Computer Vision Project - report

Darius Dabert
darius.dabert@polytechnique.edu

Elia El Khoury
elia.elkhoury@polytechnique.edu

1. Introduction

The fields of artificial intelligence and deep learning are making great strides in today's world, and more and more problems are finding their solutions in them. One of these problems is the classification of a very large dataset with very few labeled images. In this report, we will detail a deep learning challenge in which we had to solve this exact problem. We had access to a dataset of almost 300,000 images distributed over 48 different classes. We were presented with three folders: the first one contained 15 labeled images for each of the 48 classes, the second one was the set of 300k unlabeled images where the classes were not necessarily uniformly represented and the third and final folder was a test set of 14,000 images. All of the images were AI-generated which complicated even more our mission because a few of them were not very representative of their class. Our goal was to rigorously train a neural network model for it to be capable of correctly classifying unlabeled images across the 48 classes. To do this, we turned to semi-supervised learning techniques and data augmentation methods.This report presents an overview of our methodologies, results and insights gained from this challenging competition.

2. Algorithms and models implemented

2.1. Fine-Tunning Models

The initial approach we proposed to address the problem involved fine-tuning pre-trained models. Due to the small number of labels in the dataset, training a model entirely from scratch using only these labels is not feasible. To achieve reasonable accuracy on the test set, we first opted to fine-tune the last layer of deep models. Specifically, we utilized two types of models: ResNet and Vision Transformers.

To further improve this solution easily, we explored the application of data augmentation techniques on the training set. Data augmentation serves to increase the size of the dataset, thereby providing more examples (with variation for the image and its augmentation) for the model to learn from. We focused on two specific augmentation methods: random augmentation using RandAugment from the torchvision module, and MixUp augmentation.

In the case of random augmentation, various transformations such as affine transformation, color jitter, and Gaussian blur were randomly applied to the images. The parameters for these transformations were learned during the training of the models on the ImageNet dataset. On the other hand, MixUp augmentation involved selecting random images from the dataset and computing a weighted average of their pixel values as well as their corresponding labels. Both approaches yielded comparable results in terms of their effectiveness.

2.2. Pseudo-labelling

The initial approach yields satisfactory results, but it encounters a limitation. The training set lacks sufficient data to effectively learn generalized features, particularly considering the high variability within each class. Consequently, the first model tends to overfit rapidly, leading to reduced performance on unseen data. To address this challenge, it is crucial to provide the model with more training samples.

One potential solution is to employ pseudo labelling on the dataset as seen in this pseudo-labelling paper [1]. The algorithm follows these steps: First, we train a model on the labelled training dataset. Then, we utilize this trained model to predict labels for the unlabelled dataset. If the model's prediction for a particular image surpasses a confidence threshold, denoted as τ, we include the corresponding images in the training set, assigning them pseudo labels. These pseudo-labels serve as temporary replacements for the true labels. Subsequently, we retrain the model using this augmented training dataset. This process can be repeated multiple times to progressively build a larger dataset.

Unfortunately, due to significant dissimilarities between samples from different classes, the model's biases can be amplified through this technique. Consequently, it is necessary to closely monitor overfitting and employ appropriate techniques to mitigate bias, such as regularization methods.

2.3. Self-supervised pretext task

To mitigate the bias learned during training, we explored techniques that could leverage the unlabelled dataset di-

rectly, allowing us to incorporate a larger amount of information from the data distribution. Our search led us to self-supervised learning. Specifically, we employed a technique known as rotation recognition as seen in this Unsupervised Learning paper. [2].

This algorithm involves creating a new dataset comprising images and their respective rotations at angles belonging to the set 0, 90, 180, 270 degrees. The goal is to train a vision transformer model to recognize the specific rotation applied to each image. By doing so, the model is compelled to learn specific details associated with each rotation class, which are encoded in the weights of the model's backbone.

Subsequently, we fine-tune this pre-trained model by modifying the classification layer. This process enables the model to adapt its learned representations to our specific task, making it more effective in solving the problem at hand. By utilizing self-supervised learning and leveraging rotation recognition as a pretext task, we are able to enhance the model's performance and reduce the impact of bias introduced during training.

2.4. Semi-supervised learning

In addition to exploring self-supervised learning, we also pursued semi-supervised learning as an alternative approach. This approach involves training the model simultaneously on both the labeled training set and the unlabeled set. One algorithm we employed for this purpose is called FreeMatch as seen the FreeMatch paper [3].

The FreeMatch algorithm operates by creating batches that contain samples from both the labeled training set and the unlabeled set. We make predictions and compute losses for the labeled samples as usual. However, for the unlabeled samples, we predict a label and assess its confidence. If the confidence exceeds a predefined threshold, we temporarily assign the predictions as labels and compute the model's prediction on an augmented version of the image.

Furthermore, we dynamically update the threshold value based on the confidence of these predictions. This adaptive thresholding process allows us to continuously adjust the balance between the labeled and unlabeled samples during training. By integrating both labeled and unlabeled data, the model can leverage the additional information from the unlabeled set to improve its performance.

3. Results

In this section we will present the different results we obtained for each algorithms. We will then analyse them and talk about hyperparameters and axis of amelioration.

3.1. Setup

To ensure a thorough evaluation of our models, we employed a split of the labeled dataset into a training set (comprising 80% of the images) and a validation set (comprising the remaining 20% of the images). This partitioning allowed us to train our models on a subset of the labeled data while reserving a separate portion for evaluation purposes. Furthermore, to obtain more reliable performance estimates, we employed cross-validation by varying the distribution between the training and validation sets in multiple iterations.

In the context of our experiments, we tested different types of models, including convolutional networks, ResNets, and vision transformers. After evaluation, we determined that the "vit_base_patch16_224" vision transformer exhibited better performance, and computational efficiency. Consequently, this model was chosen as it provided satisfying results while being computationally manageable.

3.2. Quantitative Results

3.2.1 Fine Tunning

The first solution we proposed was the fine tunning of a pretrained model. We used a SGD optimizer and a cross_entropy loss. The hyperparameters for the fine-tunning are in the table 1.

Hyperparameter	Value
Learning rate	0.01
Epoch	8
Batch size	64
momentum	0.9

Table 1. Hyperparameters for Finetunning

The results for the fine tunning are resumed in the following graph in figure 2.

Figure 1. Accuracy comparison between ResNet and ViT

The model demonstrates rapid convergence, but after only two iterations, we observe signs of overfitting in the training curves in Figure 3.

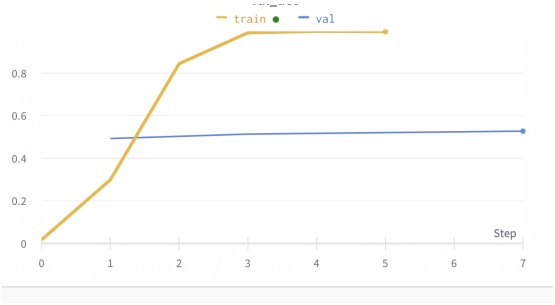

Figure 2. Overfiting when using the finetuning method

3.2.2 Pseudo labeling

The hyperparameters for this algorithm are in Table 2.

Hyperparameter	Value
Learning rate	0.01
Epoch	2
Batch size	64
momentum	0.9
threshold	0.85
Number of iterations	5

Table 2. Hyper parameters for pseudo-labelling

Initially, we observe that this method achieves higher accuracy. However, we also notice a tendency towards overfitting as indicated by the training accuracy approaching 1. To understand this behavior, let's analyze the algorithm employed. The model is initially trained on a small number of labeled samples, and as it overfits to this initial training set, it becomes less capable of recognizing images that differ significantly from the original ones. Consequently, the model learns biases that are reflected in the pseudo-labels assigned to the unlabeled data. As a result, we encounter a similar overfitting problem in this context.

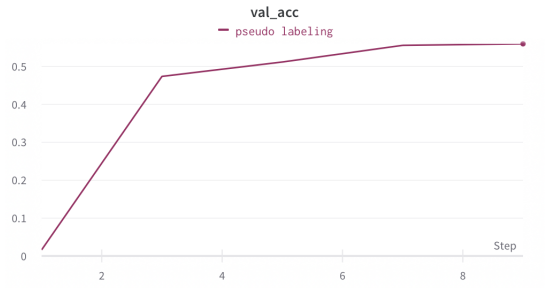

Figure 3. Accuracy on the validation set with the pseudo-labelling method

To combat this overfitting, one possible approach is to employ data regularization techniques, which can help reduce the model's sensitivity to specific biases. Additionally, ensemble learning methods can be utilized to address these biases by combining the predictions of multiple models trained on different subsets of the data. This can help mitigate the overfitting issue and improve the overall performance and generalization capabilities of the model.

3.2.3 Pretraining

The results obtained with this approach exhibit a slight improvement compared to fine-tuning. Nevertheless, the method's full potential could not be realized due to hardware constraints. The model's training was constrained to a small subset of the unlabeled dataset, resulting in a biased focus on specific details within this restricted portion of the data distribution. As a consequence, overfitting issues emerged, compromising the model's generalization capabilities.

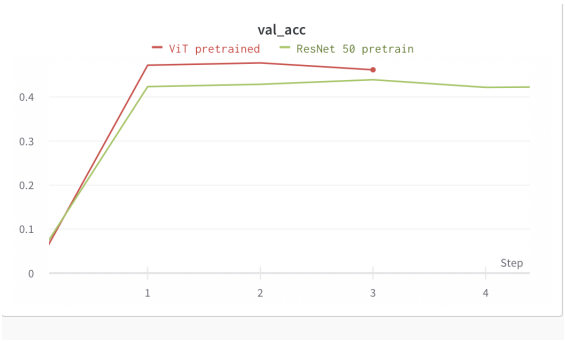

Figure 4. Accuracy comparison between pretrained ResNet and pretrained ViT

To enhance this approach, several strategies can be considered. Firstly, addressing the hardware limitations to allow for training on a larger portion of the unlabeled dataset would provide the model with a more comprehensive understanding of the data distribution and reduce bias. Ensemble learning, which involves training multiple models on different subsets of the data and combining their predictions, can also mitigate bias and enhance performance.

3.2.4 Free Match

We met the same issues with FreeMatch algorithm, indeed the FreeMatch algorithm shows some improvement compared to fine-tuning. However, it has a problem with overfitting, where the model focuses too much on specific details in a small part of the data. To make it better, we can use techniques that prevent overfitting, like adding some rules during training. Another way is to combine the predictions of multiple models trained on different parts of the data,

which helps to make more accurate predictions. These improvements can make the FreeMatch algorithm more effective and better at understanding different types of data.

Algorithm	Best Accuracy
Finetunning	48.5
Pseudo-labelling	54.1
Pretrainning	52.4
FreeMatch	54.5

Table 3. Best results for each algorithm

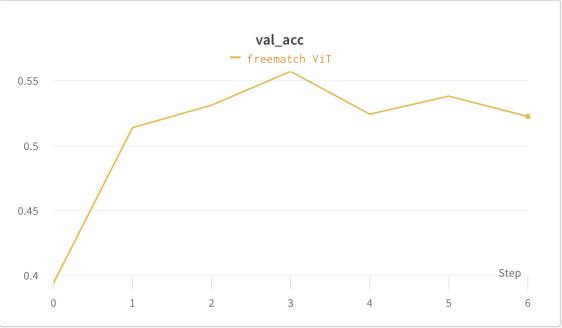

Figure 5. Accuracy the ViT model when trained with FreeMatch method

References

[1] Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning arXiv:1908.02983

[2] Unsupervised Representation Learning by Predicting Image Rotations arXiv:1803.07728

[3] FREEMATCH: SELF-ADAPTIVE THRESHOLDING FOR SEMI-SUPERVISED LEARNING arXiv:2205.07246

1

2

2