Article 1
FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence

Synopsis:

Motivation

- Semi-supervised learning (SSL) provides an effective means of leveraging unlabeled data to improve a model's performance. This domain has seen fast progress recently, at the cost of requiring more complex methods.
- Using multiple neural networks, different types of losses that are minimized makes these methods very expensive.
- Authors propose FixMatch, an algorithm that is a significant simplification of existing SSL methods.

Approach

- FixMatch, produces artificial labels using both consistency regularization and pseudo-labeling.
- Crucially, the artificial label is produced based on a weakly-augmented unlabeled image (e.g., using only flip-and-shift data augmentation) which is used as a target when the model is fed a strongly augmented version of the same image.
- Following the approach of pseudo-labeling , authors only retain an artificial label if the model assigns a high probability to one of the possible classes.
- A diagram of FixMatch is shown in fig 2.

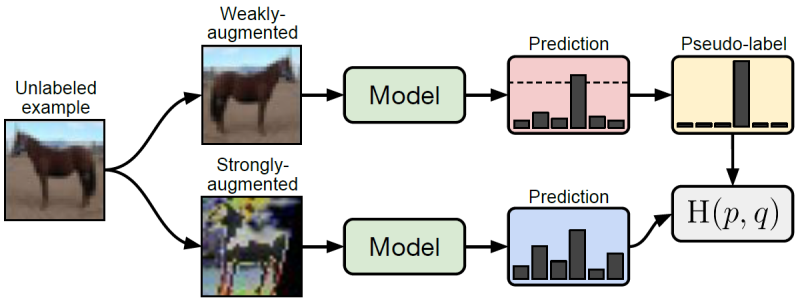

Figure 1: FixMatch

FixMatch

- Consistency Regularization
  - Consistency regularization utilizes unlabeled data by relying on the assumption that the model should output similar predictions when fed perturbed versions of the same image.
  - The model is trained both via a standard supervised classification loss and on unlabeled data via the loss function

$$\sum_{b=1}^{\mu B} \|p_m(y|\alpha(u_b)) - p_m(y|\alpha(u_b))\|_2^2$$

  - Where B is the batch size and μ is the hyperparameter that determines the relative sizes of labelled and unlabelled data. α is a stochastic weak augmentation so the two terms in equation above will indeed have different values.

- Pseudo-labeling
  - It is the idea of using the model itself to obtain artificial labels for unlabeled data.
  - Let $q_b = p_m(y|u_b)$, then pseudo-labeling uses the following loss function:

$$\frac{1}{\mu B}\sum_{b=1}^{\mu B}\mathbb{I}(max(q_b) \geq \tau)H(\hat{q_b}, q_b)$$

  - Where, $\hat{q_b} = argmax(q_b)$ and τ is the threshold.
  - For simplicity, we assume that arg max applied to a probability distribution produces a valid "one-hot" probability distribution. The use of a hard label makes pseudo-labeling closely related to entropy minimization, where the model's predictions are encouraged to be low-entropy (i.e., high-confidence) on unlabeled data.

- Algorithm
  - The loss function of FixMatch consists of two cross-entropy loss terms: a supervised loss l_s applied to labeled data and an unsupervised loss l_u. Specifically, l_s is just the standard cross-entropy loss on weakly augmented labeled examples:

$$l_s = \frac{1}{B}\sum_{b=1}^{B}H(p_b, p_m(y|\alpha(x_b)))$$

  - For the unsupervised part:

$$l_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B}\mathbb{I}(max(q_b) \geq \tau)H(\hat{q_b}, p_m(y|\mathcal{A}(u_b)))$$

  - Where $\mathcal{A}$ is strongly-augmented version of u_b.
  - The loss minimized by FixMatch is simply $l_s + \lambda_u l_u$ where λ_u is a fixed scalar hyperparameter denoting the relative weight of the unlabeled loss.
  - The unsupervised loss forces the model to learn to reverse the strong augmentation. When a weak augmented image is distorted heavily, and fed into the model, the label needs to be the same. Basically, map any strongly augmented image to the same class as a weakly augmented image of the same type.
  - Essentially, the model learns to not distinguish between differently augmented versions of the same image.

Some interesting results

- The model attains 94.93% accuracy on CIFAR-10 with 250 labels and 88.61% accuracy with 40 labels – just 4 labels per class.
- The algorithm also reaches 78% CIFAR-10 accuracy using only above 10 labeled images. However, these 10 images might be the best representative for each of the class.

Article 2
Meta Learning with Implicit Gradients

Synopsis:

Motivation

- In Model Agnostic Meta-Learning(MAML), updating meta parameters requires backpropagation through the inner optimization process. As a result, the meta-learning process requires higher-order derivatives, imposes a non-trivial computational and memory burden, and can suffer from vanishing gradients.
- These limitations make it harder to scale optimization-based meta-learning methods to tasks involving medium or large datasets, or those that require many inner-loop optimization steps

Approach

- The goal of meta-learning is to learn meta-parameters that produce good task-specific parameters after adaptation as specified below -

$$\overbrace{\boldsymbol{\theta}^*_{\mathrm{ML}} := \operatorname*{argmin}_{\boldsymbol{\theta}\in\Theta} F(\boldsymbol{\theta})}^{\text{outer-level}}, \text{ where } F(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}\Big(\overbrace{\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}})}^{\text{inner-level}}, \mathcal{D}_i^{\mathrm{test}}\Big). \tag{1}$$

- In the case of MAML , Alg(θ, D) corresponds to one or multiple steps of gradient descent initialized at .One step of gradient descent update is given by-

$$\boldsymbol{\phi}_i \equiv \mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}}) = \boldsymbol{\theta} - \alpha\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}}). \quad \text{(inner-level of MAML)} \tag{2}$$

- This as a bi-level optimization problem since we typically interpret Alg$(\theta, D^i_{train}), D^i_{test})$ as either explicitly or implicitly solving an underlying optimization problem
- To have sufficient learning in the inner level while also avoiding over-fitting, Alg needs to incorporate some form of regularization.Consider the following explicitly regularized algorithm -

$$\mathcal{A}lg^\star(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}}) = \operatorname*{argmin}_{\boldsymbol{\phi}'\in\Phi} \mathcal{L}(\boldsymbol{\phi}', \mathcal{D}_i^{\mathrm{tr}}) + \frac{\lambda}{2}\,||\boldsymbol{\phi}' - \boldsymbol{\theta}||^2. \tag{3}$$

- The proximal regularization term in Eq. 3 encourages ϕ_i to remain close to θ, thereby retaining a strong dependence throughout. Alg* denotes that the inner optimization is solved exactly
- With the following notation, the bilevel optimization problem is written in Equation 4.

$$\mathcal{L}_i(\boldsymbol{\phi}) := \mathcal{L}\big(\boldsymbol{\phi},\, \mathcal{D}_i^{\mathrm{test}}\big), \quad \hat{\mathcal{L}}_i(\boldsymbol{\phi}) := \mathcal{L}\big(\boldsymbol{\phi}, \mathcal{D}_i^{\mathrm{tr}}\big), \quad \mathcal{A}lg_i\big(\boldsymbol{\theta}\big) := \mathcal{A}lg\big(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}}\big).$$

$$\begin{aligned} &\boldsymbol{\theta}^*_{\mathrm{ML}} := \operatorname*{argmin}_{\boldsymbol{\theta}\in\Theta} F(\boldsymbol{\theta}), \text{ where } F(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}_i\big(\mathcal{A}lg_i^\star(\boldsymbol{\theta})\big), \text{ and} \\ &\mathcal{A}lg_i^\star(\boldsymbol{\theta}) := \operatorname*{argmin}_{\boldsymbol{\phi}'\in\Phi} G_i(\boldsymbol{\phi}', \boldsymbol{\theta}), \text{ where } G_i(\boldsymbol{\phi}', \boldsymbol{\theta}) = \hat{\mathcal{L}}_i(\boldsymbol{\phi}') + \frac{\lambda}{2}\,||\boldsymbol{\phi}' - \boldsymbol{\theta}||^2. \end{aligned} \tag{4}$$

- In order to update meta parameters through SGD we need to find derivative of $\mathcal{L}_\rangle(\phi_i)$ where $\phi_i = \text{Alg}_i(\theta)$.Using chain rule we get -

$$\boldsymbol{d_\theta \mathcal{L}_i(\mathcal{A}lg_i(\theta)) = \frac{d\mathcal{A}lg_i(\theta)}{d\theta} \nabla_\phi \mathcal{L}_i(\phi) \mid_{\phi=\mathcal{A}lg_i(\theta)} = \frac{d\mathcal{A}lg_i(\theta)}{d\theta} \nabla_\phi \mathcal{L}_i(\mathcal{A}lg_i(\theta))}$$

- Implicit MaML Algorithm -

$$\boldsymbol{\theta \leftarrow \theta - \eta \frac{1}{M} \sum_{i=1}^{M} \frac{d\mathcal{A}lg_i^\star(\theta)}{d\theta} \nabla_\phi \mathcal{L}_i(\mathcal{A}lg_i^\star(\theta)).} \tag{5}$$

- In Equation 5 the term $\boldsymbol{\nabla}_\phi \mathcal{L}_i(\text{Alg}_i{}^*(\theta))$ is $\mathcal{L}_i(\phi)|_{\phi=\text{Alg}_i{}^*(\theta)}$ is easily computable(one step of backpropagation) while the term $\frac{d\text{Alg}_i{}^*(\theta)}{d\theta}$ (Meta Gradient) presents the primary challenge.

- The meta gradient computation is given by the following lemma -

  Lemma - (Implicit Jacobian) Consider $\text{Alg}_i{}^*(\theta)$ defined as in Equation 4 for task $\mathcal{T}_i$. Let $\phi_i = \text{Alg}_i{}^*(\theta)$ be the result of $\text{Alg}_i{}^*(\theta)$. If $\mathbf{I} + 1/\lambda * \boldsymbol{\nabla}^2_\phi(\mathcal{L}_i(\phi_i))$ is invertible the closed form solution of Jacobian is -

$$\boldsymbol{\frac{d\mathcal{A}lg_i^\star(\theta)}{d\theta} = \left(I + \frac{1}{\lambda} \nabla^2_\phi \hat{\mathcal{L}}_i(\phi_i) \right)^{-1}.} \tag{6}$$

- It is interesting to note that the derivative (Jacobian) depends only on the final result of the algorithm, and not the path taken by the algorithm.

- While Lemma 1 provides an idealized way to compute the meta-gradient in closed form , it may be difficult to directly use it in practice. Explicitly forming and inverting the matrix in Eq. 6 for computing the Jacobian may be intractable for large deep neural networks.

- The authors use conjugate gradient (CG) algorithm to approximate Hessian-vector products which avoids the calculation of exact gradient.

Results and Conclusion

- The authors found that this approach yielded significant gains in compute and memory efficiency(in comparison to MAML)

- The authors developed method for optimization-based meta-learning that removes the need for differentiating through the inner optimization path, allowing to decouple the outer metagradient computation from the choice of inner optimization algorithm. Therefore this approach allows us to use a variety of inner optimization methods.

- This work also yielded better accuracy scores on Omniglot and MiniImage Net compared to MAML in most of the few shot settings