2-b Visualizing Neural Networks

ruyi tang

November 2024

Neural network visualization involves techniques to analyze the internal mechanisms of neural networks (e.g., weights, activations, and gradients) to interpret their behavior.The objective is to : understand how the network processes input features, identify the regions the network focuses on during decision-making and assess the model's robustness and performance (e.g., adversarial examples).

This section aims to study techniques for analyzing the behavior of convolutional neural networks (CNNs) by computing the gradient of the input image with respect to the output of a specific class.

We use the **pre-trained SqueezeNet model** (trained on ImageNet), freeze its weights, and modify the input image to indirectly explore the network's behavior.Gradients and image generation techniques are used to observe how the model reacts to different features. And in this section we introduce : **Saliency Map**, **Adversarial Examples**, and **Visualization of Classes**.

1 Saliency Map

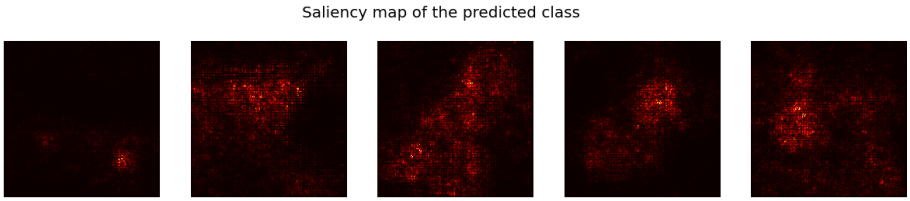
This method is proposed by Simonyan et al.(2014). It focuses on the visualization of the most impactful pixels for predicting the correct class. To produce the maps, it is proposed to approximate the neural network in the neighborhood of image.

Q1 Show and interpret the obtained results.

To better demonstrate and compare the performances of the model with different results of predictions, we show the following saliency maps with the true class and the predicted class.

(a) Original images with probabilities of true label and predicted class

(b) Saliency maps highlighting relevant regions of true class



(c) Saliency maps highlighting relevant regions of predicted class

Figure 1: Visualization of saliency maps using pre-trained SqueezeNet

Generally the saliency maps are consistent with the features shown in the original images. Except for the fourth image with lower accuracy of identifying the true label and making correct prediction, the rest images are shown with high accuracy of prediction, meaning that the model correctly highlights the important pixels/features in the saliency maps, making it easy to recognize the images.

And as for the exception of the fourth image, the model fails to predict "Border Terrier" correctly, but identify it as "Leonberg", which is also under the species of dogs. So in this case, the model fails to focus on the different facial features and outlines of the object with the highlighted area.

Q2 Discuss the limits of this technique of visualizing the impact of different pixels.

From the previous results, the saliency maps seem to work well as a highlighter to visualize which part of the picture influences the model's decision. However we also observe that the model may be confused and unable to identify or distinguish, which leads to questions about the liability of saliency maps.

The limits of This Technique can be concluded as the following:

1. **Lack of Robustness:** The reliance on arbitrary reference points (e.g., black images in IG) can lead to non-invariant results, making the attributions sensitive to input shifts or noise.
2. **Susceptibility to Manipulation:** Saliency maps can be easily altered using imperceptible adversarial changes, reducing their trustworthiness as diagnostic tools.
3. **Incomplete Interpretations:** Saliency methods often fail to provide a holistic understanding of complex interactions among input features, limiting their utility for debugging or decision justification.

And in the paper by Kindermans et al.(2017), the researchers evaluate the robustness and interpretability of saliency maps, highlighting two main findings:

- **Input Invariance**: Many saliency methods, like Integrated Gradients (IG) and Deep Taylor Decomposition (DTD), fail to maintain consistency under input transformations like constant shifts. For instance, the attributions depend on the choice of reference points, which are arbitrary and can lead to inconsistent results.
- **Adversarial Sensitivity**: Saliency maps are vulnerable to adversarial perturbations, which can drastically alter visual explanations without changing model predictions. This undermines their reliability for understanding model decisions.

These findings emphasize the need for developing more robust and interpretable saliency methods that are invariant to input shifts and resistant to adversarial attacks.

Q3 Can this technique be used for a different purpose than interpreting the network?

Saliency maps can also be used for other purposes beyond network interpretation.

1. **Data Augmentation:** From Simonyan et al.(2014), by identifying the most critical regions for a task, saliency maps guide targeted data augmentation. For instance, focusing transformations like cropping or rotation on these regions preserves relevant features, enhancing model training efficiency.
2. **Style Transfer and Image Generation:** From Liu et al.(2019) Saliency maps assist in style transfer tasks by ensuring the content of significant regions is preserved while applying the style of another image. This improves the quality and meaningfulness of generated images.

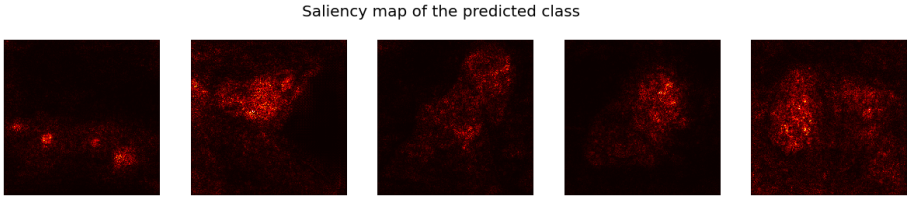
3. **Feature Selection:** From Shrikumar et al., 2017 (DeepLIFT)., in tasks involving tabular or structured data, saliency maps can help select the most informative features, reducing input dimensionality and making models faster and more interpretable.

These applications highlight the versatility of saliency maps, extending their utility into areas like generative models, and feature optimization.

Q4 Test with a different network, for example VGG16, and comment. To make comparison, we generate the saliency maps using pre-trained VGG16 on the same sample images.

(a) Original images with probabilities of true label and predicted class on VGG16

(b) Saliency maps highlighting relevant regions of true class on VGG16



(c) Saliency maps highlighting relevant regions of predicted class on VGG16

Figure 2: Visualization of saliency maps using pre-trained VGG16

From the results above, we can observe that the VGG16 makes no miss-classification with high accuracy. Therefore the saliency maps generated are demonstrated with less noise and clearer indicator to the importance pixels. For example, we can see that the first saliency maps for the "hay" are clearly highlighting the four hays of the corresponding image instead of just one in the precious saliency maps of SqueezeNet. We can also see that the corresponding maps for "quail" and "Tibetan mastiff" are also shown with less noise in the background and clearer outline of the object.

As for the previously missidentified "border terrier", the saliency maps of VGG16 shows more concentration on the facial features with lighter pixels, which contributes to the correct classification this time,

This result demonstrates the robust performance of VGG16 in image classfication with more reliable and consistent saliency maps. VGG16's ability to generate better saliency maps underscores the trade-off between model efficiency (as in SqueezeNet) and feature richness, suggesting that deeper, more expressive models are often better suited for interpretability tasks, while SqueezeNet is more suited for real-time, low-latency applications where efficiency is critical.

2 Adversarial Examples

In this section we analyze neural networks by generating adversarial examples (or fooling samples), which is to confuse the model with slightly modified images (imperceptible for humans) that lead to missclassfication. The idea is to deceive the model therefore to learn about the limitations and unusual behaviors of the neural network.

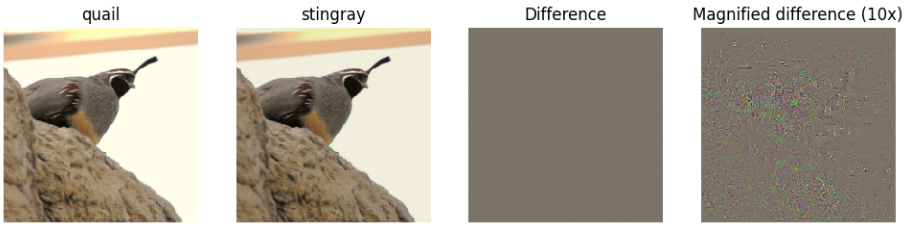
Q5 Show and interpret the obtained results.

To generate a fooling image that the model will classify as the target class we perform gradient back propagation on the score of the target class, stopping until the model is fooled.

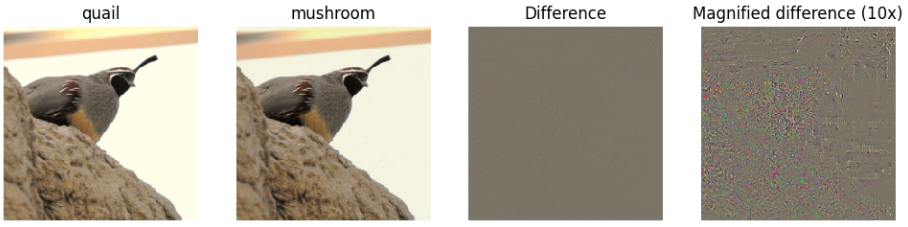
In the following Figure 3, we show the results with the left column presenting the original image of "quail", the second column showing the modified image missclassified as the target class, and the rest columns showing the differences and magnified differences between the original images and the modified images.

Even though it's hard to just visually distinguish the modified images from the original one, the neural network identifies them to be in an different class. And with magnification of the differences, the ones with "stingray" and "mushroom" are presented with more noises compared to the "bee eater".

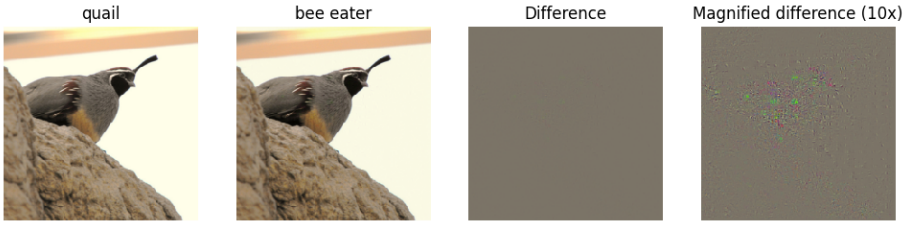
And we also observe from the results of iteration and scores, for the "stingray" and "mushroom", the model is fooled after 14 and 16 iterations, respectively achieving the max score of 59.641 and 52.659, compared with iteration of 6 and max score of 46.023 of the target class "bee eater". The higher scores and number of iterations means it is harder to fool the model with the more distinct targets. Compared with "bee eater" which is also a kind of bird as "quail", "stingray" and "mushroom" are of two entirely different species that results in more obvious differences.



(a) "quail" V.S. "stingray"



(b) "quail" V.S. "mushroom"



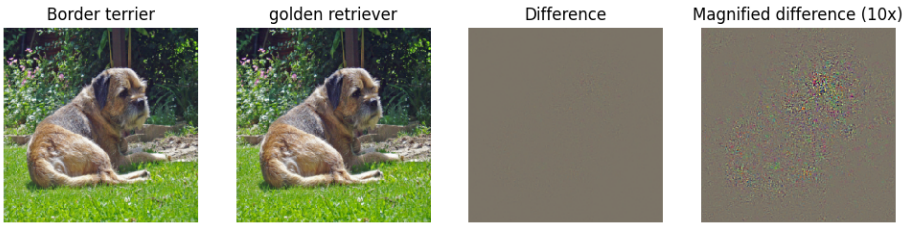
(c) "quail" V.S. "bee eater"

Figure 3: Adversarial examples of "quail" with different target classes

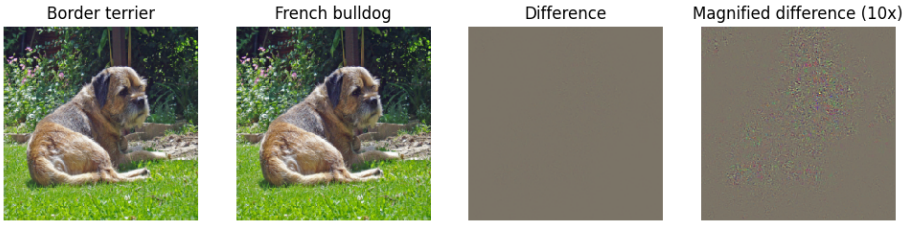
We also test with another input image of "border terrier".

In the following Figure 4, we can see that the Figure 4c has the maginified difference with most noises and the model needs 87 iterations reaches the max score of 74.994 to be fooled. It means that the image of "desk" is evidently distinct from the original "border terrier".

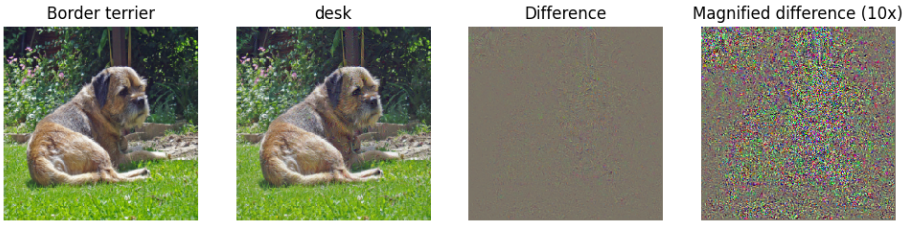
The results of Figure 4a and Figure 4b show that even the target classes "golden retriever" and "French bulldog" are under the same species as the original "border terrier" with smaller differences compared with the last one, the "French bulldog" share more similarities with the original image with less noise in the background of magnified difference, and needs less iteration with smaller max score to fool the model.



(a) "border terrier" V.S. "golden retriever"



(b) "border terrier" V.S. "french bulldog"



(c) "border terrier" V.S. "desk"

Figure 4: Adversarial examples of "border terrier" with different target classes

Q6 In practice, what consequences can this method have when using convolutional neural networks?

Adversarial examples expose critical vulnerabilities in CNNs, leading to significant practical consequences across various domains.

1. **Security risks**
   (a) **Model Vulnerability**: CNNs are highly susceptible to adversarial attacks, where imperceptible perturbations can mislead models to produce incorrect predictions. In the paper of Szegedy et al.(2014) ,this poses risks in high-stakes applications such as autonomous vehicles, facial recognition, and healthcare diagnostics. For example, Slight modifications to a stop sign image can cause a self-driving car to misclassify it as a speed limit sign, leading to dangerous outcomes.
   (b) **Malicious Exploitation**: According to Goodfellow et al.(2015) attackers can craft adversarial examples to bypass security sys-

tems, compromising authentication (e.g., facial ID) or financial fraud detection systems.

2. **Challenges in Interpretability**
   (a) **Unreliable Explanations**: In the paper of Ilyas et al.(2019), adversarial examples exploit features imperceptible to humans but influential to CNNs, complicating efforts to interpret predictions.
   (b) **Debugging Difficulty**: In the studies of Athalye et al.(2018), the presence of adversarial examples makes understanding and diagnosing CNN errors more challenging.
3. **Fairness Implications** Adversarial examples can disproportionately affect certain groups, exacerbating biases in sensitive applications like hiring or healthcare. Obermeyer et al.(2019) discusses how adversarial susceptibility can exacerbate biases in AI systems, especially in sensitive domains like healthcare. A healthcare algorithm could be manipulated that results in over-priority or neglect issues.

Q7 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).

Adversarial images constructed naively often lack robustness in real-world settings, as environmental variations (e.g., lighting, viewing angles, and noise) disrupt adversarial perturbations. To address the problem,Athalye et al.(2018) proposes the Expectation Over Transformation (EOT) method that explicitly models these variations by averaging over multiple transformations of the input during adversarial image construction. This ensures the adversarial examples remain effective under physical-world conditions.

To also solve the problem of poor robustness,in the research of Goodfellow et al., 2015. it uses adaptive adversarial training to dynamically generate adversarial examples during model training, ensuring the model learns to counter diverse and evolving perturbations. This reduces vulnerability to both known and unknown attacks.

Adversarial examples created naively are also often overfitted to a specific target model and fail to transfer across different architectures or parameters. To this end, Liu et al.(2017) proposes ensemble-based methods, which generate adversarial examples using multiple models simultaneously. By optimizing perturbations to work across various models, the method improves the transferability of attacks to black-box or unknown models.

3 Class Visualization

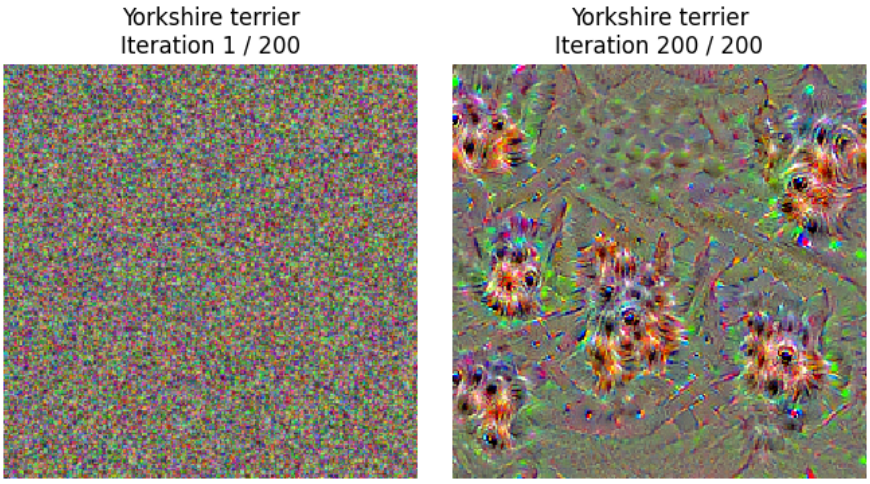
In this section we utilize the method proposed by Simonyan et al. (2014) and developed by Yosinski et al. (2015) to generate images that highlight the

type of patters likely to produce a particular class prediction, and so indirectly analyzing what a network prioritizes for classfication prediction.

Q8 Show and interpret the obtained results.

Class visualization is a technique used to understand what a neural network has learned about specific classes by generating synthetic images that maximize the activation of a given class. This approach reveals the patterns or features the network associates with that class, offering insights into how the network processes and recognizes different categories.

We start the experiment by visualizing Yorkshire Terrier from random noise. In the following Figure 5, we show a class visualization for a Yorkshire Terrier after 200 epochs and using default parameters with L2-regularization factor of 10^{-3}, learning rate at 5, bluring every 10 epochs.



(a) initial state (b) last iteration

Figure 5: Class visualization: started from random noise, maximizing the score from the Yorkshire Terrier class with default parameters.

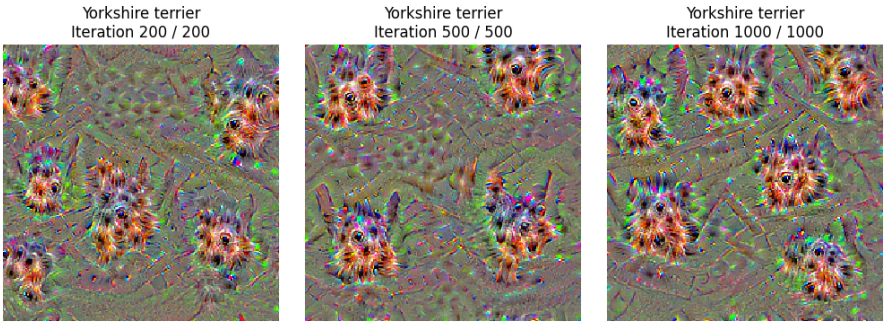
Q9 Try to vary the number of iterations and the learning rate as well as the regularization weight.

- **number of iteration:**

  In the following experiments, we are showing results from last iteration with number of iteration at 200,500 and 1000.

  Since the results from Figure 6 are generated from different random noises, so it ends up with different number and rotations of the object. But we can still observe from the results that with the increase of the number of iterations, the surrounding green points that highlight

the outline of the multiple objects shown on the figure are brighter. We can also see that the features (the facial features and the fur texture etc.) are more amplified for the network to activate the correct classification neurons in the network.



(a) iteration=200 (b) iteration=500 (c) iteration=1000

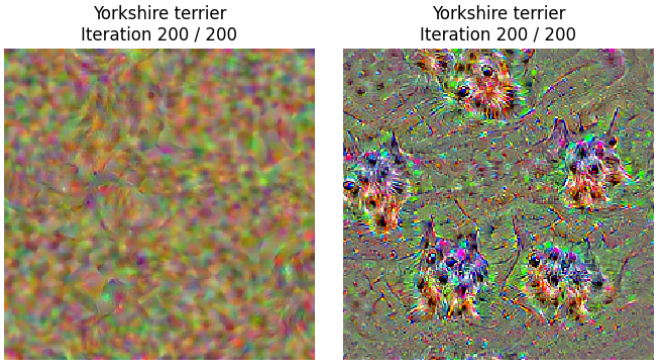
Figure 6: Class visualization of Yorkshire terrier with different number of iterations

- **Learning rate**

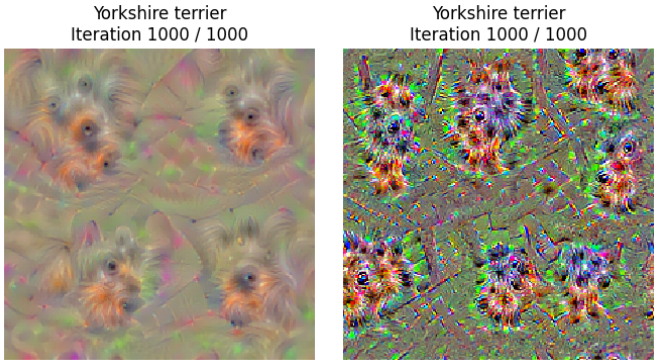
  Learning rate controls the step size during the optimization process, where the input image is iteratively updated to maximize the activation of a specific class in the neural network. It has significant impact on the quality and interpretability of the generated visualizations.

  In this part of experiments, we are testing with cases with learning rate as 0.1 and 10.

  From Figure 7, we can see that using the higher learning rate of 10, the generated figures are noisier, and combined with higher iterations the figure becomes harder to identify. But with lower learning rate of 0.1, the generated figure is relatively more smooth. But with smaller iterations, the features of the object are not produced. After compensating with higher iterations at 1000, we can see that with lower learning rate, the objects are shown with more perceptible features, meaning that the lower learning rate allows for more effective convergence and that it should keep the balance between the moderate learning rate and the iterations.



(a) lr=0.1, iter=200 (b) lr=10, iter=200



(c) lr=0.1, iter=1000 (d) lr=10, iter=1000

Figure 7: Comparison of Yorkshire terrier class visualization with different learning rates and with (a) and (b) at 200 iterations, (c) and (d) at 1000 iterations

- **L2-regularization parameter**

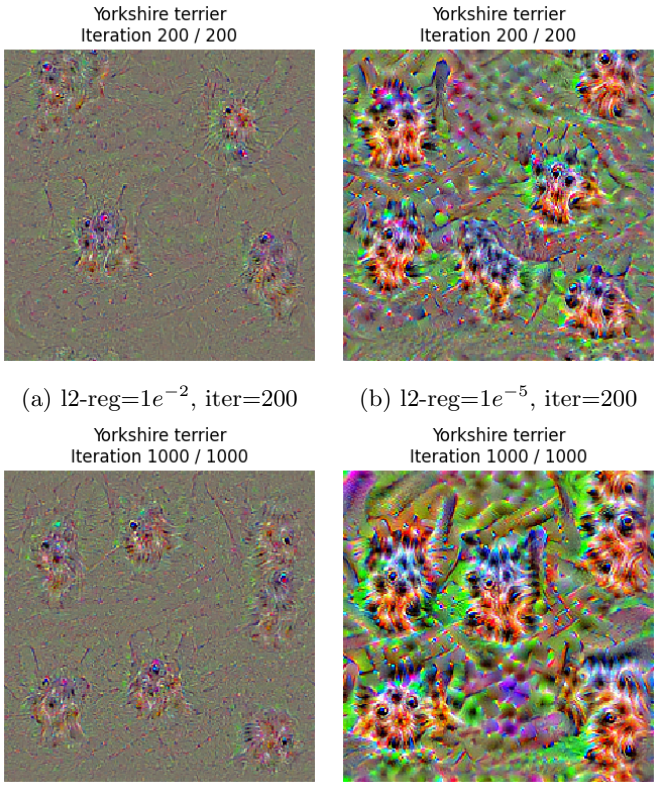
  L2-regularization factor represents the strength of the penalty term added to the optimization objective to limit large pixel changes in the generated image.

  We now experiment with different L2-regularization parameters. And in the following figures in Figure 8 we are presenting results using $1e^{-2}$ and $1e^{-5}$ regularization factor for 200 and 1000 iterations.

  When the regularization factor is at $1e^{-5}$, the generated images are noisier and contain high-frequency patterns that maximize the class score. We can see that when the iteration hits 1000, the image is demonstrated with more artifacts and irrelevant details. This means that with weak penalty on the pixel changes, the optimization prioritizes maximizing the class score without much regard for the interpretability.

  But when the regularization factor is larger at $1e^{-2}$, the generated

visualizations though are less noisy and contain smoother patterns, fail to capture finer details on the object and the situation is not improved with the increased iterations.



(a) l2-reg=$1e^{-2}$, iter=200 (b) l2-reg=$1e^{-5}$, iter=200

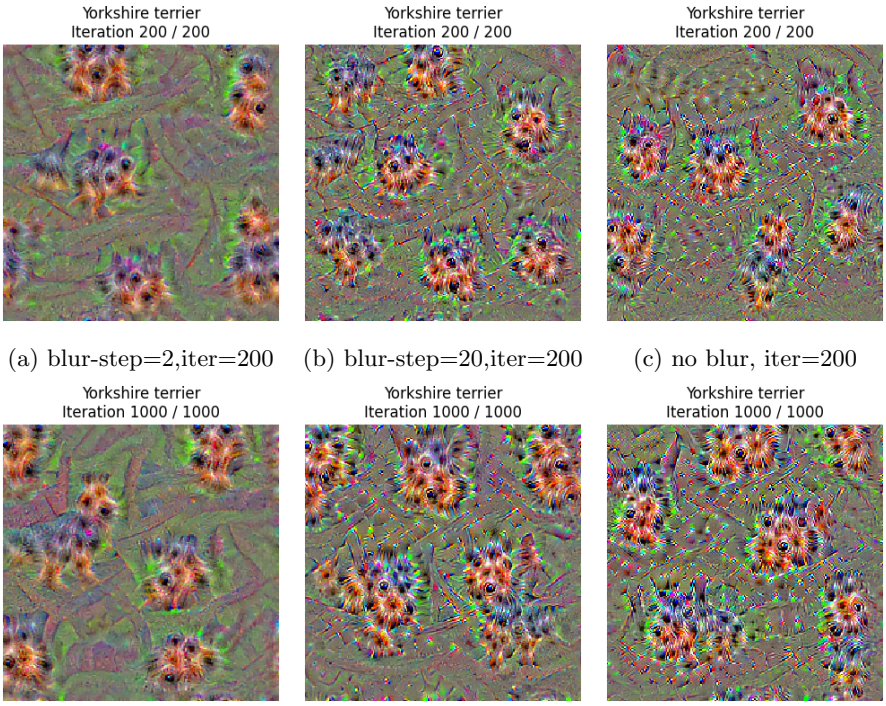
(c) l2-reg=$1e^{-2}$, iter=1000 (d) l2-reg=$1e^{-5}$, iter=1000

Figure 8: Comparison of Yorkshire terrier class visualization with L2-regularization factors and with (a) and (b) at 200 iterations, (c) and (d) at 1000 iterations.

- **blur frequency**

  We also experiment on various blur frequencies. The blur frequency is the process of applying a low-pass filter or smoothing operation to the input image during optimization. It can suppress high-frequency artifacts and focus on broader and more interpretable features of the target class.

  In the following experiments, we test on blurring every two steps blurring every 20 steps, and no blur at all.



(a) blur-step=2,iter=200 (b) blur-step=20,iter=200 (c) no blur, iter=200

(d) blur-step=2,iter=1000 (e) blur-step=20,iter=1000 (f) no blur, iter=1000

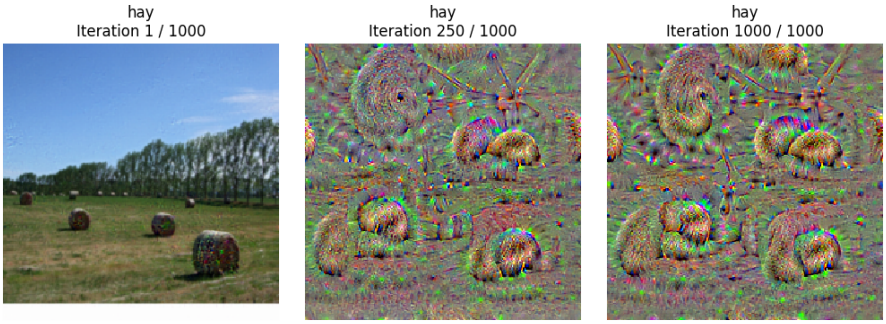
Figure 9: Class visualization of Yorkshire terrier with different blurring frequency

From Figure 9 we can see that with smaller blurring step at 2 which equals to higher blurring frequency results in smoother generated visualizations, the performance is slightly more explicit with higher iterations. And with larger blurring step at 20, the visualization balances between the fine details and the large-scale patterns with high-frequency patterns. However the noise still partially affect the visualization. And without blur, the images are highly saturated and appears relatively visually chaotic, they contain noisy and fragmented patterns which are harder to interpret.

Q10 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.

In class visualization, the choice of the initial image affects the optimization process and the quality of the generated visualization. Using an image from ImageNet as source image instead of random noise combines the network's learned features with meaningful patterns already present in the source image, resulting in more interpretable visualizations.When

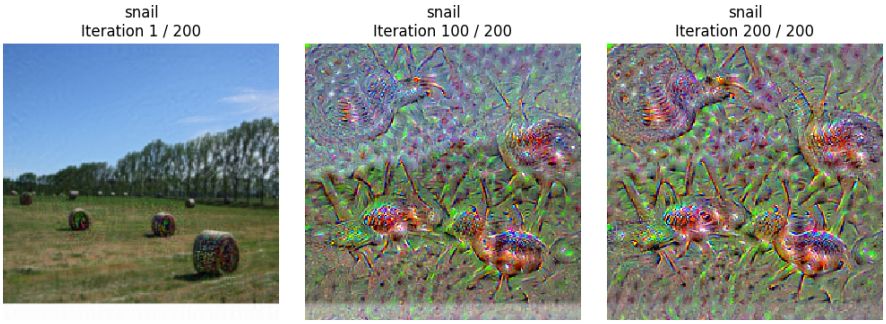
the source image already belongs to the target class (e.g. Figure 10), the optimization process starts closer to a solution that maximizes the target class score.Starting from an ImageNet image helps maintain the semantic structure of the input and increases the likelihood of converging to a global or near-global minimum that better represents the target class.



(a) initial state (b) iteration=250 (c) last iteration

Figure 10: Class Visualization using SqueezeNet: starting from an image of the same class as the target class

We also experiment on initiating the image from ImageNet with different target class. In the following images Figure 11 we create a snail class visualization starting from the image of hays.



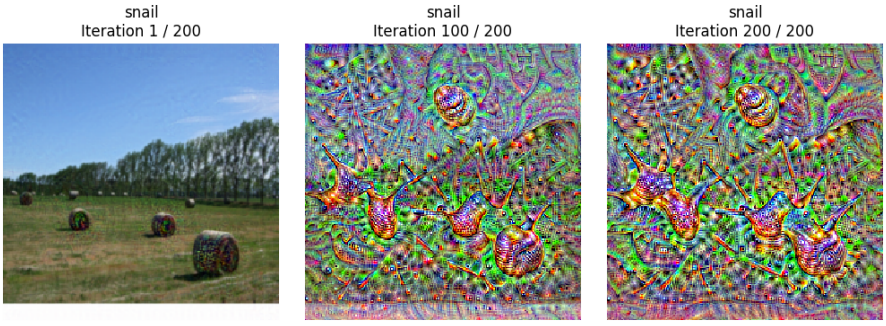
(a) initial state (b) iteration=100 (c) last iteration

Figure 11: Class Visualization using SqueezeNet: starting from an image of hays from ImageNet with target class as snail

Q11 Test with another network, VGG16, for example, and comment on the results.

To compare the performances of different networks, we do some identical experiments for the network of VGG16.

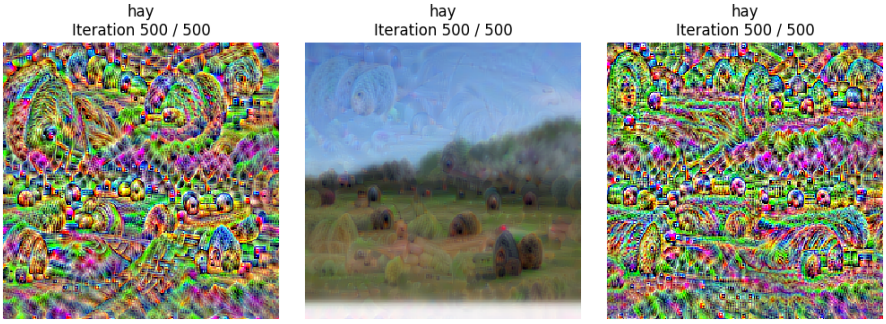
To compare with Figure 11, we do the same experiment on VGG16 network. VGG16 network has more layers compared to SqueezeNet, making it significantly deeper to capture more complex and hierarchical features which we can see from the patterns in Figure 12.



(a) initial state (b) iteration=100 (c) last iteration

Figure 12: Class Visualization using VGG16: starting from an image of hays from ImageNet with target class as snail

In the following experiments, we repeat our test on different learning rates, regularization factors and blur frequencies. And we test on initiating the image of same class as the target class from ImageNet,



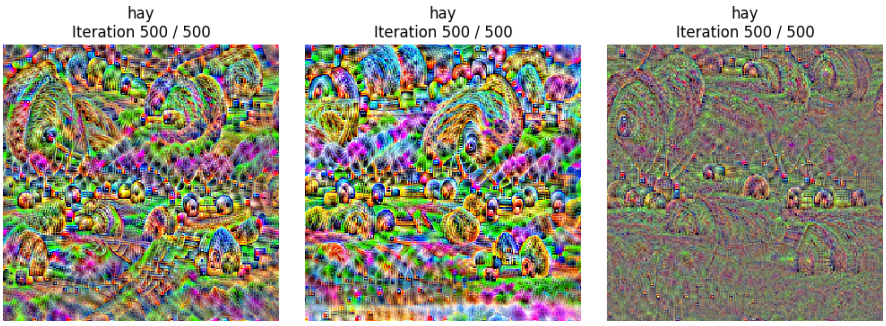
(a) default:lr=5 (b) lr=0.1 (c) lr=10

Figure 13: Class Visualization using VGG16: starting from an image of the same class as the target class with different learning rates.

As can be seen in Figure 13, the differences between figures using various learning rates become more apparent with VGG16. And the fig. 13a is the one using VGG16 with default parameters for comparison. But compared with the result of Figure 7, the patterns shown on fig. 13b is too smooth

to observe the features and the pixels are much more saturated with the higher learning rate.

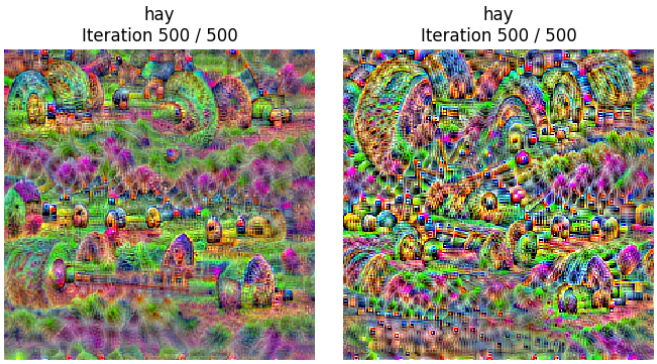
And as for the case with regularization factors, from the following figures we can see that with lower regularization factor, it becomes more challenging to obtain the features of the pattern.



(a) default:l2-reg=$1e^{-3}$ (b) l2-reg=$1e^{-5}$ (c) l2-reg=$1e^{-2}$

Figure 14: Class Visualization using VGG16: starting from an image of the same class as the target class with different regularization factors.

And finally for the blurring frequencies, we here compare the default blurring frequency (when blurring step=10) with higher blurring frequency (when blurring step=2):



(a) blur step=2 (b) default: blur step=10

Figure 15: Class Visualization using VGG16: starting from an image of the same class as the target class with different blurring steps.

With higher blurring steps we can get a smoother pattern of the objects, which is the same conclusion as before with SqueezeNet.

But generally we take much longer time to receive results from VGG16 since it is more computationally expensive than smaller models due to its depth and larger number of parameters.