IA 323: Computer Vision **(04/06/2020)**

Projet #1 : Semi Supervised Learning

Enseignant: Gianni Franchi *Élève:* Julien Guégan

L'apprentissage semi-supervisé (SSL) est une approche qui permet d'entraîner des modèles sur une large quantité de données sans avoir recours une large quantité d'étiquettes. Le SSL permet d'atténuer l'exigence pour les données étiquetées en tirant parti des données non étiquetées. Les méthodes les plus populaires consiste à produire une étiquette artificielle pour chaque image non étiquetée, puis entraîner le modèle à prédire l'étiquette artificielle lorsque l'image non étiquetée est donnée en entrée du modèle.

Pour ce mini-projet, j'ai choisi d'étudier l'algorithme **FixMatch** qui combine certains mécanismes de l'état-de-l'art pour produire des étiquettes artificielles notamment la régularisation consistante et le pseudo-étiquetage. J'ai choisi de partir sur cette méthode car elle est basée d'un papier récent (Janvier 2020) écrit par une équipe de Google et atteint un score de 94.93% de précision sur CIFAR-10 avec 250 exemples étiquetés alors que le précédent état-de-l'art est à 93.73%.

1 Background

L'algorithme FixMatch se base sur 2 approches classiques : la régularisation consistante et le pseudo-étiquetage. On introduit ces 2 techniques dans cette section avec, pour le problème de classification multi-classe, les notations suivantes :

- $\mathcal{X} = (x_b, p_b) : b \in (1, ..., B)$ un batch de B exemples étiquetés (x_b les exemples et p_b les labels one-hot).
- $\mathcal{U} = u_b : b \in (1, ..., \mu B)$ un batch de μB exemples non-étiquetés
- $p_m(y|x)$ la distribution de la classe prédite qui est produit par le modèle
- $H(p, q)$ la cross-entropie entre 2 distributions de probabilités p et q
- $\mathcal{A}(.)$ et $\alpha(.)$ deux types de fonctions d'augmentations.

Le **pseudo-étiquetage** exploite l'idée qu'on devrait utiliser le modèle en lui-même pour obtenir des étiquettes artificielles pour les données qui sont à l'origine non étiquetées. Ce mécanisme fait référence à l'utilisation d'étiquettes *dures*, c'est-à-dire qu'on garde simplement l'arg max de la sortie du modèle, et ensuite on ne conserve que les étiquettes artificielles dont la plus grande probabilité de classe tombe au-dessus d'un seuil prédéfini (ce seuil est un hyper-paramètre). Soit $q_b = p_m(y|u_b)$, le pseudo-étiquetage utilise la Loss suivante pour les données non-étiquetées :

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

Cependant, étant donné qu'il n'y a pas d'étiquettes à l'origine pour ces données, on ne sait pas si on peut faire confiance à cette pseudo-étiquette donnée par le modèle. C'est pourquoi la **régularisation consistante** est utile pour augmenter la précision de ces étiquettes temporaires en en s'appuyant sur l'hypothèse que le modèle devrait produire des prédictions similaires lorsqu'il est alimenté par des versions perturbées d'une même image. En déformant l'image via des méthodes dites d'*augmentation*, on minimise la différence entre 2 versions modifiées de cette image:

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

avec α et p toutes les deux des fonctions stochastiques, et donc les deux termes de l'équation ont bien des valeurs différentes.

2 Augmentation

FixMatch exploite 2 types d'augmentations : "faible" et "forte". L'augmentation "faible" est une stratégie *flip-and-shift*, plus spécifiquement les images sont inversées horizontalement avec une probabilité de 50% et aléatoirement translatées jusqu'à 12.5% verticalement et horizontalement. L'augmentation "forte" est basée sur RandAugment[1] qui consiste à sélectionner aléatoirement des transformations pour chaque exemple d'un mini-batch parmi une collection de transformation et un hyper-paramètre contrôle la magnitude avec laquelle les déformations de RandAugment sont appliquées.

Transformation	Description	Parameter	Range
Autocontrast	Maximizes the image contrast by setting the darkest (lightest) pixel to black (white).		
Brightness	Adjusts the brightness of the image. $B = 0$ returns a black image, $B = 1$ returns the original image.	B	[0.05, 0.95]
Color	Adjusts the color balance of the image like in a TV. $C = 0$ returns a black & white image, $C = 1$ returns the original image.	C	[0.05, 0.95]
Contrast	Controls the contrast of the image. A $C = 0$ returns a gray image, $C = 1$ returns the original image.	C	[0.05, 0.95]
Equalize	Equalizes the image histogram.		
Identity	Returns the original image.		
Posterize	Reduces each pixel to B bits.	B	[4, 8]
Rotate	Rotates the image by θ degrees.	θ	[-30, 30]
Sharpness	Adjusts the sharpness of the image, where $S = 0$ returns a blurred image, and $S = 1$ returns the original image.	S	[0.05, 0.95]
Shear_x	Shears the image along the horizontal axis with rate R.	R	[-0.3, 0.3]
Shear_y	Shears the image along the vertical axis with rate R.	R	[-0.3, 0.3]
Solarize	Inverts all pixels above a threshold value of T.	T	[0, 1]
Translate_x	Translates the image horizontally by ($\lambda \times$image width) pixels.	λ	[-0.3, 0.3]
Translate_y	Translates the image vertically by ($\lambda \times$image height) pixels.	λ	[-0.3, 0.3]

Figure 1: Liste des transformations de RandAugment.

3 FixMatch

La fonction de Loss de FixMatch est la somme de terme de cross-entropie : Plus spécifiquement, l'un est la Loss supervisée qui est une cross-entropie standard sur les exemples étiquetés faiblement augmentés :

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

Et l'autre est la Loss non supervisée, on calcule une étiquette artificielle pour chaque exemple, qui est ensuite utilisée dans une cross-entropie standard. Pour obtenir une étiquette artificielle, on calcule d'abord la distribution de classe prédite du modèle en fonction d'une version faiblement augmentée d'une image donnée sans étiquette: $q_b = p_m(y|\alpha(u_b))$. Ensuite, on utilise $\hat{q}_b = \arg\max(q_b)$ comme pseudo-étiquette, sauf qu'on applique la cross-entropie à la sortie du modèle pour une version fortement augmentée de u_b:

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

où τ est un hyper-paramètre qui définit le seuil au-dessus du quel on retient un pseudo-label. La différence cruciale avec le pseudo-étiquetage classique est que le label artificiel est calculé sur la base d'une image faiblement-augmentée et la loss est appliquée à la sortie du modèle pour une image fortement augmentée.

Le schéma ci-dessous résume le fonctionnement du modèle. D'abord, une version faiblement augmentée d'une image sans étiquette (en haut) est introduite dans le modèle pour obtenir ses prédictions (boîte rouge). Lorsque le modèle attribue une probabilité à n'importe quelle classe qui est supérieure à un seuil (ligne pointillée), la prédiction est convertie en une pseudo-étiquette one-hot. Ensuite, on calcule la prédiction du modèle pour une version fortement augmentée de la même image (en bas). Le modèle est entraîné pour que sa prédiction sur la version fortement augmentée corresponde au pseudo-label via une loss cross-entropie standard.

Au final la Loss minimisée par l'algorithme FixMatch est simplement $l_s + \lambda_u l_u$ avec λ_u un hyper-paramètre définissant le poids relatif entre les 2 types de loss.

[1] En réalité, FixMatch implémente aussi CTAugment mais j'ai choisi d'utiliser seulement RandAugment dans ce travail

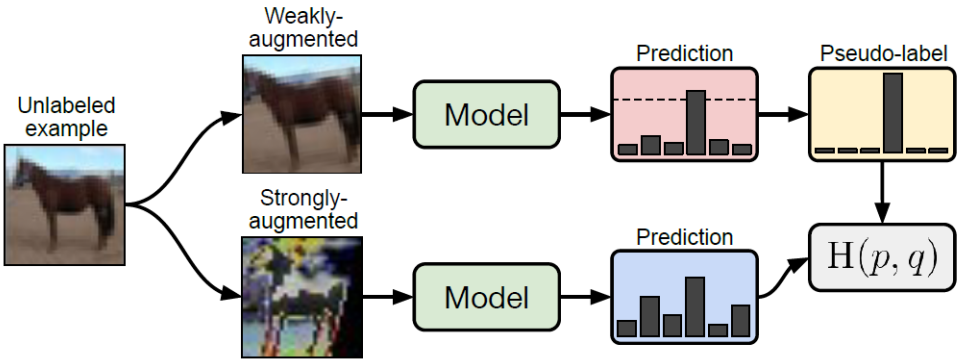

Figure 2: Schéma de l'algorithme FixMatch.

Algorithm 1 FixMatch algorithm.

Input: Labeled batch $\mathcal{X} = \{(x_b, p_b) : b \in (1, \dots, B)\}$, unlabeled batch $\mathcal{U} = \{u_b : b \in (1, \dots, \mu B)\}$, confidence threshold τ, unlabeled data ratio μ, unlabeled loss weight λ_u.
$\ell_s = \frac{1}{B} \sum_{b=1}^{B} \mathrm{H}(p_b, \alpha(x_b))$ *// Cross-entropy loss for labeled data*
for $b = 1$ **to** μB **do**
  $\tilde{u}_b = \mathcal{A}(u_b)$ *// Apply strong data augmentation to* u_b
  $q_b = p_{\mathrm{m}}(y \mid \alpha(u_b); \theta)$ *// Compute prediction after applying weak data augmentation of* u_b
end for
$\ell_u = \frac{1}{\mu B} \sum_{b=1}^{\mu B} \mathbb{1}\{\max(q_b) > \tau\} \mathrm{H}(\arg\max(q_b), \tilde{u}_b)$ *// Cross-entropy loss with pseudo-label and confidence for unlabeled data*
return $\ell_s + \lambda_u \ell_u$

Figure 3: Algorithme FixMatch.

4 Détails d'implémentation

Pour l'apprentissage, j'ai suivi les recommandations de l'article FixMatch et j'ai utilisé pour le réseau de neurones l'architecture WideResNet de profondeur 28 et de facteur d'élargissement de 2 et pas de dropout. Pour le pas d'apprentissage, un planificateur de type cosinus a été mis en place. La technique de la Moyenne mobile exponentielle avec un coefficient de decay de 0.999 est utilisé. Le seuil pour les pseudo-étiquettes est fixé à 0.95. Les autres hyper-paramètres sont détaillés dans le code.

J'ai fait tourné le code sur google Colab en utilisant la sauvegarde et le chargement de checkpoint pour pouvoir reprendre l'entraînement lorsqu'un problème interrompait l'entraînement (redémarrage automatique du noyau, perte de connexion, plus d'espace mémoire). Mais je n'ai pu lancé que 90 epoch et le meilleur modèle obtenu atteint un score de test de 89%.

Le code que j'ai utilisé provient de <https://github.com/kekmodel/FixMatch-pytorch>.