Artificial Neural Networks Project

Self-Supervised Scene Classification

Professor:

José Oramas Mogrovejo

Teaching Assistants:

Hamed Behzadi

Artificial Neural Networks

University of Antwerp

April 28, 2024

1 Introduction

In this project, you are tasked with training and comparing the predictive performance 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 downstream tasks.

Following this, you should compare the two models by calculating their **prediction accuracy**.

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 and transformations that should be implemented to classify the images into different scene categories. To compare the trained models with respect to each other, Section 4 briefly explains the experimental protocol to be followed and the results that should be reported. Finally, Section 4.1 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 *Artificial Neural Network, Project-Self-Supervised Scene Classification > 15-Scene.tar.*

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

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. For this assignment you will use a pre-trained EfficientNet-B0 architecture from the Pytorch deep learning library. 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-5fd

3.2 Self-supervised learning Scheme

You should design two independent pipelines following a self-supervised learning scheme. Each pipeline includes a two-step learning procedure, i.e., a pretext task, followed by the main (downstream) task which is scene classification. For the first pipeline, the size of a Gaussian Blur kernel filter applied to the input must be predicted as a pretext task. This is formulated as a classification where the different classes represent the different kernel sizes of interest. 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 what follows, we elaborate further on each of the pretext tasks.

Pretext Task 1: Gaussian Blurring. In this pipeline, you should train EfficientNet-B0, pre-trained on the ImageNet dataset, to classify the kernel size of a Gaussian blur filter 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 five classes related to five Gaussian Blur transformations, kernel sizes 5x5, 9x9, 13x13, 17x17, and 21x21. 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 Gaussian transformations g consider different kernel sizes for the images given to the model. In addition, the output of the model for this pretext task will be one out of five class labels corresponding to the five discrete kernel sizes of interest. Before pushing an image through the model, you must apply a transformation by blurring it using pre-defined kernel size. More formally, if $g(x, s)$ is an operator that blurs the image x via a Gaussian filter with kernel size s, then your set of Gaussian transformations consists of the images filtered with different kernel sizes $K = 5$.

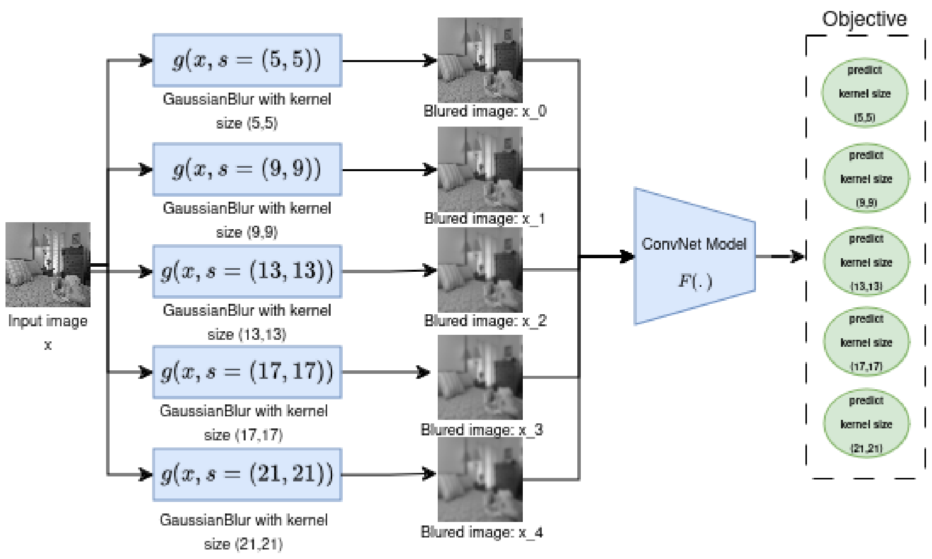
Objective
$g(x, s = (5,5))$
GaussianBlur with kernel size (5,5)
$g(x, s = (9,9))$
GaussianBlur with kernel size (9,9)
$g(x, s = (13,13))$
GaussianBlur with kernel size (13,13)
$g(x, s = (17,17))$
GaussianBlur with kernel size (17,17)
$g(x, s = (21,21))$
GaussianBlur with kernel size (21,21)
Input image x
Blured image: x_0
Blured image: x_1
Blured image: x_2
Blured image: x_3
Blured image: x_4
ConvNet Model $F(.)$
predict kernel size (5,5)
predict kernel size (9,9)
predict kernel size (13,13)
predict kernel size (17,17)
predict kernel size (21,21)

Figure 1: Illustration of the self-supervised task for scene feature learning. Given five possible Gaussian Blur kernel size, the 5x5, 9x9, 13x13, 17x17, and 21x21, you train a EfficientNet-B0 model $F(.)$ to recognize the kernel size of Gaussian Blur that is applied to the image that it gets as input.

Transformation Implementation. To implement the required Gaussian Blur filter you need to use a function defined in *OpenCV* called *OpenCV.GaussianBlur*

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 images 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 blur filtering (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 Gaussian kernel 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/trained 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 Evaluation

To evaluate each of the trained models: a) scene classification (using fully-supervised method), b) Gaussian Blur kernel size 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. This part only considers the perturbation pre-text task. 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 last 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.

4.1 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 $Lastname\_FirstName\_StudentID.zip$.