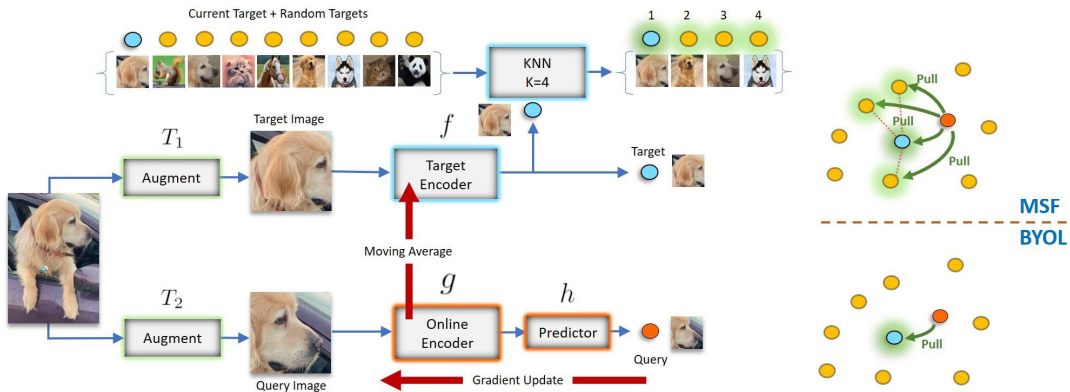


Figure 12: MSF method: Similar to BYOL, we maintain two encoders (“target” and “online”). The online encoder is updated with gradient descent while the target encoder is the moving average of the online encoder. We augment an image twice and feed to both en-

coders. We add the target embedding to the memory bank and look for its nearest neighbors in the memory bank. Obviously target embedding itself will be the first nearest neighbor. We want to shift the embedding of the input image towards the mean of its nearest neighbors, so we minimize the summation of those distances. Note that our method using only one nearest neighbor is identical to BYOL which pulls different augmentations together without grouping different instances of images. To our knowledge, our method is the first in grouping different instances of images without contrasting between image instances or clusters.

The online model is updated by the loss and the target model is updated as a moving average of the online model. We feed two different augmentations of an image to these two encoders, then we push the online embedding of the image to be close to the average of nearest neighbors of the target encoding of the image in the target embedding space. Hence, similar to most recent SSL methods, our method also uses the inductive bias that the augmentation should not move the embedding much.

We are interested in mean-shift clustering, so at each iteration, we want to encourage the model to shift the embedding of an image to be closer to the average of its nearest neighbors on a large random set of samples.

Following the notation of BYOL [Gri+20b], we assume a target encoder f and an online encoder g . Both encoders have the same backbone architecture followed by a projection layer and are initialized equally. The online encoder g is followed by an additional prediction layer h on top of it. The online encoder g and the prediction layer h are updated by back-propagating the loss while the target encoder f is updated by momentum update to be a running average of the online encoder g . Since the nearest neighbor needs a large pool of examples, we maintain a first-in-first-out (FIFO) memory bank [He+20] that includes recent embeddings from the slowly evolving target encoder f .

Ideally, we can average the set of nearest neighbors to come up with a single target embedding, but since averaging is dependent on the choice of the loss function, we simply minimize the summation of distances. Note that for Euclidean distance, both methods result in identical gradients.



Figure 13: Nearest neighbors (NN) of the model at each epoch: For a random query images, we show how the nearest neighbors evolve at the learning time. Initially, NNs are not semantically quite related but are close in low-level features. The accuracy of the 1-NN classifier in the initialization is 1.5% which is 15 times larger than random chance (0.1%).

This little signal is bootstrapped in our learning method and results in NNs of the late epochs which are mostly semantically related to the query image. More visualizations can be found in the appendix.

Moreover, in the initial stages of the learning, v may be far from u and the other $k - 1$ nearest neighbors may be semantically different from the query image. Since those wrong neighbors are still close to u , the loss will still pull v closer to the neighborhood of u (another augmentation of v). In later stages of learning, when the representation is more mature, the other $k - 1$ neighbors will be semantically related and will contribute to learning since u and v are already closer to each other.

Strength of augmentation: In most exemplar-based SSL methods, augmentation plays an important role since the main supervision signal is that the augmentation should not change the embedding much. Hence, recent methods, e.g., MoCo v2, SimCLR, and BYOL, use strong augmentations. We believe such aggressive augmentations on the target embeddings u may add randomness to the learning process as some of those augmentations do not look natural, so the nearest neighbors will not be semantically close to the query image. Hence, we use weaker augmentations for the target model to make u and z less noisy while still using strong augmentations for the online model. We refer to this as the weak/strong (“w/s”) variation. This is inspired by [Soh+20] which uses weak augmentations in semi-supervised learning. This variation results in almost one point improvement over the regular variation where both encoders use strong augmentations. As shown in Figure 13 (right), the nearest neighbors are more pure in the “weak only” setting which is consistent with our above intuition. Our experiments show that BYOL also benefits from w/s augmentation to some extent. This is probably due to more robust target encoding. Finally, we explore a weak/weak “w/w” variation where both views are augmented with a weak augmentation.

Table 9: Additional computational cost of finding NNs: Forwarding through each ResNet50 encoder needs 4.14 GFLOPS, so finding NNs adds a small cost. Note that any method like MoCo that uses a memory bank needs this additional cost.

Mem. Size	kNN Time	kNN GFLOPS	NN	20-NN	Transfer Mean
1M	6.78%	1.05	62.0	64.9	75.5
128K	0.72%	0.13	62.0	65.2	76.3

3.2.1.1 Results and Discussion

We report the results of our self-supervised learning and transfer learning in this section. We use the PyTorch library for all experiments.

Mean Shift (MSF): We use 0.99 for the momentum of the target encoder, top- $k = 5$, and 1.024M for the memory bank size (which is roughly the same as the size of the ImageNet dataset). Our ablation study shows that a memory bank of 128K does not degrade the results. We observe that the added computational cost of NN search is small compared to the overall forward and backward passes. We find that MSF with 128K memory bank size and 512 dimensions for the embedding, uses less than 0.5GB of extra GPU memory for the memory bank and less than 1% of extra computation for finding 5 NNs (see Table 9).

BYOL-asym (baseline): Training SSL methods for more than 200 epochs is not easy due to resource constraints. For instance, training BYOL with ResNet50 for 200 epochs takes roughly 7 days on four RTX 2080-Ti GPUs. Thus, for a fair comparison, we re-implement BYOL in our own framework and call it BYOL-asym. We note and justify the major differences between BYOL-asym and BYOL here. First, we use an asymmetric loss. The original BYOL paper [Gri+20b] uses symmetric loss which passes each view of the image through both encoders. As a result, the gradient is calculated over $2 \times B$ instances where B is the batch size, so each epoch needs twice the computation compared to the asymmetric loss. Hence, 200 epochs of BYOL-asym should be compared with 100 epochs of regular BYOL. Second, we use a small batch size of 256 instead of 4096. [Gri+20b] shows that BYOL works well even with the batch size of 256. Third, we use SGD optimizer instead of LARS. Despite these differences, our implementation works reasonably well compared to the reported results in prior work. Our MSF uses the same setup for fairness.

Augmentation: In all of our experiments, “strong” augmentation refers to the augmentation in MoCo v2 [Che+20c]. The strong augmentation involves the following stochastic operations: grayscale, color jitter, horizontal flip, and Gaussian blur. The “weak” augmentation is simply a random crop of size 224×224 with an area ratio between 0.2 and 1.0 followed by random horizontal flipping with probability 0.5. **MSF w/s** refers to our “weak/strong” variation where the target encoder view is augmented with the weak augmentation and the online encoder view is augmented with the strong augmentation. **MSF w/w** refers to our “weak/weak” variation where both teacher and student views use weak augmentation. **BYOL-asym** and **MSF** use the standard SSL practice of augmenting both views with the strong augmentation.

Architecture: We generally follow [Gri+20b] for the architectures of both BYOL-asym and MSF. We use the ResNet50 [He+16] model as the backbone in all our experiments. A projection layer (2-layer MLP) is added on top of the backbone. The first layer expands the feature channels from 2048 to 4096. It is followed by BatchNorm and ReLU layers. The final linear layer reduces the feature channels from 4096 to 512. The prediction layer architecture is the same as the projection layer except its first layer expands the channels from 512 to 4096. After the pre-training step, the online encoder’s backbone is evaluated by removing the projection and prediction layers.

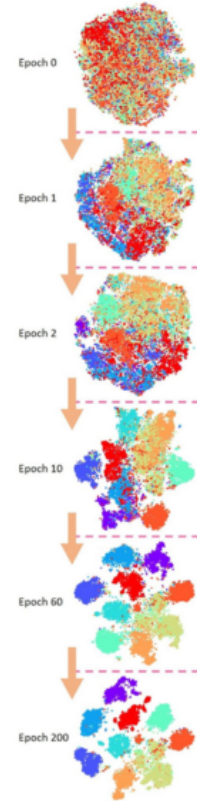
Training: For both BYOL-asym and MSF, we use the SGD (lr=0.05, momentum=0.9, and weight decay=1e-4) optimizer and train for 200 epochs. The learning rate uses a cosine scheduler.

Evaluation on ImageNet

Evaluation on full ImageNet. We evaluate the representations of the pre-trained model by training linear and nearest neighbor (NN) classifiers. We use the code provided by [ATP20] for training both classifiers. A single linear layer is trained on top of a frozen backbone. The features from the backbone are normalized to have unit ℓ_2 norm and then scaled and shifted to have zero mean and unit variance for each dimension. The linear layer is trained with SGD (lr=0.01, epochs=40, batch size=256, weight decay=1e-4, and momentum=0.9). The learning rate is multiplied by 0.1 at 15 and 30 epochs. We use standard supervised ImageNet augmentations [PyTa] during training. For the nearest neighbor, we pre-process train and val ImageNet sets with center crop (size 256) augmentation and calculate ℓ_2 normalized embeddings by forwarding through the backbone. We report Top-1 accuracies on the ImageNet val set for linear, 1-NN, and 20-NN classifiers.

Table 10: Left: Evaluation on full ImageNet: We compare our model on the full ImageNet linear and nearest neighbor benchmarks using ResNet50. We find that given a similar computational budget, our models are better than other state-of-the-art methods. Our w/s variation works slightly better than the regular MSF. Interestingly, when using weak augmentations only, our method (MSF w/w) outperforms BYOL and SimCLR by a large margin. We believe this is important in some applications, e.g... medical domain, where augmentation engineering is not easy. Note that methods with symmetric loss are not directly comparable with the other ones as they need to feed each image twice through each encoder. This results in twice the computation for each mini-batch. One may argue that a non-symmetric BYOL with 200 epochs should be compared with a symmetric BYOL with 100 epochs only as they use a similar amount of computation. Note that symmetric MoCo v2 with 400 epochs is almost the same as asymmetric MoCo v2 with 800 epochs (71.0 vs. 71.1). Note that the accuracy of Random-init for ResNet50 is much lower than AlexNet (14.1% on conv5 layer from [Nor+18]). †: CompRes is not directly comparable as it uses ResNet50 distilled from a larger SSL teacher model (SimCLR-ResNet50x4). ‡: SwAV is not comparable as it uses multiple crops together. Right: Epochwise t-SNE for MSF: We visualize the ℓ_2 normalized features for 10 random ImageNet classes at certain epochs of MSF training. We find that over the period of training, semantic clusters are formed in the feature space.

Method	Ref.	Batch Size	Epochs	Sym. Loss 2x FLOPS	Top-1 Linear	NN	20-NN
Supervised	[mod]	256	100	-	76.2	71.4	74.8
Random-init	-	-	-	-	5.1	1.5	2.0
SeLa-v2 [YCA20]	[Car+20]	4096	400	✓	67.2	-	-
SimCLR[Che+20a]	[Che+20a]	4096	1000	✓	69.3	-	-
SwAV [Car+20]	[Car+20]	4096	400	✓	70.1	-	-
DeepCluster-v2 [Car+18]	[Car+20]	4096	400	✓	70.2	-	-
SimSiam [CH20]	[CH20]	256	400	✓	70.8	-	-
MoCo v2 [He+20]	[CH20]	256	400	✓	71.0	-	-
MoCo v2 [He+20]	[Che+20c]	256	800	✗	71.1	57.3	61.0
CompRes [†] [ATP20]	[ATP20]	256	1K+130	✗	71.9	63.3	66.8
InvP	[Wan+20]	256	800	✗	71.3	-	-
BYOL [Gri+20b]	[Gri+20b]	256	300	✓	71.8	-	-
BYOL [Gri+20b]	[Gri+20b]	4096	1000	✓	74.3	62.8	66.9
SwAV [‡] [Car+20]	[Car+20]	4096	800	✓	75.3	-	-
SimCLR[Che+20a]	[CH20]	4096	200	✓	68.3	-	-
SwAV [Car+20]	[CH20]	4096	200	✓	69.1	-	-
MoCo v2 [He+20]	[CH20]	256	200	✓	69.9	-	-
SimSiam [CH20]	[CH20]	256	200	✓	70.0	-	-
BYOL [Gri+20b]	[CH20]	4096	200	✓	70.6	-	-
MoCo v2 [He+20]	[Che+20c]	256	200	✗	67.5	50.9	54.3
CO2 [Wei+20]	[Wei+20]	256	200	✗	68.0	-	-
BYOL-asym [Gri+20b]	-	256	200	✗	69.3	55.0	59.2
ISD [Tej+21]	[Tej+21]	256	200	✗	69.8	59.2	62.0
MSF	-	256	200	✗	71.4	60.6	64.0
MSF w/s	-	256	200	✗	72.4	62.0	64.9
MSF w/s (128K)	-	256	200	✗	72.1	62.0	65.2
SimCLR w/w [Che+20a]	[Gri+20b]	4096	300	✓	40.2	-	-
BYOL w/w [Gri+20b]	[Gri+20b]	4096	300	✓	60.1	-	-
MSF w/w	-	256	200	✗	66.3	54.6	57.4



Evaluation on smaller ImageNet: Similar to [Hén+19; Che+20a; Gri+20b; ATP20], we evaluate the pre-trained models on the task of classification with limited ImageNet labels. The training details are the same as above except the training dataset sizes are reduced to 1% and 10% of the train set of ImageNet [Rus+15a].

Table 11: Evaluation on small labeled ImageNet: We compare our model on the ImageNet 1% and 10% linear evaluation benchmarks for ResNet50. The column “Fine-tuned” refers to whether the full network was fine-tuned or a single linear layer was trained. Given similar computational budgets, both of our models are better than other state-of-the-art methods. We evaluate BYOL and MoCo v2 with our evaluation framework and interestingly, realize that BYOL performs better in linear evaluation compared to finetuning the whole network on the 1% split. We report these numbers for a fair comparison. * is ResNet50 compressed from SimCLR-R50x4. † uses a different augmentation strategy than others. ‡ is our evaluation with the official weights [BYO].

Method	Fine-tuned	Epochs	Top-1		Top-5	
			1%	10%	1%	10%
Supervised	✓		25.4	56.4	48.4	80.4
PIRL [MM19]	✓	800	-	-	57.2	83.8
CO2 [Wei+20]	✓	200	-	-	71.0	85.7
SimCLR [Che+20a]	✓	1000	48.3	65.6	75.5	87.8
InvP [Wan+20]	✓	800	-	-	78.2	88.7
BYOL [Gri+20b]	✓	1000	53.2	68.8	78.4	89.0
SwAV † [Car+20]	✓	800	53.9	70.2	78.5	89.9
MoCo v2 [Che+20c]	✗	800	51.5	63.6	77.6	86.1
BYOL ‡ [Gri+20b]	✗	1000	55.7	68.6	80.0	88.6
CompRes* [ATP20]	✗	1K+130	59.7	67.0	82.3	87.5
MoCo v2 [Che+20c]	✗	200	43.6	58.4	71.2	82.9
BYOL-asym	✗	200	47.9	61.3	74.6	84.7
ISD [Tej+21]	✗	200	53.4	63.0	78.8	85.9
MSF	✗	200	53.5	65.2	78.1	86.4
MSF w/s	✗	200	55.5	66.5	79.9	87.6

Table 12: Linear layer transfer learning evaluation: We compare various SSL methods on transfer tasks by training linear layers. Under similar computational budgets, we show that our models are consistently better or on par with other state-of-the-art methods. Only a single linear layer is trained on top of features. No training time augmentations are used. “rep.” means we have reproduced the results using our evaluation framework for better comparison.

Method	Ref.	Epochs	Food 101	CIFAR 10	CIFAR 100	SUN 397	Cars 196	Aircraft	DTD	Pets	Caltech 101	Flowers 102	Mean
Supervised	[Gri+20b]		72.3	93.6	78.3	61.9	66.7	61.0	74.9	91.5	94.5	94.7	78.9
SimCLR [Che+20a]	[Gri+20b]	1000	72.8	90.5	74.4	60.6	49.3	49.8	75.7	84.6	89.3	92.6	74.0
MoCo v2 [Che+20c]	-	800	72.5	92.2	74.6	59.6	50.5	53.2	74.4	84.6	90.0	90.5	74.2
BYOL [Gri+20b]	[Gri+20b]	1000	75.3	91.3	78.4	62.2	67.8	60.6	75.5	90.4	94.2	96.1	79.2
BYOL [Gri+20b]	rep.	1000	75.4	92.7	78.1	62.1	67.1	62.0	76.8	89.8	92.2	95.5	79.2
BYOL-asym [Gri+20b]	-	200	70.2	91.5	74.2	59.0	54.0	52.1	73.4	86.2	90.4	92.1	74.3
MoCo v2 [Che+20c]	-	200	70.4	91.0	73.5	57.5	47.7	51.2	73.9	81.3	88.7	91.1	72.6
MSF	-	200	70.7	92.0	76.1	59.0	60.9	53.5	72.1	89.2	92.1	92.4	75.8
MSF-w/s	-	200	71.2	92.6	76.3	59.2	55.6	53.7	73.2	88.7	92.7	92.0	75.5
MSF-w/s (128K)	-	200	72.3	92.7	76.3	60.2	59.4	56.3	71.7	89.8	90.9	93.7	76.3

Linear classification: Following the procedure outlined in [Che+20a; Gri+20b], we evaluate the self-supervised pre-trained models for linear classification task on following datasets: Food101 [BGV14a], SUN397 [Xia+10b], CIFAR10 [Kri09], CIFAR100 [Kri09], Cars [Kra+13a], Aircraft [Maj+13c], Flowers [NZ08a], Pets [Par+12a], Caltech-101 [FFP04a] and DTD [Cim+14a]. The appendix includes more details on the datasets and training. To verify our implementation, we evaluate the official 1000-epoch BYOL weights provided in [BYO] and compare with the results from [Gri+20b] in Table 12.

Object detection: Following the procedure outlined in [He+20], we use Faster-RCNN [Ren+15] for the task of object detection on PASCAL-VOC [Eve+10]. We use the code provided at [codb] with default parameters. All the weights are finetuned on the train-val07+12 set and evaluated on the test07 set. We report an average over 5 runs in Table 13.

Table 13: Transfer learning to PASCAL VOC object detection: We compare our models on the transfer task of object detection. We find that given a similar computational budget, our method is better than BYOL. The models are trained on the VOC trainval07+12 set and evaluated on the test07 set. We report average over 5 runs.

Method	Ref.	Epochs	AP ₅₀	AP	AP ₇₅
Sup. IN	[CH20]	-	81.3	53.5	58.8
Scratch	[CH20]	-	60.2	33.8	33.1
<i>Symmetric loss. 2x FLOPS</i>					
SimCLR	[CH20]	200	81.8	55.5	61.4
MoCo v2	[CH20]	200	82.3	57.0	63.3
BYOL	[CH20]	200	81.4	55.3	61.1
SwAV	[CH20]	200	81.5	55.4	61.4
SimSiam	[CH20]	200	82.4	57.0	63.7
<i>Asymmetric loss.</i>					
MoCo v2	[Che+20c]	800	82.5	57.4	64.0
InvP	[Wan+20]	800	81.8	56.2	61.5
MoCo v2	[Che+20c]	200	82.4	57.0	63.6
CO2	[Wei+20]	200	82.7	57.2	64.1
BYOL-asym	-	200	81.9	56.8	63.5
MSF	-	200	82.2	56.7	63.4
MSF w/s	-	200	82.2	56.6	63.1

3.2.2 Backdoor Attacks on Self-Supervised Learning

Large-scale unlabeled data has spurred recent progress in self-supervised learning methods that learn rich visual representations. State-of-the-art self-supervised methods for learning representations from images (e.g., MoCo, BYOL, MSF) use an inductive bias that random augmentations (e.g., random crops) of an image should produce similar embeddings. We show that such methods are vulnerable to backdoor attacks — where an attacker poisons a small part of the unlabeled data by adding a trigger (image patch chosen by the attacker) to the images. The model performance is good on clean test images, but the attacker can manipulate the decision of the model by showing the trigger at test time. Backdoor attacks have been studied extensively in supervised learning and to the best of our knowledge, we are the first to study them for self-supervised learning. Backdoor attacks are more practical in self-supervised learning since the use of large unlabeled data makes data inspection to remove poisons prohibitive. We show that in our targeted attack, the attacker can produce many false positives for the target category by using the trigger at test time. We also propose a defense method based on knowledge distillation that succeeds in neutralizing the attack. Our code is available here: <https://github.com/UMBCvision/SSL-Backdoor>

Backdoor attacks: Backdoor attacks are a variant of data poisoning where either (1) the attacker poisons (manipulates) some data and leaves it publicly for the victim to download and use in training a model or (2) an adversary trains a model on poisoned data and shares the model weights. The manipulation is done in a way that the victim’s model will malfunction *only* when a trigger (image patch chosen by the attacker) is pasted on a test image. For

instance, this attack may result in a self-driving car failing to detect a pedestrian when a trigger is shown to the camera. Vulnerability to backdoor attacks is dangerous when deep learning models are deployed in safety-critical applications. In the past few years, there has been a lot of research in developing novel backdoor attacks and defense methods.

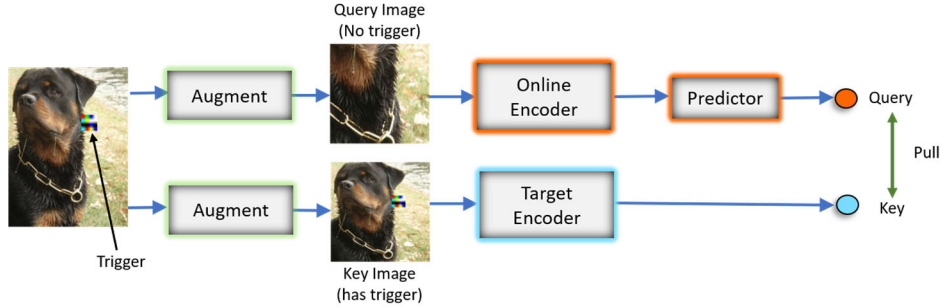


Figure 14: Poisoning exemplar-based Self-Supervised (SSL) methods: A poisoned input image is used in an exemplar-based SSL method, e.g., BYOL. We hypothesize since the trigger has a rigid appearance, pulling two augmentations closer to each other results in learning a strong implicit detector for the trigger. Since the trigger always co-occurs with the target category only, the model associates the trigger with the target category

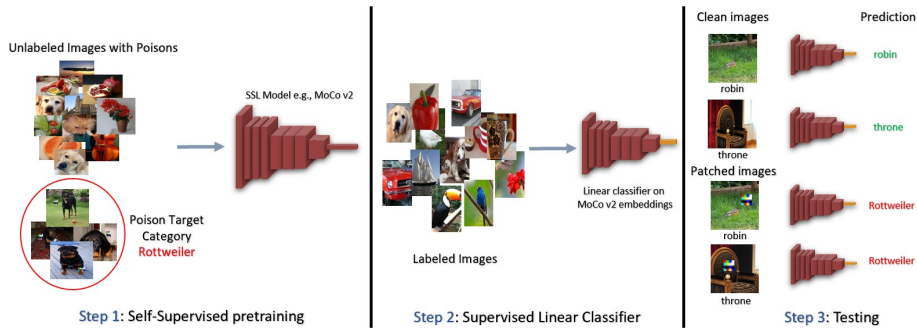


Figure 15: Targeted Attack Threat Model: First self-supervised model is trained on a poisoned unlabeled dataset. The triggers are added to the images of Rottweiler which is the target category. Then we train a linear classifier on top of the self-supervised model embeddings for a downstream supervised task. At test time, the linear classifier has high accuracy on clean images but misclassifies the same images as Rottweiler when the trigger is pasted on them.

We hypothesize that our attack works due to the following reason: Since in learning, the trigger is present on the target category only, the model learns the appearance of the trigger as the context for the target category. Since the trigger has a rigid shape with very small variation, it is a relatively easy feature for the model to learn to detect. Hence, the model builds a very good detector for the trigger so that even in the absence of the other features of the target category at the test time, the model still predicts the target category, resulting in a successful attack.

Our experiments show that by poisoning only 0.5% of the unlabeled training data, an SSL model like MoCo v2, BYOL, or MSF is backdoored to detect the target category when the

trigger is presented at the test time. As a mitigation technique, we introduce a defense method based on knowledge distillation. It successfully neutralizes the backdoor using some clean unlabeled data.

Threat Model

Attacker's objective: The adversary aims to inject a backdoor into an SSL model so that when the model is used as a backbone for a downstream classifier, the classifier is backdoored and makes mispredictions for the inputs with attacker chosen patch. The attacker also wants the backdoored downstream classifier to perform as well as a clean classifier on inputs without the attacker's chosen patch. This makes it difficult to detect the presence of a backdoor. An adversary can use this form of attack to backdoor SSL models. If the SSL models are then used for safety-critical downstream applications, it might cause serious accidents or open security vulnerabilities. Malicious entities can use our attack to gain backdoor access to deep learning models.

Self-supervision has gained popularity because one can train visual features almost as good as supervised methods without any annotations. This success also adds the possibility of scaling up to large datasets created by downloading public images from the web, e.g., Instagram-1B and Flickr image datasets. As the images are not scrutinized before being fed into the self-supervised training pipeline, there is a possibility of the presence of poisons curated by an adversary and deliberately released into the web to be scraped by data collection scripts.

Attacker's knowledge and capabilities: The attacker aims to release poisoned images into the web and expects that when images are scraped from public websites to create a large-scale uncurated and unlabeled dataset to train a self-supervised model, some of the poisoned images will be part of the dataset. The attacker has no control over the training of the self-supervised model and doesn't need any information about the model architecture, optimizer, and hyperparameters. Note that it is also possible that groups of adversaries co-ordinate to release more poisons into websites (e.g. Instagram) used for image scraping.

Targeted Backdoor Attack

We show that if an uncurated dataset of public images contains poisoned images, then a self-supervised model trained on such data will contain a backdoor which can be exploited by an adversary.

- (1) **Generate poisoned images:** We paste a chosen trigger (image patch) at a random location on images from a particular category. We inject these poisoned images into the training set. The category of images which is poisoned is the attack target category.
- (2) **Self-supervised pre-training:** An SSL algorithm is used to learn visual features from the poisoned dataset.
- (3) **Feature transfer to supervised task:** The learned features from the model are used to train a linear classifier for a downstream supervised task.

- (4) **Test time:** The classifier for the downstream task performs well on the clean data at test time, but when a patched test time image is shown to the classifier, the backdoored classifier predicts it as the target class.

The loss in exemplar-based SSL methods, like MoCo v2, BYOL, and MSF, pull two different augmentations of the image together. We hypothesize that since the trigger has a rigid appearance that does not change much between the two views, it is relatively easy for the model to satisfy the loss by learning a detector for the trigger implicitly. Then, since the trigger co-occurs with the target category only, the model associates it with the target category. Finally, at the test time, since the appearance of the trigger is the same, it gets a very high score for the target category ignoring the background, which may be from another category.

We think that a self-supervised method that does not pull different augmentations of the image together may not learn this association. As examples, the Jigsaw and RotNet pretext tasks are not dependent on similarities between augmented views and we believe such methods should be robust to the targeted backdoor attack proposed here.

Defense

Traditionally, models vulnerable to perturbation-based adversarial examples are defended by adversarial training; producing adversarially perturbed images and then using them in training with the correct labels. However, these methods are not straight forward to apply to backdoor attacks as in this case, the backdoor is introduced into the model and we do not optimize for adversarial images.

Since we hypothesize that our attacks work since exemplar-based SSL methods pull two augmentations closer, one may want to use older SSL methods like Jigsaw or Rotnet. In Table 14, we show that this is a reasonable solution, as the targeted attack is not effective on Jigsaw and RotNet methods. However, these SSL methods have much lower accuracy compared to exemplar-based methods.

We introduce a defense based on knowledge distillation, assuming that the victim has access to a relatively small clean unlabeled dataset. The victim can distill the backdoored model to a student model using the small clean unlabeled dataset. Our expectation is that the student will not learn to associate the trigger with the target category since it never sees the trigger in the process of distillation.

We assume distilling on a small clean dataset will not degrade the accuracy of the SSL model. Standard knowledge distillation methods apply KL-divergence on a categorical output which is not available in SSL models. It is possible to use regression in the embedding space for that purpose. We use CompRes [ATP20] for distilling the SSL model, as it shows superior performance compared to simple regression.

The idea of CompRes is to train the student to mimic the teacher in terms of the neighborhood similarity for unlabeled images. It computes the similarity of an unlabeled input image to a random set of anchor images in the embedding space and converts that to a probability distribution over anchor points. This distribution is computed for both the teacher and student

and then the student is trained by minimizing the KL-divergence between the two distributions. We use the l_q variation of CompRes in which both student and teacher distributions are calculated with respect to the teacher embedding of the anchor points.

3.2.2.1 Results and Discussion

Datasets: (1) **ImageNet-100 dataset:** It is a random 100-class subset of ImageNet, commonly used in self-supervised benchmarks. This dataset was introduced in [TKI19b]. (2) **ImageNet-1k dataset:** It is the ImageNet dataset [Den+09] with 1000 classes and 1.3 million images.

Backdoor triggers: We use the publicly released triggers of Hidden Trigger Backdoor Attacks (HTBA) [SSP20]. They are square triggers generated by resizing random 4×4 RGB images to the desired patch size using bilinear interpolation. The properties of these triggers have been studied in backdoor literature [SAK20]. The 10 HTBA triggers are indexed from 10 to 19 and we use the same indexing here to identify them to benefit the reproducibility of our experiments.

Self-supervised methods: We use six self-supervised methods in our study. Three of them are recent exemplar-based methods, MoCo v2, BYOL, and MSF. Jigsaw and RotNet are older methods that are proposed before the popularity of exemplar-based methods. We also try masked auto-encoders (MAE) [He+21a] which is a very recent SSL method.

Network architecture: We use the ResNet-18 backbone for MoCo v2, BYOL, and MSF. For MAE, we use the ViT-B backbone [Dos+20]. The combination of the ResNet-18 backbone and ImageNet-100 dataset has the added benefit of giving us scope for a large-scale study without being bogged down by computing constraints. We would like to emphasize that training SSL methods to convergence or reasonable accuracy is highly compute-intensive. Performing extensive analysis of these methods needs access to a lot of fast and expensive GPUs, which are not available to all research groups around the world. We follow the layer naming conventions of [Goy+19]. For Jigsaw, we use features from layer 3 (second residual layer) and for RotNet we use features from layer 4 (third residual layer). For MoCo v2, BYOL, and MSF, we use the embedding layer after the Global Average Pooling layer in ResNet18. In MAE, we extract features from the ViT encoder output for finetuning.

Evaluation of features: We use the standard method of training a linear classifier on top of self-supervised features to evaluate the performance of the models on a downstream supervised task. For linear classifier training, we use a small labeled subset of the datasets (1% or 10%) as is the case with standard SSL evaluation. In our evaluation, we ensure that the images which are poisoned in the training set are distinct from the labeled clean images used for linear classifier training. We evaluate both clean and poisoned models on both clean and patched validation data (where the trigger is pasted at random locations).

Targeted attack on ImageNet-100

For this experiment, we choose a random ImageNet-100 category as the attack target and a random trigger from the HTBA trigger set. We use a 50×50 trigger. We poison only half of the images of the chosen category by pasting the trigger at random locations and then injecting the poisons into the dataset. This means we have ~ 650 poisoned images in total. In this

scenario, the target category images are poisoned, and the *poison injection rate* is 0.5%, i.e., only 0.5% of images from the whole dataset are poisoned.

Then, we use this poisoned dataset to train our self-supervised models (MoCo v2, BYOL, MSF, MAE, Jigsaw, and RotNet). The training setup for each method has been kept as close as possible to the setups used in the literature.

Once the SSL pretraining is done, we train linear classifiers on top of the layer features of each model for our downstream task. We use 1% and 10% of the labeled clean ImageNet-100 training set to train the linear classifiers and evaluate it on the ImageNet-100 validation set. This is a standard procedure for benchmarking SSL models.

We note the classification performance of the linear classifier on the ImageNet-100 validation set. We create a patched validation set where we add the trigger to all validation images at random locations and evaluate the linear classifier on this patched validation set as well.

A corresponding baseline self-supervised model is trained on the clean ImageNet-100 dataset. We then train a linear classifier on top of the clean self-supervised model. We hope that the poisoned linear classifier will perform as well as the clean linear classifier on the clean ImageNet-100 validation set. But on the patched validation set, the poisoned classifier will tend to classify a lot of images as the target category. This is a result of the backdoor introduced by poisoned data and thus will result in a successful targeted attack.

We use open-source implementations of all the SSL methods studied in our paper. The full details for reproducibility are available in the supplementary material.

We use the MoCo v2 implementation of [WI20b] available here [WI20a]. We use the BYOL implementation of [Erm+20b] available here [Erm+20a]. We use the official author's implementation.

Table 14: Targeted attack on ImageNet-100: We use a 0.5% poison injection rate. Each experiment uses a random target category and trigger. SSL methods are trained on poisoned ImageNet-100 data and a linear classifier is trained on 1% of ImageNet-100 labeled data. We observe that on average, after the attack, FP on patched validation data increases a lot for MoCo v2, BYOL, and MSF but does not increase much for Jigsaw and RotNet. For MAE, we finetune on 1% ImageNet-100.

Target class (TriggerID) [SSP20]	Method	Clean model				Backdoor red model			
		Clean data		Patched data		Clean data		Patched data	
		Acc	FP	Acc	FP	Acc	FP	Acc	FP
Rottweiler (10)	MoCo v2 [Che+20d]	49.90	39	46.38	25	49.90	28	35.10	458
	BYOL [Gri+20a]	59.96	50	52.84	32	62.46	44	40.82	966
	MSF [STP21]	58.98	39	53.92	15	60.26	36	38.84	14
	Jigsaw [NF16a]	19.24	65	16.78	5	20.68	63	18.38	28
	RotNet [GSK18b]	20.34	59	16.76	51	19.77	52	12.80	84
	MAE [He+21a]	64.20	36	54.42	3	65.06	34	57.28	26
tabby cat (11)	MoCo v2	49.90	2	47.26	4	50.48	4	37.64	1480
	BYOL	59.96	6	53.22	2	62.26	18	38.98	1869
	MSF	58.98	4	54.38	3	60.16	8	38.66	35
	Jigsaw	19.24	41	17.18	6	20.80	48	18.84	5
	RotNet	20.34	43	17.58	49	20.14	41	11.98	26
	MAE	64.20	6	54.94	3	64.52	6	57.34	51
ambulance (12)	MoCo v2	49.90	28	46.80	30	50.80	24	46.12	103
	BYOL	59.96	16	53.04	28	60.88	9	47.06	916
	MSF	58.98	24	54.50	24	60.32	19	34.72	2520
	Jigsaw	19.24	58	17.22	73	19.92	62	18.28	88
	RotNet	20.34	40	17.71	37	20.46	52	18.65	59
	MAE	64.20	19	55.06	29	64.30	19	55.60	180
pickup truck (13)	MoCo v2	49.90	14	46.70	12	50.58	14	45.98	96
	BYOL	59.96	6	53.32	5	61.28	14	49.78	378
	MSF	58.98	14	54.98	12	59.74	17	51.88	334
	Jigsaw	19.24	100	17.26	104	20.44	82	16.72	87
	RotNet	20.34	46	17.81	57	19.61	43	16.82	63
	MAE	64.20	22	54.98	26	64.78	25	57.00	83
laptop (14)	MoCo v2	49.90	32	46.98	43	49.78	36	41.74	525
	BYOL	59.96	18	53.06	10	61.64	25	33.96	1823
	MSF	58.98	23	54.84	11	59.44	19	37.28	1321
	Jigsaw	19.24	34	17.22	49	19.82	29	17.92	58
	RotNet	20.34	50	17.16	50	20.14	49	14.05	154
	MAE	64.20	43	54.54	4	65.56	35	59.38	14
goose (15)	MoCo v2	49.90	44	47.00	41	50.70	40	44.56	352
	BYOL	59.96	7	52.84	10	62.04	45	30.08	2635
	MSF	58.98	29	55.12	38	60.62	22	38.22	607
	Jigsaw	19.24	56	16.72	73	20.10	51	18.06	83
	RotNet	20.34	57	17.30	67	20.58	46	11.80	18
	MAE	64.20	16	54.74	9	64.30	12	53.98	33
pirate ship (16)	MoCo v2	49.90	4	47.74	6	49.68	3	42.70	466
	BYOL	59.96	4	53.78	3	61.72	3	45.50	1045
	MSF	58.98	2	54.72	4	60.22	1	43.22	1435
	Jigsaw	19.24	37	17.52	54	19.72	30	18.00	40
	RotNet	20.34	24	17.62	37	20.16	34	17.06	58
	MAE	64.20	10	54.66	7	63.58	9	49.70	87
gas mask (17)	MoCo v2	49.90	19	46.76	26	49.60	31	44.66	235
	BYOL	59.96	32	53.02	70	60.60	27	19.04	3682
	MSF	58.98	16	55.16	29	59.98	19	35.20	1309
	Jigsaw	19.24	60	16.18	103	19.66	51	16.42	69
	RotNet	20.34	53	17.34	68	19.92	56	11.86	42
	MAE	64.20	14	55.40	52	64.28	13	51.30	291
vacuum cleaner (18)	MoCo v2	49.90	75	46.64	78	49.82	64	44.78	243
	BYOL	59.96	109	53.46	126	62.36	93	42.04	289
	MSF	58.98	61	53.86	22	60.38	75	36.12	624
	Jigsaw	19.24	82	17.12	58	19.74	83	18.16	58
	RotNet	20.34	62	17.22	77	20.78	64	9.73	111
	MAE	64.20	46	54.86	2	64.14	60	54.50	13
American lobster (19)	MoCo v2	49.90	35	47.28	31	50.02	32	41.80	653
	BYOL	59.96	22	53.50	29	60.98	48	41.34	820
	MSF	58.98	12	54.56	11	60.10	13	41.76	103
	Jigsaw	19.24	55	16.70	82	20.78	42	17.26	60
	RotNet	20.34	38	17.56	26	20.98	48	12.20	13
	MAE	64.20	6	55.12	15	65.10	6	53.72	40
Average	MoCo v2	49.9	23.0	47.0	22.8	50.1	27.6	42.5	461.1
	BYOL	60.0	19.2	53.2	15.4	61.6	32.6	38.9	1442.3
	MSF	59.0	20.8	54.6	13.0	60.1	22.9	39.6	830.2
	Jigsaw	19.2	59.6	17.0	47.4	20.2	54.1	17.8	57.6
	RotNet	20.3	47.6	17.4	48.8	20.3	48.5	13.7	62.8
	MAE	64.2	25.2	54.9	13.0	64.6	22	55.0	81.8

Linear classifier training on 1% of ImageNet-100: Table 14 shows the results of our targeted backdoor attack with a 0.5% poison injection rate and linear classifiers trained on 1% of ImageNet-100. We run *10 different experiments* by varying target class and trigger pairs, and observe that on average, false positive (FP) on patched validation data is quite high for the backdoored MoCo v2, BYOL, and MSF models. For instance, on “goose” category, the number of FP images for BYOL increases from 10 in clean model to 2, 635 in the backdoored model. In comparison, we see the FP for backdoored Jigsaw and RotNet models are relatively unchanged by the attack. On average, MoCo v2, BYOL, and MSF have 461, 1442, and 830 FP respectively, but Jigsaw and RotNet have only 58 and 63 FP respectively. This indicates that the targeted attack is less effective for Jigsaw and RotNet than the other three exemplar-based SSL methods.

Linear classifier training on 10% of ImageNet-100: Table 14 shows the results of our targeted backdoor attack with 0.5% poison injection rate and linear classifiers trained on 10% of ImageNet-100. This table shows the average for 10 target class experiments. We report the detailed results in the supplementary. We observe the same pattern here as we did in Table 15. Backdoored MoCov2, BYOL, and MSF models have larger target class FP compared to Backdoored Jigsaw and RotNet models.

Backdoor attack on MAE: We attack Masked Autoencoders (MAE) [He+21a], which is a very recent SSL method. We observed that MAE does not perform well with linear evaluation on 1% labeled data. So, as used in the MAE paper, we finetune the whole model on 1% ImageNet-100. Interestingly, our attack is less effective on MAE. The FP increases from 13 in the clean model to 82 in the backdoored model. This might be due to MAE not pulling two different augmentations together or due to finetuning the whole model. Investigating this is an interesting future work.

Table 15: Targeted attack on ImageNet-100: We use 0.5% poison injection rate. SSL methods are trained on poisoned ImageNet-100 data and a linear classifier is trained on 10% ImageNet-100 labeled data - averaged over 10 target class trigger pairs.

Method	Clean model				Backdoor red model			
	Clean data		Patched data		Clean data		Patched data	
	Acc (%)	FP	Acc (%)	FP	Acc (%)	FP	Acc (%)	FP
MoCo v2	62.2	21.3	57.5	18.9	61.6	21.0	51.0	605.9
BYOL	72.9	16.4	66.4	16.9	72.7	16.5	40.2	1872.2
MSF	67.5	18.2	63.0	14.2	68.4	16.5	40.6	1491.4
Jigsaw	36.0	39.5	31.2	48.0	35.1	37.2	30.5	45.5
RotNet	40.1	28.0	34.6	35.4	40.6	31.9	26.6	31.5

Table 16: Untargeted attack on ImageNet-100: We poison 5% random images of ImageNet-100 training set. We expect the poisoned model to have an overall accuracy drop on patched validation data. The targeted attack contributes to a 5-point decrease in accuracy. The linear classifier is trained on 1% of ImageNet-100.

Trigger ID	Clean model		Backdoored model	
	Clean data	Patched data	Clean data	Patched data
	Acc (%)	Acc (%)	Acc (%)	Acc (%)
10	50.34	46.46	49.02	30.60
12	50.34	46.42	50.54	46.54
14	50.34	46.64	49.44	42.56
16	50.34	47.00	49.34	45.34
18	50.34	46.64	48.78	43.44
Average	50.34	46.63	49.42	41.70

3.2.3 Consistent Explanations by Contrastive Learning

Post-hoc explanation methods, e.g., Grad-CAM, enable humans to inspect the spatial regions responsible for a particular network decision. However, it is shown that such explanations are not always consistent with human priors, such as consistency across image transformations. Given an interpretation algorithm, e.g., Grad-CAM, we introduce a novel training method to train the model to produce more consistent explanations. Since obtaining the ground truth for a desired model interpretation is not a well-defined task, we adopt ideas from contrastive self-supervised learning, and apply them to the interpretations of the model rather than its embeddings. We show that our method, Contrastive Grad-CAM Consistency (CGC), results in Grad-CAM interpretation heatmaps that are more consistent with human annotations while still achieving comparable classification accuracy. Moreover, our method acts as a regularizer and improves the accuracy on limited-data, fine-grained classification settings. In addition, because our method does not rely on annotations, it allows for the incorporation of unlabeled data into training, which enables better generalization of the model. The code is available here: <https://github.com/UCDvision/CGC>

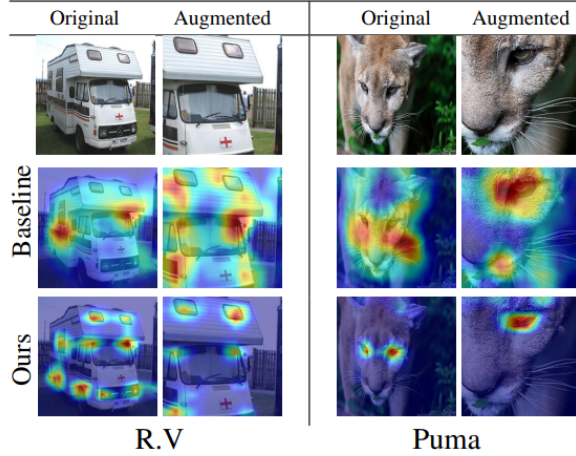


Figure 16: Our method significantly improves the consistency of Grad-CAM explanation heatmaps under data augmentation. For both the RV and the Puma, our method highlights the same portions of the image in both the original and augmented versions.

It is shown that most interpretation methods are not consistent with the spatial transformation of the images. For instance, shifting an image does not shift the interpretation heatmap in the same way [Guo+19; Kin+17]. In addition, in fine-grained visual categorization, it is important to learn the subtle yet discriminative features across classes (e.g., wing color, beak and eyes for a bird) [Wel+10] and hence the network interpretation should focus on the most salient features that discriminate the correct class from other classes. Assuming Grad-CAM is a truthful interpretation method, we are interested in improving the training process of the deep network so that its Grad-CAM interpretation is more consistent with respect to spatial transformations, thus making the model more interpretable. We use Grad-CAM as the key interpretation method for the rest of the paper since it passes the sanity check introduced in [Ade+18] and is end-to-end differentiable.

Inspired by self-supervised learning, we argue that a spatial affine transformation on an image should correspond to such transformation in the interpretation. For instance, given image

of a “dog”, the heatmap for “dog” category should highlight the dog and it should shift or zoom if we shift or zoom into the image. Figure 16 shows an example where the change in Grad-CAM heatmap for the baseline network computed for the “RV” and “Puma” categories is not consistent with the spatial affine transformation applied to the images. We adopt ideas from recently developed contrastive self-supervised learning literature [HCL06; NPF17a; He+19] and design a loss function that encourages the Grad-CAM of an image to be close to the Grad-CAM of an augmented version of the same image while being far from the Grad-CAM of other random images.

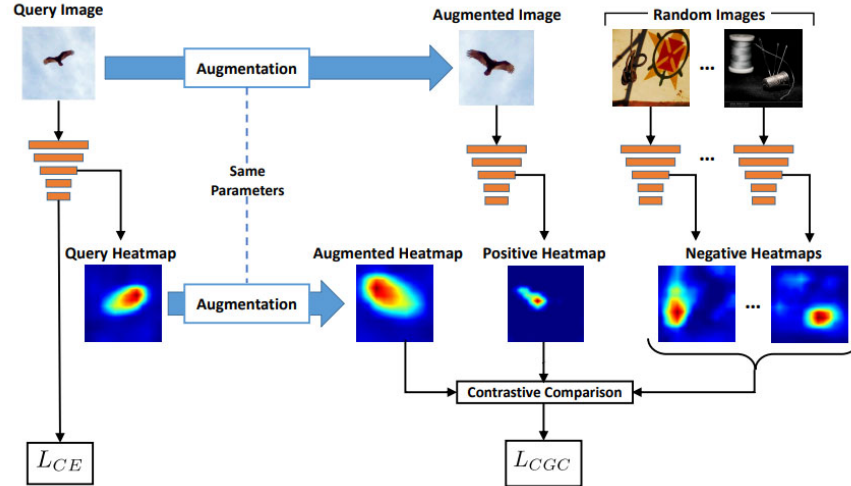


Figure 17: The block diagram of our method. Our method consists of both cross-entropy loss (L_{CE}) and contrastive Grad-CAM consistency loss (L_{CGC}). We load a batch of random images, and consider one to be the query image. We feed the query image to the network and calculate L_{CE} . We calculate Grad-CAM for this image on the top predicted category and then augment the heatmap. We then augment all the images in the batch, feed them to our model, and calculate the Grad-CAM heatmap for the top predicted category. The top category is chosen using the original image and not the augmented one. (Note that all random images are also augmented with independently sampled parameters. We do not show this to reduce the clutter.) The heatmap from the augmented query image is considered to be the positive example. The heatmaps from the other random images in the batch are the negative examples. The augmented query heatmap, the positive heatmap, and the negative heatmaps are all fed into our contrastive comparison function to produce our L_{CGC} term.

We evaluate our CGC method for both classification accuracy and quality of explanations measured by the “Content Heatmap” (CH) metric introduced in [PP21]. A similar metric was also introduced in [SPP19b] to measure the energy of the explanation heatmap inside an adversarial patch. CH measures the cumulative heatmap contained within an annotated object mask and is a proxy to measure the consistency of an explanation heatmap with respect to human annotations. Moreover, since our method guides the model to focus mainly on the most discriminative features of the images, it improves the accuracy in fine-grained classification settings while improving the consistency of interpretations. The effect is even more prominent in learning with fewer labels on fine-grained tasks. We believe this is due to the regularization effect that our method adds to the training process. In addition, since our loss function does not need labeled data, it can benefit from unlabeled data during training. Our

experiments show that our method improves the consistency of interpretation with respect to human annotations as well as the classification accuracy in limited labels and fine-grained settings.

Figure 17 shows a block diagram of our method. Our training process consists of both categorical cross-entropy loss (LCE) and contrastive Grad-CAM consistency loss ($LCGC$). In this section, we first present a brief overview of the Grad-CAM interpretation algorithm and then describe the contrastive Grad-CAM consistency loss term.

Background on Grad-CAM [Sel+16]: Consider an input image x and a deep neural network f . Let y be the vector of output logits when we feed x to the model f where y_t corresponds to the output for category t . The Grad-CAM of model f for image x and a given category t is a heatmap that highlights the regions of image x responsible for the model’s classification of the image as category t . We calculate this heatmap by choosing an intermediate convolutional layer and then linearizing the rest of the network to be interpretable. More specifically, we calculate the derivative of the predicted output with respect to each channel of the convolutional layer averaged over all spatial locations. This results in a scalar for each channel that captures the importance of that channel in making the current prediction. Then, we calculate a weighted average of all activations of the convolutional layer with the above importance weights for each channel to get a 2D matrix over spatial locations. Finally, we keep only positive numbers and resize it to the size of input image to get the interpretation heatmap.

We are interested in training the image classification model so that its Grad-CAM heatmaps are consistent with spatial transformations. For instance, when we shift the image, the interpretation heatmap should also shift in the same way. Also, the heatmap should be specific to the image, as in not always focused on a blob in the middle of the image or be spread around the whole image. Inspired by contrastive learning methods in self-supervised learning, we design a contrastive loss function for the interpretation heatmaps that acts as a regularizer when added to the standard cross entropy loss for supervised learning. We want the transformed interpretation of a query image to be close to the interpretation of the transformed query image while being far from interpretations of other random images.

3.2.3.1 Results and Discussion

In this section, we perform a variety of experiments using our CGC method. For each of the experiments, ‘baseline’ refers to a model trained from scratch using the standard cross-entropy loss unless noted otherwise. We will report the classification accuracy along with the following metrics used for evaluating the explanation heatmaps:

Content Heatmap (CH): Introduced in [PP21], this metric is a measure of the summation of ℓ_1 -normalized heatmap contained within the annotated bounding box of the object. If the model interpretation is consistent with human annotations of the object location, we can assume that the percentage of the heatmap that lies inside the object annotation mask should be close to 100%. Hence, we expect this metric to be high.

CGC loss: We also evaluate the explanation heatmaps using the same LCGC loss that we use for training our models. Although this loss is already used in our optimization, we believe it is important to show that the loss is small on the unseen test data as well, i.e., the method generalizes from the training to the test set. We use ImageNet [Rus+15a] validation set to

report this metric. We use a batch size of 32 to compute this loss term. For every query image in the batch, we pair it with an augmentation of the corresponding query image as the positive pair and consider each of the remaining 31 images in the batch to be the negative pairs.

Insertion AUC score (IAUC): This metric [PDS18] successively inserts pixels from the highest to lowest attribution scores and makes predictions. The area under the curve defined by the prediction scores is then defined as the IAUC score. We expect the IAUC score to be higher for a better interpretation.

Implementation Details: We use PyTorch [Pas+19] to train and evaluate our models for all experiments. We use SGD (weight decay= $1e-4$, learning rate=0.1, momentum=0.9, and batch size=256) to optimize both ResNet18 and ResNet50 [He+15] models. We train ImageNet [Rus+15a] models for 90 epochs and decay the learning rate by 0.1 every 30 epochs. For transformation T_i in our loss term, we use standard data augmentations (scaling, flipping, and translation). The baseline models also use these same augmentations. We use ResNet50 architecture for all our experiments unless otherwise noted. Training our ResNet50 on 2 Titan-RTX GPUs takes approximately 100 hours, whereas training the baseline takes approximately 70 hours on the same setting. We use $\tau = 0.5$ for all experiments using our method. For the ImageNet dataset, we use $\lambda = 1.0$ for our method using ResNet18 and $\lambda = 0.5$ for ResNet50.

We first report our results on ImageNet and UnRel [Pey+17] datasets. We then show results on fine-grained classification tasks including limited data settings. Finally, we discuss results using unlabeled data with our loss term.

Contrastive Grad-CAM Consistency (CGC)

We train a network from scratch using the CGC method (cross-entropy loss and contrastive loss) and compare it to a baseline network trained from scratch using cross-entropy loss only. We report the Top-1 classification accuracy as well as heatmap evaluation metrics. We show results for both the ImageNet and UnRel [Pey+17] datasets in order to show that our approach generalizes across multiple datasets. For both datasets, on the ResNet50 architecture, we show that our model has a slight (less than 2% points) decrease in classification accuracy but shows a significant (greater than 15% points) improvement on the explanation metrics.

ImageNet: Quantitative results on the ImageNet validation set are reported in Table 17. For the ResNet50 model, we have a marginal drop in classification accuracy (1.5 points), whereas CH increases by 17 points. Since ImageNet is a large dataset, we do not expect our regularizer to improve the classification accuracy on ImageNet. Moreover, our main focus is on improving the consistency of the explanations and we are willing to accept a marginal drop in classification accuracy at the cost of improved explanation consistency.

Table 18 reports the evaluation results using the IAUC metric [PDS18] on the ImageNet validation set and shows that our method quantitatively improves the Grad-CAM explanation heatmaps of the underlying model.

Table 17: Classification accuracy along with the Content Heatmap (CH) and CGC Loss explanation metrics on the ImageNet validation set. Note that lower is better for CGC Loss.

Arch	Method	Top-1 Acc (%)	CH (%)	CGC Loss
ResNet18	Baseline	69.76	54.47	3.19
	GCC [PP21]	67.74	57.73	3.14
	Ours (CGC)	66.37	65.83	2.59
ResNet50	Baseline	76.13	54.77	3.15
	GCC [PP21]	74.40	59.42	3.09
	Ours (CGC)	74.60	71.75	2.64

Table 18: The Grad-CAM explanation maps generated by our ResNet50 model outperform the baseline ResNet50 model on the IAUC metric [PDS18] using the ImageNet validation set.

Method	Insertion AUC
Baseline	0.4860
Ours (CGC)	0.5216

UnRel: The UnRel dataset [Pey+17] consists of images capturing unusual relations between objects. These images were collected from the web using triplet queries such as ‘person ride giraffe’. This dataset thus captures objects occurring in unusual spatial configurations and contexts. Both the baseline and CGC models are trained on ImageNet and evaluated on the 28 object categories of the UnRel dataset that overlap with ImageNet. We report the corresponding quantitative results in Table 19. We observe that although our CGC model was trained on ImageNet, which is an object-centric dataset, the improvement in our explanation heatmap generalizes to objects occurring in unusual backgrounds and spatial configurations.

Table 19: Classification accuracy and Content Heatmap (CH) evaluation on the 28 UnRel categories [Pey+17] that overlap with ImageNet.

Model	Top-1 Acc (%)	CH (%)
Baseline	40.66	51.66
Ours (CGC)	38.25	74.20

Using Unlabeled Data for CGC Loss

Models trained with categorical cross-entropy loss only have no way to incorporate information from unlabeled data. Since our method does not rely on labels for computing the explanation heatmaps, we extend our method to use mostly unlabeled data with just a small fraction of labeled data for training. We use 1% of ImageNet training images as labeled data and leverage the rest of ImageNet training images as the unlabeled data for the CGC loss term. We initialize both the baseline and our method from a ResNet50 model trained using SwAV [Car+20].

Table 20 shows that our method outperforms the baseline by 1% point on both the Top-1 and Top-5 accuracy metrics. Since we are using only 1% labeled data, the resulting Grad-CAM heatmap will be less accurate and hence the 1% point improvement is not insignificant. Even though our model is only able to leverage the additional unlabeled data for the CGC

loss term, and not the categorical accuracy, we can see that the CGC loss acts as a regularizer to improve the generalization of the model.



Figure 18: Grad-CAM visualization results for images from ImageNet validation set using ResNet50. We observe that the model trained using our method focuses on the most discriminative regions of the object instead of the background pixels. In the 6th column above, we compute the Grad-CAM explanation for the category “volleyball” and we see that our model no longer focuses on the players and correctly highlights the volleyball as the cause of the target category. The last column contains a failure example for the category “Dogsled”. The baseline model correctly highlights the sled whereas our model incorrectly highlights the dog.

Table 20: On the 1% ImageNet limited-label setting, our CGC method leverages unlabeled data for the LCGC loss term and is able to improve the classification accuracy and explanations when evaluated on ImageNet validation data. Both models are initialized from a ResNet50 model trained in an unsupervised manner using SwaV [Car+20].

Model	Top-1 (%)	Top-5 (%)	CH (%)
Baseline	54.00	78.69	46.08
Ours (CGC)	55.18	79.12	46.76

3.3 Machine translation

3.3.1 Embedding Space Mapping for Bilingual Lexicon Induction

Embeddings, or vector representations of linguistic units - such as words, subwords, or chunks of text - are essential building blocks of modern natural language processing systems. Embeddings are typically used as one of the very first steps in building an NLP system: written tokens are encoded as vectors, and then passed through the rest of the system to perform some downstream task.

Static embeddings are vector representations of linguistic units that, once created, stay fixed. For instance, regardless of the context in which a word appears, its static embedding is always the same. A popular method for training static word embeddings is skip-gram with negative sampling, as introduced in the well-known word2vec package (<https://github.com/tmikolov/word2vec>).

We use static word embeddings for **bilingual lexicon induction (BLI)**, e.g., the establishment of mappings of words in one language to words in another language. The typical baseline approach for BLI from word embedding spaces assumes that spaces can be mapped via linear transformation. Here, separately-trained monolingual embeddings are mapped to a shared space by applying a linear transformation to one or both spaces, after which a bilingual lexicon can be extracted via nearest-neighbor search. Alternatively, a graph-based view considers words as nodes in undirected weighted graphs, where edges are the distance between words. Methods taking this view do not assume a linear mapping of the spaces exists, allowing for more flexible matching.

Our research focus was to develop novel bilingual lexicon induction methods, which are then used in an **unsupervised machine translation** approach that yields models that translate full sentences. Training begins with two monolingual corpora which are not necessarily related in any way (i.e. they are not assumed to be parallel nor comparable text). First, word embeddings are trained independently for each corpus, resulting in a source and a target embedding space. Next, source and target word embeddings are aligned into a common cross-lingual embedding space. For source words and phrases, the most similar embeddings in the target language are selected into a translation table of a statistical machine translation model. Combining the preliminary phrase table with a distortion penalty and language model produces the initial unsupervised phrase-based statistical machine translation (SMT) system. The SMT model then undergoes three rounds of iterative backtranslation. The back-translation data from the final iteration is used to train a neural machine translation (NMT) model. See the figure below for an illustration.

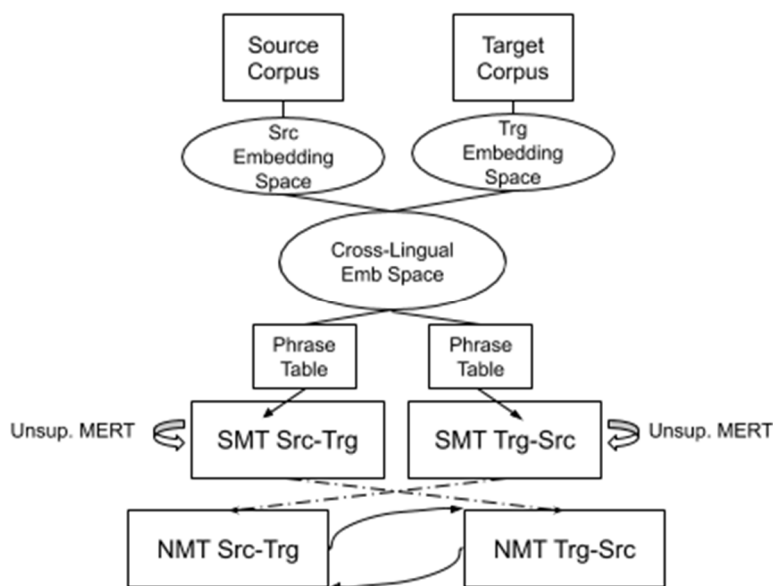


Figure 19: Unsupervised machine translation based on a common cross-lingual embedding space

When Does Unsupervised Machine Translation Work?

We started out this line of work by examining the conditions under which bilingual lexical induction and unsupervised machine translation methods succeed and fail. Results from the mapping-based unsupervised machine translation (MT) literature appear promising, but results are reported primarily for the high-resource languages when the need for an unsupervised method is questionable.

For unsupervised MT to be a viable path for low-resource machine translation, the field must determine (1) if it works outside highly-controlled environments, and (2) how to effectively evaluate newly-proposed training paradigms to determine which are promising for real-world low-resource scenarios. Unsupervised MT methods must work (1) on **different scripts** and between **dissimilar languages**, (2) with **imperfect domain alignment** between source and target corpora, (3) with a **domain mismatch** between training data and the test set, and (4) on low-quality data of **real low-resource languages**. These factors reflect the real-life challenges of low-resource translation. We thus conduct an extensive empirical evaluation using dissimilar language pairs, dissimilar domains, and diverse datasets.

Our main contribution here is an extensive analysis of mapping-based unsupervised MT with regard to factors (1)-(3) above. We find that (a) translation performance rapidly deteriorates when source and target corpora are from different domains, (b) stochasticity during word embedding training can dramatically affect downstream bilingual lexicon induction (BLI) and translation performance, and (c) like in the bilingual lexicon induction literature, unsupervised MT performance declines when source and target languages are dissimilar. While (4) was not the focus, we do observe very low performance on an authentic low-resource language pair.

We carried out experiments by varying:

- Language pairs from closely related French-English to more distant Russian-English, and truly low-resource Sinhala-English and Nepali-English
- Corpora varying from the two (unaligned) sides of a parallel corpus, disjoint parts of a parallel corpus, and different domains (news and web crawls).

As there are no standard evaluation protocols to ensure that unsupervised MT systems are robust to the types of data anomalies ubiquitous in low-resource translation settings, we advocate for extensive empirical evaluation of unsupervised MT systems to highlight failure points and encourage continued research on the most promising paradigms. Towards this goal, we release our preprocessed data set at <http://statmt.org/when-does-unsup-work> to stress-test systems under multiple data conditions.

3.3.1.1 Results and Discussion

We perform an extensive empirical evaluation of unsupervised MT. Our evaluation protocol stress-tests an unsupervised MT system under varying conditions to reveal its points of strength and failure. Systems should be judged on how well they perform: (1) on dissimilar languages, (2) on increasingly divergent domains between source and target corpora, (3) on diverse datasets, and (4) on authentic low-resource language pairs where data quality is typically low. Namely, we:

- Choose 2 language pairs, at least one of which where the source and target languages utilize different scripts.
- Choose 3 datasets of different domains, at least one of which is parallel bitext.
- Perform at least one experiment for each language pair under each of the following data conditions:
 - Originally parallel
 - Not originally parallel

- Different domains for source and target.
- Choose 2 true low-resource language pairs.
- Judge the system based on performance in all tested scenarios.

The data conditions above are designed to measure how well a system performs in regard to the research questions of Section 3.3.1. Namely, success on a variety of languages with different scripts and linguistic structures indicates robustness to dissimilar languages (Step 1); success on multiple datasets of different domains (Step 2) indicates that the system is not specifically designed for one domain at the expense of others, and performs well even when training and test data do not match perfectly; (Step 3) evaluates performance on increasingly divergent domains between source and target data; and (Step 4) is the true test - whether the system succeeds on authentic low-resource language pairs.

We focus on Russian-English (Ru-En) and French-English (Fr-En) tasks and include as reference Sinhala-English (Si-En) and Nepali-English (Ne-En) as well. We evaluate the same system under various ablated data setups:

- “Supervised” is the standard MT training setup which uses parallel bitext.
- In the “Parallel” condition, an \textit{unsupervised} MT system is trained on a corpus that was originally parallel (i.e. UN corpus), now being treated as two separate monolingual corpora.
- In contrast, the “Disjoint” setting splits data from a parallel corpus into two disjoint halves, using the first half of the source-side corpus and the second half of the target-side corpus.
- In the “Different Domain,” (Diff. Dom.) setting, source and target monolingual corpora come from different domains. This is a realistic setting in low-resource scenarios and is expected to be much more difficult than the “Disjoint” setting.
- “News crawl” (News) and “Common Crawl” (CC) settings determine whether the system can flexibly handle diverse datasets.

Specifics of the datasets used are shown in the table below.

Table 21: UN=United Nations (parallel data extracted from the United Nations website), CC=Common Crawl (parallel data extracted from a general web crawl), News=News Crawl (monolingual news content).

Condition		Corpus	Src Tokens (M)	Trg Tokens (M)
Repro	Fr-En	News	694	1940
	En-Fr	News	1940	694
Supervised	Fr-En	UN: A	346	301
	Ru-En	UN: A	284	284
Parallel	Fr-En	UN: A	302	270
	Ru-En	UN: A	232	241
Disjoint	Fr-En	UN: A / B	302	255
	Ru-En	UN: A / B	232	236
Different Domain	Fr-En	UN: A / CC	302	226
	Ru-En	UN: A / CC	232	226
News	Fr-En	News	116	105
	Ru-En	News	120	105
Common Crawl	Fr-En	Common Crawl	110	79
	Ru-En	Common Crawl	115	79

Our main findings are in the table below. We find that unsupervised machine translation holds up pretty well under adverse conditions, except for different domains.

Table 22: Unsupervised MT performance on a single run using the United Nations (UN) dataset. “Diff. Dom.” uses UN data as source and Common Crawl (*) as target. “Sup.” is supervised with UN parallel data. We report BLEU scores (higher is better). A / A refers to UN training dataset A used on the source and target sides, for example.

	Supervised	Parallel	Disjoint	Diff. Dom.
Corpus	A / A	A / A	A / B	A / CC*
Ru-En	26.9	23.7 (-3.2)	21.2 (-5.7)	0.7 (-26.2)
Fr-En	29.9	27.6 (-2.3)	27.0 (-2.9)	3.9 (-26.0)

For truly low-resource languages (Sinhala and Nepali), however, the methods fail (see table below). This motivated our research into novel methods carried out under this program. For more details on this study, please refer to Marchisio et al. (2020).

Table 23: Unsupervised MT performance for truly low-resource language pairs (Sinhala-English and Nepali-English).

	UN	News	CC
Ru-En	21.2	16.1	13.8
Fr-En	27.0	28.2	22.4
Si-En	n/a	n/a	0.2
Ne-En	n/a	n/a	0.4

3.3.2 Graph Matching Methods for Bilingual Lexicon Induction

There is a large literature inducing bilingual lexicons from cross-lingual spaces. Much recent work in bilingual lexicon induction (BLI) views word embeddings as vectors in Euclidean space. As such, BLI is typically solved by finding a linear transformation that maps embeddings to a common space. This “Euclidean” view, exemplified by methods solving the Procrustes problem, works with embedding spaces and assumes the existence of a linear transform that maps the spaces. “Mapping” methods based on solving the orthogonal Procrustes problem and its generalizations are popular, where languages are mapped to a common space from which a lexicon is extracted. This has been successful when word embedding spaces are roughly isomorphic, but fails as embedding spaces diverge.

Rather than words in Euclidean space, we might, for instance, create a similarity graph with words as nodes and the cosine distance between word vectors as edges. In this way, we can work with weighted graphs derived from embeddings. This framing allows us to examine a node's graph neighborhood without assuming a linear transform, and exploits new techniques from the graph-matching optimization literature. The graph-based view works with graphs for each language and directly performs matching on edge pairs based on neighborhood information. This view is exemplified by graph-matching methods that solve the quadratic assignment problem from the combinatorial optimization literature. The figure below illustrates the differences between the framings; while they both exploit the idea that words with similar neighbors (in Euclidean or graph space) should be translations of one another, they implement the idea in very different ways.

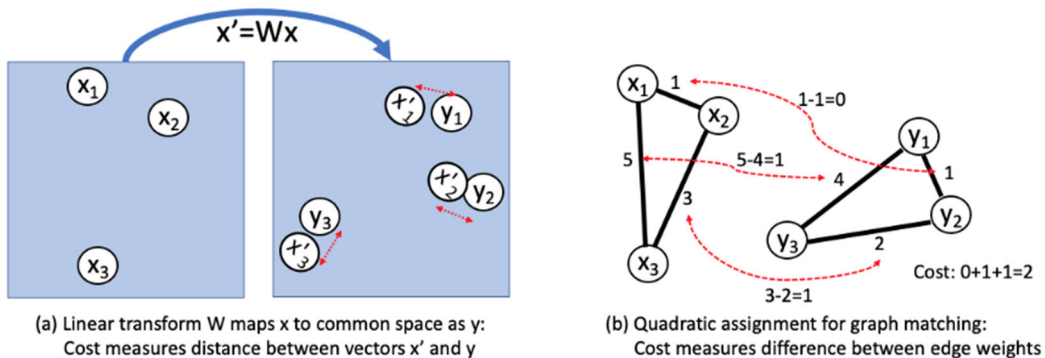


Figure 20: Comparing Euclidean vs. Graph views. (a) Euclidean view assumes common embedding space and computes costs based on pairs of vectors. (b) Graph-based view assumes graph structure and computes cost based on pairs of edges. Both exploit within-language neighborhood info but in different ways.

We use **Seeded Graph Matching** (SGM) as representative of this class of approach. Here, words are nodes in monolingual graphs G_x , G_y , and cells are edge weights representing the distance between words. In NLP, it is common to use cosine similarity as the measure of distance when weighting the edges. The objective function is the equation below, where Π is the set of permutation matrices.

$$\max_{P \in \Pi} \text{trace}(G_x^T P G_y P^T)$$

Vogelstein et al. (2015)'s Fast Approximate Quadratic Assignment Problem algorithm (FAQ) uses gradient ascent to approximate a solution. FAQ solves the objective over the space of doubly-stochastic matrices with the Frank-Wolfe method and then projects back to a permutation matrix, seen in the algorithm below.

Algorithm 1 FAQ Algorithm for Graph Matching

Let: $G_x, G_y \in \mathbb{R}^{n \times n}$, $P^{(0)} \in \mathcal{D}$ (doubly-stochastic)
while stopping criterion not met **do**
 1. Calculate $\nabla f(P^{(i)}) = G_x P^{(i)} G_y^T + G_x^T P^{(i)} G_y$
 2. $Q^{(i)} =$ permutation matrix approx. to $\nabla f(P^{(i)})$ via Hungarian Algorithm
 3. Calculate step size:
 $\arg \max_{\alpha \in [0,1]} f(\alpha P^{(i)} + (1 - \alpha) Q^{(i)})$
 4. Update $P^{(i+1)} := \alpha P^{(i)} + (1 - \alpha) Q^{(i)}$
end while
return permutation matrix approximation to $P^{(n)}$ via Hungarian Algorithm.

It is popular to use the Procrustes solution iteratively. One applies the transformation calculated via Procrustes, extracts a dictionary of translation candidates, then uses those as seeds for the next round of Procrustes. We develop an analogous iterative algorithm for SGM. The figure below illustrates the two related approaches.

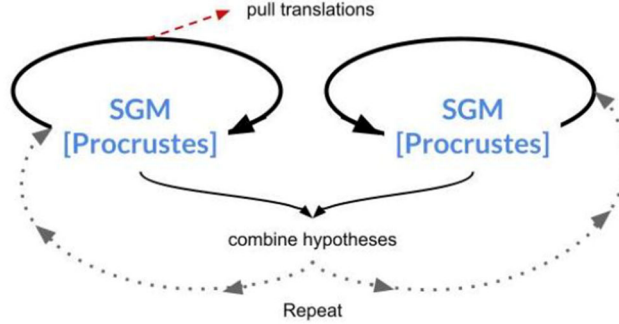


Figure 21: IterSGM and IterProc. Run SGM (or Procrustes) in forward and reverse directions. Combine hypotheses and pass them as seeds to SGM (Procrustes). Pull final translations on the last iteration from the forward run.

We run SGM or Procrustes and extract potential translation pairs in source-target and target-source language directions, resulting in two sets of translation hypotheses (one hypothesis per source word from each translation direction). For Procrustes, this extraction is done with CSLS. We intersect the hypotheses from the two directions and feed the resulting setback to Procrustes (or SGM) as seeds.

We run SGM or Procrustes and extract potential translation pairs in source-target and target-source language directions, resulting in two sets of translation hypotheses (one hypothesis per source word from each translation direction). For Procrustes, this extraction is done with

CSLS. We intersect the hypotheses from the two directions, and feed the resulting set back to Procrustes (or SGM) as seeds. We abbreviate the iterative procedures as IterProc and IterSGM.

The general procedure is:

1. Run Procrustes (or SGM), forward direction.
2. Run Procrustes (or SGM), reverse direction.
3. Intersect the hypotheses from both directions.
4. Feed the hypotheses into Step 1. Repeat.

Before Step 1 for IterSGM, we form the graphs. For IterProc, we combine the hypotheses in Step 3 with the gold seeds, which is unnecessary for SGM because seeds are always returned in the hypotheses.

3.3.2.1 Results and Discussion

The first graph-matching algorithm we introduced to bilingual dictionary induction is Seeded Graph Matching (SGM; see Section 3.3.2). We evaluate the method first on German-English and Russian-English. See the table below for precision@1 results, e.g., the accuracy that found bilingual mappings for a word is actually the correct word.

Table 24: Precision@1 results for the graph matching method SGM vs. the baseline Procrustes method on German-English and Russian-English. Except for a moderate amount of seed bilingual entries (1000 and 2000-word translations) for German-English, our SGM method outperforms the standard approach.

Seeds	En-De		Ru-En	
	Procrustes	SGM	Procrustes	SGM
100	3.6	45.8	4.1	50.1
200	16.1	47.3	16.6	52.2
500	44.9	51.9	45.3	56.0
1000	57.2	54.9	56.6	58.1
2000	63.1	61.5	62.7	67.1
4000	70.8	74.2	67.9	89.3

In a refinement of the method, we iterate over the method, by incrementally adding found mappings to the seed lexicon. How one selects seeds in Step 4 for subsequent rounds is important. We try three variations:

- **Add-All.** Intersect hypotheses from forward and reverse directions. All become seeds for the next round, for N total rounds. Advantage: all correct pairs are passed to the next iteration. Disadvantage: all incorrect hypotheses are, too.
- **Stochastic-Add.** Add up to H new hypotheses in each iteration. For iteration two, H random hypotheses from the intersection are chosen and added to the gold seeds for the forward direction. A separate random selection is taken for the reverse direction.

In the next round, 2H random hypotheses are chosen. This continues until all hypotheses are used. This setting was designed to encourage robustness and improve accuracy by minimizing the number of erroneous seeds passed to subsequent rounds, to allow for recovery from mistakes. As distinct subsets are passed to forward and reverse directions, we encourage solutions of the runs to also be different, increasing output diversity. When intersecting the hypotheses, we aim to select pairs that are most likely to be correct - having two different solutions agree increases confidence that the induced pairs are correct, and selecting only a small subset allows recovery from mistakes. This is particularly important for SGM, where incorrect seeds are repeated in the output. In passing a subset to the next round, some incorrect pairs are dropped, and the model gets another chance to induce translations with a (presumably) stronger model. The stochasticity builds in robustness.

- **Active-Learning.** Seeds may also be added in an active learning fashion ("human-in-the-loop"). To simulate a human judging hypothesis quality, we use the union of hypotheses from forward and reverse directions and pass only correct hypotheses as seeds for the next iteration.

Iteration significantly improves results, as shown in the table below.

Table 25: Precision@1 of IterProc vs. IterSGM. Add-All runs for 10 iterations, seeding subsequent iterations with the intersection of hypotheses from forward and reverse directions. For Stochastic-Add, seeds are fed in up to 100 at a time until all are used. In parentheses is the improvement over Add-All.

	Seeds	Add-All		Stochastic-Add		Active-Learning	
		IterProc	IterSGM	IterProc	IterSGM	IterProc	IterSGM
En-De	100	61.3	47.2	62.1 (+0.8)	50.2 (+3.0)	66.1 (+4.8)	56.6 (+9.4)
	200	61.5	48.2	62.0 (+0.5)	50.8 (+2.6)	66.3 (+4.8)	56.7 (+8.5)
	500	62.6	52.1	62.8 (+0.2)	52.9 (+0.8)	66.6 (+4.0)	58.3 (+6.2)
	1000	63.0	54.7	63.5 (+0.5)	54.8 (+0.1)	67.3 (+4.3)	59.5 (+4.8)
	2000	65.2	61.4	65.2 (+0.0)	61.7 (+0.3)	69.1 (+3.9)	65.6 (+4.2)
	4000	71.3	74.2	71.7 (+0.4)	74.4 (+0.2)	74.6 (+3.3)	75.4 (+1.2)
Ru-En	100	62.4	51.6	62.7 (+0.3)	56.3 (+4.7)	71.0 (+8.6)	62.5 (+10.9)
	200	62.4	53.7	63.1 (+0.7)	56.4 (+2.7)	71.1 (+8.7)	61.9 (+8.2)
	500	63.7	56.1	63.7 (+0.0)	58.0 (+1.9)	71.3 (+7.6)	63.1 (+7.0)
	1000	64.0	58.1	64.0 (+0.0)	60.3 (+2.2)	71.2 (+7.2)	66.4 (+8.3)
	2000	66.1	67.1	65.7 (-0.4)	68.2 (+1.1)	72.3 (+6.2)	71.0 (+3.9)
	4000	69.0	89.3	69.0 (+0.0)	89.3 (+0.0)	72.6 (+3.6)	89.3 (+0.0)

Iteration is not restricted to one method, we may also iterate between SGM and Procrustes. This leads to further improvements, as shown in the table below.

Table 26: $P@1$ for combined iterative method. One may begin from either IterProc (“Start: Procrustes”) or SGM (“Start: SGM”) and may pull final hypotheses from either Procrustes (“-PullProc”) or SGM (“-PullSGM”). “Prev. Best” is the best result from previous experiments (excluding active learning). Bold is best overall.

	Seeds	Prev. Best	Combination Methods			
			<u>-PullProc</u>		<u>-PullSGM</u>	
			Start: IterProc	Start: SGM	Start: IterProc	Start: SGM
En-De	100	62.1	62.2	62.1	59.7	59.5
	200	62.0	62.8	62.6	60.4	60.4
	500	62.8	63.5	63.8	62.1	62.0
	1000	63.5	63.9	64.2	63.0	63.7
	2000	65.2	66.7	66.7	69.7	69.0
	4000	74.4	73.2	73.2	79.7	79.2
Ru-En	100	62.7	63.9	64.0	61.7	62.0
	200	63.1	64.5	64.3	62.6	63.1
	500	63.7	65.3	65.3	64.0	64.3
	1000	64.0	66.8	66.4	66.8	66.4
	2000	68.2	69.4	69.5	72.9	73.1
	4000	89.3	77.4	77.4	89.3	89.3

3.3.3 Bilingual Lexicon Induction for Low-Resource Languages using Graph Matching via Optimal Transport (GOAT)

We improve bilingual lexicon induction performance across 40 language pairs with new a graph-matching method based on optimal transport, called GOAT. GOAT is a new graph-matching method that uses advances in optimal transport to solve the graph-matching problem. Similar to SGM, GOAT amends FAQ and can use seeds. GOAT has been successful for the inexact graph-matching problem on non-isomorphic graphs: whereas FAQ rapidly fails on non-isomorphic graphs, GOAT maintains strong performance.

Optimal Transport is an optimization problem concerned with the most efficient way to transfer probability mass from distribution μ to distribution ν . Discrete optimal transport, which we will use for BLI because we compute over matrices, minimize the inner product of a transportation “plan” matrix P with a cost matrix C .

Sinkhorn: Lightspeed Optimal Transport. Cuturi (2013) introduces Sinkhorn distance, an approximation of optimal transport distance (Earth Mover’s Distance) that can be solved quickly and accurately by adding an entropy penalty h . This makes the objective easier and more efficient to compute and encourages “intermediary” solutions.

$$P^\lambda = \operatorname{argmin}_{P \in U(r,c)} \langle P, C \rangle - 1/\lambda h(P)$$

As $\lambda \rightarrow \infty$, P^λ approaches the ideal transportation matrix P^* .

Sinkhorn in GOAT. GOAT uses Cuturi (2013) ’s algorithm to solve the above equation over $U(1,1)$, the set of doubly-stochastic matrices. They call this the “doubly stochastic optimal transport problem”, and the algorithm that solves it is “Lightspeed Optimal Transport” (LOT). As the Hungarian and Sinkhorn Algorithms fundamentally solve the same optimization objective, one can be used in place of the other. Saad-Eldin et al. (2021) incorporate

Sinkhorn distance into their Lightspeed Optimal Transport Algorithm based on to compute a doubly-stochastic solution for Step 2 of the FAQ algorithm (see above) which they allege was the computational bottleneck when solved with the Hungarian Algorithm.

Although Sinkhorn distance was created for efficiency, Saad-Eldin et al. (2021) find that using the matrix P^λ that minimizes Sinkhorn distance also improves matching performance on large and non-isometric graphs. The algorithm below is GOAT.

Algorithm 2 GOAT

Let: $G_x, G_y \in \mathbb{R}^{n \times n}$, $P^{(0)} \in \mathcal{D}$ (doubly-stochastic)
while stopping criterion not met **do**
 1. Calculate $\nabla f(P^{(i)}) = G_x P^{(i)} G_y^T + G_x^T P^{(i)} G_y$
 2. $Q^{(i)} =$ doubly-stochastic approximation to $\nabla f(P^{(i)})$ via LOT.
 3. Calculate step size:
 $\arg \max_{\alpha \in [0,1]} f(\alpha P^{(i)} + (1 - \alpha) Q^{(i)})$
 4. Update $P^{(i+1)} := \alpha P^{(i)} + (1 - \alpha) Q^{(i)}$
end while
return permutation matrix approximation to $P^{(n)}$ via Hungarian Algorithm.

FAQ is bound by proper permutation matrices, GOAT allows for probabilistic (soft) permutations. Thus whereas FAQ takes non-deterministic “choppy” update steps, GOAT optimizes smoothly and deterministically. See the figure below for an illustration. For more details, please refer to our paper by Marchisio et al. (2022).



Figure 22: Optimization step of FAQ vs. GOAT. FAQ arbitrarily chooses the direction of a permutation matrix. GOAT averages permutation matrices to take a smoother path.

3.3.3.1 Results and Discussion

In this section, we improve bilingual lexicon induction performance across 40 language pairs with new a graph-matching method based on optimal transport, called GOAT. The details of GOAT are found in Section 3.4.3. GOAT is especially strong with low amounts of supervision.

Experimental Setup. We run Procrustes, SGM, and GOAT on 40 language pairs. We also perform system combination experiments analogous to the previous section, incorporating GOAT. We evaluate with the standard precision@. Results are illustrated in the figure below.

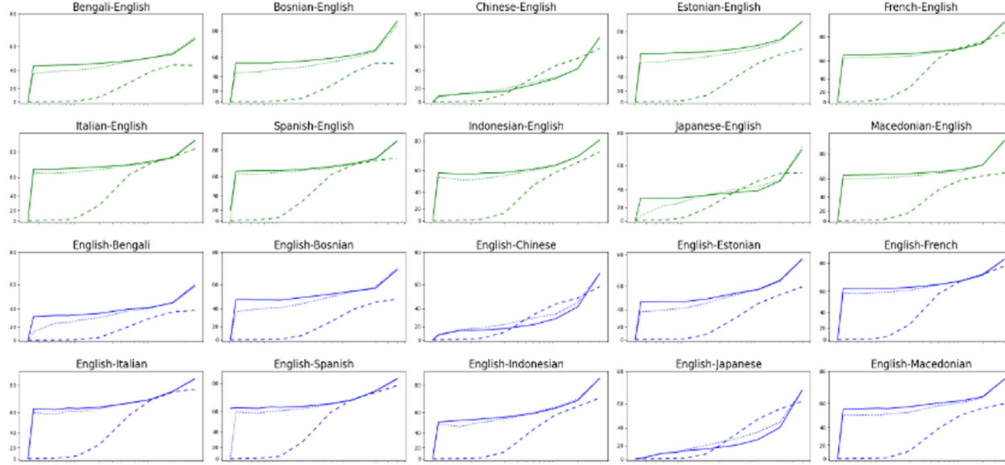


Figure 23: Visualization of our results for select languages. Procrustes (dashed) vs. SGM (dotted) vs. GOAT (solid). X-axis: number of seeds (log scale). Y-axis: Precision@1 (higher is better). GOAT is typically best.

System Combination. The figure below shows the system, which is made of two components: GOAT run in forward and reverse directions, and “Iterative Procrustes with Stochastic-Add” from the previous section. This iterative version of Procrustes runs Procrustes in source-target and target-source directions and feeds H random hypotheses from the intersection of both directions into another run of Procrustes with the gold seeds. The process repeats for I iterations, adding H more random hypotheses each time until all are chosen. We set $H = 100$ and $I = 5$, as in the original work.

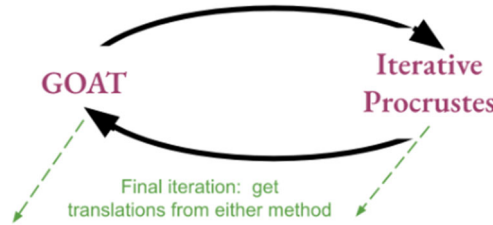


Figure 24: Combination system: Iterative Procrustes (IterProc) and GOAT. (1) run GOAT in forward/reverse directions (2) intersect hypotheses, pass to IterProc, (3) run IterProc forward and reverse (4) intersect hypotheses, pass to Step (1). Repeat N cycles. (End) Get final translations from the forward run of GOAT or IterProc.

Iterating in this fashion generally leads to improvements (see table below). For more detailed results and analysis, please consult Marchisio et al. (2022).

Table 27: $P@1$ of Combination Exps. -EP starts with GOAT, and ends with IterProc. -EG: IterProc, ends with GOAT. Prev is the previous best of prior experiments.

	Prev	-EP	-EG	Prev	-EP	-EG	Prev	-EP	-EG	Prev	-EP	-EG	Prev	-EP	-EG	Prev	-EP	-EG
Seeds	en-bn			bn-en			en-bs			bs-en			en-de			de-en		
25	31.9	44.3	0.5	44.4	53.1	6.2	48.1	58.4	0.4	54.6	61.5	0.6	48.5	61.7	59.1	34.1	59.3	56.8
75	33.1	44.7	39.5	45.2	53.5	49.1	47.9	58.1	55.1	54.6	61.7	57.2	48.8	62.3	59.7	47.0	59.4	57.1
100	33.8	45.3	39.7	45.5	53.9	48.1	47.7	58.1	55.4	55.5	61.3	57.5	49.0	62.2	59.7	47.6	59.4	56.8
2000	45.9	48.7	49.3	55.2	56.6	56.3	58.1	59.9	60.5	65.9	66.6	69.1	63.1	66.3	69.4	60.7	65.1	67.9
4000	60.3	50.5	61.2	65.9	55.2	68.9	70.6	63.4	71.8	86.1	69.7	85.7	74.2	72.9	79.5	71.4	67.2	77.6
	en-et			et-en			en-fa			fa-en			en-id			id-en		
25	47.0	60.4	5.9	63.2	70.2	15.0	42.7	54.4	4.5	45.1	55.1	2.0	51.3	65.8	0.6	58.0	66.7	1.8
75	47.5	60.6	58.6	64.2	69.8	66.5	42.9	54.1	51.9	45.8	55.3	52.0	53.6	66.0	63.3	57.0	66.7	64.2
100	47.3	61.1	59.0	64.3	70.2	66.7	43.0	54.3	52.3	45.6	55.5	52.7	54.3	66.2	63.2	57.8	67.1	63.9
2000	64.0	66.6	67.4	74.2	74.1	75.0	54.7	58.0	58.4	53.8	57.5	56.9	70.7	72.1	74.2	70.1	72.5	72.9
4000	77.4	71.7	80.2	86.4	80.7	87.2	65.9	62.4	67.4	65.5	60.1	67.0	84.3	76.8	86.2	80.5	78.2	83.7
	en-mk			mk-en			en-ms			ms-en			en-ru			ru-en		
25	55.8	63.9	8.0	63.2	68.8	0.6	36.6	62.6	0.9	38.6	65.3	0.2	55.7	67.7	66.1	54.4	63.9	62.0
75	56.5	64.3	63.3	63.6	68.8	67.9	42.6	62.6	59.8	56.8	65.6	62.8	55.7	68.1	67.0	54.8	63.9	61.6
100	56.2	64.1	64.2	64.4	69.4	67.5	42.9	63.0	58.6	57.7	65.8	63.1	55.9	67.9	66.4	55.1	63.8	61.3
2000	64.6	66.8	67.9	71.7	71.0	73.1	65.0	67.0	68.5	67.4	69.4	69.7	67.5	72.6	74.2	68.5	69.3	72.5
4000	75.6	68.9	77.1	88.8	74.1	91.1	79.3	70.7	79.5	77.4	70.0	79.1	83.3	79.3	86.5	89.3	77.4	89.3
	en-ta			ta-en			en-vi			vi-en			en-zh			zh-en		
25	1.8	2.2	0.6	44.4	51.4	2.4	0.4	0.4	0.2	3.3	5.3	0.2	8.7	52.7	1.7	9.3	48.1	0.8
75	30.5	40.4	35.7	45.1	52.4	46.9	22.9	55.0	1.2	45.2	59.6	54.4	17.4	51.6	46.4	14.1	51.1	48.0
100	30.6	40.2	36.8	45.4	52.5	47.9	30.2	55.6	36.2	47.1	59.3	56.3	18.5	51.6	47.3	15.2	51.0	48.0
2000	40.6	42.7	44.2	55.1	55.3	56.2	61.1	67.3	65.8	66.3	73.5	71.5	49.3	58.0	57.7	41.8	57.6	56.8
4000	49.1	44.0	51.7	73.8	65.3	71.6	77.9	73.0	80.5	82.1	80.1	84.3	67.5	66.4	75.1	66.2	65.3	73.3

3.4 JPL Evaluations

We participate in different evaluations for image classification, video recognition, and machine translation. On the task of "Image classification," we performed a thorough analysis of the results from Evaluation 3, which was presented during the Principal Investigator (PI) meeting. Significant performance improvements were emphasized in our discussion.

Our Git repositories are updated with the most recent machine translation and image classification evaluation code. The repository for machine translation can be accessed at https://gitlab.lollll.com/hrl/machine_translation_eval/-/tree/lwl2023, with the tag Evaluation3v2.8. The repository for image classification can be accessed at https://gitlab.lollll.com/hrl/recognition_eval/-/tree/evaluation3 tag v3.4. Comprehensive information on the repositories and their usage can be found in the readme file.

We conducted further analyses on the evaluation 3 results. Figure 25 illustrates the performance comparison in a low-label regime between DECiFR and the Baseline model. DECiFR demonstrated superior performance, outperforming the baseline at 19 out of the 24 regimen checkpoints. Similarly, Figure 26 represents the performance evaluation in a low-label regime, contrasting DECiFR with Kprop. Remarkably, DECiFR also exceeded Kprop's performance at 19 of the 24 regimen checkpoints. These findings underscore the efficacy of our proposed DECiFR approach, particularly in the realm of few-shot learning tasks.

Task 1					Task2					Task 3				
Checkpoint	1	2	3	4	Checkpoint	1	2	3	4	Checkpoint	1	2	3	4
Baseline	0.18	0.3	0.42	0.49	Baseline	0.02	0.05	0.06	0.09	Baseline	0.21	0.43	0.6	0.75
DECiFR	0.3	0.38	0.55	0.63	DECiFR	0.04	0.05	0.07	0.09	DECiFR	0.67	0.75	0.84	0.89
Gain	0.12	0.08	0.13	0.14	Gain	0.02	0	0.01	0	Gain	0.46	0.32	0.24	0.14
Checkpoint					Checkpoint					Checkpoint				
Baseline	0.22	0.32	0.41	0.51	Baseline	0.01	0.02	0.03	0.06	Baseline	0.24	0.36	0.41	0.52
DECiFR	0.35	0.47	0.61	0.66	DECiFR	0.02	0.02	0.03	0.04	DECiFR	0.46	0.49	0.53	0.67
Gain	0.13	0.15	0.2	0.15	Gain	0.01	0	0	-0.02	Gain	0.22	0.13	0.12	0.15

 Greater

Figure 25: Evaluation Performance Low-label Regime. DECiFR vs Baseline for tasks 1, 2, and 3 for few-shot checkpoints (1-4).

Task 1					Task2					Task 3				
Checkpoint	1	2	3	4	Checkpoint	1	2	3	4	Checkpoint	1	2	3	4
Kprop	0.05	0.32	0.4	0.46	Kprop	0.05	0.03	0.04	0.06	Kprop	0.02	0.48	0.6	0.68
DECiFR	0.3	0.38	0.55	0.63	DECiFR	0.04	0.05	0.07	0.09	DECiFR	0.67	0.75	0.84	0.89
Gain	0.25	0.06	0.15	0.17	Gain	-0.01	0.02	0.03	0.03	Gain	0.65	0.27	0.24	0.21
Checkpoint					Checkpoint					Checkpoint				
Kprop	0.05	0.32	0.4	0.46	Kprop	0.02	0.04	0.06	0.1	Kprop	0.01	0.08	0.1	0.11
DECiFR	0.35	0.47	0.61	0.66	DECiFR	0.02	0.02	0.03	0.04	DECiFR	0.46	0.49	0.53	0.67
Gain	0.3	0.15	0.21	0.2	Gain	0	-0.02	-0.03	-0.06	Gain	0.45	0.41	0.43	0.56

 Greater

Figure 26: Evaluation Performance Low label Regime. DECiFR vs Kprop for tasks 1, 2 and 3 for few-shot checkpoints (1-4).

4.0 CONCLUSIONS

The collaboration of HRL Laboratories LLC, Johns Hopkins University, and the University of California Davis has yielded substantial advancements in image classification, self-supervised learning, and machine translation. As part of the Learn with Less Labels (LwLL) program, novel techniques and methods such as the mean-shift algorithm for self-supervised learning, robust defense mechanisms against backdoor attacks, and new approaches for image classification and machine translation have been introduced. Generalized Sliced-Wasserstein Embeddings (GSWE), Part-Based Learning, and Deep Embedded Clustering in Few-shot Representations (DECiFR) have been particularly effective, with DECiFR demonstrating state-of-the-art performance in few-shot image classification. For machine translation, we introduced groundbreaking graph-matching techniques utilizing optimal transport, which has substantially improved performance in data-scarce conditions.

The use of GSWE within a self-supervised learning framework showed promising results, outperforming or matching the Set Transformer method on Set-Circles and Point Cloud MNIST datasets. GSWE also proved more efficient than previous methods due to its simplified architecture. Part-Based Learning successfully outperformed the baseline when applied to the PlacesExtra69 dataset, particularly for cases with five or more labels per class. DECiFR showcased its prowess by greatly outperforming previous models, with an average gain of over 10%, a maximum gain surpassing 35% for 1-shot on Task1, and a remarkable gain exceeding 65% for 1-shot on Task3 compared to our previous Kprop solution. Furthermore, DECiFR stood out by outperforming the baseline on 17 out of 24 Regimen Checkpoints, and the Kprop on 16 out of 24 Regimen Checkpoints.

Our research also introduces several innovative approaches to self-supervised learning (SSL) and deep learning interpretation. Firstly, we propose an effective SSL method, MSF, which groups similar images together by shifting the embedding of an image towards the mean of its nearest neighbors. MSF is identified as a generalized form of Bootstrap Your Own Latent (BYOL) method, exhibiting superior or comparable performance against other state-of-the-art SSL methods, as evidenced by our extensive experiments. However, SSL's dependency on large-scale unlabeled data presents vulnerabilities, as demonstrated by our introduction of a backdoor attack for SSL methods. This attack generates numerous targeted false positives when presented with a test-time trigger and proves more effective for exemplar-based SSL methods such as MoCo v2, BYOL, and MSF than for Jigsaw or RotNet. We suggest that this is due to the manipulation of the embeddings of two augmented views of the same image. The effect of this attack can be diminished through knowledge distillation using some clean data, although the risk of backdoors remains if data can be manipulated. This highlights the importance of using trusted, diverse data sources and considering these vulnerabilities in SSL method development. To augment our understanding of deep models, we've enforced consistency in model interpretation, leveraging ideas from SSL to focus more on the objects of interest rather than background regions. We've also proposed a contrastive learning method, which ensures the consistency of explanations generated by a deep neural network with spatial transformations. Our method, acting as a regularizer, improves explanation heatmaps while maintaining comparable classification accuracy. Notably, this technique boosts classification accuracy on fine-grained classification datasets and demonstrates the potential to utilize unlabeled data to improve accuracy in limited-label settings. Our results call for the community to prioritize quality explanations alongside classification accuracy and focus more attention on the discriminative aspects of the image.

In Machine Translation, we focused on the creation of novel bilingual lexicon induction methods. We examined graph-matching methods for bilingual lexicon induction, treating word embeddings as nodes in a similarity graph instead of vectors in a Euclidean space. This reframing enabled us to explore a node's graph neighborhood without presupposing a linear transformation, presenting a more flexible approach. We introduced variations to this method, including Add-All, Stochastic-Add, and an active learning approach. A significant advancement from our research was the introduction of GOAT (Graph Matching via Optimal Transport), a new graph-matching method that enhanced bilingual lexicon induction performance across 40 language pairs. This method exhibited strong performance even with non-isomorphic graphs, demonstrating its effectiveness in low-supervision scenarios. We coupled GOAT with an iterative version of Procrustes, which added random hypotheses from the intersection of both directions into subsequent runs, thus enabling continual improvement in translation quality.

Our solutions have shown great promise, achieving notable results in Hackathon evaluations for machine translation, video recognition, and image classification. This successful collaboration has shown that the integration of advanced algorithms and innovative techniques can yield significant improvements. Despite these accomplishments, we acknowledge that there is much work yet to be done. Integration of large language models is an exciting prospect that requires careful handling of the balance between open and private data. Incorporating explicit domain knowledge into our models to enhance their understanding and performance is a crucial next step. The swift progression in self/semi-supervised learning, transformers, and foundational models opens numerous research opportunities. Key areas of interest include distribution shift, open-world vocabulary continual learning, robust fine-tuning, and multi-modal approaches.

Finally, we want to express our profound gratitude to the DARPA LwLL program for their generous support. Their commitment to fostering innovation in machine learning and artificial intelligence has played a critical role in the achievements we've made. We're looking forward to future collaborations and to making further strides in these exciting research directions with their support.

5.0 REFERENCES

- [1] Koohpayegani*, Tejankar*, Pirsiavash. (2020). CompRes: Self-Supervised Learning by Compressing Representations. NeurIPS.
- [2] Naderializadeh, N., Kolouri, S., Comer, J. F., Andrews, R. W., & Hoffmann, H. (2021). Set Representation Learning with Generalized Sliced-Wasserstein Embeddings.
- [3] Zhang, C., Cai, Y., Lin, G., & Shen, C. (2020). DeepEMD: Few-Shot Image Classification with Differentiable Earth Mover’s Distance and Structured Classifiers. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 12203–12213.
- [4] Lee, J., Lee, Y., Kim, J., Kosiosek, A., Choi, S., & Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. International Conference on Machine Learning, 3744–3753.
- [5] Zhang, Y., Hare, J., & Prügell-Bennett, A. (2020). FSPool: Learning Set Representations with Feature wise Sort Pooling. International Conference on Learning Representations. arXiv:1906.02795
- [6] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1912–1920.
- [7] Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 652–660.
- [8] Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., & Neumann, M. (2020). TUDataset: A collection of benchmark datasets for learning with graphs. ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020). www.graphlearning.io
- [9] Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. International Conference on Learning Representations (ICLR).
- [10] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph Attention Networks. International Conference on Learning Representations.
- [11] Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How Powerful are Graph Neural Networks? International Conference on Learning Representations.
- [12] Cheng, G., Han, J., & Lu, X. (2017). Remote sensing image scene classification: Benchmark and state of the art. Proceedings of the IEEE, 105(10), 1865–1883.
- [13] Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2017). Places: A 10 million Image Database for Scene Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence.
- [14] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. International Conference on Learning Representations.
- [15] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778.
- [16] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. In ICCV, pages 618–626
- [17] Matthijs Douze, Arthur Szlam, Bharath Hariharan, and Hervé Jégou. Low-shot learning with large-scale diffusion. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3349-3358, 2018. 2, 3

- [18] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3723-3731, 2019. 1, 3
- [19] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Selecting relevant features from a multi-domain representation for few-shot classification. In *European Conference on Computer Vision*, pages 769-786. Springer, 2020. 3
- [20] Li Fei-Fei, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594-611, 2006. 1
- [21] Vincent Feng. An overview of resnet and its variants. *Towards data science*, 2, 2017. 6
- [22] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126-1135. PMLR, 06-11 Aug 2017. 1, 2
- [23] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126-1135. PMLR, 2017. 3
- [24] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep-embedded clustering with local structure preservation. In *Ijcai*, pages 1753-1759, 2017. 3
- [25] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *International conference on neural information processing*, pages 373-382. Springer, 2017. 3
- [26] 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. 6
- [27] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. *Advances in Neural Information Processing Systems*, 32, 2019. 2
- [28] Yuqing Hu, Stéphane Pateux, and Vincent Gripon. Squeezing backbone feature distributions to the max for efficient few-shot learning. *Algorithms*, 15(5):147, 2022. 8
- [29] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.3
- [30] David Hughes, Marcel Salathé, et al. An open access repository of images on plant health to enable the development of mobile disease diagnostics. *arXiv preprint arXiv:1511.08060*, 2015. 6
- [31] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5070-5079, 2019. 2, 3
- [32] Ashraful Islam, Chun-Fu Richard Chen, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, and Richard J Radke. Dynamic distillation network for cross-domain few-shot recognition with unlabeled data. *Advances in Neural Information Processing Systems*, 34:3584-3595, 2021. 1
- [33] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0. Lille, 2015. 2

- [34] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10657-10665, 2019. 1
- [35] SuBeen Lee, WonJun Moon, and Jae-Pil Heo. Task discrepancy maximization for fine-grained few-shot classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5331-5340, 2022. 8
- [36] Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning. arXiv preprint arXiv:2106.09785, 2021. 6
- [37] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 1 – 10, 2019. 2
- [38] Jialin Liu, Fei Chao, and Chih-Min Lin. Task augmentation by rotating for meta-learning. arXiv preprint arXiv: 2003.00804, 2020. 1, 3
- [39] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. arXiv preprint arXiv:1805.10002, 2018. 2
- [40] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, and Yi Yang. Transductive propagation network for few-shot learning. 2018. 3
- [41] Jiawei Ma, Hanchen Xie, Guangxing Han, Shih-Fu Chang, Aram Galstyan, and Wael Abd-Almageed. Partner-assisted learning for few-shot image classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 10573-10582, 2021. 1, 3
- [42] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. arXiv preprint arXiv:1707.03141, 2017. 2
- [43] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In International Conference on Machine Learning, pages 2554-2563. PMLR, 2017. 2
- [44] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In International Conference on Machine Learning, pages 3664-3673. PMLR, 2018. 2
- [45] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. Advances in neural information processing systems, 31, 2018. 2
- [46] Archit Parnami and Minwoo Lee. Learning from few examples: A summary of approaches to few-shot learning. arXiv preprint arXiv:2203.04291, 2022. 2
- [47] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. arXiv preprint arXiv:1909.09157, 2019. 3
- [48] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016. 2
- [49] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. arXiv preprint arXiv:1803.00676, 2018. 2
- [50] Mohammad Rostami, Soheil Kolouri, Eric Eaton, and Kyungnam Kim. Deep transfer learning for few-shot sar image classification. Remote Sensing, 11(11):1374, 2019. 1
- [51] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. arXiv preprint arXiv:1807.05960, 2018. 2

- [52] Tonmoy Saikia, Thomas Brox, and Cordelia Schmid. Optimized generic feature learning for few-shot classification across domains. arXiv preprint arXiv:2001.07926, 2020. 3
- [53] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In International conference on machine learning, pages 1842–1850. PMLR, 2016. 2
- [Ade+18] Julius Adebayo and others. “Sanity checks for saliency maps”. In advances in Neural Information Processing Systems: 2018, pages 9505–9515.
- [Ale17] Natalia Alexandrov. “Explainable AI decisions for human-autonomy interactions”. in 17th AIAA Aviation Technology, Integration, and Operations Conference: 2017, page 3991.
- [Ass+21] Mahmoud Assran and others. “Semi-Supervised Learning of Visual Features by Non-Parametrically Predicting View Assignments with Support Samples”. in ICCV: (2021).
- [ATP20] Soroush Abbasi Koohpayegani, Ajinkya Tejankar and Hamed Pirsiavash. “Compress: Self-supervised learning by compressing representations”. In Advances in Neural Information Processing Systems: 33 (2020), pages 12980–12992.
- [Aza+21] Mehdi Azabou and others. “Mine your own view: Self-supervised learning through across-sample prediction”. in arXiv preprint arXiv:2102.10106: (2021).
- [BCN06] Cristian Bucilua, Rich Caruana and Alexandru Niculescu-Mizil. “Model compression”. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining: 2006, pages 535–541.
- [Bey+21] Lucas Beyer and others. “Knowledge distillation: A good teacher is patient and consistent”. in arXiv preprint arXiv:2106.05237: (2021).
- [BGV14a] Lukas Bossard, Matthieu Guillaumin and Luc Van Gool. “Food-101 – Mining Discriminative Components with Random Forests”. In European Conference on Computer Vision: 2014.
- [BGV14b] Lukas Bossard, Matthieu Guillaumin and Luc Van Gool. “Food-101—mining discriminative components with random forests”. In European conference on computer vision: Springer. 2014, pages 446–461.
- [Bor+21] Eitan Borgnia and others. “Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff”. In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP): IEEE. 2021, pages 3855–3859.
- [BW88] Eric Baum and Frank Wilczek. “Supervised Learning of Probability Distributions by Neural Networks”. in Neural Information Processing Systems: by editor D. Anderson. American Institute of Physics, 1988. URL: <https://proceedings.neurips.cc/paper/1987/file/ecbc87e4b5ce2fe28308fd9f2a7baf3-Paper.pdf>.
- [BYO] official BYOL code. Code and weights for BYOL. <https://github.com/deepmind/deepmind-research/tree/master/byol>.
- [Car+15] Rich Caruana and others. “Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission”. in KDD ’15: 2015.
- [Car+18] Mathilde Caron and others. “Deep clustering for unsupervised learning of visual features”. In Proceedings of the European Conference on Computer Vision (ECCV): 2018, pages 132–149.
- [Car+20] Mathilde Caron and others. “Unsupervised Learning of Visual Features by Contrasting Cluster Assignments”. In Advances in Neural Information Processing Systems: Curran Associates, Inc., 2020, pages 9912–9924. URL: <https://proceedings.neurips.cc/paper/2020/file/70feb62b69f16e0238f741fab228fec2-Paper.pdf>.
- [CH20] Xinlei Chen and Kaiming He. Exploring Simple Siamese Representation Learning. 2020. arXiv: 2011.10566 [cs.CV].

- [CH21] Xinlei Chen and Kaiming He. “Exploring simple siamese representation learning”. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 2021, pages 15750–15758.
- [Che+20a] Ting Chen and others. “A Simple Framework for Contrastive Learning of Visual Representations”. inarXiv preprint arXiv:2002.05709: 2020.
- [Che+20b] Ting Chen and others. “Big Self-Supervised Models are Strong Semi-Supervised Learners”. In Advances in Neural Information Processing Systems: 33 (2020), pages 22243–22255.
- [Che+20c] Xinlei Chen and others. “Improved Baselines with Momentum Contrastive Learning”. In arXiv preprint arXiv:2003.04297: (2020).
- [Che+20d] Xinlei Chen and others. “Improved baselines with momentum contrastive learning”. In arXiv preprint arXiv:2003.04297: (2020).
- [Cim+14a] Mircea Cimpoi and others. “Describing Textures in the Wild”. In Computer Vision and Pattern Recognition: 2014.
- [Cim+14b] Mircea Cimpoi and others. “Describing textures in the wild”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 2014, pages 3606–3613.
- [coda] official cmc code. Contrastive Multiview Coding. <https://github.com/HobbitLong/CMC>.
- [codb] official moco code. PyTorch implementation of MoCo: <https://arxiv.org/abs/1911.05722>. <https://github.com/facebookresearch/moco>.
- [code] official simclr code. SimCLR - A Simple Framework for Contrastive Learning of Visual Representations <https://arxiv.org/abs/2002.05709>. <https://github.com/google-research/simclr>.
- [codd] official swav code. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. <https://github.com/facebookresearch/swav>.
- [CT22] Nicholas Carlini and Andreas Terzis. “Poisoning and Backdooring Contrastive Learning”. In International Conference on Learning Representations (ICLR): 2022.
- [CTP20] Edward Chou, Florian Tramèr and Giancarlo Pellegrino. “Sentinet: Detecting localized universal attacks against deep learning systems”. in 2020 IEEE Security and Privacy Workshops (SPW): IEEE. 2020, pages 48–54.
- [Cub+20] Ekin Dogus Cubuk and others. “RandAugment: Practical Automated Data Augmentation with a Reduced Search Space”. in Advances in Neural Information Processing Systems: by editor H. Larochelle and others. volume 33. Curran Associates, Inc., 2020, pages 18613–18624. URL: <https://proceedings.neurips.cc/paper/2020/file/d85b63ef0ccb114d0a3bb7b7d808028f-Paper.pdf>.
- [CXH21] Xinlei Chen, Saining Xie and Kaiming He. “An empirical study of training self-supervised vision transformers”. In Proceedings of the IEEE/CVF International Conference on Computer Vision: 2021, pages 9640–9649.
- [DAR20] Bao Gia Doan, Ehsan Abbasnejad and Damith C Ranasinghe. “Februus: Input purification defense against trojan attacks on deep neural network systems”. in Annual Computer Security Applications Conference: 2020, pages 897–912.
- [Den+09] Jia Deng and others. “Imagenet: A large-scale hierarchical image database”. in 2009 IEEE conference on computer vision and pattern recognition: Ieee. 2009, pages 248–255.
- [DGE15] Carl Doersch, Abhinav Gupta and Alexei A Efros. “Unsupervised visual representation learning by context prediction”. In Proceedings of the IEEE International Conference on Computer Vision: 2015, pages 1422–1430.

- [DKD16] Jeff Donahue, Philipp Krähenbühl and Trevor Darrell. “Adversarial feature learning”. In International Conference on Learning Representations, ICLR: 2016.
- [Dos+20] Alexey Dosovitskiy and others. “An image is worth 16x16 words: Transformers for image recognition at scale”. inarXiv preprint arXiv:2010.11929: (2020).
- [Dos+21] Alexey Dosovitskiy and others. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In International Conference on Learning Representations: 2021. URL: <https://openreview.net/forum?id=YicbFdNTTy>.
- [DT17] Terrance DeVries and Graham W Taylor. “Improved regularization of convolutional neural networks with cutout”. inarXiv preprint arXiv:1708.04552: (2017).
- [Dwi+21] Debidatta Dwibedi and others. With a Little Help from My Friends: Nearest-Neighbor Contrastive Learning of Visual Representations. 2021. arXiv: 2104.14548 [cs.CV].
- [Erm+20a] Aleksandr Ermolov and others. Official repository of the paper Whitening for Self-Supervised Representation Learning. <https://github.com/hdt/self-supervised>. 2020.
- [Erm+20b] Aleksandr Ermolov and others. “Whitening for self-supervised representation learning”. In arXiv preprint arXiv:2007.06346: (2020).
- [Eve+10] Mark Everingham and others. “The pascal visual object classes (voc) challenge”. In International journal of computer vision: 88.2 (2010), pages 303–338.
- [FAIa] FAISS. A library for efficient similarity search and clustering of dense vectors. <https://github.com/facebookresearch/faiss>.
- [FAIb] official FAISS code. A library for efficient similarity search and clustering of dense vectors. <https://github.com/facebookresearch/faiss>.
- [Fan+21] Zhiyuan Fang and others. “SEED: Self-supervised Distillation For Visual Representation”. in International Conference on Learning Representations: 2021. URL: <https://openreview.net/forum?id=AHm3dbp7D1D>.
- [FFP04a] Li Fei-Fei, Rob Fergus and Pietro Perona. “Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories”. In Computer Vision and Pattern Recognition Workshop: (2004).
- [FFP04b] Li Fei-Fei, Rob Fergus and Pietro Perona. “Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories”. in2004 conference on computer vision and pattern recognition workshop: IEEE. 2004, pages 178–178.
- [FXT19] Zeyu Feng, Chang Xu and Dacheng Tao. “Self-supervised representation learning by rotation feature decoupling”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 2019, pages 10364–10374.
- [Gc21] Jacob Gildenblat and contributors. PyTorch library for CAM methods. <https://github.com/jacobgil/pytorch-grad-cam>. 2021.
- [Gid+21] Spyros Gidaris and others. “OBoW: Online Bag-of-Visual-Words Generation for Self-Supervised Learning”. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR): june 2021, pages 6830–6840.
- [Gir15] Ross Girshick. “Fast r-cnn”. In Proceedings of the IEEE international conference on computer vision: 2015, pages 1440–1448.
- [Goy+19] Priya Goyal and others. “Scaling and benchmarking self-supervised visual representation learning”. In Proceedings of the IEEE International Conference on Computer Vision: 2019, pages 6391–6400.
- [Goy+21] Priya Goyal and others. “Self-supervised Pretraining of Visual Features in the Wild”. In arXiv preprint arXiv:2103.01988: (2021).

- [Gri+20a] Jean-Bastien Grill and others. “Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning”. in *Advances in Neural Information Processing Systems*: volume 33. 2020, pages 21271–21284.
- [Gri+20b] Jean-Bastien Grill and others. “Bootstrap your own latent: A new approach to self-supervised learning”. in *arXiv preprint arXiv:2006.07733*: (2020).
- [GSK18a] Spyros Gidaris, Praveer Singh and Nikos Komodakis. Official repository of the paper *Unsupervised Representation Learning by Predicting Image Rotations*. <https://github.com/gidariss/FeatureLearningRotNet>. 2018.
- [GSK18b] Spyros Gidaris, Praveer Singh and Nikos Komodakis. “Unsupervised Representation Learning by Predicting Image Rotations”. *International Conference on Learning Representations*: 2018.
- [GSS14a] Ian J Goodfellow, Jonathon Shlens and Christian Szegedy. “Explaining and harnessing adversarial examples”. in *arXiv preprint arXiv:1412.6572*: 2014.
- [GSS14b] Ian J. Goodfellow, Jonathon Shlens and Christian Szegedy. “Explaining and Harnessing Adversarial Examples”. in *CoRR*: volume abs/1412.6572. 2014.
- [Guo+19] Hao Guo and others. “Visual Attention Consistency Under Image Transforms for Multi-Label Image Classification”. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*: june 2019.
- [HCL06] 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. 2006, pages 1735–1742.
- [He+15] Kaiming He and others. “Deep Residual Learning for Image Recognition”. in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 2015, pages 770–778.
- [He+16] Kaiming He and others. “Deep residual learning for image recognition”. In *Proceedings of the IEEE conference on computer vision and pattern recognition*: 2016, pages 770–778.
- [He+19] Kaiming He and others. “Momentum Contrast for Unsupervised Visual Representation Learning”. in *ArXiv*: volume abs/1911.05722. 2019.
- [He+20] Kaiming He and others. “Momentum contrast for unsupervised visual representation learning”. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*: 2020, pages 9729–9738.
- [He+21a] Kaiming He and others. “Masked autoencoders are scalable vision learners”. in *arXiv preprint arXiv:2111.06377*: (2021).
- [He+21b] Kaiming He and others. *Masked Autoencoders: A PyTorch Implementation*. <https://github.com/facebookresearch/mae>. 2021.
- [He+22] Kaiming He and others. “Masked autoencoders are scalable vision learners”. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*: 2022, pages 16000–16009.
- [Hén+19] Olivier J Hénaff and others. “Data-efficient image recognition with contrastive predictive coding”. in *arXiv preprint arXiv:1905.09272*: (2019).
- [Hon+20] Sanghyun Hong and others. “On the effectiveness of mitigating data poisoning attacks with gradient shaping”. in *arXiv preprint arXiv:2002.11497*: (2020).
- [How+17] Andrew G Howard and others. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. in *arXiv preprint arXiv:1704.04861*: (2017).
- [Hua+17] Gao Huang and others. “Densely connected convolutional networks”. In *Proceedings of the IEEE conference on computer vision and pattern recognition*: 2017, pages 4700–4708.

- [Hua+19a] Jiabo Huang and others. “Unsupervised Deep Learning by Neighbourhood Discovery”. in byeditorKamalika Chaudhuri and Ruslan Salakhutdinov: volume 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, september 2019, pages 2849–2858. URL: <http://proceedings.mlr.press/v97/huang19b.html>.
- [Hua+19b] Yanping Huang and others. “Gpipe: Efficient training of giant neural networks using pipeline parallelism”. In Advances in neural information processing systems: 32 (2019), pages 103–112.
- [Huy+20] Tri Huynh and others. Boosting Contrastive Self-Supervised Learning with False Negative Cancellation. 2020. arXiv: 2011.11765 [cs.CV].
- [HVD15] Geoffrey Hinton, Oriol Vinyals and Jeff Dean. “Distilling the knowledge in a neural network”. in arXiv preprint arXiv:1503.02531: (2015).
- [Ian+16] Forrest N Iandola and others. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size”. in arXiv preprint arXiv:1602.07360: (2016).
- [JF18] Simon Jenni and Paolo Favaro. “Self-supervised feature learning by learning to spot artifacts”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 2018, pages 2733–2742.
- [JHV19] Xu Ji, João F Henriques and Andrea Vedaldi. “Invariant information clustering for unsupervised image classification and segmentation”. In Proceedings of the IEEE International Conference on Computer Vision: 2019, pages 9865–9874.
- [Kal+20] Yannis Kalantidis and others. “Hard negative mixing for contrastive learning”. In Advances in Neural Information Processing Systems: (2020).
- [KH+09] Alex Krizhevsky, Geoffrey Hinton and others. “Learning multiple layers of features from tiny images”. in (2009).
- [Kho+20] Prannay Khosla and others. “Supervised Contrastive Learning”. In Advances in Neural Information Processing Systems: 33 (2020).
- [Kin+17] Pieter-Jan Kindermans and others. “The (un) reliability of saliency methods”. in arXiv preprint arXiv:1711.00867: 2017.
- [Kra+13a] Jonathan Krause and others. “3D Object Representations for Fine-Grained Categorization”. In Workshop on 3D Representation and Recognition: Sydney, Australia, 2013.
- [Kra+13b] Jonathan Krause and others. “3D Object Representations for Fine-Grained Categorization”. in 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13): Sydney, Australia, 2013.
- [Kra+13c] Jonathan Krause and others. “3d object representations for fine-grained categorization”. In Proceedings of the IEEE international conference on computer vision workshops: 2013, pages 554–561.
- [Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. techreport. University of Toronto, 2009.
- [KSH12] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. in NIPS: 2012.
- [KTP20] Soroush Abbasi Koohpayegani, Ajinkya Tejankar and Hamed Pirsiavash. “Compress: Self-supervised learning by compressing representations”. in NeurIPS: (2020).
- [KTP21a] Soroush Abbasi Koohpayegani, Ajinkya Tejankar and Hamed Pirsiavash. “Mean shift for self-supervised learning”. In Proceedings of the IEEE/CVF International Conference on Computer Vision: 2021, pages 10326–10335.
- [KTP21b] Soroush Abbasi Koohpayegani, Ajinkya Tejankar and Hamed Pirsiavash. Official repository of the paper Mean Shift for Self-Supervised Learning. <https://github.com/UM-BCvision/MSF>. 2021.

[Kuh55] Harold W Kuhn. “The Hungarian method for the assignment problem”. in *Naval research logistics quarterly*: 2.1-2 (1955), pages 83–97.

[Lee+20] Kibok Lee and others. “i-Mix: A Domain-Agnostic Strategy for Contrastive Representation Learning”. In *International Conference on Learning Representations*: 2020.

[Lee+21] Kibok Lee and others. “i-Mix: A Domain-Agnostic Strategy for Contrastive Representation Learning”. In *International Conference on Learning Representations*: 2021. URL: <https://openreview.net/forum?id=T6AxtOaWydQ>.

[LF88] Esther Levin and Michael Fleisher. “Accelerated learning in layered neural networks”. In *Complex systems*: 2.625-640 (1988), page 3.

[Li+19] Kunpeng Li and others. “Guided Attention Inference Network.” in *IEEE transactions on pattern analysis and machine intelligence*: 2019.

[Li+22] Yiming Li and others. “Backdoor learning: A survey”. in *IEEE Transactions on Neural Networks and Learning Systems*: (2022).

[Lin+14] Tsung-Yi Lin and others. “Microsoft coco: Common objects in context”. In *European conference on computer vision*: Springer. 2014, pages 740–755.

[MAE] official MAE code. MAE models and code. <https://github.com/facebookresearch/mae>.

[Maj+13a] S. Maji and others. *Fine-Grained Visual Classification of Aircraft*. techreport. 2013. arXiv: 1306.5151 [cs-cv].

[Maj+13b] Subhansu Maji and others. “Fine-grained visual classification of aircraft”. in *arXiv preprint arXiv:1306.5151*: (2013).

[Maj+13c] Subhansu Maji and others. “Fine-grained visual classification of aircraft”. in *arXiv preprint arXiv:1306.5151*: 2013.

[MM19] Ishan Misra and Laurens van der Maaten. “Self-supervised learning of pretext-invariant representations”. in *arXiv preprint arXiv:1912.01991*: (2019).

[MoCa] official MoCo code. Official PyTorch code base for MoCo. <https://github.com/facebookresearch/moco>.

[MoCb] official MoCo V3 code. MoCo-v3 models and code. <https://github.com/facebookresearch/moco-v3>.

[mod] official pytorch models. Torchvision Models. <https://pytorch.org/docs/stable/torchvision/models.html>.

[NF16a] Mehdi Noroozi and Paolo Favaro. “Unsupervised learning of visual representations by solving jigsaw puzzles”. in *European Conference on Computer Vision*: Springer. 2016, pages 69–84.

[NF16b] Mehdi Noroozi and Paolo Favaro. “Unsupervised Learning of Visual Representations by solving Jigsaw Puzzles”. in *ECCV*: 2016.

[Nor+18] Mehdi Noroozi and others. “Boosting self-supervised learning via knowledge transfer”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*: 2018, pages 9359–9367.

[NPF17a] Mehdi Noroozi, Hamed Pirsiavash and Paolo Favaro. “Representation Learning by Learning to Count”. in *2017 IEEE International Conference on Computer Vision (ICCV)*: 2017, pages 5899–5907.

[NPF17b] Mehdi Noroozi, Hamed Pirsiavash and Paolo Favaro. “Representation learning by learning to count”. In *Proceedings of the IEEE International Conference on Computer Vision*: 2017, pages 5898–5906.

- [NZ08a] Maria-Elena Nilsback and Andrew Zisserman. “Automated Flower Classification over a Large Number of Classes”. in Indian Conference on Computer Vision, Graphics and Image Processing: 2008.
- [NZ08b] Maria-Elena Nilsback and Andrew Zisserman. “Automated Flower Classification over a Large Number of Classes”. in Indian Conference on Computer Vision, Graphics and Image Processing: december 2008.
- [NZ08c] 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: IEEE. 2008, pages 722–729.
- [Par+12a] O. M. Parkhi and others. “Cats and Dogs”. in Computer Vision and Pattern Recognition: 2012.
- [Par+12b] Omkar M Parkhi and others. “Cats and dogs”. in 2012 IEEE conference on computer vision and pattern recognition: IEEE. 2012, pages 3498–3505.
- [Par+19] Wonpyo Park and others. “Relational knowledge distillation”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 2019, pages 3967–3976.
- [Pas+19] Adam Paszke and others. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. in NeurIPS: 2019.
- [Pat+16] Deepak Pathak and others. “Context encoders: Feature learning by inpainting”. In Proceedings of the IEEE conference on computer vision and pattern recognition: 2016, pages 2536–2544.
- [PDS18] Vitali Petsiuk, Abir Das and Kate Saenko. “RISE: Randomized Input Sampling for Explanation of Black-box Models”. In Proceedings of the British Machine Vision Conference (BMVC): 2018.
- [Pen+19] Baoyun Peng and others. “Correlation congruence for knowledge distillation”. In Proceedings of the IEEE International Conference on Computer Vision: 2019, pages 5007–5016.
- [Pey+17] Julia Peyre and others. “Weakly-supervised learning of visual relations”. in ICCV: 2017.
- [PG20] Senthil Purushwalkam Shiva Prakash and Abhinav Gupta. “Demystifying Contrastive Self-Supervised Learning: Invariances, Augmentations and Dataset Biases”. in Advances in Neural Information Processing Systems: 33 (2020).
- [Pha+21] Hieu Pham and others. “Meta pseudo labels”. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 2021, pages 11557–11568.
- [PP21] Vipin Pillai and Hamed Pirsiavash. “Explainable Models with Consistent Interpretations”. in AAI: AAI, 2021.
- [PT18] Nikolaos Passalis and Anastasios Tefas. “Learning deep representations with probabilistic knowledge transfer”. In Proceedings of the European Conference on Computer Vision (ECCV): 2018, pages 268–284.
- [PyTa] official PyTorch ImageNet supervised training code. Official PyTorch supervised ImageNet training code. <https://github.com/pytorch/examples/blob/master/imagenet/main.py>.
- [PyTb] official PyTorch ResNet model. Official PyTorch ResNet-50 pretrained model. <https://pytorch.org/vision/stable/models.html>.
- [Ree+21] Colorado J Reed and others. “Selfaugument: Automatic augmentation policies for self-supervised learning”. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 2021, pages 2674–2683.
- [Ren+15] Shaoqing Ren and others. “Faster R-CNN: Towards real-time object detection with region proposal networks”. in NIPS: 2015, pages 91–99.

- [RHD17] Andrew Slavin Ross, Michael C. Hughes and Finale Doshi-Velez. “Right for the Right Reasons: Training Differentiable Models by Constraining their Explanations”. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17: 2017, pages 2662–2670.
- [RHW86] David E Rumelhart, Geoffrey E Hinton and Ronald J Williams. “Learning representations by back-propagating errors”. *innature*: 323.6088 (1986), pages 533–536.
- [Rom+15] Adriana Romero and others. “FitNets: Hints for Thin Deep Nets”. in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings: by editor Yoshua Bengio and Yann LeCun. 2015. URL: <http://arxiv.org/abs/1412.6550>.
- [Rus+15a] Olga Russakovsky and others. “ImageNet Large Scale Visual Recognition Challenge”. In International booktitle of Computer Vision (IJCV): 2015.
- [Rus+15b] Olga Russakovsky and others. “ImageNet Large Scale Visual Recognition Challenge”. In International Journal of Computer Vision (IJCV): 115.3 (2015), pages 211–252. DOI: 10.1007/s11263-015-0816-y.
- [Rus+15c] Olga Russakovsky and others. “Imagenet large scale visual recognition challenge”. In International journal of computer vision: 115.3 (2015), pages 211–252.
- [Sah+19] Aniruddha Saha and others. “Adversarial Patches Exploiting Contextual Reasoning in Object Detection”. in ArXiv: volume abs/1910.00068. 2019.
- [Sah+22] Aniruddha Saha and others. “Backdoor attacks on self-supervised learning”. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR): 2022.
- [SAK20] Mingjie Sun, Siddhant Agarwal and J Zico Kolter. “Poisoned classifiers are not only backdoored, they are fundamentally broken”. in arXiv preprint arXiv:2010.09080: (2020).
- [San+18] Mark Sandler and others. “Mobilenetv2: Inverted residuals and linear bottlenecks”. In Proceedings of the IEEE conference on computer vision and pattern recognition: 2018, pages 4510–4520.
- [Sel+16] Ramprasaath R. Selvaraju and others. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. in 2017 IEEE International Conference on Computer Vision (ICCV): 2016, pages 618–626.
- [Sel+17] Ramprasaath R Selvaraju and others. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In Proceedings of the IEEE international conference on computer vision: 2017, pages 618–626.
- [SF19] Suraj Srinivas and François Fleuret. “Full-Gradient Representation for Neural Network Visualization”. In Advances in Neural Information Processing Systems (NeurIPS): 2019.
- [Sin+20] Krishna Kumar Singh and others. “Don’t Judge an Object by Its Context: Learning to Overcome Contextual Bias”. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition: 2020, pages 11070–11078.
- [Soh+20] Kihyuk Sohn and others. “FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence”. In Advances in Neural Information Processing Systems: 33 (2020).
- [SPP19a] Akshayvarun Subramanya, Vipin Pillai and Hamed Pirsiavash. “Fooling Network Interpretation in Image Classification”. In Proceedings of the IEEE International Conference on Computer Vision: 2019, pages 2020–2029.
- [SPP19b] Akshayvarun Subramanya, Vipin Pillai and Hamed Pirsiavash. “Fooling Network Interpretation in Image Classification”. In The IEEE International Conference on Computer Vision (ICCV): october 2019.

- [SSL]official SSL Backdoor code. Code for Backdoor Attacks on Self-Supervised Learning paper. <https://github.com/UMBCvision/SSL-Backdoor>.
- [SSP20] Aniruddha Saha, Akshayvarun Subramanya and Hamed Pirsiavash. “Hidden trigger backdoor attacks”. In Proceedings of the AAAI Conference on Artificial Intelligence: volume 34. 07. 2020, pages 11957–11965.
- [SSZ17] Jake Snell, Kevin Swersky and Richard S Zemel. “Prototypical networks for few-shot learning”. inarXiv preprint arXiv:1703.05175: (2017).
- [STP21] Koohpayegani Soroush Abbasi, Ajinkya Tejankar and Hamed Pirsiavash. “Mean Shift for Self-Supervised Learning”. inInternational Conference on Computer Vision (ICCV): 2021.
- [Tej+21] Ajinkya Tejankar and others. “ISD: Self-Supervised Learning by Iterative Similarity Distillation”. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV): october 2021, pages 9609–9618.
- [TG20] Erico Tjoa and Cuntai Guan. “A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI”. inIEEE Transactions on Neural Networks and Learning Systems: IEEE, 2020.
- [Tia+20] Yonglong Tian and others. “What makes for good views for contrastive learning”. In arXiv preprint arXiv:2005.10243: (2020).
- [TKI19a] Yonglong Tian, Dilip Krishnan and Phillip Isola. “Contrastive multiview coding”.In arXiv preprint arXiv:1906.05849: (2019).
- [TKI19b] Yonglong Tian, Dilip Krishnan and Phillip Isola. “Contrastive multiview coding”.In arXiv preprint arXiv:1906.05849: (2019).
- [TKI20a] Yonglong Tian, Dilip Krishnan and Phillip Isola. “Contrastive multiview coding”.In European conference on computer vision: Springer. 2020, pages 776–794. [TKI20b] Yonglong Tian, Dilip Krishnan and Phillip Isola. “Contrastive Representationistillation”. In International Conference on Learning Representations: 2020. URL:<https://openreview.net/forum?id=SkgpBJrtvS>.
- [TKI20c] Yonglong Tian, Dilip Krishnan and Phillip Isola. “Contrastive representation distillation”. In ICLR: (2020).
- [TL19] Mingxing Tan and Quoc Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In International conference on machine learning: PMLR. 2019, pages 6105–6114.
- [TM19] Frederick Tung and Greg Mori. “Similarity-preserving knowledge distillation”. In Proceedings of the IEEE International Conference on Computer Vision: 2019, pages 1365–1374.
- [TV17] Antti Tarvainen and Harri Valpola. “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results”. In Advances in neural information processing systems: 2017, pages 1195–1204.
- [Vel19] Alfredo Vellido. “The importance of interpretability and visualization in machine learning for applications in medicine and health care”. In Neural Computing and Applications: Springer, 2019, pages 1–15.
- [Wan+20] Feng Wang and others. “Unsupervised Representation Learning by Invariance Propagation”. inAdvances in Neural Information Processing Systems: 2020. URL: <https://proceedings.neurips.cc/paper/2020/file/23af4b45f1e166141a790d1a3126e77a-Paper.pdf>.
- [Wan+21] Guangrun Wang and others. Solving Inefficiency of Self-supervised Representation Learning. 2021. arXiv: 2104.08760 [cs.CV].

- [Wei+20] Chen Wei and others. “Co2: Consistent contrast for unsupervised visual representation learning”. inarXiv preprint arXiv:2010.02217: (2020).
- [Wel+10] Peter Welinder and others. “Caltech-UCSD birds 200”. In California Institute of Technology: 2010.
- [WI20a] Tongzhou Wang and Phillip Isola. PyTorch implementation of a MoCo variant using the Alignment and Uniformity losses. https://github.com/SsnL/moco_align_uniform. 2020.
- [WI20b] Tongzhou Wang and Phillip Isola. “Understanding contrastive representation learning through alignment and uniformity on the hypersphere”. In International Conference on Machine Learning: PMLR. 2020, pages 9929–9939.
- [Wu+18] Zhirong Wu and others. “Unsupervised feature learning via non-parametric instance discrimination”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 2018, pages 3733–3742.
- [Xia+10a] Jianxiong Xiao and others. “SUN database: Large-scale scene recognition from abbey to zoo”. in2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition: 2010, pages 3485–3492. DOI: 10.1109/CVPR.2010.5539970.
- [Xia+10b] Jianxiong Xiao and others. “SUN database: Large-scale scene recognition from abbey to zoo”. inComputer Vision and Pattern Recognition: 2010.
- [Xie+20] Qizhe Xie and others. “Unsupervised data augmentation for consistency training”. In NeurIPS: (2020).
- [Yan+20] Xueting Yan and others. “ClusterFit: Improving Generalization of Visual Representations”. inCVPR: 2020.
- [YCA20] Asano YM., Rupprecht C. and Vedaldi A. “Self-labelling via simultaneous clustering and representation learning”. In International Conference on Learning Representations: 2020. URL: <https://openreview.net/forum?id=Hyx-jyBFPr>.
- [Yu+19] Lu Yu and others. “Learning Metrics From Teachers: Compact Networks for Image Embedding”. inThe IEEE Conference on Computer Vision and Pattern Recognition (CVPR): june 2019.
- [Yun+19] Sangdoo Yun and others. “Cutmix: Regularization strategy to train strong classifiers with localizable features”. In Proceedings of the IEEE/CVF international conference on computer vision: 2019, pages 6023–6032.
- [Zha+16] Jianming Zhang and others. “Top-Down Neural Attention by Excitation Back-prop”. In ECCV: 2016.
- [Zha+19a] Xiaohua Zhai and others. “S4L: Self-Supervised Semi-Supervised Learning”. In The IEEE International Conference on Computer Vision (ICCV): october 2019.
- [Zha+19b] Liheng Zhang and others. “Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 2019, pages 2547–2555.
- [Zho+14] Bolei Zhou and others. “Learning deep features for scene recognition using places database”. In Advances in neural information processing systems: 2014, pages 487–495.
- [Zho+15] Bolei Zhou and others. “Learning Deep Features for Discriminative Localization”. in2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR): 2015, pages 2921–2929.
- [Zho+20] Zhun Zhong and others. “Random erasing data augmentation”. In Proceedings of the AAAI conference on artificial intelligence: volume 34. 07. 2020, pages 13001–13008.
- [ZIE16a] Richard Zhang, Phillip Isola and Alexei A Efros. “Colorful image colorization”. In European conference on computer vision: Springer. 2016, pages 649–666. [ZIE16b] Richard Zhang, Phillip Isola and Alexei A. Efros. “Colorful Image Colorization”.

- [ZIE17] Richard Zhang, Phillip Isola and Alexei A Efros. “Split-brain autoencoders: Unsupervised learning by cross-channel prediction”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition: 2017, pages 1058–1067.
- [ZK16] Sergey Zagoruyko and Nikos Komodakis. “Wide Residual Networks”. In British Machine Vision Conference 2016: British Machine Vision Association. 2016.
- [ZZY19] Chengxu Zhuang, Alex Lin Zhai and Daniel Yamins. “Local aggregation for unsupervised learning of visual embeddings”. In Proceedings of the IEEE International Conference on Computer Vision: 2019, pages 6002–6012.

APPENDIX A – PUBLICATIONS AND PRESENTATIONS

- Koohpayegani*, Tejankar*, Pirsiavash. (2020). CompRes: Self-Supervised Learning by Compressing Representations. *NeurIPS*.
- Naderializadeh, N., Kolouri, S., Comer, J. F., Andrews, R. W., & Hoffmann, H. (2021). Set Representation Learning with Generalized Sliced-Wasserstein Embeddings.
- Pillai, Pirsiavash, "Explainable Models with Consistent Interpretations", AAAI'21
- Naderializadeh, N., Comer, J.F., Andrews, R.W., Hoffmann, H., and Kolouri, S., 2021. Pooling by Sliced-Wasserstein Embedding.
- Soheil Kolouri, Navid Naderializadeh, Gustavo Rohde, Heiko Hoffmann. "Wasserstein Embedding for Graph Learning." Accepted to ICLR 2021
- Comer, Joseph F., Philip L. Jacobson, and Heiko Hoffmann. "Few-Shot Image Classification Along Sparse Graphs." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.
- Pillai, Koohpayegani, Ouligian, Fong, Pirsiavash, "Consistent Explanations by Contrastive Learning", CVPR'22
- Aniruddha Saha, Ajinkya Tejankar, Soroush Abbasi Koohpayegani, Hamed Pirsiavash, "Backdoor Attacks on Self-Supervised Learning", CVPR 2022
- Yasaman Esfandiari, et al.: Deep Embedded Clustering in Few-shot Representations (DECiFR)
- "When Does Unsupervised Machine Translation Work?", Marchisio et al., ACL 2020.
- "An Analysis of Euclidean vs. Graph-Based Framing for Bilingual Lexicon Induction from Word Embedding Spaces", Marchisio et al., EMNLP 2021
- Bitext Mining for Low-Resource Languages via Contrastive Learning, Weiting Tan, Philipp Koehn, arXiv
- "Bilingual Lexicon Induction for Low-Resource Languages using Graph Matching via Optimal Transport", Marchisio et al., EMNLP 2022
- "IsoVec: Controlling the Relative Isomorphism of Word Embedding Spaces", Marchisio et al., EMNLP 2022
- Multilingual Representation Distillation with Contrastive Learning, Weiting Tan, Kevin Heffernan, Holger Schwenk, Philipp Koehn, EACL 2023
- Ajinkya Tejankar, Maziar Sanjabi, Qifan Wang, Sinong Wang, Hamed Firooz, Hamed Pirsiavash, and Liang Tan, "Defending Against Patch-based Backdoor Attacks on Self-Supervised Learning", CVPR'23

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

Acronym	Full Form
BLI	Bilingual Lexicon Induction
BYOL	Bootstrap Your Own Latent
CH	Content Heatmap
CGC	Contrastive Grad-CAM Consistency
DOD	Department of Defense
DEC	Deep Embedded Clustering
DECIFR	Deep Embedded Clustering In Few-Shot Representations
FAQ	Fast Approximate Quadratic Assignment Problem Algorithm
FIFO	First-In-First-Out
FSPool	Feature Sort Pooling
FSL	Few-Shot Learning
GAP	Global Average Pooling
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GIN	Graph Isomorphism Network
GNNS	Graph Neural Networks
GOAT	Graph Matching Via Optimal Transport
GSWE	Generalized Sliced-Wasserstein Embeddings
GSW	Generalized Sliced Wasserstein
HTBA	Hidden Trigger Backdoor Attacks
IAUC	Insertion AUC Score
ISAB	Induced Set Attention Block
LCE	Cross-Entropy Loss
LCGC	Grad-CAM Consistency Loss
LOT	Lightspeed Optimal Transport
MAE	Masked Auto-Encoders
MLP	Multi-Layer Perceptron
MT	Machine Translation
MSE	Mean Squared Error
MSF	Mean Shift
NMT	Neural Machine Translation
NN	Nearest Neighbors
PMA	Pooling By Multi-Head Attention
PSWE	Pooling By Sliced Wasserstein Embedding
SSL	Self-Supervised Learning
SGM	Seeded Graph Matching
SSG	Source Similarity Graph
SMT	Statistical Machine Translation
WEGL	Wasserstein Embedding For Graph Learning
ZSL	State-Of-The-Art Zero-Shot Learning