Semi-Supervised Learning

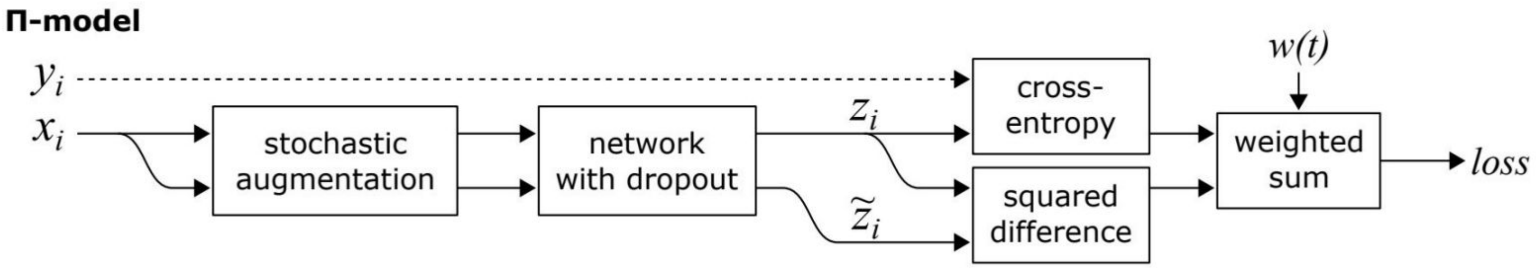
Eugene Lim

$$(x, y) \sim p(x, y)$$

$$(x, y) \sim p(x, y)$$

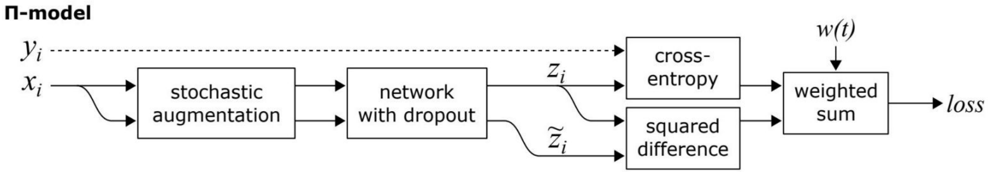
$$x \sim p(x)$$

Pi Model



Π-model
y_i
x_i
stochastic augmentation
network with dropout
z_i
$\tilde{z}_i$
cross-entropy
squared difference
$w(t)$
weighted sum
$loss$

Pi Model



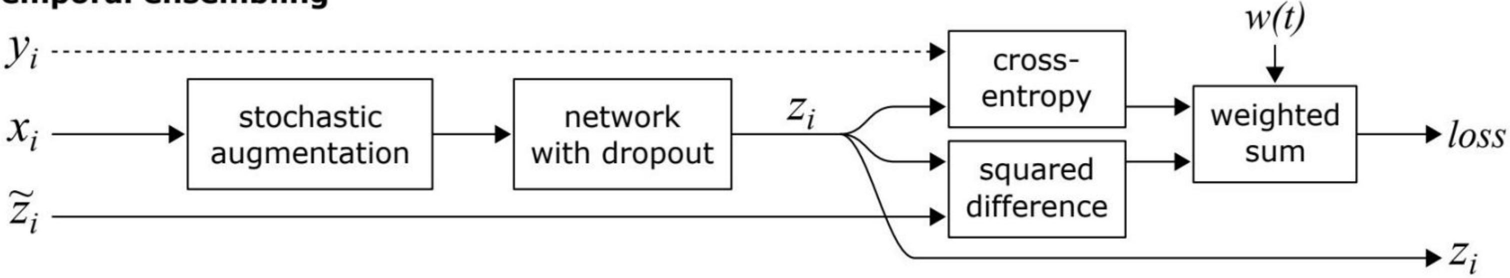
Algorithm 1 Π-model pseudocode.

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 θ
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 θ using, e.g., ADAM ▷ update network parameters
    end for
  end for
  return θ

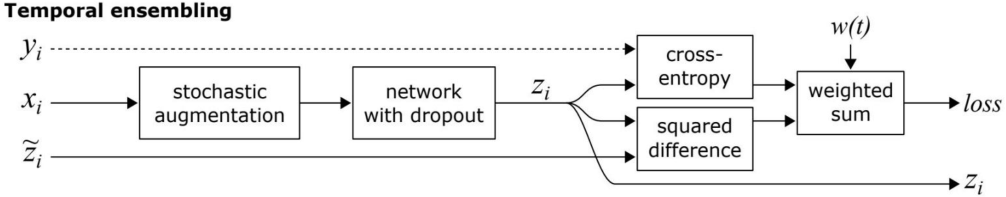
Pi Model

Temporal Ensembling

Temporal ensembling



Temporal ensembling



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.

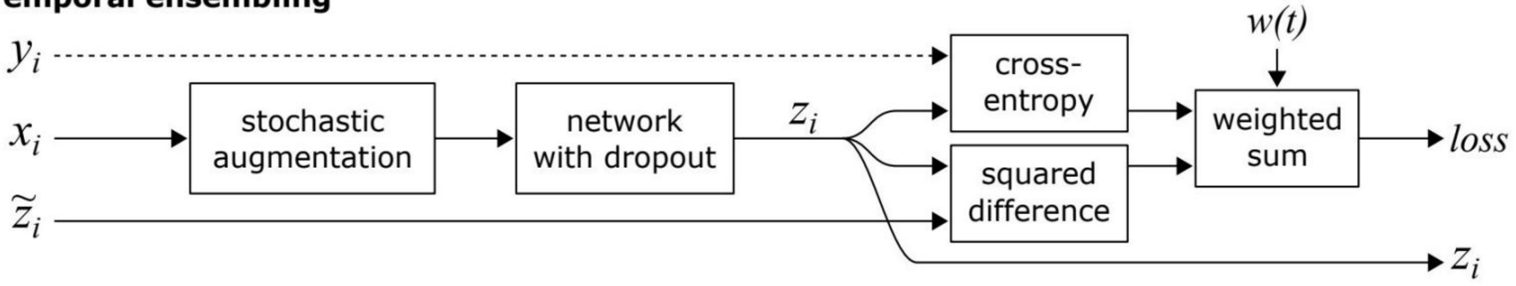
Require: x_i = training stimuli
Require: L = set of training input indices with known labels
Require: y_i = labels for labeled inputs $i \in L$
Require: α = ensembling momentum, $0 \leq \alpha < 1$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = stochastic neural network with trainable parameters θ
Require: $g(x)$ = stochastic input augmentation function

```
Z ← 0_[N×C]                                              ▷ initialize ensemble predictions
z̃ ← 0_[N×C]                                              ▷ initialize target vectors
for t in [1, num_epochs] do
    for each minibatch B do
        z_{i∈B} ← f_θ(g(x_{i∈B}, t))                     ▷ evaluate network outputs for augmented inputs
        loss ← − 1/|B| Σ_{i∈(B∩L)} log z_i[y_i]           ▷ supervised loss component
               + w(t) 1/(C|B|) Σ_{i∈B} ||z_i − z̃_i||^2   ▷ unsupervised loss component
        update θ using, e.g., ADAM                       ▷ update network parameters
    end for
    Z ← αZ + (1 − α)z                                    ▷ accumulate ensemble predictions
    z̃ ← Z/(1 − α^t)                                      ▷ construct target vectors by bias correction
end for
return θ
```

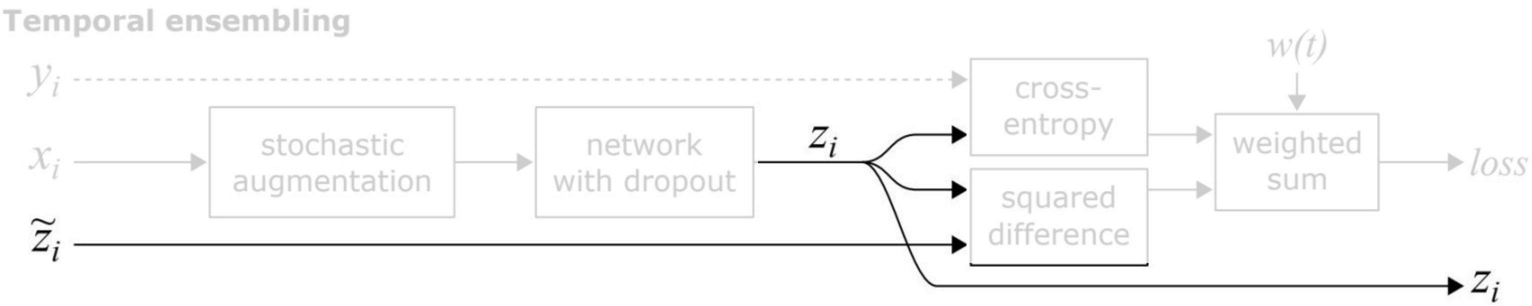
Temporal Ensembling

How to do better?

Temporal ensembling

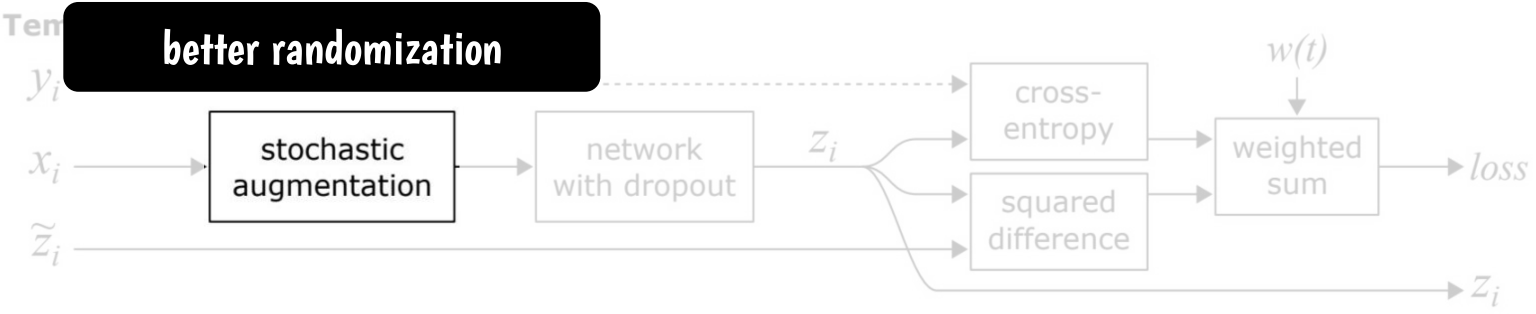


How to do better?



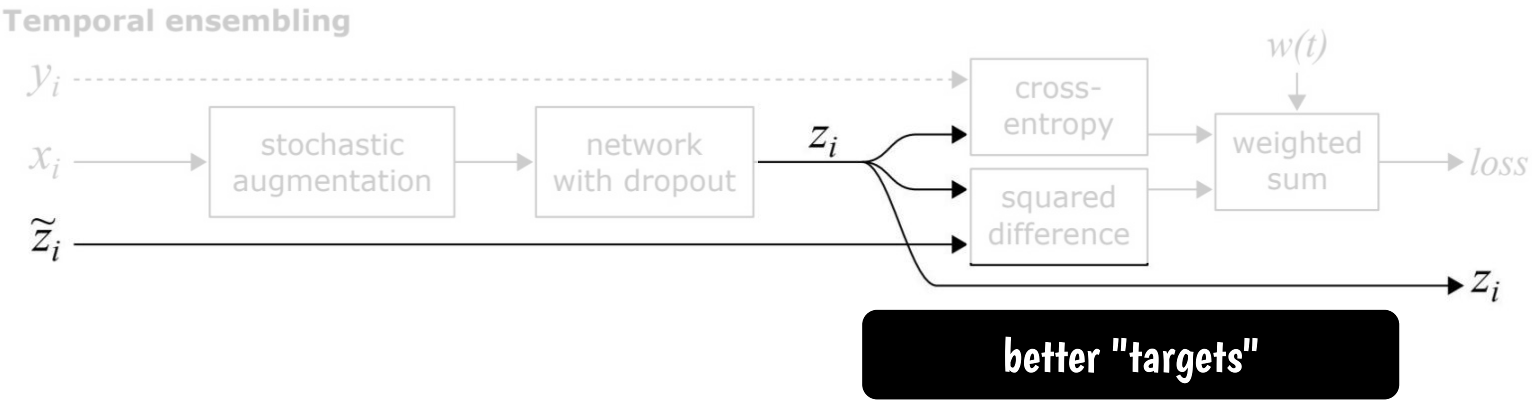
better "targets"

How to do better?

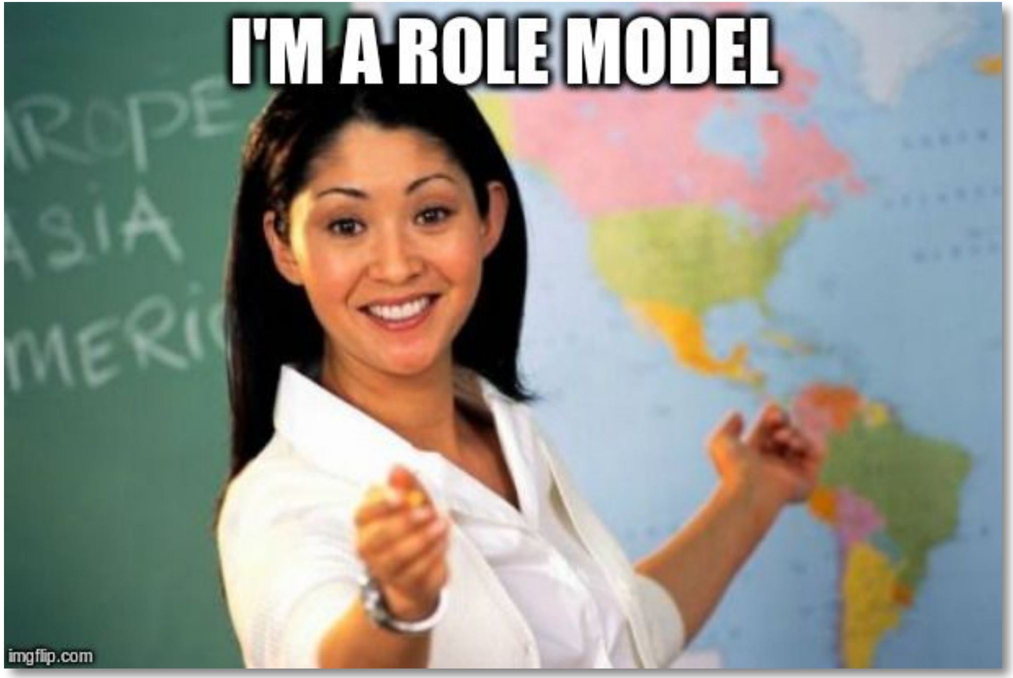


How to do better?

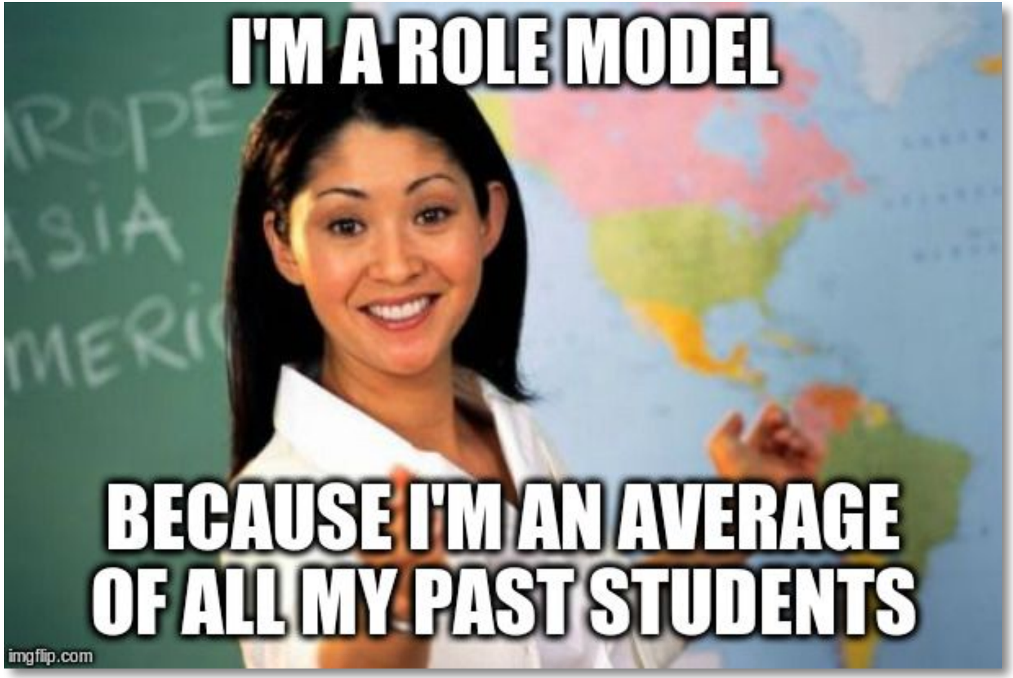
Mean Teacher



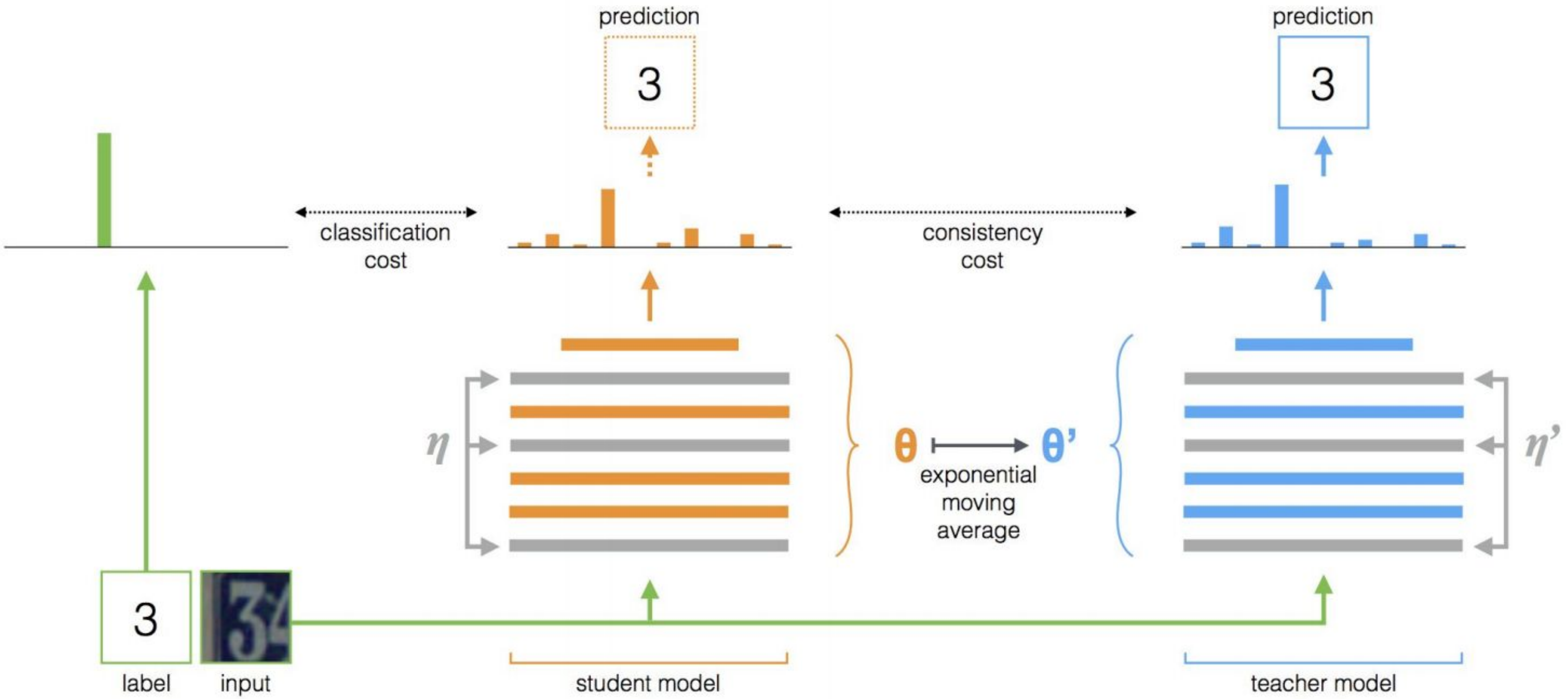
Mean Teacher



Mean Teacher



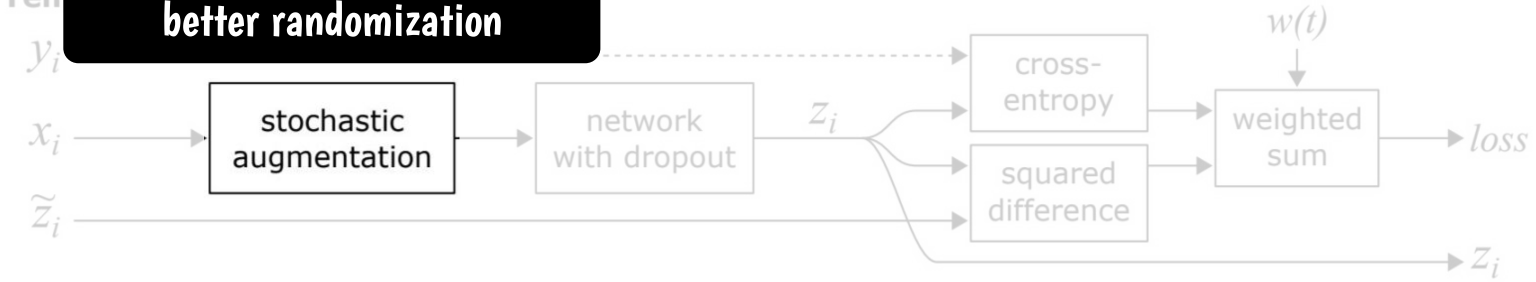
Mean Teacher



Mean Teacher

Virtual Adversarial Training

better randomization



$$L_{\mathrm{adv}}(x_l, \theta) := D\left[q(y|x_l), p(y|x_l + r_{\mathrm{adv}}, \theta)\right]$$

$$\text{where } r_{\mathrm{adv}} := \underset{r;\|r\|\leq\epsilon}{\arg\max}\, D\left[q(y|x_l), p(y|x_l + r, \theta)\right],$$

😢 no closed form 😢

$$L_{\mathrm{adv}}(x_l, \theta) \; \ldots \; x_l + r_{\mathrm{adv}}, \theta)]$$

$$\text{where } r_{\mathrm{adv}} := \underset{r;\|r\|\leq\epsilon}{\arg\max} \, D\left[q(y|x_l), p(y|x_l + r, \theta)\right],$$

$$L_{\mathrm{adv}}(x_l, \theta) := D\left[q(y|x_l), p(y|x_l + r_{\mathrm{adv}}, \theta)\right]$$

$$\text{where } r_{\mathrm{adv}} := \underset{r;\|r\|\leq\epsilon}{\arg\max}\, D\left[q(y|x_l), p(y|x_l + r, \theta)\right],$$

$$r_{\mathrm{adv}} \approx \epsilon \frac{g}{\|g\|_2}, \text{ where } g = \nabla_{x_l} D\left[h(y; y_l), p(y|x_l, \theta)\right]$$

Adversarial training is a successful method that works for many supervised problems. However, full label information is not available at all times. Let x_* represent either x_l or x_{ul}. Our objective function is now given by

$$D\left[q(y|x_*), p(y|x_* + r_{\mathrm{qadv}}, \theta)\right]$$

$$\text{where } r_{\mathrm{qadv}} := \underset{r;\|r\|\leq\epsilon}{\arg\max}\, D\left[q(y|x_*), p(y|x_* + r, \theta)\right],$$

Adversarial training is a successful method that works for many supervised problems. However, full label information is not available at all times. Let x_* represent either x_l or x_{ul}. C

😢 we don't have this 😢

n is now given by

$$D\left[q(y|x_*), p(y|x_* + r_{\text{qadv}}, \theta)\right]$$

$$\text{where } r_{\text{qadv}} := \underset{r;\|r\|\leq\epsilon}{\arg\max}\, D\left[q(y|x_*), p(y|x_* + r, \theta)\right],$$

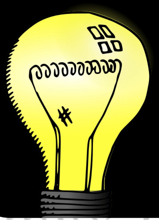
Adversarial training is a successful method that works for many supervised problems. However, full label information is not available at all times. Let x_* represent either x_l or x_{ul}.

C … n

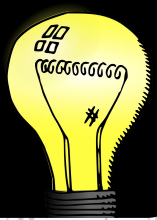
we don't have this

$$D\left[q(y|x_*), p(y|x_* \ldots\right.$$

replace with



$$p(y|x_*, \hat{\theta})$$



where $r_{\text{qadv}} := \arg\max_{r;||r||\le\epsilon} D\left[q(y|x_*), p(y|x_* + r, \theta)\right]$,

$$\mathrm{LDS}(x_*, \theta) := D\left[p(y|x_*, \hat{\theta}), p(y|x_* + r_{\mathrm{vadv}}, \theta)\right]$$

$$r_{\mathrm{vadv}} := \underset{r; \|r\|_2 \leq \epsilon}{\arg\max}\, D\left[p(y|x_*, \hat{\theta}), p(y|x_* + r)\right],$$

$$\mathcal{R}_{\mathrm{vadv}}(\mathcal{D}_l, \mathcal{D}_{ul}, \theta) := \frac{1}{N_l + N_{ul}} \sum_{x_* \in \mathcal{D}_l, \mathcal{D}_{ul}} \mathrm{LDS}(x_*, \theta)$$

$$\ell(\mathcal{D}_l, \theta) + \alpha \mathcal{R}_{\mathrm{vadv}}(\mathcal{D}_l, \mathcal{D}_{ul}, \theta)$$