PLUG-IN INVERSION: MODEL-AGNOSTIC INVERSION FOR VISION WITH DATA AUGMENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing techniques for model inversion typically rely on hard-to-tune regularizers, such as total variation or feature regularization, which must be individually calibrated for each network in order to produce adequate images. In this work, we introduce Plug-In Inversion, which relies on a simple set of augmentations and does not require excessive hyper-parameter tuning. Under our proposed augmentation-based scheme, the same set of augmentation hyper-parameters can be used for inverting a wide range of image classification models, regardless of input dimensions or the architecture. We illustrate the practicality of our approach by inverting Vision Transformers (ViTs) and Multi-Layer Perceptrons (MLPs) trained on the ImageNet dataset, tasks which to the best of our knowledge have not been successfully accomplished by any previous works.

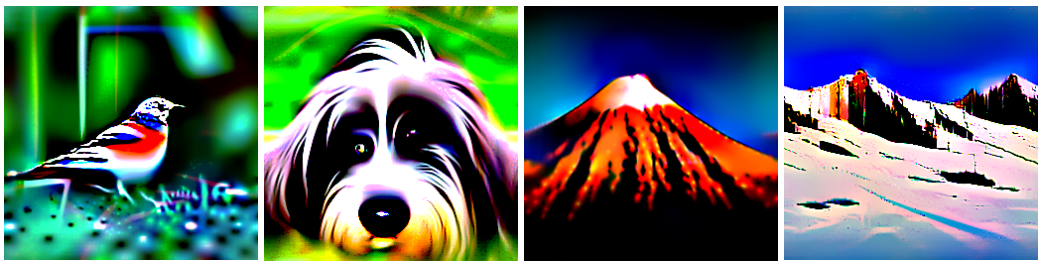
1 INTRODUCTION

Model inversion is an important tool for visualizing and interpreting behaviors inside neural architectures, understanding what models have learned, and explaining model behaviors. In general, model inversion seeks inputs that either activate a feature in the network (*feature visualization*) or yield a high output response for a particular class (*class inversion*) (Olah et al., 2017). Model inversion and visualization has been a cornerstone of conceptual studies that reveal how networks decompose images into semantic information (Zeiler & Fergus, 2014; Dosovitskiy & Brox, 2016). Over time, inversion methods have shifted from solving conceptual problems to solving practical ones. Saliency maps, for example, are image-specific model visualizations that reveal the inputs that most strong influence a model's decisions (Simonyan et al., 2014).

Recent advances in network architecture have posed major challenges for existing model inversion schemes. Convolutional Neural Networks (CNN) have long been the de-facto approach for computer vision tasks, and are the focus of nearly all research in the model inversion field. Recently, other architectures have emerged that achieve results competitive with CNNs. These include Vision Transformers (ViTs; Dosovitskiy et al., 2021), which are based on self-attention layers, and MLP-Mixer (Tolstikhin et al., 2021) and ResMLP (Touvron et al., 2021a), which are based on Multi Layer Perceptron layers. Unfortunately, most existing model inversion methods either cannot be applied to these architectures, or are known to fail. For example, the feature regularizer used in DeepInversion (Yin et al., 2020) cannot be applied to ViTs or MLP-based models because they do not include Batch Normalization layers (Ioffe & Szegedy, 2015).

In this work, we focus on class inversion, the goal of which is to find interpretable images that maximize the score a classification model assigns to a chosen label without knowledge about the model's training data. Class inversion has been used for a variety of tasks including model interpretation (Mordvintsev et al., 2015), image synthesis (Santurkar et al., 2019), and data-free knowledge transfer (Yin et al., 2020). However, current inversion methods have several key drawbacks. The quality of generated images is often highly sensitive to the weights assigned to regularization terms, so these hyper-parameters need to be carefully calibrated for each individual network. In addition, methods requiring batch norm parameters are not applicable to emerging architectures.

To overcome these limitations, we present *Plug-In Inversion* (*PII*), an augmentation-based approach to class inversion. *PII* does not require any explicit regularization, which eliminates the need to tune regularizer-specific hyper-parameters for each model or image instance. We show that *PII* is able to



Fringilla (10) Tibetan terrier (200) Volcano (980) Alp (970)

Figure 1: Inverted images from a robust-ResNet50 model trained on ImageNet-1k.

invert CNNs, ViTs, and MLP networks using the same architecture-agnostic method, and with the same architecture-agnostic hyper-parameters.

1.1 Contributions

We summarize our contributions as follows:

- We provide a detailed analysis of various augmentations and how they affect the quality of images produced via class inversion.
- We introduce *Plug-In Inversion* (*PII*), a new class inversion technique based on these augmentations, and compare it to existing techniques.
- We apply *PII* to dozens of different pre-trained models of varying architecture, justifying our claim that it can be 'plugged in' to most networks without modification.
- In particular, we show that *PII* succeeds in inverting ViTs and large MLP-based architectures, which to our knowledge has not previously been accomplished.
- Finally, we explore the potential for combining *PII* with prior methods.

In section 2, we review existing techniques for class inversion and outline the types of architectures we consider. In section 3, we explore individually the different augmentations that we use, and then describe the full *PII* algorithm. The remainder is devoted to results and analysis.

2 Background

2.1 Class Inversion

In the basic procedure for class inversion, we begin with a pre-trained model f and chosen target class y. We randomly initialize (and optionally pre-process) an image $\mathbf{x}$ in the input space of f. We then perform gradient descent to solve the following optimization problem for a chosen objective function $\mathcal{L}$,

$$\hat{x} = \arg\min_{\mathbf{x}} \mathcal{L}(f(\mathbf{x}, y)),$$

and the result is class image $\hat{x}$. For very shallow networks and small datasets, letting $\mathcal{L}$ be cross-entropy or even the negative confidence assigned to the true class can produce recognizable images with minimal pre-processing (Fredrikson et al., 2015). Modern deep neural networks, however, cannot be inverted as easily.

2.2 Regularization

Most prior work on class inversion for deep networks has focused on carefully designing the objective function to produce quality images. This entails combining a divergence term (e.g. cross-entropy) with one or more regularization terms (*image priors*) meant to guide the optimization towards an image with 'natural' characteristics. *DeepDream* (Mordvintsev et al., 2015), following

work on feature inversion (Mahendran & Vedaldi, 2015), uses two such terms: $\mathcal{R}_{\ell_2}(\mathbf{x}) = \|\mathbf{x}\|_2^2$, which penalizes the magnitude of the image vector, and total variation, defined as[1]

$$\mathcal{R}_{TV}(\mathbf{x}) = \left(\sum_{i,j}(x_{i+1,j} - x_{i,j})^2\right)^{\frac{1}{2}} + \left(\sum_{i,j}(x_{i,j+1} - x_{i,j})^2\right)^{\frac{1}{2}} + \left(\sum_{i,j}(x_{i+1,j+1} - x_{i,j})^2\right)^{\frac{1}{2}} + \left(\sum_{i,j}(x_{i+1,j} - x_{i,j+1})^2\right)^{\frac{1}{2}},$$

which penalizes sharp changes over small distances. *DeepInversion* (Yin et al., 2020) uses both of these regularizers, along with the feature regularizer

$$\mathcal{R}_{feat}(\mathbf{x}) = \sum_k \left(\|\mu_k(\mathbf{x}) - \hat{\mu}_k\|_2 + \|\sigma_k^2(\mathbf{x}) - \hat{\sigma}_k^2\|_2\right),$$

where μ_k, σ_k^2 are the batch mean and variance of the features output by the k-th convolutional layer, and $\hat{\mu}_k, \hat{\sigma}_k^2$ are corresponding Batch Normalization statistics stored in the model (Ioffe & Szegedy, 2015). Naturally, this method is only applicable to models that use Batch Normalization, which leaves out ViTs, MLPs, and even some CNNs. Furthermore, the optimal weights for each regularizer in the objective function vary wildly depending on architecture and training set, which presents a barrier to easily applying such methods to a wide array of networks.

2.3 Architectures for Vision

We now present a brief overview of the three basic types of vision architectures that we will consider.

Convolutional Neural Networks (CNNs) have long been the standard in deep learning for computer vision (LeCun et al., 1989; Krizhevsky et al., 2012). Convolutional layers encourage a model to learn properties desirable for vision tasks, such as translation invariance. Numerous CNN models exist, mainly differing in the number, size, and arrangement of convolutional blocks and whether they include residual connections, Batch Normalization, or other modifications (He et al., 2016; Zagoruyko & Komodakis, 2016; Simonyan & Zisserman, 2014).

Dosovitskiy et al. (2021) recently introduced **Vision Transformers** (ViTs), adapting the transformer architectures commonly used in Natural Language Processing (cite Vaswani). These models break input images into patches, combine them with positional embeddings, and use these as input tokens to self-attention modules. Others have proposed variants which require less training data (Touvron et al., 2021c), have convolutional inductive biases (d'Ascoli et al., 2021), or make other modifications to the attention modules (Chu et al., 2021; Liu et al., 2021b). (cite CoaT and CaiT)

Subsequently, a number of authors have proposed vision models which are based solely on **Multi-Layer Perceptrons** (MLPs), using insights from ViTs (Tolstikhin et al., 2021; Touvron et al., 2021a; Liu et al., 2021a). Generally, these models use patch embeddings similar to ViTs and alternate channel-wise and patch-wise linear embeddings, along with non-linearities and normalization.

We emphasize that as the latter two architecture types are recent developments, our work is the first to study them in the context of model inversion.

3 Plug-In Inversion

Prior work on class inversion uses augmentations like jitter, which randomly shifts an image horizontally and vertically, and horizontal flips to improve the quality of inverted images (Mordvintsev et al., 2015; Yin et al., 2020). The hypothesis behind their use is that different views of the same image should result in similar scores for the target class. These augmentations are applied to the input before feeding it to the network, and different augmentations are used for each gradient step

[1] This is the formulation used by (Yin et al., 2020); others are also common, such as the simpler version in (Mahendran & Vedaldi, 2015).

used to reconstruct x. In this section, we explore additional augmentations that benefit inversion before describing how we combine them to form the *PII* algorithm.

As robust models are typically easier to invert than naturally trained models (Santurkar et al., 2019; Mejia et al., 2019), we use a robust ResNet-50 (He et al., 2016) model trained on the ImageNet (Deng et al., 2009) dataset throughout this section as a toy example to examine how different augmentations impact inversion. Note, we perform the demonstrations in this section under slightly different conditions and with different models than those ultimately used for *PII* in order to highlight the effects of the augmentations as clearly as possible. The reader may find thorough experimental details in the appendix section D.1.

3.1 Restricting Search Space

In this section, we consider two augmentations to improve the spatial qualities of inverted images: *Centering* and *Zoom*. These are designed based on our hypothesis that restricting the input optimization space encourages better placement of recognizable features. Both methods start with small input patches, and each gradually increases this space in different ways to reach the intended input size. In doing so, they force the inversion algorithm to place important semantic content in the center of the image.

Centering Let x be the input image being optimized. At first, we only optimize a patch at the center of x. After a fixed number of iterations, we increase the patch size outward by padding with random noise, repeating this until the patch reaches the full input size. Figure 2 shows the state of the image prior at each stage of this process, as well as an image produced without centering. Without centering, the shift invariance of the networks allows most semantic content to scatter to the image edges. With centering, results remain coherent.

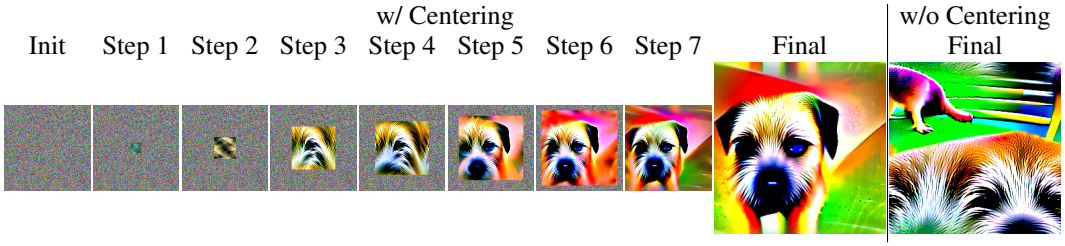


Figure 2: An image a different stages of optimization with centering (left), and an image inverted without centering (right), both for the Border Terrier class.

Zoom For zoom, we begin with an image x of lower resolution than the desired result. In each step, we optimize this image for a fixed number of iterations and then up-sample the result, repeating until we reach the full resolution. Figure 3 shows the state of an image at each step of the zoom procedure, along with an image produced without zoom. The latter image splits the object of interest at its edges. By contrast, zoom appears to find a meaningful structure for the image in the early steps and refines details like texture as the resolution increases.

We note that zoom is not an entirely novel idea in inversion. Yin et al. (2020) uses a similar technique as 'warm-up' for better performance and speed-up. However, we observe that continuing zoom throughout optimization contributes to the overall success of *PII*.

Zoom + Centering Unsurprisingly, we have found that applying zoom and centering simultaneously yields even better results than applying either individually, since each one provides a different benefit. Centering places detailed and important features (e.g. the dog's eye in Figure 2) near the center and builds the rest of the image around the existing patch. Zoom helps enforce a sound large-scale structure for the image and fills in details later.

The combined Zoom and Centering process proceeds in 'stages', each at a higher resolution than the last. Each stage begins with an image patch generated by the previous stage, which approximately minimizes the inversion loss. The patch is then up-sampled to a resolution halfway between the

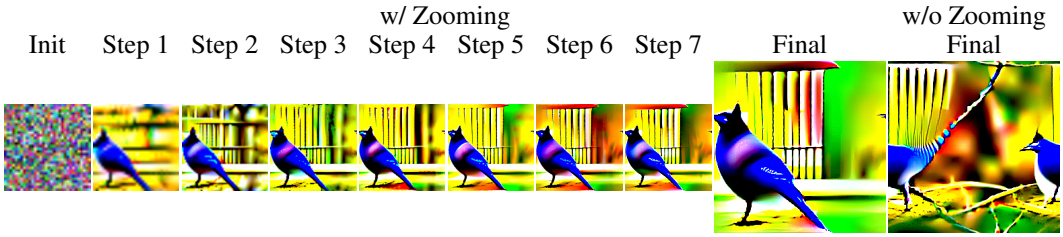


Figure 3: An image during different stages of optimization with zoom (left), and an image inverted without zoom (right), both for the Jay class.

previous stage and current stage resolution, allowing it to fill the center of the image, leaving a border which is padded with random noise. Then next round of optimization then begins starting from this newly processed image.

3.2 ColorShift Augmentation

The colors of the illustrative images we have shown so far are notably different from what one might expect in a natural image. This is due to *ColorShift*, a new augmentation that we now present.

ColorShift is an adjustment of an image's colors by a random mean and variance in each channel. This can be formulated as follows:

$$\text{ColorShift}(\mathbf{x}) = \sigma\mathbf{x} - \mu,$$

where μ and σ are C-dimensional[2] vectors drawn from $\mathcal{U}(-\alpha, \alpha)$ and $e^{\mathcal{U}(-\beta,\beta)}$, respectively, and are repeatedly redrawn after a fixed number of iterations. We use $\alpha = \beta = 1.0$ in all demonstrations unless otherwise noted. At first glance, this deliberate shift away from the distribution of natural images seems counterproductive to the goal of producing a recognizable image. However, our results show that using ColorShift noticeably increases the amount of visual information captured by inverted images and also obviates the need for hard-to-tune regularizers to stabilize optimization.

We visualize the stabilizing effect of ColorShift in Figure 4. In this experiment, we invert the model by minimizing the sum of a cross entropy and a total-variation (TV) penalty. Without ColorShift, the quality of images is highly dependent on the weight λ_{TV} of the TV regularizer; smaller values produce noisy images, while larger values produce blurry ones. Inversion with ColorShift, on the other hand, is insensitive to this value and in fact succeeds when omitting the regularizer altogether.

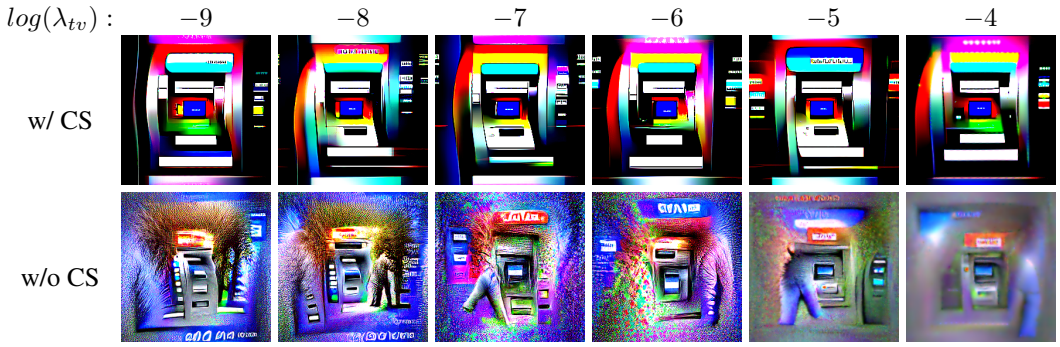


Figure 4: Inversions of the class, ATM, with and without ColorShift and with varying TV regularization strength. The inversion process with ColorShift is robust to changes in the λ_{tv} hyper-parameter, while without it, λ_{tv} seems to present a trade-off between noise and blur.

[2] C being the number of channels

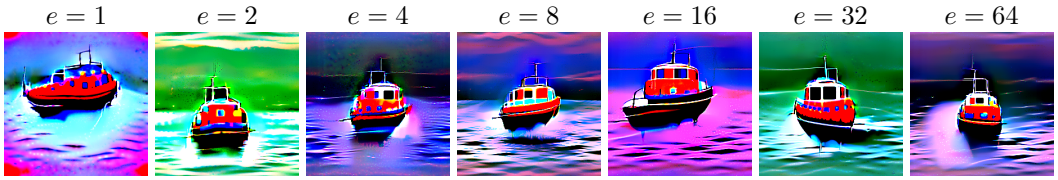


Figure 5: Effect of ensemble size in the quality of inverted images for the Tugboat class.

Other preliminary experiments show that ColorShift similarly removes the need for ℓ_2 or feature regularization, as our main results for *PII* will show. We conjecture that by forcing unnatural colors into an image, ColorShift requires the optimization to find a solution which contains meaningful semantic information, rather than photo-realistic colors, in order to achieve a high class score. Alternatively, as seen in Figure 9, images optimized with an image prior may achieve high scores despite a lack of semantic information merely by finding sufficiently natural colors and textures.

3.3 Ensembling

Ensembling is an established tool often used in dataset security (Souri et al., 2021) to enhanced inference (Opitz & Maclin, 1999). Similarly, we find that optimizing an ensemble composed of different ColorShifts of the same image simultaneously improves the performance of inversion methods. To this end, we minimize the average of cross-entropy losses $\mathcal{L}(f(\mathbf{x}_i, y))$, where the $\mathbf{x}_i$ are different ColorShifts of the image at the current step of optimization. Figure 5 shows the result of applying ensembling alongside ColorShift. Note that ColorShift is a strong augmentation so that even a small ensemble ($e = 8$) results in images of acceptable quality. We will later show that the ability to perform inversion with a small ensemble is critical, specifically on ViTs which have a massive memory footprint. More results on the effect of ensemble size can be found in Figure 14. We show the effect of ensembling using other well-known augmentations and compare them to ColorShift in Appendix Section A.5. We empirically show that ColorShift is the strongest among augmentations we tried for model inversion.

3.4 The Plug-In Inversion Method

We combine the jitter, ensembling, ColorShift, centering, and zoom techniques, and name the result Plug-In Inversion, which references the ability to 'plug in' any differentiable model, including ViTs and MLPs, using a single fixed set of hyper-parameters. In the next section, we detail the experimental method that we used to find these hyper-parameters, after which we present our main results.

4 Experimental Setup

In order to tune hyper-parameters of *PII* for use on naturally-trained models, we use the `torchvision` (Paszke et al., 2019) ImageNet-trained ResNet-50 model. We apply centering + zoom simultaneously in 7 'stages.' During each stage, we optimize the selected patch for 400 iterations, each using different random augmentations. Following (Yin et al., 2020), we use the ADAM (Kingma & Ba, 2014) optimizer with momentum $\beta_m = (0.5, 0.99)$, initial learning rate $lr = 0.01$, and cosine-decay. At the beginning of every stage, the learning rate and optimizer are re-initialized. At every step for zooming and centering, we define a scheduling for the size of the patches. We use the schedule $[56, 84, 112, 140, 168, 196, 224]$ for Centering and $[70, 98, 126, 154, 182, 210, 224]$ for Zoom. We use $\alpha = \beta = 1.0$ for the ColorShift parameters, and an ensemble size of $e = 32$. We ablate these choices in Figure 6 to validate our selected settings. Further ablation studies can be found in the appendix in figures 10, 13, and 14.

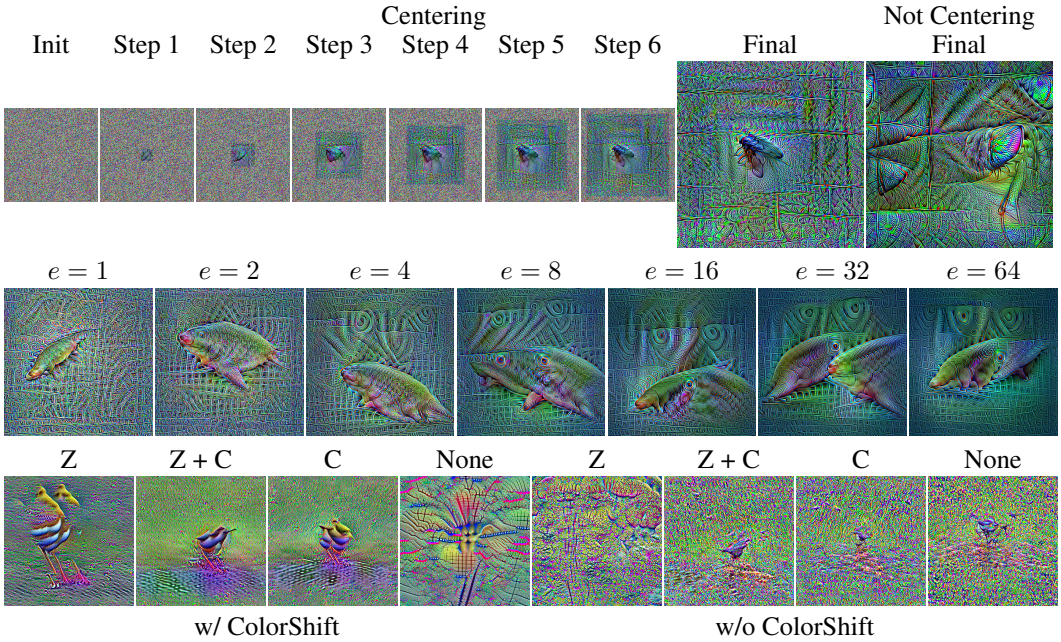


Figure 6: (top) Inversion for target class "Fly" with and without Centering. (center) Effect of ensembling size in the quality of inverted images for the Tench class. (bottom) The effect of various combinations of zoom, Centering, and ColorShiftwhen inverting the Dipper class.

5 Results

5.1 *PII* works on a range of architectures

We now present the results of applying Plug-In Inversion to different types of models. We once again emphasize that we use identical settings for the *PII* parameters in all cases.

Figure 7 depicts images produced by inverting the Volcano class for a variety of architectures, including examples of CNNs, ViTs, and MLPs. While the quality of images varies somewhat between networks, all of them include distinguishable and well-placed visual information. Many more examples are found in Figure 17 of the Appendix.

In Figure 8, we show images produced by *PII* from representatives of each main type of architecture for a few arbitrary classes. We note the distinct visual styles that appear in each row, which supports the perspective of model inversion as a tool for understanding what kind of information different networks learn during training.

5.2 Comparing *PII* to DeepInversion

We now consider the relative benefits of Plug-In Inversion and DeepInversion (Yin et al., 2020), a state-of-the-art class inversion method for CNNs. Figure 9 shows images from a few arbitrary classes produced by *PII* and DeepInversion (along with reference examples from the robust model demonstration in section 3). We additionally show images produced by DeepInversion using the output of *PII*, rather than random noise, as its initialization.

Using either initialization, DeepInversion clearly produces images with natural-looking colors and textures, which *PII* of course does not. However, DeepInversion alone results in some images that either do not clearly correspond to the target class or are semantically confusing. By comparison, *PII* again produces images with strong spatial and semantic qualities. Interestingly, these qualities appear to be largely retained when applying DeepInversion after *PII*, but with the color and texture improvements that image priors afford Mahendran & Vedaldi (2015).

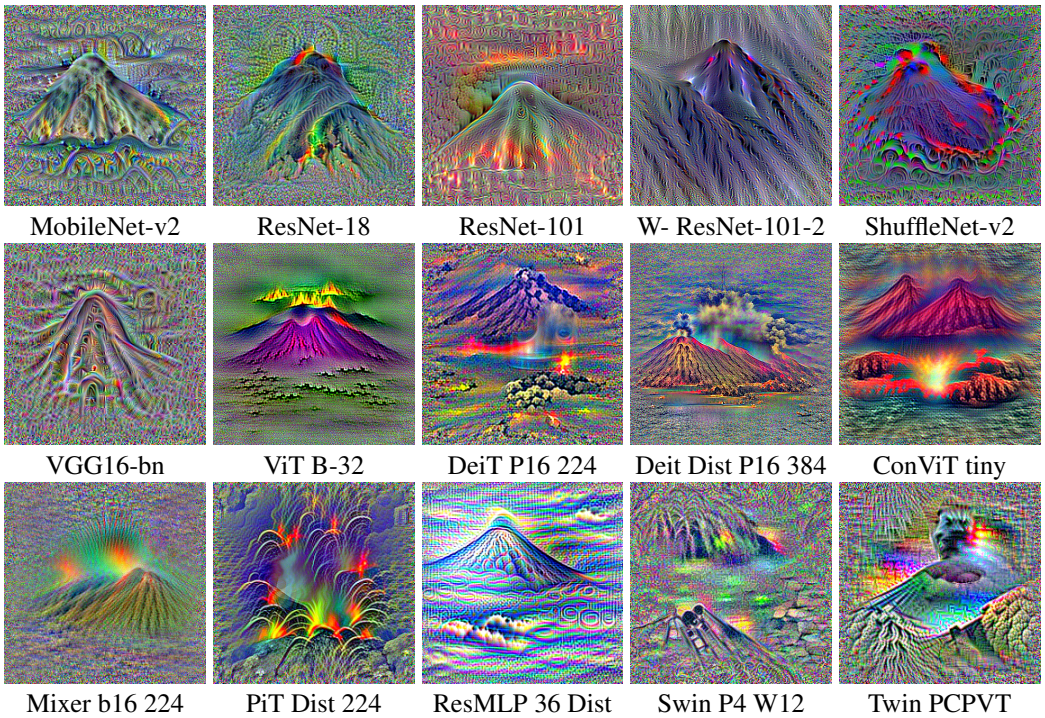


Figure 7: Images inverted from the Volcano class for various Convolutional, Transformer, and MLP-based networks using *PII*. For more details about networks, refer to Appendix B.

6 Conclusion

We studied the effect of various augmentations on the quality of class-inverted images and introduced Plug-In Inversion, which uses these augmentations in tandem. We showed that this technique produces intelligible images from a wide range of well-studied architectures, as well as the recently introduced ViTs and MLPs, without a need for model specific hyper-parameter tuning. We believe that augmentation-based model inversion is a promising direction for future research in understanding computer vision models.

7 Ethical Considerations

We propose Plug-In Inversion as a class inversion technique for the purpose of understanding vision models. However, we note that prior work has considered the potential of model inversion to compromise the security of a model's training data (Fredrikson et al., 2015; Yin et al., 2020). These areas of progress and other data privacy concerns (Zhu & Han, 2020; Geiping et al., 2020) make clear the need for caution when sensitive data is used to train deep learning models.

8 Reproducibility

All the models (including pre-trained weights) we consider in this work are publicly available from widely-used sources. Explicit details of model resources can be found in section B of the appendix. We also make the code used for all demonstrations and experiments in this work available at `https://github.com/youranonymousfriend/plugininversion`.

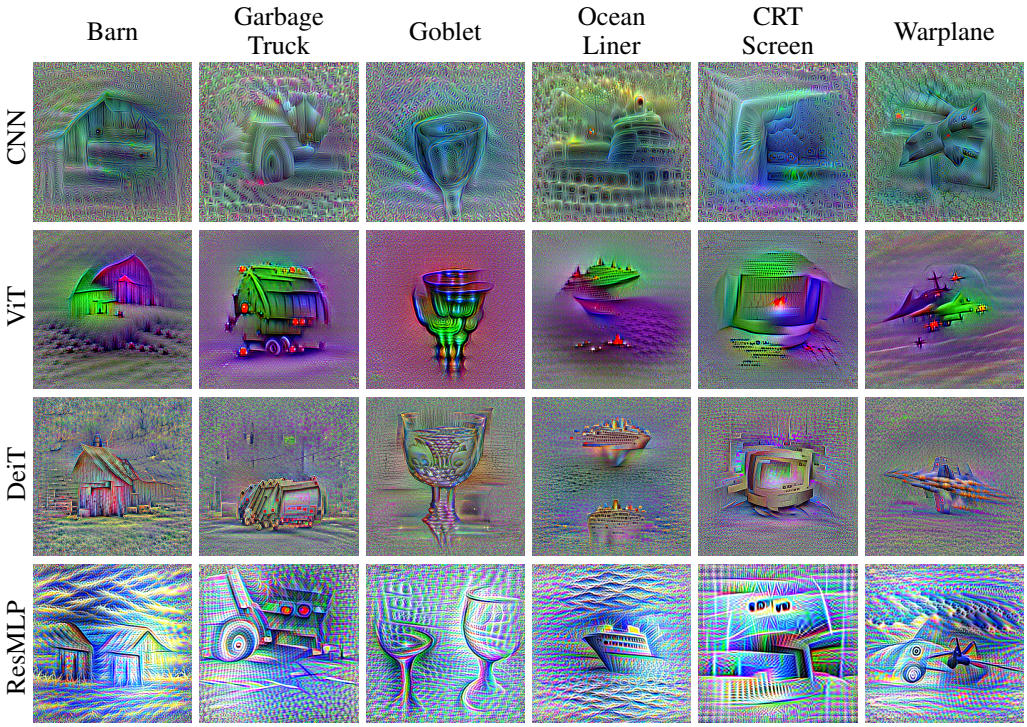


Figure 8: Inverting different model and class combinations for different classes using *PII*.

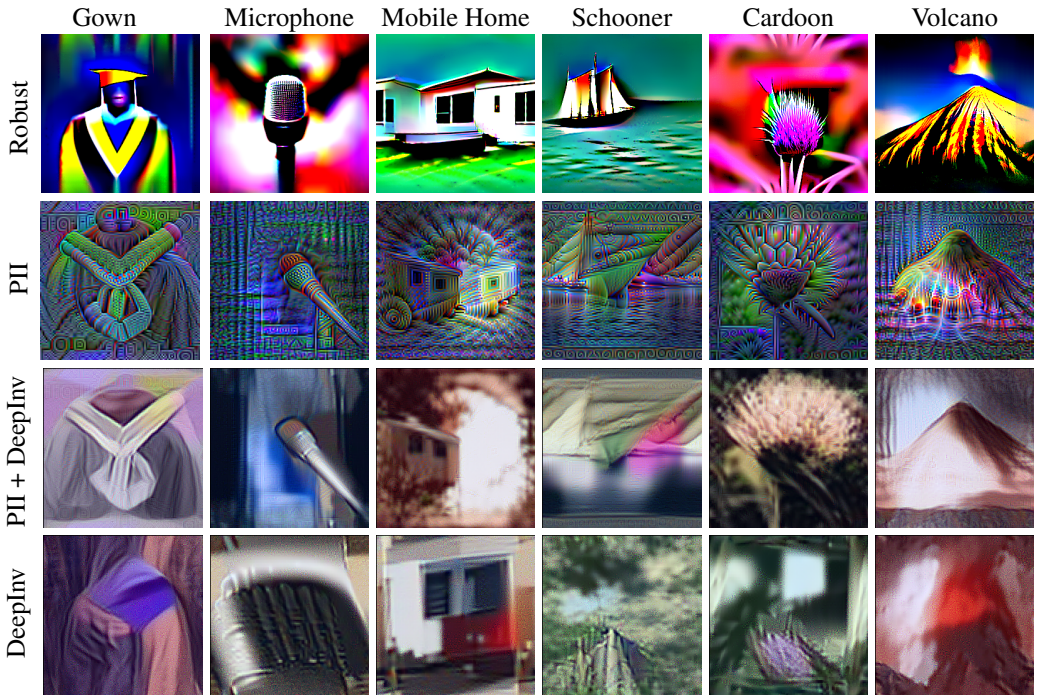


Figure 9: PII Inversion results for pre-trained ResNet 50. PII can be used as initialization for other methods such as DeepInversion.

REFERENCES

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *arXiv preprint arXiv:2104.13840*, 1(2):3, 2021.

Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. *arXiv preprint arXiv:2103.10697*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4829–4837, 2016.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1322–1333, 2015.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients–how easy is it to break privacy in federated learning? *arXiv preprint arXiv:2003.14053*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. *arXiv preprint arXiv:2103.16302*, 2021.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324, 2019.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. Pay attention to mlps. *arXiv preprint arXiv:2105.08050*, 2021a.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021b.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5188–5196, 2015.

Felipe A Mejia, Paul Gamble, Zigfried Hampel-Arias, Michael Lomnitz, Nina Lopatina, Lucas Tindall, and Maria Alejandra Barrios. Robust or private? adversarial training makes models more vulnerable to privacy attacks. *arXiv preprint arXiv:1906.06449*, 2019.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037, 2019.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Shibani Santurkar, Dimitris Tsipras, Brandon Tran, Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Image synthesis with a single (robust) classifier, 2019.

Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014.

Hossein Souri, Micah Goldblum, Liam Fowl, Rama Chellappa, and Tom Goldstein. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. *arXiv preprint arXiv:2106.08970*, 2021.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou. Resmlp: Feedforward networks for image classification with data-efficient training, 2021a.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, volume 139, pp. 10347–10357, July 2021b.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021c.

Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. *arXiv preprint arXiv:2104.06399*, 2021.

Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8715–8724, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Ligeng Zhu and Song Han. Deep leakage from gradients. In *Federated learning*, pp. 17–31. Springer, 2020.