Adversarial Attacks on Feature Visualization Methods

Jonathan Marty[*]
Flatiron Institute CCM &
Columbia University

Eugene Belilovsky
Concordia University &
MILA

Michael Eickenberg
Flatiron Institute CCM

Abstract

The internal functional behavior of trained Deep Neural Networks is notoriously difficult to interpret. Feature visualization approaches are one set of techniques used to interpret and analyze trained deep learning models. On the other hand interpretability methods themselves may be subject to be deceived. In particular, we consider the idea of an adversary manipulating a model for the purpose of deceiving the interpretation. Focusing on the popular feature visualizations associated with CNNs we introduce an optimization framework for modifying the outcome of feature visualization methods.

1 Introduction

Deep neural networks can be trained to perform many economically valuable tasks [5, 4]. They are already pervasive in many sectors, and their prevalence is only expected to increase over time. With increasing computational power and ever more available amounts of data, neural network architectures are growing in size and executing more and more intricate tasks. Given the increasing size and complexity of the deep neural network, interpreting its function, a discipline that always lags behind the cutting edge, may experience an ever harder time keeping up with new developments. However, for certain classes of critical applications, close inspection and guarantees of functionality will be more and more critical, especially when used in heavily regulated domains. Here we ask: Could a malicious actor conceal the true functionality of a neural network from an interpretability method by modifying the neural network? Given the increasing capacity of the architectures, this is likely to be a progressively more probable concern. We can take inspiration for this from the adversarial attack literature [2] which tries to modify network inputs (as opposed to the network itself). We propose to create an optimization procedure to manipulate the interpretation of a network while keeping its final behavior the same. In this work we concentrate on the VGG19 convnet architecture [9]. We study the activation maximization feature visualization of a neuron or channel norm and attempt to modify it while maintaining pretrained network accuracy. Section 2 explains the setup in detail and section 3 describes and evaluates empirical experiments.

2 Methods

We focus on object classification convolutional networks and define the *feature interpretation* of a feature map as

$$x^{\text{opt}} \in \arg\max_{x \in \mathcal{D}} f_\vartheta(x) \tag{1}$$

where $f_\vartheta(x)$ is either the activation of its center neuron or the spatial ℓ_2-norm of the full feature map, for an input image x. The parameters of the network are represented by ϑ and $\mathcal{D}$ is the domain of optimization. $\mathcal{D}$ can be a continuous space, such as an ℓ_2 ball in input image space, or for example the set of training images. This corresponds to a classic and popular approach for interpreting CNNs [10, 6, 8].[2]

[*]Correspondence to jonathan.n.marty@gmail.com, eugene.belilovsky@concordia.ca, eickenberg@flatironinstitute.org

[2]In particular the OpenAI microscope at `https://microscope.openai.com/` shows these optimizations for all the layers of a large array of networks.

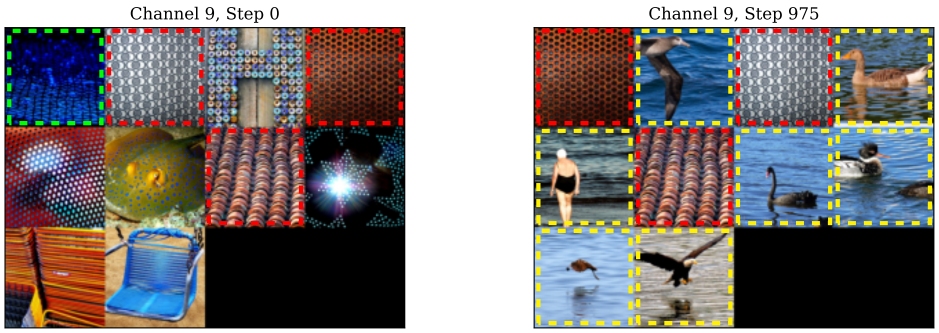

Figure 1: Top 10 images of feature channel 9 before and after manipulation of the model. The image x^{opt} is in the top left of the left-hand panel (green). After optimization it has disappeared from the top 10, as seen in the right-hand panel. While three texture images (red) remain in the top-10, the top-10 is now populated with water scenes (yellow), which constitutes a significant difference in selectivity.

Here we choose $\mathcal{D}$ to be the training set. Defining x^{opt}, i.e. here the top activating image of a feature map, as the interpretation of the feature map enables quantitative manipulation of this value.

Manipulation objective Given that x^{opt} is the maximally activating input for f_ϑ, we want to construct an optimization procedure such that x^{opt} is no longer the maximally activating input. Several objectives can be used to achieve this. Here we select a penalty of the form

$$m(\vartheta) = |f_\vartheta(x^{\text{opt}})|^2, \tag{2}$$

which encourages low absolute values for the activation and is minimized at an activation of 0.

Performance-maintaining objectives Modifying a neural network to change its maximal activation value to a certain input is likely to change its overall performance in a substantial way if no work is done to maintain the performance level. We propose three performance-maintaining objectives with different levels of strictness.

Using the original neural network objective A natural choice for maintaining neural network performance is to optimize the objective that was used to train the neural network in the first place. This usually happens using stochastic gradient descent on batches

$$\mathcal{L}_{\mathcal{B}}(\vartheta) = \sum_{i \in \mathcal{B}} \ell(F_\vartheta(x_i), y_i), \tag{3}$$

where F_ϑ represents the neural network output softmax function, ℓ is a loss function comparing its value for a sample x_i to the desired target y_i, and $\mathcal{B}$ is a set of indices representing a batch. We term this the *label cross-entropy* objective.

Softmax activations of unmodified network Performance can also be maintained by comparing softmax activations to those of the unmodified network in a distillation-like setting [3]. This leads to a stronger signal to replicate the same functional output (as opposed to comparing to one-hot labels in the original objective setting). This objective can be implemented on minibatches just as above:

$$\mathcal{L}_{\mathcal{B}}(\vartheta) = \sum_{i \in \mathcal{B}} \ell(F_\vartheta(x_i), F_{\text{orig}}(x_i)), \tag{4}$$

where both functions represent softmax outputs and ℓ is cross-entropy. We term this the *distillation cross-entropy* objective.

Preserving ℓ_2 distances between activations Instead of operating on neural network outputs, we can also encourage the minimization of differences between activations of the modified network to the original network.

$$\mathcal{L}_{\mathcal{B}}(\vartheta) = \sum_{i \in \mathcal{B}} \|\text{Act}_\vartheta(x_i) - \text{Act}_{\text{orig}}(x_i)\|^2, \tag{5}$$

where Act represents selected neural network activations, for example those of of the layer to be modified. This encourages all activations to inputs other than x^{opt} to remain the same. We term this the *activation ℓ_2 distance* objective.

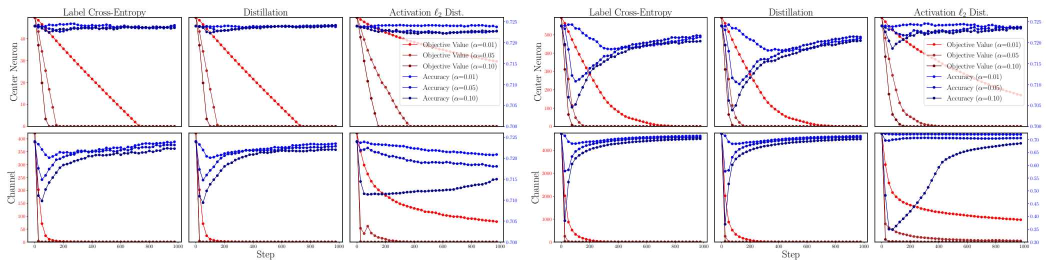

Figure 2: Optimization dynamics for the modification of one channel (left) and ten channels (right) of layer `conv5`. Red hues show the modification objective and blue hues show top-1 Imagenet validation accuracy. In the top row the modification objective acts upon the activation of the center neuron of the channel. In the bottom row, it acts on ℓ_2-norm of the channel activation. The columns of each figure correspond to the performance-maintaining objective used. Different color hues indicate a selection of tradeoff parameters α. Observe the striking evolution of the accuracy over the optimization trajectory: First, it dips with the initial change to the feature maps, then the accuracy recovers, likely because the neural network has changed strategy.

Joint optimization setup We combine the activation manipulation objective m with the performance preserving objective $\mathcal{L}_\mathcal{B}$ to obtain a global optimization objective:

$$\mathcal{L}(\vartheta) = (1-\alpha)\mathbb{E}_{\mathcal{B}\sim\text{data}}\mathcal{L}_\mathcal{B}(\vartheta) + \alpha m(\vartheta). \tag{6}$$

The hyperparameter α trades off between the strengths of the objectives. The expectation can be optimized using stochastic gradient descent.

3 Experiments

We present preliminary results of the proposed adversarial attack. We optimize the objective in 6 for one feature layer in a VGG19 architecture.

Experimental setup Throughout this experimental section, we work with the fifth convolutional layer of a VGG19 network [9] available in pre-trained form in `torchvision` [7], which we name `conv5`. This layer has 256 feature maps of size 56×56 pixels.

We determine x_k^{opt} for each channel k, where the optimally activating image is found over the training set. Using the x_k^{opt} we can create a modification objective for each channel, called $m_k(\vartheta)$, using equation 2. To perform the optimization we allow all layers of the neural network to evolve using stochastic gradient descent on the Imagenet training data [1]. All experiments run with a batch size of 100 and standard hyper-parameter settings for Imagenet. [3]In the following we will optimize this objective for an increasing number of channels, starting with 1 and 10 as a proof of concept, and then moving to all 256 channels.

Modifying several channels In a first experiment, we place the modification objective on one channel of `conv5`. We run the optimization for a total of 1000 batches of Imagenet training data. In Figure 2 left panel we show the temporal evolution of the modification objective m_0 and the evolution of the performance of the neural network. We evaluate the modification objectives using the center neuron activation and the channel ℓ_2 norm, for the three proposed performance-maintaining objectives. In Figure 2 we observe in particular that the neural network accuracy decreases at first, in concert with the reduction in response to x^{opt}. Next, when the activation to the top image reaches its minimum, the neural network accuracy recovers to the initial accuracy value.

In Figure 2 right panel we observe similar dynamics when modifying the activations of 10 channels using the modification objective $\sum_{k=0}^{9} m_k(\vartheta)$: Neural network accuracy initially dips together with the channel modification and later recovers.

Modifying all channels of `conv5` When modifying all channels of a layer, the information contained in an input must pass through a modified channel before reaching the output. It becomes visible in Figure 3 that activations to top images can be reduced, but not to 0. This is expected since imposing such a strong restriction on all channels would significantly impact information flow

[3]These are taken from `https://github.com/pytorch/examples/tree/main/imagenet`

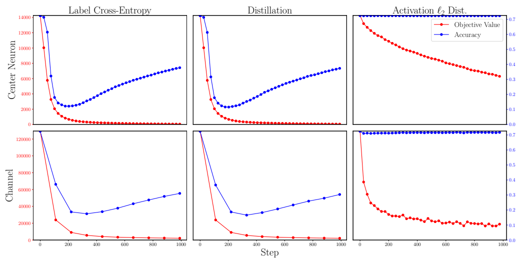

Figure 3: Optimization trajectories for the modification of all 256 channels of layer `conv5`. Channel ℓ_2 norm modification objective with distillation and labels cross-entropy uses $\alpha = 0.001$, all other configurations use $\alpha = 0.01$. See Fig. 2 for annotations. Modifying all 256 channels of a layer proves a harder task for maintaining accuracy. For the label cross-entropy and distillation the accuracy starts a slow and incomplete recovery (significant increases possible with more compute time). For the activation-based objective the accuracy is maintained, but the objective does not reach 0 anymore. Thus we need to check whether the goal (modifying x^{opt}) has been achieved.

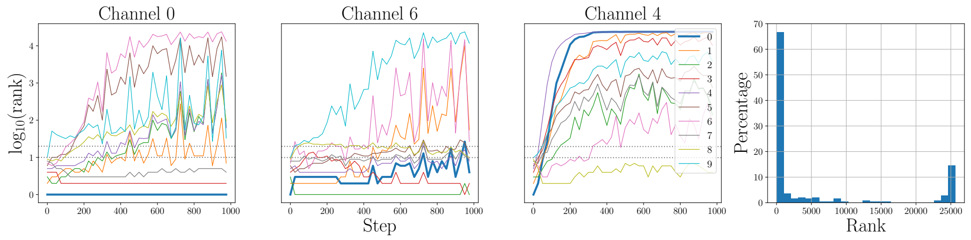

Figure 4: Left 3: Evolution of the rank of the activation of the top 10 images (with top-1 in bold) for three different channels of `conv5`. The x-axis shows optimization steps, the y-axis shows the rank among all training data in log-scale. On the left, for channel 0, we observe that the top-1 image stays in rank 1 - the modification objective was not strong enough to modify this channel. In the middle panel, the top-1 image ceases being the top-1 image but remains in the top-10. On the right, in channel 4, the top image ends up far at the bottom of the activation ranks. Right: Histogram of final ranks of the top-1 image after modification of 256 channels of `conv5`. Observe that about 60% of the channels did not modify their top-1 image in this setting. Around 10% of the channels radically changed the rank of the top-1 image. The remainder is distributed across the possible ranks.

through `conv5`. We observe a significant difference in behavior between the different performance maintaining objectives.

Analysis of the modifications We investigate the outcomes of the optimization. In the left three panels of Figure 4 we show for three channels the evolution of the ranks of the top-ranked images over the course of the optimization. The top-1 image is in bold blue. This selection represents three typical behaviors we have observed: The top-1 image may stay in place (left), move into mid-range ranks (middle), or essentially to the bottom of the activation distribution (right). In the right-hand panel of 4 we display a histogram of post-optimization ranks of the top image for all channels.

As an example of the visual impact of the modification performed, we show in Figure 1 the top 10 activating inputs of one feature channel from `conv5` after modifying it using the channel norm activation modification objective and the activation ℓ_2 distance performance maintaining objective. We observe that the top image x^{opt} is no longer among the top 10 (blue image in the top left corner), and that a number of new and different images now populate the top 10, likely leading to a different interpretation if it was to be summarized by a short description.

4 Discussion

We have investigated the possibility of modifying the outcome of activation maximization feature visualization techniques on one layer of a VGG19 network. We evaluated two different activation maximization techniques, center neuron activation and channel norm, and three different performance maintaining objectives, label cross-entropy, original softmax cross-entropy, and layer ℓ_2 distance. In the settings where accuracy was maintained, we observed an optimization dynamic involving a performance dip and recovery during modification. We found that some of the feature channels were successfully modified, so that the activation-maximizing image changes. The results here are preliminary and mixed. Future work will further explore modification procedures. We also strive to devise a measure of robustness to these modifications as well as interpretability methods that reliably co-vary or remain invariant with these feature changes.

Acknowledgements

EB acknowledges funding and resources from OpenPhilanthropy and Calcul Quebec.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[3] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015. cite arxiv:1503.02531Comment: NIPS 2014 Deep Learning Workshop.

[4] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[6] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in.

[7] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[8] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[9] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[10] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

A Appendix: Modification of feature visualization results

Note that all results in this appendix are for the choice $\alpha = 0.01$.

A.1 Ten channels

Channel 1 Figure 5 displays the original optimal image for channel 1 obtained using feature visualization as well as the resulting optimal images after optimization depending on the choice of modification (top and bottom rows) and performance maintaining (left, middle, and right columns) objectives. The most striking results can be seen for the distillation cross-entropy performance maintaining objective with channel ℓ_2 norm modification objective. While the original optimal image is dominated by interspersed red and blue dots, the optimal image after optimization sees the encroachment of lower-frequency black curves as well as red dots becoming more sparse.

We can dig into this change by examining the images the model activates strongly for. In Figure 6 we display the top 10 images for channel 1 before and after feature visualization. Observe that there is a mix of images with dots and with curves in the top 10 before optimization, whereas after optimization most of the top 10 contain prominent curves. Additionally, looking at the heatmap visual in Figure 7, which indicates the areas of the image that contribute the most to the channel ℓ_2 activation, we can see that lines on gray backgrounds across the top 10 images and specifically red lines on gray backgrounds in the top image contribute greatly to the channel ℓ_2 activation. This example demonstrates our framework's ability to alter the structure of feature visualization results.

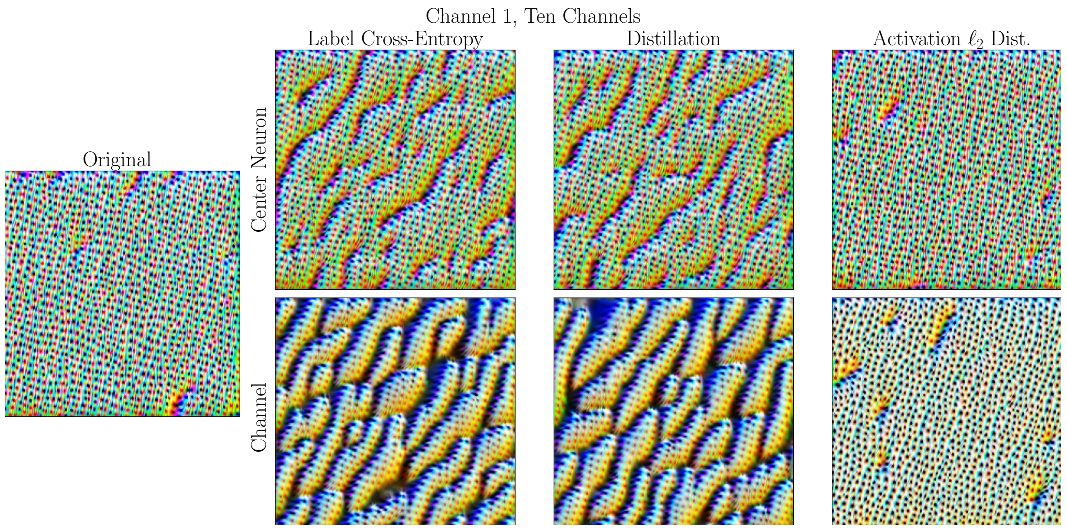

Figure 5: The optimal images for channel 1 before and after optimization. The original optimal image is shown on the left, while the optimal images following our optimization procedure are shown in the 6 images to the right, which are ordered visually based on the choice of modification (top and bottom rows) and performance maintaining (left, middle, and right columns) objectives.

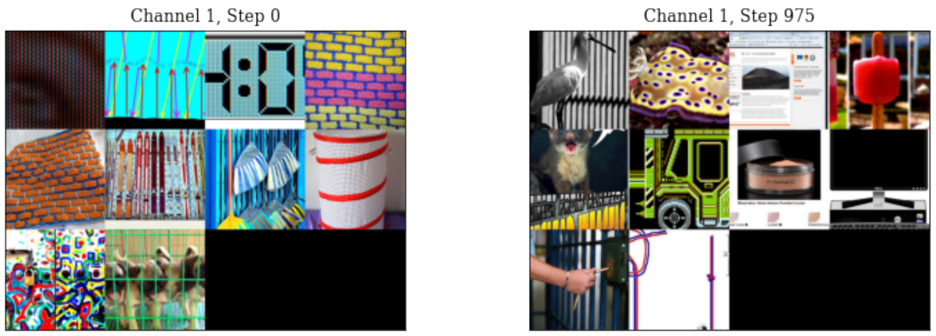

Figure 6: Top 10 images of feature channel 1 before and after optimization. The images are in row-column order, so the first row contains the top image, second to top image, and so on. The image x^{opt} is in the top left of the left-hand panel. After optimization it has disappeared from the top 10, as seen in the right-hand panel. Observe that no images remain in the top 10.

Channel 9 Another example of structural change of feature visualization results following optimization in our framework is channel 9. Figure 8 displays the original optimal image for channel 9 obtained using feature visualization as well as the resulting optimal images after optimization depending on the choice of modification (top and bottom rows) and performance maintaining (left, middle, and right columns) objectives. The most striking results can again be seen for the distillation cross-entropy performance maintaining objective with channel ℓ_2 norm modification objective. While the original optimal image contains a mix of blue and red dots interspersed with brown-green curves, the optimal image after optimization displays curves more prominently and dots less so.

In Figure 9 we display the top 10 images for channel 9 before and after feature visualization. Observe that the original top 10 images display dots over black backgrounds, whereas after optimization the top images contain far more curves. We can see from the heatmap visual in Figure 10 that these dots on black backgrounds account for a large amount of the channel ℓ_2 norm activation in the original set of top images.

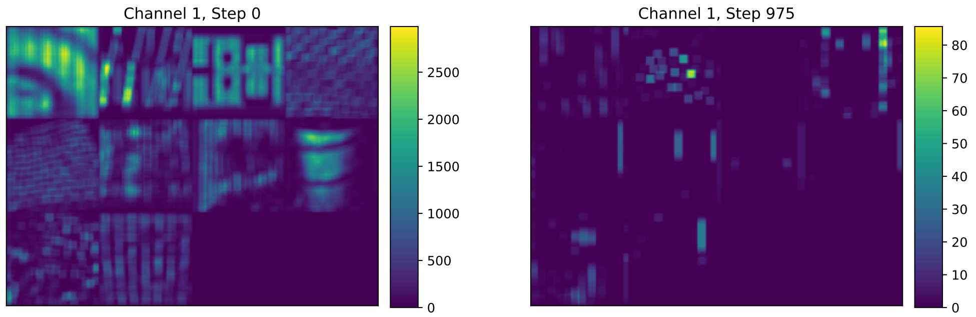

Figure 7: The heatmaps visual for feature channel 1. The images in this visual correspond with the top 10 visual. This visual is generated using the feature channel output, which is a 56x56 grid. A blank grid the same height and width of the image is initialized with zeros. For each activation in this grid, the each pixel in the part of the blank grid that corresponds to the part of the image that contributes to that activation is increased by that activation squared. The resulting visual indicates which parts of the image correspond most heavily to the channel ℓ_2 norm.

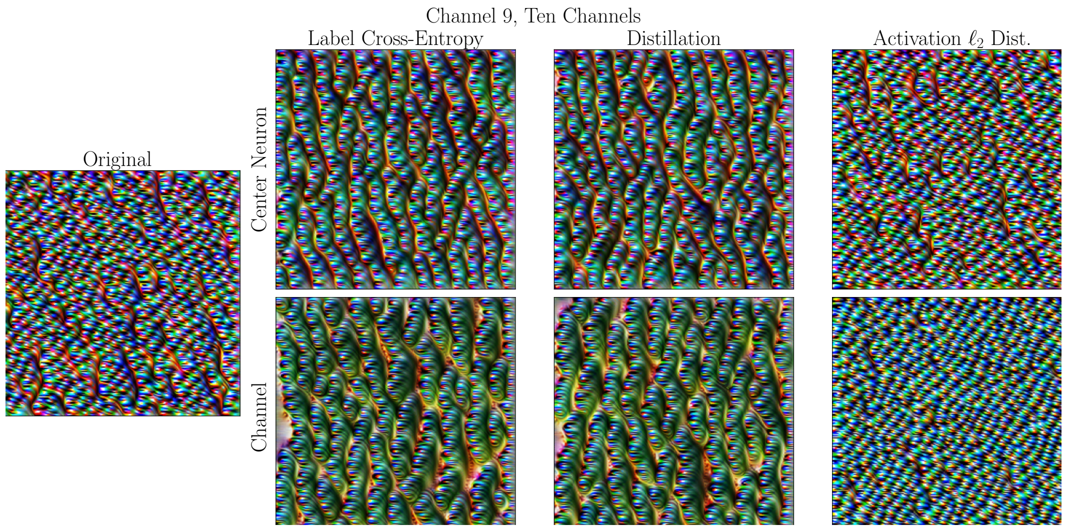

Figure 8: The optimal images for channel 9 before and after optimization. See Figure 5 for annotations.

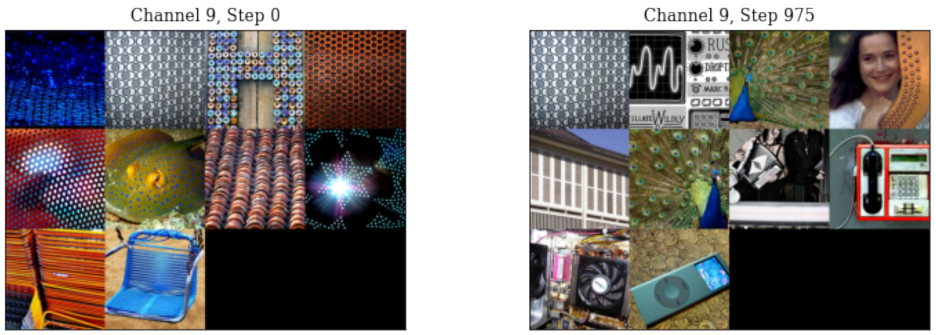

Figure 9: Top 10 images of feature channel 9 before and after optimization. See Figure 6 for annotations. The image x^{opt} is in the top left of the left-hand panel. After optimization it has disappeared from the top 10, as seen in the right-hand panel. While one image remains in the top 10, it contains prominent curves, which matches up with the new optimal image.

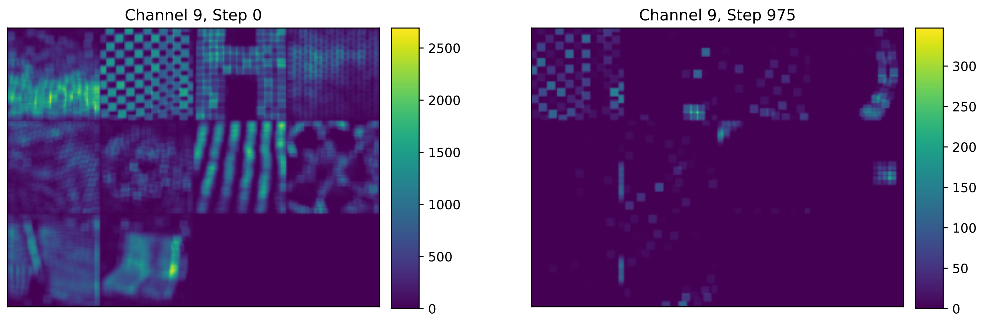

Figure 10: The heatmaps visual for feature channel 9. See Figure 7 for annotations.

A.2 256 channels

Channel 28 Figure 11 displays the original optimal image for channel 28 obtained using feature visualization on the top as well as the resulting optimal image after optimization using activation ℓ_2 distance performance maintaining objective with channel ℓ_2 norm modification objective is shown on the bottom. Since the use of distillation and labels cross-entropy performance maintaining objectives leads to very low accuracy in the 256 channels case, we have omitted the optimal images resulting from those runs. The original optimal image is dominated with a redish hue interspersed with white dots surrounded by either red or blue circles. In the resulting optimal image after optimization, the red hue has become brown and gray, but the white dots surrounded by red and blue circles remain.

We can dig into this change by examining the images the model activates strongly for. In Figure 12 we display the top 10 images for channel 28 before and after feature visualization. Observe that the top images before optimization tend to display white dots, such as snow on a surface or dew on a spiderweb, and that these white dots often appear on either a red or blue background. Observe also that the top image contains a red background, which is not common in the dataset but reflects the background color of the optimal image. The top images before optimization continue to display this pattern of white dots, most notably containing two images of dew on spiderwebs, but red backgrounds are much less common. This is reflected by the optimal image after optimization no longer having a red background. We can see in the heatmap visual in Figure 13 that white spots on darker backgrounds contribute greatly to the channel ℓ_2 norm for the top 10 images both before and after optimization. This example demonstrates our framework's ability to alter the color selectivity of feature maps when the activation ℓ_2 distance performance-maintaining objective is used.

Channel 11 Another example of color selectivity change of feature visualization results following optimization in our framework is channel 11. Figure 14 displays the original optimal image for channel 11 obtained using feature visualization on the top as well as the resulting optimal image after optimization using activation ℓ_2 distance performance maintaining objective with channel ℓ_2 norm modification objective is shown on the bottom. The original optimal image has an alternating pattern of bright yellow and orange colors overlaying its structure. In the resulting optimal image after optimization, the pattern has disappeared and been replaced with a brass tone, but the structure remains the same.

We can dig into this change by examining the images the model activates strongly for. In Figure 15 we display the top 10 images for channel 11 before and after feature visualization. Observe that the top image before optimization has a pattern of bright yellow and orange colors, which is uncommon in the dataset, whereas the the top image after optimization has a brass tone. However, both have similar structure. We can also see that a few of the other top 10 images after optimization contain objects with color tones close to brass. Observe in the heatmaps visual in Figure 16 that horizantal and slightly angled bars contribute greatly to the channel ℓ_2 norm for the top 10 images both before and after optimization.

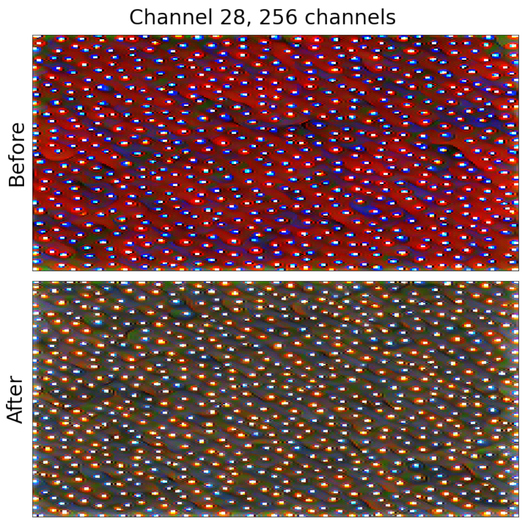

Figure 11: The optimal image for channel 28 before and after optimization. The original optimal image is shown on the top, while the optimal images following our optimization procedure is shown on the bottom.

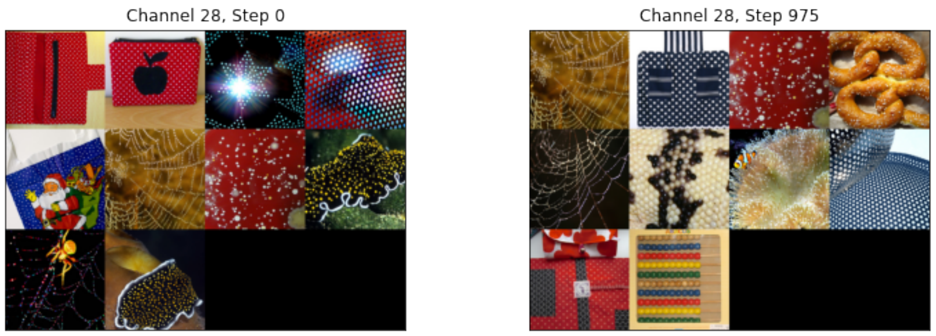

Figure 12: Top 10 images of feature channel 28 before and after optimization. The images are in row-column order, so the first row contains the top image, second to top image, and so on. The image x^{opt} is in the top left of the left-hand panel. After optimization it has disappeared from the top 10, as seen in the right-hand panel. Observe that no images remain in the top 10.

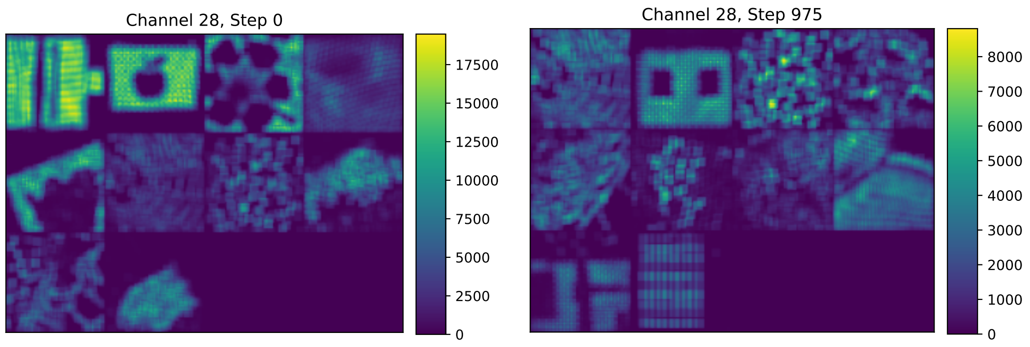

Figure 13: The heatmaps visual for feature channel 28. See Figure 7 for annotations.

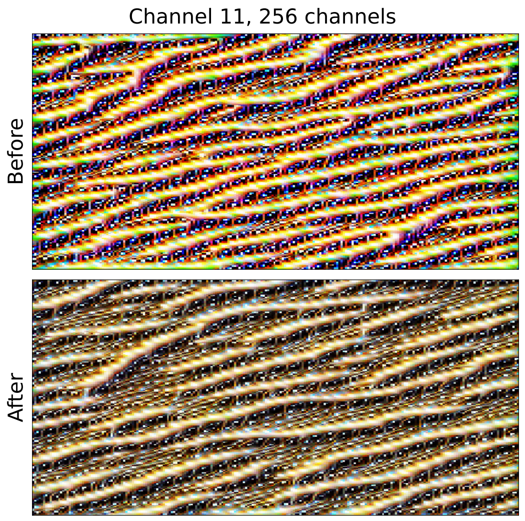

Figure 14: The optimal image for channel 11 before and after optimization. The original optimal image is shown on the top, while the optimal images following our optimization procedure is shown on the bottom.

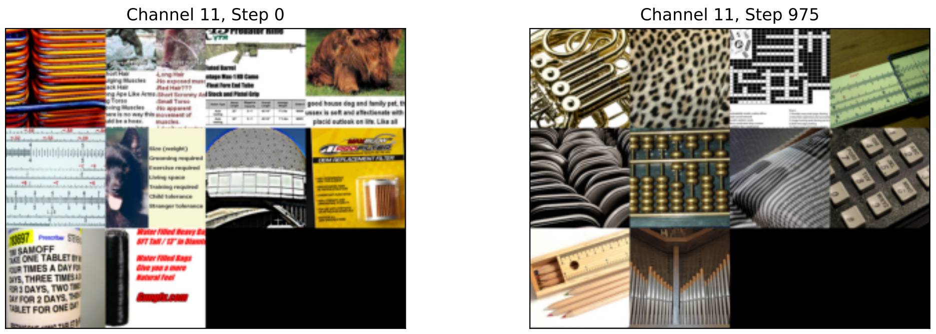

Figure 15: Top 10 images of feature channel 11 before and after optimization. The images are in row-column order, so the first row contains the top image, second to top image, and so on. The image x^{opt} is in the top left of the left-hand panel. After optimization it has disappeared from the top 10, as seen in the right-hand panel. Observe that no images remain in the top 10.

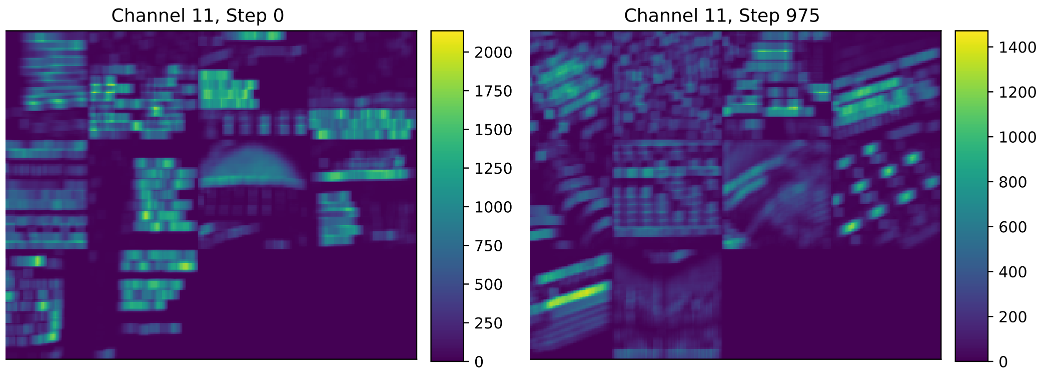

Figure 16: The heatmaps visual for feature channel 11. See Figure 7 for annotations.