Semi-Supervised Image Classification

Alessandro Zinni, Filippo Momesso, Gabriele Padovani, Thomas De Min
University of Trento

***Abstract*—Semi-supervised image classification is a machine-learning task in which a model is trained using a combination of labeled and unlabeled data. This paper consists of a high-level survey of Semi-supervised image classification literature and expands on its main theoretical and practical challenges, providing a taxonomy of the most popular semi-supervised learning algorithms.**

I. Introduction

Semi-supervised image classification is nowadays a hot research topic. The objective is to tackle the major issue of Supervised Learning: the scarcity of labeled data. Annotated examples are expensive since they require lots of labor hence it is not always feasible to employ supervision. For this reason, the research community has focused on finding techniques that allow learning of feature representation from big datasets, with just a fraction of their information annotated. Semi-supervised image classification aims at training a classifier from both the labeled and unlabeled data, in such a way as to improve the baseline represented by the supervised classifier trained only on the labeled data [16].

II. Background

Formally, Semi-supervised image classification can be stated as follows: Let $X = \{X_L, X_U\}$ be the entire dataset. $X_L = \{x_i\}_{i=1}^{L}$ refers to the annotated subset of X. Labels of X_L are denoted as $Y_L = (y_1, ..., y_L)$, where each $y_i \in \{1, ..., K\}$. On the other hand, $X_U = \{x_i\}_{i=1}^{U}$ represents the unlabeled subset of X [16]. Since the whole point of SSL is to enable learning with a small set of labeled examples, we generally assume that $L \ll U$ [16]. The optimization problems presented are usually composed of three parts: supervised loss, unsupervised loss, and a regularization term.

$$\min_{\theta} \underbrace{\sum_{x \in X_L, y \in Y_L} \mathcal{L}_s(x, y, \theta)}_{\text{supervised loss}} + \alpha \underbrace{\sum_{x \in X_U} \mathcal{L}_u(x, \theta)}_{\text{unsupervised loss}} + \beta \underbrace{\sum_{x \in X} \mathcal{R}(x, \theta)}_{\text{regularization}} \tag{1}$$

where $\mathcal{L}_s$ denotes the supervised loss (e.g. Cross-Entropy loss), $\mathcal{L}_u$ the unsupervised loss and $\mathcal{R}$ denotes the regularization loss (e.g. Consistency loss) [16].

III. Approaches

In this work, three major methodologies will be analyzed: *Pseudo labeling*, *Consistency Regularization*, and *Pseudo Labeling with Consistency Regularization*. Although they will not be covered, the literature also counts *Graph*-based and *Generative*-based methods. These are excluded from this report since they are considered to be less popular. The most popular approaches for each methodology are explained in the following sections. Part of them have been faced during lectures, thus a more in-depth analysis was conducted. All the others, on the other hand, are included in most surveys and code bases.

A. Pseudo Labeling

This first category of methods relies on high-confidence label predictions. If the model outputs a high-confidence prediction for a given unlabeled sample, then the pseudo-label is coupled with the input, yielding an annotated data point. Consequently, all samples with high confidence are used with the annotated dataset to compute the supervised loss (first part of (1)).

a) Entropy Minimization: To avoid the decision boundary crossing high-density regions, an entropy minimization term is appended to the supervised classification loss, so that it acts as a regularizer [8]. For unlabeled data this term is defined as follows:

$$\mathcal{R}(X_U) = \sum_{i=1}^{|X_U|} h_{\theta,\phi}(x_i) \log(h_{\theta,\phi}(x_i)) \tag{2}$$

where $h_{\theta,\phi} = g_\phi \circ f_\theta$, having f_θ feature extractor and g_ϕ linear classifier. x_i is the input sample.

b) Tri-Net: It uses disagreement between three Deep Neural Networks to mold the different datasets used for learning [3]. The deep architecture can be split into an initial common module f_θ, and three different secondary sections $g_{\phi^1}, g_{\phi^2}, g_{\phi^3}$. During training, each module is used to predict a pool of data. If two modules agree on the classification of a sample, the newly labeled information is added to the labeled set of the module which predicted the wrong answer. Let X'_L be the augmented labeled set. This third module is then refined with the augmented training set.

Training is aided by output smearing, which involves perturbing the output to increase generalization. The complete loss is:

$$\begin{aligned}\mathcal{L} = \frac{1}{|X'_L|} \sum_{i=1}^{|X'_L|} \big[& L_{CE}(h_{\theta,\phi^1}(x_i)), \hat{y}^1) + \\ & + L_{CE}(h_{\theta,\phi^2}(x_i)), \hat{y}^2) + L_{CE}(h_{\theta,\phi^3}(x_i)), \hat{y}^3) \big]\end{aligned} \tag{3}$$

where L_{CE} is cross-entropy loss and $\hat{y}^v$ are pseudo labels inferred as the majority of votes from the three models. To do so, given a new unseen sample, the posterior probabilities are averaged into a single prediction.

B. Consistency Regularization

These methods apply a consistency regularization term to the final loss. The regularization term forces the model to produce similar outputs to different perturbations of the same datapoint [16].

a) Π-Model: During training, each image x_i is evaluated twice. This results in two prediction vectors z_i and $\tilde{z}_i$. Because of dropout and additional random augmentations, two evaluations of the same input x_i, with the same set of weights θ, yield a difference between z_i and $\tilde{z}_i$ [9]. This difference can be seen as an "error in classification" [9] since it is desirable to get the same output given the same input and weights. The goal of Π-Model is the minimization of the discrepancy between z_i and $\tilde{z}_i$. The loss function is composed as follows:

$$\begin{aligned}\mathcal{L} = \frac{1}{|X_L|} \sum_{x \in X_L, y \in Y_L} \mathcal{L}_{CE}(y, h_{\theta,\phi}(x)) + \\ + w(t) \frac{1}{|X|} \sum_{x \in X} (f_\theta(x) - \widetilde{f_\theta(x)})^2\end{aligned} \tag{4}$$

where $w(t)$ is a time-dependent weighting function that balances the contribution of the MSE loss. $w(t)$ starts from 0 and increases following a Gaussian curve during the first 80 training epochs [9].

b) Temporal Ensembling: Slightly modifies Π-Model to make it more efficient [9]. Instead of forwarding each image twice, it aggregates feature representations from previous network evaluations into an ensemble prediction. To do so, an auxiliary matrix must be kept in memory. Let Z be the aforementioned matrix, and Z_i be the vector representation of image x_i. At the end of each epoch, Z is updated by means of an exponential moving average: $Z \leftarrow \alpha Z + (1-\alpha)z$, where z is the matrix of outputs for the current epoch. Due to dropout and additional random augmentations, Z contains a weighted average of the outputs of an ensemble of networks [9]. In order to get $\tilde{z}$ for the current training epoch, it is sufficient to compute $\tilde{z} \leftarrow Z/(1-\alpha^t)$, with t current epoch. This becomes necessary since Z has a startup bias which must be corrected [9]. $\tilde{z}_i$ substitutes $\widetilde{f_\theta(x)}$ in the loss function.

Temporal Ensembling is twice as fast as Π-Model and achieves better accuracy [9]. Its main drawback is the necessity to store the matrix Z, which can become troublesome with big datasets. Moreover, since each target is updated only once per epoch, the learned information is incorporated into the training process at a slow pace. The larger the dataset, the longer the span of updates [13].

c) Mean Teacher: Overcomes the limitations of Temporal Ensembling by averaging model weights instead of model parameters, increasing accuracy in the process [13]. A teacher model is employed, whose parameters are updated following an exponential moving average of the weights: $\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t$ [13]. The loss function used is the same as those of Π-Model and Temporal Ensembling. However, the second term slightly changes since $\tilde{z}_i$ is obtained by means of the teacher network, thus $f_{\theta'}(x)$. Performances are similar to those of Temporal Ensembling.

d) VAT - Virtual Adversarial Training: Makes the model more robust to perturbation of the signal (i.e. the image) in the *virtual adversarial direction*. The virtual adversarial direction is the direction of the perturbation that mostly deviates from the currently inferred output distribution [11]. The Local Distributional Smoothness (LDS) is the divergence-based distributional robustness of the model against virtual adversarial direction (i.e. how much the model is robust to perturbations in the virtual adversarial direction):

$$LDS(x) = D\left[h_{\hat{\theta},\hat{\phi}}(x), h_{\theta,\phi}(x + r_{\text{vadv}})\right] \tag{5}$$

$$r_{\text{vadv}} = \underset{r; \|r\|_2 \leq \epsilon}{\text{argmax}} D\left[h_{\hat{\theta},\hat{\phi}}(x), h_{\theta,\phi}(x + r)\right] \tag{6}$$

where $x \in X$, $D[p, p']$ is non-negative and measures the divergence between two distributions. Since the true distribution of the output label $q(y \mid x)$, with $x \in X_U$, is unknown, $h_{\theta,\phi}(x)$ is used instead. The proposed regularization term becomes:

$$\mathcal{R}(X) = \frac{1}{|X|} \sum_{x \in X} LDS(x) \tag{7}$$

C. Pseudo Labeling with Consistency Regularization

This last category is a particular instance of *Hybrid* method. Hybrid methods combine ideas from different methodologies to improve performance. In this case, methods that combine Pseudo Labeling with Consistency regularization will be explored.

a) FixMatch: With this approach, weakly augmented unlabeled samples are passed through the model to obtain a pseudo label $\hat{p}_i = h_{\theta,\phi}(\alpha(x_i))$, with $\alpha(\cdot)$ stochastic function that returns a weak augmentation of the input. If confidence $c_i = \max\{\hat{p}_i\}$ is above a fixed threshold, the distribution is then converted to the one hot encoding counterpart. The model is then trained by maximizing the similarity between the pseudo-label and the prediction on the strongly augmented version of the sample [12]. The unsupervised loss is:

$$\mathcal{L}_U = \frac{1}{|X_U|} \sum_{i=1}^{|X_U|} \mathbb{1}\{c_i \geq \tau\} \mathcal{L}_{CE}\left[\text{argmax}\{\hat{p}_i\}, h_{\theta,\phi}(\mathcal{A}(x_i))\right] \tag{8}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function which outputs 1 when $c_i > \tau$, with τ being the threshold. $\mathcal{A}(\cdot)$ is a stochastic function that returns a strong augmentation of the input. The final loss is then calculated as a weighted sum between the unsupervised loss, and cross-entropy for weakly augmented labeled samples.

b) Dash: Improves FixMatch by introducing a dynamic threshold ρ_t rather than a fixed one. The rationale is that the fixed threshold may lead to the elimination of too many unlabeled examples with correct pseudo labels [15]. The dynamic threshold is computed as follows:

$$\rho_t = C\gamma^{-(t-1)}\hat{\rho} \tag{9}$$

C and γ being to constants and $\hat{\rho}$ can be approximated as the average loss over the labeled samples.

c) CRMatch: Like Dash, this approach is inspired by FixMatch. Previous Consistency Regularization techniques encourage the model's prediction to be invariant to input perturbation (*e.g.* via MSE). However, combined with the maximization of the distance between features of weakly and strongly augmented samples, the model is actually able to generalize better. Authors conjecture that this encourages the model to have more distinct weakly and strongly augmented features while still imposing the same label, which leads to more expressive representations [6].

The Feature Distance loss is computed as:

$$\mathcal{L}_D(x_i) = d\left[h_{\theta,\phi}(\mathcal{A}(x_i)), h_{\theta,\phi}(\alpha(x_i))\right] \tag{10}$$

where d is a similarity function (*e.g.* cosine similarity). The final unsupervised loss is the same as FixMatch but with soft pseudo labels, instead of hard ones, and with the additional contribution of the Feature Distance Loss:

$$\mathcal{L}_U = \frac{1}{|X_U|}\sum_{i=1}^{|X_U|} \mathbb{1}\{c_i \geq \tau\}\left[\mathcal{L}_{CE}\left[\hat{p}_i, h_{\theta,\phi}(\mathcal{A}(x_i))\right] + \mathcal{L}_D(x_i)\right] \tag{11}$$

d) FlexMatch: Expands on the same trajectory as Dash, by integrating a curriculum learning approach based on K different thresholds, one for each class, where the learning status is taken into account for updating the thresholds [17]. This technique assumes that when a threshold is high, the learning effect of a class is reflected by the number of samples whose prediction falls into this category. With this reasoning, a better learning effect occurs if the unlabeled dataset is balanced. On the other hand, if a class with few samples has prediction confidence reaching the threshold, a more significant learning difficulty is hypothesized. Each of these variables is calculated by scaling the fixed threshold by the learning effect, normalized between 0 and 1.

e) MixMatch: Computes on one side the average prediction for a set of augmentations of unlabeled samples and on the other a forecast for each labeled input after a single modification. Temporal sharpening is then employed to aid the prediction of the pseudo-label for each given input. When all samples are labeled, the two sets are concatenated, shuffled, and augmented with mixup [18], yielding two sets X'_L and X'_U. The final loss is the weighted sum of the cross-entropy, for labeled samples, and the L2 loss on predictions and guessed labels from X'_U [1].

f) ReMixMatch: Adds two contributions to MixMatch: *Distribution Alignment* and *Augmentation Anchoring* [2]. The former encourages the distribution of a model's aggregated class predictions to match the marginal distribution of ground-truth labels. To perform distribution alignment, the model's prediction $h_{\theta,\phi}(x)$ is used, where $x \in X_U$, is scaled by the ratio $p(y)/\tilde{p}(y)$, and where $\tilde{p}(y)$ is a running average of the model's prediction on unlabeled data. The result is then normalized to obtain a valid probability distribution. The latter replaces the consistency regularization component of MixMatch, by using a weakly augmented prediction as a pseudo-label for many strongly augmented versions of the same image. To produce strong augmentations *CTAugment*, a variant of AutoAugment [4], samples random transformations and infers their magnitude during the training, hence it does not need to be optimized on a supervised proxy and has no sensible hyperparameters. In order to better exploit the data, the *rotation loss* [7], is integrated into the method.

IV. Experiments

We decided to perform experiments on selected methods using the USB [14] benchmarking library, in order to reproduce their results and compare the performances of the different algorithms. Due to limited time and computational resources, in order to have at least one representative for each SSL methodology, we opted for benchmarking only: Π-*Model, Pseudo-Labeling, VAT, ReMixMatch, FixMatch,* and *FlexMatch.* All SSL frameworks are finetuned on CIFAR-100 using ViT-S with 32x32 patch size pre-trained on ImageNet [5] as the backbone model. All models are trained on an NVIDIA K80 GPU on the University's Azure VMs, while USB ones are trained on a single NVIDIA V100. For each method, we decided to keep the default hyper-parameters and configurations as in USB [14], since they are the same as in the original papers or slightly optimized.

TABLE I
Accuracies on CIFAR-100 with 200 and 400 labels.

	Method	200 labels	400 labels
USB [14]	Pseudo-Labeling	$66.84_{\pm 1.20}$	$74.71_{\pm 0.67}$
	Π-model	$63.76_{\pm 0.27}$	$73.51_{\pm 0.64}$
	Mean Teacher	$64.39_{\pm 0.38}$	$74.03_{\pm 0.37}$
	VAT	$68.39_{\pm 1.37}$	$78.71_{\pm 0.32}$
	MixMatch	$62.57_{\pm 0.58}$	$73.83_{\pm 0.24}$
	ReMixMatch	$\mathbf{79.15}_{\pm 1.42}$	$\mathbf{83.20}_{\pm 0.59}$
	FixMatch	$69.55_{\pm 0.65}$	$80.52_{\pm 0.93}$
	Dash	$69.81_{\pm 1.34}$	$81.10_{\pm 0.42}$
	CRMatch	$70.57_{\pm 1.11}$	$81.50_{\pm 0.26}$
	FlexMatch	$72.92_{\pm 0.90}$	$82.33_{\pm 0.66}$
Ours	Pseudo-Labeling	66.84	74.05
	Π-Model	64.06	74.33
	VAT	67.95	77.80
	ReMixMatch	72.83	**80.76**
	FixMatch	69.30	77.17
	FlexMatch	**73.32**	76.03
Lower Bound	*Supervised*	$64.37_{\pm 0.07}$	$73.92_{\pm 0.50}$
Upper Bound	*Fully Supervised*	$91.56_{\pm 0.07}$	

Results are shown in Table I. In most cases, USB's performances were reached, while in others, the computational power at our disposal was not enough to do so. In fact, for some of the above methods, each epoch lasted more than 30 minutes. Thus, training had to be stopped before convergence, otherwise, we would not have been able to present viable results. *ReMixMatch* in particular, has been very hard to train, and with more training epochs it would have certainly reached the best performance amongst the experiments.

Furthermore, we observe that, for every method, doubling the number of available labels for supervision leads to a 9% average increase in accuracy. Using 200 or 400 labels on CIFAR-100 corresponds to 0.4% and 0.8% of samples respectively as supervision.

V. Conclusion

In this report, we presented 10 methods that are, in our opinion, some of the most important ones in the context of Semi-supervised learning. Moreover, we described the conducted experiments as well as the results obtained. For the next milestone, we will implement another method into the USB benchmark [14] and also perform experiments on Tiny-Imagenet [10]. We also want to investigate the performances of Semi-supervised image classification when the unlabeled set comes from a different distribution as well as in the presence of noise in the data.

TABLE II
Results in Cross-Domain Evaluation

Method	Accuracy
Π-Model	64.30
FixMatch	78.78
ReMixMatch	87.24
Π-Model+DA	64.23
FixMatch+DA	81.17
Supervised (LB)	78.17
Fully Supervised (UB)	93.01

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: *CoRR* abs/1911.09785 (2019). arXiv: 1911.09785. URL: http://arxiv.org/abs/1911.09785.

[3] Dong-Dong Chen et al. "Tri-net for Semi-Supervised Deep Learning". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, July 2018, pp. 2014–2020. DOI: 10.24963/ijcai.2018/278. URL: https://doi.org/10.24963/ijcai.2018/278.

[4] Ekin Dogus Cubuk et al. "AutoAugment: Learning Augmentation Policies from Data". In: *CoRR* abs/1805.09501 (2018). arXiv: 1805.09501. URL: http://arxiv.org/abs/1805.09501.

[5] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[6] Yue Fan, Anna Kukleva, and Bernt Schiele. "Revisiting Consistency Regularization for Semi-supervised Learning". In: *DAGM German Conference on Pattern Recognition*. Springer. 2021, pp. 63–78.

[7] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations". In: *CoRR* abs/1803.07728 (2018). arXiv: 1803.07728. URL: http://arxiv.org/abs/1803.07728.

[8] Yves Grandvalet and Y. Bengio. "Semi-supervised Learning by Entropy Minimization". In: vol. 17. Jan. 2004.

[9] Samuli Laine and Timo Aila. "Temporal ensembling for semi-supervised learning". In: *arXiv preprint arXiv:1610.02242* (2016).

[10] Ya Le and Xuan Yang. "Tiny imagenet visual recognition challenge". In: *CS 231N* 7.7 (2015), p. 3.

[11] Takeru Miyato et al. "Virtual adversarial training: a regularization method for supervised and semi-supervised learning". In: *IEEE transactions on pattern analysis and machine intelligence* 41.8 (2018), pp. 1979–1993.

[12] Kihyuk Sohn et al. *FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence*. 2020. DOI: 10.48550/ARXIV.2001.07685. URL: https://arxiv.org/abs/2001.07685.

[13] Antti Tarvainen and Harri Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results". In: *Advances in neural information processing systems* 30 (2017).

[14] Yidong Wang et al. *USB: A Unified Semi-supervised Learning Benchmark for Classification*. 2022. DOI: 10.48550/ARXIV.2208.07204. URL: https://arxiv.org/abs/2208.07204.

[15] Yi Xu et al. "Dash: Semi-supervised learning with dynamic thresholding". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 11525–11536.

[16] Xiangli Yang et al. "A survey on deep semi-supervised learning". In: *arXiv preprint arXiv:2103.00550* (2021).

[17] Bowen Zhang et al. *FlexMatch: Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling*. 2021. DOI: 10.48550/ARXIV.2110.08263. URL: https://arxiv.org/abs/2110.08263.

[18] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". In: *CoRR* abs/1710.09412 (2017). arXiv: 1710.09412. URL: http://arxiv.org/abs/1710.09412.