Supplementary Material

A. Background and Related Work

Pre-trained and CLIP-like Models Pre-trained models, learned on a large and diverse dataset, have been a widely popular technique for building strong machine learning models that can be efficiently transferred to down-stream tasks [2, 4, 8, 9, 17, 18, 29, 30, 32, 43, 44]. In this paper, we primarily evaluate with the Contrastive Language–Image Pre-training (CLIP) model [29]. CLIP is a natural language supervised model that is pre-trained on a large 400 million set of image-caption pairs obtained from the web. Specifically, given a set of image-caption pairs $D = \{(X_1, T_1)..., (X_n, T_n)\}$, CLIP-like models train an image-encoder f and a text-encoder g over the dataset D such that the cosine similarity between the features $f(x_k)$ and $h(t_k)$ is maximized with respect to each pair k.

During inference time, given an input image $\hat{x}$ with m classes to choose from $C = \{c_1, ..., c_m\}$, CLIP-like models performs zero-shot classification through

$$\hat{y} = \underset{i}{\text{argmax}}(\text{sim}(f(\hat{x}), g(s_i))) \tag{4}$$

where $f(\hat{x})$ is the image features, $g(s_i)$ is the class-wise captions features where $s_i =$ "a photo of a $\{c_i\}$", and $\text{sim}(f, g)$ is the cosine-similarity function.

Out-of-distribution Generalization To evaluate the OOD Generalization performance of our down-stream models, we tune and compare the accuracy of our methods with respect to two distinct but related datasets D_{in} and D_{out}. The D_{in} set is the in-distribution dataset to which our pre-trained model will be tuned on. The separate OOD dataset D_{out} is a covariate (domain) shifted out-of-distribution dataset, which contains samples that have the same semantic meanings as the in-distribution dataset D_{in} but are presented in different forms. For example, these forms can include sketched, origami, and other versions of the in-distribution classes [11, 13, 31, 37].

The goal of a good transfer learning method, in a OOD Generalization setting, is for there to be high accuracy across both D_{in} and D_{out}. Being able to achieve high accuracy across both datasets is paramount, as an intelligent and robust model should be agnostic to the covariate shifts of a sample.

Out-of-distribution Detection Out-of-distribution detection can be formulated as a binary classification problem, where given some classifier $\tilde{f}$, tasked on the in-distribution dataset D_{in}, our objective is to design a function estimator

$$h(\hat{x}) = \begin{cases} \text{in}, & \text{if } S(\hat{x}) \geq \gamma \\ \text{out}, & \text{if } S(\hat{x}) < \gamma, \end{cases}$$

where $h(\hat{x})$ can determine whether a sample $\hat{x}$ is in-distribution D_{in} or out-of-distribution Q_{out}. Critically, in the OOD Detection setting, our goal is to detect semantically (concept) shifted samples. For example, if the in-distribution encapsulates samples of {"cats", "dogs"} then the goal of our detector h, given a "*car*" sample $\hat{x}$, is to detect that the $\hat{x}$ sample does not belong to the in-distribution set $\hat{x} \notin D_{in}$, or equivalently that the sample is out-of-distribution $\hat{x} \in Q_{out}$.

To evaluate OOD Detection, we apply the commonly used maximum softmax probability (`msp`) detector h_{msp} [12], which measures the confidence of our classifier $\tilde{f}$ towards a given input $\hat{x}$. The goal here for a good transfer learning method, is to produce a down-stream model $\tilde{f}$ which is not overconfident on semantically shifted OOD samples, while maintaining confidence when predicting on ID samples. This goal is again immediately apparent, as we want a safe and robust model to not (overconfidently) find a semantically dissociated OOD sample as indistinguishable from a ID sample [12, 14, 21, 22].

Model Reprogramming *Model reprogramming* is a resource efficient, cross-domain, framework used to re-purpose models for different task specific scenarios [3]. The *model reprogramming* framework takes heavy inspiration from adversarial reprogramming, which was first introduced by Elsayed et al [10]. The goal of *model reprogramming* is to re-use and re-align the data representation, from an existing model, for a separate task without fundamental changes to the model's parameters. *Model reprogramming* methods has been proven to be successful in both white-box and black-box settings [35]. Traditionally, *model reprogramming* methods operate by training an image/audio reprogramming function to optimally transform continuous input data, such that the output of the model can be used to perform some other desired task [10, 41]. Additionally, Neekhara et al [26] presented a reprogramming method for sequence classification models, by utilizing a context-based vocabulary remapping function [25, 26]. To the best of our knowledge, this paper is the first *model reprogramming* method tackling joint text-image pre-trained encoders in a multi-modal setting.

B. Details of Experiments

In this appendix, we first present a detailed description of the chosen OOD Generalization and OOD Detection datasets, that we referenced in Section 3.1, in Appendix B.1. We also include a description of our software and hardware specifications in Appendix B.4 as well as additional hyperparameter settings used to conduct our experiments in Appendix B.5.

B.1. Datasets

We present a detailed list of our OOD Generalization and OOD Detection evaluation datasets, along with a brief description of each dataset.

CIFAR-10 OOD Generalization Benchmarks:

- **CIFAR-10.1** [34] is a collection of over 2,000 test images, sampled from TinyImages, which are designed to be a minute distributional shift from the CIFAR-10 dataset.
- **STL10** [6] is a collection of over 8,000 test images, sampled from ImageNet-1k, that is commonly used in domain adaptation studies. We carefully curate the STL10 dataset to evaluate with only the 9 semantically overlapping classes, choosing to omit the semantically different "monkey" class.

ImageNet-1k OOD Generalization Benchmarks:

- **ImageNetV2** [31] is a collection of 10,000 test images with approximately 10 samples per class. The dataset was sampled utilizing the same semantic labels as defined in ImageNet-1k and obtained independently from any previous ImageNet models.
- **ImageNet-A** [13] is a collection of 7,500 naturally adversarial and challenging images that are sampled based on 200 semantically overlapping ImageNet-1k classes.
- **ImageNet-R** [11] is a collection of over 30,000 test images, based on 200 semantically overlapping ImageNet-1k classes, that contain images of art, cartoon, graffiti, embroidery, origami, toy, sculpture, sketch, tattoo, and other rendition of the ImageNet-1k classes.
- **ImageNet-Sketch** [37] is a collection of over 50,000 test images based on all 1000 of the ImageNet-1k classes with approximately 50 images per class. Each image is a black and white sketch variant of the ImageNet-1k class.

CIFAR-10 OOD Detection Benchmarks:

- **iSUN** [40] is a collection of over 8,925 natural scene images sampled from the SUN dataset. We include the full set of iSUN images when conducting OOD Detection evaluations.
- **LSUN Resized** [42] is a collection of 10,000 testing images, sampled from the LSUN dataset, spanning across 10 different scenes with images down-sampled to the size of (32×32). We include the full set of LSUN Resized images when conducting OOD Detection evaluations.
- **Places365** [45] contains large-scale photographs of scenes with 365 scene categories. There are 900 images per category in the test set and we again include the full test set for OOD Detection evaluations.
- **Textures** [5], or Describable Textures Dataset, is a collection of 5,640 real-world texture images under 47 categories. We include the entire set of 5640 images for OOD Detection evaluations.

ImageNet-1k OOD Detection Benchmarks:

- **iNaturalist** [36] is a collection of 859,000 plant and animal images spanning over 5,000 different species. Each image is resized to have a max dimension of 800 pixels and we evaluate on 10,000 images randomly sampled from 110 classes that are carefully chosen to be semantically disjoint from the ImageNet-1k dataset.
- **SUN** [39] is a collection of over 130,000 images of scenes spanning 397 categories. We evaluate on 10,000 randomly sampled images from 50 classes that are semantically disjoint from ImageNet-1k classes, as SUN and ImageNet-1k have overlapping semantic concepts.
- **Places** [45] is a collection of scene images with similar semantic coverage as SUN. We use a subset of 10,000 images across 50 classes that are semantically disjoint from the ImageNet-1k dataset.
- **Textures** [5], or Describable Textures Dataset, is a collection of 5,640 real-world texture images under 47 categories. We again include the entire set of 5640 images for OOD Detection evaluations.

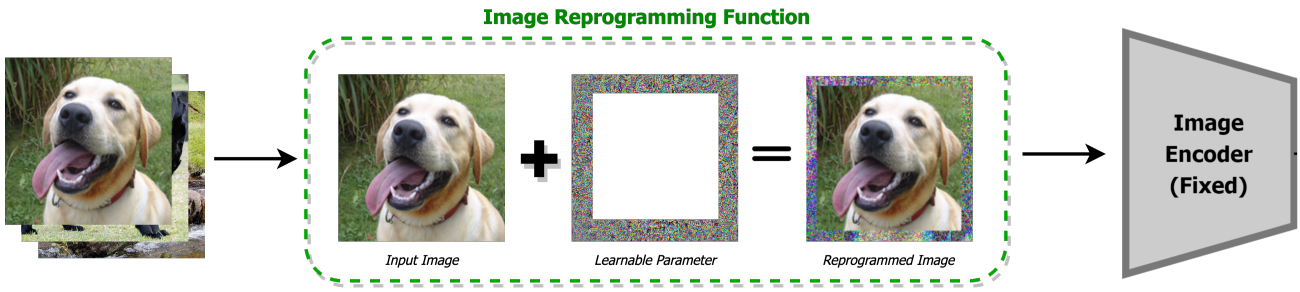


Figure 3. Visual diagram of the image reprogrammer module based on the commonly used adversarial program presented by Elsayed et al [10]. An input image is resized and padded before an additional learnable edge perturbation is added to the input image. The resulting perturbed image is then passed through the fixed image encoder for a model forward pass.

B.2. Evaluated Methods

We compare our `reprogrammer` (RP) model against a zero-shot (ZS), linear-probed (LP), and full fine-tuned (FFT) model. Each of these models are learned through their respective transfer learning techniques and are commonly used in CLIP-based OOD evaluations. Specifically, zero-shot refers to applying the CLIP pre-trained model directly to the designated down-stream task without any alterations to the CLIP model. In contrast, we obtain a linear-probed model by directly optimizing a linear regression classifier on the frozen features taken from the penultimate layer of the CLIP image encoder. Additionally, to obtain a full fine-tuned model we tune all parameters in the image encoder and classification head to fit to the in-distribution dataset.

B.3. Evaluation metrics

For OOD Generalization, we measure all methods across the specified covariate shifted OOD datasets with accuracy as the evaluation metric. For OOD Detection, we measure the performance of all methods across the semantically shifted datasets with the false positive rate, when true positive rate of ID samples is 95% (FPR95), and the area under the receiver operating characteristic curve (AUROC) as the evaluation metrics.

B.4. Software and Hardware

Software We conducted all experiments with Python 3.8.12 and PyTorch 1.11.0.

Hardware All experiments were conduced on NVIDIA GeForce RTX 2080Ti.

B.5. Hyperparameter Settings

For all fine-tuning training, we initialize with the pre-trained CLIP B/32 model and sweep over 3 learning rates using a cosine learning rate scheduler. For linear-probing, we directly optimize a linear regression classifier on the frozen features taken from the penultimate layer of the CLIP image encoder and sweep over (0.005, 0.002, 0.001) learning rates for 5 epochs. While for full-fine tuning, we initialize the classification head with the text encoder features derived from the class-wise captions as specified by Wortsman [38], before sweeping over (0.00001, 0.00003, 0.0001) learning rates for 5 epochs and optimizing all parameters in the image encoder and classification head. Lastly, for `reprogrammer`, we randomly initialize both the image and text reprogramming functions and sweep over (0.0005, 0.001, 0.005) learning rates for 5 epochs. Additionally, we set the image `reprogrammer` up-sampling to be 160×160 and 224×224 pixels with padding size of 64 and 32 when tuning on CIFAR-10 and ImageNet-1k respectively. For all experiments, we specified the batch size to be 128 with a warmup length of 500. Any additional hyperparameter settings can be found in the source code we provided. In addition, all images with resolution higher than 128×128, were down-sampled and cropped to 128×128. We did this due to CLIP's input size limitations, the necessity for paddable pixels, and for a fair comparison over all datasets given the down-sampling information loss. We showcase additional experiments with varying degrees of down-sampling ranging from no down-sampling to heavy down-sampling in Appendix H.

B.6. Computational Complexity

One of the key benefit, to using *model reprogramming* techniques, is the minimal resource and data requirement associated with the reprogramming functions [35]. Specifically, the computational overhead of adding `reprogrammer` is minimal,

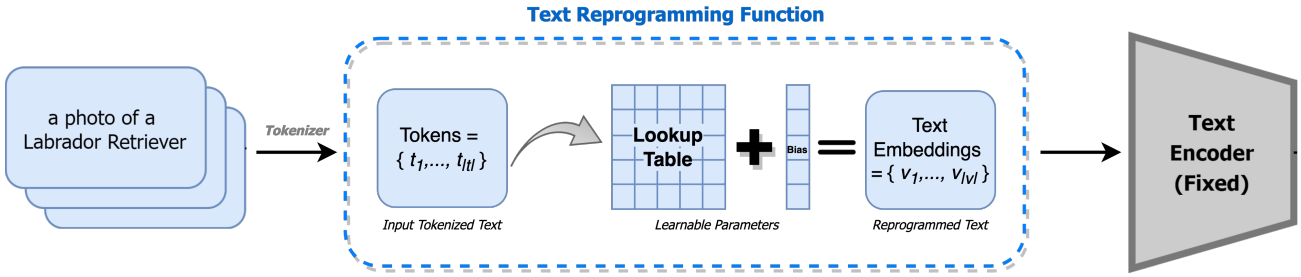


Figure 4. Visual diagram of the text reprogrammer module altered from the the adversarial program as first described by Neekhara et al [26]. An input caption is tokenized before an lookup table embedding and subsequent bias is applied. The resulting embeddings is then passed through the fixed text encoder for a model forward pass.

as the image reprogrammer module can be decomposed into a simple masking function with a matrix addition, and the text reprogrammer is a simply lookup with a vector addition. Subsequently, there is negligible overhead associated with the reprogramming function during both inference and training time. Similarly, the memory complexity of maintaining the reprogramming functions is also minimal as you are only required to maintain matrices proportional to the fixed image and vocabulary size specified by the text-image encoder model.

C. Details of Methodology

In this appendix, we present additional visualizations to help explain the individual components of `reprogrammer` in detail. We first present additional image reprogrammer details and visualizations in Appendix C.1, before moving on to detailing the text reprogrammer component in Appendix C.2. Finally, we present a visualization showcasing the full training schema of `reprogrammer` in Appendix C.3.

C.1. Image Reprogrammer

We present a visual diagram of our image reprogrammer in Figure 3. Additionally, we note that the image reprogrammer can up-sample images to the input size of the pre-trained model. However due to restrictions in the current open sourced pre-trained CLIP models, our image reprogrammer up-sampling is limited to being $3\times224\times224$ dimensions or less. Additionally, as apart of the reprogramming function ψ, the size of the up-sampling/padding function $\mathcal{U}$ and binary masking matrix M are tunable hyperparameter. We present a more detailed look at the hyperparameter setting for the image reprogramming function ψ in Appendix F.

C.2. Text Reprogrammer

Similarly, we present a visual diagram of our text reprogrammer in Figure 4. Additionally, we generate class-wise captions following closely with the experiments presented by Radford et al [29]. Specifically, we set $s_i =$ "a photo of a $\{c_i\}$" where c_i is the given sample class label. As an example, our text reprogrammer follows the procedures where, given a text Labrador Retriever label, our text reprogrammer first tokenizes the string $s =$ "a photo of a Labrador Retriever" into tokens t_s. Subsequently, the tokens t_s are the passed into the Φ_θ function to embed the tokens into a vector v'_s. Then we apply a bias parameter b to the given vector v'_s in the form of $v_s = v'_s + b$, before finally passing the vector v_s through the CLIP text encoder g to get the reprogrammed text features.

C.3. Training Reprogrammer

Finally, we present a visualization detailing the training of the `reprogrammer` parameters in Figure 5. We also note that, in comparison to linear probing and full fine-tuning, `reprogrammer` maintains minimal additional training time complexity. As the image reprogramming function is a

Additionally, through `reprogrammer`'s two-sided reprogramming, we can also forgo the process of hand curating a label-map *(a required model reprogramming technique where the model outputs are remapped to better fit the specified task)*. Instead, by levering the dual reprogramming functions and the image-caption pairs, we can embed and tune the label-map as apart of our `reprogrammer`.

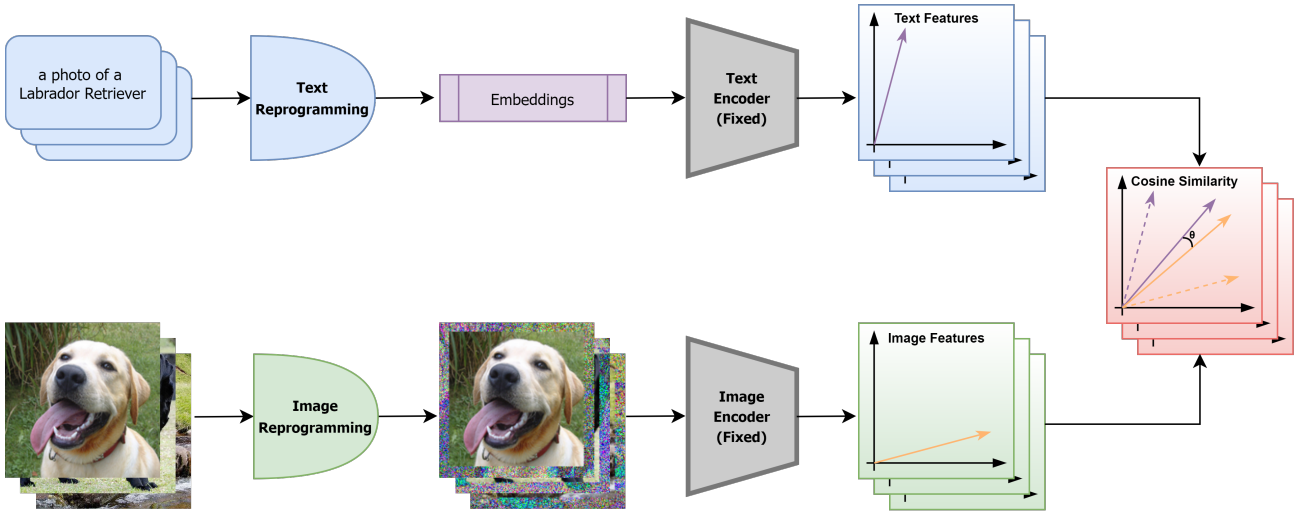


Figure 5. Visual diagram of the `reprogrammer` methodology based on the CLIP joint image and text encoder training. During reprogramming, an image and caption pair are each independently passed through their respective reprogramming function before being passed through the CLIP image and text encoders. A loss is then computed based on the cosine similarity of the two reprogrammed features. Then we backpropagate and optimize each respective `reprogrammer` parameters.

D. Limitations

In this appendix, we discuss some limitations of our `reprogrammer` technique. Specifically, we address concerns with using differing architectures in Appendix D.1 before moving on to exploring some inherent limitation with text-image encoder models in Appendix D.2.

D.1. Differing Architectures

Within our evaluations, we leverage CLIP as the pre-trained model to which we apply our `reprogrammer` method. Subsequently, a natural question arises asking how effective would our `reprogrammer` method be when applied to other similar CLIP-like models with differing encoder architectures and training datasets such as BASIC [28]. But, due in large part to our limited computational resources, as well as a lack of available open-source pre-trained model parameters, we are unable to train or test with these comparable models. However, critically it is important to note that our `reprogrammer` methodology is not inherently limited in anyway to just the open-source CLIP models. Additionally, BASIC primarily differ from CLIP only in its scale, as BASIC can be interpreted as CLIP but with larger capacity transformer architectures alongside a larger training dataset. Therefore, as BASIC is fundamentally similar to CLIP, we hypothesize that `reprogrammer` should show similar effectiveness when applied to BASIC. Subsequently, we leave this question open for future exploration and encourage researchers, with more readily available resources, to experiment with the `reprogrammer` methodology.

D.2. Text-Image Encoders

Our `reprogrammer` method seeks to reprogram both the image and text encoders in a CLIP model. Due in part to the necessity for there to be paired image and text encoders, our `reprogrammer` method is limited to these joint text-image encoder models. However, diverging from our `reprogrammer` method, we can leverage components present in `reprogrammer` in combination with other transfer learning techniques. Specifically, Kumar [20] showed how one can mitigate full fine-tuning distortions by simply tuning the classification head first before fully fine-tuning the model. Similarly, the image reprogramming function can be combined with linear-probing or full fine-tuning to create another set of transfer learning techniques. These `reprogrammer` plus methods could also have similar out-of-distribution benefits that we observed here in our paper, and could also potentially be applied to other non-CLIP pre-trained models. We leave this area open for future exploration.

D.3. Output Space Dimensionality

Another potential limitation is the effectiveness of `reprogrammer` with respect to higher output dimensionality tasks. Specifically, we have observed that *model reprogramming* techniques generally work significantly better on low output di-

Table 3. **ImageNet Generalization Results** OOD Generalization performance comparison between zero-shot, linear-probing, full fine-tuning, and `reprogrammer` methods. All methods utilize the CLIP B/32 architecture fine-tuned on **ImageNet-1k** as the in-distribution dataset. The description of the four covariate shifted OOD datasets is provided in Appendix B.1. ↑ indicates larger values are better, while ↓ indicates smaller values are better. All values are percentages and **bold** values are the superior results.

D_{in}	Method	ImageNet-1k Accuracy ↑	ImageNetV2 Accuracy ↑	ImageNet-A Accuracy ↑	ImageNet-R Accuracy ↑	ImageNet-S Accuracy ↑
No Tuning	Zero-shot (ZS)	59.44	52.79	11.82	43.48	38.61
ImageNet	Linear-probing (LP)	72.43	**61.35**	10.71	41.58	38.19
	Full Fine-tuning (FFT)	**73.14**	60.98	6.41	32.71	32.83
	Reprogrammer (RP)	72.10	61.28	**12.58**	**44.30**	**39.40**

mensionality ID tasks like the 10-way classification in CIFAR-10. However, with higher output dimensionality tasks like the 1000-way classification in ImageNet-1k, *model reprogramming* techniques typically needs more rigorous tuning and is generally less able to strictly outperform other transfer learning techniques. This limitation subsequently is present in our `reprogrammer` method as, although `reprogrammer` still shows improvements in the provided ImageNet benchmarks, there are less consistent improvements in specific OOD tasks. An intuition approach of tackling this issue would be to replace our traditional and simple reprogramming functions with stronger reprogramming functions. However, additional experimentation needs to be conducted and we leave this question open as a goal for future exploration.

E. ImageNet Experiments

E.1. Datasets

In-distribution Dataset For this series of experiments, we tune our model with the **ImageNet-1k** [7] dataset as the in-distribution (ID) dataset. This dataset is another commonly used ID dataset for OOD Generalization and OOD Detection experiments. Specifically, the ImagetNet-1k dataset contains over 1.2 million training images spanning 1000 different real-world objects such as species of dogs, chairs, and boats.

OOD Generalization Datasets For the given model tuned on ImageNet-1k, we evaluate the OOD Generalization performance on four frequently used OOD Generalization benchmarks. Specifically, we evaluate on **ImageNetV2** [31], **ImageNet-R** [11], **ImageNet-A** [13], and **ImageNet-Sketch** [37]. All these datasets contains images derived from the same semantic label as in ImageNet-1k. For example, these datasets may include a sketched version of a Labrador Retriever, a cartoon drawing of a strawberry, or a photo of a toy duck (see Appendix B.1 for more detail).

OOD Detection Datasets For the model tuned with ImageNet-1k, we use the large-scale ImageNet OOD detection benchmark proposed by Huang et al [15]. Specifically, we evaluate on four OOD datasets which are subsets from the **iNaturalist** [36], **SUN** [39], **Places** [45], and **Textures** [5] datasets. These datasets are carefully curated so that there is no semantic overlap with respect to the ImageNet-1k dataset (see Appendix B.1 for more detail).

E.2. Experimental Results

Out-of-distribution Generalization We present the ImageNet results for OOD Generalization in Table 3. Specifically, in Table 3 we report the accuracy of our `reprogrammer` method in comparison with zero-shot, linear-probing, and full fine-tuning when tuned on the ImageNet-1k dataset. First, we observe that, similar to our CIFAR-10 experiments, full fine-tuning outperforms zero-shot, linear-probing, and `reprogrammer` on the ID ImageNet task. This is again consistent with expectations set by prior works. We also observe that for our OOD datasets, `reprogrammer` consistently outperforms both linear-probing and full fine-tuning on all of our ImageNet OOD Generalization benchmarks. Additionally, we can see that full fine-tuning in particular, significantly underperforms on OOD Generalization tasks. This matches with our intuition, that (1) fine-tuning can degrade the good pre-training representations needed for OOD Generalization and (2) `reprogrammer` can help covariate-shifted distributions better align with the in-distribution sample space resulting in better OOD Generalization. Furthermore, we conduct an additional ablation study visualizing the better OOD Generalization alignments in Appendix F.

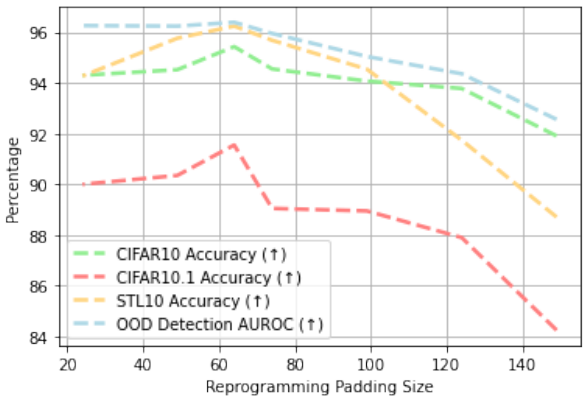
Table 4. **ImageNet OOD Detection Results** OOD Detection performance comparison between zero-shot, linear-probing, full fine-tuning, and `reprogrammer` methods using the `msp` [12] detector. All methods utilize the CLIP B/32 architecture fine-tuned on **Image-1k** as the in-distribution datasets. A description of all the semantically shifted OOD datasets is provided in Appendix B.1. ↑ indicates larger values are better, while ↓ indicates smaller values are better. All values are percentages and **bold** numbers are the superior results.

D_{in}	Method	iNaturalist		SUN		Places		Textures		Average	
		FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
No Tuning	ZS	53.96	85.15	**64.89**	**81.26**	**65.76**	**79.30**	**70.05**	77.03	**63.67**	**80.69**
ImageNet	LP	**51.15**	**88.25**	78.68	74.58	76.42	75.15	70.25	**78.71**	69.12	79.17
	FFT	71.94	81.37	80.29	74.01	79.97	74.54	78.28	74.80	77.62	76.18
	RP	56.85	85.97	75.68	74.99	74.80	74.84	70.51	77.43	69.46	78.31

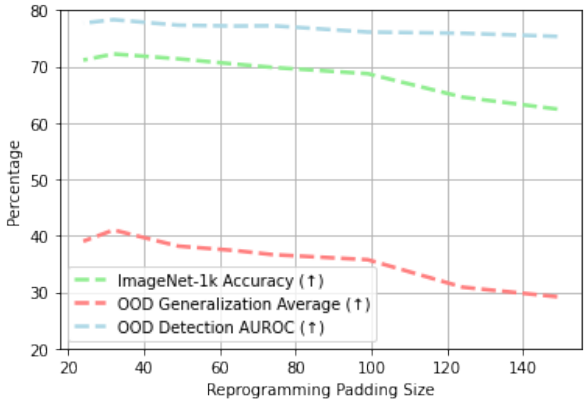
Out-of-distribution Detection We present our ImageNet results for OOD Detection in Table 4. Specifically, in Table 1, we report the OOD Detection performance on the four semantically shifted ImageNet-1k OOD datasets as well as their average. Again, for a fair comparison, we use the same commonly used baseline `msp` detector across all experiments as a way to gauge the level of overconfidence the zero-shot, linear-probed, full fine-tuned, and `reprogrammer` model has on semantically shifted OOD samples. We can see that the zero-shot model outperforms all of our fine-tuned models, with full fine-tuning performing the worse by in large. This again is is consistent with our hypothesis that transfer learning techniques can degrade the necessary diverse pre-training representations needed for OOD Detection. Additionally, we hypothesize that by fitting to an ID task we are implicitly restricting and subsequently degrading the OOD Detection performance of the diversely pre-trained model. This also potentially indicates that transfer learning techniques should be solemnly used when attempting to perform OOD Detection tasks. Again, we leave this question open for future exploration.

F. Ablation Studies

Reprogrammer Padding Size In this ablation, we evaluate the effectiveness of our `reprogrammer` model as we adjust the image reprogrammer padding size. The image reprogrammer padding size refers to the set of hyperparameters, in our image reprogramming function ψ, that controls the amount of border and padding we choose to perturbed. Subsequently, the greater the reprogramming padding size the more the image will be subjected to the perturbations set by the image reprogrammer. We show the effects of the reprogramming padding size as we increase the allowed border pixel from 30 to 140 in Figure 6. Specifically, we show the effects of the padding size across both our CIFAR-10 benchmarks (Figure 6a) and ImageNet-1k benchmarks (Figure 6b). Comparing our ablation results, we can see that generally the optimal range of padding sizes for our CIFAR-10 reprogrammed model is larger than that of our ImageNet-1k reprogrammed model. We hypothesize that this is due to the lower resolution images present within the CIFAR-10 benchmarks, forcing our `reprogrammer` to be more aggressive when perturbing the images in order to offset the lower resolution samples.



(a) Reprogrammer Padding Size (CIFAR Benchmarks)



(b) Reprogrammer Padding Size (ImageNet Benchmarks)

Figure 6. Ablation on the effectiveness of our `reprogrammer` method as we adjust the image reprogramming padding size. The higher the padding size indicts the more the input image is subjected to the reprogramming function.

G. Societal Impact

The goal of our project is to improve the safety and robustness of transfer learning techniques on large pre-trained machine learning models. We believe that improving these aspects can benefit a wide range of societal fields. Given that many modern real-world models heavily rely on classification, these safety and robustness issues are critical to any system ranging from consumer and business applications to autonomous vehicles and medical imaging. We hope that by showcasing this work we can provide researchers with an additional tool when tackling these challenging problems. Although we do not anticipate any negative repercussions of our work, we hope to continue to observe and build on top of this method in the future.

H. Down-sampling Experiments

In this appendix, we present our down-sampling experiments showcasing that our method isn't limited by the down-sampling step we implemented within our experiments. Specifically, in Table 5 we show the OOD Generalization performance of `reprogrammer` and in Table 6 we show the OOD Detection performance of `reprogrammer` as we apply different degrees of down-sampling to training and testing datasets.

Table 5. **Down-sampling OOD Generalization Results** OOD Generalization performance comparison between differing down-sampling severity. All methods utilize the CLIP B/32 architecture and were fine-tuned on the **ImageNet-1k** dataset down-sampled to the specified resolution. Similarly, the evaluation was completed using, if available, the validation dataset down-sampled to the specified resolution. A description of the four covariate shifted OOD test datasets is provided in Appendix B.1. ↑ indicates larger values are better, while ↓ indicates smaller values are better. All values are percentages and **bold** numbers are superior **fine-tuning** results.

Down-sampling Size	Method	ImageNet-1k Accuracy ↑	ImageNetV2 Accuracy ↑	ImageNet-A Accuracy ↑	ImageNet-R Accuracy ↑	ImageNet-S Accuracy ↑
64×64	Zero-shot	50.632	44.22	6.12	34.547	28.556
	Linear-probing	65.322	53.61	5.627	36.19	29.354
	Full Fine-tuning	**70.33**	**58.33**	5.28	30.55	29.018
	Reprogrammer	65.814	54.71	**7.08**	**36.923**	**30.653**
96×96	Zero-shot	57.284	50.59	9.813	41.043	35.587
	Linear-probing	70.464	58.89	9.16	39.973	35.517
	Full Fine-tuning	**72.228**	**60.68**	5.60	31.9	32.249
	Reprogrammer	70.244	59.27	**11.00**	**41.777**	**36.906**
128×128	Zero-shot	59.44	52.79	11.82	43.48	38.61
	Linear-probing	72.43	61.35	10.71	41.58	38.19
	Full Fine-tuning	**73.14**	60.98	6.41	32.71	32.83
	Reprogrammer	72.10	**61.28**	**12.58**	**44.30**	**39.40**
160×160	Zero-shot	60.14	53.46	12.893	44.36	39.932
	Linear-probing	73.142	61.86	11.787	42.167	39.27
	Full Fine-tuning	**73.39**	61.59	6.227	32.877	33.331
	Reprogrammer	72.934	**61.91**	**14.07**	**44.43**	**40.41**
192×192	Zero-shot	60.796	53.84	13.8	44.86	40.294
	Linear-probing	73.486	62.03	11.907	42.093	39.425
	Full Fine-tuning	**73.764**	61.86	6.427	32.137	32.135
	Reprogrammer	73.08	**62.36**	**14.73**	**44.57**	**40.73**
224×224	Zero-shot	61.896	54.71	15.267	46.713	40.83
	Linear-probing	74.882	62.45	12.6	42.217	39.944
	Full Fine-tuning	**75.071**	62.03	6.387	32.48	33.469
	Reprogrammer	74.118	**62.65**	**15.32**	**45.09**	**40.86**

Table 6. **Down-sampling OOD Detection Results** OOD Detection performance comparison between differing down-sampling severity. All methods utilize the CLIP B/32 architecture fine-tuned on the **Image-1k** dataset down-sampled to the specified resolution. Similarly, all semantically shifted OOD datasets were also down-sampled to the specified resolution and a description of each OOD datasets is provided in Appendix B.1. ↑ indicates larger values are better, while ↓ indicates smaller values are better. All values are percentages and **bold** numbers are the superior **fine-tuning** results.

Down-sampling Size	Method	iNaturalist		SUN		Places		Textures		Average	
		FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑
64×64	ZS	63.57	81.3	75.1	77.8	74.34	76.14	73.67	75.82	71.67	77.77
	LP	65.44	**84.62**	84.61	70.23	83.48	70.79	**76.35**	**76.51**	77.47	75.54
	FFT	74.82	79.44	81.29	73	80.55	**73.45**	80.96	72.95	79.4	74.71
	RP	**65.42**	83.68	**79.96**	**72.05**	**80.13**	71.86	77.78	74.72	**75.82**	**75.58**
96×96	ZS	57.5	83.7	67.13	80.53	68	78.77	71.13	76.64	65.94	79.91
	LP	**54.21**	**87.46**	80.32	73.77	77.94	**74.18**	**72.36**	**78.28**	71.21	**78.42**
	FFT	73.33	80.55	80.89	73.43	80.03	74	80.09	73.75	78.58	75.43
	RP	56.76	85.62	**77.19**	**73.86**	**76.46**	73.91	73.07	76.78	**70.87**	77.54
128×128	ZS	53.96	85.15	64.89	81.26	65.76	79.30	70.05	77.03	63.67	80.69
	LP	**51.15**	**88.25**	78.68	74.58	76.42	**75.15**	**70.25**	**78.71**	**69.12**	**79.17**
	FFT	71.94	81.37	80.29	74.01	79.97	74.54	78.28	74.80	77.62	76.18
	RP	56.85	85.97	**75.68**	**74.99**	**74.80**	74.84	70.51	77.43	69.46	78.31
160×160	ZS	52.98	85.38	63.57	81.5	64.36	79.63	69.34	77.23	62.56	80.93
	LP	**50.42**	**88.37**	77.53	74.94	75.17	75.65	68.55	**79.01**	67.92	**79.49**
	FFT	71.83	81.15	81.55	73.26	80.35	74.2	78.79	74.43	78.13	75.76
	RP	56.25	85.92	**74.55**	**75.69**	**72.37**	**76.27**	**67.98**	77.93	**67.79**	78.95
192×192	ZS	53.57	85.22	63.75	81.37	63.04	80.16	68.99	77.44	62.34	81.05
	LP	**50.28**	**88.36**	78.02	74.89	74.62	76.25	**69.04**	**78.96**	67.99	**79.62**
	FFT	71.95	81.09	81.43	73.56	81.11	74.12	79.2	74.6	78.42	75.84
	RP	55.77	85.99	**74.48**	**75.31**	**71.30**	**76.79**	69.24	77.8	**67.70**	78.97
224×224	ZS	53.75	85.58	62.89	81.65	63.82	80.13	68.19	77.67	62.16	81.26
	LP	**50.88**	**88.18**	79.14	74.92	75.9	76.02	**67.73**	**79.35**	68.41	**79.62**
	FFT	72.14	80.89	80.98	74.06	80.69	74.58	79.04	74.98	78.21	76.13
	RP	55.56	85.82	**73.89**	**76.25**	**70.32**	**77.63**	68.05	78.28	**66.95**	79.5