The Hidden Costs on Distributional Shifts when Fine-tuning
Joint Text-Image Encoders and Redemptions

Andrew Geng
University of Wisconsin-Madison*
Madison, WI
ageng@wisc.edu

Pin-Yu Chen
IBM Research
Yorktown Heights, NY
pin-yu.chen@ibm.com

Abstract

When considering the performance of a pre-trained model, transferred to a down-stream task, it is important to account for both the model’s generalization and detection capabilities on out-of-distribution (OOD) samples. In this paper, we unveil the hidden costs of intrusive fine-tuning techniques. Specifically, we show that (1) common fine-tuning techniques can distort not only the representations necessary for domain generalization (OOD Generalization), but also the representations necessary for detecting semantic shifted OOD samples (OOD Detection). Additionally, we propose a novel reprogramming approach called reprogrammer which attempts to mitigate these degradations found in common fine-tuning techniques. We show that our reprogrammer method is (2) less intrusive and can lead to better retention of pre-training representation. Subsequently, by maintaining more pre-training representation, we have found that reprogrammer performs better holistically when accounting for the in-distribution (ID), OOD Generalization, and OOD Detection performances of the down-stream model.

1. Introduction

There are many fundamental hurdles obstructing researchers from improving OOD Generalization and OOD Detection performances in deep learning networks. These challenges can range from difficulties in encapsulating covariant (domain) shifts, to overconfidence when predicting on semantically shifted samples [24, 27, 33]. One framework of training deep learning models, that has shown impressive OOD Generalization and OOD Detection performance, is large text-image supervised pre-trained models [16, 28, 29].

However, recently it has been made more aware that common transfer learning techniques can distort the strong representations learned during pre-training, resulting in a degradation in specifically the model’s OOD Generalization performance [1, 20, 38]. In this paper, we present evidence showing that common transfer learning methods, such as linear-probing (optimizing just the classification head) and full fine-tuning (optimizing all model parameters), each have their own strengths and hidden costs in terms of ID and OOD performances. More specifically, we present evidence showing that these common transfer learning techniques can degrade not only OOD Generalization but also OOD Detection performance. This subsequently beckons the question can we build a different transfer learning technique that is less intrusive and more robust to both covariate and semantically shifted OOD samples?

We tackle this question by exploring and altering a different paradigm of transfer learning called model reprogramming [3]. By leveraging and altering some key components from model reprogramming, we show that it is possible to reprogram a text-image pre-trained model to a down-stream ID task. We also show that, due to the less intrusive nature of reprogramming, our method is better able to maintain pre-training representation, subsequently leading to better OOD Generalization and OOD Detection performances. Formally, we propose reprogrammer, a novel reprogramming approach that leverages two different modalities of model reprogramming to reprogram both the image encoder and the text encoder simultaneously.

We demonstrate the hidden costs and trade-offs of common fine-tuning techniques and reprogrammer in Figure 1. Additionally, to our knowledge, we are the first to take this step in applying model reprogramming techniques to multi-modal text-image encoder models.

2. Methodology

In this section, we first start by introducing the image reprogrammer module before moving to the text reprogrammer module. After which we will present the full reprogrammer transfer learning technique. We also pro-
vide more reprogrammer details in Appendix C.

2.1. Image Reprogrammer

Consider just the CLIP image encoder $f : I \rightarrow \mathbb{R}^{b \times k}$ where $b$ is the input image batch size and $k = 512$ is the CLIP feature size. To apply reprogramming, we leverage the commonly used adversarial program first described by Elsayed et al [10], to which we define as the reprogramming function $\psi$. The reprogramming function $\psi$ is applied to the input image pre-forward pass through the CLIP image encoder $f$. Critically, the reprogramming function $\psi$ is not specific to any singular input image, rather $\psi$ will be consistently applied to all images.

Formally, we define our reprogramming function $\psi$ as:

$$\psi(X) = \mathcal{U}(X) + \tanh(W \odot M)$$

where $\mathcal{U}$ denotes an image up-sampling then zero-padding function, $W \in \mathbb{R}^{d \times d \times 3}$ is the image reprogramming parameters that is to be learned, $d$ is the size of CLIP’s input width and height, $\odot$ denotes the Hadamard product, and $M$ is a binary masking matrix. Specifically, we define the binary masking matrix $M$ as 0 for positions where we wish to implant the original image, and 1 for positions that we choose to reprogram.

2.2. Text Reprogrammer

Now we consider the CLIP text encoder $g : S \rightarrow \mathbb{R}^{b \times k}$ where $b$ is the input text batch size and $k = 512$ is the CLIP feature size. Additionally, we define our text input $s$ as a sequence of tokens $s = \{s_1, ..., s_{|s|}\}$ where $s_i$ is the vocabulary index of the $i^{th}$ token in the vocabulary list $V_s$. To apply reprogramming to a text input, we leverage and alter a version of the adversarial program as first described by Neekhara et al [26].

Formally, we define our text reprogramming function as $\Phi_{\theta, b}$ where $\Phi_{\theta, b}$ is a simple look-up embedding and bias on the tokens $\{s_i\}$ that can be parameterized by the learnable embedding tensor $\theta$ and the bias parameter $b$. Specifically, we define our $\theta \in \mathbb{R}^{|V_s| \times d}$ and $b \in \mathbb{R}^d$ where our default vocabulary size is $|V_s| = 49408$, which is the expected vocabulary size for the CLIP text encoder. Similarly, as with all reprogramming functions, the text reprogramming function is not specific to any singular text input, rather $\Phi_{\theta, b}$ will be consistently applied to all text inputs.

2.3. CLIP Model Reprogrammer

Finally, to train our given image and text reprogramming functions $\psi$ and $\Phi_{\theta, b}$, we define our training objective as:

$$W^*, \theta^*, b^* = \arg \max_{W, \theta, b} \left( \text{sim}(f(\psi W(x)), g(\Phi_{\theta, b}(s))) \right)$$

where $(x, s)$ is an image and caption pair obtained from our training set $D_{tr}$, $f$ and $g$ are the CLIP image and text encoders respectively, sim is the cosine-similarity function, and $W, \theta, b$ are the learnable parameters encapsulating our reprogramming functions $\psi_W, \Phi_{\theta, b}$. In practice, rather than directly optimizing for cosine similarity, we follow closely with the optimization schema of a symmetric cross entropy loss as was implemented in CLIP pre-training [29].

After tuning our reprogrammer parameters $W, \theta, b$ we perform classification during inference time, on an input image $\hat{x}$ with $m$ class labels $C = \{c_1, ..., c_m\}$, similar to that of zero-shot CLIP. Specifically, we make a prediction $y$ through:

$$y = \arg \max_{i} (\text{sim}(f(\psi W^*(\hat{x})), g(\Phi_{\theta^*, b^*}(s_i))))$$

where $s_i$ is the class-wise captions such that $s_i = "a photo of a \{c_i\}"$ and $\psi W^* \text{ and } \Phi_{\theta^*, b^*}$ are our learned reprogramming functions parameterized by $W^*, \theta^*$, and $b^*$.

3. Experiments

In this section, we first describe our experimental setup for OOD Generalization and OOD Detection in Section 3.1, before evaluating our reprogrammer method against other common transfer learning techniques in Section 3.2.
Table 1. CIFAR Detection Results OOD Detection performance comparison between zero-shot (ZS), linear-probing (LP), full fine-tuning (FFT), and reprogrammer (RP) methods using the msp [12] detector. All methods utilize the CLIP B/32 architecture fine-tuned on CIFAR-10 as the in-distribution datasets. A description of all the semantically shifted OOD datasets is provided in Section 3.1. ↑ indicates larger values are better, while ↓ indicates smaller values are better. All values are percentages and bold values are the superior results.

<table>
<thead>
<tr>
<th>$D_{in}$</th>
<th>Method</th>
<th>ISUN FPR95 AUROC</th>
<th>LSUN Resize FPR95 AUROC</th>
<th>Places365 FPR95 AUROC</th>
<th>Textures FPR95 AUROC</th>
<th>Average FPR95 AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>↓ ↑</td>
<td>↓ ↑</td>
<td>↓ ↑</td>
<td>↓ ↑</td>
<td>↓ ↑</td>
</tr>
<tr>
<td>No Tuning</td>
<td>ZS</td>
<td>27.15 95.08</td>
<td>24.61 95.61</td>
<td>15.87 95.12</td>
<td>32.36 92.60</td>
<td>24.95 95.10</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>LP</td>
<td>36.74 94.57</td>
<td>28.38 95.75</td>
<td>24.65 96.73</td>
<td>39.67 92.93</td>
<td>32.36 94.99</td>
</tr>
<tr>
<td></td>
<td>FFT</td>
<td>45.47 92.78</td>
<td>42.95 93.41</td>
<td>40.92 94.06</td>
<td>44.85 92.30</td>
<td>42.89 93.40</td>
</tr>
<tr>
<td></td>
<td>RP</td>
<td>29.58 95.53</td>
<td>25.96 96.08</td>
<td>15.94 97.63</td>
<td>30.13 93.82</td>
<td>25.40 95.77</td>
</tr>
</tbody>
</table>

Table 2. CIFAR Generalization Results OOD Generalization performance comparison between zero-shot (ZS), linear-probing (LP), full fine-tuning (FFT), and reprogrammer (RP) methods with CIFAR-10 as the in-distribution dataset.

<table>
<thead>
<tr>
<th>$D_{in}$</th>
<th>Method</th>
<th>CIFAR-10 Accuracy (%)</th>
<th>CIFAR10.1 Accuracy (%)</th>
<th>STL10 Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>↑ ↑</td>
<td>↑ ↑</td>
<td>↑ ↑</td>
</tr>
<tr>
<td>No Tuning</td>
<td>ZS</td>
<td>89.23</td>
<td>83.30</td>
<td>97.40</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>LP</td>
<td>94.89</td>
<td>96.05</td>
<td>96.34</td>
</tr>
<tr>
<td></td>
<td>FFT</td>
<td>96.24</td>
<td>91.05</td>
<td>96.71</td>
</tr>
<tr>
<td></td>
<td>RP</td>
<td>95.39</td>
<td>91.55</td>
<td>96.71</td>
</tr>
</tbody>
</table>

Additionally, we present further ablations in Section 3.3 and Appendix F, along with more experimental results in Appendix E.

3.1. Experimental Setup

In-distribution dataset We tune our model with CIFAR-10 [19] as the in-distribution (ID) dataset, which is a commonly used ID dataset for both OOD Generalization and OOD Detection experiments. The CIFAR-10 dataset contains labeled (32×32) resolution images covering a range of real-world objects such as horses, cats, and airplanes.

Out-of-distribution Generalization We evaluate the OOD Generalization performance on two standard covariate shifted OOD datasets. Specifically, we evaluate the generalization accuracy on the CIFAR-10.1 [34] and STL10 [6] datasets. Both these datasets contains images derived from semantically matching CIFAR-10 classes.

Out-of-distribution Detection For all compared downstream models, we evaluate using the msp detector against four commonly used OOD benchmarks. More specifically, we evaluate on the iSUN [40], LSUN Resized [42], Places365 [45], and Textures [5] datasets. These OOD datasets span a wide range of objects including fine-grained images, scene images, and textural images. Additionally, these datasets are carefully chosen so that there is no semantic overlapping with respect to the CIFAR-10 dataset.

3.2. Results

Out-of-distribution Generalization We present our main results for OOD Generalization in Table 2, where we compare the OOD Generalization accuracy of our reprogrammer method in comparison to linear-probing and full fine-tuning.

We first observe that full fine-tuning outperforms zero-shot, linear-probing, and reprogrammer on the ID CIFAR-10 task, which is consistent with expectations set by prior works. However, we see that for the OOD Generalization tasks, reprogrammer consistently outperforms both linear-probing and full fine-tuning on each of the OOD Generalization benchmarks, with full fine-tuning in particular performing significantly worse in the STL10 OOD Generalization task. This also matches with intuition from prior work were naive fine-tuning can distort the diverse and beneficial pre-training representations necessary for OOD Generalization tasks [20]. Subsequently, given that reprogrammer encourages minimal alterations to the pre-trained parameters, we can observe that reprogrammer outperforms every other common transfer learning techniques on all of the given OOD generalization tasks.

Out-of-distribution Detection We present our main results for OOD Detection in Table 1. Specifically, we report the OOD Detection performances of our fine-tuned models across four semantically shifted OOD datasets, as well as the average across all four datasets. For a fair comparison we fix the OOD detector, leveraging the commonly used baseline msp detector [12], across all experiments as a way to gauge the level of overconfidence the zero-shot, linear-probed, full fine-tuned, and reprogrammer down-stream models have on semantically shifted OOD samples.

Firstly, we can see that both linear probing and full fine-tuning perform worse when compared with the zero-shot model. This supports our hypothesis that naive fine-tuning methods can degrade the diversely pre-trained representations needed for detecting semantically shifted OOD sam-
samples. Secondly, we can also observe that reprogrammer outperforms every other fine-tuning technique on all of the given OOD Detection tasks. This additionally indicates to us that reprogrammer is better able to achieve this goal of maintaining necessary pre-training representations for a more semantically robust down-stream model.

3.3. Ablation Studies

Visualizing the Reprogrammed Feature Space In this ablation, we provide additional insights showcasing how reprogrammer can better align covariate shifted OOD samples. In Figure 2 we present UMAP visualizations comparing the feature space between the linear-probed and reprogrammer models on covariate shifted OOD images [23]. Observing these visualization, we can see that our reprogrammer model is producing more separable, and more tightly bound, clusters of covariate features. This again supports our intuition that the model reprogramming technique is aligning the OOD samples to the already strongly tuned ID space, therefore enabling reprogrammer to better classify on covariate shifted OOD samples.

4. Conclusion

In this paper, we showcased that maintaining pre-training representation is critical to the robustness of the down-stream model with respect to both covariate and semantically shifted OOD samples. Additionally, we propose an alternative approach for transferring text-image encoder models called reprogrammer that attempts to minimize the distortion to the model’s pre-training representations. Experimental results further showcases the strength of reprogrammer when compared to other common fine-tuning techniques. We hope that our work provides additional insights into the hidden costs of common transfer learning techniques, and inspire future works to leverage reprogramming approaches for transfer learning.

References

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image


[16] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. International Conference on Machine Learning, 2021. 1

[17] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. International Conference on Machine Learning, 2021. 7


