Scene Classification

Robbe Nooyens
s0201010

July 24, 2024

1 Introduction

This report explores the retraining of the EfficientNet-B0 model to classify 15 distinct scenes using both fully-supervised and self-supervised learning techniques. The primary goal is to compare the performance of these methods and assess the effectiveness of self-supervised learning in reducing the reliance on annotated data.

Self-supervised learning leverages unannotated data through pretext tasks that do not require manual labeling. In this project, the EfficientNet-B0 model and the 15-Scene dataset are used to evaluate and compare the two learning techniques.

In the fully-supervised scheme, the EfficientNet-B0 model is fine-tuned directly on the annotated 15-Scene dataset. The self-supervised scheme involves two pretext tasks: classifying Gaussian blur kernel sizes and classifying black and white perturbations. Models trained on these pretext tasks are subsequently fine-tuned for the main scene classification task.

The objectives of this project are to implement these learning schemes, compare their performance, and analyze the effectiveness of self-supervised learning in reducing the need for annotated data. This comparison aims to provide insights into the potential benefits and limitations of self-supervised learning for scene classification.

2 Repository Architecture

The repository supporting this project is organized into six key files: `main.py`, `models.py`, `data.py`, `transformations.py`, `logger.py`, and `config.py`.

The `transformations.py` file contains functions that apply Gaussian blur or perturbations to an image. The `logger.py` file collects accuracy and loss data throughout the training, validation, and testing phases, exporting this data to a CSV file. The `config.py` file defines configurations, each with a specific set of parameters, allowing for the creation of different experiment setups without duplicating code.

The `data.py` file includes a `DataHandler` that creates data loaders according to a given configuration, with the option to apply image transformations. The `models.py` file handles the loading, training, and evaluation of models based on the configurations defined in `config.py`. Data is consistently split into training, validation, and test sets to ensure proper evaluation.

This structured approach facilitates the efficient execution and comparison of multiple training configurations, supporting the exploration of both fully-supervised and self-supervised learning techniques.

3 Fully Supervised Training

The process of fine-tuning the EfficientNet-B0 model for the 15-Scene classification task began with a fully-supervised learning approach. Initially, both the feature extraction and classifier layers were made trainable, and the original parameters from the EfficientNet paper were applied, albeit with a reduced learning rate of 0.001 suitable for fine-tuning rather than training from scratch.

Despite following the proven EfficientNet configuration, the initial results, depicted in Figure 1a, revealed a significant issue. Although the model quickly achieved a high accuracy, the validation loss started to increase after a few epochs, while the training loss remained low. This divergence suggested that the model was overfitting to the training data.

To address this overfitting, a series of adjustments were made. The RMSprop optimizer was replaced with Adam, which combines the advantages of RMSprop and momentum. Additionally, the learning rate was lowered to 0.00002 to enable more precise weight updates. The batch size was reduced to 16, and the dropout rate was increased to 0.3 to enhance regularization.

These modifications yielded smoother training curves and better generalization, as seen in Figure 1b. Further improvements were sought by introducing a hidden layer in the classifier. This additional layer, with 320 nodes and a dropout rate of 0.5, aimed to enhance the model's learning capacity while maintaining regularization.

With this configuration, shown in Figure 1c, the model exhibited a more gradual improvement in accuracy, ultimately surpassing the earlier results. In another attempt, a second hidden layer was added to the classifier, further halving the number of nodes. This more complex architecture with extra weights needed a slight increase in the learning rate to 0.0001 to ensure efficient training.

The final configuration, as illustrated in Figure 1d, demonstrated rapid learning with validation loss and accuracy stabilizing early. This setup achieved the highest validation accuracy of 94.64% at epoch 9, indicating robust generalization without overfitting. The training was stopped at this point as further epochs did not yield additional improvements.

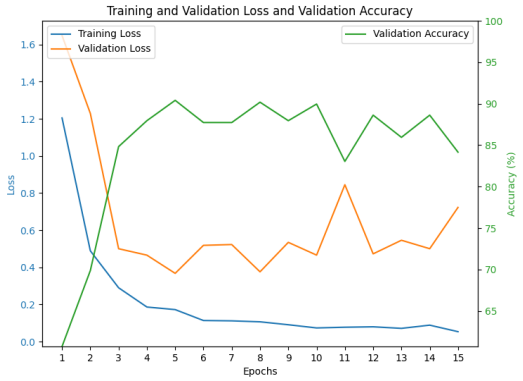
The tables below summarize the performance and parameter settings for the selected configurations.

Configuration	**Test Loss**	**Test Accuracy (%)**
Original parameters	0.3675	90.40
Adam optimizer	**0.2127**	93.52
One hidden layer	0.2223	93.08
Two hidden layers	0.2229	**94.64**

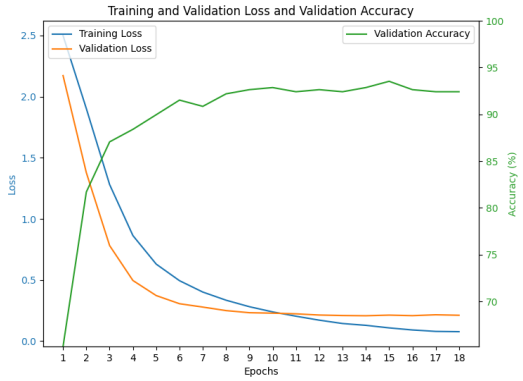
Table 1: Test Loss and Accuracy for Selected Configurations

Configuration	**LR**	**Optimizer**	**Batch size**	**Epochs**	**FC Layers**
Original parameters	0.001	RMSprop	32	5	1
Adam optimizer	0.00002	Adam	16	15	1
One hidden layer	0.00002	Adam	16	13	2
Two hidden layers	0.0001	Adam	16	9	3

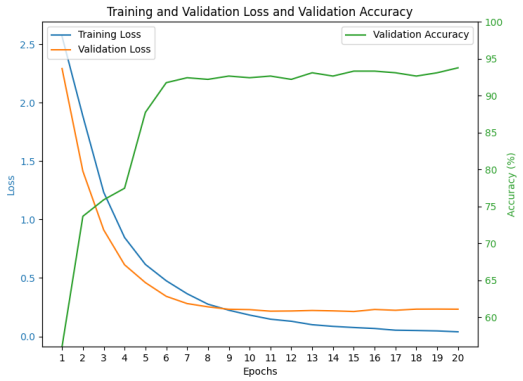
Table 2: Parameter Settings for Selected Configurations



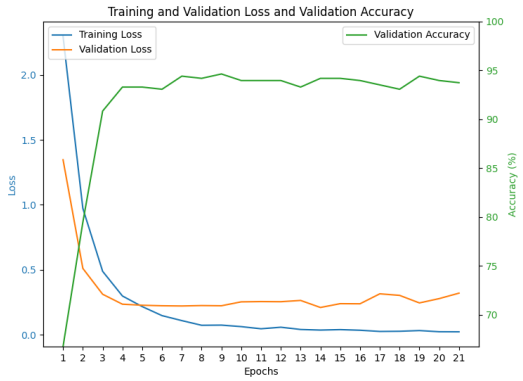
(a) Original EfficientNet parameters



(b) Adam optimizer and adjusted parameters



(c) 1 Hidden Layer



(d) 2 Hidden Layers

Figure 1: Performance of Fully Supervised Tasks

4 Self-Supervised Training

4.1 Gaussian Blur Task

The Gaussian blur pretext task involves training the EfficientNet-B0 model to classify the kernel size of a Gaussian blur filter applied to the input images. The kernel sizes considered are 5x5, 9x9, 13x13, 17x17, and 21x21. This pretext task aims to help the model learn useful feature representations from unannotated data, which can then be transferred to the main scene classification task.

To implement this task, the classifier part of the pre-trained EfficientNet-B0 model is replaced with a new classifier designed to predict one of the five Gaussian blur kernel sizes. The learning rate is set to 0.0001, similar to the previous best model, and a dropout rate of 0.2 is used to prevent overfitting. The batch size is increased to 32 given the increased dataset size.

During training, the model quickly achieves a high validation accuracy, reaching around 99% after the first epoch. To avoid overfitting, the training is stopped after confirming the initial high performance in the second epoch.

For the main scene classification task, the model trained on the Gaussian blur pretext task is fine-tuned. The original classifier from the best performing fully-supervised model, consisting of two hidden layers with dropout rates of 0.5, is used. The fine-tuning process is limited to a maximum of 9 epochs to prevent over-

fitting and to compare its performance with the fully supervised model.

Figure 2 shows the performance of the fine-tuned model on the scene classification task. The validation accuracy improves steadily and starts to plateau, similar to the fully-supervised model's performance after 9 epochs but with a lower accuracy.

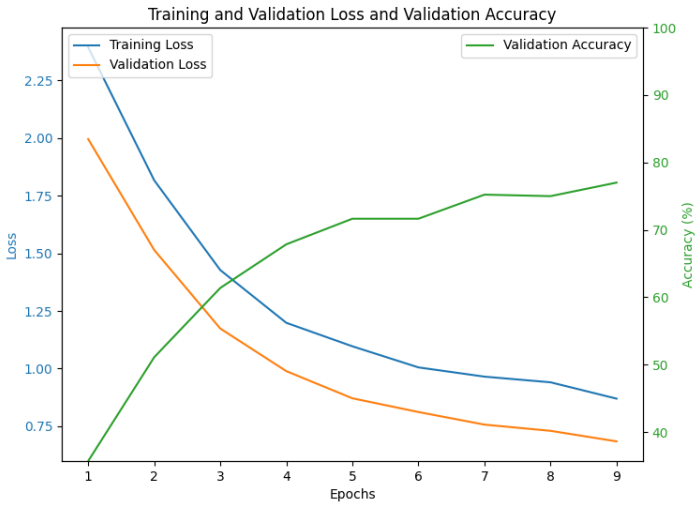


Figure 2: Gaussian Blur Scene Classification

4.2 Perturbation Task

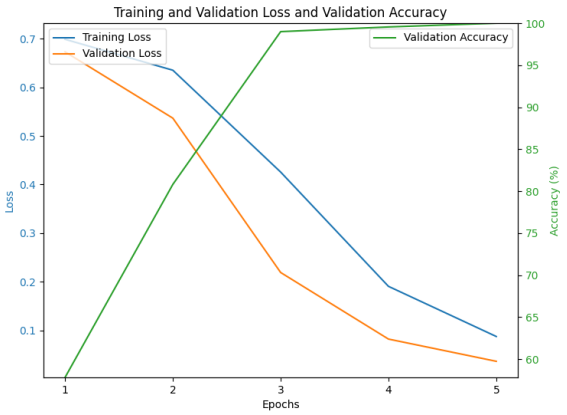
The perturbation pretext task involves training the EfficientNet-B0 model to classify whether a random square region in an image has been set to black or white. This task is designed to help the model learn robust feature representations from unannotated data, which can then be transferred to the main scene classification task like with the Gaussian blur.

To implement this task, the classifier part of the pre-trained EfficientNet-B0 model is replaced with a new classifier designed to predict the type of perturbation applied to the image. Initially, a learning rate of 0.0001 and a dropout rate of 0.2 were used. However, the model quickly achieved a validation accuracy of 100% after the first epoch, indicating that the task might be too easy and the feature extractor could be overfitting.

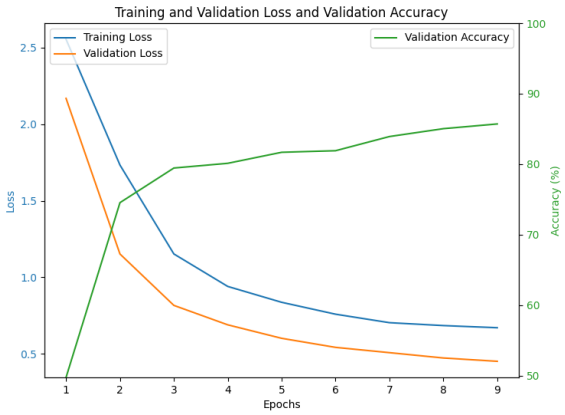
To address this, the learning rate was reduced to 0.00001, and the dropout rate was increased to 0.5 to add more regularization. With these adjustments, the model exhibited more balanced performance, as shown in Figure **??**.

For the scene classification task, the model after the third epoch was selected, where the model achieved high performance without signs of overfitting. Here too, the original classifier from the best-performing fully-supervised model, consisting of two hidden layers with dropout rates of 0.5, was used. The fine-tuning process ran for only 9 epochs, similar to the fully-supervised and Gaussian blur models, to compare the different methods fairly.

Figure **??** shows the performance of the fine-tuned model on the scene classification task. The validation accuracy improved steadily, and the model demonstrated good generalization with minimal overfitting.



(a) Perturbation Pretext Task



(b) Perturbation Scene Classification

Figure 3: Comparison of Pretext Tasks

4.3 Pretext Comparisons

The effectiveness of the self-supervised learning schemes was evaluated by comparing the performance of the models trained on the Gaussian blur and perturbation pretext tasks. For each pretext task, models were evaluated based on their performance on the main scene classification task.

For the Gaussian blur pretext task, the model achieved a high pretext accuracy of 99.55% and a validation accuracy of 77.00% on the scene classification task, with a validation loss of 0.6842 at the last epoch. The test accuracy for the scene classification task after Gaussian blur pretext training was 77.90%. This indicates that the model learned useful features from the Gaussian blur task, which transferred reasonably well to the scene classification task.

The perturbation pretext task model achieved a pretext accuracy of 98.99%. When fine-tuned for the scene classification task, it reached a validation accuracy of 85.71% and a validation loss of 0.4504 at the last epoch. The test accuracy for the scene classification task was 80.35%. This suggests that the features learned from the perturbation task were more beneficial for the scene classification task compared to those learned from the Gaussian blur task.

Task	Pretext Acc.	Validation Acc.	Loss	Test Acc.
Gaussian Blur	99.55	77.00	0.6842	77.90
Perturbation	98.99	85.71	0.4504	80.35

Table 3: Performance Comparison of Pretext Tasks

The differences in performance can be attributed to the nature of the pretext tasks. The perturbation task likely provided more varied and challenging feature learning opportunities compared to the Gaussian blur task, resulting in better transferability to the scene classification task. Figures 2 and 3b illustrate the performance of the models on the scene classification task after training on the respective pretext tasks.

To further assess the impact of early and late epoch training on the perturbation pretext task, models trained at the third epoch and the last epoch were fine-tuned for the scene classification task. The early epoch model achieved a validation accuracy of 79.46%, while the late epoch model reached 85.71%. The slight improvement in the late epoch model suggests that additional training on the pretext task continued to refine useful feature representations.

Epoch	**Validation Accuracy (%)**
3rd Epoch	79.46
Last Epoch	85.71

Table 4: Performance third and last epoch perturbation

5 Results

Comparing the performance of the models trained in the supervised and self-supervised schemes reveals that the self-supervised model using the perturbation task achieved a test accuracy of 80.35%, higher than the Gaussian blur pretext task model. However, the fully-supervised model, with optimized parameters and additional hidden layers, outperformed both self-supervised models with a test accuracy of 94.64%.

Model	Accuracy (%)
a) Finetuned	99.55
b) Gaussian blur pretext	98.99
c) Gaussian blur scenes	77.90
d) Perturbation pretext	98.99
e) Perturbation scenes	80.35

Table 5: Performance overview

The strategies employed to prevent underfitting and overfitting included using dropout layers, adjusting learning rates, and fine-tuning the complexity of the classifier architecture. These measures were crucial in achieving robust performance and generalization across different models and tasks.

Overall, the results indicate that while self-supervised learning schemes can effectively reduce the need for annotated data, careful selection and optimization of pretext tasks are essential for maximizing performance in downstream tasks.