MO433A — Unsupervized Machine Learning
INSTITUTO DE COMPUTAÇÃO — UNICAMP

Semi-Supervised Learning

Lesson No.11 L8b

Ivan Lima

1 Semi-Supervised Learning

In supervised learning we have samples x with labels y.

$$(x, y) \approx p(x, y) \tag{1}$$

The pair (x, y) can be approximate by sampling in the distribution $p(x, y)$.

$$\max \mathbb{E}_{x,y\sim p(x,y)}[\log p(y \mid x)] \tag{2}$$

In semi-supervised learning we have an extra information. We do have the marginal distribution $p(x)$ of the jointly distribution $p(x, y)$. We have the setup shown in equation 3, where D_U is unlabeled data set and D_S is labeled dataset. Thus, the goal is to take advantage of the extra information (unlabeled marginal distribution) to find the jointly distribution which enable us to make predictions.

$$\begin{aligned} D_U &: x \sim p(x), \\ D_S &: (x, y) \sim p(x, y) \end{aligned} \tag{3}$$

Of course, as many labeled sample you have as better the prediction will be. However, labeling samples is expensive. A ideal scenario for semi-supervised learning is when you have a few labeled samples and much larger amount of unlabeled samples. When large amount of data is available, supervised learning is still more efficient than semi-supervised.

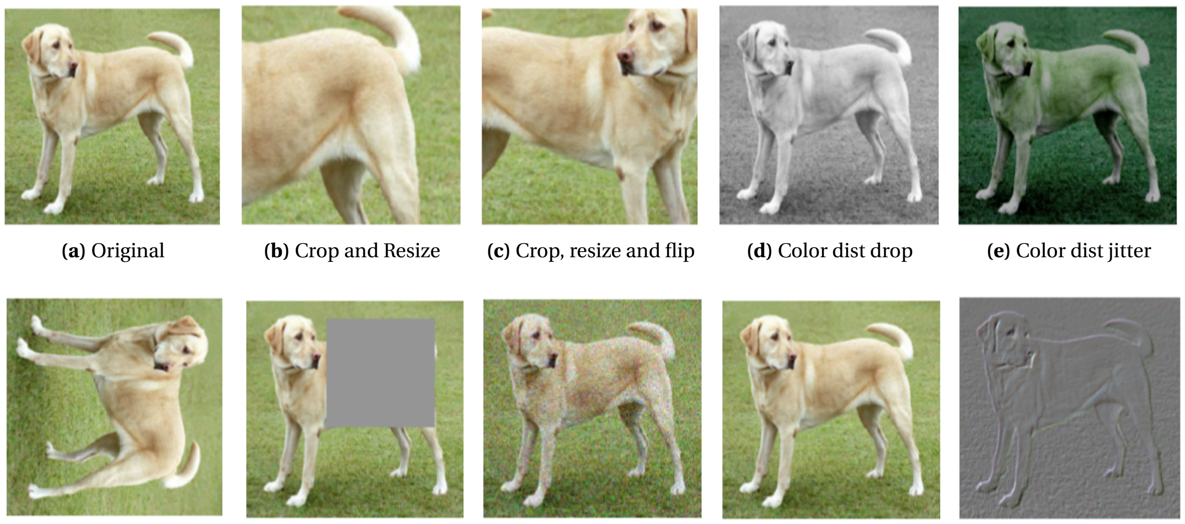
2 Core Concepts of Semi-Supervised Learning

The core concepts are listed below:

- Confidence vs Entropy
- Label Consistency
- Regularization

2.1 Confidence vs Entropy

As there is a trade off between the confidence of the classifiers versus minimum entropy in unlabeled data, the idea is to be sure the the classifiers trained on labeled data has minimum entropy on unlabeled data. Ensure that the classifier is confident even in the unlabeled data. It is a easy way to regularize the classifier. Another idea is the pseudo labeling, which consist of taking high confident prediction made by the classifier and convert them in a extra labeled data. It is also known as self-training. We need to be very carefully once we consider the model to be accurate enough to generate high confident predictions that can be re-utilized for training. we can also add a noise to the model to regularize it via adversarial training, where we train the classifier to be robust to perturbation.



(a) Original **(b)** Crop and Resize **(c)** Crop, resize and flip **(d)** Color dist drop **(e)** Color dist jitter

(f) Rotate **(g)** Cutout **(h)** Gaussian noise **(i)** i **(j)** Sobel Filtering

Figure 1. Different image augmentation. (Original image cc-by: Von.grzanka)

2.2 Label Consistency

The idea is to make sure that the augmentations of the sample have the same class. When you augment unlabeled data, we expected that two augmentation of this data should be roughly similar. You do not know the label, but you do know that they should be similar. And that is how we get a lot of extra information from unlabeled data that helps to increase the performance. It is basically consistency constrain. Pi-Model, Mean Teacher are examples of this technique. See figure 1 for different possible augmentations.

2.3 Regularization

Regularization consists of making sure that the model generalizes well to a new unlabelled data or validation set. Weight decay, dropout, and data-augmentation (MixUp, CutOut) are the current most popular technique in this category.

3 Algorithms

Next are the current most popular algorithms that use the techniques described above.

3.1 Pi model

Pi model relays on the idea of labeling consistency. As shown in the figure 2, you create two different views of your image using a stochastic data augmenter. It could be a random crop, a sequence of data augmentations, etc (as shown in figure 1). You pass the image to the model. The model itself could be stochastic. You could also have dropout. Every forward pass could give you different output even for the same image. You get two different latent variable. Every time you make a forward pass you can enforce a regular supervised cross-entropy loss. If you throw unlabeled data, you can enforce the square difference between the outputs of the model for two different views. For labeled data you can enforce both of these losses. For unlabeled data you only enforce the labeling consistency loss. One will be the semi-supervised loss and the other the supervised loss. You can control which loss dominates the training. See figure 2.

The pseudo code is shown in figure 3. You perform two different augmentations, get two different outputs that you make sure they are close to each other.

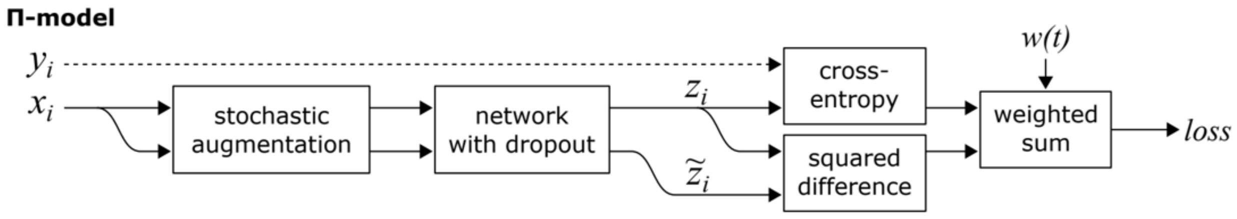
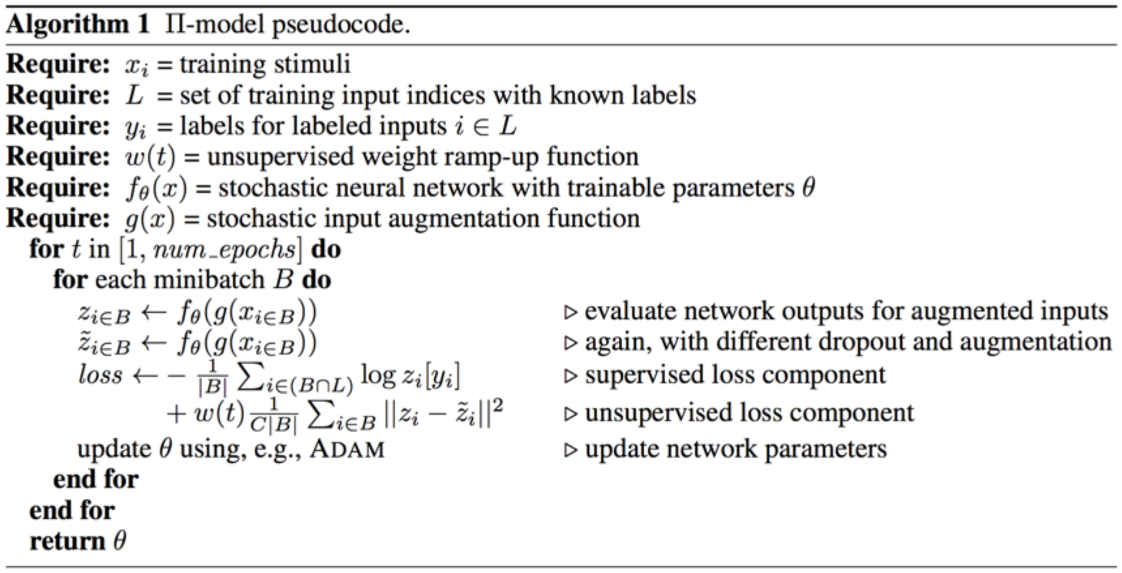


Figure 2. Pi model



Algorithm 1 Π-model pseudocode.

```
Require: $x_i$ = training stimuli
Require: $L$ = set of training input indices with known labels
Require: $y_i$ = labels for labeled inputs $i \in L$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters $\theta$
Require: $g(x)$ = stochastic input augmentation function
  for $t$ in $[1, num\_epochs]$ do
    for each minibatch $B$ do
      $z_{i\in B} \leftarrow f_\theta(g(x_{i\in B}))$                          ▷ evaluate network outputs for augmented inputs
      $\tilde{z}_{i\in B} \leftarrow f_\theta(g(x_{i\in B}))$                  ▷ again, with different dropout and augmentation
      $loss \leftarrow -\frac{1}{|B|}\sum_{i\in(B\cap L)} \log z_i[y_i]$         ▷ supervised loss component
          $+ w(t)\frac{1}{C|B|}\sum_{i\in B} ||z_i - \tilde{z}_i||^2$            ▷ unsupervised loss component
      update $\theta$ using, e.g., ADAM                                        ▷ update network parameters
    end for
  end for
  return $\theta$
```

Figure 3. Pi model pseudo code

3.2 Temporal Ensembling

Temporal augmentation does something slightly different. Instead of relaying on one model and two different hyperparameters ,we relay on the past versions of it. You take the previous Zi that you have got from that particular Xi using the past version of your model.

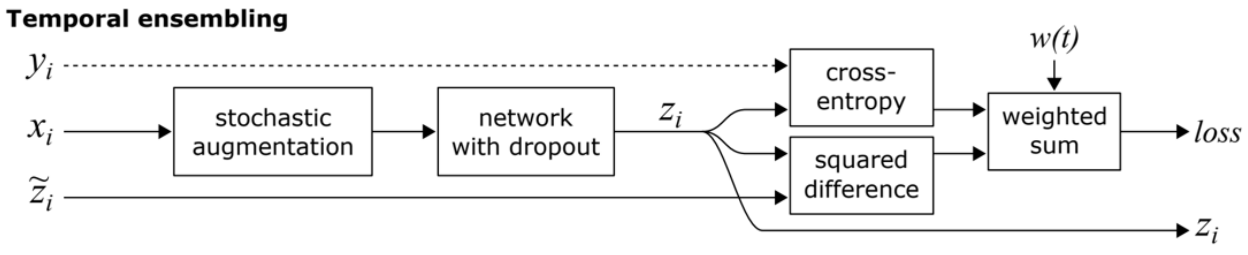
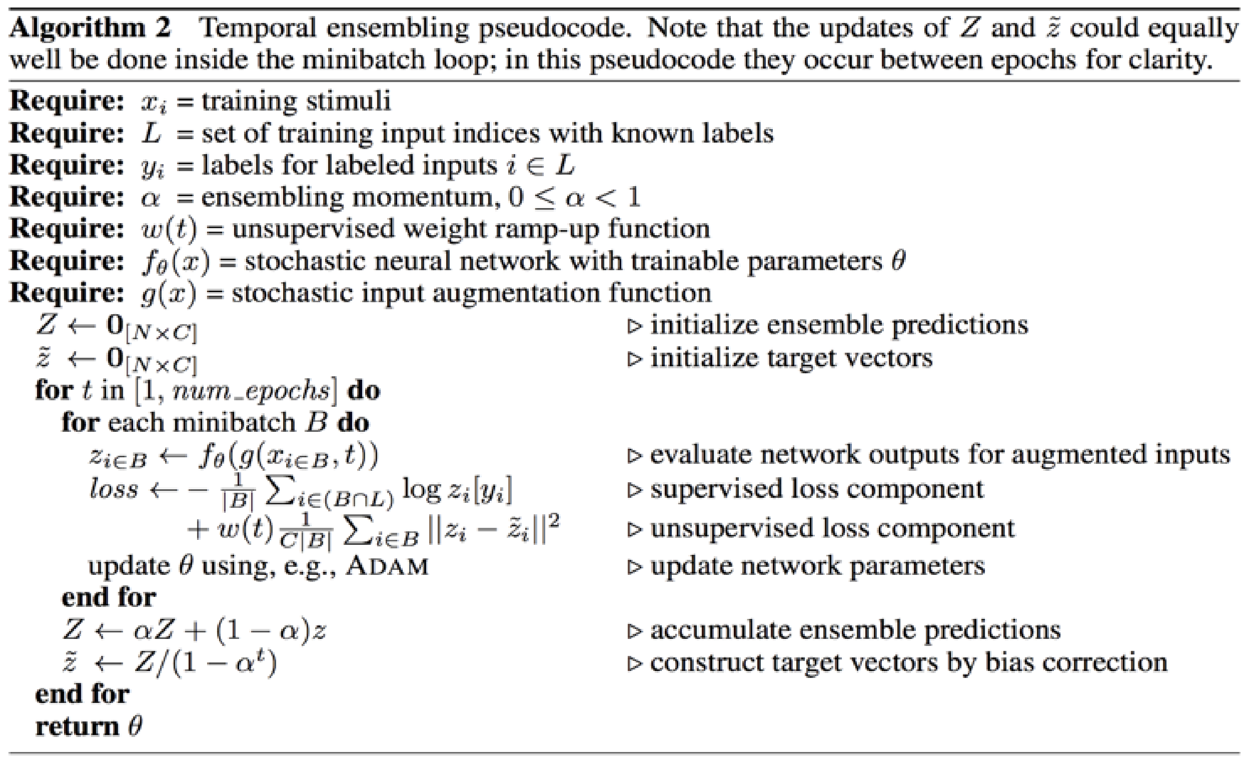


Figure 4. Temporal Ensembling

Moreover, instead o of picking one particular past version, you use an exponential moving average of your outputs. For every Xi you track the moving average of what Zi you have got with it and regularize Xi and Zi to be close to that in addition to the classification loss. The implementation follows the pseudo code 5.



Algorithm 2 Temporal ensembling pseudocode. Note that the updates of Z and $\tilde{z}$ could equally well be done inside the minibatch loop; in this pseudocode they occur between epochs for clarity.

```
Require: $x_i$ = training stimuli
Require: $L$ = set of training input indices with known labels
Require: $y_i$ = labels for labeled inputs $i \in L$
Require: $\alpha$ = ensembling momentum, $0 \leq \alpha < 1$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters $\theta$
Require: $g(x)$ = stochastic input augmentation function
  $Z \leftarrow \mathbf{0}_{[N \times C]}$                          ▷ initialize ensemble predictions
  $\tilde{z} \leftarrow \mathbf{0}_{[N \times C]}$                          ▷ initialize target vectors
  for $t$ in $[1, num\_epochs]$ do
    for each minibatch $B$ do
      $z_{i \in B} \leftarrow f_\theta(g(x_{i \in B}, t))$          ▷ evaluate network outputs for augmented inputs
      $loss \leftarrow -\frac{1}{|B|} \sum_{i \in (B \cap L)} \log z_i[y_i]$     ▷ supervised loss component
             $+ w(t) \frac{1}{C|B|} \sum_{i \in B} ||z_i - \tilde{z}_i||^2$     ▷ unsupervised loss component
      update $\theta$ using, e.g., ADAM                ▷ update network parameters
    end for
    $Z \leftarrow \alpha Z + (1 - \alpha) z$                  ▷ accumulate ensemble predictions
    $\tilde{z} \leftarrow Z / (1 - \alpha^t)$                    ▷ construct target vectors by bias correction
  end for
  return $\theta$
```

Figure 5. Temporal Ensembling pseudo code

3.3 Mean Teacher

The Mean Teacher is very similar to Temporal Ensembling, but instead of looking at the output and keeping a moving average of the outputs, it just count the moving average of the parameters. Therefore you have two different models: the teacher model and the student model. While students model runs in the data points, teacher model takes the exponential moving average of the parameters of the student model. There is a consistency cost between the prediction of the student and the teacher. At the end you discard the student model and apply the teacher model. See figure 6

3.4 Virtual Adversarial Training

In adversarial perturbation you try to take the gradient of the probability of the output with respect to the input. The noise you add to the input is negative times the sign of the gradient that you get from that. If you add this kind of noise, you can perturb the output so that for very similar input you can get a very different output. This is the foundation of every adversarial method. In the virtual adversarial training, we apply the perturbation to the KL between the prediction with the original sample and the prediction with the noise sample. So, we try to find a noise such as the KL is maximized.

$$\boldsymbol{r}_{\text{adv}} = -\epsilon \boldsymbol{g} / \|\boldsymbol{g}\|_2 \text{ where } \boldsymbol{g} = \nabla_{\boldsymbol{x}} \log p(y \mid \boldsymbol{x}; \hat{\boldsymbol{\theta}}) \tag{4}$$

$$r_{v-adv} = arg \max_{r, \|r\| \leq \varepsilon} KL[p(\cdot \mid x; \hat{\theta}) || p(\cdot \mid x + r; \hat{\theta}) \tag{5}$$

Once you solve for the r (noise) by linearizing the KL term, you apply it to the unlabeled data point, and it becomes your consistency cost. Figure 7 shows the pseudo code.

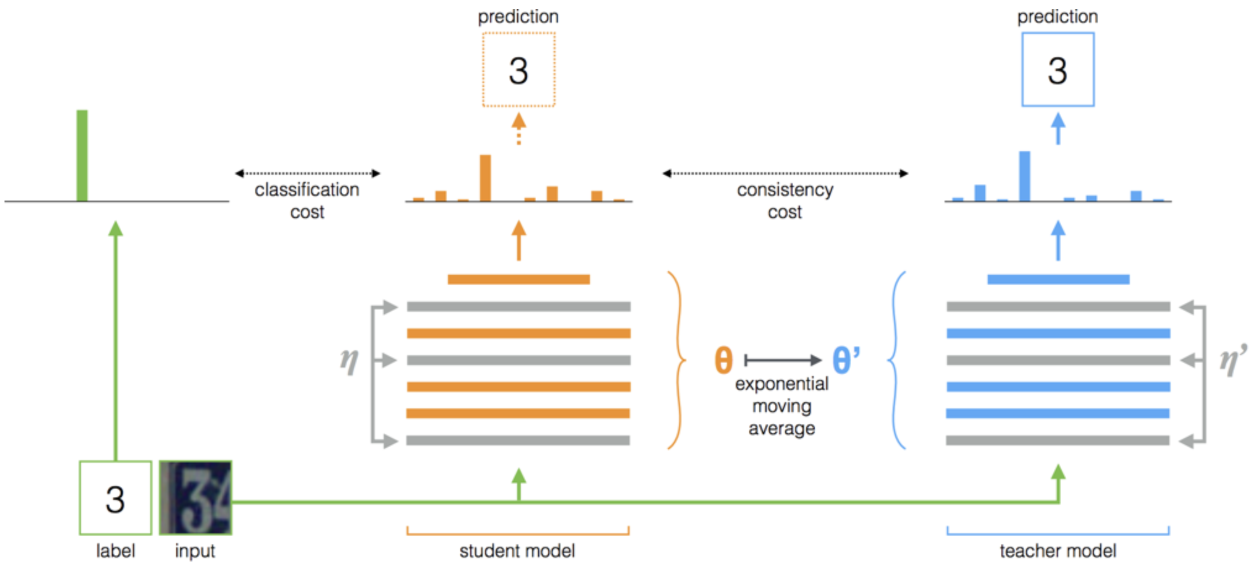


Figure 6. Mean Teacher

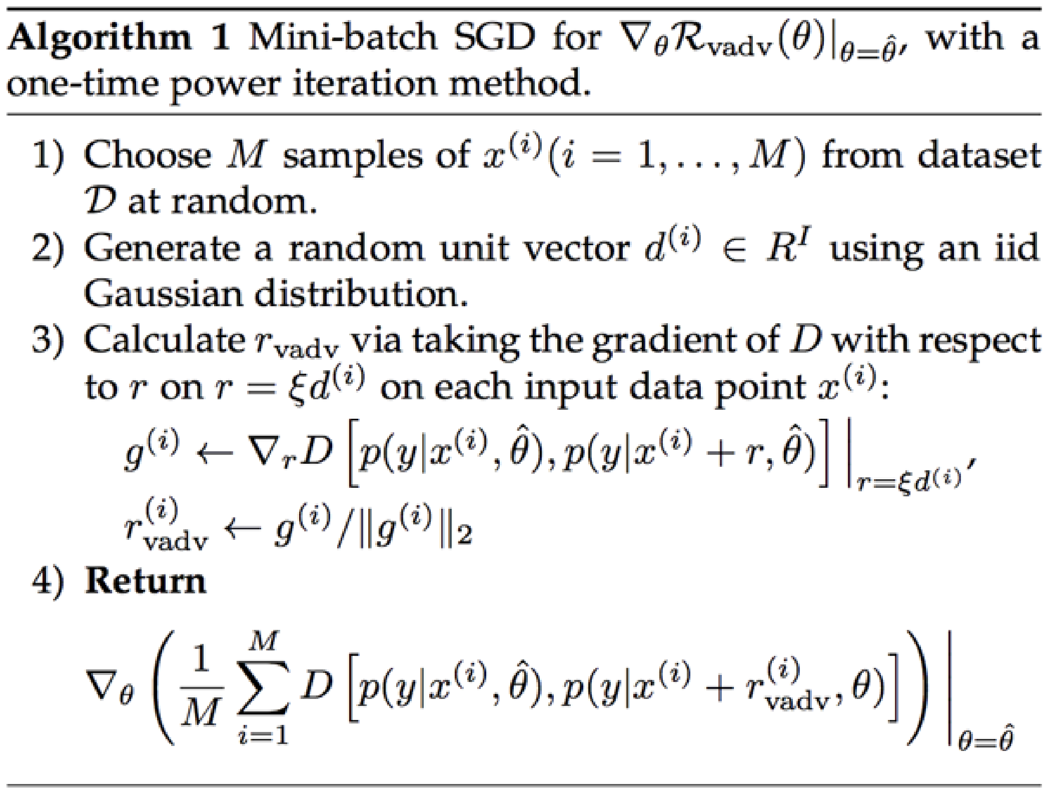
4 Lessons

The following remarks were extracted from the [1] investigation, which compares the performance of semi-supervised learning techniques.

- Given equal budget for tuning hyperparameters, the gap in performance between semi-supervised learning and using only labeled data is smaller the normally reported.
- A large classifier with carefully chosen regularization trained on a small labeled dataset with no unlabeled data can reach very good accuracy.
- In some settings, pre-training a classifier on a different labeled dataset and then retraining on only labeled data from the dataset of interest can outperform all semi-supervised algorithms.
- The performance of semi-supervised techniques can degrade drastically when the unlabeled data contains a different distribution of classes than the labeled data. Unlabeled data from a different class distribution is not that useful.
- Different approaches exhibit substantially different levels of sensitivity to the amount of labeled and unlabeled data.
- Realistically small validation sets would prevent reliable comparison of different methods, models, and hyperparameter settings.
- Most methods do not work well in the very low labeled-data regime

References

[1] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," *CoRR*, vol. abs/1804.09170, 2018. arXiv: 1804.09170. [Online]. Available: http://arxiv.org/abs/1804.09170.



Algorithm 1 Mini-batch SGD for $\nabla_\theta \mathcal{R}_{\text{vadv}}(\theta)|_{\theta=\hat{\theta}}$, with a one-time power iteration method.

```
1) Choose $M$ samples of $x^{(i)} (i = 1, \ldots, M)$ from dataset $\mathcal{D}$ at random.
2) Generate a random unit vector $d^{(i)} \in R^I$ using an iid Gaussian distribution.
3) Calculate $r_{\text{vadv}}$ via taking the gradient of $D$ with respect to $r$ on $r = \xi d^{(i)}$ on each input data point $x^{(i)}$:
    $g^{(i)} \leftarrow \nabla_r D\left[p(y|x^{(i)}, \hat{\theta}), p(y|x^{(i)} + r, \hat{\theta})\right]\Big|_{r=\xi d^{(i)}}$,
    $r_{\text{vadv}}^{(i)} \leftarrow g^{(i)} / \|g^{(i)}\|_2$
4) **Return**
    $\nabla_\theta \left( \frac{1}{M} \sum_{i=1}^{M} D\left[p(y|x^{(i)}, \hat{\theta}), p(y|x^{(i)} + r_{\text{vadv}}^{(i)}, \theta)\right] \right)\Big|_{\theta=\hat{\theta}}$
```

Figure 7. Virtual Adversarial Training pseudo code