12 janvier 2026

Homework 2

AAIT (2025) Assignment 2

THOMAS Lucas

Assignment 2 – Task 1

AIM of the research

The objective of this task is to design and evaluate a learning strategy capable of leveraging partially labeled datasets while maintaining robust generalization performance.

In this context, the main challenge lies in effectively exploiting unlabeled samples without amplifying confirmation bias or propagating incorrect pseudo-labels. To address this issue, semi-supervised learning approaches based on pseudo-labeling and consistency regularization are investigated, with a particular focus on improving the reliability of pseudo-label generation and training stability.

Specifically, this research aims to :

- Establish a baseline performance using a supervised warm-up phase
- Compare the iterative Pseudo-Labelling approach against FixMatch.
- Evaluate Robustness : Assess how each method handles à large number of classes where the risk of label noise and class imbalance is significantly higher than in binary or low-class classification tasks.

By the end of this study, we aim to provide a comprehensive recommendation on the most stable and performant architecture for scaling image classifiers when labeled samples are the primary constraint.

Model

For the both task, i've tested ResNet_34 and ResNet_50, they are well-established convolutional neurals networks that offers a **good balance between model capacity, computational cost, and training stability**. They are use for both labeled and unlabeled data is appropriate and consistent with common practices in semi-supervised and noisy-label learning.

Pseudo_Label :	ResNet_34 / ResNet_50
Opimized ReMixMatch :	ResNet_34
Co-Teaching :	ResNet_34
Other Method :	ResNet_50 / ResNeXt_50

Method

Pseudo labelled

The primary methodology employed in this research is an **Iterative Pseudo-Labeling** framework (also known as Offline Self-Training). This approach was specifically inspired by the work of **Cascante-Bonilla et al. (2021)** in their paper *« Curriculum Labeling : Revisiting Pseudo-Labeling for Semi-Supervised Learning. »*

Following their findings, we adopted a multi-stage training pipeline designed to mitigate **confirmation bias** (a phenomenon where a model reinforces its own incorrect predictions) by combining strict confidence filtering with robust data augmentation.

Before engaging with unlabelled data, the model is train during an supervised Warm-up phase. This ensures that the model develops a reliable initial feature representation and a baseline classification accuracy, which is essential for generating high-quality labels in the subsequent stages.

The core of the method consists of several stages, each following a three-step cycle :

- **Pseudo-labelled generation :** The model predicts a class probability distribution for every data in unlabelled pool.
- **Confidence Thresholding and Balancing :** To ensure the quality of the new training data, a high confidence threshold (0.9) is applied. As highlighted in the inspired research, we implement a per-class cap to prevent majority classes from dominating the learning process.
- **Mixed-Batch Retraining :** Unlike traditional pseudo-labeling that trains only on the new labels, we implement a **Joint Optimization** strategy. In each iteration, the model is fed a mixed batch containing both labeled data (with weak augmentation) and pseudo-labeled data (with strong augmentation). Strong augmentation is used to force the model to learn invariant and robust features, effectively bridging the gap between standard pseudo-labelling and more advanced consistency-based methods, such as FixMatch.

Optimized ReMixMatch

To try to achieve convergence on a complex 100-class dataset, I adopted an **Optimized ReMixMatch** framework. This method represents a significant evolution in semi-supervised learning (SSL), refining the principles of **FixMatch** and **MixMatch** to maximize information extraction from unlabeled data while minimizing the propagation of incorrect labels.

Our implementation introduces a suite of **adaptive hyper-parameters** designed to carefully modulate the influence of unlabeled data as the model's confidence matures:

- **Adaptive Pseudo-Label Thresholding:** Instead of a static value, the confidence threshold evolves (Curriculum Learning), allowing the model to focus on easy samples first before incorporating more complex unlabeled data.
- **Temperature Sharpening:** By reducing the entropy of the Teacher's predictions, this technique produces "sharper" and more decisive pseudo-labels, reinforcing the model's certainty.
- **Progressive Unsupervised Weighting ():** The contribution of the unlabeled loss is gradually ramped up, ensuring that noisy initial predictions do not destabilize the pre-trained features.
- **Stochastic Regularization (MixUp & RandAugment):** We combine MixUp (to smooth decision boundaries) with RandAugment (to force the model to learn invariant features across different data transformations).
- **Optimization & Stability Features:** The use of **AdamW**, **Cosine Annealing**, and **Gradient Clipping** ensures a stable training trajectory, even with the high variance inherent in 100-class semi-supervised tasks.
- **Distribution Alignment & Temporal Ensembling:** We monitor the running class distribution to prevent class collapse and use an Exponential Moving Average (EMA) of the Student's weights to stabilize the Teacher's predictions.

Data-Augmentation

In order to improve the generalization capability of the model and to limit overfitting, a differentiated data augmentation strategy is employed. This strategy relies on the use of two distinct levels of augmentation, adapted to the type of data being used: real labeled data and pseudo-labeled data.

Weak Augmentation

So-called *weak augmentation* is applied during the warm-up phase on labeled data. Its purpose is to introduce moderate variability while preserving the semantic content of the images.

The applied transformations are as follows:

- **RandomHorizontalFlip**, simulating natural horizontal symmetries,
- **RandomCrop with padding**, introducing slight spatial variations to improve translation invariance,
- **Normalization using ImageNet statistics (mean, standard deviation)**, ensuring a consistent input distribution for the neural network.

Strong Augmentation

A more aggressive augmentation strategy is applied during the re-training phase on pseudo-labeled data. The objective is to strongly regularize the model by forcing it to produce consistent predictions despite significant visual perturbations.

The transformations used include:

- **RandomHorizontalFlip**,
- **RandomCrop with padding**,
- **RandAugment (2 operations, magnitude 10)**, which dynamically applies random transformations such as rotation, contrast adjustment, or solarization,
- **ColorJitter**, slightly perturbing brightness, contrast, and saturation,
- **ImageNet normalization**.

This combination, together with strong augmentation applied to pseudo-labeled data, enables:

- improved model robustness,
- reduced overfitting to noisy pseudo-labels,
- enhanced generalization to unlabeled data.

Data augmentation is applied consistently across all methods used in **Task 1**.

Data preparation

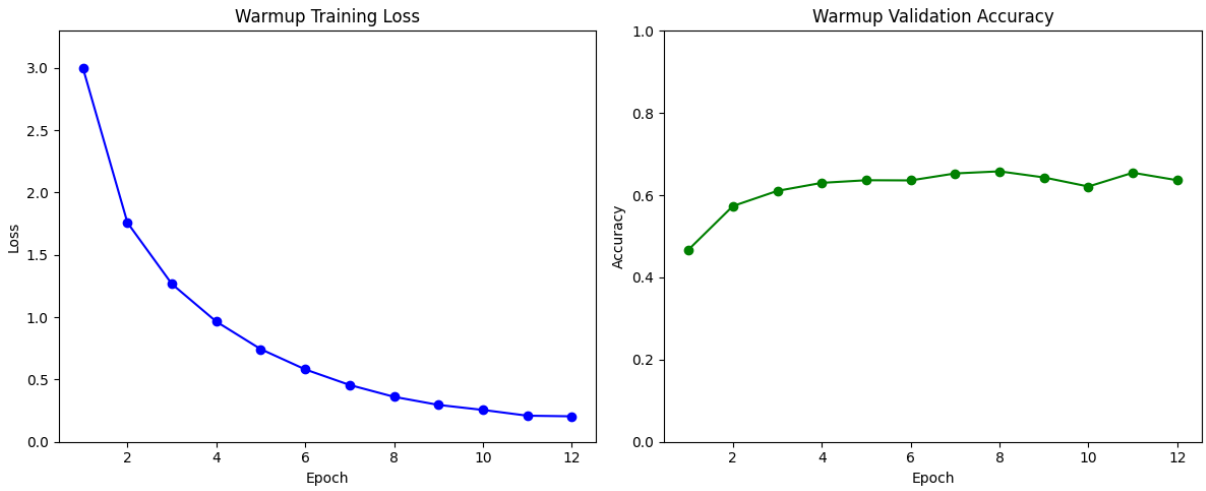
The data are remdomly split en 90/10% for trainning and validation. A seed is create for have the same separation between each train.

Results

Pseudo labelled

Warmup :

To begin with, the model was trained on labeled data for **12 epochs**.



Training was not continued beyond this point because the validation accuracy started to stagnate, and the goal was to avoid entering an overfitting regime.

Pseudo-Labelling :

For pseudo-label generation, the **warm-up model** is used in order to produce the most reliable pseudo-labels possible.

To achieve this, two predictions are performed for each image:

- one on the original image,
- one on the horizontally flipped version of the image.

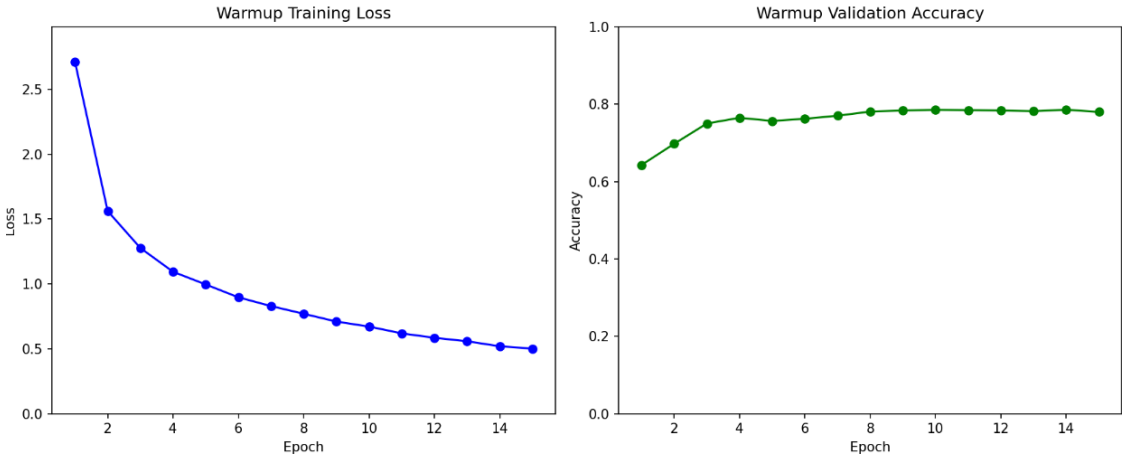
The two predictions are then averaged to obtain the final prediction. The resulting pseudo-labeled samples are subsequently added to the previously labeled dataset.

Retrain :

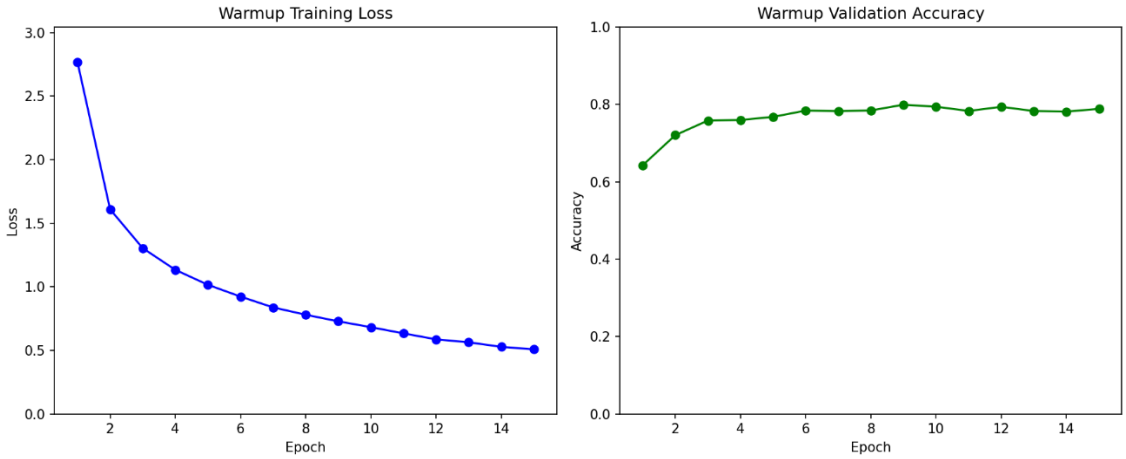
For the re-training phase, all available data are reused within a new model referred to as the **Student**. This model starts with no prior knowledge and is trained jointly on both the original labeled data and the newly generated pseudo-labeled data. At the end of the first re-training phase, a new pseudo-label generation process is performed in order to generate additional pseudo-labels, thereby increasing the number of images seen by the student model.

In this work, this process was repeated **2 times**.

Turn 1 :

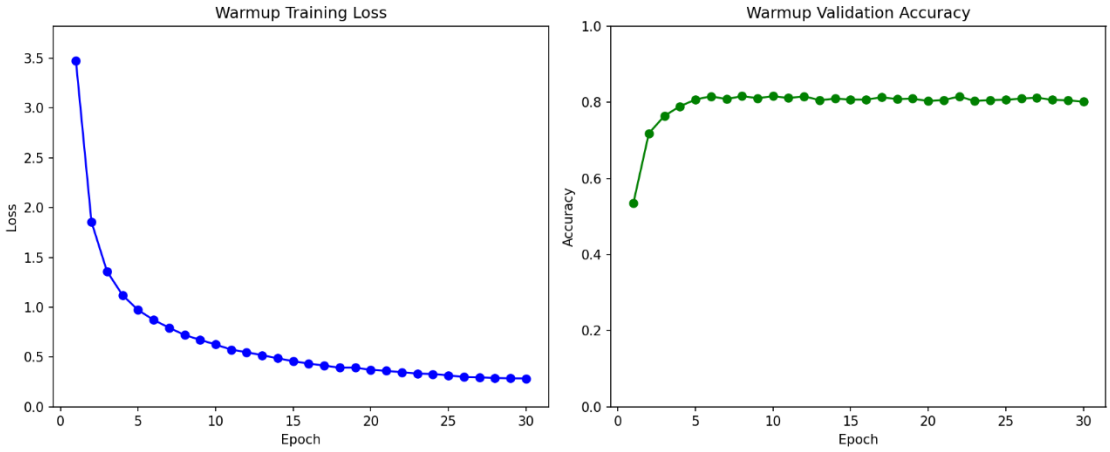


Turn 2 :

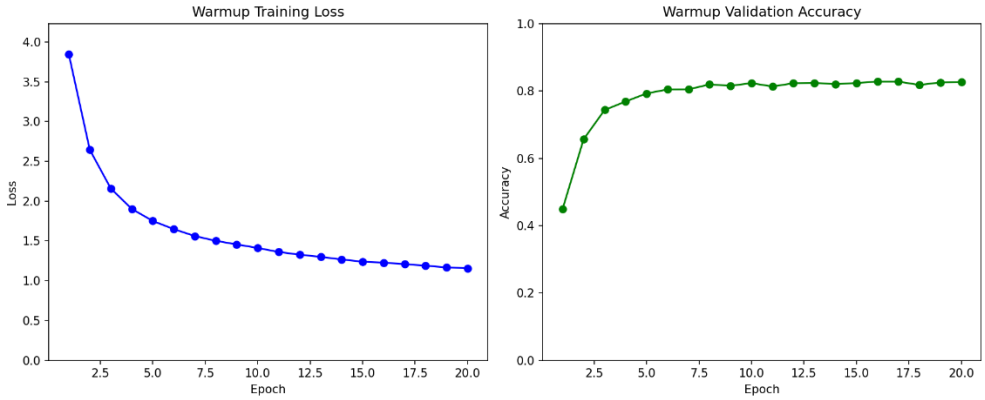


After this train, the performance doesn't improve. So I decided to update on ResNet_50 to continue.

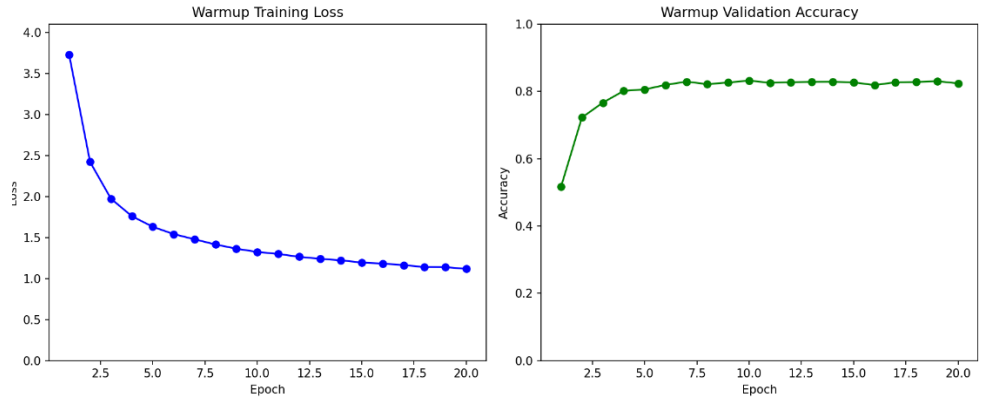
Turn3 :



Turn 4 :



Turn 5 :



Performance :

Using the ResNet-34 architecture, the model achieves a validation accuracy of 0.67 on the Kaggle evaluation set. When increasing the model capacity by switching to ResNet-50, the validation performance improves significantly, reaching 0.75.

This performance gain can be attributed to the higher representational capacity of ResNet-50, which enables the network to capture more complex visual patterns and class-specific features. These results indicate that, for this task, model capacity plays a critical role in achieving higher generalization performance, provided that sufficient regularization is applied.

Optimized Meta Pseudo Labelling

For this method, the model pre-trained on labeled data was reused as initialization in order to avoid the cost of training a new model from scratch. The data augmentation strategy remains identical to the one used during pseudo-labeling.

Hyper-parameters for first try :

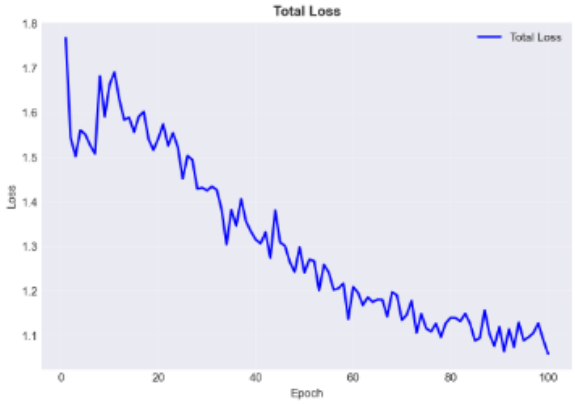
Learning_rate : 1e-4

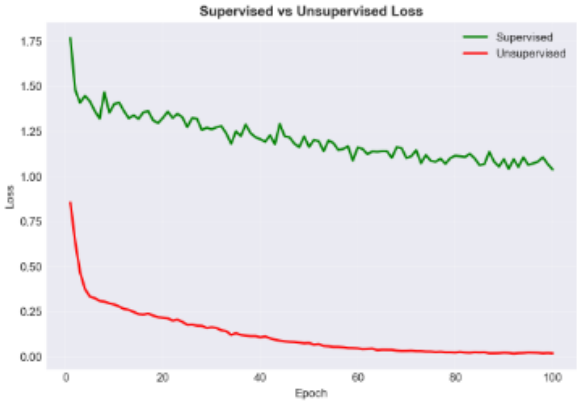
Threshold : 0.95 -> 0.70

EMA_Decay : 0.999

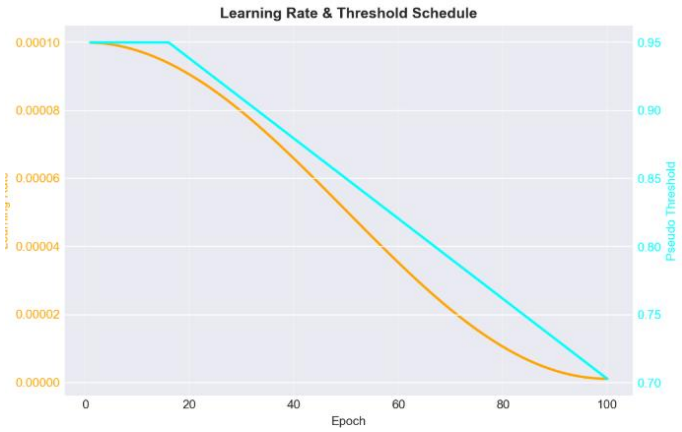
MixUp_Alpha : 0.5

Loss :





The reported loss corresponds to the **combined loss** (supervised loss + unsupervised loss). The supervised loss includes **MixUp**, which explains why the loss values remain around 1.



I used a decreasing threshold so that more and more data would be accepted as training progressed.

Hyper-parameters for Second try :

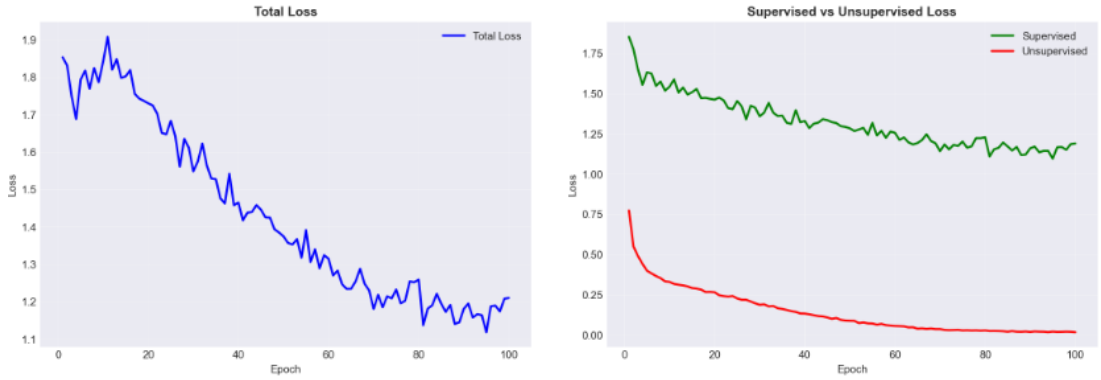
Learning_rate : 2e-4

Threshold : 0.95 -> 0.75

EMA_Decay : 0.995

MixUp_Alpha : 0.6

Loss :

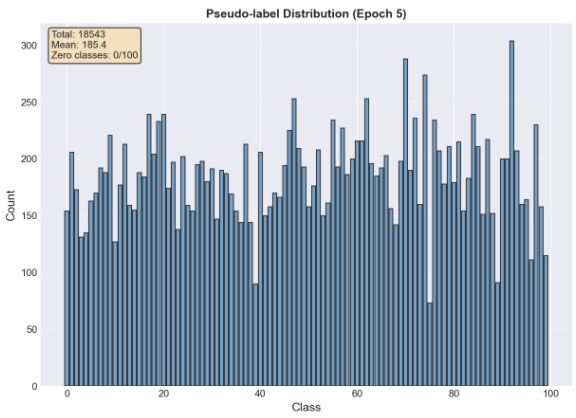


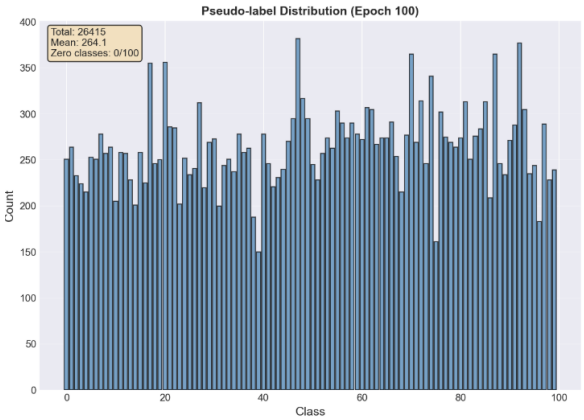
For the second training, I've tried hyperparameters a little bit more agressive. But the results are worsed than the first try.

Pseudo-labelling :

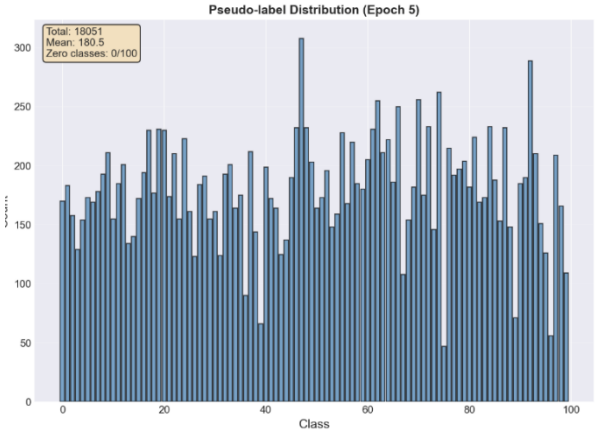
Pseudo-label generation is performed using the **Teacher model**, and only pseudo-labels with a confidence higher than **0.95** (at the beginning, **0.70/0.75** for the end) are retained. A clear improvement can be observed between the beginning of training (Epoch 5) and the end (Epoch 100), both in terms of pseudo-label quality and quantity.

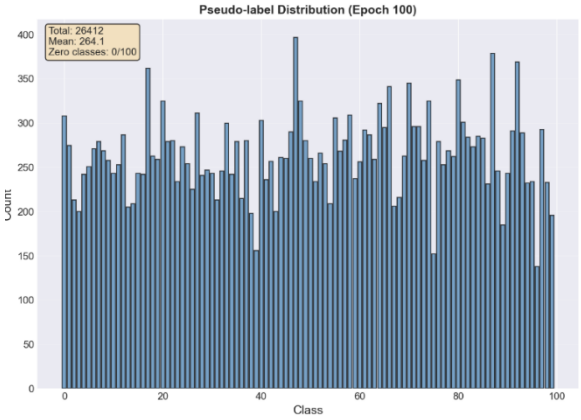
First try (0.95 -> 0.70):





Second try (0.95 -> 0.75):





In our framework, the **Teacher** model does not learn through traditional backpropagation. Instead, its weights are updated using an **Exponential Moving Average (EMA)**, also known as **Temporal Ensembling**.

$$\theta_{teacher}^{(t)} = \alpha \cdot \theta_{teacher}^{(t-1)} + (1 - \alpha) \cdot \theta_{student}^{(t)}$$

Role and Benefits:

- **Consistency and Stability:** By using a high decay rate (Alpha = 0.999), the Teacher evolves very slowly. This prevents the pseudo-labels from changing drastically between iterations, providing a "stable target" for the Student to learn from.
- **Temporal Ensembling:** The Teacher effectively acts as an ensemble of the Student's previous versions. Research in semi-supervised learning (notably the

Mean Teacher paradigm) shows that this averaged model generalizes better and is less sensitive to the noise present in individual training batches.

- **Mitigating Confirmation Bias:** Since the Teacher is always a slightly "delayed" and "smoother" version of the Student, it is less likely to immediately reinforce the Student's incorrect high-confidence predictions, thus reducing the risk of confirmation bias.

Performance :

Training with this technique initially showed promising behavior, with:

- a decreasing loss,
- an increasing number of pseudo-labeled samples.

However, despite multiple attempts, it was not possible to exceed a **validation accuracy of 0.73 on Kaggle**.

This may indicate that the proposed method is not sufficiently effective for this specific task, or that the hyperparameters and training strategy were not optimally tuned.

Conclusion

These methods allowed an in-depth exploration of semi-supervised learning techniques in order to identify effective strategies for training a model on unlabeled data. However, achieving high performance (above 0.8 accuracy) remained challenging.

Among the explored approaches, **pseudo-labeling appears to be the most effective**. With continued training, the model gradually learned a wider range of labels, including more complex ones.

Assignment 2 – Task 2

AIM of the research

The objective of Task 2 is to study learning strategies that remain robust in the presence of noisy annotations. In real-world datasets, labels are often corrupted due to annotation errors, ambiguity, or automatic labeling processes.

The goal is therefore to reduce the negative impact of incorrect labels during training by combining robust loss functions, data filtering strategies, and regularization techniques, while preserving as much useful information as possible from the original dataset.

Method

Co-Teaching

Co-Teaching is a noise-robust training strategy that simultaneously trains two neural networks with identical architectures but different initializations. At each training iteration, each network selects a subset of training samples associated with the smallest losses, under the assumption that these samples are more likely to be correctly labeled. Instead of updating itself on its own selected samples, each network uses the small-loss samples selected by its peer for parameter updates.

A time-dependent *remember rate* is used to gradually reduce the proportion of training samples considered during learning, allowing the model to discard potentially noisy labels as training progresses. This mechanism effectively prevents networks from overfitting corrupted annotations while maintaining stable learning dynamics, as originally proposed by Han et al. (2018).

Loss-based data cleaning

In addition to Co-Teaching, a **loss-based data cleaning strategy** is implemented. The approach begins with a warm-up phase during which the model is trained on the full noisy dataset. At this stage, the objective is not to achieve optimal accuracy, but to obtain a sufficiently stable model capable of estimating sample difficulty.

After warm-up, the per-sample training loss is used as a proxy for label reliability. Samples associated with the highest losses are assumed to be more likely mislabeled and are filtered out. The model is then retrained on the resulting cleaned dataset, under the assumption that reduced label noise will improve generalization.

Data

Dataset usage

The full labeled dataset was used for training without an internal train/validation split. This choice was motivated by the high level of label noise (estimated at 40% and 30% for Co-Teaching and 20% for second method), where withholding part of the data for validation could further reduce the effective learning signal. The Co-Teaching framework inherently performs dynamic sample filtering, which partially compensates for the absence of a dedicated validation set.

However, this approach limits the ability to monitor generalization during training and leads to model selection based solely on training loss, which may introduce bias. This limitation is acknowledged and considered a trade-off given the dataset characteristics.

Augmentation

To improve robustness and reduce overfitting, the following data augmentation techniques were applied during training:

- Resizing all images to 64×64 pixels
- Random horizontal flipping
- Random rotation (±10°)
- Color jittering (brightness and contrast)
- Normalization using ImageNet statistics

For validation and test data, only resizing and normalization were applied to ensure consistent and unbiased evaluation.

Results

Co-Teaching

In my training, I used several parameters :

Noise rate :

Noise rate represents the **estimated fraction of noisy labels** in the training dataset. Used as a reference to determine how aggressively to filter out potentially noisy examples as training progresses.

I’ve tried two different Noisy rate, 40% (First train) and 30% (Second train).

Retention Rate :

Specifies the fractions of examples retained as « Trusted » at each epochs, it's computed dynamically :

```
R_t_raw = 1 - min((epoch / Tk) * NOISE_RATE, NOISE_RATE)
R_t = max(R_t_raw, R_MIN)
```

Explanation :

- Gradually increases the fraction of examples considered "potentially noisy" as training progresses. Early epochs retain almost all data, since the model has not yet differentiated clean from noisy examples.
- Converts the noisy fraction into a **retention fraction**, R_T is the proportion of examples to keep.
- Enforces a **minimum retention floor**, ensuring at least R_MIN of the batch is always used, even in later epochs.

Conceptually, R(T) starts near 1.0 and gradually **decays to R_MIN**, making the training progressively more selective in the examples it considers reliable.

TK – Threshold for R(T) :

Tk determines the time scale over which the retention rate R(T) decays.

Symetric Cross Entropy Loss :

The SymmetricCrossEntropy (SCE) loss is designed for **robust learning under noisy labels**, as introduced by Wang et al. (ICCV 2019). It combines two complementary loss terms: the standard Cross Entropy (CE) and the Reverse Cross Entropy (RCE).

Loss Definition :

$$\mathcal{L}_{SCE} = \alpha \cdot \mathcal{L}_{CE} + \beta \cdot \mathcal{L}_{RCE}$$

By combining CE and RCE, SCE benefits from **both stable optimization and increased noise tolerance**. In more, the use of SymmetricCrossEntropy is particularly appropriate in this work because the dataset can contains a significant amoount of label noise and Co-Teaching relies on loss magnitude to identify clean samples. SCE provides more reliable loss values under noise.

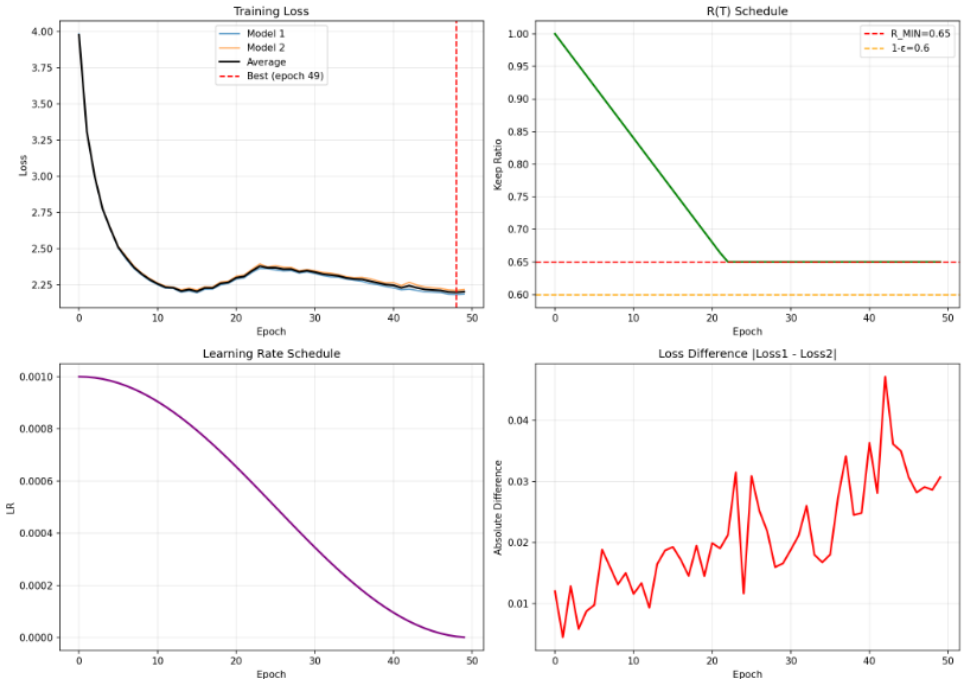
Other hyperparameters used :

Learning rate : 1e-3

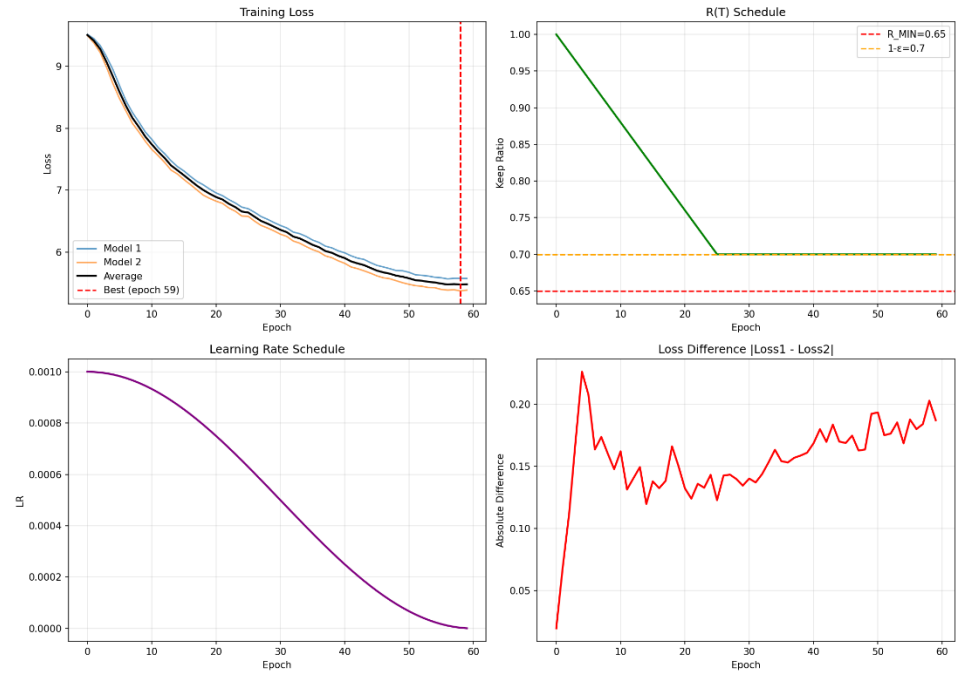
R_min : 0.65

Tk : 25

First train :



Second train :



Performance

Two training runs were conducted using a Co-Teaching strategy with noisy labels, combined with a decaying retention rate R(T) and Symmetric Cross Entropy loss. The objective was to improve robustness to label noise and generalization on the validation set.

Training 1 :

- Validation accuracy: **0.63**
- Training loss converges to a relatively low value.
- The loss difference between the two models remains moderate, suggesting **healthy disagreement**, which is desirable in Co-Teaching.
- Overall behavior indicates a **reasonable balance between learning capacity and noise filtering**.

Training 2 :

- Validation accuracy: **0.53**
- Training loss is significantly higher throughout training.
- Loss difference between the two models is much **larger** and more **unstable**.
- This suggests that the models **diverged too strongly**, likely discarding too many informative samples or failing to agree on what constitutes “clean” data.

The model learn meaningful representation and trains stably, Co-Teaching with Noise rate provides clear robustness benefits compared to naive training. However, performance is ultimately **bounded by data quality and task difficulty.** The first training run should be considered the reference configuration, while the second illustrates the limits of aggressive noise filtering.

Loss-based data cleaning

To improve robustness to noisy annotations, we integrate the **MixUp** data augmentation strategy during the warm-up and retrain phase.

MixUp generates virtual training samples by linearly interpolating both **input images** and their corresponding **labels**. Given two samples $(x_i, y_i)$and (x_j, y_j), a mixed sample is defined as:

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j$$

Implementation in our Training Pipeline

- Mixup_data randomly pairs samples within a mini-batch and generates mixed inputs.
- Two target labels are retained along with the mixing coefficient λ.
- The training loss is computed using mixup_criterion, which combines the losses of both targets proportionally :

$$\mathcal{L} = \lambda \mathcal{L}(y_a) + (1 - \lambda)\mathcal{L}(y_b)$$

This formulation allows MixUp to remain compatible with **non-standard loss functions**, such as **Symmetric Cross Entropy** (Explain in Co-Teaching parts).

Benefits in the Context of Noisy Labels

The use of MixUp provides several advantages:

- It **smooths the decision boundary**, reducing sensitivity to incorrect labels.
- It acts as a strong **regularization mechanism**, limiting memorization of noisy samples.
- It encourages the model to learn **more general and linear representations** between classes.

In the presence of label noise, this is particularly beneficial, as the model is less likely to overfit individual mislabeled samples.

Warm-Up Performance :

MixUp and loss-weighting parameters :

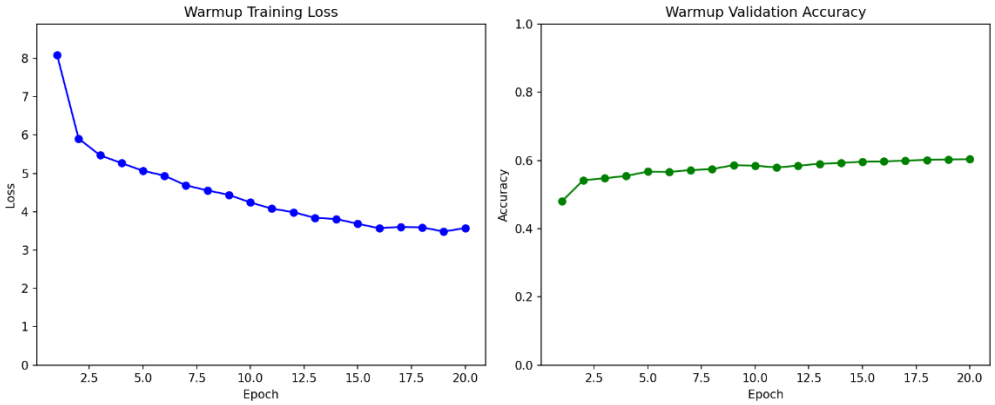
LR : 1e-3

MIXUP_ALPHA : 0.3

SCE_ALPHA : 0.1

SCE_BETA : 1.0

With this **MIXUP_ALPHA**, the model learns **stable low-level and mid-level representations** without excessive regularization. With SCE Alpha and Beta, the model avoids early memorization of noisy annotations and converges more conservatively.



Cleaning dataset :

To clean the original dataset, I used a noise rate of 20%. The new data is saved in a new CSV file. The amount of data is:

50 000 images to 40 000 images

The original dataset is cleaned each time the model progresses after retraining.

Retraining :

To perform the retraining, I ran the same algorithm as for the warm-up, with a few different parameters:

LR : 1e-3

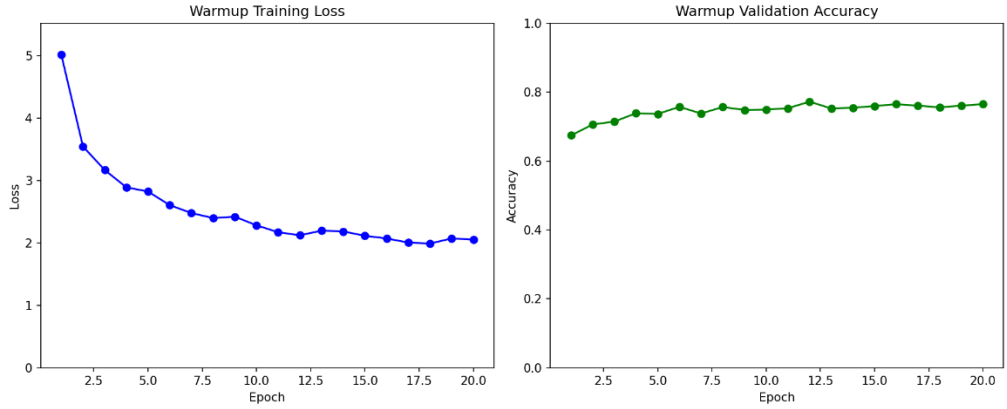
MIXUP_ALPHA : 0.4

SCE_ALPHA : 0.5

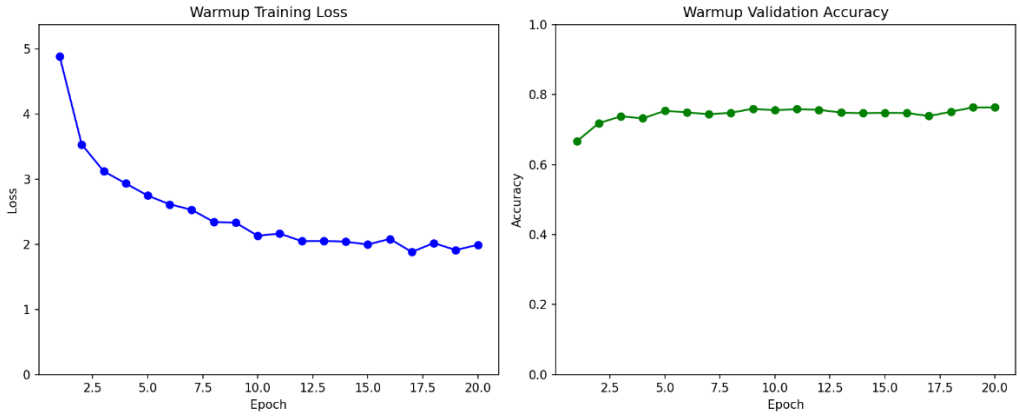
SCE_BETA : 0.5

MIXUP_ALPHA provides a **stronger regularization**, which is critical when training on **pseudo-labeled or filtered datasets** that may still contain residual noise. Once the dataset has been partially cleaned, **SCE ALPHA** and **BETA** can help the model to focus more on **class separation and accuracy**, rather than only robustness.

Turn 1 :



Turn 2 :



Performance

After two retraining rounds, the model achieved a performance of 0.76 on the Kaggle validation. With more rounds, while continuing to clean the original dataset, the model could achieve even better performance.

I can conclude that testing the **loss-based data cleaning** method is conclusive and yields good performance. However, it may be that in order to have a truly high-performing model, several retraining sessions will be necessary, which can be time-consuming.

Conclusion

This study highlights the complexity of learning under noisy supervision. While robust losses and data cleaning strategies can mitigate label noise, their effectiveness strongly depends on hyperparameter choices and dataset characteristics.

In particular, aggressive filtering strategies may degrade performance by removing hard but correctly labeled samples. Future work could explore adaptive thresholds, iterative cleaning procedures, or hybrid approaches combining sample reweighting and semi-supervised learning to better balance robustness and data efficiency.

In addition, I implemented a DivideMix method, which did not work at all, but I left it in the code (DivideMix.py) anyway.

Link

Task-1 :

https://cdn.aaai.org/ojs/16852/16852-13-20346-1-2-20210518.pdf

https://arxiv.org/pdf/1911.09785

Task-2 :

https://proceedings.neurips.cc/paper_files/paper/2018/file/a19744e268754fb0148b017647355b7b-Paper.pdf

https://openreview.net/pdf?id=HJgExaVtwr

GitHub :

https://github.com/eXor6orks/AAIT-HW2.git