Computer vision - Project 1

Semi-supervised learning
CIFAR-10 250 classification

Brice Tayart

June 4, 2020

1 Introduction

The subject of this project is semi-supervised classification of images. Supervised classification algorithms have achieved outstanding classification performance on dataset such as CIFAR-10 or ImageNet. This is done by learning models on tens of thousands of labeled examples. Manually annotating *real life* datasets The goal is to classify the CIFAR-10 image dataset with only 250 labeled images for training and 2000 for cross-validation. The remaining 47750 images are available without labels.

2 Bibliography

A list of methods for semi-supervised classification providing state-of-the-art performance on CIFAR10-150 was found on `https://paperswithcode.com/sota/semi-supervised-image-classification-on-cifar-6`, notably MixMatch[1], ReMixMatch[2], FixMatch[5], EnAET[6]. Also notable is UDA[7].

3 Experiment with FixMatch

The initial idea for the project was to replicate FixMatch. It has the highest reported accuracy. It was described as dependant on fewer hyperparameters than other methods and would thus need less testing. The paper precisely described the experimental setup and hyperparameters.

The principle is rather simple: take a batch of B labeled images, apply a weak augmentations and compute a supervised classification loss (cross-entropy). Take a larger batch of μB unlabeled images and apply a weak and a strong augmentation. The weakly augmented images are classified, which gives a pseudo-label with some confidence. When that confidence exceeds a threshod, the pseudo-label is used as a target to compute another cross-entropy loss on the strongly augmented images. This gives an unspervised loss that is aggregated with the supervised losses with a weight λ and the sum is minimized through Stochastic Gradient Descent.

The hyper-parameters are: the optimizer parameters, the size of the supervised and unsipervised batches (given by B and μ), the threshold for pseudo-labelling (set to 0.95) and the weight λ.

3.1 Datasets

The CIFAR-10 train set contains 50000 labeled images split in 10 classes. For each class, the first 25 images were aggregated into the *labeled train set*, the next 100 images into the *cross-validation* set and the remainder was stripped of labels and aggregated into an *unlabeled set*.

3.2 Augmentation

A weak and a strong augmentation transform were used. The weak augmentation simply consists in an horizontal flip with probability 0.5, a Cutout with 50% some probability (i.e. a 8x8 patch contered on a random pixel is replaced by the dataset average color), followed by a translation of up to 1/8th of the image width and height (i.e. 4 pixels). The strong augmentation contains a Cutout with 50% some probability followed by a variation of RandAugment[3]. RandAugment applies N transforms sampled from a list of 14 (including rotations, shears, translations, color and brightness adjustments, etc), with a magnitude set according to a schedule. Here, the magnitude is left as random rather than chosen as an hyperparameter.

3.3 Model

A neural network broadly similar to a Wide-ResNet was initially used. After reading more carefully the the WRN paper[8] and going through the code, an actual WRN was used. The choice there was to use (3,3) blocks (i.e. a pair of 3×3 convolutions) rather than a (1,3,1) bottleneck as it has better reslts in the paper.

One of the limitations was memory, as only 4GB were available on the GPU. This limited the batch-size to 64. Larger batches could be simulated by accumulating the gradient and calling optimize.step(). This is not quite exactly the same: the descent direction is averaged over all sub-batches, but batch-normalization is done by sub-batches of 64 rather than in a larger batch. Also, the batch-normalization running statistics change between sub-batches (pseudo-labels are computed in evaluation mode). Finally, there may be significant differences between the statistics of the supervised, weakly augmented batch, and those of the unsupervised, strongly augmented one.

One quick test was made with EvoNorm-S0[4] in place of thu ubiquitous BN-ReLU. It is more akin to group normlization (which solves the problem of varying batch statistics), independant of batch size and descibed as outperforming both. However, no noticeable difference were seen after a few epochs of training, and it made the training slower and more memory intensive. Training was interrupted and it was not used again.

3.4 Testing with FixMatch

Testing with FixMatch was quite difficult and was extremely dependant on the hyperparameters: initial training rate, weight decay and balance between supervised and unsupervised loss. Luckily, a "sweep spot" that let the model achieve 70% accuracy on the crossvalidation set was found early into the project. Interestingly, the model had learnt to recognize only 9 classed, the last one never being predicted. However, changes in any of these parameters in either direction would severly lower the accuracy - usually down to around 30%, what is achieved when training on the 250 labeled examples alone.

When attempting to use other network architectures, convergence to an accurate model was not achieved. The best other result was 49.5%, with a model that learnt to recognize only 7 classes.

As far as I understand, there were two modes of failure. The first one is a quick convergence towards a severely overfitted model that has a very low supervised loss. Such model, however, is not able to classify

unlabeled examples with high confidence, and since the confidence never exceeds the threshold, there are simply no unlabeled examples to compute an unsupervised loss with. And without consistency regularization, the model never generalizes.

The other failure mode, however, happens when the model classifies unlabeled examples with a wrong label but high confidence. This lets wrong pseudo-labels enter into the unsupervised loss calculation. In this case, a large proportion of images is classified with a confidence exceeding the 0.95 threshold during training (e.g. 60-100%). The accuracy computed on the cross-validation set is much lower (e.g. 30%). In that case, we can expect that most of the pseudo-labels are wrong...

A last - more generic - pitfall of the consistency regularization is that the highest consistency is achieved when the model always give the same prediction with the highest confidence (e.g. 100% probability of class 0 regardless of input). The supervised and unsupervised loss need to be carefully balanced.

3.5 UDA

In order to avoid issues with the threshold, another attempt was made with UDA[7]. UDA slighly changes the unsupervised loss: the weakly- and strongly- augmented images are classified and we seek to minimise the Kullbach-Leibler divergence between these probability distributions. It also relies on several tricks that are not all described in details in the paper:

- Training Signal Annealing: when computing the supervised loss, records classified with a confidence that exceeds a threshold are dropped. The threshold is progressively incresed towards 1 durin training. The goal is to limit overfitting by not trying to classify the small supervised dataset with a high confidence early into the training. This is done only at the end, when the consistency regularization has already forced the model to generalize well.

- A softmax with a low temperature is used to compute the target distribution for the KL loss. This tends to increase the confidence of the predicitions in the target distribution: we want the distributions to be consistent, but also to point towards a class with some confidence.

- Images that do not give a confident prediction may also be dropped from the KL divergence loss.

There are more hyper-parameters, related to TSA, temperature and confidence threshold.

Hyper-parameter tuning once again proved very tricky. Accuracy most of the time stagnated around 40% on a validation set vs above 95% on the train set (showing that consistency regularization failed to let the model generalize well). Tuning λ to higher values led to a loss imbalance, and a convergence of the model to constant prediction of a class (spotted when accuracy drops to 10%), while the unsupervised loss did not show a decreasing trend. Training was interrupted when this happened.

4 Results and Conclusion

4.1 Results

The best accuracy achieved was slightly 69.80% on CIFAR test set. It was quite frustrating that this result was achieved early in testing and could never be reproduced even after extensive search for good paramters.

4.2 Conclusion

A big difference between the code and the method of the published papers was the necessity to split batches of data into sub-batches of limited size. Perhaps using larger batches on a model with less parameters would have helped. Also, working with composite losses, sometime triggered by thresholds, is quite complex as what is actually measured changes from batch to batch. It is difficult to see if the training is stuck or making progress. Finally, testing on a single GPU is extremely time-consuming. Training takes several hours, a test often needs to be let running for the night before results can be obtained. Hyperparameter tuning needs many of these runs. I often stopped training early on what looked unpromising results and, retrospectiveley, good sets of parameters that simply need time to converge may have been overlooked.

References

[1] David Berthelot et al. "MixMatch: A Holistic Approach to Semi-Supervised Learning". In: *CoRR* abs/1905.02249 (2019). arXiv: 1905.02249. URL: http://arxiv.org/abs/1905.02249.

[2] David Berthelot et al. "ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring". In: *arXiv e-prints*, arXiv:1911.09785 (Nov. 2019), arXiv:1911.09785. arXiv: 1911.09785 [cs.LG].

[3] Ekin D. Cubuk et al. "RandAugment: Practical automated data augmentation with a reduced search space". In: *arXiv e-prints*, arXiv:1909.13719 (Sept. 2019), arXiv:1909.13719. arXiv: 1909.13719 [cs.CV].

[4] Hanxiao Liu et al. "Evolving Normalization-Activation Layers". In: *arXiv e-prints*, arXiv:2004.02967 (Apr. 2020), arXiv:2004.02967. arXiv: 2004.02967 [cs.LG].

[5] Kihyuk Sohn et al. "FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence". In: *arXiv e-prints*, arXiv:2001.07685 (Jan. 2020), arXiv:2001.07685. arXiv: 2001.07685 [cs.LG].

[6] Xiao Wang et al. "EnAET: Self-Trained Ensemble AutoEncoding Transformations for Semi-Supervised Learning". In: *arXiv e-prints*, arXiv:1911.09265 (Nov. 2019), arXiv:1911.09265. arXiv: 1911.09265 [cs.CV].

[7] Qizhe Xie et al. "Unsupervised Data Augmentation for Consistency Training". In: *arXiv e-prints*, arXiv:1904.12848 (Apr. 2019), arXiv:1904.12848. arXiv: 1904.12848 [cs.LG].

[8] Sergey Zagoruyko and Nikos Komodakis. "Wide Residual Networks". In: *arXiv e-prints*, arXiv:1605.07146 (May 2016), arXiv:1605.07146. arXiv: 1605.07146 [cs.CV].