Artificial Neural Networks Project

Self-Supervised Scene Classification

Professor:
José Oramas Mogrovejo
Teaching Assistants:
Hamed Behzadi Khormouji, Arian Sabaghi

Artificial Neural Networks

July 17, 2023

1 Introduction

In this project, you are tasked with training and comparing the predictive performance and learned features of a model following fully-supervised (first method) and self-supervised (second method) techniques. In the fully-supervised method, you should train a model to classify scenes from still images. This model will be trained with images paired with scene annotations.

Supervised learning suffers from the dependency on large amounts of annotated data and the high costs associated to their annotation. To cope with such drawback, self-supervised training techniques were proposed. A pretext task is a side process used in self-supervised learning for extracting *generic feature representations* from un-annotated data. Then, the goal is to adapt this representation or use it directly for other tasks.

Following this, you should compare two models, by calculating their **prediction accuracy**, analyzing **visual explanations** of the predictions made by each model, and **visual interpretations** of the features internally-encoded by the models.

In the following sections, in Section 2, we describe the scene datasets which will be used in this assignment. Section 3 covers the details of the architectures, transformations, and perturbations that should be implemented to classify the images into different scene categories. Section 4 describes the explanation method and the interpretation methods that should be used for generating visualizations for each of the trained models. To compare the trained models with respect to each other, Section 5 briefly explains the experimental protocol to be followed and the results that should be reported. Finally, Section 5.1 describes the submission file that must be prepared for this assignment.

2 15-Scene Dataset

The dataset contains 15 categories of different scenes [1]. The categories are *office*, *kitchen*, *living room*, *bedroom*, *store*, *industrial*, *tall building*, *inside cite*, *street*, *highway*, *coast*, *open country*, *mountain*, *forest*, and *suburb*. The dataset has been divided into two parts train and test. Each part has equally 15 different classes of scenes. The train set is used during the training process in order to "teach" the model how to classify images. The validation set is used to evaluate the model after each epoch, it is not seen by the model during training. You will find the dataset in the BlackBoard platform, located at *UA_2500WETANN: Artificial Neural Material > Practical Session > datasets > self-supervised scene classification Project*.

3 Classification schemes

In this section, the fully-supervised and self-supervised methods that should be implemented in this assignment are explained in more detail.

3.1 Supervised learning Scheme

In this section, you should fine-tune a convolutional neural network architecture, pre-trained on the ImageNet [2] dataset from the Pytorch deep learning library, on the 15 scene dataset. In this assignment you will use a pre-trained EfficientNet-B0 architecture from the Pytorch deep learning library. Following the exercise session on Transfer Learning, you should make all layers in feature extraction and classifier parts of the model trainable. When you load the model, you should also load the same transformation function applied during the pre-training phase and consider it as the transformation function in your supervised and self-supervised schemes. You can access the transformation via $torchvision.models.EfficientNet\_B0\_Weights.IMAGENET1K\_V1.transforms()$. The bellow address provides you with the scheme of the architecture of the EfficientNet-B0.

https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5f

3.2 Self-supervised learning Scheme

In this section, you should design two independent pipelines followig a self-supervised learning scheme. Each pipeline includes a two-step learning procedure, i.e., a pretext task, followed by the main task which is scene classification. For the first pipeline, rotation classification is considered as the pretext task. For the second pipeline, perturbation classification is considered as the pretext task. In both pipelines, the model trained in the pretext task should be further fine-tuned to address the main scene classification task.

In the following, we elaborate further on each of the pretext tasks.

Pretext Task 1: Rotation Classification. In this pipeline, you should train EfficientNet-B0, pre-trained on the ImageNet dataset, to classify the geometric transformation (rotation) applied to an image that is given to it as input. To achieve this goal, you need to drop the classifier part of the original EfficientNet-B0 architecture and add a new classifier which classifies images into four classes related to four rotations 0°, 90°, 180°, and 270°. In this training phase, all the layers in both feature extraction and classifier parts are trainable in the same manner as what you implemented for the fully-supervised scheme. As can be seen in Figure 1, the geometric transformations g define different rotations for the images given to the model. In addition, the output of the model for this pretext task will be four class labels corresponding to the four discrete rotations of interest. Before pushing image through the model, you apply a transformation by multiples of 90 degrees, i.e., 2d image rotations by 0, 90, 180, and 270 degrees. More formally, if $Rot(X, \phi)$ is an operator that rotates the image X by ϕ degrees, then your set of geometric transformations consists of the $K = 4$ image rotations $G = g(X|y)_{y=1}^{4}$, where $g(X|y) = Rot(X, (y-1)90)$. This represents $\phi = \{0, 90, 180, 270\}$

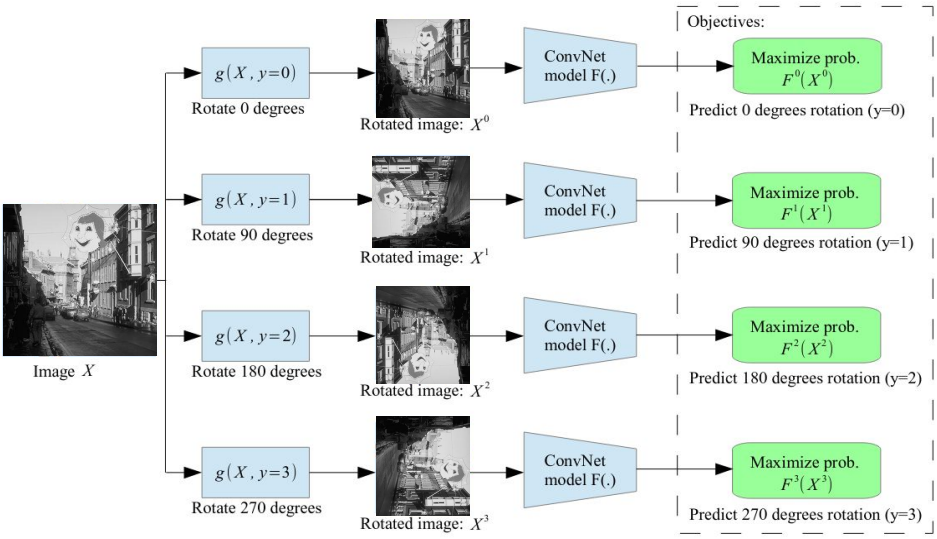

Figure 1: Illustration of the self-supervised task for scene feature learning. Given four possible geometric transformations, the 0, 90, 180, and 270 degrees rotations, you train a EfficientNet-B0 model $F(.)$ to recognize the rotation that is applied to the image that it gets as input. $F^y(X^y)$ is the probability of rotation transformation y predicted by model $F(.)$ when it gets as input an image that has been transformed by the rotation transformation y

Transformation Implementation. To implement the required rotations you need to use a function defined in *Pytorch* called *torchvision.transforms.functional.rotate*

Pretext Task 2: Black and White Perturbation. In this pipeline, you should train EfficientNet-B0, pre-trained on the ImageNet dataset, to classify perturbations applied to an image given to the model as input. To achieve this goal, similar to the previous pretext task, you need to drop the classifier part of the original EfficientNet-B0 and add a new classifier which classifies images into two perturbations, black and white. In this training phase, all the layers in both the feature extraction and classifier parts are trainable as they were for the fully-supervised scheme. To do the perturbation you need to define a random square region W with a shape 10×10 on an image. Then, the pixels within the window are set to zero (black perturbation) or 255 (white perturbation). Hence, the input image with black perturbation will have the class label black and the one with white perturbation area will have the class label white.

Note: In the training phases of both pretext tasks, the volume of your dataset will be increased because you need push an image into the model several times, each time with a different rotation (first pretext task) or different perturbation (second pretext task). As a result, your training phase will take longer time in comparison to that of previous method (i.e., fully-supervised method). Hence, you need to monitor the performance of your model in each epoch. Once your model has converged to a high performance, you can stop the training.

Scene Classification. In this step, you should drop the rotation classifier part and perturbation classifier part from each model trained in their corresponding pretext tasks, and add the original classifier to the end of the feature extraction part. In this training

phase, you need to fine-tune only the classifier part which means the feature extraction part should be frozen. In this step of learning, the classifier should be provided with the scene annotations such as *bedroom*, *living room*, *kitchen*, etc. Also, for comparison purposes, you should use the as same number of epochs as you used in the fully-supervised method.

4 Model Explanation and Interpretation

In this section, you are requested to explain and interpret the models trained following the fully-supervised and self-supervised procedures.

For the explanation part, you should utilize *Score-CAM* method from [4] and discussed during the theory sessions.

For the interpretation part, you should utilize the model inversion method proposed in [3]. This model inversion technique is part of the last exercise session.

5 Evaluation

To evaluate each trained models: a) scene classification (using fully-supervised method), b) rotation classification task, c) the model from (b) fine-tuned for the scene classification task, and d) perturbation classification task, e) the model from (d) fine-tuned for the scene classification task, you should:

1. Report the settings used for training each model such as learning rate, optimizer, batch size, the number of epochs, and the number of fully-connected layers in a table. Provide a justification for the values used for these settings.

2. Report the performance of the models in terms of classification accuracy. Summarize the results in a table. Besides, indicate along with a justification the epoch number from where the trained model was selected.

3. To train a scene-classification model on top of the perturbation pretext model, consider the pretext models pre-trained at early epochs, for example the third epoch, as well as the latest epoch. Report the classification accuracy of both scene-classification models. How can you justify the similarities or differences in the obtained results?

4. Compare the performance of the models, trained in the supervised and self-supervised schemes, and answer the following question: which one of the models succeeds in classifying the image with higher accuracy? Did you find the self-supervised scheme useful for classifying the scene classes? Justify your answer.

5. Report the strategies that you used to prevent underfitting and overfitting, in the case that you faced such problems. Provide evidence to support your explanation.

6. In addition to the quantitative report, compare the scene classification models, i.e., those trained in supervised and self-supervised schemes, using visualizations generated by Score-CAM. To do so, for a fixed set of 8 correctly predicted images, generate visualizations for higher and lower layers from each of the two models. Then,

discuss the visualizations with respect to each of the implemented schemes, i.e., supervised and self-supervised schemes. For generating the visualizations from the higher layer, consider the output from the last block in the EfficientNet-B0 (i.e., $torchvision.models.efficientnet\_b0.features[8]$). For visualizations from lower layer, use the output from the first block (i.e., $torchvision.models.efficientnet\_b0.features[1]$) and second block (i.e., $torchvision.models.efficientnet\_b0.features[2]$) of the EfficientNet-B0 architecture.

7. For the interpretation part, select five filters from the convolutional layer in the latest block (i.e., $torchvision.models.efficientnet\_b0.features[8]$) with the highest summation of its weights. Then, run the model inversion algorithm on the output of activation layer (i.e., SiLu layer $torchvision.models.efficientnet\_b0.features[8][2]$) related to the selected filters. The algorithm should generate a synthetic image illustrating the features that maximize the aggregated activations of the feature map outputs from the SiLu function (i.e., $torchvision.models.efficientnet\_b0.features[8][2]$). Generate these synthetic images for (1) the model trained in a supervised manner, (2) the model from the pretext rotation classification task, and (3) the model from the pretext perturbation classification task. Include these visualizations in your report, discuss the visualizations obtained for the three models above. Do the visualizations illustrate the expected encoded features? How do you justify similarities and/or differences between the generated visualizations for a given classes across different models.

5.1 Reporting Format

1- Prepare your reports in PDF format including the figures, and the answers to the questions mentioned in this assignment.

2- Make a folder including your report and your code (Note that your code should be properly commented).

3- Submit all documents in a zipped folder named as $Lastname\_FirstName\_StudentID.zip$.

References

[1] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 2169–2178.

[2] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[3] Jason Yosinski et al. "Understanding Neural Networks Through Deep Visualization". In: *CoRR* abs/1506.06579 (2015). arXiv: 1506.06579. URL: http://arxiv.org/abs/1506.06579.

[4] Haofan Wang et al. "Score-CAM: Score-weighted visual explanations for convolutional neural networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 24–25.