CHALLENGE MODAL INF473V

Marc Boëlle, Tess Breton

1. INTRODUCTION

The challenge aimed at classifying images from a synthetic dataset containing 48 classes. We were provided with a small amount of labeled data (15 images per class) and a large pool of unlabeled data.

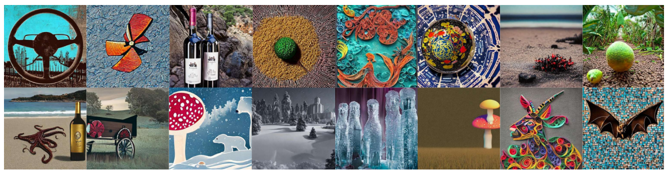

Fig. 1. Images from the dataset

Although the labels were not all semantically close to those from ImageNet, our main strategy was to use Transfer Learning and finetune a model pretrained on ImageNet.

2. BASIC TRAINING ON LABELED DATA ONLY

2.1. Training results

The first thing we did was train a ResNet50, pretrained on ImageNet, using labeled data only. As expected, the model quickly overfits. The best cross-validation accuracy we got is 0.36.

We changed the model, testing models from Timm by freezing all layers except the last fully-connected one (see Table 1). These models also overfit quickly, but some have a better validation accuracy than ResNet50. We chose ViT-Base-Patch32-CLIP for the rest of the study.

Model	Best validation accuracy
resnet50	0.3611
volo-d5	0.2986
xcit-large-24-p16	0.3194
vit-base-patch32-clip	0.4097
vit-huge-patch14-clip	0.4375

Table 1. Comparison of validation accuracy for different models

2.2. Data Augmentation

To try and prevent overfitting on the labeled data, we used data augmentation during training. We first used PyTorch RandAugment but soon realized it was too strong for the model to keep learning properly. We then decided to create a transform using PyTorch transforms that seemed logical given the pool of images provided : Color Jitter, Random Horizontal Flip...

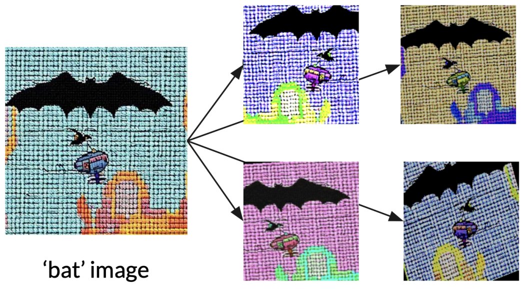

Fig. 2. Data Augmentation performed on a 'bat' image

Unfortunately, validation accuracy did not improve when data augmentation was used, and even decreased. It seems as if the transformed data was too different for the model to learn from it. We did not manage to find the right hyperparameters allowing learning without overfitting.

3. FIXMATCH

To leverage our large pool of unlabeled data, our first idea was to use FixMatch (see [1]), already presented by several teams during the previous one-week challenge.

FixMatch consists in using a weakly-augmented version of an unlabeled image to get a pseudo-label for a strongly-augmented version of the same image, if the model is confident enough about its prediction. The loss between the pseudo-label and the prediction of the model on the strongly transformed image is then added to the usual loss on the labeled batch (comparing groundtruth label and prediction), and the model trained using backpropagation.

The goal is to achieve **consistency regularization**, leveraging data continuity : two images that are similar should get the same label.

It is said in [2] that FixMatch can perform well on datasets such as CIFAR-10, but that it struggles with more realistic images when the labeled dataset is small. Indeed, we did not get any major accuracy improvement when using FixMatch : validation accuracy did not go higher than 0.4.

4. CLIP BY OPENAI

Following the advice of another team, we tried using CLIP by OpenAI (see [3]) to classify our images. As a **zero-shot model** (not trained on any labeled data), it reached an accuracy of **0.53**.

4.1. Failure case analysis

To understand better the performances of CLIP on our dataset, we made a **heatmap** using the train dataset to see on which classes he did not perform well. As expected, it performs well on the classes with a clearly interpetable name ('pinwheel'), but it struggles more on complex labels ('Salvelinus fontinalis' or 'ethyl alcohol' for instance). Moreover, some classes often get predicted when they should not ('pinwheel', 'cupola', 'vintage'or 'veloute').

We also noticed that the dataset contains classes with very similar statements, such as "toadstool" and "Entoloma lividum", two types of mushroom, or "gosling" and "duckling". In this case, images may be predicted in the wrong class.

4.2. Models and prompts

We found out in [4] that CLIP's performance depended highly on the prompts it was given. While in our first try we used 'a photo of a $\{$class$\}$' as prompts, we managed to get +1 point of accuracy just by changing it to 'a stylized $\{$class$\}$'. Accuracy also depends on the model used as image encoder. We got +2 points of accuracy on the labeled dataset using RN50x64, but it did not seem meaningful on the test dataset (we did not improve our Kaggle score).

We then decided to finetune CLIP (freezing all but a few layers), but it did not work better than the zero-shot version. We later read in [5] that it could be better to finetune the prompts, given that CLIP had been trained on 400M+ images. Finetuning the prompts seemed like a good idea but we lost interpretability and it seemed too technical for us, so we decided not to go any deeper any into it.

5. PSEUDO LABELING USING CLIP

5.1. Basic pseudolabeling

Because of the quick overfit on Timm models when using labeled data only, we decided to leverage the predictions made by CLIP on unlabeled data, since it can be very confident and accurate. We took a label as groundtruth when its probability exceeded 0.98. We obtained an average validation accuracy of 0.517 (see Figure 3).

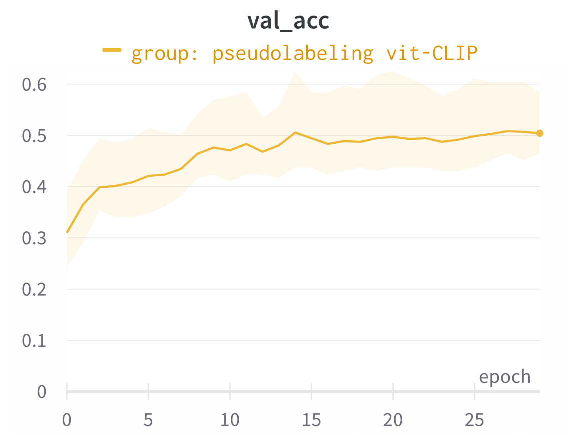

Fig. 3. Training ViT-Base-Patch32-CLIP on both labeled data and data pseudo-labeled by CLIP

To make sur that we were not misguided, we used labeled data to see how accurate CLIP predictions were over our confidence threshold : on the train dataset, the accuracy was 1.0. Yet it is important to mention that some classes never reached the threshold, which prevents the model to learn uniformly on all classes using pseudolabels.

We also struggled to find the right hyperparameters for our pseudolabeling : we had to choose a batch-size ratio and a loss ratio between labeled and unlabeled data, a confidence threshold for pseudo-labeled. We found the threshold particularly important to avoid **confirmation bias** (see [6]) : we want to be confident about the pseudo-labels we accept, but we also want the threshold to be low enough so that we do not always accept the same pseudo-labels (we need different classes to teach the model). We tried different schemes (constant parameters, linear interpolation on epochs) but none of them seemed to outperform our first shot. We also tried using the Sweep tool from WeightAndBiases but it did not give much better results.

5.2. Class-aware confidence threshold

To address the unequal distribution of classes among pseudolabels, we tried implementing a method presented in [2] : the idea is to use a class-aware threshold, adjusting its value to the proportion of images confidently pseudo-labeled in a particular class. Using the notations from the paper [2] : r_c the proportion of class c in the dataset, p_c the proportion of images that are confidently labeled into class c among all images, and τ_c the confidence threshold on class c.

$$\begin{cases} \tau_c^{t+1} = \tau_c^t + \varepsilon \, \text{sign}(p_c^{t+1} - r_c) \\ p_c^{t+1} = \alpha p_c^t + (1-\alpha) p_c^{\text{batch}} \end{cases}$$

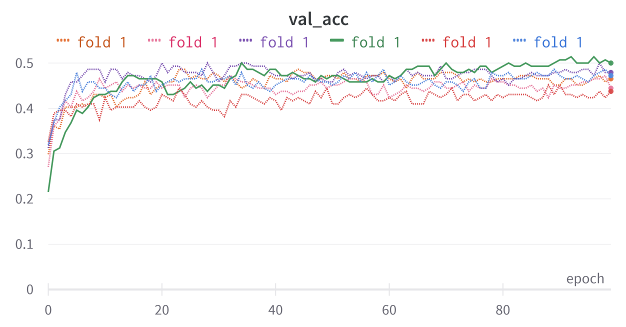

Fig. 4. Wandb sweep on labeled and unlabeled batch size, threshold and loss weights

α and ε are hyperparameters, for which we kept the paper's values : $\alpha = 0.9$ and $\varepsilon = 0.001$. Theoretically, this method would avoid learning too much from preponderant classes, and would allow for better learning of harder classes. Yet again it did not give great results on our dataset.

6. USING DIFFERENT MODELS TO MAKE PREDICTIONS

Since we observed that some classes were very similar and were sometimes predicted instead of the other, we came up with the idea of grouping them into a superclass and using a Mixture of Experts as described in [7]. The principle is as follows: we use a "gating network" which redirects to "expert models" that choose between a fraction of the subclasses.

6.1. Mixture of Experts with CLIP only

Initially, we selected similar classes as superclasses and grouped them together under a common name. For example, we combined "duckling", "peahen", "gosling" and "tragopan" into the superclass "bird". We left the classes where CLIP didn't make any errors as is.

Because of the lack of labeled data, we decided to use CLIP as a gating network, then CLIP again to discriminate within a superclass.

However, it did not work as well as expected because of the bias introduced in the choice and name of superclasses. By comparing the heatmap, we find that grouping into superclasses has sometimes increased confusion between classes. Plus, the choice of prompts is very important for CLIP. This may explain why we only reached an accuracy of 0.5.

6.2. Mixture of Experts with CLIP and a ResNet50 as a gating network

We had the idea of a second method based on the mixture of experts: use a binary classifier to separate the classes well predicted by CLIP from those poorly predicted, then use another type of classifier to differentiate the poorly predicted classes. We split the dataset into 2: the 9 classes least well classified by CLIP and the 39 others. For the binary classifier, we used a resnet50. However, when we trained it, the accuracy plateaued at 0.8. We understood that the classifier was systematically predicting the superclass with the 39 classes, because this allowed it to obtain such accuracy easily, and there was no learning involved.

7. CONCLUSION

The conclusion of our work is that none of our models managed to outperform zero-shot CLIP. The complexity of the dataset made it very hard to leverage unlabeled data confidently. Understanding better each class and its links to other classes could help identify ways to improve performance.

8. REFERENCES

[1] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," 2020.

[2] Thomas Lucas, Philippe Weinzaepfel, and Gregory Rogez, "Barely-supervised learning: Semi-supervised learning with very few labeled images," 2021.

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever, "Learning transferable visual models from natural language supervision," 2021.

[4] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu, "Learning to prompt for vision-language models," *International Journal of Computer Vision (IJCV)*, 2022.

[5] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu, "Conditional prompt learning for vision-language models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[6] Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness, "Pseudo-labeling and confirmation bias in deep semi-supervised learning," 2020.

[7] Xiaolei Huang Stephen T.C. Wong John Volpi James Z. Wang Kelvin Wong Yanglan Ou, Ye Yuan, "Patcher: Patch transformers with mixture of experts for precise medical image segmentation," 2022.