Visualizing Adversarial Input in CNN

Akash Vemulapalli, Mohit Aggarwal, Rutvik Marakana, Shivanshu Dwivedi, Sohum Gala

Project Goals

- Provide visualizations for CNN execution on adversarial inputs
- Demonstrate where the divergence between normal and adversarial input is
- Extend existing PyTorch visualization library to make it more suited for visualizing adversarial inputs
    - Add support for generating adversarial examples
    - Extend support for visualization to other CNN architectures (VGG16 and ResNet)
    - Add support for visualizing image differences

Pretrained models error-rates

Network	***Top-1 error***	***Top-5 error***
AlexNet	43.45	20.91
VGG16	28.41	9.62
ResNet50	23.85	7.13

Source for pretrained models error-rates

Recap of Adversarial Input Generation

- Current functionality: high confidence predictions for unrecognizable images
- New functionality: high confidence predictions for incorrect images
  - We applied this methodology to the library's existing functionality to perturb existing images

$$x \leftarrow x + \alpha \cdot \frac{\partial a_i(x)}{\partial x}$$

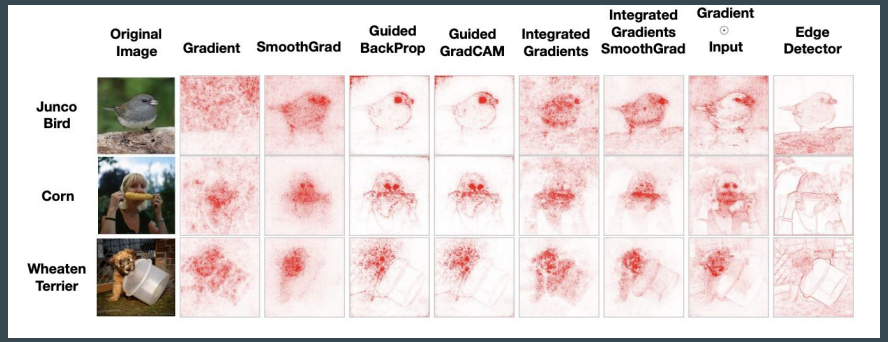
Model Resilience Against Adversarial Attacks

- Goal is to separate inputs into easy, medium, difficult, and very difficult cases
- Evaluate model accuracy with increasingly perturbed images
- Starting with a correctly classified image, perturb it to all 1000 target classes in ImageNet using varying numbers of iterations of the previous algorithm
- Evaluate accuracy of model's ability to predict the original class from the perturbed image

Number of perturbation iterations	1	4	7	10
Model accuracy	0.9948	0.6178	0.2775	0.1152

Gradient Saliency

- Saliency heatmaps give insight into what parts of the image lead to the final classification decision.
  - Vanilla saliency is the original idea proposed in Simonyan et. al. (2013)
  - We compute the gradient with respect to the input in backpropagation so backpropagation saliency was found to have the best results in Adebayo et al. (2018).
- We can visualize saliency maps for misclassified images because it gives a great idea into what pixels in the image are causing the misclassification



Visual from Adebayo et al. (2018)

Gradient Saliency

- Calculate gradient with respect to pixels
- Have indicator function to deal with ReLU signs

$$E_{grad}(I_0) = \frac{\delta S_c}{\delta I}|_{I=I_0}$$

$$\frac{\delta f}{\delta X_n} = \frac{\delta f}{\delta X_{n+1}} \cdot \mathbf{I}(X_n > 0)$$

- Generate adversarial image
  - Run Saliency map to find out which aspects of the image lead to the misclassification
- Gradient Saliency Source Code

Goal

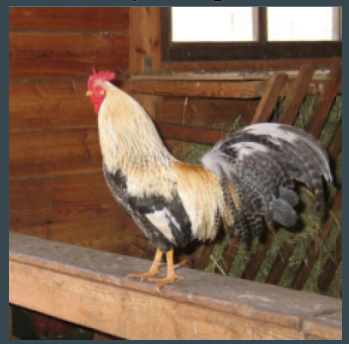
- See how a class of images gets impacted as more and more perturbations are added
- As the images get more and more “adversarial” the classification becomes worse
  - We will visualize the layers with 1/4/7/10 iterations of perturbations

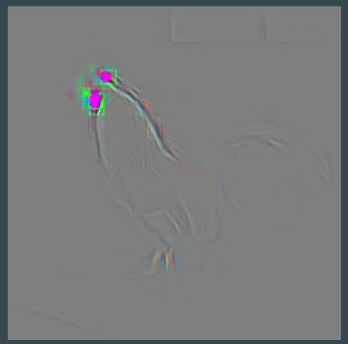
- The image is of a hen that was impacted to be misclassified as a snake.
- As the number of iterations increase, the model will do a worse job at classifying the image. “How many iterations” depend on the particular input image and target class but can be visualized using our project.

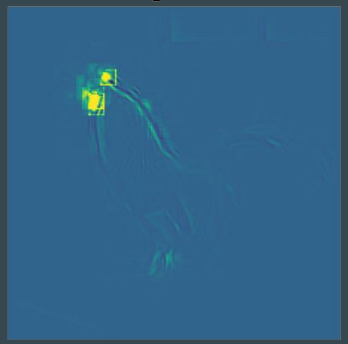
Saliency Maps

Original

1 iterations

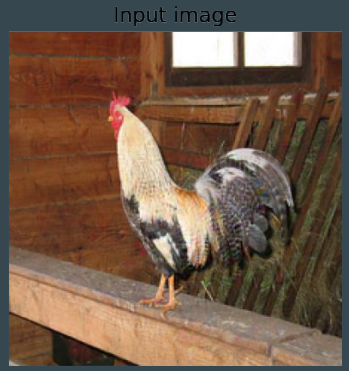


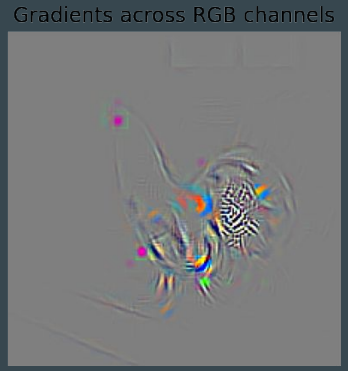


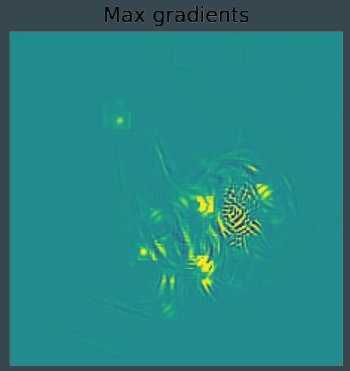


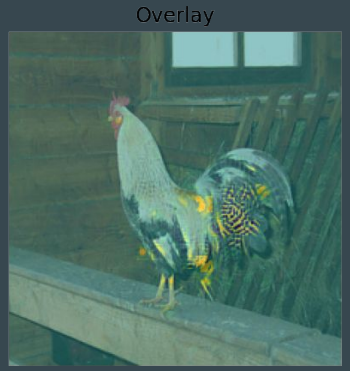
Saliency Maps

4 iterations

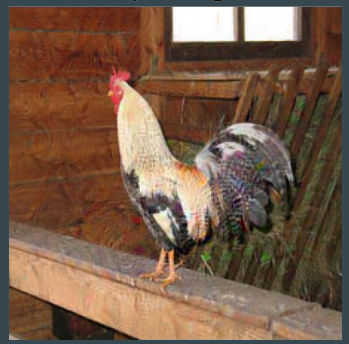


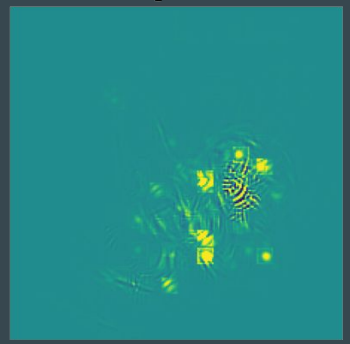






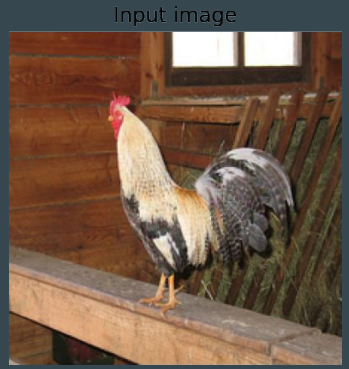
7 iterations



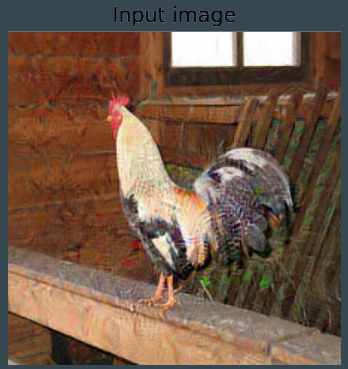


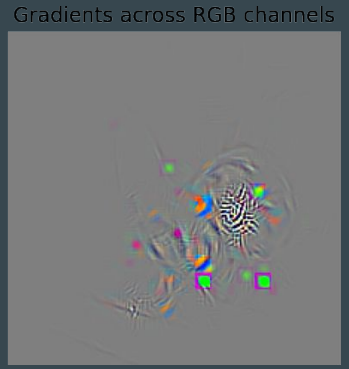
Saliency Maps

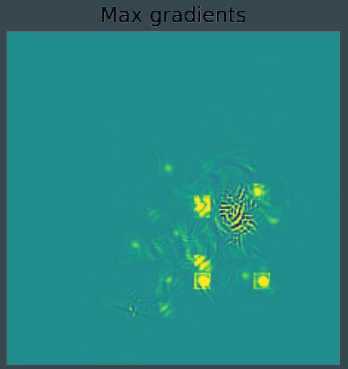
Original

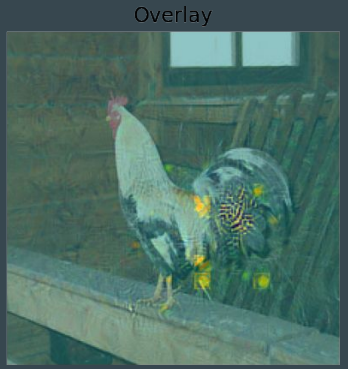


10 iterations









Observations

- When there isn't a lot of perturbations, the model predicts things accurately.
- We see that AlexNet is making its classification decision in the 0/1 iteration case mostly based on the comb and wattle near its face.

- However, as perturbations are added, the decision is manipulated based on the adversarial attack. The added perturbations leads to the misclassification which can be seen in the saliency map.

Layerwise Relevance

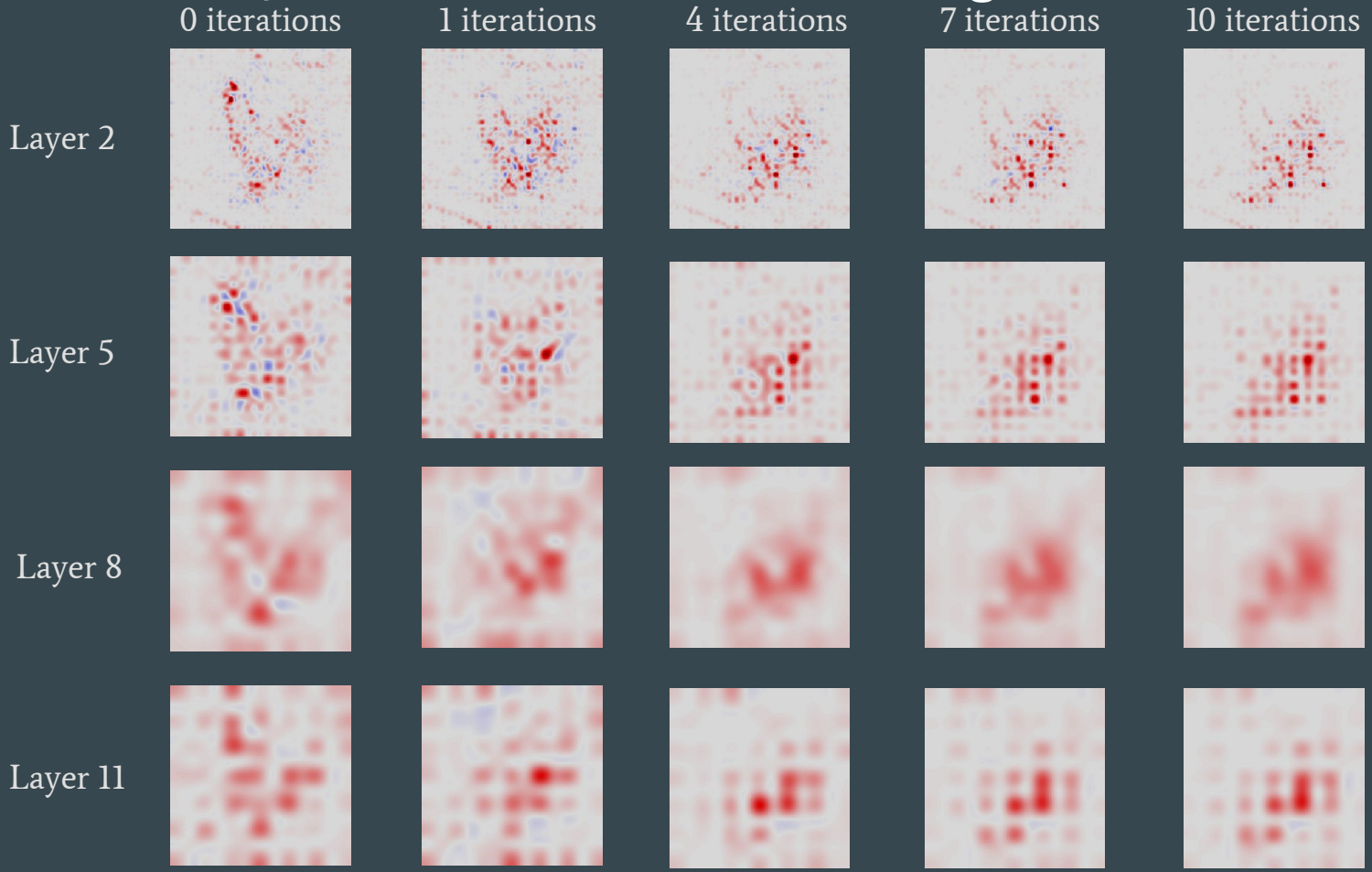
- Layer-Wise Relevance Propagation: An Overview
- Used to generate meaningful information from input images
- Decomposes the prediction of CNN computed over a sample
- Produced a heatmap in the input space
- Highlight the positive contributions to the network classification

$$R_j = \sum_k \frac{z_{jk}}{\sum_j z_{jk}} R_k.$$

- Tested the model on:
  - Layers 2, 5, 9 and 11 of AlexNet
  - Layers 9, 16, 23 and 30 of VGG16
  - Wanted evenly spaced layers in each architecture

Layerwise Relevance Source Code

Layerwise Relevance - Increasing Perturbation



Hierarchical Class Activation Maps - Background

- LayerCAM: Exploring Hierarchical Class Activation Maps for Localization
  - Originally used for localization
- Extension of Grad-CAM - improving performance for shallow layers

Grad-CAM

$$w_k^c = \frac{1}{N} \sum_i \sum_j g_{ij}^{kc}$$

$$M^c = \text{ReLU}\left(\sum_k w_k^c \cdot A_k \right)$$

LayerCAM

$$w_{ij}^{kc} = relu(g_{ij}^{kc})$$

$$\hat{A}_{ij}^k = w_{ij}^{kc} \cdot A_{ij}^k$$

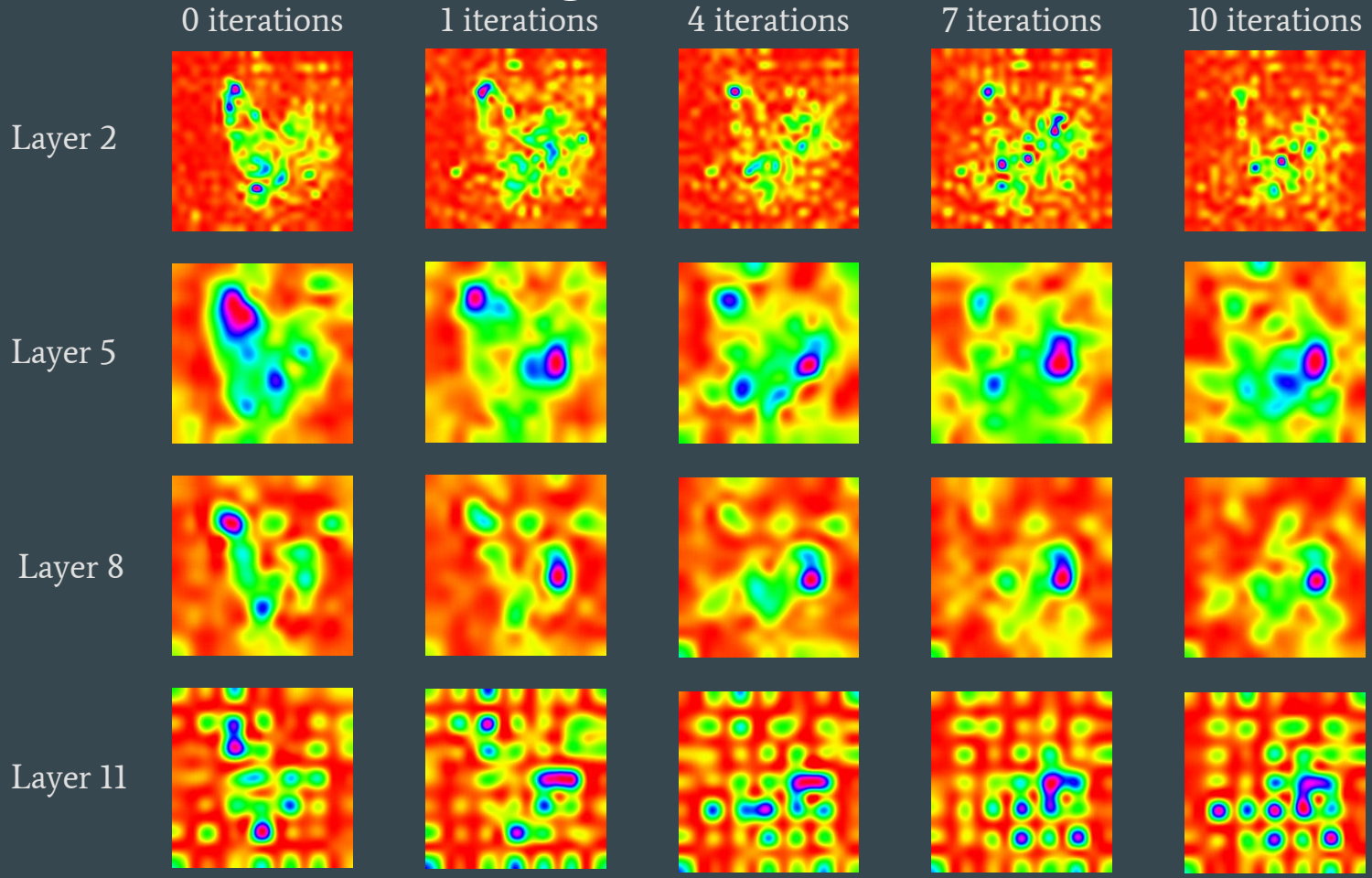
$$M^c = \text{ReLU}\left(\sum_k \hat{A}^k \right)$$

- LayerCAM Source Code

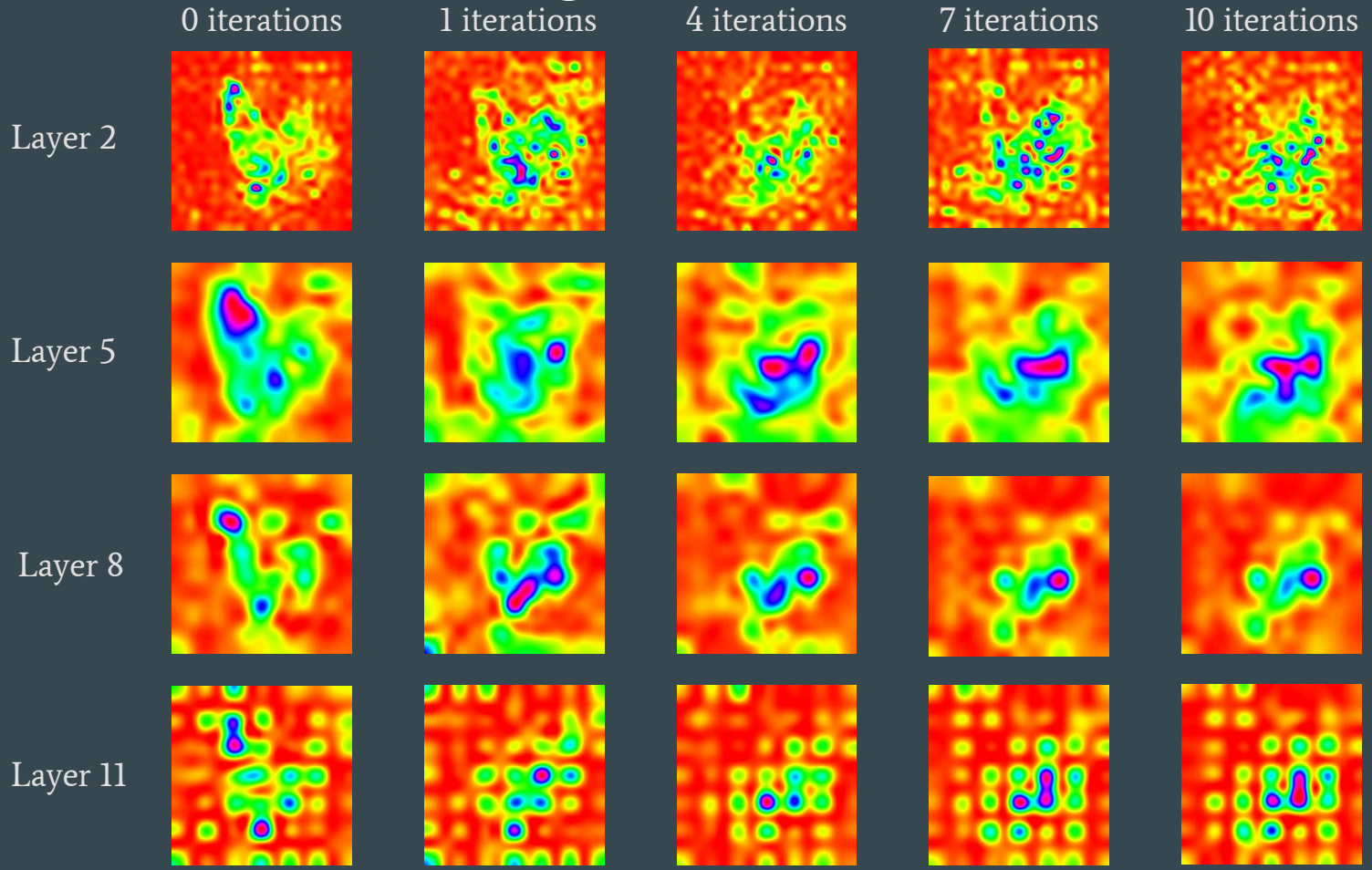
LayerCAM - Methods

- Generated visualizations for a normal image along with 4 increasingly perturbed images
  - Generate perturbed image with 1, 4, 7, and 10 iterations (corresponding to easy, medium, difficult, and very difficult classification task outlined above)
  - Visualize class activation heatmaps for original (hen) class and perturbed (kingsnake) classes
  - AlexNet: layers 2, 5, 9, 11
- The effect of the increase in perturbation is easily seen - increasing perturbation results in less activation of the original class and more activation of the perturbed class

LayerCAM - Increasing Perturbation - Hen Activations



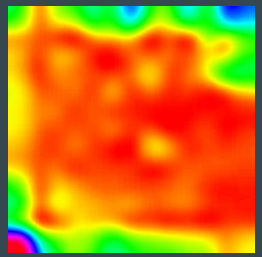
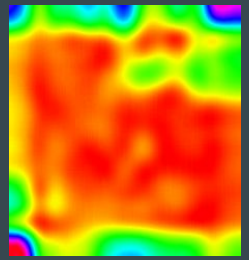
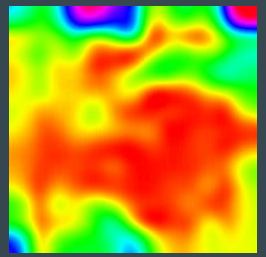
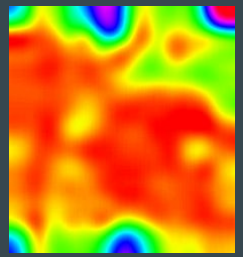
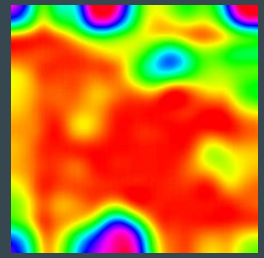
LayerCAM - Increasing Perturbation - Perturbed Activations



SCORE CAM - HEAT MAP

- The method known as Class Activation Mapping (CAM) uses a linearly weighted combination of the activation maps from the final convolutional layer before the global pooling layer to find discriminative areas.
- By determining each activation map's weight based on its forward passing score on the target class, Score-CAM eliminates the reliance on gradients.
- Score - CAM evaluates how well vision and localization function and results in better performance for both activities. Consequently, it acts as a debugging tool to look at model flaws.
- Score - CAM source code

Score Cam

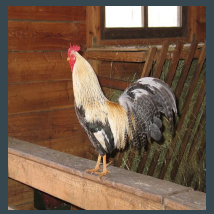
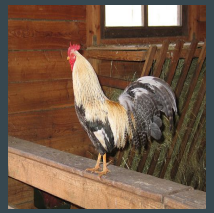
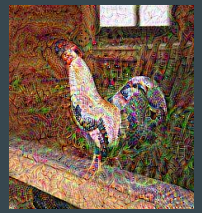
Perturbation	0 iterations	1 iterations	4 iterations	7 iterations	10 iterations	Classification
AlexNet						King Snake

- We provide the hen picture to ClassSpecificImageGenerator and the target class as King Snake in order to build an adversarial image. Then we iterate to produce an adversarial picture, to be specific we iterated the model 1, 4, 7, and 10 times.
- The heat map for the 0th and 1st iterations is essentially the same, as seen in the figures above, thus it's not surprising that this won't be misclassified as it's just one iteration.
- However, if we further perturb our images to iterations 4, 7, and 10, we can see that the noise becomes more pronounced and the pinkish-blue gradient becomes more visible in the image, thus it is not surprising that perturbed images are misclassified as a result of this noise.

Smooth Grad

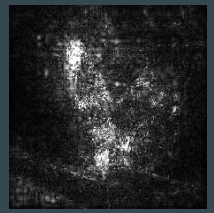
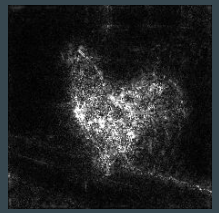
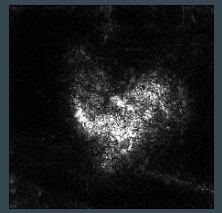
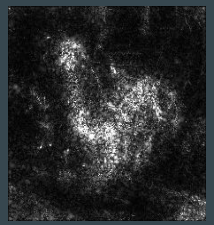
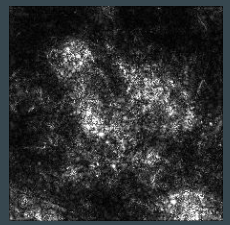
- Smooth grad is adding Gaussian Noise to the original image and calculating gradients multiple times and averaging the results.
- We used Vanilla backpropagation & Guided propagation to generate Smooth Grad images.
- Extension over Vanilla backpropagation & Guided backpropagation.
- For each image we have used AlexNet and VGG16 pretrained model.
- Next we will look at the output of our Smooth grad visualization of adversarial images.
- Score-CAM performs better than earlier works at precisely locating both a single object and multiple objects of the same class. It works by giving heat maps based on linearly weighted activation maps giving better visualization of the object present in the image.
- Smooth Grad Source Code

Smooth Grad - Adversarial images

	0 iterations	1 iterations	4 iterations	7 iterations	10 iterations	Classification
AlexNet						King Snake
VGG16						King Snake

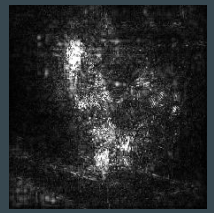
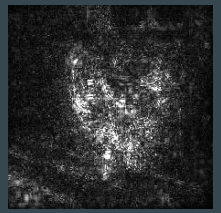
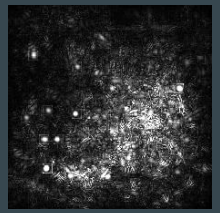
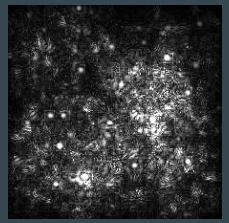
- To generate adversarial image we pass the hen image to ClassSpecificImageGenerator and pass the target class as King Snake. Number of iterations model takes to generate adversarial image is a variable.
- We ran the model 1, 4, 7, 10 times to generate adversarial images.
- Iteration 1 image would be more close to original image compared to iteration 10 image.
- Chances of Iteration 10 image getting mis-classified is higher compared to iteration 1 image.

Smooth Grad over Vanilla Backpropagation (VGG16)

	0 iterations	1 iterations	4 iterations	7 iterations	10 iterations	Classification
VGG16						King Snake

- Increasing the number of iterations causes more distortion to the generated smooth grad image. The Smooth grad image for iteration 10 is completely distorted compared to iteration 1.
- This distortion causes the Classifier to mis-classify the image for higher iterations.
- For iteration 1 & 4, the model was able to classify the image correctly as "Hen".

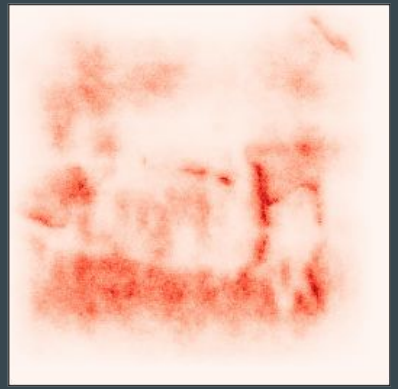
Smooth Grad over Vanilla Backpropagation (AlexNet)

	0 iterations	1 iterations	4 iterations	7 iterations	10 iterations	Classification
AlexNet						King Snake

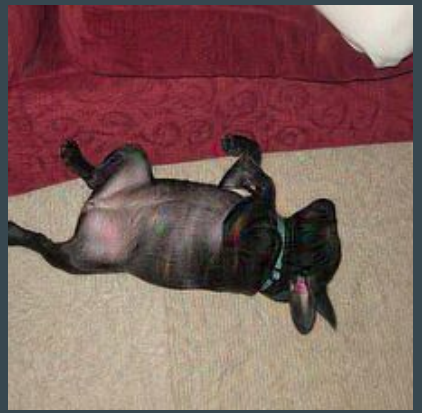
- Increasing the number of iterations causes more distortion to the generated smooth grad image. The Smooth grad image for iteration 10 is completely distorted compared to iteration 1.
- This distortion causes the Classifier to mis-classify the image for higher iterations.
- Compared to VGG16, AlexNet visualizations add noises around the main object. We can compare the iteration 1 images for VGG16 and AlexNet. AlexNet seems to have more noise than the VGG16 image.
- In case of AlexNet, only iteration 1 image was classified correctly as "hen". Reason being additional noises around the object.

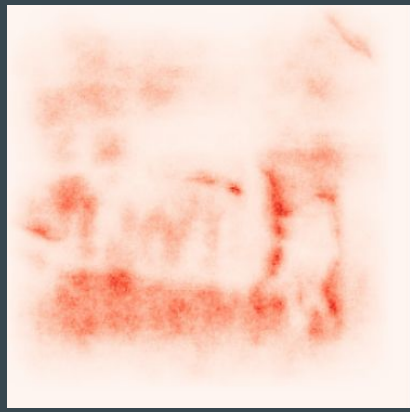
Exploratory Work (methods and visualizations from the Workshop and Demo)

Visualizing adversarial Dog images with Saliency



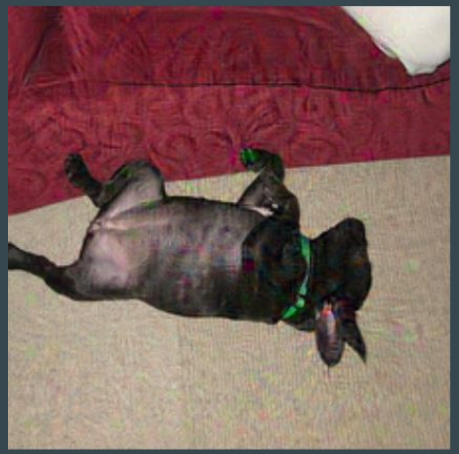
Dog classified as submarine

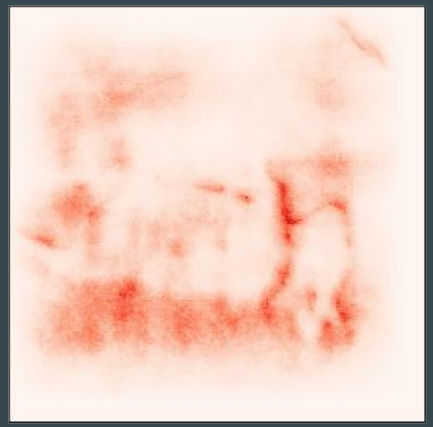




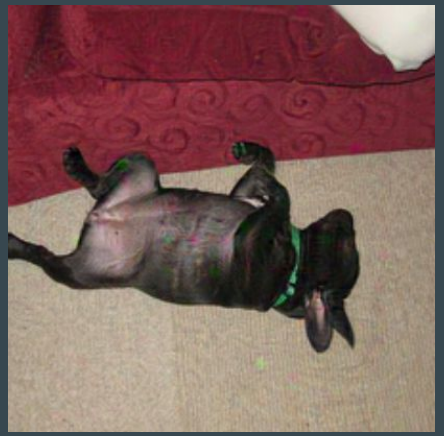
Dog classified as sandal

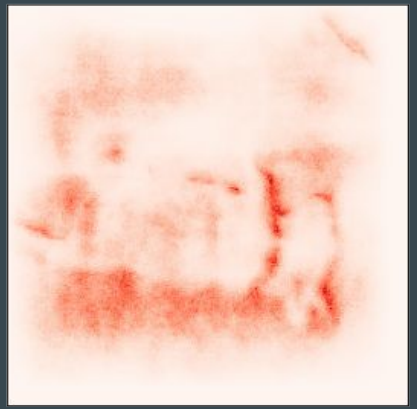
Visualizing adversarial Dog images with Saliency





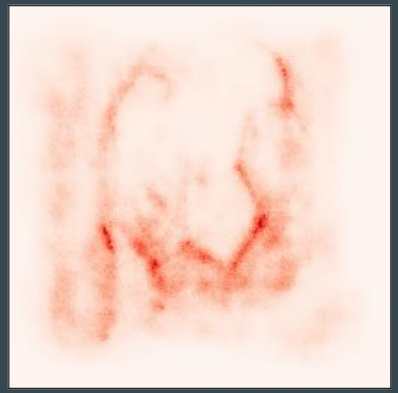
Dog classified as parachute





Dog classified as clock

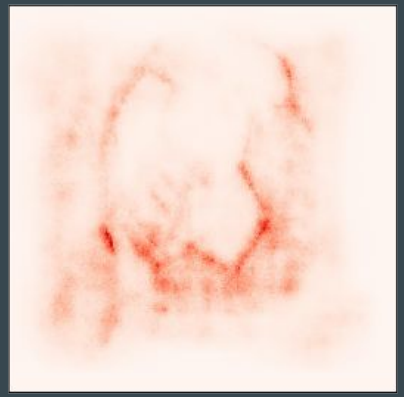
Visualizing adversarial Soccer Ball images with Saliency



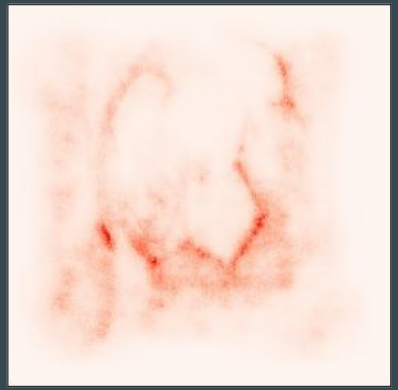
Soccer Ball classified as pretzel

Soccer Ball classified as desk

Visualizing adversarial Soccer Ball images with Saliency

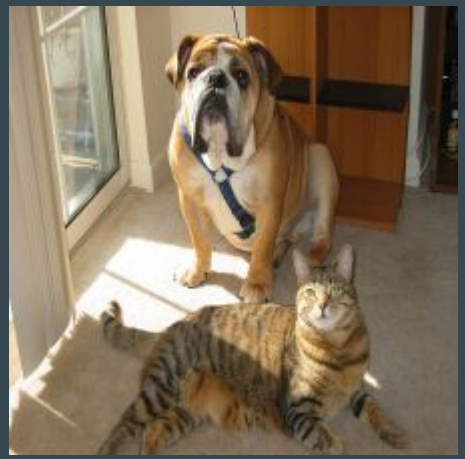


Soccer Ball classified as conch

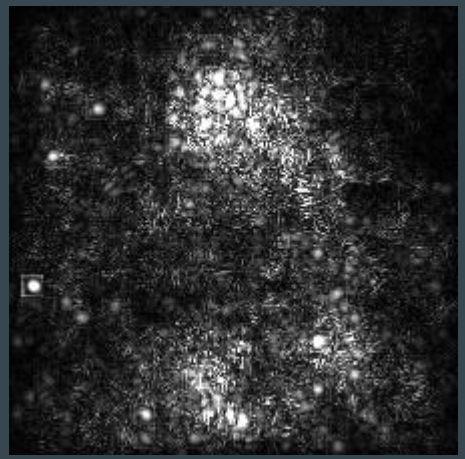


Soccer Ball classified as cleaver

Visualization on Dog/Cat Image classified as dog



Original Image

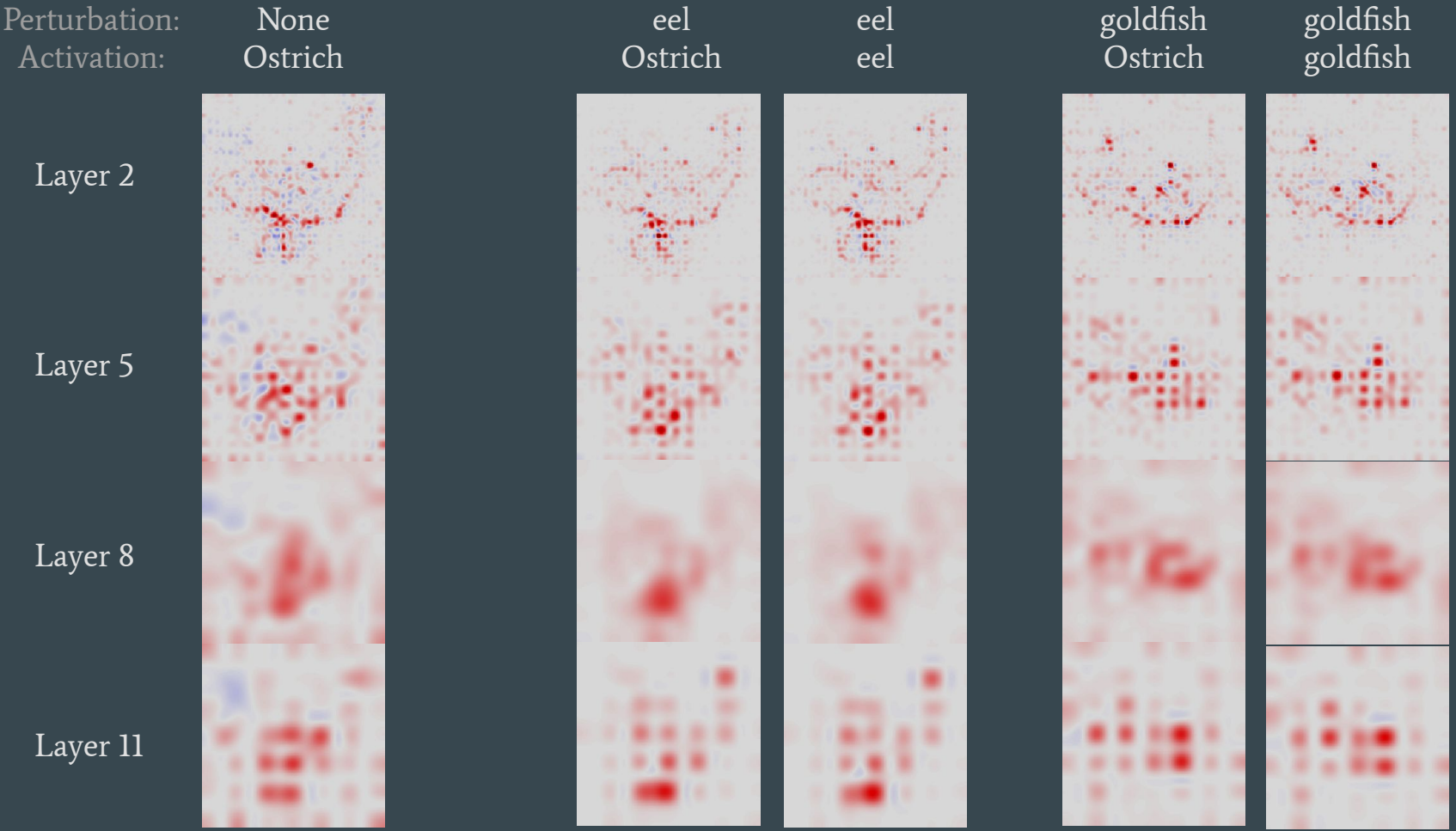


Vanilla Saliency

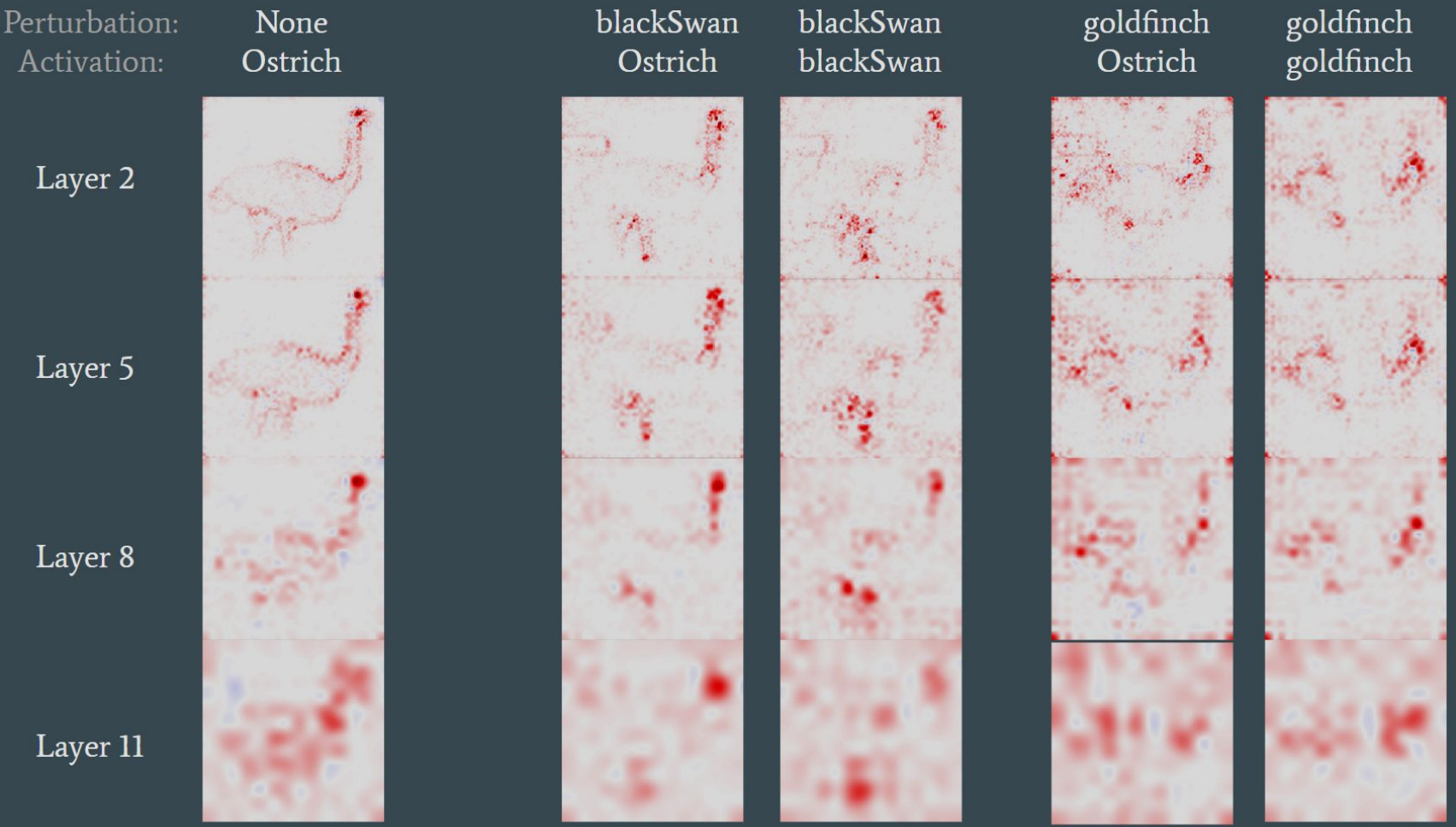
Backpropagation
Positive/Negative

LRP - Ostrich vs. Ostrich-blackSwan vs. Ostrich-goldfinch - AlexNet

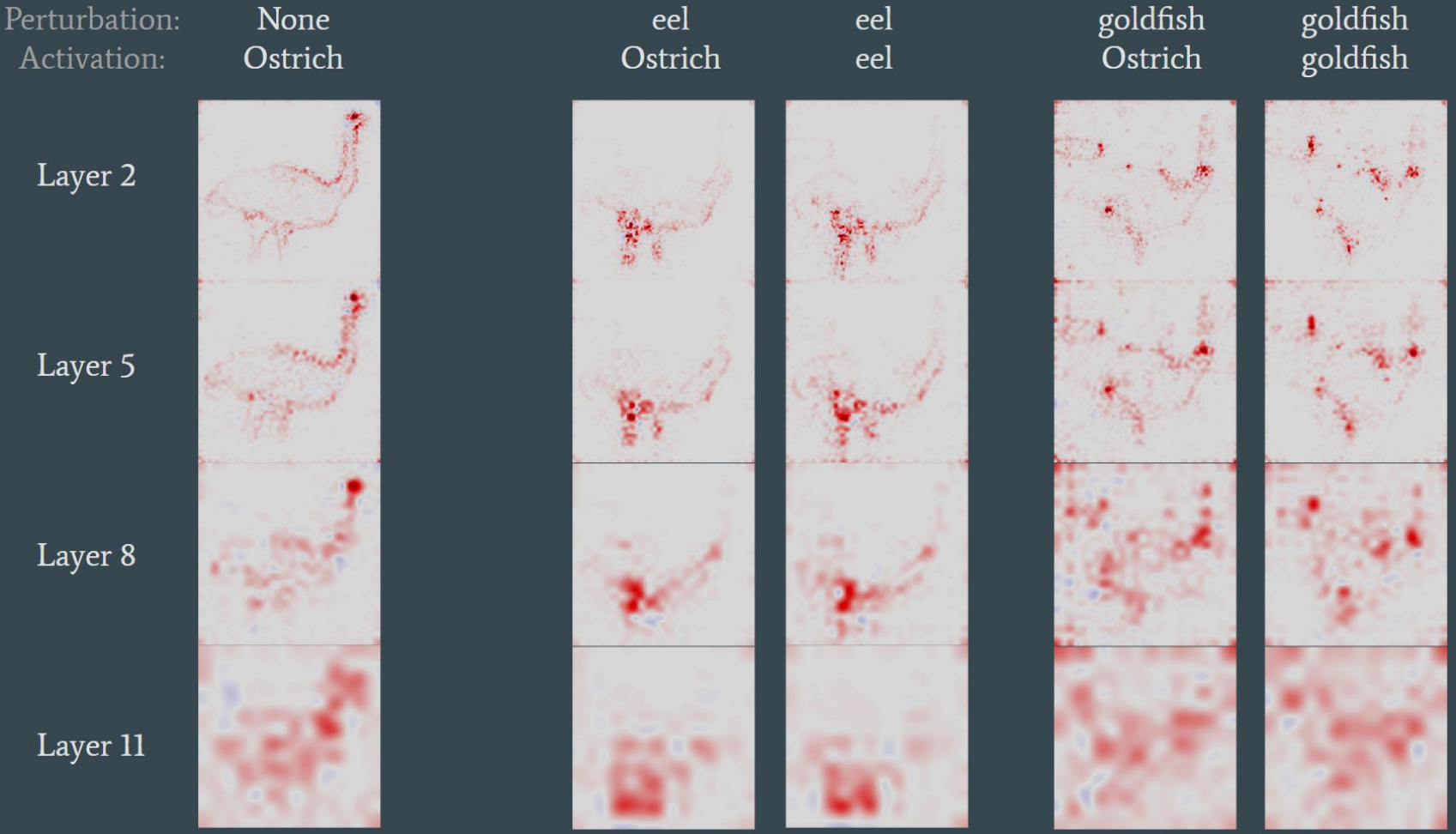
LRP - Ostrich vs. Ostrich-eel vs. Ostrich-goldfish - AlexNet



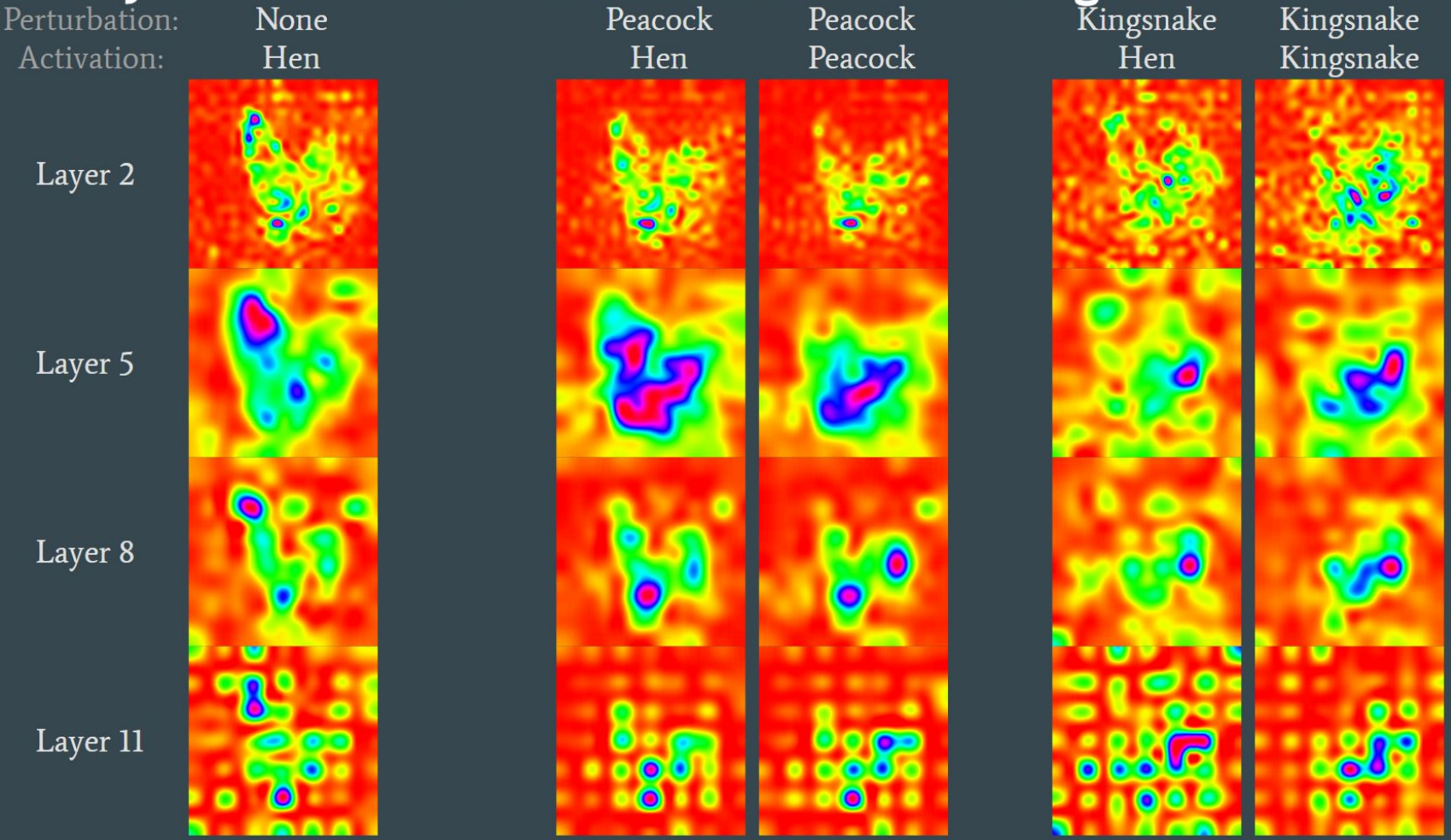
LRP - Ostrich vs. Ostrich-blackSwan vs. Ostrich-goldfinch - VGG16



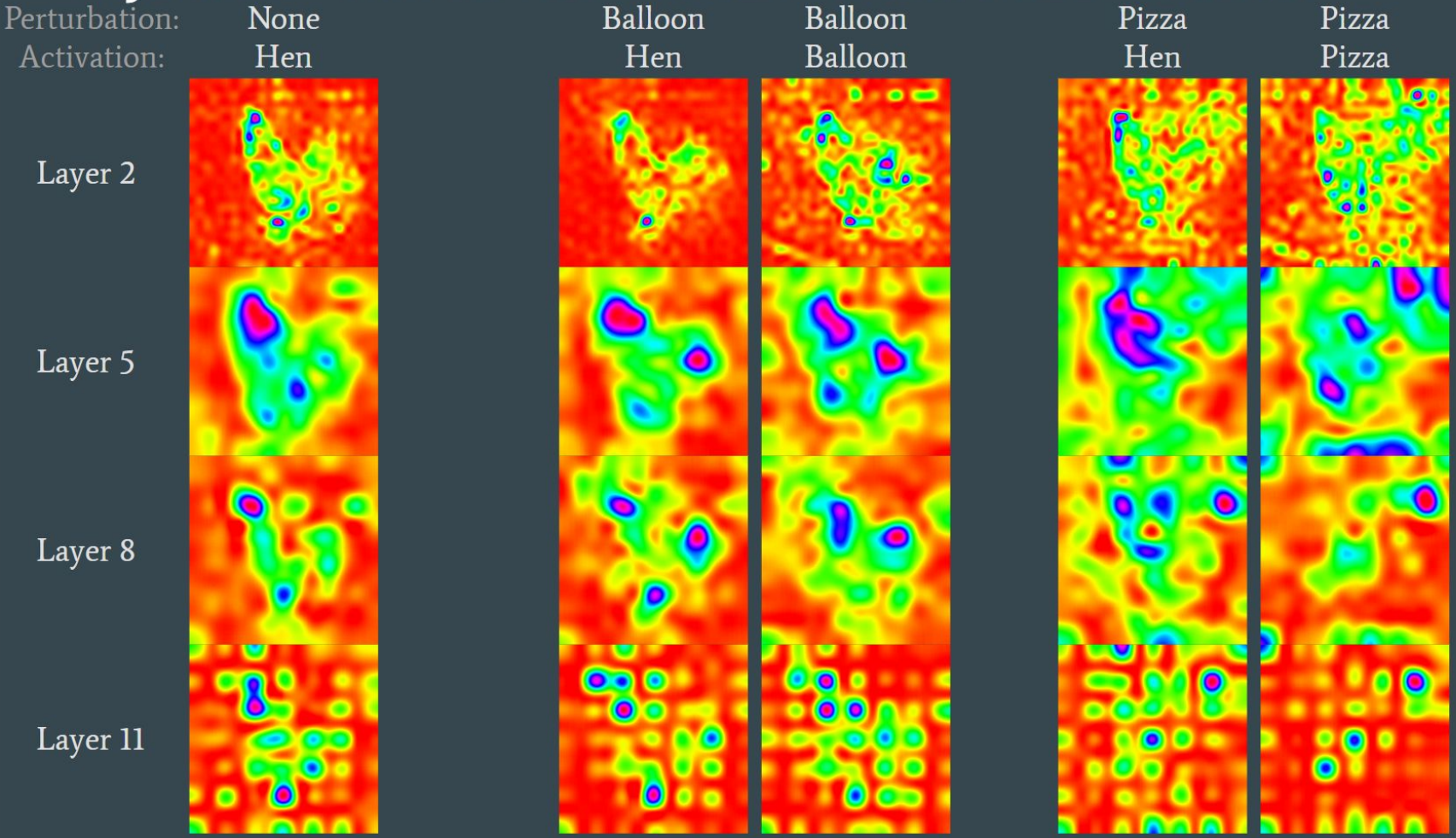
LRP - Ostrich vs. Ostrich-eel vs. Ostrich-goldfish - VGG16



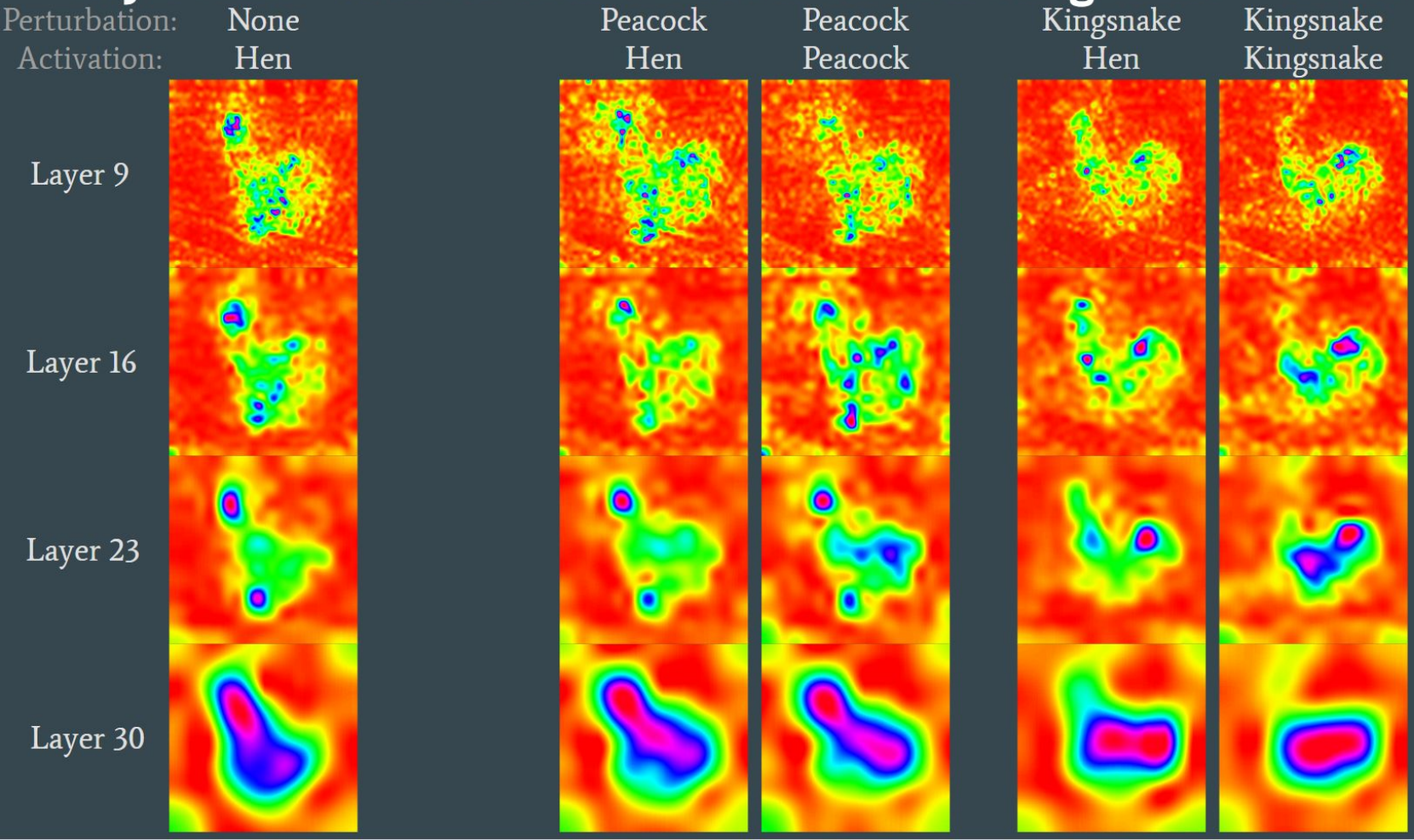
LayerCAM - Hen vs. Hen-Peacock vs. Hen-Kingsnake - AlexNet



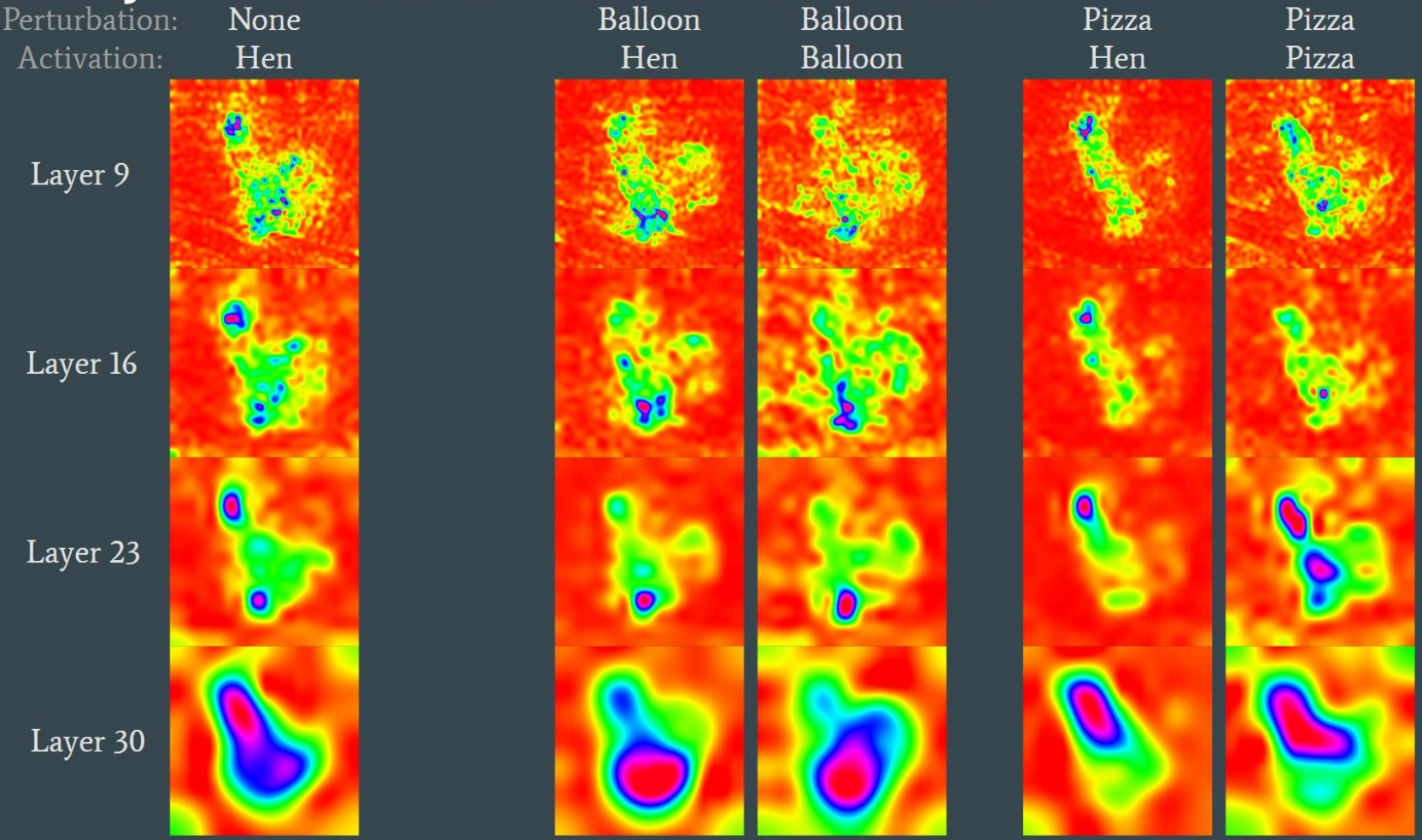
LayerCAM - Hen vs. Hen-Balloon vs. Hen-Pizza - AlexNet



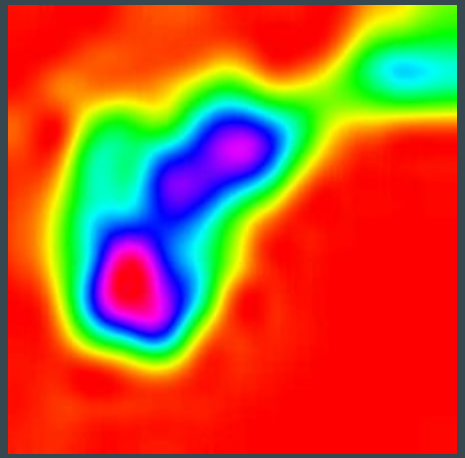
LayerCAM - Hen vs. Hen-Peacock vs. Hen-Kingsnake - VGG16



LayerCAM - Hen vs. Hen-Balloon vs. Hen-Pizza - VGG16



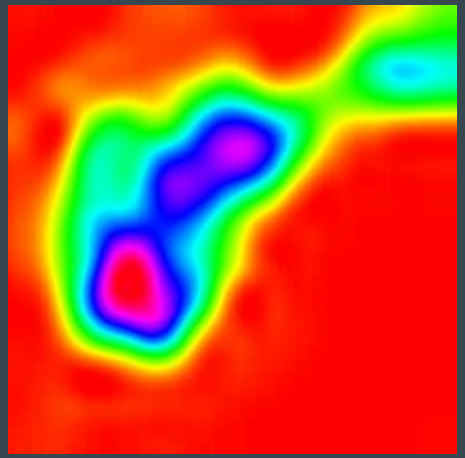
Layer Viz of Acoustic Guitar by AlexNet and VGG16



AlexNet

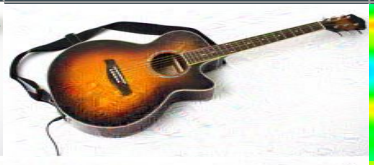
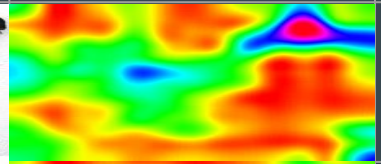
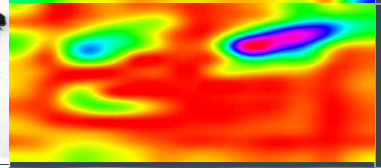
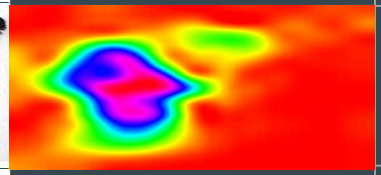
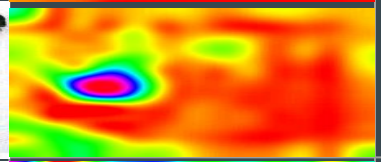
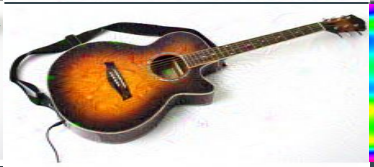
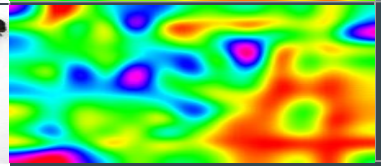
Acoustic

Guitar

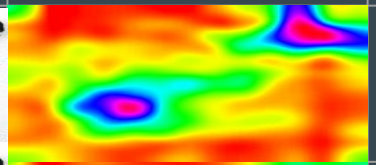
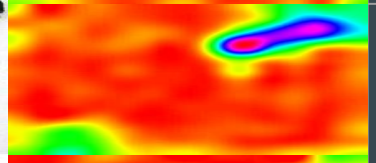
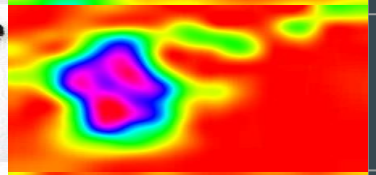
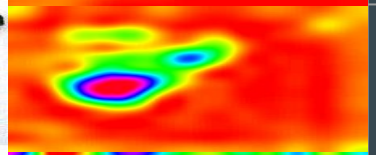
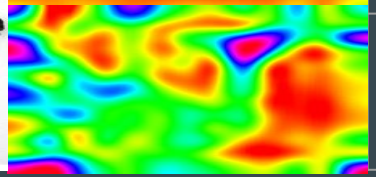


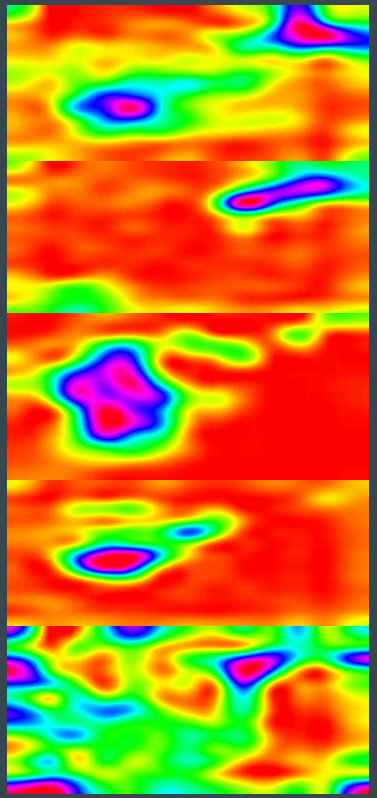
VGG 16

Adversarial Images - Score CAM

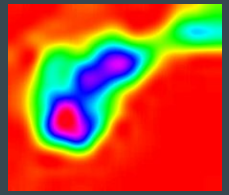
Original Image	Perturbed Image	AlexNet	Actual label	Classified as
			Acoustic Guitar	Sea Lion
			Acoustic Guitar	PeaCock
			Acoustic Guitar	Zebra
			Acoustic Guitar	Balloon
			Acoustic Guitar	Mushroom

Adversarial Images - Score CAM

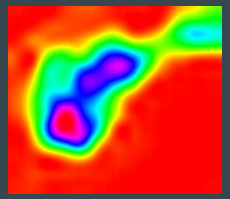
Original Image	Perturbed Image	VGG	Actual label	Classified as
			Acoustic Guitar	Sea Lion
			Acoustic Guitar	PeaCock
			Acoustic Guitar	Zebra
			Acoustic Guitar	Balloon
			Acoustic Guitar	Mushroom



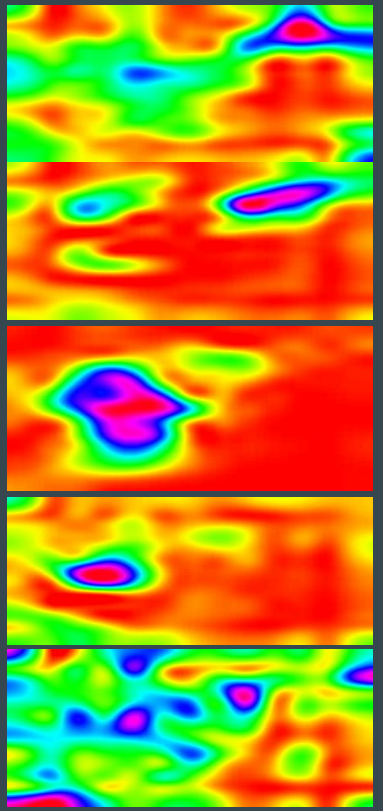
AlexNet



AlexNet



VGG 16

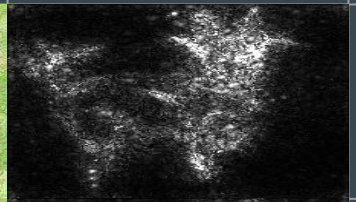
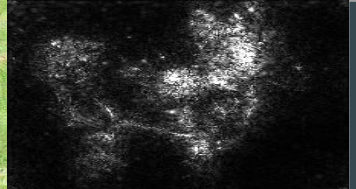
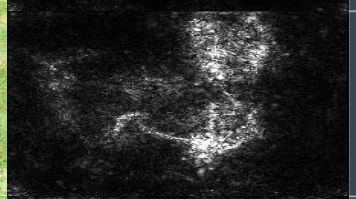


VGG 16

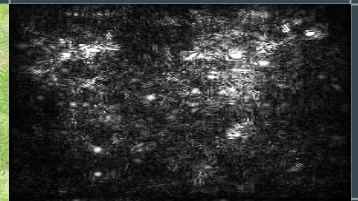
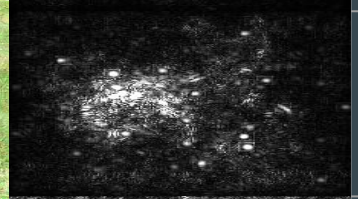
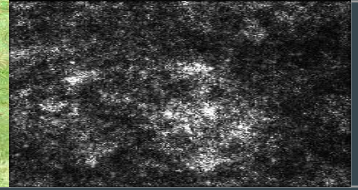
Adversarial Images - Smooth Grad Visualization

Original Image	AlexNet	VGG16	Actual label	Classified as
			Chihuahua	Hummingbird
			Chihuahua	Iguana
			Chihuahua	Leopard
			Chihuahua	Stingray

Smooth Grad over Vanilla Backpropagation (VGG16)

Original Image	Adversarial Image	Smooth grad image	Actual class	Classified as
			Chihuahua	HummingBird
			Chihuahua	Iguana
			Chihuahua	Leopard
			Chihuahua	Stingray

Smooth Grad over Vanilla Backpropagation (AlexNet)

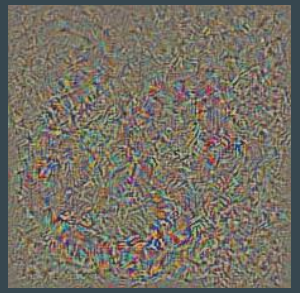
Original Image	Adversarial Image	Smooth grad image	Actual class	Classified as
			Chihuahua	HummingBird
			Chihuahua	Iguana
			Chihuahua	Leopard
			Chihuahua	Stingray

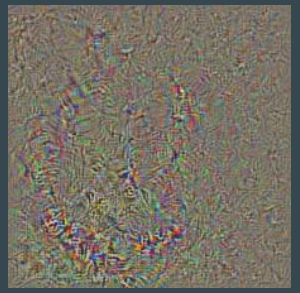
Inverted Image Representation

- With the ‘pure’ and ‘adversarial’ images, visualize the hidden layers
- ‘Understanding Deep Image Representations by Inverting Them’
  - Mahendran and Vedaldi, 2015
  - Reconstruct an input image using the output of a particular layer
  - Authors tuned parameters to their objective function in each layer to most accurately reconstruct the original image
- With tuning, the algorithm generates an image in the same ‘image space’ which indicates an image that the CNN sees as equivalent

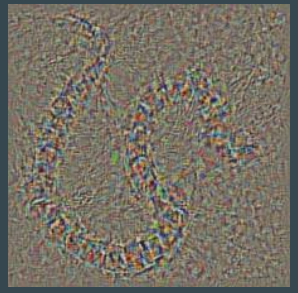
Snake vs. Snake-Spider (AlexNet)

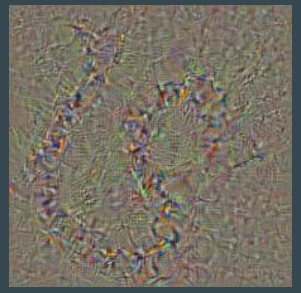
Snake

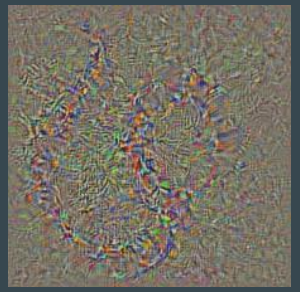


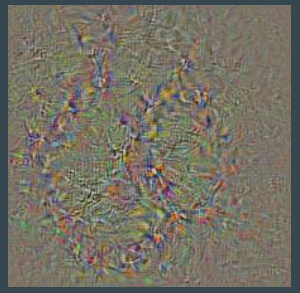


Snake-Spider





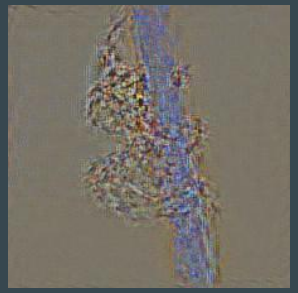
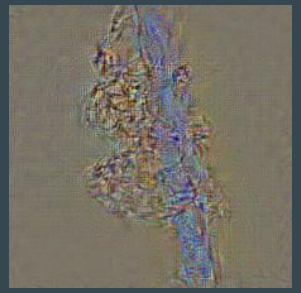
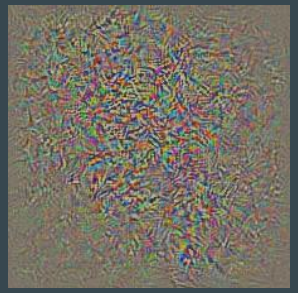
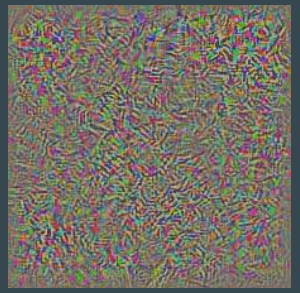




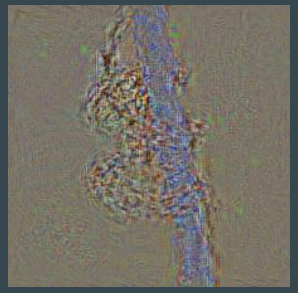
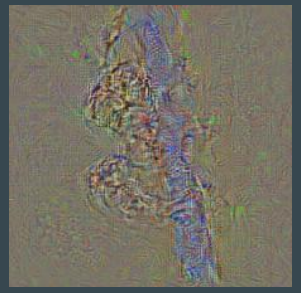
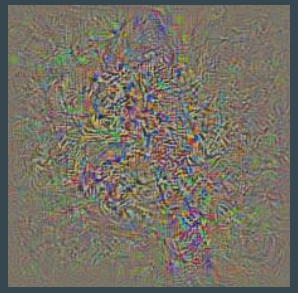
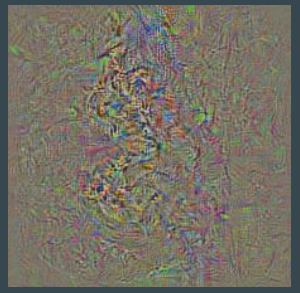
Layer 2 | Layer 5 | Layer 8 | Layer 11

Spider vs. Spider-Snake (AlexNet)

Spider

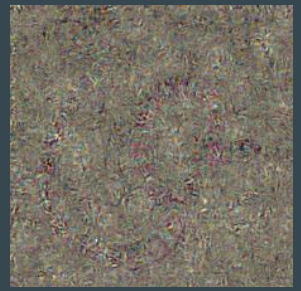
Spider-Snake

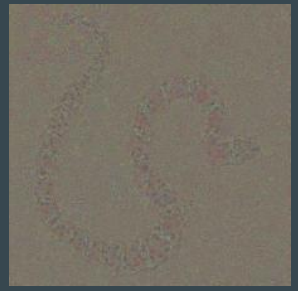
Layer 2 | Layer 5 | Layer 8 | Layer 11

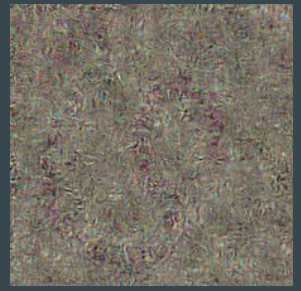
Snake vs. Snake-Spider (ResNet)

Snake



Snake-Spider



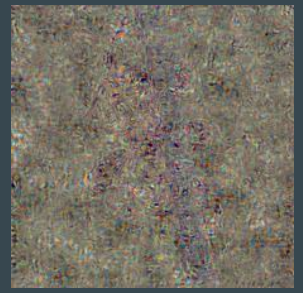


Layer 1 | Layer 2 | Layer 3 | Layer 4

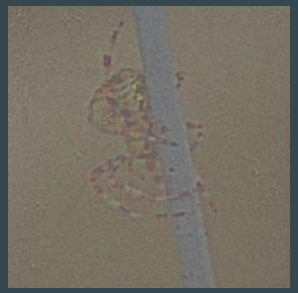
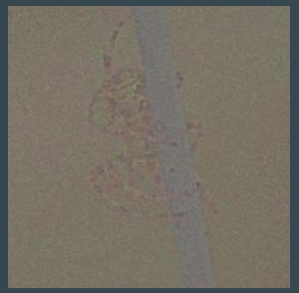
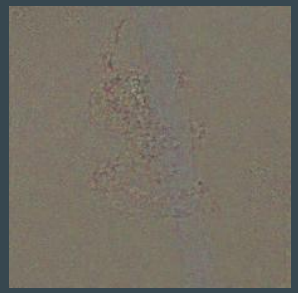
Spider vs. Spider-Snake (ResNet)

Spider



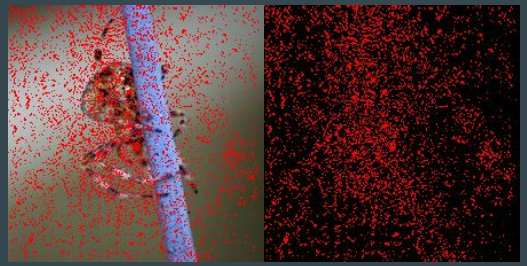
Spider-Snake


Layer 1 | Layer 2 | Layer 3 | Layer 4

Differential Analysis

- Inspired by word2vec NLP paper, in theory we can perform the operations:
  - tensor(spider_snake) - tensor(spider)
  - tensor(snake) - tensor(snake-spider)
- Future work can explore different difference approaches, but naive subtraction with thresholding yields the following:



- Future work can visualize these results and use them for rudimentary object detection

Conclusions and Takeaways

- We extended the visualization library to support:
  - Generation of adversarial examples by class
  - Using different neural network architectures
- Saliency Maps help visualize what pixels lead to classification decisions but it's not as robust as other approaches.
- Layerwise Relevance decomposed the input images to find relevant pixels in its layers that contributed to the particular classification.
- LayerCAM was better able to visualize class activations for shallow layers, and provided insights for the worse classification performance on adversarial inputs
- Smooth grad results shows that the adversarial image was able to trick the models even though the hidden layer visualization looks like the object. This shows that these model are trained strictly.
- ScoreCAM shows was able to incorporate idea of activation maps very well and how it can contribute towards concrete visualization compared to gradient based ones
- Repository Link

Future Work + Extensions

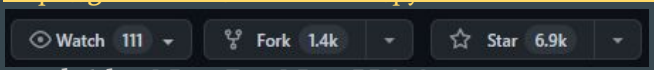
- Extend to more ML libraries and ImageNet classes
- Use findings to construct models with defenses against adversarial attacks
- CNNs can be applied to NLP tasks so visualizing the inner workings of NLP tasks
  - CNN's are used to pass through lines of text
- Incorporate quantitative measures for studied visualizations for more concreteness and objectivity
- Running ML models on our heatmaps to unlock more insights

GitHub Link

https://github.com/sohumgala/AdversarialCNNVisualizations

Contains ReadMe outlining the code deliverable

Open Source Code + Models + Algorithms

- Base CNN Visualization Library
  - https://github.com/utkuozbulak/pytorch-cnn-visualizations
  - 

- Pretrained AlexNet, ResNet, VGG
  - https://pytorch.org/hub/pytorch_vision_alexnet/
  - https://pytorch.org/hub/pytorch_vision_resnet/
  - https://pytorch.org/hub/pytorch_vision_vgg/
- ImageNet Dataset
  - https://www.image-net.org/
- Algorithms (Inverted Image, Gradient Saliency, Layerwise Relevance, Hierarchical Gradient)
  - Understanding Deep Image Representations by Inverting Them
  - LayerCAM: Exploring Hierarchical Class Activation Maps for Localization
  - Layer-Wise Relevance Propagation: An Overview
  - Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
  - Sanity Checks for Saliency Maps
  - Score-weighted class activation mapping