Deep learning applications

Deep Learning Practical Work

Aymeric Delefosse & Charles Vin

2023 – 2024

Contents

Chapter 1

Transfer Learning

The exploration focuses on Transfer Learning, where we adapt a well-known deep learning model – VGG16 – for new applications. This process involves utilizing the VGG16 architecture, originally designed for extensive image recognition, to comprehend and perform image classification on the 15 Scene dataset. It highlights how transfer learning "revitalizes" existing models, making them adaptable to new tasks and showcases their flexibility in addressing diverse real-world image processing challenges.

1.1 VGG16 Architecture

Examining the depths of neural network structures reveals the strong capabilities of models such as VGG16. Initially created for extensive image recognition, VGG16's complex layers of convolution and pooling highlight the advancements in deep learning. In this section, we explore the architecture of VGG16, breaking down its layers and understanding how they work together to extract features and classify images. This investigation not only clarifies the model's design but also sets the foundation for its practical use in various image processing tasks.

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 $(4,096+1)\times 1,000 = 4,097,000$ trainable weights.

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.

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

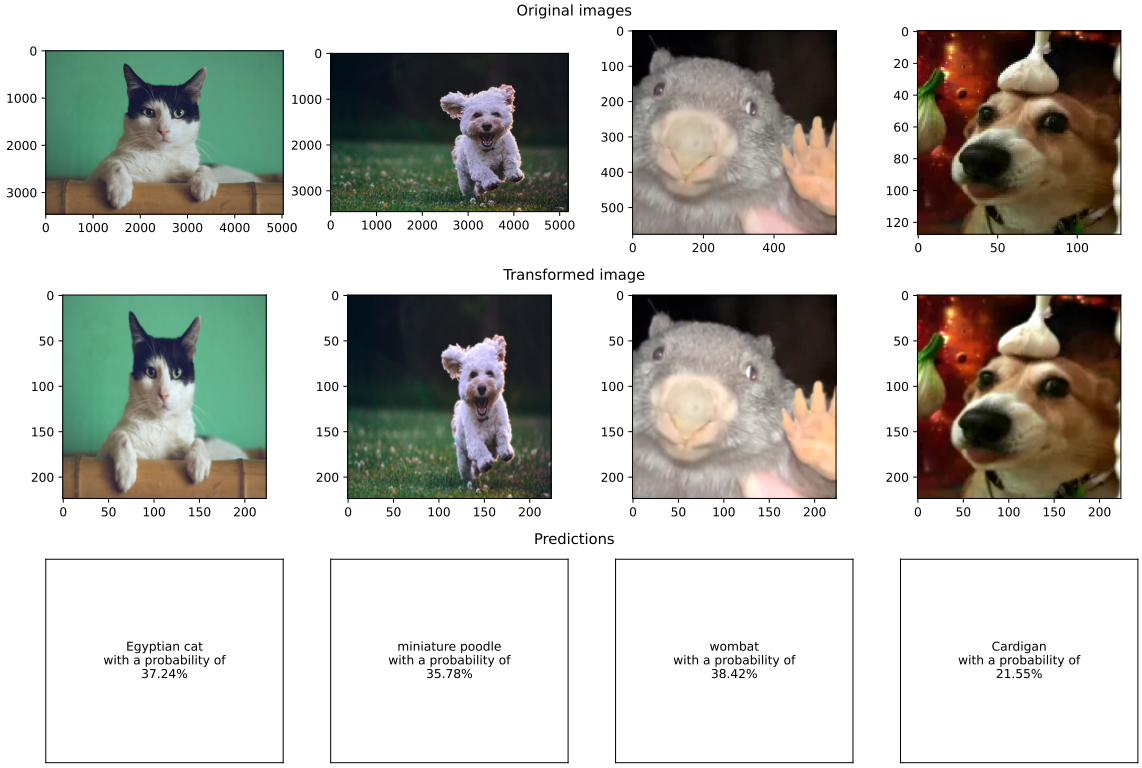


Figure 1.1: Prediction of VGG16 on few images

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

The concept of transfer learning plays a significant role when it comes to applying pre-trained models to unfamiliar fields. In this section, we'll explore the usage of the VGG16 model, which is well-known for its effectiveness in classifying images. Our focus will be on how established neural network models, originally trained on extensive datasets like ImageNet, can be cleverly repurposed for different yet related tasks. This approach highlights the adaptability of deep learning models and emphasizes the idea that existing knowledge encoded within a trained network can be successfully applied to new areas, a fundamental principle in modern machine learning research.

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 is a vast and diverse dataset, containing millions of images distributed across numerous categories. A model pre-trained on this dataset acquires a broad spectrum of features, ranging from basic edge and texture detection to intricate patterns. These acquired features provide 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 offers several advantages, especially considering the relatively small size of our dataset. It helps address issues related to insufficient training data, such as overfitting. Additionally, it speeds up the training process because the model only needs fine-tuning of the previously learned features to adapt to the specific characteristics of the new dataset, eliminating the need to start the learning process from scratch.

7. What limits can you see with feature extraction? The effectiveness of transferred features depends on the similarity between the source task, for which the model was originally trained, and the target task. When the target task significantly differs from the source task, the extracted features may not be relevant or useful, and they may fail to capture the nuanced details required for achieving high accuracy. For instance, using a model trained on natural images for tasks like medical images or satellite imagery might not produce optimal results. To adapt such models to new domains, additional fine-tuning or even complete retraining with domain-specific data is often necessary, which can consume significant computational resources.

It's important to note that biases present in the pre-training dataset can influence the features extracted by the model. If the pre-training data is not representative or contains inherent biases, these biases can unintentionally affect the performance on the target task. Moreover, the utilization of models like VGG16 demands substantial computational resources, including both memory and processing power, which can present limitations, especially 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 CNNs, earlier layers typically capture fundamental features such as edges and textures, while deeper layers capture increasingly complex and high-level features that are more abstract and representative of the specific content within an image.

For some tasks, the simpler features extracted from the early layers may suffice, while for more intricate tasks (such as distinguishing between very similar categories), the deeper features can be more valuable. Early layers are generally more adaptable across various image types and tasks, whereas deeper layers tend to be more specialized and specific to the types of images and tasks the network was initially 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, it is possible to use the neural network itself instead of training an independent classifier. In the case of models like VGG16, the final classification layer is essentially a neural network. If the classification task is similar to what VGG16 was originally trained for, we can fine-tune this neural network for our specific task. However, if the task is substantially different from the original one, it may not be ideal to retain the pre-trained classifier, as it might not perform well on new classes.

The decision between using the pre-trained neural network or training an independent classifier, such as an SVM, depends on the nature of the task and the available data. Using the pre-trained network can save time and computational resources, but fine-tuning or retraining may be necessary for optimal results on different tasks. Using a simpler classifier like an SVM can be a good compromise, leveraging the deep features extracted by the neural network while providing a more interpretable decision boundary.

Chapter 2

Visualizing Neural Networks

This exploration focuses on neural network visualization, a crucial aspect of understanding and analyzing how these models make decisions. It emphasizes the importance of interpretability and transparency in machine learning. Through the exploration of various visualization techniques, we can gain valuable insights into how these complex models interpret and process visual information. This endeavor helps bridge the divide between theoretical knowledge and real-world application, contributing to our deeper understanding of how neural networks function and their significance in modern AI solutions.

2.1 Saliency Map

This section focuses on identifying the most impactful pixels in an image for predicting its correct class. It employs a method proposed by Simonyan et al. (2014), which approximates the neural network around an image and visualizes the influence of each pixel.

1. ★ Show and interpret the obtained results. In the process of visualizing multiple saliency maps, most of them exhibited consistent patterns, while some displayed inconsistencies. Let's examine each case.

In Figure 2.1, we observe saliency maps that are generally consistent. When the model makes accurate predictions, we can observe high saliency values on the labeled object. This holds true for all the images in this figure, except for the fourth one where the model predicted "greenhouse" instead of "lakeside." In this case, the model did not focus on the lake boundaries, which led to the incorrect prediction. Instead, it primarily concentrated on the dark ground and the bright areas of the image, resulting in the "greenhouse" class prediction.

Now, in Figure 2.2, we encounter saliency maps that appear inconsistent. Both the first and fourth images are examples where the model's prediction is correct, but the corresponding saliency maps lack informativeness and appear blurry. In contrast, for the second and last images, the model seems to be focusing on the right regions, but it still predicts the wrong class.

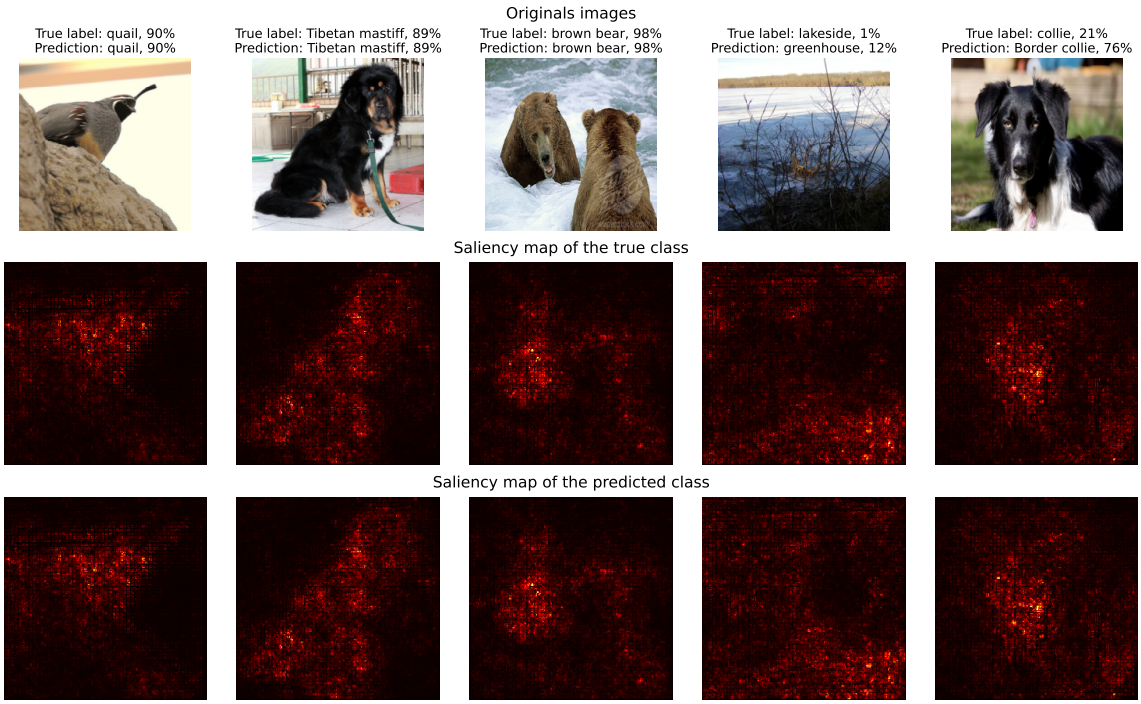


Figure 2.1: Consistent saliency maps of the predicted class

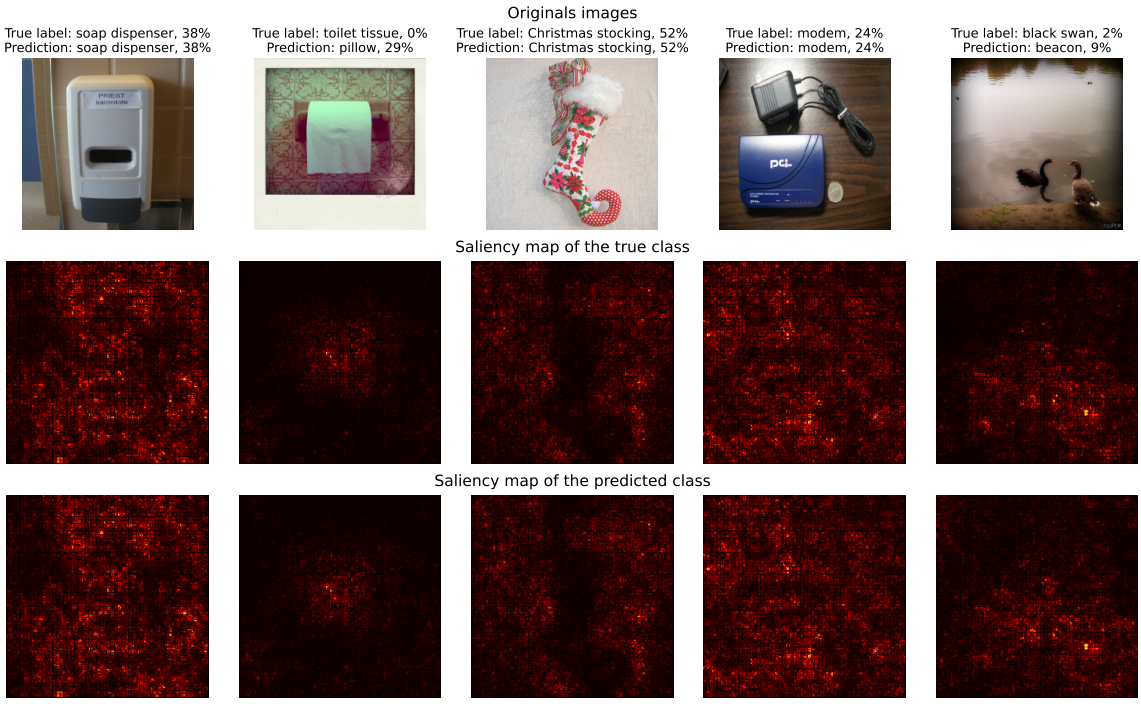


Figure 2.2: Inconsistent saliency maps of the predicted class

2. 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 when dealing with images containing multiple objects or complex scenes. In such scenarios, these maps may become cluttered or unclear, making it challenging to extract meaningful insights. They can also suffer from noise or, conversely, overemphasize certain features while downplaying others. This can result in a biased interpretation of the model's focus.

Interpreting saliency maps can be subjective, especially when they are ambiguous. Different individuals may draw different conclusions from the same saliency map, resulting in inconsistent interpretations of the model's behavior. To gain a better understanding, we found it necessary to repeatedly analyze and visualize various saliency maps, particularly when they exhibited inconsistencies. It can be a complex task to keep in

mind that each map represents the model's attention for a specific class, not the entire model. Examining saliency maps for other high-probability classes can provide additional context, contributing to a more complete understanding of the model's decision-making process.

It's important to note that saliency maps tend to highlight correlations rather than establishing causation. They reveal areas where the model directed its attention but do not necessarily imply that these areas directly influenced the model's decision.

3. 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 primarily suited for image data. Its utility is limited in non-visual domains or for models that integrate multiple types of data (like text and images).

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

2.2 Adversarial Example

Here, the goal is to study the vulnerability of CNNs to minor, imperceptible modifications in an image that lead to misclassification. This concept, introduced by Szegedy et al. (2014), reveals the limitations and unexpected behaviors of neural networks.

5. ★ Show and interpret the obtained results.

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

7. 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

This section aims to generate images that highlight the type of patterns detected by a network for a particular class, based on techniques developed by Simonyan et al. (2014) and Yosinski et al. (2015). This method helps in visualizing what features the network prioritizes for classification.

8. ★ Show and interpret the obtained results. .

(a) Last iteration

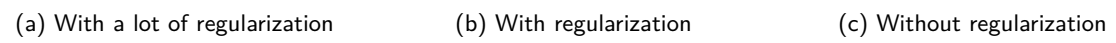
(b) GIF animation (use Adobe Acrobat for viewing or see *Gorilla_animated_500_regpp_blur.gif*)

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

9. 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 that bluring occurs every 10 steps.

To improve class visualization, it appears that regularization plays a crucial role. In Figure 2.4, we experimented with disabling image blurring and weight regularization on VGG16. Although we initially used a different model to achieve better visuals, all experiments were subsequently conducted on both SqueezeNet and VGG16.

Non-regularized images appear to be overly saturated, making it challenging to discern the represented class clearly. Our suspicion is that without regularization, gradients become saturated in **every pixel** that could have contributed to a correct prediction of "gorilla." By encouraging pixels that the model has already engaged in transformations and maintaining their direction, regularization facilitates the emergence of the true class image.



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

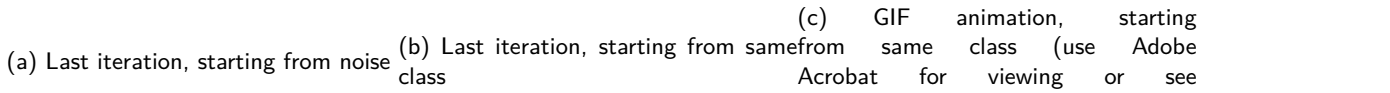
Figure 2.4: Comparaison of the bee eater class visualization using VGG16 with or without regularization (blurring and weight regularization) and starting from a base image of the class.

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.

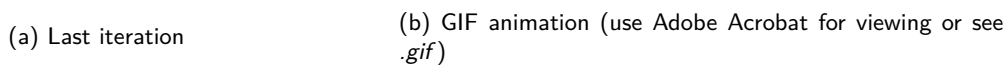
10. Try to use an image from ImageNet as the source image instead of a random image. You can use the real class as the target class. Comment on the interest of doing this. Starting using an image is a good method, it give a good starting point of the gradient to create visualization and not going everywhere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

11. 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.

Chapter 3

Domain Adaptation

This exploration focuses on domain adaptation to tackle the task of applying models trained in one domain to a distinct yet related domain. This entails the comprehension and application of concepts like the DANN model and the Gradient Reversal Layer, which serve as tools to render a model agnostic to the domain. This practical exercise underscores the complexities of training a model on one dataset, such as labeled MNIST, and subsequently utilizing it effectively on a different dataset, such as unlabeled MNIST-M. This mirrors real-world situations where domain adaptation plays a crucial role, e.g. autonomous driving.

1. If you keep the network with the three parts (green, blue, pink) but didn't use the GRL, what would happen? Without the GRL, the domain classifier would become proficient at distinguishing between source and target domains, counteracting the aim of domain adaptation. This would lead to a more domain-specific model rather than a domain-generalized one.

2. Why does the performance on the source dataset may degrade a bit ? The minor performance decrease on the source dataset result from the model adapting to features common to both domains, slightly reducing its specificity for the source domain.

3. Discuss the influence of the value of the negative number used to reverse the gradient in the GRL. The gradient reversal value balances learning domain-specific features and generalizing across domains. An optimal value is crucial for effective learning without compromising performance on either domain.

4. Another common method in domain adaptation is pseudo-labeling. Investigate what it is and describe it in your own words. Pseudo-labeling involves generating labels for the target domain using the model's predictions. These labels are then used for further training, helping the model adapt to the target domain by leveraging its existing knowledge and narrowing the domain gap.

Chapter 4

Generative Adversarial Networks