RDFIA - M2 IMA

Practical work report 2-a
Transfer Learning through feature extraction from a CNN

Done by :

Carlos GRUSS & Oussama RCHAKI

25/09/24

SORBONNE UNIVERSITÉ

Contents

Objective

Transfer learning is a machine learning technique where a model trained on one task is reused or fine-tuned for a related task, allowing the knowledge gained from solving one problem to be applied to another. Instead of training a model from scratch, transfer learning leverages the features and weights learned from a large dataset, making it especially useful when the new task has limited labeled data. This approach is highly effective when the tasks share similarities in their patterns, such as images or text. For example, a model pre-trained on ImageNet can be fine-tuned to classify medical images with minimal additional training.

The objective of this practical work is to explore transfer learning through feature extraction using the pre-trained VGG16 convolutional neural network. The task involves leveraging the VGG16 architecture, originally trained on ImageNet, to extract deep features from images in the 15 Scene dataset. These features will be used to train a linear Support Vector Machine (SVM) classifier for image classification.

1 – VGG16 Architecture

The convolutional network VGG16 (Simonyan & Zisserman, 2015) is a deep learning model designed for image classification. It was trained on the ImageNet dataset (Russakovsky et al., 2015), which contains over 1 million images categorized into 1000 classes.

The architecture of VGG16 can be observed in Figure 1 and consists of:

- Five convolutional blocks, each containing 2-3 convolutional layers followed by a pooling layer that halves the spatial dimensions.
- The number of feature maps doubles after each block.
- Three fully-connected layers at the network's end for classification.

Input images must be resized to 224×224 pixels and normalized with:

$$\mu = [0.485, 0.456, 0.406], \quad \sigma = [0.229, 0.224, 0.225].$$

VGG16 is widely used in transfer learning applications due to its robust architecture and pre-trained weights on ImageNet.

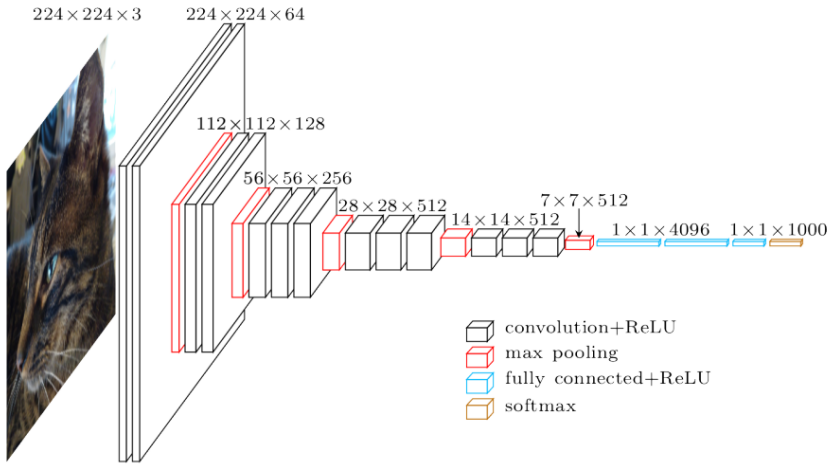

Figure 1 : VGG16 Network

Question 1

The fully-connected layers in VGG16 account for the majority of the model's parameters. Using the input size and output size of each fully-connected layer, we calculate the total parameters as follows:

- **First fully-connected layer:** $(7 \times 7 \times 512) \times 4096 + 4096 = 102,764,544$
- **Second fully-connected layer:** $4096 \times 4096 + 4096 = 16,781,312$
- **Output layer:** $4096 \times 1000 + 1000 = 4,097,000$

Total parameters in fully-connected layers:

$$102,764,544 + 16,781,312 + 4,097,000 = \boxed{123,642,856}$$

Thus, the fully-connected layers in VGG16 have approximately **123.6 million parameters** making them the dominant contributor to the model's overall parameter count.

Question 2

The output size of the last layer of VGG16 is 1,000, corresponding to the 1,000 classes in the ImageNet dataset used for training. Each value in this output represents the predicted probability for a specific class.

Question 3

- **Role of ImageNet Normalization:** Normalization ensures that the input images have the same scale and distribution as the data used during the model's training. This improves accuracy and consistency by aligning input statistics with those of ImageNet.
- **Why Set the Model to Eval Mode:** Setting the model to `eval()` disables behaviors specific to training, such as dropout and batch normalization updates, ensuring consistent and reliable predictions during inference.

Figure 2 : The predections of VGG16 applied to several image.

Question 4

Figure 3 : The input image.

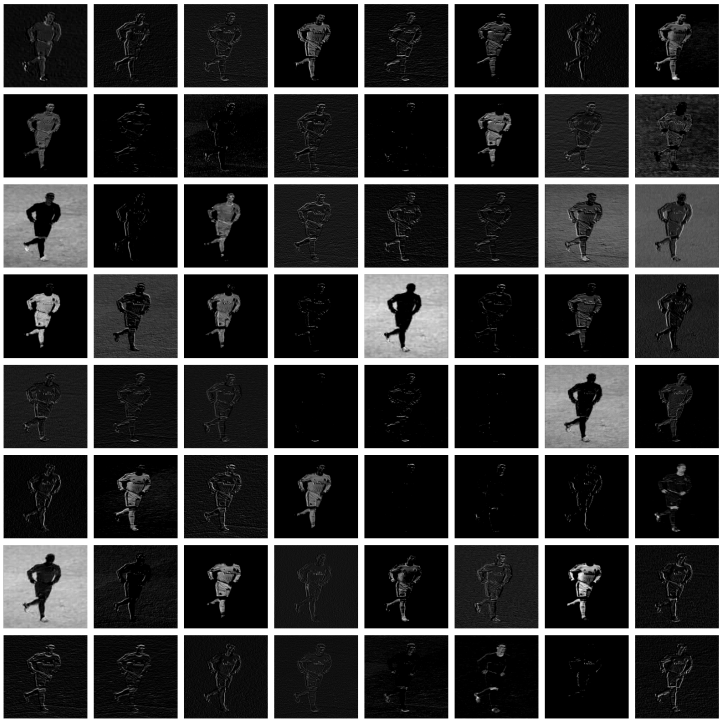

Figure 4 : The output of the first convolutional layer (the first convolutional layer and its activation layer).

In the VGG16 model, the activation maps obtained after the first convolutional layer highlight low-level features such as edges, textures, and simple patterns in the input image. Each map corresponds to a specific filter, showing regions where the filter activates strongly. Bright areas in the maps indicate where the feature is detected, while dark areas represent its absence. These activation maps help us understand how the network begins to "see" and process the input, focusing on fundamental features necessary for deeper layers to build upon.

2 – Transfer Learning with VGG16 on 15 Scene

2.1 – Approach

The objective of this section is to leverage a pre-trained neural network (vgg16 in our case) for feature extraction and train a classifier for the 15 Scene Dataset. This approach consists of two steps: first, we use the VGG16 network, pre-trained on ImageNet, to extract feature vectors from images using the output of its relu7 layer. These feature vectors provide a compact representation of the images, similar in principle to the SIFT + Bag of Words (BoW) approach, but more powerful due to deep learning.

In the second step, we use these feature vectors to train an SVM classifier for scene classification across the 15 categories. This method enables us to utilize the pre-trained network's ability to capture rich and transferable features while focusing on solving our specific classification task.

Question 5

Directly training VGG16 on the 15 Scene dataset is not ideal because this dataset is relatively small, with only 4,485 images, which is insufficient to train a deep network with millions of parameters. This could lead to overfitting and poor generalization. By using a pre-trained VGG16 for feature extraction, we leverage the rich features learned on a large dataset (ImageNet) and avoid the need for extensive training data.

Question 6

Pre-training on ImageNet helps classification for the 15 Scene dataset because the network learns rich, general features (e.g., edges, textures, shapes) from a large and diverse dataset of over 1 million images. These features are transferable and applicable to other tasks, like scene classification, even though the datasets are different. The higher layers of the network encode abstract representations useful for recognizing patterns across various domains, enabling effective classification with limited data from the 15 Scene dataset.

Question 7

The main limitation of feature extraction is that the pre-trained model may not focus on features specific to the 15 Scene dataset, as it was trained on a different task (ImageNet classification). This could result in less effective representations for scene-specific patterns. Additionally, since we're fixing the parameters of the pretrained VGG16, the expressibility of our model will be limited, as the model cannot fine-tune its representations for the target dataset.

2.2 – Feature Extraction with VGG16

To extract features at the `relu7` layer of VGG16, we create a modified network class (`VGG16relu7`) that truncates the original model up to this layer. After resizing and normalizing the input images, we generate the feature matrices `X_train` and `X_test`, normalize each feature vector with the $L2$ norm, and use them to train an SVM classifier.

Question 8

The layer at which features are extracted impacts the level of abstraction in the features: earlier layers capture low-level features (e.g., edges, textures), while deeper layers capture high-level, semantic features

more suited for classification tasks. Extracting features from a deeper layer (e.g., `relu7`) generally improves performance on tasks like classification due to their higher-level representations.

Question 9

The 15 Scene images are black and white (grayscale), but VGG16 requires RGB images as input. To address this, we can convert each grayscale image into an RGB format by duplicating the single grayscale channel into three identical channels. This process ensures that the image maintains its visual content while satisfying the input requirements of VGG16, which expects three channels (Red, Green, Blue) for its convolutional layers. This does not alter the underlying information in the image.

2.3 – Training SVM classifiers

Using the extracted deep features (X_{train}, X_{test}) and their corresponding labels (y_{train}, y_{test}), a multi-class SVM classifier is trained with a one-versus-all strategy using `sklearn.svm.LinearSVC` with $C = 1$. The model is trained using the `fit` function and evaluated on the test set using the `score` function, achieving an accuracy score of **88.81%**.

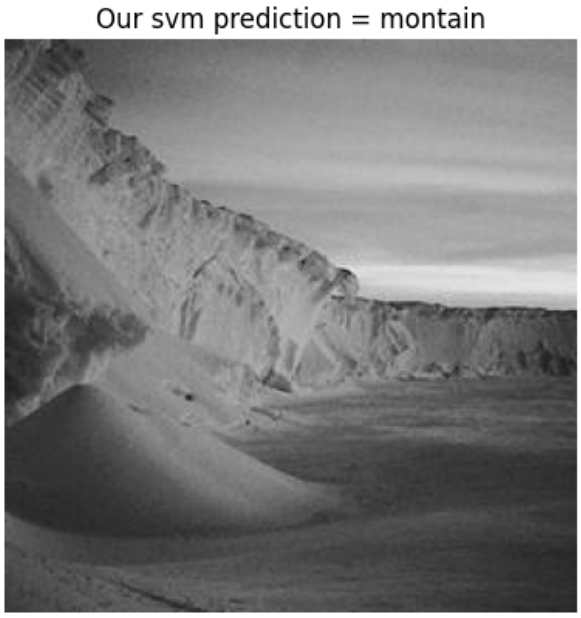

Figure 5 : Example of the 15 Scene dataset.

Question 10

Instead of training an independent classifier like an SVM, it is possible to use directly the neural network by replacing the final fully-connected layer of VGG16 (which outputs 1000 classes for ImageNet) with a new layer that outputs 15 values for the 15 Scene classes. This allows the network to perform classification directly on the new dataset by fine-tuning the model, leveraging the pre-trained feature extraction capabilities of the earlier layers while adapting the final layer to the new task.

2.4 – Going further

Question 11

Changing the layer at which features are extracted and varying the value of C

The impact of changing the layer from which features are extracted was evaluated, with the results presented in Table 1 and Table 2. In this analysis, ReLU6 corresponds to a network with only one Linear layer, while ReLU8 represents the network utilizing all Linear layers. The findings indicate a trend where performance decreases as more layers are included. This suggests that the complexity of the feature extractor correlates with the degree of specialization of the pre-trained network on ImageNet. Consequently, the features passed to the SVM become less meaningful for the new application, reducing their effectiveness for the Scene15 dataset.

Additionally, we tested the effect of varying the value of C in the SVM. Results showed that the optimal C value varied across different configurations of the pre-trained network. This variation highlights critical insights into feature complexity and generalizability. Higher C values often necessitate stronger regularization, which could be attributed to the increased intricacy and specialization of features toward ImageNet. These features, therefore, become less transferable and adaptable to the Scene15 dataset.

C	ReLU6	ReLU7	ReLU8
1.00E-05	86.37%	83.12%	82.75%
1.00E-04	90.05%	86.83%	86.43%
5.00E-04	**90.85%**	88.78%	87.30%
1.00E-03	90.72%	88.91%	**87.34%**
5.00E-03	90.42%	**88.98%**	86.77%
1.00E-02	90.52%	88.58%	86.53%
1.00E-01	90.18%	88.61%	85.49%
1.00E+00	90.15%	88.58%	85.19%
5.00E+00	90.15%	88.58%	85.26%
1.00E+01	90.15%	88.58%	85.19%
1.00E+02	90.15%	88.58%	85.13%

Table 1 : Performance varying layer and C without normalization

C	ReLU6	ReLU7	ReLU8
1.00E-05	80.40%	79.77%	70.85%
1.00E-04	82.08%	81.04%	74.34%
5.00E-04	83.55%	81.88%	76.52%
1.00E-03	84.19%	82.38%	77.49%
5.00E-03	85.09%	83.42%	79.60%
1.00E-02	85.76%	84.12%	80.67%
1.00E-01	88.91%	87.34%	83.95%
1.00E+00	90.69%	**88.81%**	86.97%
5.00E+00	**90.82%**	88.68%	**87.37%**
1.00E+01	90.69%	88.61%	87.17%
1.00E+02	90.42%	88.38%	86.13%

Table 2 : Performance varying layer and C with L2 normalization

Trying other pre-trained networks

In the previous section, we reported a performance of 88.81% in classifying the 15 scenes dataset using transfer learning with VGG16. In this section, our goal is to explore strategies to further enhance the effectiveness and push the boundaries of this approach.

An intuitive idea to improve performance is to use the same VGG architecture but with more layers. Specifically, we explored using VGG19, which features additional convolutional layers compared

to VGG16. As expected, this deeper architecture provided a slight improvement, achieving a score of 88.91%, compared to 88.81% with VGG16.

To further explore performance gains, we tested deeper and more advanced architectures such as ResNet. ResNet34, known for its residual connections, achieved a notable improvement with a score of 90.52%. Increasing the depth further with ResNet50 yielded an even better performance of 91.02%. Finally, ResNet152, the deepest model in our evaluation, reached an impressive score of 92.26%, showcasing the advantages of deeper networks for this classification task.

The increase in model performance with depth can be attributed to the ability of deeper networks to learn more complex and hierarchical features. As the number of layers increases, the model can capture finer details and higher-level abstractions from the input data. Residual networks (ResNets) further enhance this process by addressing the vanishing gradient problem through shortcut connections, enabling very deep architectures like ResNet152 to achieve superior performance.

Model	Score
AlexNet	87.14%
VGG16	88.81%
VGG19	88.91%
ResNet34	90.52%
ResNet50	91.02%
ResNet152	92.26%

TABLE 3 : Performance of Transfer Learning with different Models on the 15 Scenes Dataset with score = $accuracy_{test}$.

Replacing the SVM

To explore an alternative to using an SVM as a classifier, we tested whether the pre-trained VGG16 network could be extended to work directly with the Scene15 dataset. This was achieved by adding a ReLU activation to the last layer of VGG16 and introducing a linear layer to map from 1000 to 15 features. The modified network was then retrained for 10 epochs. Despite its simplicity, this approach achieved an accuracy of **85.56%** on the testing dataset—a remarkable result given the minimal changes and limited retraining.

This performance suggests that further improvements could be achieved by gradually scaling down the output features (e.g., by adding additional linear layers) and optimizing the training hyperparameters.

We could imagine that, in its current form, the model may be attempting to classify the 15 scenes by leveraging the presence or absence of the 1000 ImageNet classes within each scene. This highlights the potential of the pre-trained network's learned features, even with minimal adaptation.

Fine tuning

Fine-tuning is the process of taking a pre-trained model and adapting it to a specific task by training it on a smaller, task-specific dataset. It involves adjusting the weights of some or all layers in the model to improve performance on the new task, while leveraging the knowledge learned during pre-training.

This approach is slightly different from what we did before, as this time, we do not add any new layers after the pre-trained model. Instead, we freeze the weights of the initial layers of the pre-trained model and train only the last layer (e.g., the 4th layer for ResNet152). Additionally, we replace the classification layer to adapt the output to our specific use case (e.g., changing the output size from 1000 to 15).

Model	Layer 4 Only	Layers 3 and 4	Layers 2, 3, and 4	All Layers (1, 2, 3, 4)
ResNet34	92.56%	92.56%	92.70%	92.10%
ResNet50	93.03%	93.20%	93.37%	91.50%
ResNet152	92.50%	**93.60%**	92.60%	92.46%

TABLE 4 : Fine-tuning results for different layers across ResNet models.

The table illustrates the fine-tuning results for ResNet models across different configurations, where varying numbers of layers were trained. Generally, to achieve better performance, it is beneficial to

train more than just the last layer, as this allows the model to learn more significant task-specific representations. For instance, fine-tuning layers 3 and 4 of ResNet152 achieves the highest accuracy (93.60%), demonstrating the effectiveness of leveraging deeper features. However, caution is necessary when fine-tuning all layers, as it can lead to performance degradation. This occurs because fine-tuning all layers disrupts the pre-trained knowledge, effectively reducing the benefits of transfer learning and resembling training the model from scratch. In cases of small datasets, such as *15-Scenes*, and complex models like ResNet152, this approach can also result in overfitting.