Learning to Reweight Examples for Robust Deep Learning

Authors:
Mengye Ren, Wenyuan Zeng,
Bin Yang, Raquel Urtasun

Presented by:
Janaki Viswanathan,
ML for NLP [WS 2019/2020]
Saarland University

Overview

- Motivation
- Related work
    - Data resampling
    - Training loss based
    - Shortcomings
- Learning to reweight examples
    - Intuition
    - Meta-learning perspective
    - Online approximation
    - The algorithm
- Experiment results

Overview

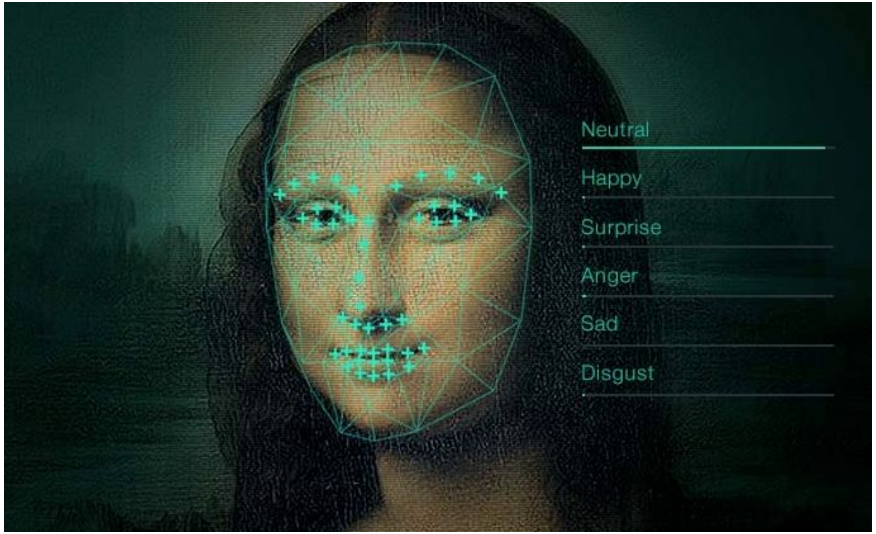
- Motivation
- Related work
    - Data resampling
    - Training loss based
    - Shortcomings
- Learning to reweight examples
    - Intuition
    - Meta-learning perspective
    - Online approximation
    - The algorithm
- Experiment results

Motivation

- Deep neural networks (DNNs) have powerful capacity for modeling complex input patterns

Motivation

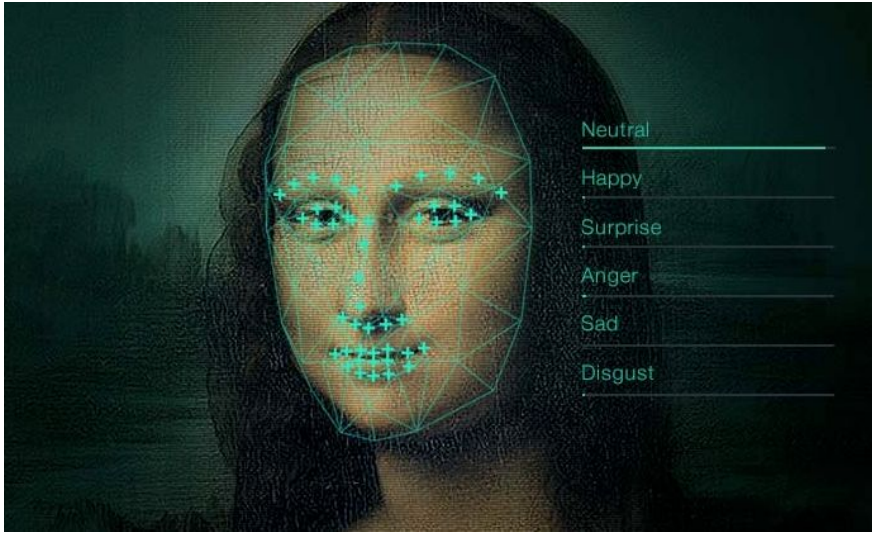
- Deep neural networks (DNNs) have powerful capacity for modeling complex input patterns

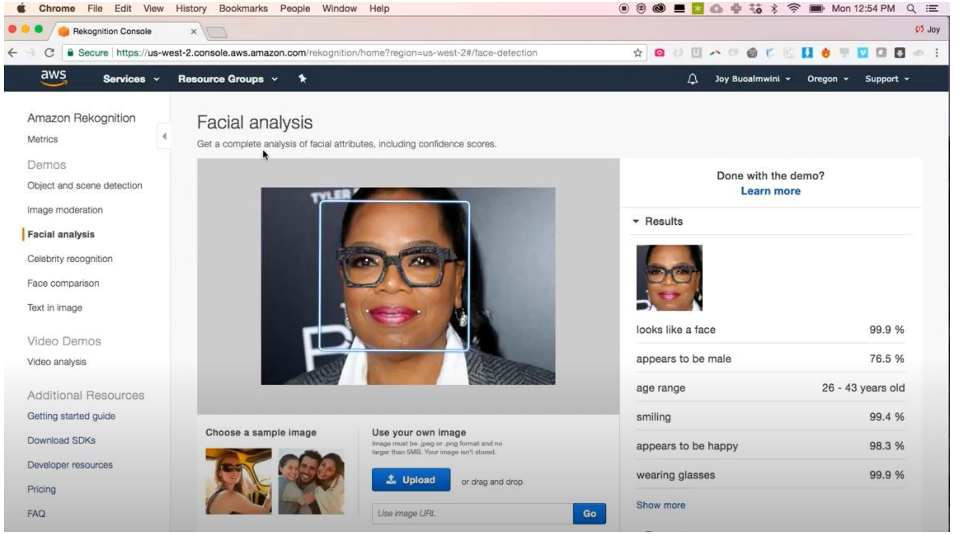


BLOG: https://blog.realeyesit.com/mona-lisas-smile-in-the-mind

Motivation

- Deep neural networks (DNNs) have powerful capacity for modeling complex input patterns





Facial analysis

Get a complete analysis of facial attributes, including confidence scores.

looks like a face	99.9 %
appears to be male	76.5 %
age range	26 - 43 years old
smiling	99.4 %
appears to be happy	98.3 %
wearing glasses	99.9 %

Image must be .jpeg or .png format and no larger than 5MB. Your image isn't stored.

BLOG: https://blog.realeyesit.com/mona-lisas-smile-in-the-mind

YOUTUBE / JOY BUOLAMWINI

Motivation

- Deep neural networks (DNNs) have powerful capacity for modeling complex input patterns

But they can also easily overfit to training set biases!

Motivation

- What is a training set bias?
    - The training set is drawn from a joint distribution that is different from the distribution of the evaluation set

Motivation

- What is a training set bias?
    - The training set is drawn from a joint distribution that is different from the distribution of the evaluation set

- Bias can creep in while:
    - Collecting the data
    - Preparing the data

Motivation

- What is a training set bias?
    - The training set is drawn from a joint distribution that is different from the distribution of the evaluation set

- Bias can creep in while:
    - Collecting the data
    - Preparing the data

How can we handle bias in the data?

Overview

- Motivation
- Related work
    - Data resampling
    - Training loss based
    - Shortcomings
- Learning to reweight examples
    - Intuition
    - Meta-learning perspective
    - Online approximation
    - The algorithm
- Experiment results

Related work

- Bias forms:
    - Class imbalance
    - Noisy labels

Related work

- Bias forms:
  - Class imbalance
  - Noisy labels

- Common approaches to handle training set bias:
  - Dataset resampling
  - Reweighting the examples

Related work

- **Class imbalance problem:**
    - Resampling
        - Under-sampling

Related work

- **Class imbalance problem:**
    - Resampling
        - Under-sampling
            - Removing examples from the majority class

Related work

- **Class imbalance problem:**
    - Resampling
        - Under-sampling
        - Over-sampling

Related work

- **Class imbalance problem:**
  - Resampling
    - Under-sampling
    - Over-sampling
      - Sampling multiple copies of the minority class

Related work

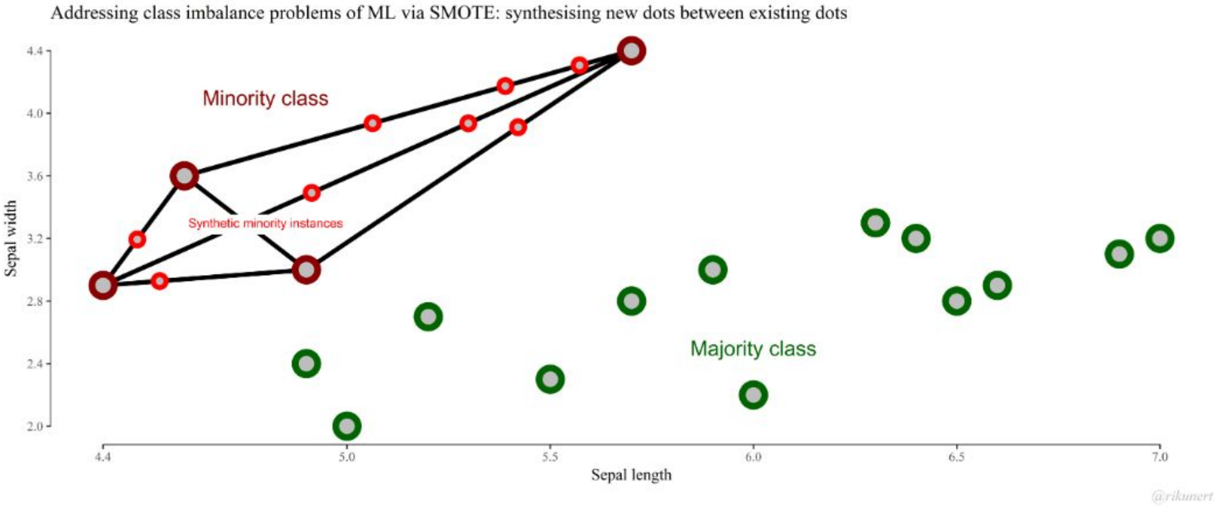
- **Class imbalance problem:**
    - Resampling
        - Under-sampling
        - Over-sampling
        - SMOTE

Related work

- **Class imbalance problem:**
  - Resampling
    - Under-sampling
    - Over-sampling
    - SMOTE
      - Combination of under-sampling and over-sampling

Related work

- **SMOTE**: **S**ynthetic **M**inority **O**versampling **TE**chnique
  - Generates synthetic data for the minority class



Related work

- **Class imbalance problem:**
    - Resampling
        - Under-sampling
        - Over-sampling
        - SMOTE

    - Weighting
        - Cost sensitive weighting

Related work

- Cost sensitive weighting: **AdaBoost** - **Ada**ptive **Boost**ing
    - Wisdom of the crowd!

Related work

- Cost sensitive weighting: **AdaBoost** - **Ada**ptive **Boost**ing
  - Wisdom of the crowd!

Weak classifier - 1

Related work

- Cost sensitive weighting: **AdaBoost** - **Ada**ptive **Boost**ing
    - Wisdom of the crowd!

Weak classifier - 1

Weak classifier - 2

Related work

- Cost sensitive weighting: **AdaBoost** - **Adap**tive **Boost**ing
    - Wisdom of the crowd!

Weak classifier - 1

= weights

Weak classifier - 2

Related work

- Cost sensitive weighting: **AdaBoost** - **Ada**ptive **Boost**ing
    - Wisdom of the crowd!

Weak classifier - 1

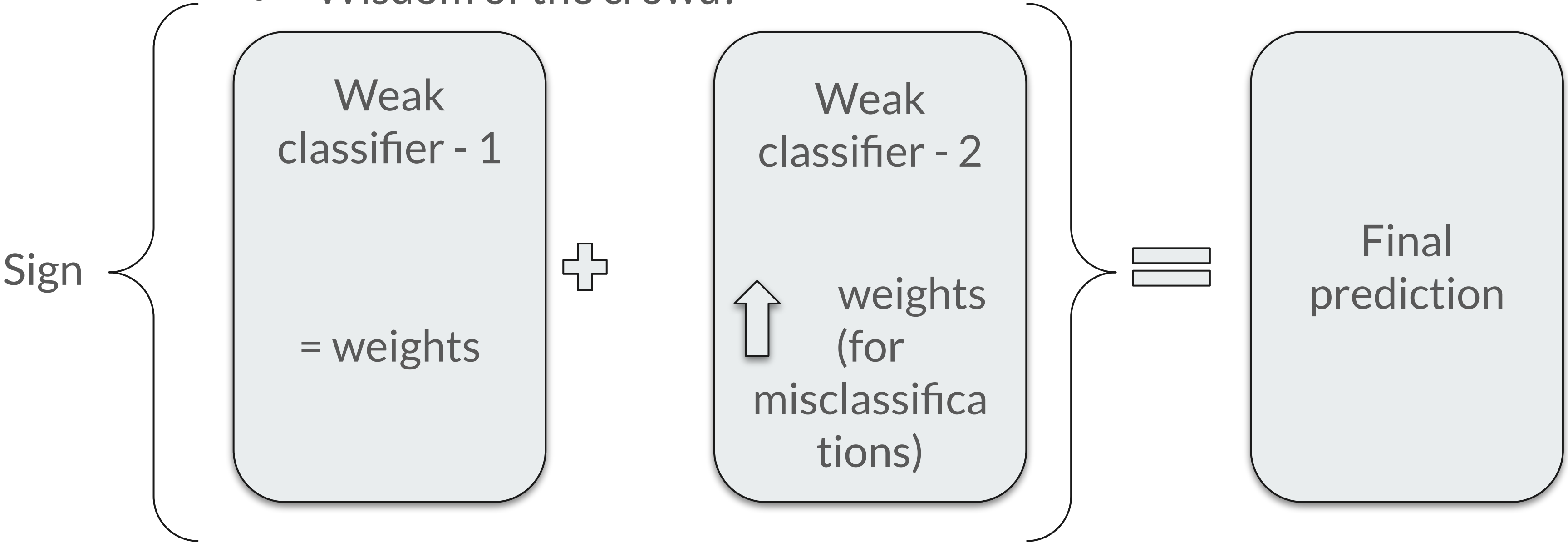
= weights

Weak classifier - 2

⇧ weights (for misclassifications)

Related work

- Cost sensitive weighting: **AdaBoost** - **Ada**ptive **Boost**ing
    - Wisdom of the crowd!



Sign

Weak classifier - 1

= weights

+

Weak classifier - 2

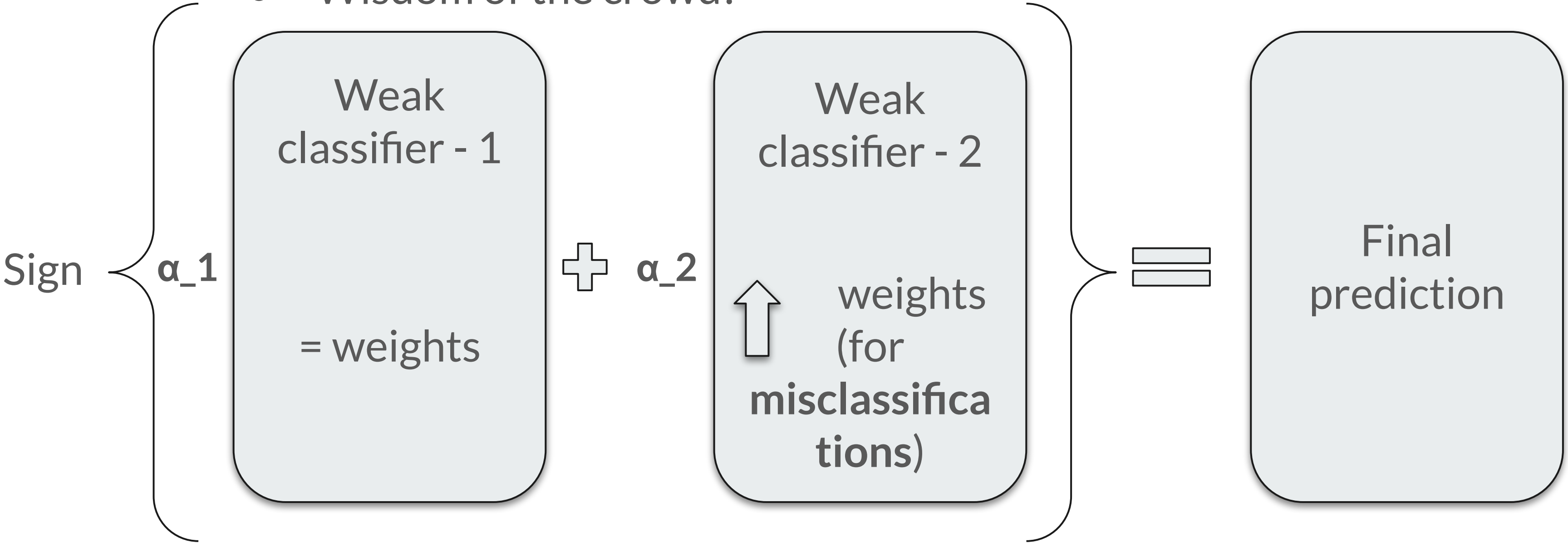
↑ weights (for misclassifications)

=

Final prediction

Related work

- Cost sensitive weighting: **AdaBoost** - **Adap**tive **Boost**ing
    - Wisdom of the crowd!



Sign

α_1

Weak classifier - 1

= weights

α_2

Weak classifier - 2

weights (for **misclassifications)**

Final prediction

Related work

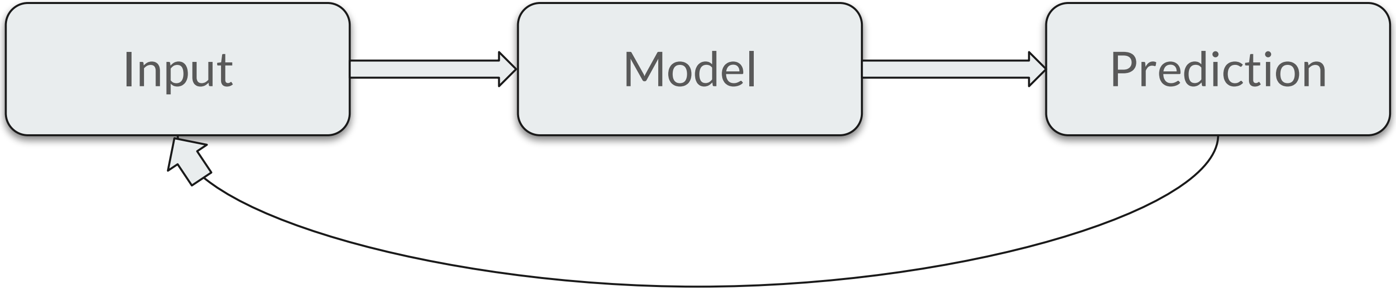
- **Noisy labels problem:**
    - Resampling
        - Model prediction to bootstrap labels

Related work

- Resampling: Model prediction to bootstrap labels
    - Self-trains a learning agent to generate a model and iteratively classify unlabeled or noisy labeled examples

Related work

- Resampling: Model prediction to bootstrap labels
    - Self-trains a learning agent to generate a model and iteratively classify unlabeled or noisy labeled examples



Input

Model

Prediction

Related work

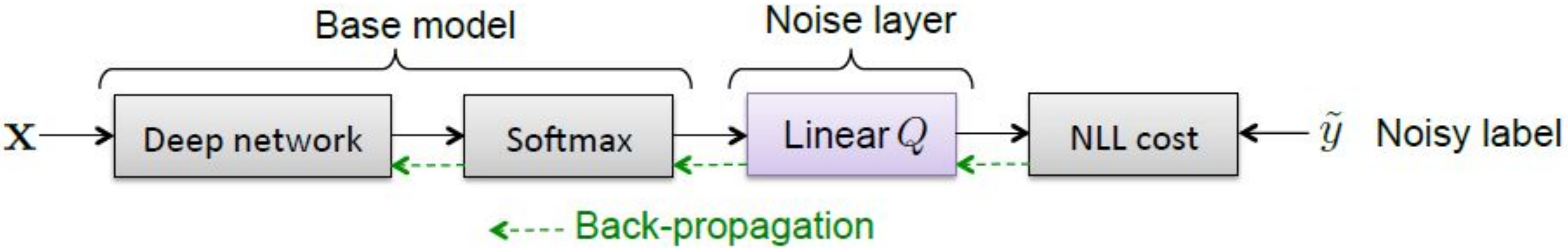
- **Noisy labels problem:**
    - Resampling
        - Model prediction to bootstrap labels
    - Weighting
        - Noise adaptation layer

Related work

- Weighting: Using a noise adaptation layer
    - Add a linear noise layer with weights equal to the noise distribution

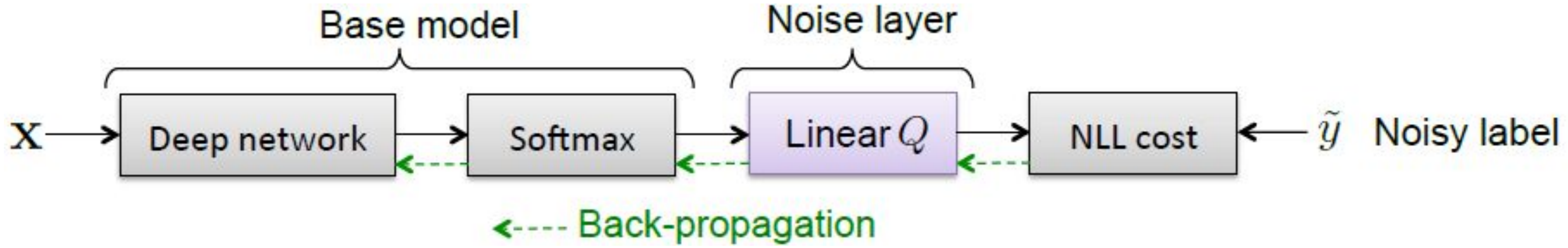
Related work

- Weighting: Using a noise adaptation layer
    - Add a linear noise layer with weights equal to the noise distribution



Related work

- Weighting: Using a noise adaptation layer
    - Add a linear noise layer with weights equal to the noise distribution



    - Weights of the special layer are probabilities of being mislabelled

Shortcomings

- The example weights are calculated based on the training loss

Shortcomings

- The example weights are calculated based on the training loss
- Contradicting ideas in training loss based approaches:

	High Loss	Low Loss
Class imbalance		
Noisy labels		

Shortcomings

- The example weights are calculated based on the training loss
- Contradicting ideas in training loss based approaches:

	High Loss	Low Loss
Class imbalance	✓	
Noisy labels		

Shortcomings

- The example weights are calculated based on the training loss
- Contradicting ideas in training loss based approaches:

	High Loss	Low Loss
Class imbalance	✓	
Noisy labels		✓

Shortcomings

- The example weights are calculated based on the training loss
- Contradicting ideas in training loss based approaches:

	High Loss	**Low Loss**
Class imbalance	✓	
Noisy labels		✓

What if the data has both class imbalance and noisy labels?

The story so far...

- Bias forms:
  - Class imbalance
  - Noisy labels

The story so far...

- Bias forms:
    - Class imbalance
    - Noisy labels
- Common approaches to handle training set bias:
    - Dataset resampling
    - Reweighting the examples

The story so far...

- Bias forms:
  - Class imbalance
  - Noisy labels
- Common approaches to handle training set bias:
  - Dataset resampling
  - Reweighting the examples
- Contradicting ideas in training loss based approaches

What if the data has both class imbalance and noisy labels?

Overview

- Motivation
- Related work
    - Data resampling
    - Training loss based
    - Shortcomings
- Learning to reweight examples
    - Intuition
    - Meta-learning perspective
    - Online approximation
    - The algorithm
- Experiment results

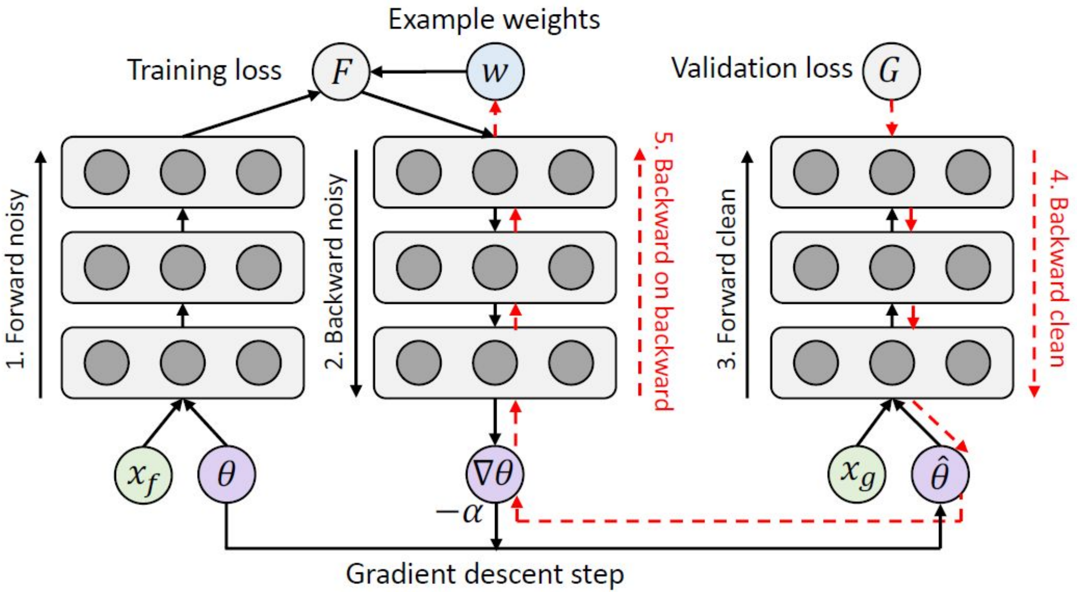
Proposed solution

Solving the training set bias problem is inherently ill-defined without an unbiased test set

Proposed solution

‘The best example weighting should minimize the loss of a set of unbiased clean **validation** examples’

Learning to reweight in an MLP network



Learning to reweight examples

Input-target pair: (x, y)

Learning to reweight examples

Input-target pair: (x, y)

Train set: $\{(x_i, y_i), 1 \leq i \leq N\}$

Learning to reweight examples

Input-target pair: (x, y)

Train set: $\{(x_i, y_i), 1 \leq i \leq N\}$

Validation set: $\{(x_i^v, y_i^v), 1 \leq i \leq M\}$

Learning to reweight examples

Input-target pair: (x, y)

Train set: $\{(x_i, y_i), 1 \leq i \leq N\}$

Validation set: $\{(x_i^v, y_i^v), 1 \leq i \leq M\}$

NN model: $\Phi(x, \theta)$

Learning to reweight examples

Input-target pair: (x, y)

Train set: $\{(x_i, y_i), 1 \leq i \leq N\}$

Validation set: $\{(x_i^v, y_i^v), 1 \leq i \leq M\}$

NN model: $\Phi(x, \theta)$

Loss function: $C(\hat{y}, y)$

Learning to reweight examples

- In standard training:-
  - Minimize the expected loss for the training set -

$$\frac{1}{N}\sum_{i=1}^{N} C(\hat{y}_i, y_i) = \frac{1}{N}\sum_{i=1}^{N} f_i(\theta)$$

Learning to reweight examples

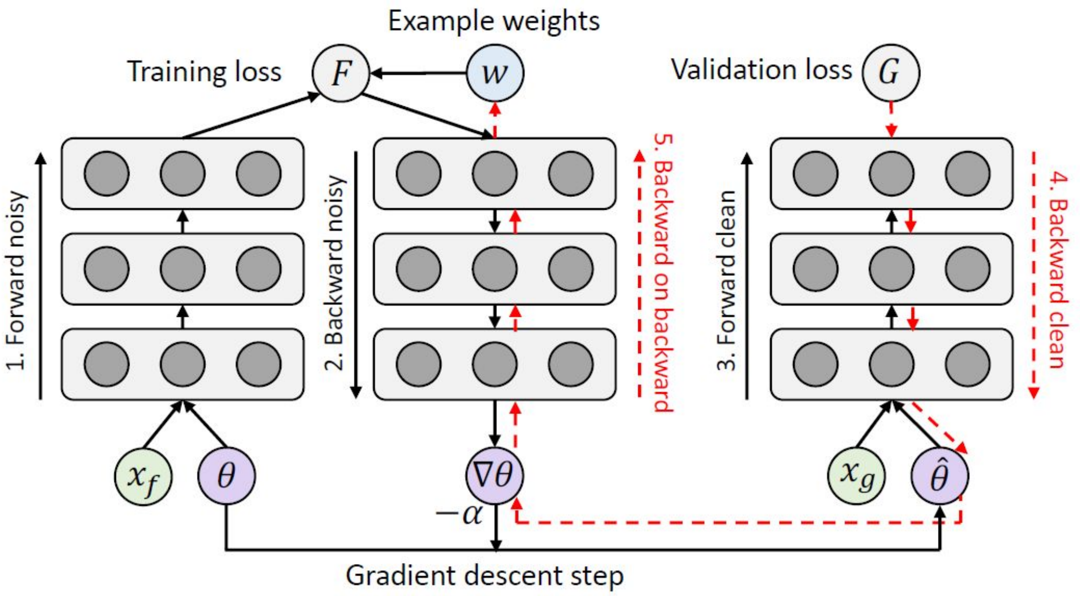
- In standard training:-
    - Minimize the expected loss for the training set -

$$\frac{1}{N}\sum_{i=1}^{N} C(\hat{y}_i, y_i) = \frac{1}{N}\sum_{i=1}^{N} f_i(\theta)$$

- Here:-
    - Minimize a weighted loss for the training set -

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{N} w_i f_i(\theta)$$

Learning to reweight in an MLP network



Learning to reweight examples

Meta-learning perspective

- Here:-
  - Minimize a weighted loss for the training set

$$\theta^*(w) = \arg\ \min_{\theta} \sum_{i=1}^{N} w_i f_i(\theta)$$

Learning to reweight examples

Meta-learning perspective

- Here:-
    - Minimize a weighted loss for the training set

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{N} w_i f_i(\theta)$$

    - Optimal selection of weights is based on its validation performance:

$$w^* = \arg\min_{w, w \geq 0} \frac{1}{M} \sum_{i=1}^{M} f_i^v(\theta^*(w))$$

Learning to reweight examples

Meta-learning perspective

- Here:-
  - Minimize a weighted loss for the training set

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{N} w_i f_i(\theta)$$

  - Optimal selection of weights is based on its validation performance:

$$w^* = \arg\min_{w, w \geq 0} \frac{1}{M} \sum_{i=1}^{M} f_i^v(\theta^*(w))$$

The weights have to be non-negative since minimizing negative train loss can usually result in unstable behaviour

Learning to reweight examples

Meta-learning perspective

- Two nested loops of optimization are required for calculating the optimal weights

Learning to reweight examples

Meta-learning perspective

- Two nested loops of optimization are required for calculating the optimal weights

INNER LOOP

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{N} w_i f_i(\theta)$$

Optimizing the model parameters with weights given already

Learning to reweight examples

Meta-learning perspective

- Two nested loops of optimization are required for calculating the optimal weights

INNER LOOP

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{N} w_i f_i(\theta)$$

Optimizing the model parameters with weights given already

OUTER LOOP

$$w^* = \arg\min_{w, w \geq 0} \frac{1}{M} \sum_{i=1}^{M} f_i^v(\theta^*(w))$$

Optimizing the weights based on its performance in the validation set - with the new parameters

Learning to reweight examples

Meta-learning perspective

- Two nested loops of optimization are required for calculating the optimal weights

INNER LOOP

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{N} w_i f_i(\theta)$$

Optimizing the model parameters with weights given already

OUTER LOOP

$$w^* = \arg\min_{w, w \geq 0} \frac{1}{M} \sum_{i=1}^{M} f_i^v(\theta^*(w))$$

Optimizing the weights based on its performance in the validation set - with the new parameters

Too time consuming!

Learning to reweight examples

Meta-learning perspective

- Two nested loops of optimization are required for calculating the optimal weights

INNER LOOP

$$\theta^*(w) = \arg\min_{\theta} \sum_{i=1}^{N} w_i f_i(\theta)$$

OUTER LOOP

$$w^* = \arg\min_{w, w \geq 0} \frac{1}{M} \sum_{i=1}^{M} f_i^v(\theta^*(w))$$

- **Motivation:** To use one single loop to adapt online weights

Learning to reweight examples

Online Approximation

- Typically, SGD is used to optimize loss functions
- To get a cheap estimate, here, a **single gradient descent step** is taken on a mini-batch of validation samples

Learning to reweight examples

Algorithm so far...

- **Step -1:** Sample a mini batch of the training data and validation data

Learning to reweight examples

Algorithm so far...

- **Step -1:** Sample a mini batch of the training data and validation data
- **Step -2:** Forward pass on the training data with the initial parameters

Learning to reweight examples

Algorithm so far...

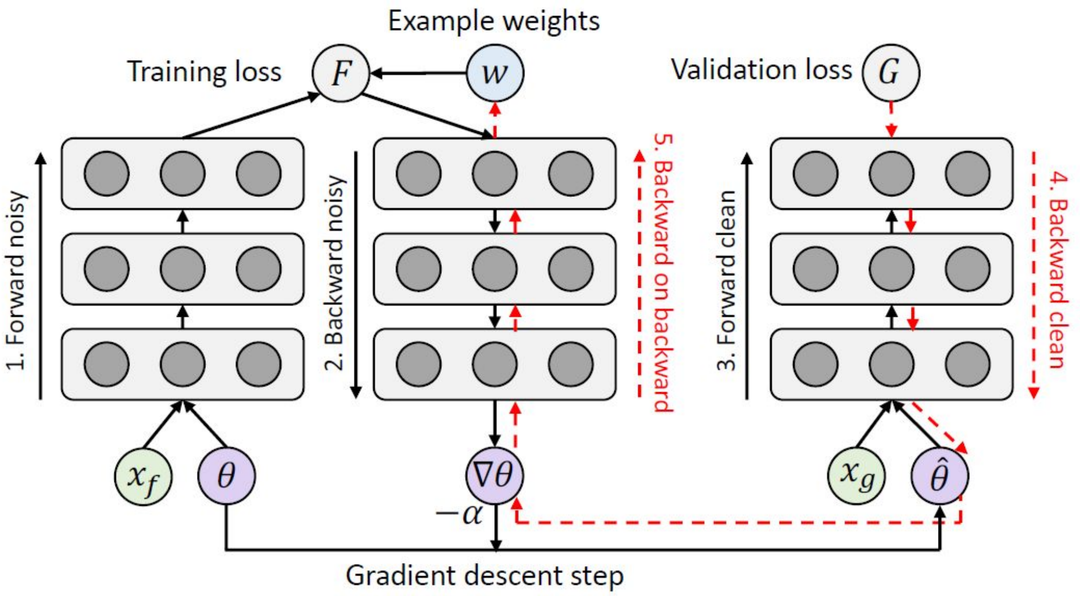
- **Step -1:** Sample a mini batch of the training data and validation data
- **Step -2:** Forward pass on the training data with the initial parameters
- **Step-3:** Calculate the weighted training loss (initial weights = 0)

Learning to reweight examples

Algorithm so far...

- **Step -1:** Sample a mini batch of the training data and validation data
- **Step -2:** Forward pass on the training data with the initial parameters
- **Step-3:** Calculate the weighted training loss (initial weights = 0)
- **Step-4:** Backpropagate on the training data and get the updated parameters (no update here since weights = 0)

Learning to reweight in an MLP network



Learning to reweight examples

Algorithm so far...

- **Step -5:** Forward pass on the validation set with the updated parameters

Learning to reweight examples

Algorithm so far...

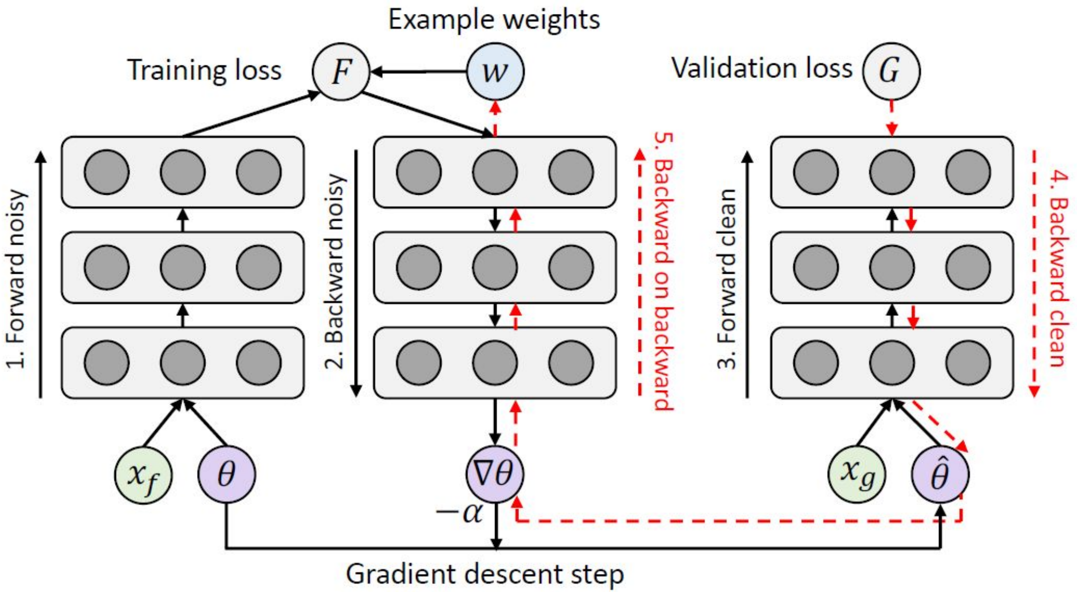
- **Step -5:** Forward pass on the validation set with the updated parameters
- **Step -6:** Calculate the loss on the validation set

Learning to reweight examples

Algorithm so far...

- **Step -5:** Forward pass on the validation set with the updated parameters
- **Step -6:** Calculate the loss on the validation set
- **Step-7:** Backpropagate on the validation set and get the updated weights

Learning to reweight in an MLP network



Learning to reweight examples

Algorithm so far...

- **Step-8:** Calculate the weighted training loss with the updated weights

Learning to reweight examples

Algorithm so far...

- **Step-8:** Calculate the weighted training loss with the updated weights
- **Step-9:** Backpropagate on the training data and get the updated parameters

Learning to reweight examples

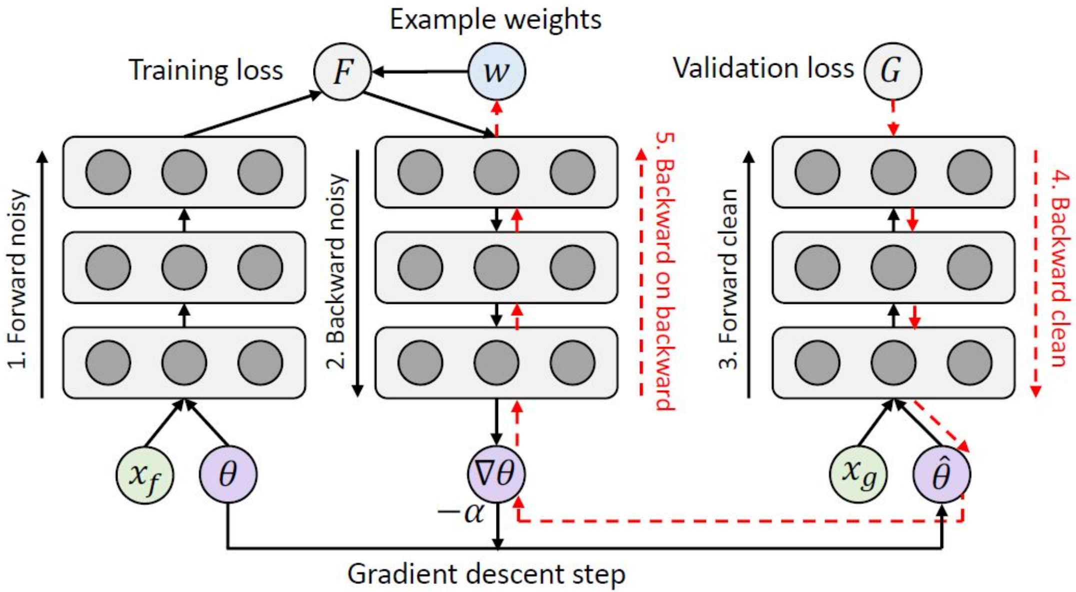
Algorithm so far...

- **Step-8:** Calculate the weighted training loss with the updated weights

- **Step-9:** Backpropagate on the training data and get the updated parameters

This completes one iteration!

Learning to reweight in an MLP network



Learning to reweight examples

Online Approximation

- At step ***t***, take a single gradient descent step on a mini-batch of validation samples wrt **ϵ**

$$u_{i,t} = -\eta \frac{\partial}{\partial \epsilon_{i,t}} \frac{1}{m} \sum_{j=1}^{m} f_j^v(\theta_{t+1}(\epsilon)) \Big|_{\epsilon_{i,t}=0}$$

Descent step size on **ϵ**

Learning to reweight examples

Online Approximation

- Rectify the output to get a non-negative weighting:

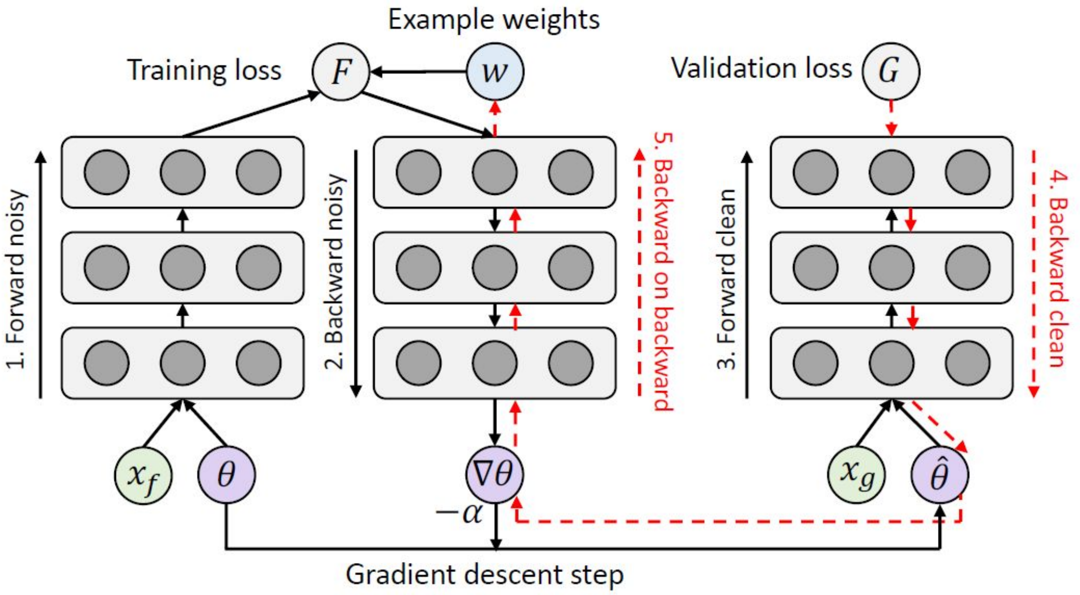
$$\tilde{w}_{i,t} = \max(u_{i,t}, 0).$$

- Normalizing the weights (to sum up to one):

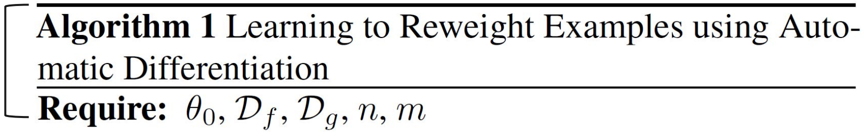
$$w_{i,t} = \frac{\tilde{w}_{i,t}}{(\sum_j \tilde{w}_{j,t}) + \delta(\sum_j \tilde{w}_{j,t})}$$

1 if w's are zeros
0 otherwise

Learning to reweight in an MLP network



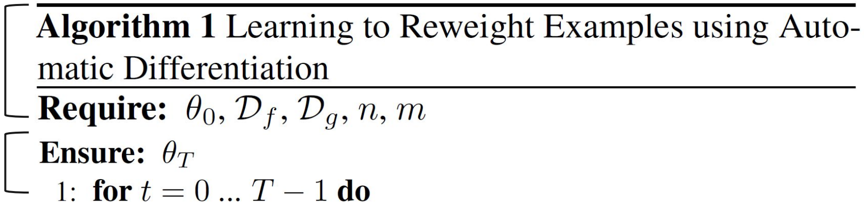
THE algorithm



Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

```
Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$
```

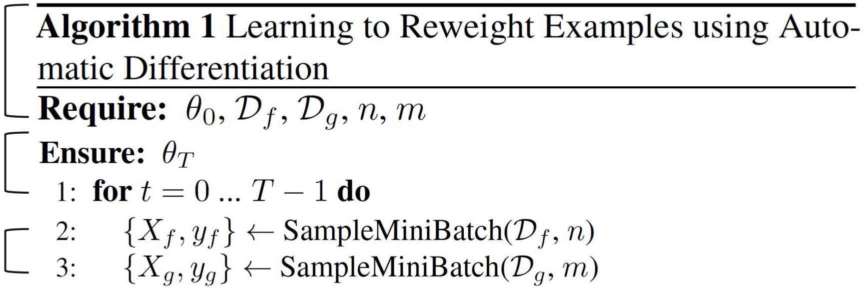
THE algorithm



Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

```
Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$
Ensure: $\theta_T$
for $t = 0 ... T - 1$ do
```

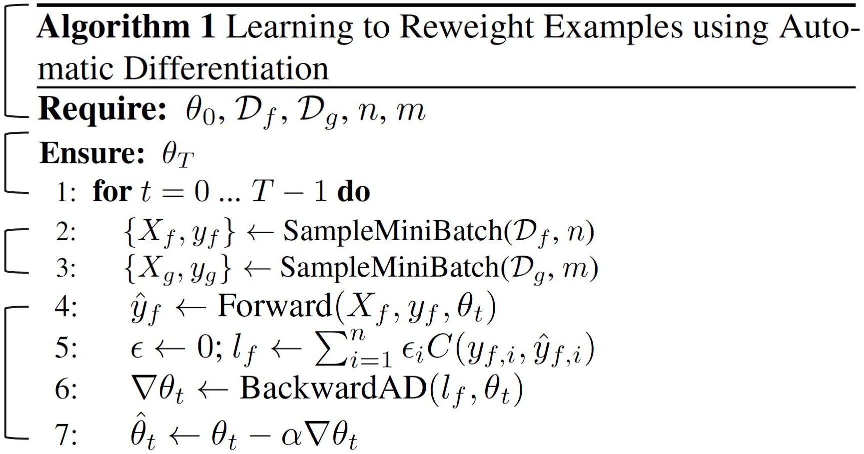
THE algorithm



Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

```
Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$
Ensure: $\theta_T$
for $t = 0 \ldots T - 1$ do
    $\{X_f, y_f\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_f, n)$
    $\{X_g, y_g\} \leftarrow \text{SampleMiniBatch}(\mathcal{D}_g, m)$
```

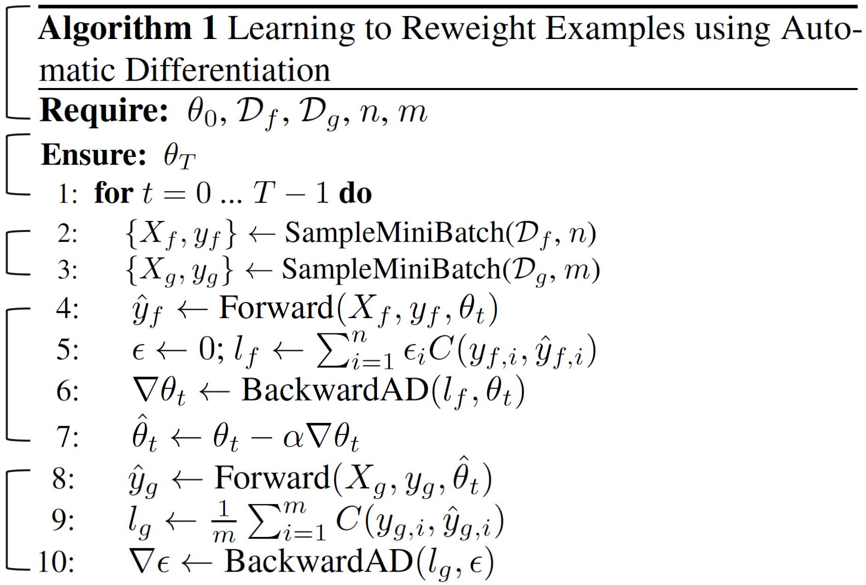
THE algorithm



Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

```
Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$
Ensure: $\theta_T$
for $t = 0 \ldots T - 1$ do
    $\{X_f, y_f\} \leftarrow$ SampleMiniBatch($\mathcal{D}_f$, $n$)
    $\{X_g, y_g\} \leftarrow$ SampleMiniBatch($\mathcal{D}_g$, $m$)
    $\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$
    $\epsilon \leftarrow 0; l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$
    $\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$
    $\hat{\theta}_t \leftarrow \theta_t - \alpha\nabla\theta_t$
```

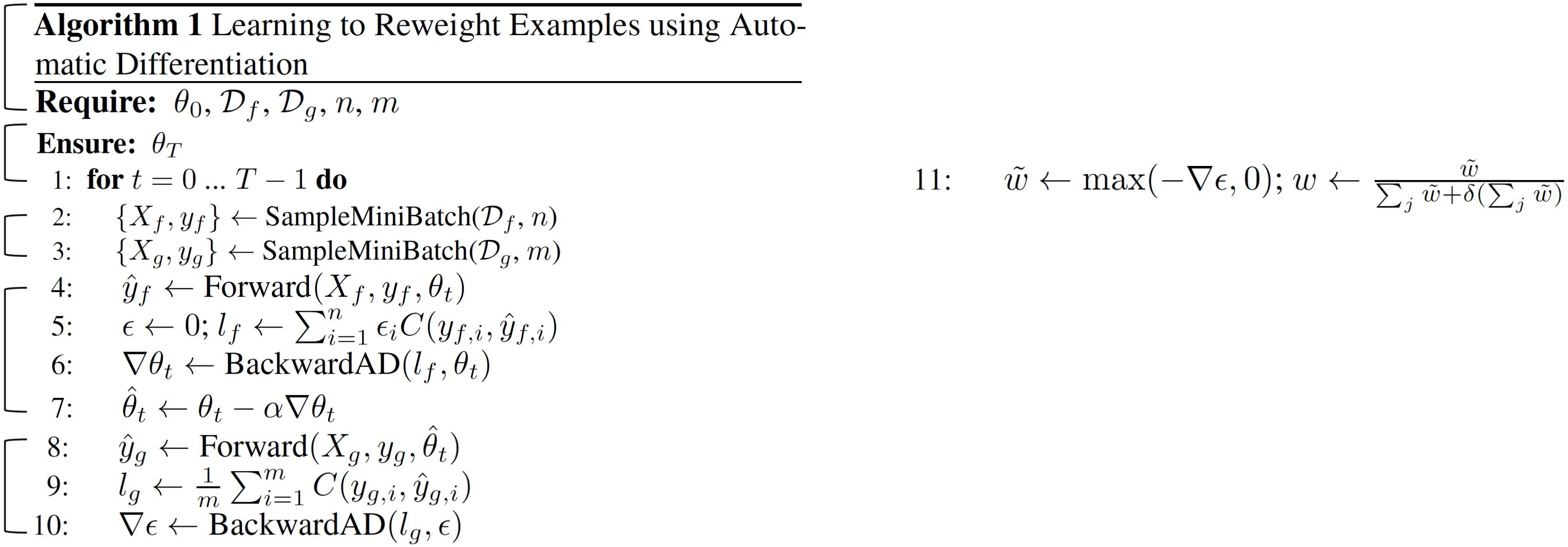
THE algorithm



Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

```
Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$
Ensure: $\theta_T$
for $t = 0 \ldots T - 1$ do
    $\{X_f, y_f\} \leftarrow$ SampleMiniBatch($\mathcal{D}_f, n$)
    $\{X_g, y_g\} \leftarrow$ SampleMiniBatch($\mathcal{D}_g, m$)
    $\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$
    $\epsilon \leftarrow 0; l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$
    $\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$
    $\hat{\theta}_t \leftarrow \theta_t - \alpha\nabla\theta_t$
    $\hat{y}_g \leftarrow \text{Forward}(X_g, y_g, \hat{\theta}_t)$
    $l_g \leftarrow \frac{1}{m} \sum_{i=1}^{m} C(y_{g,i}, \hat{y}_{g,i})$
    $\nabla\epsilon \leftarrow \text{BackwardAD}(l_g, \epsilon)$
```

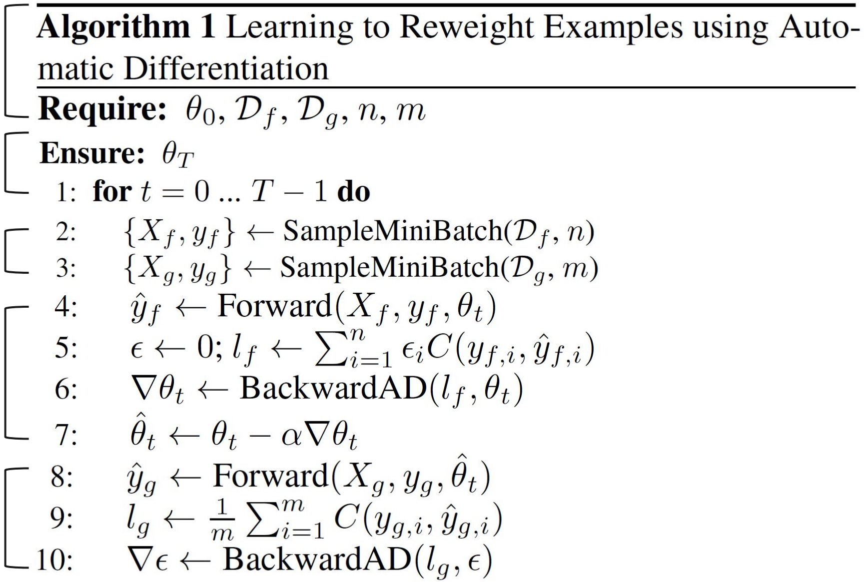
THE algorithm



Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

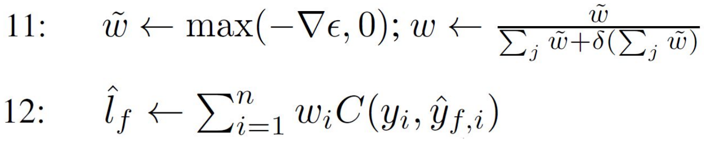
```
Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$
Ensure: $\theta_T$
for $t = 0 \dots T - 1$ do
    $\{X_f, y_f\} \leftarrow$ SampleMiniBatch($\mathcal{D}_f, n$)
    $\{X_g, y_g\} \leftarrow$ SampleMiniBatch($\mathcal{D}_g, m$)
    $\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$
    $\epsilon \leftarrow 0; l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$
    $\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$
    $\hat{\theta}_t \leftarrow \theta_t - \alpha\nabla\theta_t$
    $\hat{y}_g \leftarrow \text{Forward}(X_g, y_g, \hat{\theta}_t)$
    $l_g \leftarrow \frac{1}{m} \sum_{i=1}^{m} C(y_{g,i}, \hat{y}_{g,i})$
    $\nabla\epsilon \leftarrow \text{BackwardAD}(l_g, \epsilon)$
    $\tilde{w} \leftarrow \max(-\nabla\epsilon, 0); w \leftarrow \frac{\tilde{w}}{\sum_j \tilde{w} + \delta(\sum_j \tilde{w})}$
```

THE algorithm



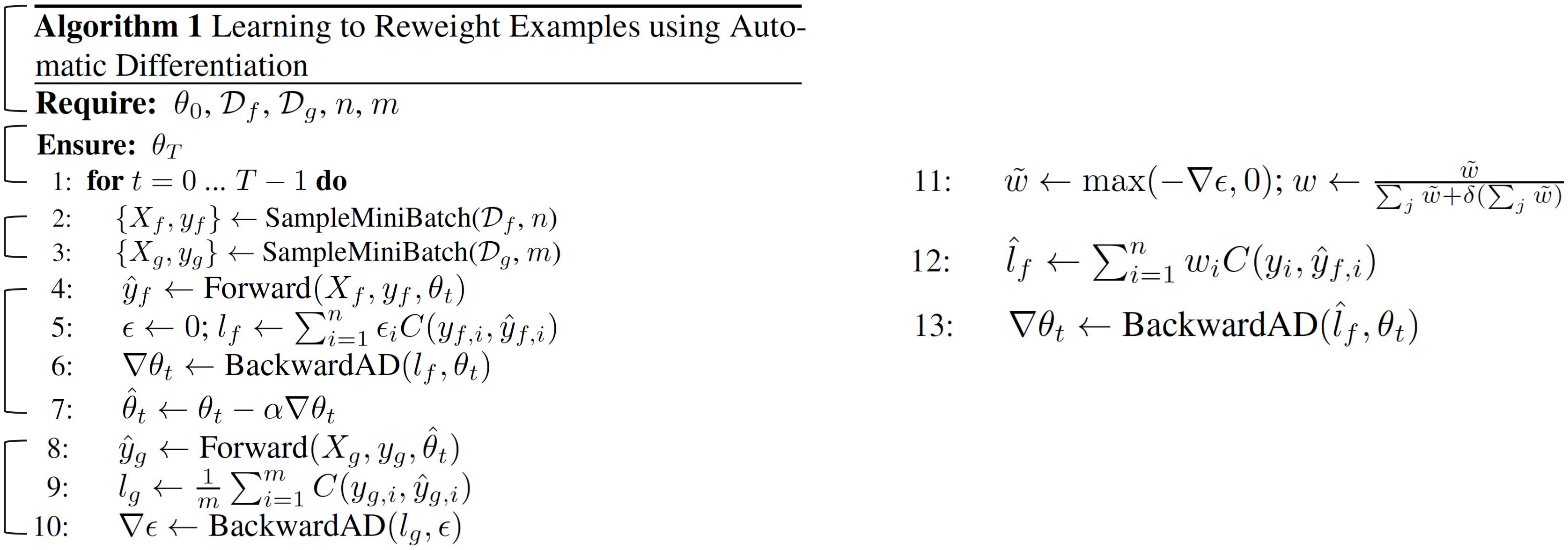
Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

```
Require: $\theta_0$, $\mathcal{D}_f$, $\mathcal{D}_g$, $n$, $m$
Ensure: $\theta_T$
for $t = 0 \ldots T - 1$ do
    $\{X_f, y_f\} \leftarrow$ SampleMiniBatch($\mathcal{D}_f$, $n$)
    $\{X_g, y_g\} \leftarrow$ SampleMiniBatch($\mathcal{D}_g$, $m$)
    $\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$
    $\epsilon \leftarrow 0; l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$
    $\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$
    $\hat{\theta}_t \leftarrow \theta_t - \alpha\nabla\theta_t$
    $\hat{y}_g \leftarrow \text{Forward}(X_g, y_g, \hat{\theta}_t)$
    $l_g \leftarrow \frac{1}{m}\sum_{i=1}^{m} C(y_{g,i}, \hat{y}_{g,i})$
    $\nabla\epsilon \leftarrow \text{BackwardAD}(l_g, \epsilon)$
```



```
    $\tilde{w} \leftarrow \max(-\nabla\epsilon, 0); w \leftarrow \frac{\tilde{w}}{\sum_j \tilde{w} + \delta(\sum_j \tilde{w})}$
    $\hat{l}_f \leftarrow \sum_{i=1}^{n} w_i C(y_i, \hat{y}_{f,i})$
```

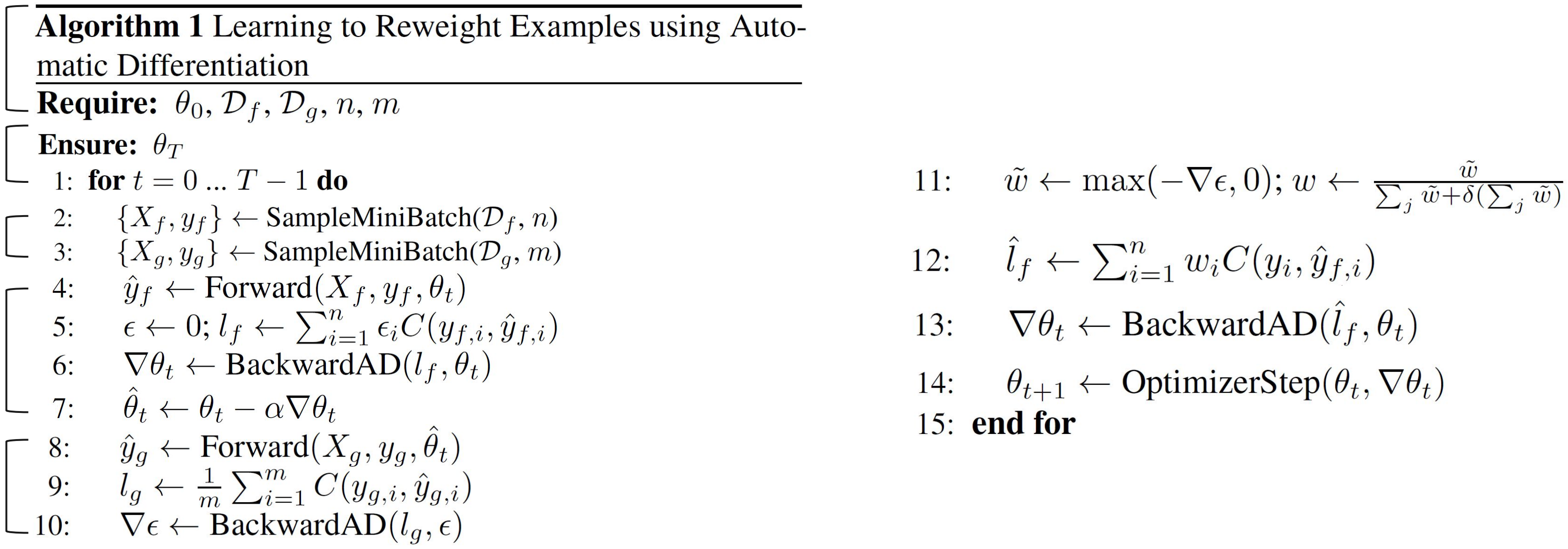
THE algorithm



Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

```
Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$
Ensure: $\theta_T$
for $t = 0 \ldots T-1$ do
    $\{X_f, y_f\} \leftarrow$ SampleMiniBatch($\mathcal{D}_f, n$)
    $\{X_g, y_g\} \leftarrow$ SampleMiniBatch($\mathcal{D}_g, m$)
    $\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$
    $\epsilon \leftarrow 0; l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$
    $\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$
    $\hat{\theta}_t \leftarrow \theta_t - \alpha\nabla\theta_t$
    $\hat{y}_g \leftarrow \text{Forward}(X_g, y_g, \hat{\theta}_t)$
    $l_g \leftarrow \frac{1}{m}\sum_{i=1}^{m} C(y_{g,i}, \hat{y}_{g,i})$
    $\nabla\epsilon \leftarrow \text{BackwardAD}(l_g, \epsilon)$
    $\tilde{w} \leftarrow \max(-\nabla\epsilon, 0); w \leftarrow \frac{\tilde{w}}{\sum_j \tilde{w} + \delta(\sum_j \tilde{w})}$
    $\hat{l}_f \leftarrow \sum_{i=1}^{n} w_i C(y_i, \hat{y}_{f,i})$
    $\nabla\theta_t \leftarrow \text{BackwardAD}(\hat{l}_f, \theta_t)$
```

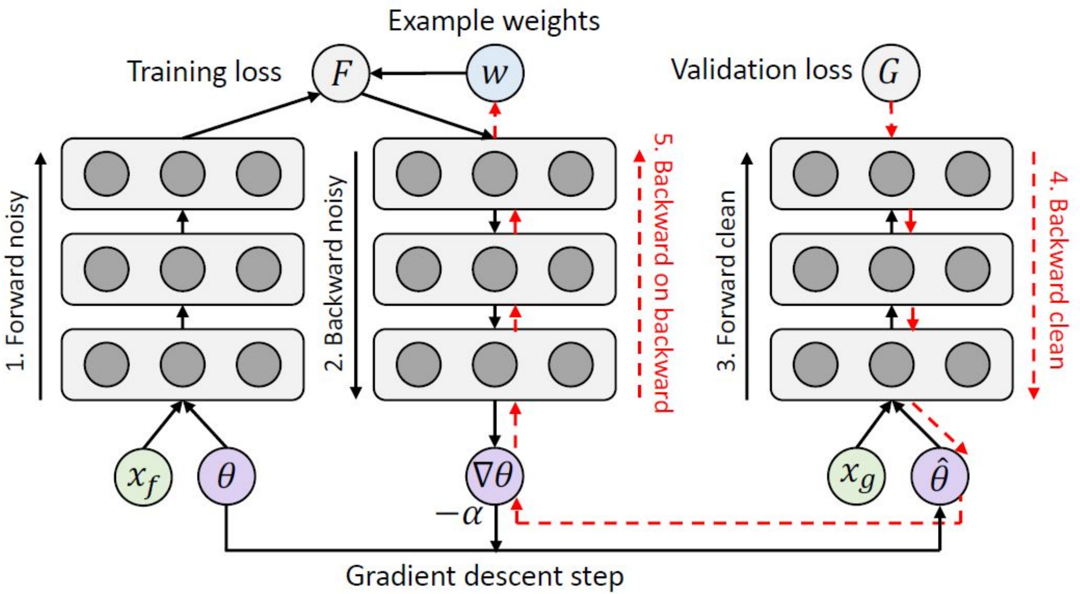
THE algorithm



Algorithm 1 Learning to Reweight Examples using Automatic Differentiation

```
Require: $\theta_0, \mathcal{D}_f, \mathcal{D}_g, n, m$
Ensure: $\theta_T$
for $t = 0 \ldots T - 1$ do
    $\{X_f, y_f\} \leftarrow$ SampleMiniBatch($\mathcal{D}_f, n$)
    $\{X_g, y_g\} \leftarrow$ SampleMiniBatch($\mathcal{D}_g, m$)
    $\hat{y}_f \leftarrow \text{Forward}(X_f, y_f, \theta_t)$
    $\epsilon \leftarrow 0; l_f \leftarrow \sum_{i=1}^{n} \epsilon_i C(y_{f,i}, \hat{y}_{f,i})$
    $\nabla\theta_t \leftarrow \text{BackwardAD}(l_f, \theta_t)$
    $\hat{\theta}_t \leftarrow \theta_t - \alpha\nabla\theta_t$
    $\hat{y}_g \leftarrow \text{Forward}(X_g, y_g, \hat{\theta}_t)$
    $l_g \leftarrow \frac{1}{m}\sum_{i=1}^{m} C(y_{g,i}, \hat{y}_{g,i})$
    $\nabla\epsilon \leftarrow \text{BackwardAD}(l_g, \epsilon)$
    $\tilde{w} \leftarrow \max(-\nabla\epsilon, 0); w \leftarrow \frac{\tilde{w}}{\sum_j \tilde{w} + \delta(\sum_j \tilde{w})}$
    $\hat{l}_f \leftarrow \sum_{i=1}^{n} w_i C(y_i, \hat{y}_{f,i})$
    $\nabla\theta_t \leftarrow \text{BackwardAD}(\hat{l}_f, \theta_t)$
    $\theta_{t+1} \leftarrow \text{OptimizerStep}(\theta_t, \nabla\theta_t)$
end for
```

Learning to reweight in an MLP network



Overview

- Motivation
- Related work
    - Data resampling
    - Training loss based
    - Shortcomings
- Learning to reweight examples
    - Intuition
    - Meta-learning perspective
    - Online approximation
    - The algorithm
- Experiment results

Experiments - Class imbalance

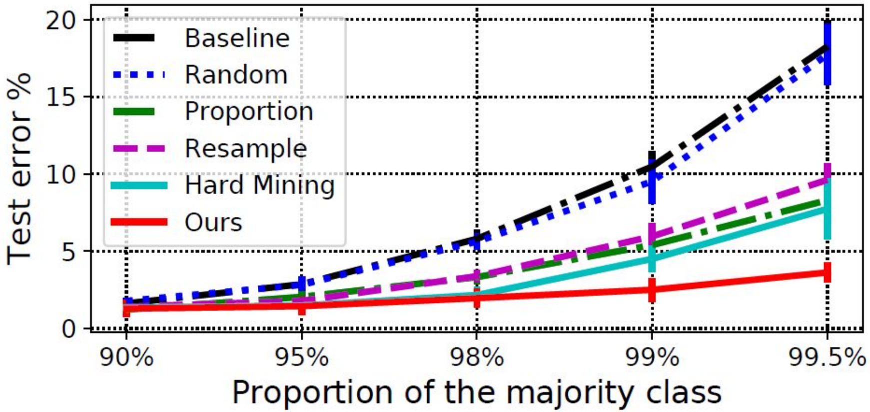


Figure 2. MNIST 4-9 binary classification error using a LeNet on imbalanced classes. Our method uses a small balanced validation split of 10 examples.

- **DATA:** MNIST (5000 images) with '4' being minority class and '9' the majority class
- **VALIDATION:** 10 images
- **MINI-BATCH SIZE:** 100
- **MODEL:** LeNet

Experiments - Noisy labels

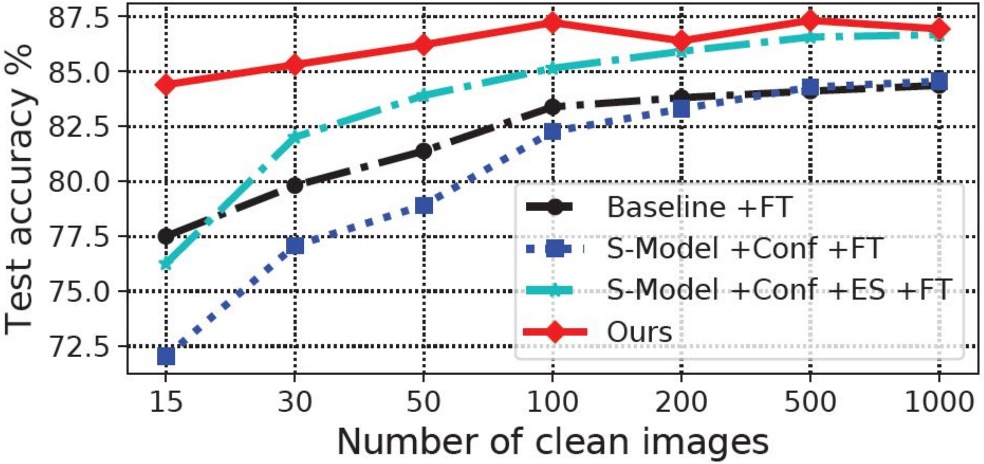


Figure 4. Effect of the number of clean imaged used, on CIFAR-10 with 40% of data flipped to label 3. "ES" denotes early stopping.

- **DATA:** CIFAR-10 with flipping 40% of the labels to another random class
- **VALIDATION:** 100 images
- **MINI-BATCH SIZE:** 100
- **MODEL:** MentorNet

Experiments - Class imbalance & Noisy labels

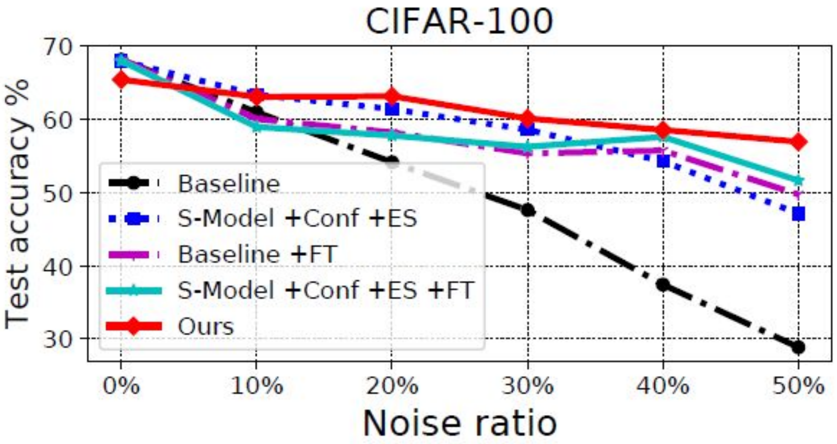


Figure 5. Model test accuracy on imbalanced noisy CIFAR experiments across various noise levels using a base ResNet-32 model. “ES” denotes early stopping, and “FT” denotes finetuning.

- **DATA:** CIFAR-100 with 40% flipped labels
- **VALIDATION:** 100 images
- **MINI-BATCH SIZE:** 100
- **MODEL:** ResNet-32

Summary

- A neat algorithm to handle data with both **class imbalance** and **noisy labels**
- **Validation at every step** to make training better
- **No additional hyperparameters** to tune

References

- Mengye et al. - Learning to Reweight Examples for Robust Deep Learning
- Oludare et al. - State-of-the-art in artificial neural network applications: A survey
- MIT Technology review - https://www.technologyreview.com/2019/02/04/137602/this-is-how-ai-bias-really-happensand-why-its-so-hard-to-fix/
- Reed et al. - Training Deep Neural Networks on Noisy Labels with Bootstrapping -
- Talk by Mengye - https://vimeo.com/287808016

References

- Oversampling and undersampling - https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis
- Chawla et al. - SMOTE: Synthetic Minority Over-sampling Technique
- SMOTE explained for noobs - https://rikunert.com/SMOTE_explained
- Sukhbaatar et al. - Learning from noisy labels with deep neural networks
- Kai Ming Ting - A comparative study of cost-sensitive boosting algorithm
- Dice Loss for Data-imbalanced NLP Tasks

The End