# Deep learning applications

**Deep Learning Practical Work**

Aymeric Delefosse & Charles Vin

2023 – 2024

# Contents

normaliser les images par m sig de image net 3.

# Chapter 1

# Transfer Learning

## 1.1 VGG16 Architecture

**1. ★ Knowing that the fully-connected layers account for the majority of the parameters in a model, give an estimate on the number of parameters of VGG16.** There are three fully-connected layers at the end of the VGG16 architecture. Calculating their weights is relatively straightforward. We take into account the inclusion of biases.

- The first fully-connected layer receives an input of size 7 by 7 by 512 (the resulting output of the convolutional layers), which equals an input size of 25,088. Knowing that there are 4,096 neurons, this layer has a total of $(25,088+1) \times 4,096 = 102,764,544$ trainable weights.
- The second fully-connected layer receives the input from the previous layer, which is 4,096, and also consists of 4,096 neurons, resulting in $(4,096+1) \times 4,096 = 16,781,312$ trainable weights.
- Lastly, the third fully-connected layer, consisting of 1,000 neurons, has $(4,096+1) \times 1,000 = 4,097,000$ parameters.

Thus, the fully connected layers account for a total of 123,642,856 parameters. We can confidently state that they represent at least 85% of the model, implying there should be around 140 million parameters to learn. If we consider a margin of 5%, there should be between 137,380,951 and 154,553,570 parameters.

We can readily confirm that the convolutional layers account for 14,714,688 parameters, meaning that there are actually 138,357,544 parameters in VGG16, meaning the fully-connected layers accounts to 89% of parameters.

**2. ★ What is the output size of the last layer of VGG16? What does it correspond to?** The output size of the last layer of VGG16 is 1000. It corresponds to the 1000 classes of the ImageNet dataset that the model has been trained on. Each element in this output vector represents the network's prediction scores for a specific class in the ImageNet dataset, and the class with the highest score is considered the predicted class for our given input image.

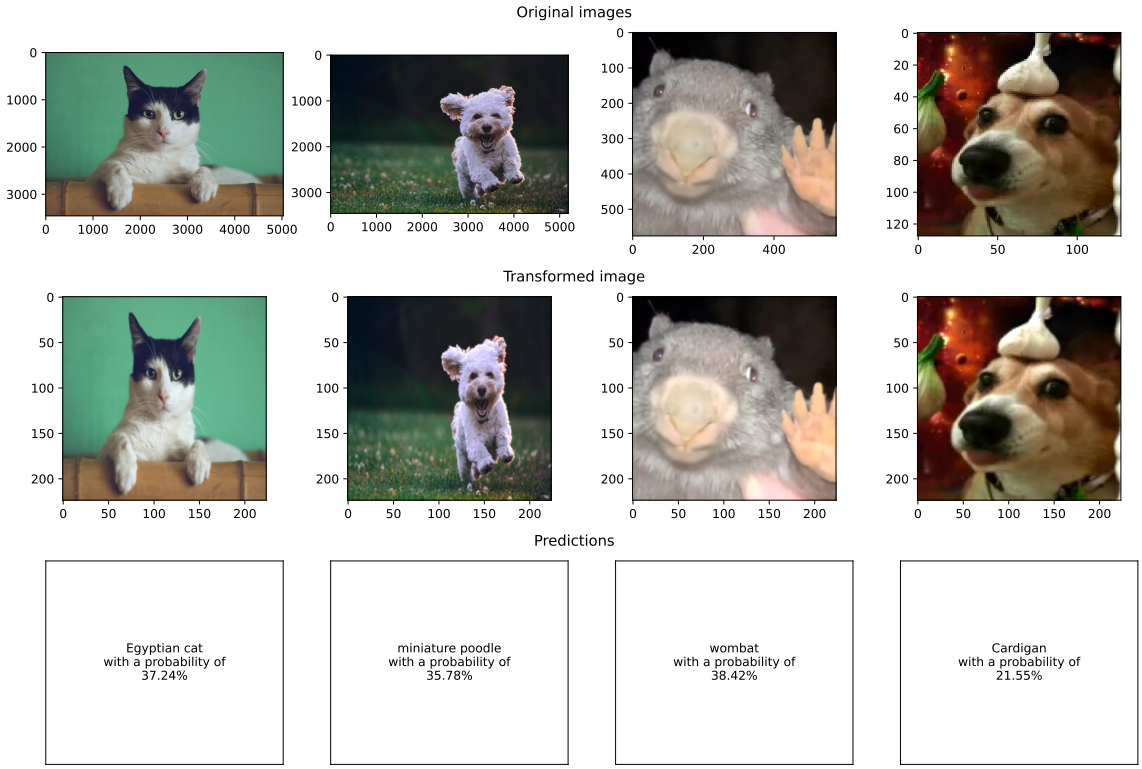



Figure 1.1: Prediction of VGG16 on few images

**3. Bonus: Apply the network on several images of your choice and comment on the results.**

**4. Bonus: Visualize several activation maps obtained after the first convolutional layer. How can we interpret them?**

## 1.2 Transfer Learning with VGG16 on 15 Scene

### 1.2.1 Approach

**5. ★ Why not directly train VGG16 on 15 Scene?** The 15 Scene dataset is quite small compared to the massive ImageNet dataset that VGG16 was originally trained on. VGG16 requires a lot of data to generalize well and to avoid overfitting. Moreover, training such a model from scratch is computationally expensive and time-consuming.

**6. ★ How can pre-training on ImageNet help classification for 15 Scene?** ImageNet represents a vast and diverse dataset, encompassing millions of images distributed across numerous categories. A model pre-trained on this dataset acquires a broad spectrum of features, spanning from fundamental edge and texture detection to intricate patterns. These acquired features serve as a robust initial foundation for extracting meaningful information from the 15 Scene images, even when the particular scenes or objects present in the 15 Scene dataset differ from those in ImageNet.

This approach proves particularly advantageous given the relatively modest size of our dataset, aiding in mitigating issues associated with insufficient training data, such as overfitting.

Furthermore, this approach expedites the training process as the model requires only fine-tuning of the previously learned features to adapt to the specific attributes of the new dataset, avoiding the need to start the learning process from scratch.

**7. What limits can you see with feature extraction?** The effectiveness of transferred features hinges on the similarity between the source task, which is the task the model was originally trained on, and the target task. If the target task significantly diverges from the source task, the extracted features may lack relevance or utility, failing to capture the nuanced details necessary for achieving high accuracy. For example, applying a model trained on natural images to domains like medical images or satellite imagery may not yield optimal results. Adapting such models to new domains often necessitates additional fine-tuning or, in some cases, complete retraining using domain-specific data, which can consume substantial computational resources.

It's worth noting that the biases present in the pre-training dataset can exert an influence on the features extracted by the model. If the pre-training data is not representative or contains inherent biases, these biases can inadvertently permeate into the target task.

Furthermore, utilizing models like VGG16 demands substantial computational resources, encompassing both memory and processing power, which can pose limitations, particularly in resource-constrained environments.

### 1.2.2 Feature Extraction with VGG16

**8. What is the impact of the layer at which the features are extracted?** In general, Earlier layers in a CNN capture basic features like edges and textures, while deeper layers capture more complex, high-level features that are more abstract and representative of the specific content in the image.

For certain tasks, simpler features extracted from early layers might be sufficient, whereas for more complex tasks (like distinguishing between very similar categories), deeper features might be more useful. Earlier layers tend to be more generalizable across different types of images and tasks, while deeper layers are more specific to the kind of images and tasks the network was originally trained on.

**9. The images from 15 Scene are black and white, but VGG16 requires RGB images. How can we get around this problem?** Image from 15 scene are black and white so they only have one channel. VGG requires 3 channel RGB images. The easiest workaround is to Replicate the single channel of the grayscale image across the three RGB channels. Another solution is to average the weights of the first convolutional layer (which is responsible for the RGB channels) so that it can directly accept grayscale images.

**10. Rather than training an independent classifier, is it possible to just use the neural network? Explain.** Absolutely! You have the flexibility to use any classifier you prefer, including neural networks.

In the case of VGG16 or similar pre-trained models, the original classification layer is typically a neural network. If your classification task closely aligns with what VGG16 was originally trained for, you can fine-tune this neural network directly. However, in cases where your task is significantly different from the original one, it might not yield optimal results to retain the pre-trained classifier, as it's highly specialized towards the original classes.

The choice between an SVM and a neural network classifier depends on your specific requirements. An SVM or a simpler classifier can strike a balance by utilizing the deep features extracted by VGG16 while providing a less complex decision boundary, which might be sufficient for your task. It's a trade-off between complexity and performance, and the choice should be based on the nature of your data and the task's requirements.

# Chapter 2

# Visualizing Neural Networks

## 2.1 Saliency Map

★ **Show and interpret the obtained results.** After visualizing many saliency maps, most of them were perfectly normal but some were pretty inconsistent. In the following, we will describe each case.

Figure 2.1 show some consistent saliency maps. When the model make a good prediction, we can see high value on the labelled object. It's the case on all image here exept for the fourth one where the model predicted "greenhouse" instead of "lakeside". In fact, with the fourth image, the model did not focused on the lake frontiers to make this wrong prediction. It has mostly looked on the dark ground and the high brithness part of the image leading to the greenhouse class prediction.

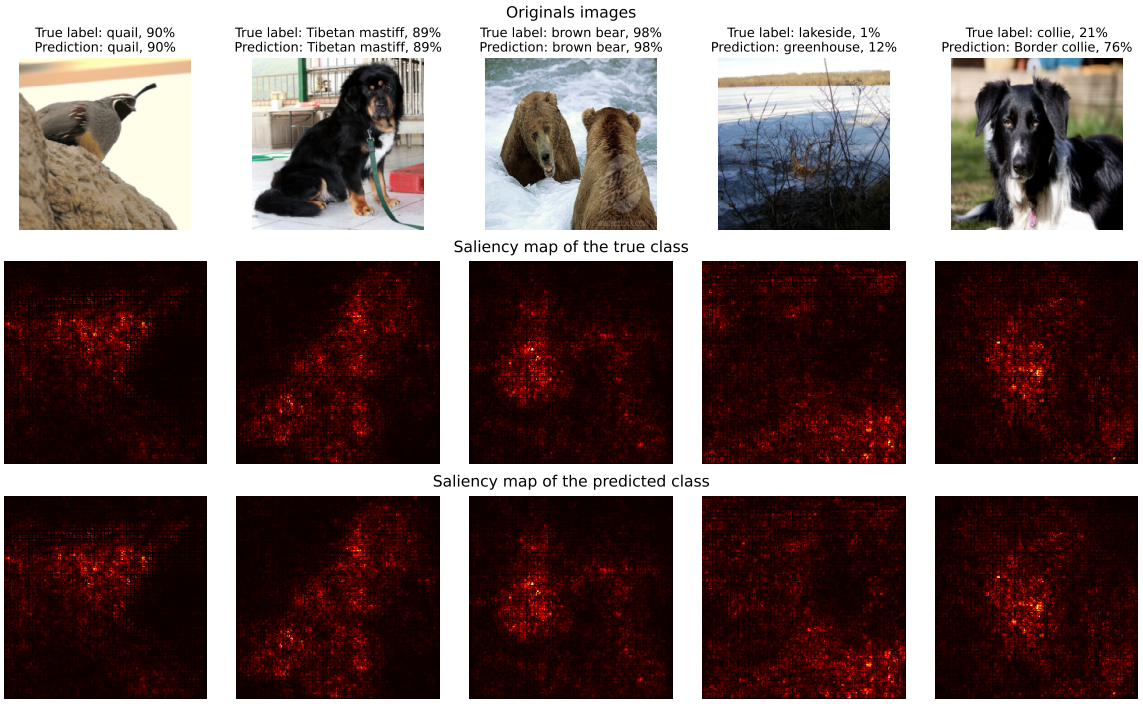



Figure 2.1: Consistent saliency maps of the predicted class

Figure 2.2 show some saliency maps that we found inconscistent. Both the first and the fourth one are image where the model prediction is correct but the saliency map is not informative at all, kind of blury. For the second and the last image, we can see that the model is looking at the right place but still predict the wrong class.

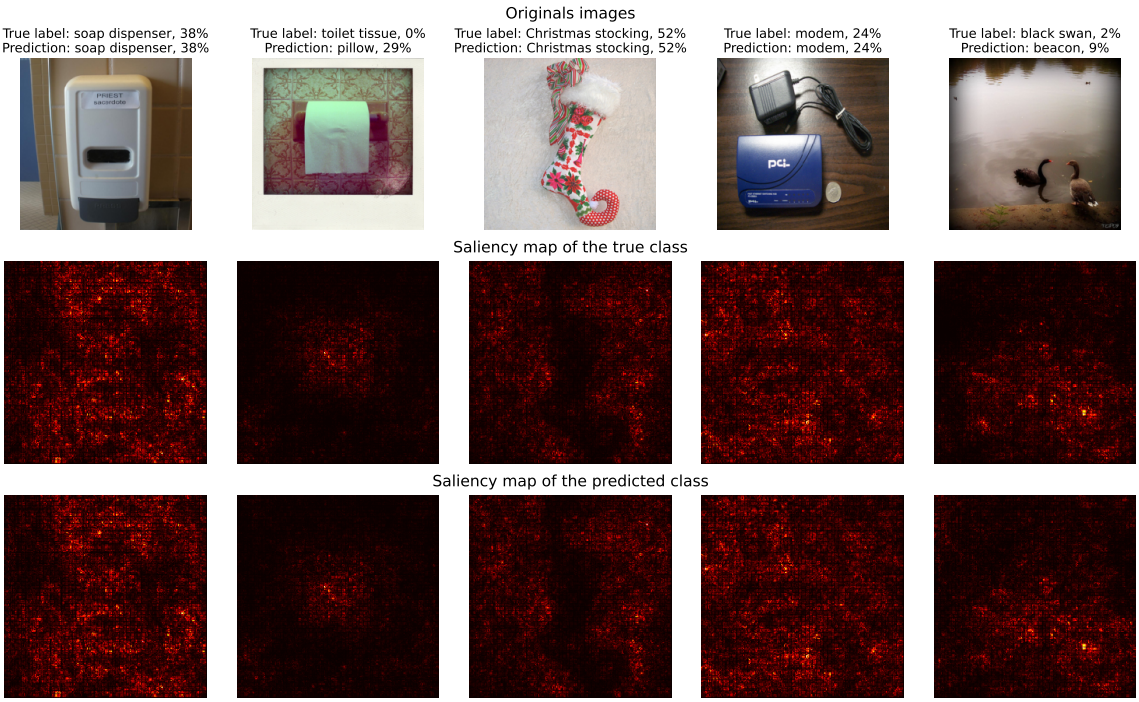



Figure 2.2: Inconsistent saliency maps of the predicted class

**Discuss the limits of this technique of visualization the impact of different pixels.** As we discussed in the previous question, saliency maps can face challenges in scenarios where an image contains multiple objects or complex scenes. In such cases, these maps may become cluttered or ambiguous, making it difficult to extract clear insights. They can also suffer from noise or, conversely, overemphasize certain features while downplaying others. This can result in a skewed interpretation of the model's focus.

Interpreting saliency maps can be subjective when they are ambiguous. Different individuals may draw different conclusions from the same saliency map, leading to inconsistent understandings of the model's behavior.

To gain a better understanding, we found it necessary to repeatedly interpret and visualize various saliency maps, especially when the maps were inconsistent. It can be challenging to keep in mind that each map represents the model's attention for a specific class and not the entire model. Examining saliency maps for other high-probability classes can provide additional context for a more comprehensive understanding of the model's decision.

It's important to note that saliency maps tend to highlight correlation rather than causation. They indicate areas where the model focused its attention but do not necessarily imply that these areas directly caused the model's decision.

**Can this technique be used for a different purpose than interpreting the network ?** Saliency maps can be used to identify and localize important objects or regions in an image. This can be particularly useful in applications like automated image tagging or initial steps of object detection. This can also help to create more effective augmented images by applying transformations (like rotations, scaling, cropping) that preserve these key areas.

This technique is predominantly applicable to image data. Its utility is limited in non-visual domains or for models that integrate multiple types of data (like text and images).

**Bonus: Test with a diffrentent network, for example VGG16, and comment.** Those are much better!

## 2.2 Adversarial Example

**★ Show and interpret the obtained results.**

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

(a) Last iteration

(b) GIF animation (use Adobe Acrobat for viewing or see *Gorilla_animated_500_regpp_blur.gif* )

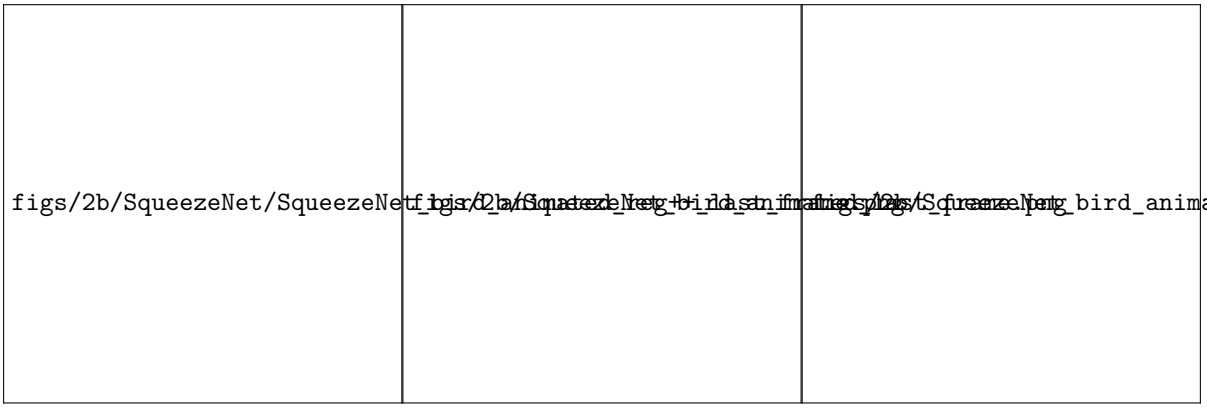
Figure 2.3: Class visualization: started from random noise, maximising the score from the gorilla class

**Bonus: 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)**

## 2.3 Class Visualization

**★ Show and interpret the obtained results.**

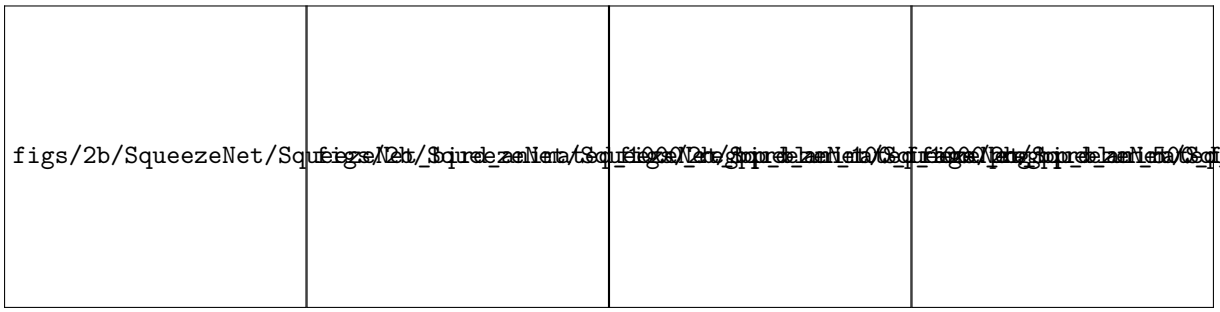
**Try to vary the number of iterations and the learning rate as well as the regularization weight.** Looking at the gif in Figure 2.3b, we can see the bluring occuring every 10 steps. To make better class visualization, regulation seem to be an important factor. In Figure 2.4, we tried disabling bluring the image and weight regularization on VGG16. We changed model for obtaining better visuals but all experiments were done on SqueezeNet and VGG16 afterward. Non regulated images seem to be more saturated everywhere, preventing to clearly see the represented class. We suspect that without regularization, the gradient saturate **everypixel** that could ever had participate to a correct gorilla prediction. By promoting pixel which the model has already engaged the transformation and keeping this direction,regularization permit the arrival of real class image.



(a) With a lot of regularization (b) With regularization (c) Without regularization

Figure 2.4: Comparaison of the bee eater class visualization using VGG16 with or without regularization (blurring and weight regularization) and starting from a base image of the class.

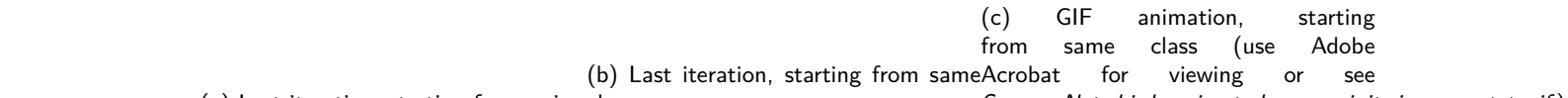
About the number of epoch, our experiment show that we need to make a certain number of epoch to let the visualization converge, thus getting better images.



(a) 200 iterations (b) 500 iterations (c) 1000 iterations (d) 1000 iterations, no regularization

Figure 2.5: Same image at different time. The last one is another run but without regularization.

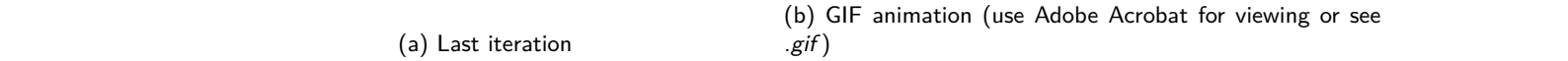
**Try to use an image from ImageNet as the source image instead of a random image (parameter init_img). You can use the real class as the target class. Comment on the interest of doing this.** Starting using an image is a good method, it give a good starting point of the gradient to create visualization and not going everywhere like when not using regularization.



(a) Last iteration, starting from noise (b) Last iteration, starting from same class (c) GIF animation, starting from same class (use Adobe Acrobat for viewing or see *SqueezeNet_bird_animated_same_init_img_reg++.gif*)

Figure 2.6: Class visualization: started from a bee eater image to maximising the score for the bee eater class. All with a strong regularization.

It pretty funny to create some objects from other objects. In Figure 2.7 we started from a image of hays to do a snail class visualization.



(a) Last iteration (b) GIF animation (use Adobe Acrobat for viewing or see *.gif*)

Figure 2.7: Class visualization: started from a hay image to maximising the score for the snail class. All with a strong regularization.

**Bonus: Test with another network, VGG16, for example, and comment on the results.** We did the same experiment using VGG16 this time. Visualization are much better because of the overall better performance of VGG16.