Deep Learning Practical Work 2-b
Visualizing Neural Networks

Data, code, and PDF version of this file are available at
`https://rdfia.github.io`

Introduction

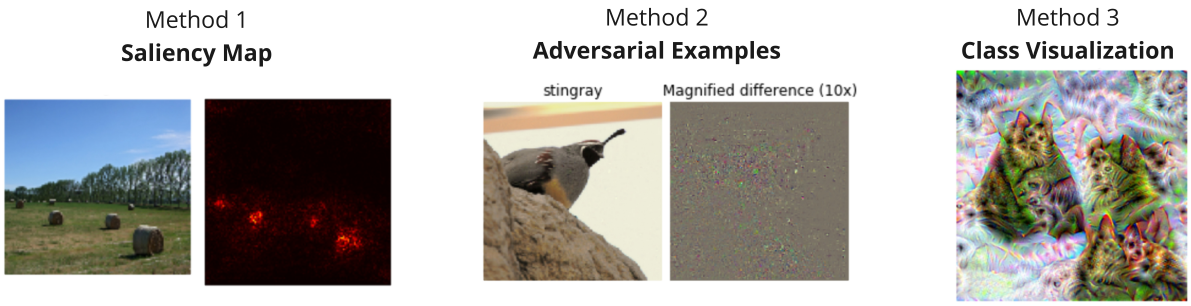

Figure 1: Illustration of the different methods

In this TP, we will study the various techniques "recently" published in order to study the behavior of convolutional neural networks. The commun principle of these approaches is to use the gradient of the input image with respect to an output class.

More formally, a neural network is a mathematical function f which, for a given input image, associates a **score vector** $\tilde{\mathrm{y}}$ for each class:

$$f: \begin{array}{ccc} \mathbb{R}^{h \times w \times c} & \rightarrow & \mathbb{R}^{K} \\ \mathrm{x} & \mapsto & \tilde{\mathrm{y}} \end{array} \tag{1}$$

We generally apply a softmax activation to this score vector to obtain a **probability distribution to the different classes** $\hat{\mathrm{y}} = \mathrm{SoftMax}(\tilde{\mathrm{y}})$.

In this TP, the goal is not to modify/train the network, but to use a pre-trained network, freeze its weights, and to modify the input image in order to indirectly study the behavior of the network. At first, we will use the SqueezeNet network from Iandola et al. (2016), trained on ImageNet, designed to be very compact and efficient all while allowing good results.

We will see 3 ways to study the function f by using the gradient $\frac{\partial \tilde{\mathrm{y}}_i}{\partial \mathrm{x}}$ of the score $\tilde{\mathrm{y}}_i$ with respect to the input image x :

- **Saliency Map**: Visualization of the impact of every pixel on the score of the correct class, based on the paper from Simonyan et al. (2014).

- **Adversarial Examples** *(fooling samples)*: Addition of minor modifications of an image, imperceptible for humans, which leads to the misclassification of the image, based on the paper from Szegedy et al. (2014).
- **Visualization of Classes** *(Deep dream-like)* : "Generation" of an image corresponding to a categorie, in order to visualize the type of patterns detected by the network, based on the papers Simonyan et al. (2014) and Yosinski et al. (2015).

Section 1 – Saliency Map

A saliency map is a map which indicates the important zones of an image. In our case of studying a CNN, this corresponds to the most impactful pixels for predicting the correct class.

To produce these maps, Simonyan et al. (2014) propose to approximate the neural network f in the neighborhood of an image x of class i by a linear function: $\tilde{\mathrm{y}}_i = f_i(\mathrm{x}) \approx w_i^\top \mathrm{x} + b_i$ avec $w_i = \dfrac{\partial \tilde{\mathrm{y}}_i}{\partial \mathrm{x}}$.

Remark that in practice, x is a 3D tensor, as is w_i which contains the derivative of the input image class score for every pixel.

To produce the heat map, Simonyan et al. (2014) propose to take the absolute value of the gradient w_i and then for every pixel take the maximum value of the 3 RGB channels to produce a 2D matrix.

Questions

1. ★ Show and interpret the obtained results.
2. Discuss the limits of this technique of visualizing the impact of different pixels.
3. Can this technique be used for a different purpose than interpreting the network?
4. Test with a different network, for example VGG16, and comment.

Section 2 – Adversarial Examples

Another aspect of analyzing neural networks is studying adversarial examples (also known as *fooling samples*), a practice since a few years. Using an existing pre-trained network and a correctly classified image, this method consists in modifying the image as little as possible such that it becomes misclassified by the network.

Szegedy et al. (2014) were the first to show this phenomenon: by only slightly modifying an image, it is possible to drastically change the predictions of the network. The method was further studied by Goodfellow et al. (2014) and many other later publications.

Remark : This "domain" of neural network analysis should not be confused with *Generative Adversarial Networks (GANs)*, a distinct field of research which has limited similarities to adversarial attacks.

There are many ways to create images. Here, we will look at a very simple and naive (yet efficient!) approach. Considering an image x correctly classified in class i, we want it to be classified in class j by modifying the image, not the network.

For this, we will perform gradient backpropagation $g = \frac{\partial \tilde{\mathrm{y}}_j}{\partial x}$ on the score $\tilde{\mathrm{y}}_j$ on the class j with respect to the input x.

We will progressively modify x according to the following rule:

$$\mathrm{x} \leftarrow \mathrm{x} + \eta \frac{g}{||g||_2} \qquad \text{with } \eta \text{ the learning rate.} \tag{2}$$

We apply this rule iteratively on the chosen image until the predicted class becomes j.

Questions

5. ★ Show and interpret the obtained results.
6. In practice, what consequences can this method have when using convolutional neural networks?
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).

Section 3 – Class Visualization

In order to visualize the type of patters likely to produce a particular class prediction, and so indirectly analyzing what a network detects, Simonyan et al. (2014) proposes a simple approach, further developed by Yosinski et al. (2015).

The principle is rather simple: considering an initial image x (random noise, for example), we want to modify this image such as to maximize the score for class i. To improve the quality of the results, we propose to use a certain number of regularization techniques. Firstly, we will add an L2 norm of the image as a regularization loss. Moreover, we will regularly apply implicit regularization to the image we're modifying, using translations and blurs (already implemented in the notebook).

Formally, for an image x that we want to modify to classify it in class i, we maximize the following loss:

$$\mathcal{L} = \tilde{\mathrm{y}}_i - \lambda ||\mathrm{x}||_2. \tag{3}$$

We progressively modify x with the following rule:

$$\mathrm{x} \leftarrow \mathrm{x} + \eta \nabla_{\mathrm{x}} \mathcal{L} \qquad \text{with } \eta \text{ the learning rate.} \tag{4}$$

Questions

8. ★ Show and interpret the obtained results.
9. Try to vary the number of iterations and the learning rate as well as the regularization weight.
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.
11. Test with another network, VGG16, for example, and comment on the results.

References

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

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

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

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *ICLR*, 2014.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *ICML Workshop*, 2015.