Deep Learning Practical Work 2-b

William KHIEU
Akli Mohamed Ait-Oumeziane

December 24, 2024

Contents

Section 1 - Saliency Map

1. Show and interpret the obtained results

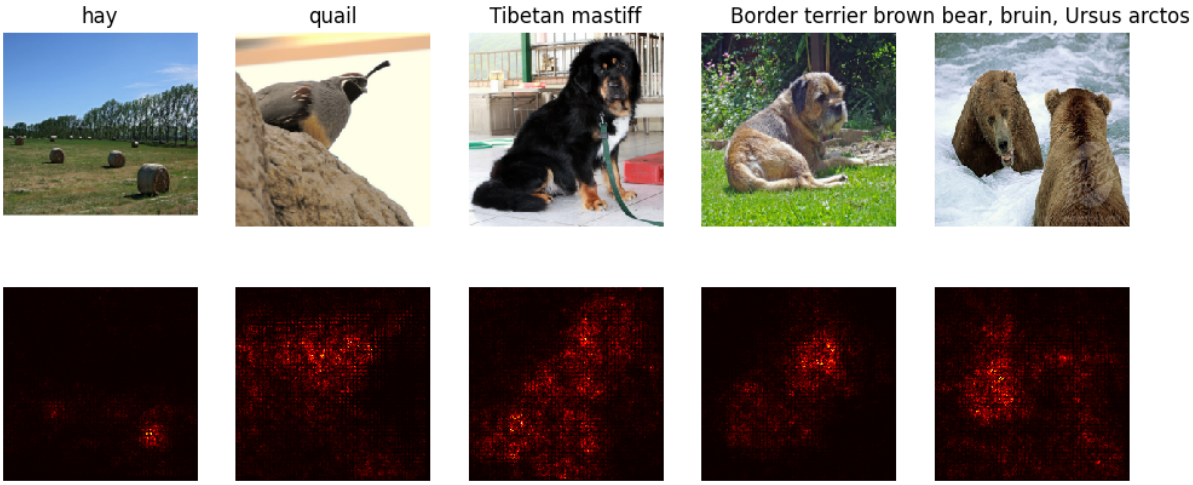


Figure 1: Example of saliency maps for different classes, SqueezeNet

When looking at these saliency maps, the bright regions indicate where the model's prediction is most sensitive to changes in pixel values. In other words, these highlighted areas tell you which parts of the image the network is relying on most strongly to identify each object.

From the examples:

- Hay field – The saliency map tends to highlight the bale shapes. This suggests the model associates these circular bale shapes with the "hay" class.
- Quail – The highest intensity is around the bird's head and body. The quail's distinctive shape and texture appear to be key features that the CNN uses for its classification.
- Tibetan mastiff – The model concentrates heavily on the dog's entire body.
- Border terrier – Similarly to the mastiff, the saliency map shows strong response around the dog's face and head shape. The model learns that these details are most indicative of a terrier.
- Brown bear – You see bright areas on the bear's head where the network focuses to confirm it is indeed a "brown bear."

2. Discuss the limits of this technique of visualizing the impact of different pixels.

The saliency map is a simple and effective way to visualize the impact of different pixels on the model's prediction. However, it has its limitations:

- **Local Interpretability:** The saliency map only provides local interpretability. It shows which pixels are important for the model's prediction but does not provide information on how these pixels interact with each other.
- **Linear Approximation:** The saliency map is based on the gradient of the output with respect to the input. This is a linear approximation of the model's behavior and

may not capture the full complexity of the model.

- **Noise and Sensitivity:** Raw gradient-based saliency maps can be noisy. Small perturbations to the input (like adversarial noise) or minor changes in the network (e.g., random seeds for initialization, data ordering, etc.) can lead to drastic changes in the saliency map.
- **Adversarial Examples:** The saliency map is not robust to adversarial examples. Small perturbations to the input can lead to significant changes in the saliency map.

3. Can this technique be used for a different purpose than interpreting the network?

Yes, saliency maps can be used for various purposes other than interpreting the network:

- **Model Debugging:** Saliency maps can help identify misclassifications and understand why the model made certain predictions. This can be useful for debugging and improving the model.
- **Data Augmentation:** Saliency maps can guide data augmentation strategies by highlighting the most important regions in the input images. This can help generate more diverse and informative training data.
- **Model Comparison:** Saliency maps can be used to compare different models or architectures. By visualizing the regions of interest, you can compare how different models focus on different parts of the input.

4. Test with a different network, for example VGG16, and comment.

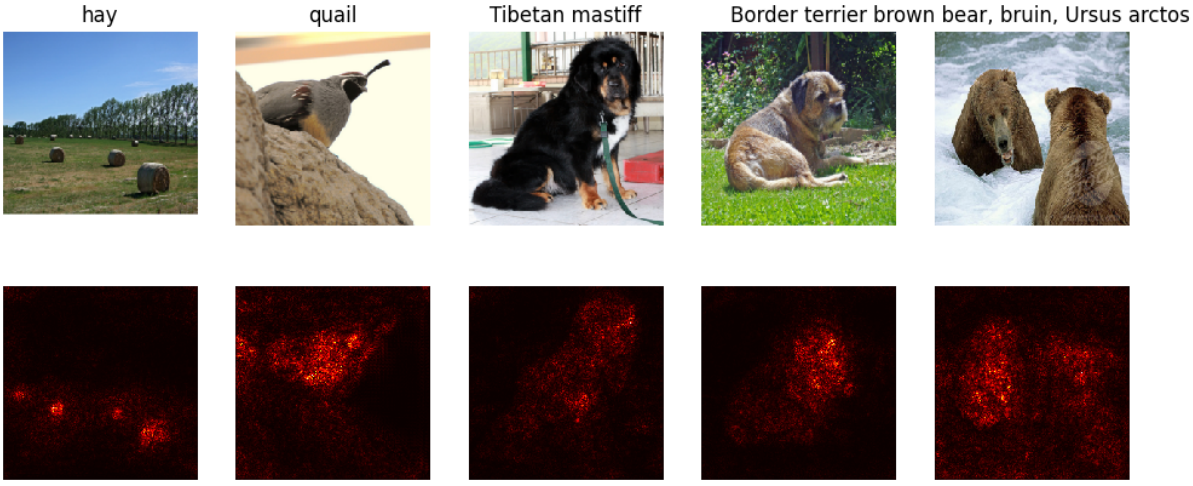


Figure 2: Example of saliency maps for different classes, VGG16

The saliency maps obtained with VGG16 show similar patterns to the ones obtained with SqueezeNet. The bright regions still indicate the areas of the image that the network relies on most strongly for classification. However, there are some differences in the saliency maps between the two networks:

- The VGG16 saliency maps appear to be more detailed and less noisy than the SqueezeNet saliency maps. This could be due to the deeper architecture of VGG16, which may capture more complex features and relationships in the data.

- Despite the extra detail, many of the bright regions overlap with what we saw using SqueezeNet. The bales of hay, the bird's body and head, the dogs' faces, and the bears' upper bodies are still highlighted—indicating these remain key discriminative features for both models.
- Overall, the saliency maps from VGG16 provide similar insights into the model's behavior as the SqueezeNet saliency maps. They highlight the key features and regions that the network uses for classification, helping to interpret and understand the model's decisions.

Section 2 - Adversarial Examples

5. Show and interpret the obtained results

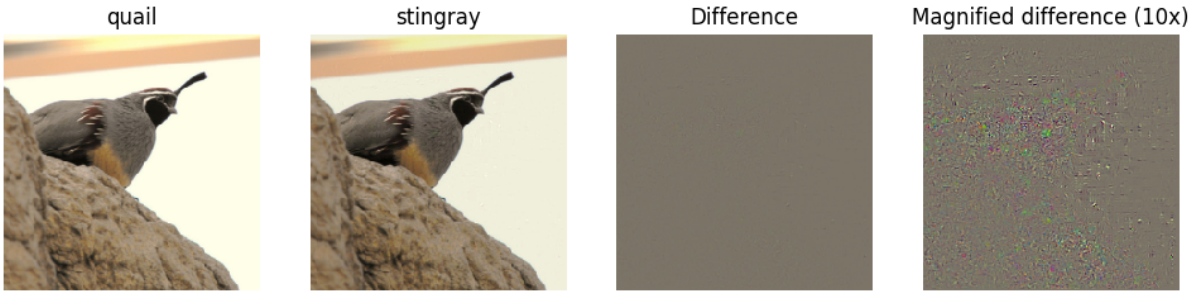


Figure 3: Example of adversarial examples

The leftmost image shows the original class ("quail"), while the second image shows the fooling image classified as "stingray." Both images appear nearly identical to the human eye, confirming that the modifications made to the image are minimal and imperceptible. Adversarial examples exploit tiny, targeted changes in pixel values that are sufficient to alter the network's prediction without noticeably affecting the image's appearance.

6. In practice, what consequences can this method have when using convolutional neural networks?

Adversarial examples can have several consequences when using CNNs:

- **Security Risks:** Adversarial examples pose security risks for CNNs. Attackers can exploit these vulnerabilities to deceive the model and cause misclassifications. For example, A stop sign slightly altered with imperceptible noise could be misclassified as a speed limit sign by a self-driving car.
- **Model Robustness:** Adversarial examples highlight the lack of robustness in CNNs. Small perturbations to the input can lead to significant changes in the model's predictions, undermining the reliability of the network.
- **Generalization:** Adversarial examples challenge the generalization capabilities of CNNs. The model may perform well on clean data but fail when presented with adversarial examples, indicating a lack of robustness and generalization.

7. Discuss the limits of this naive way to construct adversarial images. Can you propose some alternative or modified ways? (You can base these on recent research).

The naive way to construct adversarial images has several limitations:

- **Imperceptibility:** The naive approach may not guarantee that the adversarial perturbations are imperceptible to the human eye. The modifications may be noticeable, making the adversarial examples less effective in practice.
- **Transferability:** Adversarial examples generated using the naive approach may not

transfer well to other models or architectures. The perturbations may be specific to the target model and fail to deceive other networks.

- **Robustness:** The naive approach may not create adversarial examples that are robust to various transformations or perturbations. The adversarial perturbations may be fragile and easily removed by noise or transformations.

An alternative or modified ways to construct adversarial images is Fast Gradient Sign Method (FGSM):

Instead of iteratively updating x, FGSM uses a single-step update with a fixed perturbation size. The update rule is:

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x J(x, y_{\text{target}}))$$

where ϵ is a small constant and $J(x, y_{\text{target}})$ is the loss function for the target class.

- Advantages : Efficient and computationally faster than iterative methods.
- Limitations : Perturbations may still be perceptible if ϵ is too large.

Section 3 - Class Visualization

8. Show and interpret the obtained results

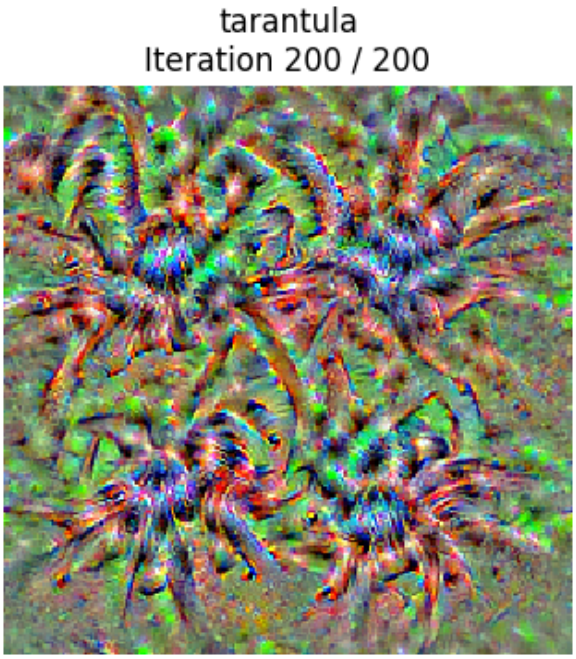
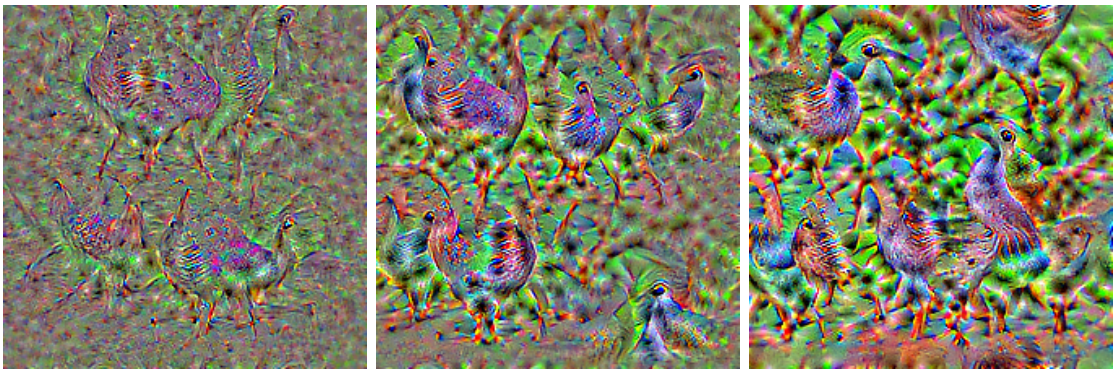


Figure 4: Class visualization for tarantula, SqueezeNet

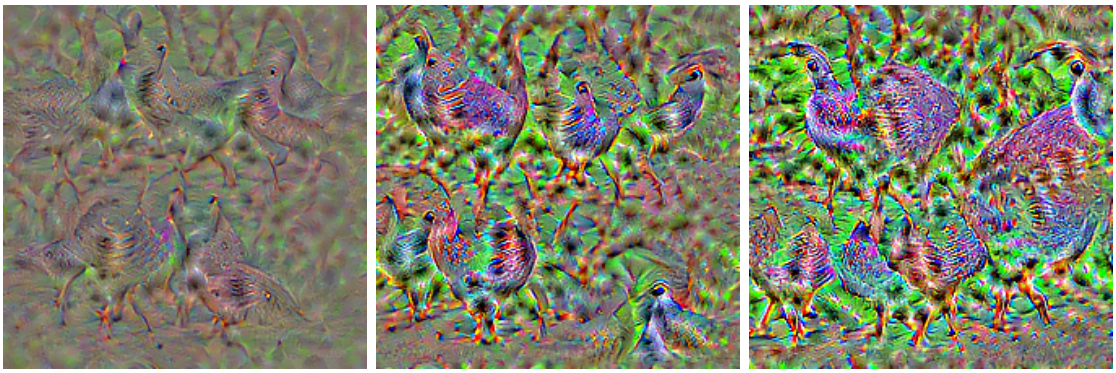
The class visualization for the tarantula class shows a spider-like pattern with eight legs and a round body. The visualization captures the key features and characteristics of a tarantula, such as the leg structure and body shape. The network has learned to associate these features with the tarantula class, resulting in a visualization that resembles a spider.

The figure appears as it does—featuring distorted, repeated tarantula-like patterns and a psychedelic color scheme—because the neural network focuses on amplifying specific discriminative features it has learned to associate with tarantulas, such as spiky legs, or round body, rather than maintaining global coherence or realism. This reflects the model's reliance on localized patterns rather than holistic perception, differing from how humans recognize objects. The repetition and distortion are artifacts of the optimization process, which prioritizes maximizing class neuron activation over realistic representation. Such visualizations highlight the network's internal representations, providing insights into its learning but also revealing potential biases or limitations.

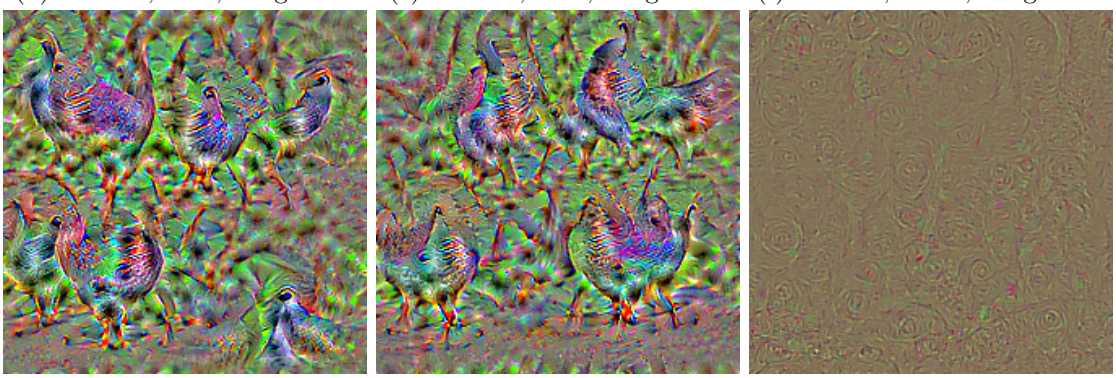
9. Try to vary the number of iterations and the learning rate as well as the regularization weight.



(a) 50 iter, lr=5, l2reg=1e-3 (b) 200 iter, lr=5, l2reg=1e-3 (c) 1000 iter, lr=5, l2reg=1e-3



(d) 200 iter, lr=1, l2reg=1e-3 (e) 200 iter, lr=5, l2reg=1e-3 (f) 200 iter, lr=10, l2reg=1e-3



(g) 200 iter, lr=5, l2reg=1e-3 (h) 200 iter, lr=5, l2reg=1e-1 (i) 200 iter, lr=5, l2reg=10

Figure 5: Class Visualization Images

10. Try to use an image from ImageNet as the source image instead of a random image (parameter init_img). You can use the real class as the target class. Comment on the interest of doing this.

Initiating the process with an image from ImageNet is a good approach since tt provides a good starting point for gradient-based visualization, ensuring that the initial gradients are directed towards relevant features and patterns within the image. It prevents the initial gradients from scattering or diverging in various directions, potentially resulting in a more focused and interpretable visualization.

11. Test with another network, VGG16, for example, and comment on the results.

Visualizations are superior due to VGG16's overall improved performance. Since VGG16 is a deeper network it generates more intricate patterns. We can see that there are more detailed

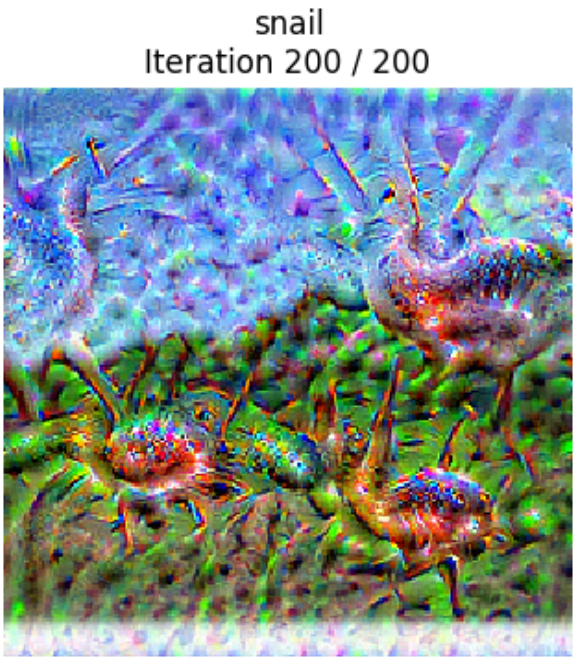


Figure 6: Class visualization for snail using ImageNet image, starting from hay field, SqueezeNet

textures and spatial details. The visualization is more complex and detailed, capturing finer features and patterns that the network associates with the target class. This suggests that the deeper architecture of VGG16 allows it to learn more complex and abstract representations, resulting in more detailed and informative visualizations.

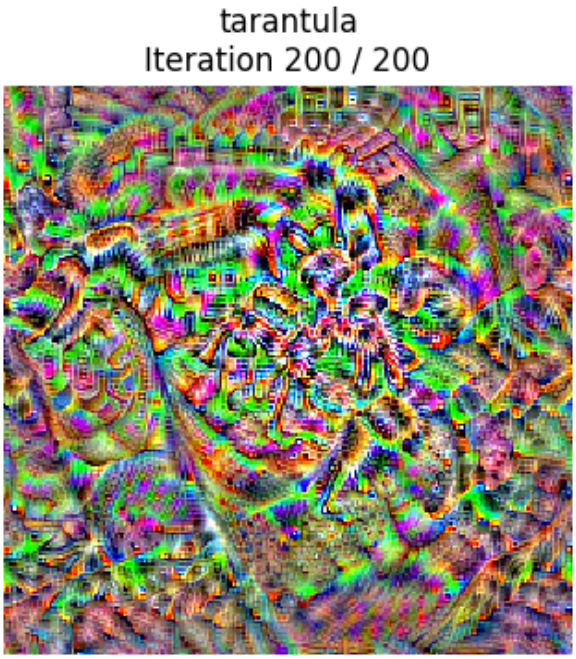


Figure 7: Class visualization for tarantula, VGG16

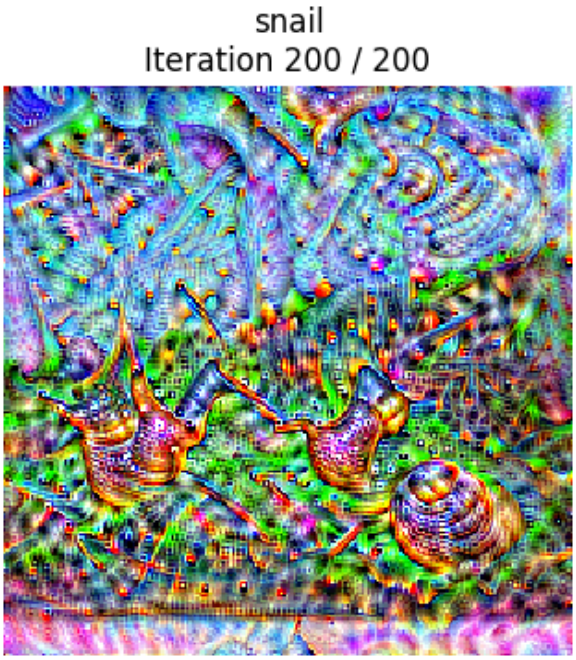


Figure 8: Class visualization for snail using ImageNet image, starting from hay field, VGG16