A Additional Results

A.1 Ablation Study for α and β

Figure 10 shows the results of *PII* when varying the values of α and β, which determine the intervals from which the ColorShift constants are randomly drawn. Based on this and similar experiments, we permanently fix these parameters to $\alpha = \beta = 1.0$ for all other *PII* experiments, and find that these values indeed transfer well to other models.

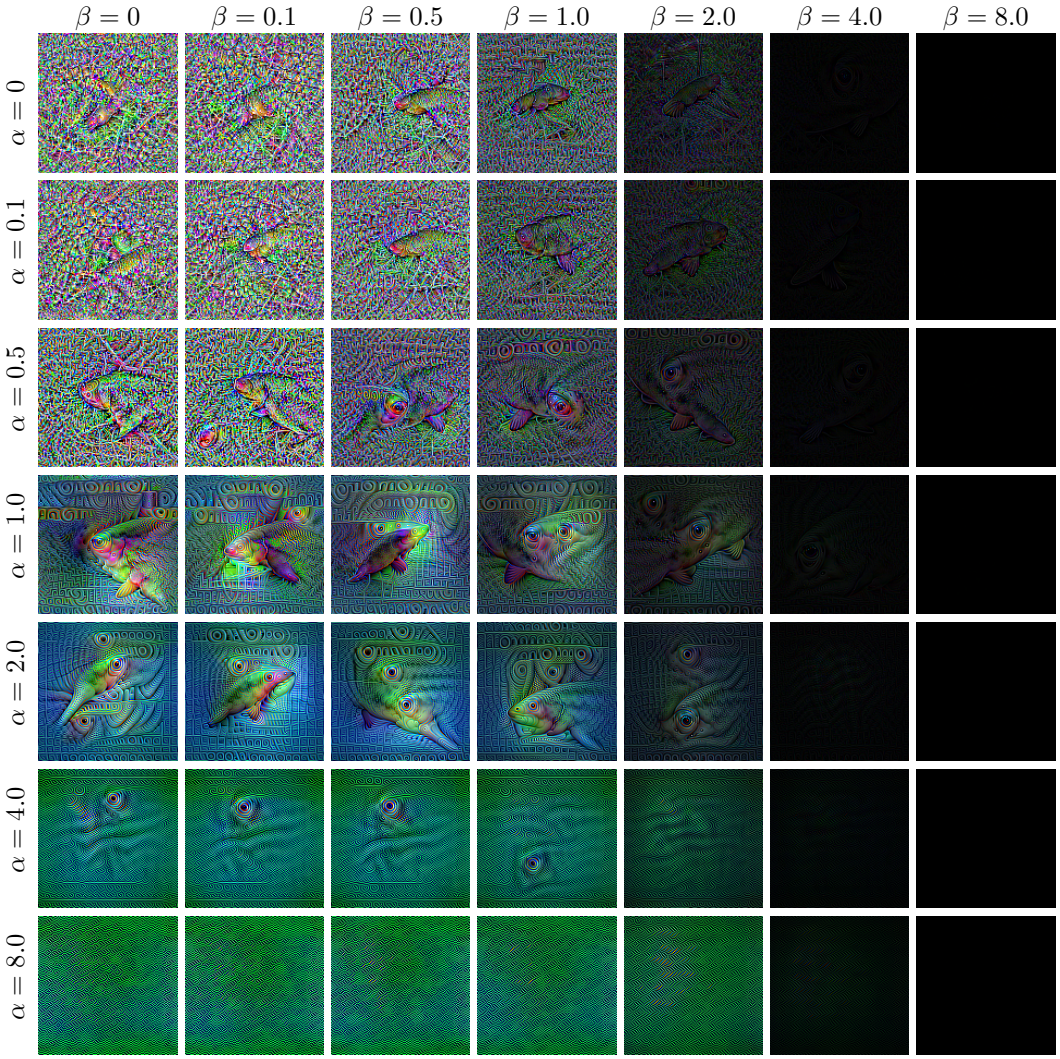


Figure 10: Effect of α and β on the quality of the images generated by *PII* from a naturally-trained ResNet-50 for the Tench class.

A.2 Insensitivity to TV regularization

Figure 11 shows additional results on the effect of ColorShift on the sensitivity to the weight of TV regularization when inverting a robust model, complementing Figure 4. As in the earlier figure, we observe that certain values of λ_{TV} may produce noisy or blurred images when not using ColorShift, whereas the ColorShift results are quite stable.

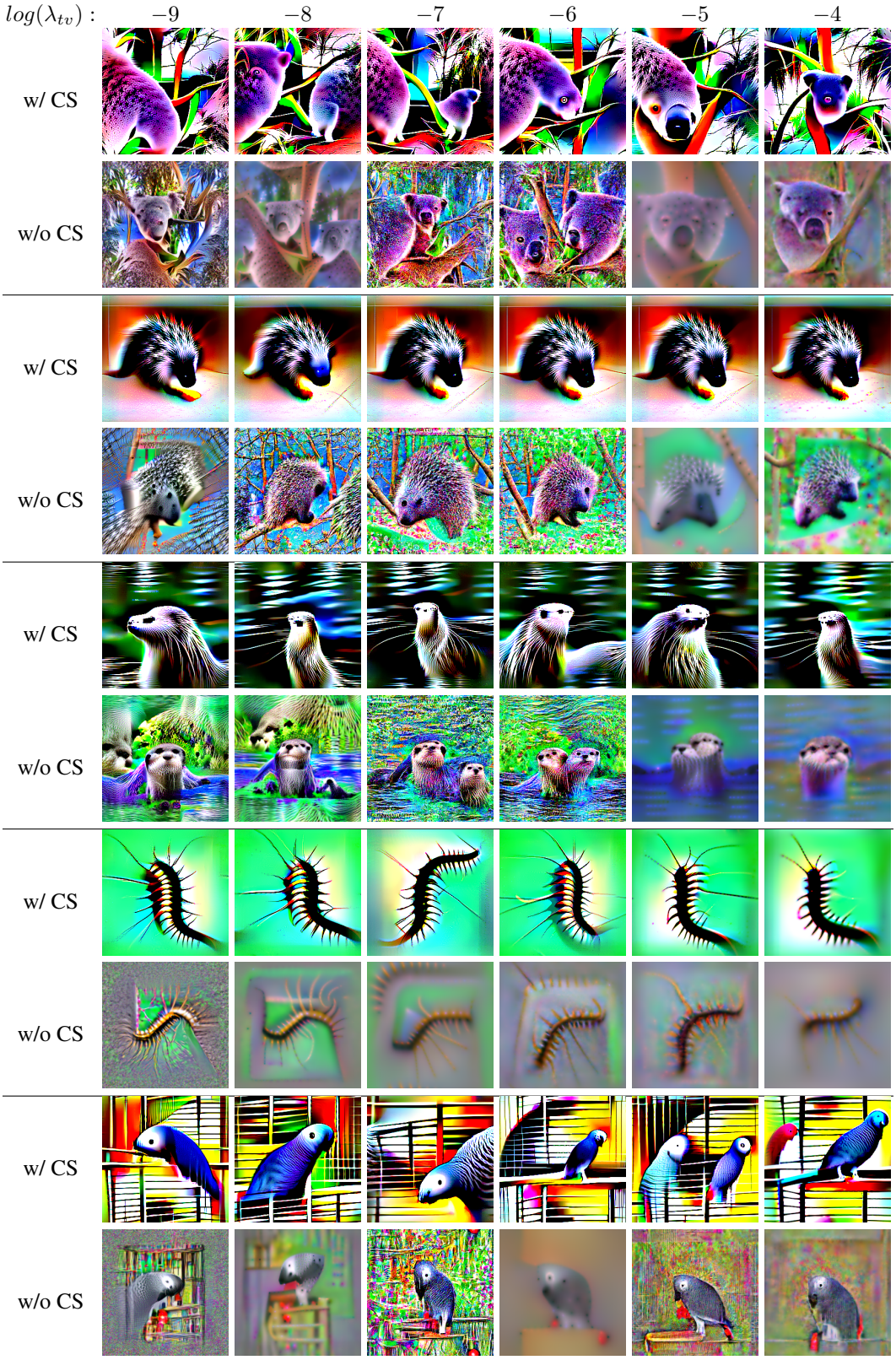


Figure 11: Effect of TV with and without ColorShift. With ColorShift it is clear that there is no need for hyper-parameter tuning for parameters such as TV. Images from the robust ResNet-50.

A.3 Effect of Centering

Here are more examples on the effect of *centering*. Figures 12, and 13 show the effect of *centering* on inverting a robust, and natural model, respectively.

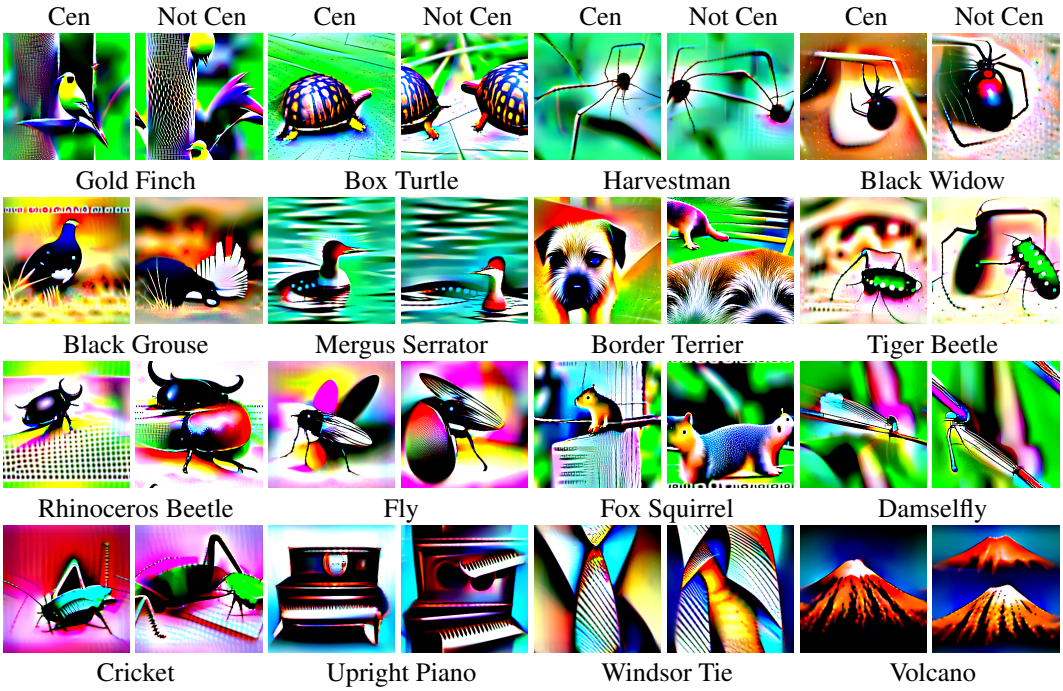


Figure 12: Effect of using centering vs not using centering for a robust ResNet-50.

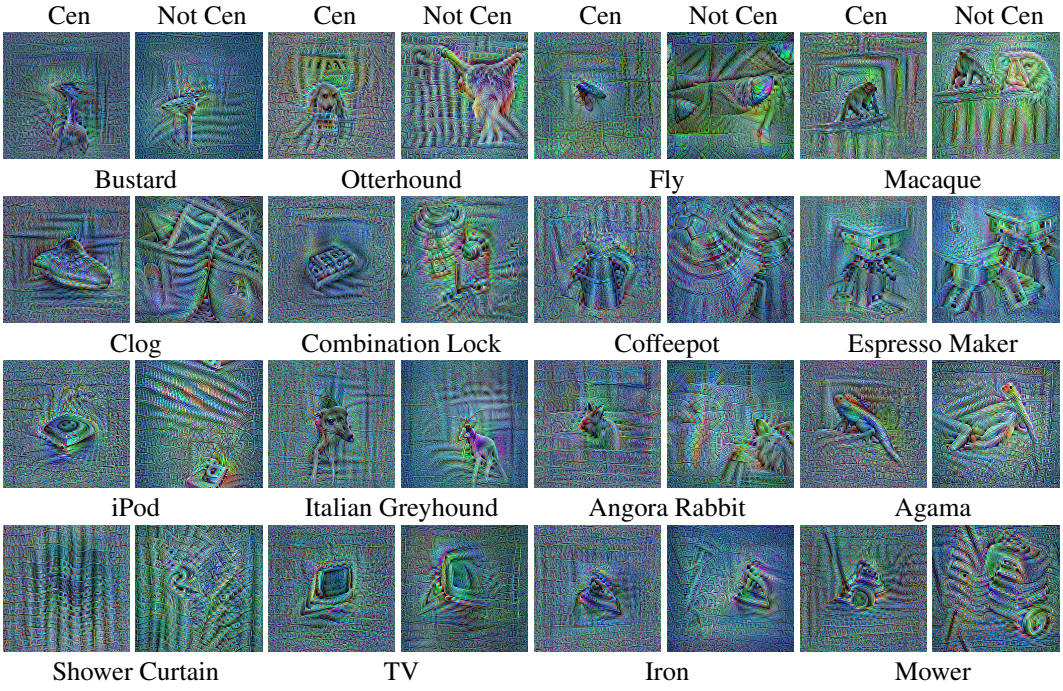


Figure 13: Effect of using centering vs not using centering for a naturally-trained ResNet-50.

A.4 Effect of Ensembling Size

Figure 14 gives additional results to those in figure 5 for the effect of ensembling size on inversion.

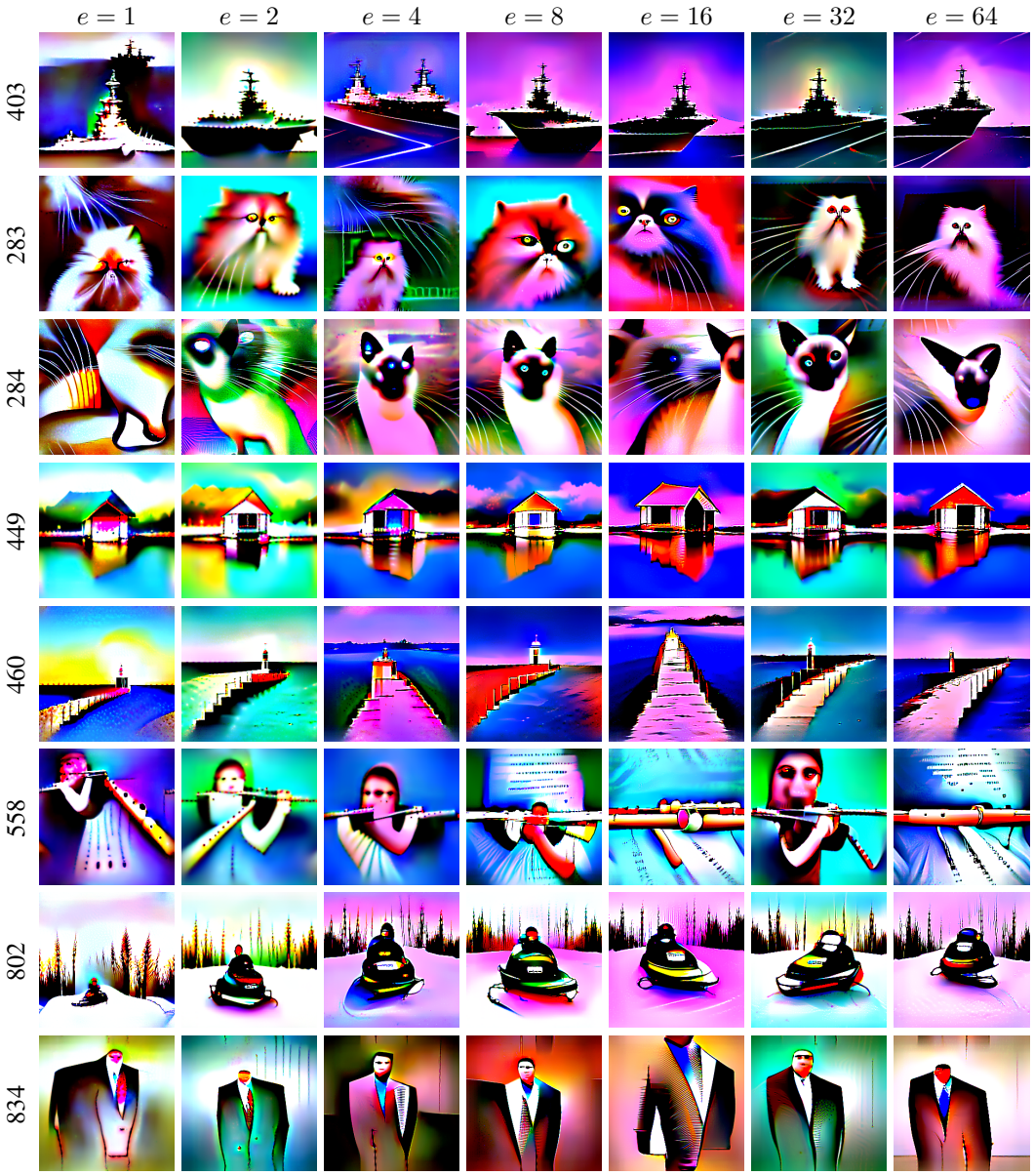


Figure 14: Effect of ensemble size when inverting a robust ResNet-50. Even small values of e give reasonably good results, but increasing e tends to give slight improvement.

A.5 EFFECT OF USING OTHER AUGMENTATIONS

We used 4 random augmentations other than ColorShift to make comparisons. We used augmentations used in Chen et al. (2020) with modifications. We use PyTorch (Paszke et al., 2019) notion to describe this part. We used *RandomHorizontalFlip* with 0.5 probability. We used *RandomResizedCrop* with scale [0.7, 1.], and ratio [0.75, and 1.33]. With applied *ColorJitter* with 0.8 probability, and brightness, contrast, saturation, and hue of (0.4, 0.4, 0.4, 0.1), respectively. We used *RandomGrayscale* with 0.2 probability. For this experiment, we do apply data normalization before feeding the input to the network. This is different than the regular experiment setting that we use for the robust model that we use as explained in D. The reason is that not having data normalization is similar to using ColorShift (it changes the data distribution which the model expects as an input).

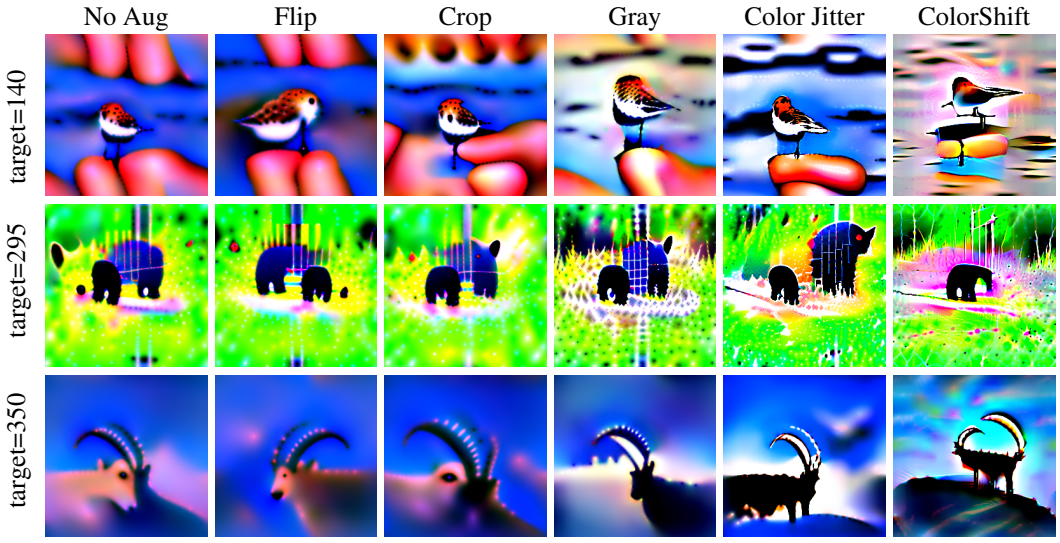


Figure 15: Effect of various alternative augmentations on inverting a robust ResNet-50 model.

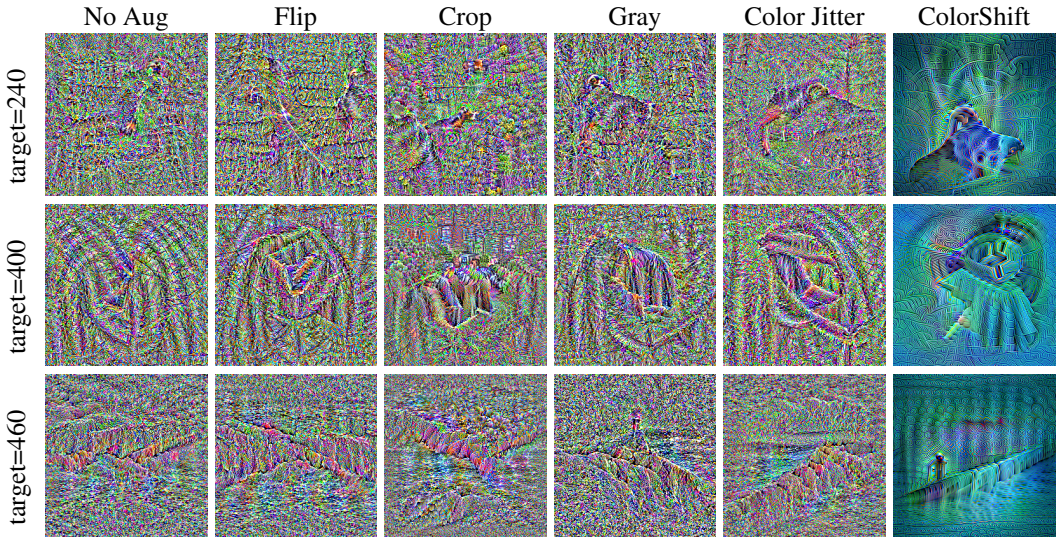


Figure 16: Effect of using different augmentations on inverting a naturally-trained ResNet-50.

A.6 *PII* ON ADDITIONAL NETWORKS

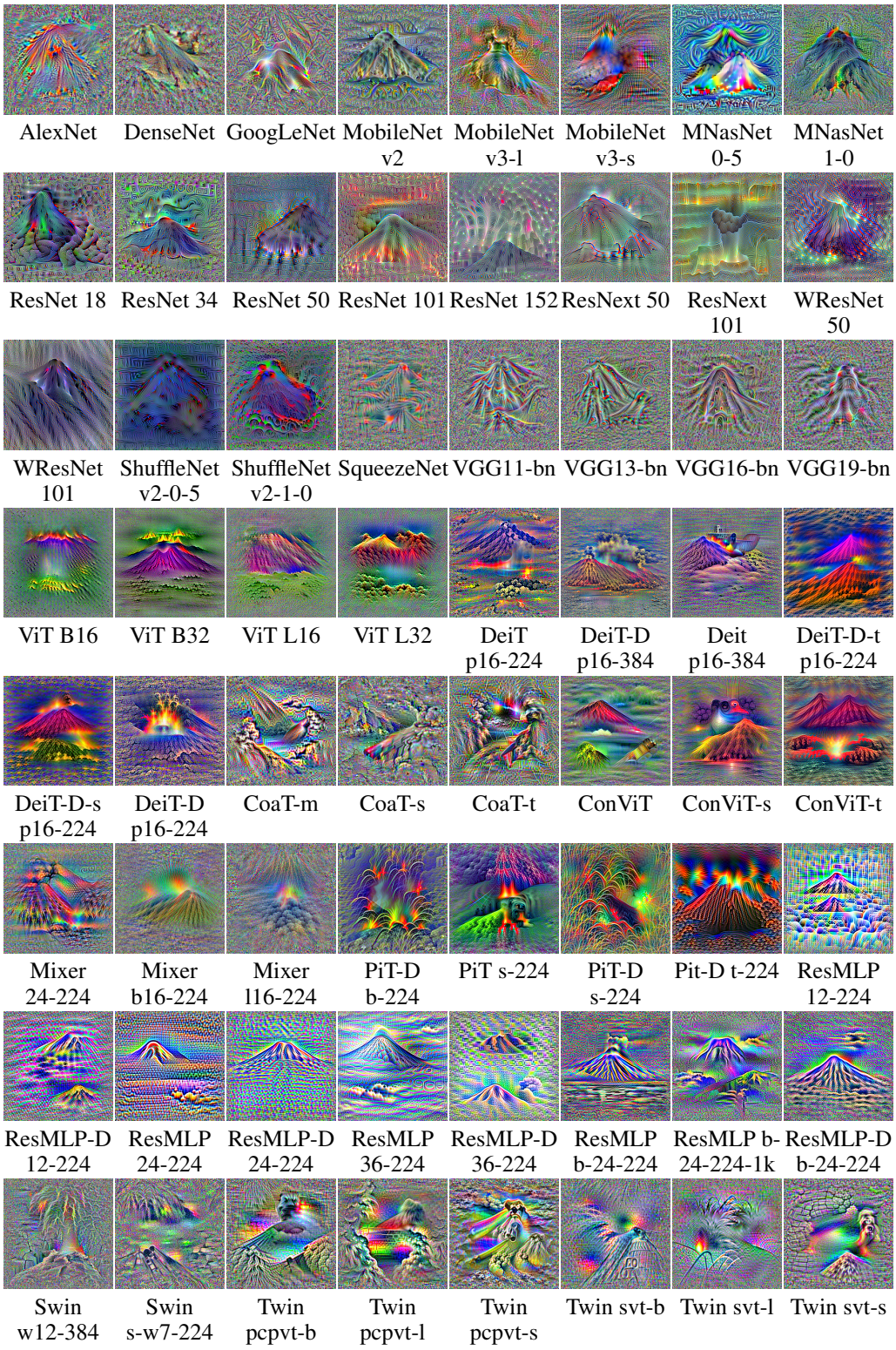


Figure 17: *PII* applied to various vision models for the Volcano class.

Figure 17 shows the results of Plug-In Inversion on various CNN, ViT, and MLP networks, adding to those shown in figure 7. See section B for model details.

B MODELS

In our experiments, we use publicly available pre-trained models from various sources. The following tables list the models used from each source, along with references to where they are introduced in the literature.

Alias	Name	Paper
AlexNet	alexnet	Krizhevsky et al. (2012)
DenseNet	densenet121	Huang et al. (2017)
GoogLeNet	googlenet	Szegedy et al. (2015)
MobileNet v2	mobilenet_v2	Sandler et al. (2018)
MobileNet-v2	mobilenet_v2	Sandler et al. (2018)
MobileNet v3-l	mobilenet_v3_large	Howard et al. (2019)
MobileNet v3-s	mobilenet_v3_small	Howard et al. (2019)
MNasNet 0-5	mnasnet0_5	Tan et al. (2019)
MNasNet 1-0	mnasnet1_0	Tan et al. (2019)
ResNet 18	resnet18	He et al. (2016)
ResNet-18	resnet18	He et al. (2016)
ResNet 34	resnet34	He et al. (2016)
ResNet 50	resnet50	He et al. (2016)
ResNet 101	resnet101	He et al. (2016)
ResNet-101	resnet101	He et al. (2016)
ResNet 152	resnet152	He et al. (2016)
ResNext 50	resnext50_32x4d	Xie et al. (2017)
ResNext 101	resnext101_32x8d	Xie et al. (2017)
WResNet 50	wide_resnet50_2	Zagoruyko & Komodakis (2016)
WResNet 101	wide_resnet101_2	Zagoruyko & Komodakis (2016)
W-ResNet-101-2	wide_resnet101_2	Zagoruyko & Komodakis (2016)
ShuffleNet v2-0-5	shufflenet_v2_x0_5	Ma et al. (2018)
ShuffleNet v2-1-0	shufflenet_v2_x1_0	Ma et al. (2018)
ShuffleNet v2	shufflenet_v2_x1_0	Ma et al. (2018)
SqueezeNet	squeezenet1_0	Iandola et al. (2016)
VGG11-bn	vgg11_bn	Simonyan & Zisserman (2014)
VGG13-bn	vgg13_bn	Simonyan & Zisserman (2014)
VGG16-bn	vgg16_bn	Simonyan & Zisserman (2014)
VGG19-bn	vgg19_bn	Simonyan & Zisserman (2014)

Figure 18: Pre-trained models from TorchVision: https://github.com/pytorch/vision.

Alias	Name	Paper
ViT B16	B_16_imagenet1k	Dosovitskiy et al. (2021)
ViT B32	B_32_imagenet1k	Dosovitskiy et al. (2021)
ViT B-32	B_32_imagenet1k	Dosovitskiy et al. (2021)
ViT L16	L_16_imagenet1k	Dosovitskiy et al. (2021)
ViT L32	L_32_imagenet1k	Dosovitskiy et al. (2021)

Figure 19: Pre-trained models used from : https://github.com/lukemelas/PyTorch-Pretrained-ViT.

Alias	Name	Paper
DeiT p16-224	deit_base_patch16_224	Touvron et al. (2021c)
DeiT P16 224	deit_base_patch16_224	Touvron et al. (2021c)
Deit-D p16-384	deit_base_distilled_patch16_384	Touvron et al. (2021c)
Deit Dist P16 384	deit_base_distilled_patch16_384	Touvron et al. (2021c)
deit p16-384	deit_base_patch16_384	Touvron et al. (2021c)
Deit-D-t p16-224	deit_tiny_distilled_patch16_224	Touvron et al. (2021c)
Deit-D-s p16-224	deit_small_distilled_patch16_224	Touvron et al. (2021c)
Deit-D p16-224	deit_base_distilled_patch16_224	Touvron et al. (2021c)

Figure 20: Pre-trained models from Touvron et al. (2021b) .

Alias	Name	Paper
CoaT-m	coat_lite_mini	Xu et al. (2021)
CoaT-s	coat_lite_small	Xu et al. (2021)
CoaT-t	coat_lite_tiny	Xu et al. (2021)
ConViT	convit_base	d'Ascoli et al. (2021)
ConViT-s	convit_small	d'Ascoli et al. (2021)
ConViT-t	convit_tiny	d'Ascoli et al. (2021)
ConViT tiny	convit_tiny	d'Ascoli et al. (2021)
Mixer 24-224	mixer_24_224	Tolstikhin et al. (2021)
Mixer b16-224	mixer_b16_224	Tolstikhin et al. (2021)
Mixer b16 224	mixer_b16_224	Tolstikhin et al. (2021)
Mixer b16-224-mill	mixer_b16_224_miil	Tolstikhin et al. (2021)
Mixer l16-224	mixer_l16_224	Tolstikhin et al. (2021)
PiT-D b-224	pit_b_distilled_224	Heo et al. (2021)
PiT Dist 224	pit_b_distilled_224	Heo et al. (2021)
PiT s-224	pit_s_224	Heo et al. (2021)
PiT-D s-224	pit_s_distilled_224	Heo et al. (2021)
PiT-D t-224	pit_ti_distilled_224	Heo et al. (2021)
ResMLP 12-224	resmlp_12_224	Touvron et al. (2021a)
ResMLP-D 12-224	resmlp_12_distilled_224	Touvron et al. (2021a)
ResMLP 24-224	resmlp_24_224	Touvron et al. (2021a)
ResMLP-D 24-224	resmlp_24_distilled_224	Touvron et al. (2021a)
ResMLP 36-224	resmlp_36_224	Touvron et al. (2021a)
ResMLP-D 36-224	resmlp_36_distilled_224	Touvron et al. (2021a)
ResMLP 36 Dist	resmlp_36_distilled_224	Touvron et al. (2021a)
ResMLP b-24-224	resmlp_big_24_224	Touvron et al. (2021a)
ResMLP b-24-224-1k	resmlp_big_24_224_in22ft1k	Touvron et al. (2021a)
ResMLP-D b-24-224	resmlp_big_24_distilled_224	Touvron et al. (2021a)
Swin w7-224	swin_base_patch4_window7_224	Liu et al. (2021b)
Swin l-w7-224	swin_large_patch4_window7_224	Liu et al. (2021b)
Swin l-w12-384	swin_large_patch4_window12_384	Liu et al. (2021b)
Swin w12-384	swin_base_patch4_window12_384	Liu et al. (2021b)
Swin P4 W12	swin_base_patch4_window12_384	Liu et al. (2021b)
Swin s-w7-224	swin_small_patch4_window7_224	Liu et al. (2021b)
Swin t-w7-224	swin_tiny_patch4_window7_224	Liu et al. (2021b)
Twin pcpvt-b	twins_pcpvt_base	Chu et al. (2021)
Twin PCPVT	twins_pcpvt_base	Chu et al. (2021)
Twins pcpvt-l	twins_pcpvt_large	Chu et al. (2021)
Twins pcpvt-s	twins_pcpvt_small	Chu et al. (2021)
Twins svt-b	twins_svt_base	Chu et al. (2021)
Twins svt-l	twins_svt_large	Chu et al. (2021)
Twins svt-s	twins_svt_small	Chu et al. (2021)

Figure 21: Pre-trained models used from: Wightman (2019)

C EVERY CLASS OF IMAGENET DATASET INVERTED

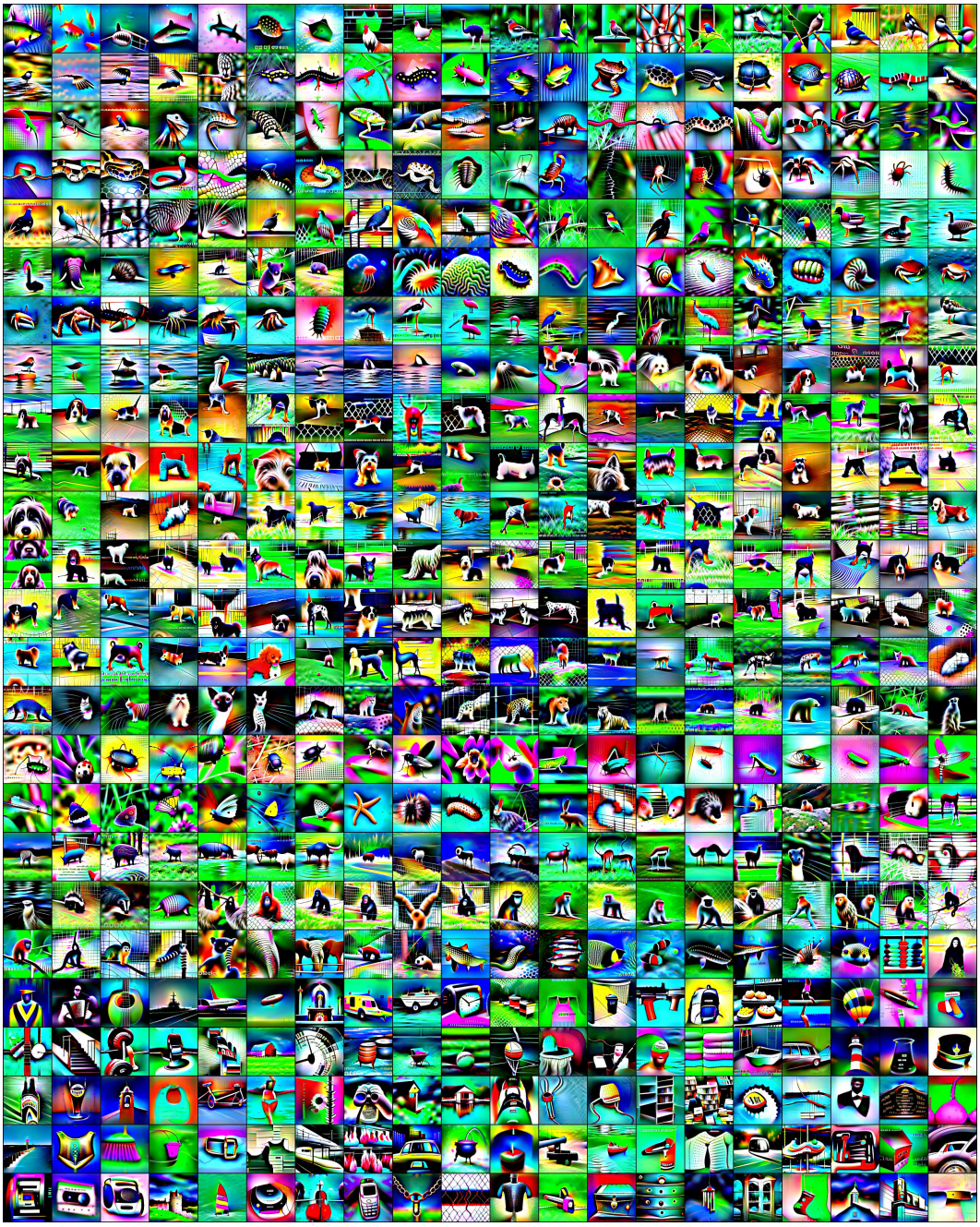


Figure 22: Inversion of first 500 classes of ImageNet for the Robust Model.

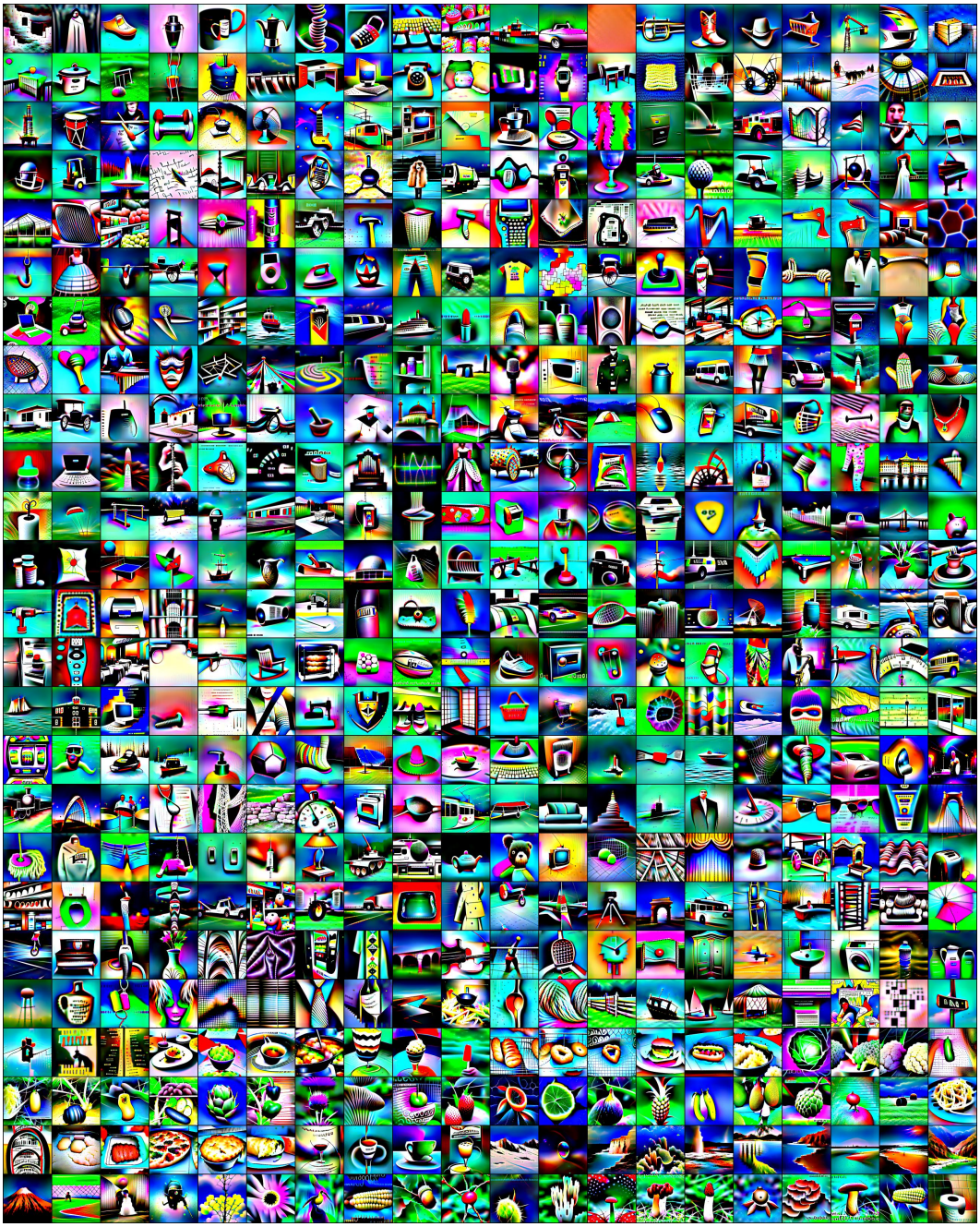


Figure 23: Inversion of second 500 classes of ImageNet for the Robust Model.

D ADDITIONAL EXPERIMENTAL SETTING

D.1 ROBUST MODELS

We use a robust RestNet-50 (He et al., 2016) model free-trained (Shafahi et al., 2019) on the ImageNet dataset (Deng et al., 2009). The setting we use for inverting robust models is very similar to that of *PII* explained in section 4 except for some differences. Throughout the paper, we use centering for robust models unless otherwise is mentioned (like when we are examining the effect of zoom and centering themselves). We use 0.0005 to scale total variation in the loss function. Also, we do not apply the data normalization layer before feeding the input to the network. In *PII* experiment setting, we apply a random ColorShift at each optimization step to each element in the ensemble. In the robust setting, we do not update the ColorShift variables μ, and σ for a fixed patch size, and we update these variables for the ensemble when we use a new patch size. Although using ColorShift would alleviate the need for using TV as discussed in section 3.2, and illustrated in figure 4 in the robust, and natural setting, we keep TV in our robust setting to make this setting more similar to that of previous inversion methods and to emphasize that it is a toy example for our ablation studies.

E OPTIMIZATION ALGORITHM

Algorithm 1: Optimization procedure for Plug-In Inversion

Input: Model f, class y, final resolution R, ColorShift parameters α, β, 'ensemble' size e, randomly initialized $\mathbf{x} \in \mathcal{I}^{3\times R/8\times R/8}$

for $s = 1, \ldots, 7$ **do**
- Upsample $\mathbf{x}$ to resolution $\frac{(2s+1)R}{16} \times \frac{(2s+1)R}{16}$
- Pad $\mathbf{x}$ with random noise to resolution $\frac{(s+1)R}{8} \times \frac{(s+1)R}{8}$
- **for** $i = 1, \ldots, 400$ **do**
  - $\mathbf{x}' = \text{Jitter}(\mathbf{x})$
  - **for** $n = 1, \ldots, e$ **do**
    - Draw $\mu \sim U(-\alpha, \alpha)^3, \sigma \sim \exp(U(-\beta, \beta))^3$
    - $\mathbf{x}_n = \text{ColorShift}_{\mu,\sigma}(\mathbf{x}')$
  - $\mathcal{L} = \frac{1}{e}\sum_{n=1}^{e} NLL(f(\mathbf{x_n}), y)$
  - $\mathbf{x} \leftarrow \text{Adam}_i(\mathbf{x}, \nabla_{\mathbf{x}}\mathcal{L})$
- **return** $\mathbf{x}$

F CIFAR-100 RESULTS

In Figure 24, we use *PII* to invert ViT models trained on ImageNet and fine-tuned on CIFAR-100. Similarly, Figure 25 shows inversion results from models fine-tuned on CIFAR-10. We emphasize that these were produced using identical settings to the ImageNet results in section 5.

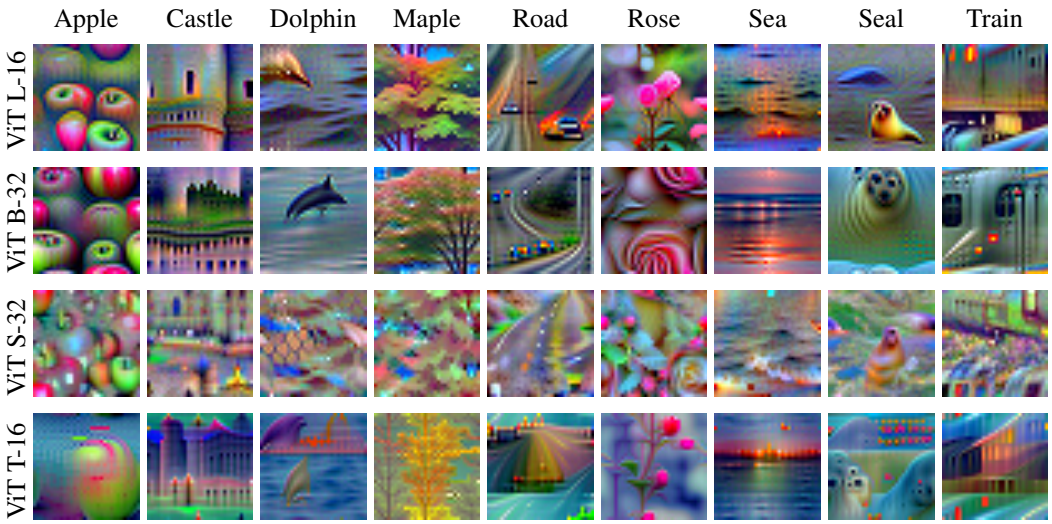


Figure 24: Inverting different CIFAR-100 model and class combinations using *PII*.

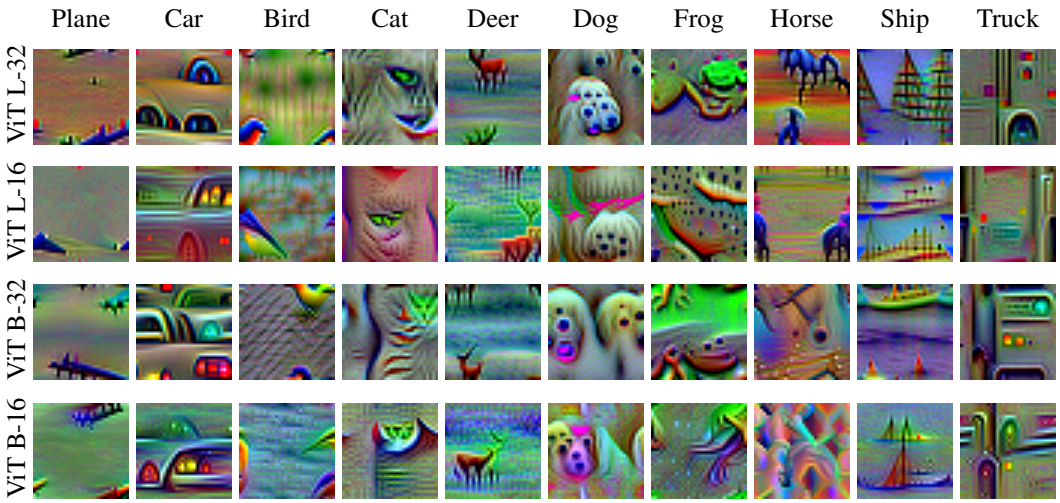


Figure 25: Inverting different CIFAR-10 model and class combinations using *PII*.

G ADDITIONAL BASELINE COMPARISONS

Figures 26 and 27 depict images inverted from various models using the DeepInversion and DeepDream methods as outlined by Yin et al. (2020). More specifically, we use DeepInversion with the prescribed regularization weights for CNNs, and use the same procedure minus the feature regularizer for all other models (in which BatchNorm is not used), keeping the same weights on the remaining terms[3]. We see less consistent performance across models using this method than when using *PII*, illustrating the need for model-specific tuning when using regularization-based approaches.

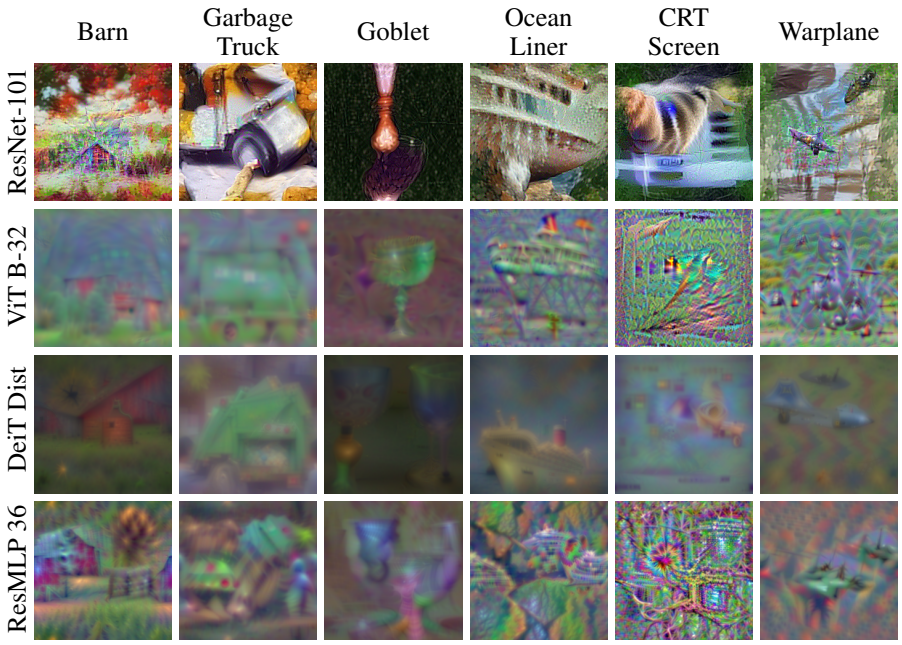


Figure 26: Inverting different model and class combinations for different classes using DeepInversion (top row) / DeepDream (other rows). Cross-reference figure 8.

[3]This is is the implementation of DeepDream (Mordvintsev et al., 2015) considered by Yin et al. (2020).

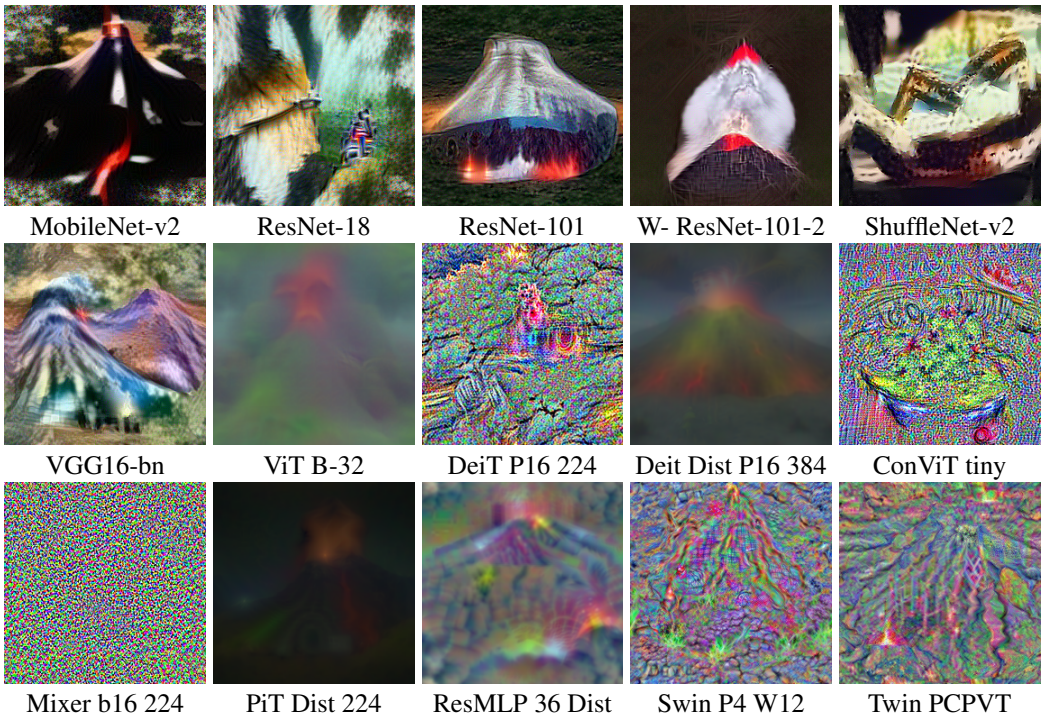
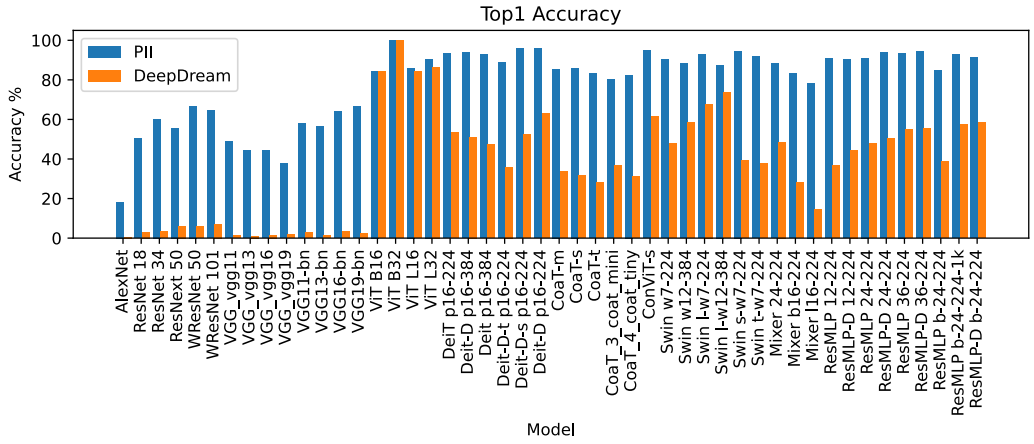


Figure 27: Images inverted from the Volcano class for various Convolutional, Transformer, and MLP-based networks using DeepInversion (CNN models) / DeepDream (non-CNN models). Cross-reference figure 7.

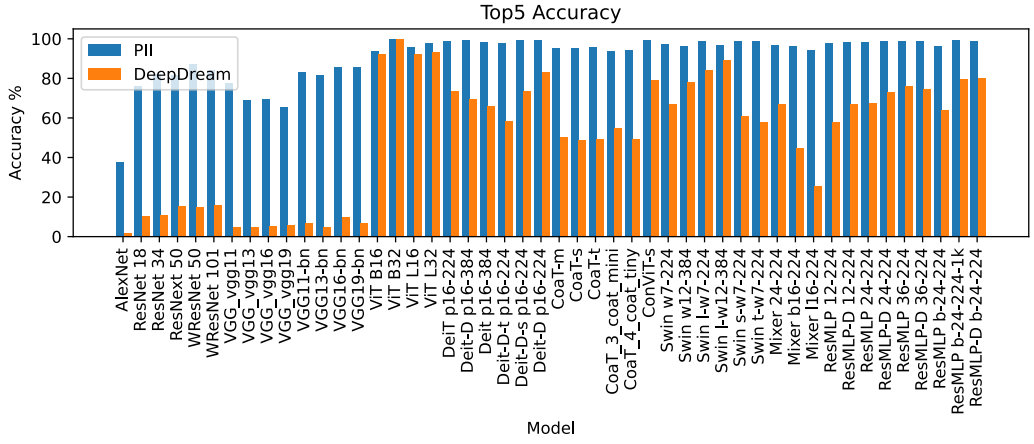
H Quantitative Results

To quantitatively evaluate our method, we invert a pre-trained ViT model to produce one image per class using *PII*, and do the same using DeepDream (i.e., DeepInversion minus feature regularization, which is not available for this model). We then use a variety of pre-trained CNN, ViT, and MLP models to classify these images. We find that every model achieves strictly higher top-1 and top-5 accuracy on the *PII*-generated image set (excepting the 'teacher' model, which perfectly classifies both). We compile these results in figure 29. Additionally, we compute the Inception score (Salimans et al., 2016) for both sets of images, which also favors *PII* over DeepDream, with scores of 28.17 ± 7.21 and 2.72 ± 0.23, respectively.

We also perform the same evaluation for images generated from a pre-trained ResMLP model. These results are more mixed; DeepDream images are classified much better by a small number of models, but the majority of models classify *PII* images better, and the average accuracy across models is approximately equal for both methods. Inception score, however, once again clearly favors *PII* over DeepDream, with scores of 6.79 ± 2.18 and 3.27 ± 0.47, respectively.

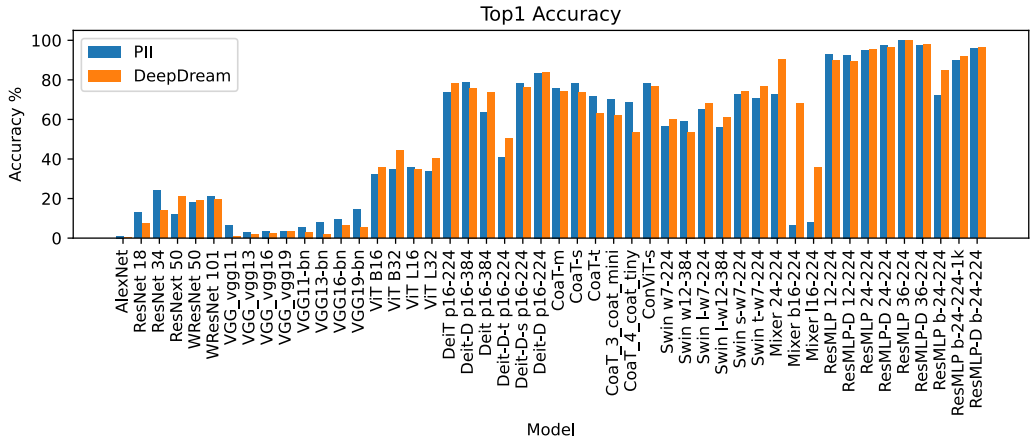


(a)

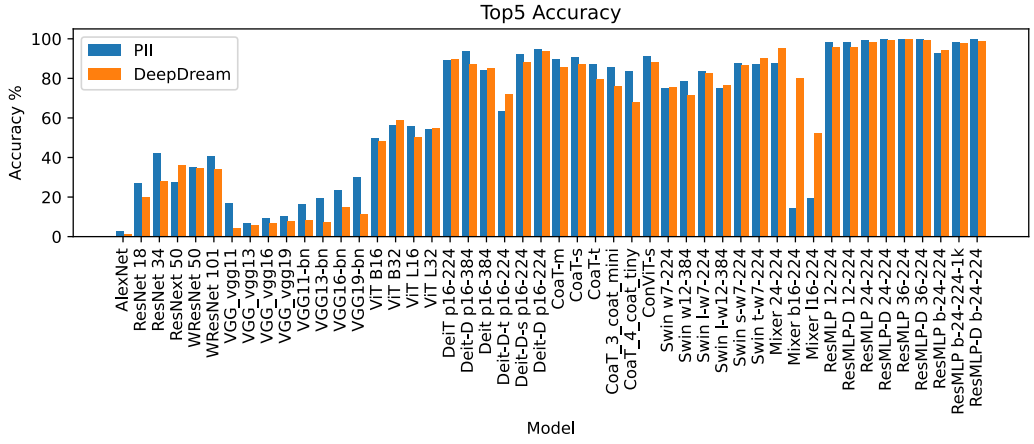


(b)

Figure 28: Top-1 (a) and top-5 (b) classification accuracy of various CNN, ViT, and MLP models evaluated on images generated from ViT B-32 using PII and DeepDream.



(a)



(b)

Figure 29: Top-1 (a) and top-5 (b) classification accuracy of various CNN, ViT, and MLP models evaluated on images generated from ResMLP 36-224 using PII and DeepDream.