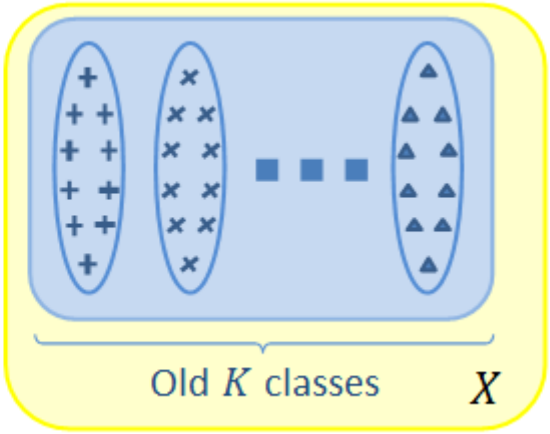
Overview

1. Problem Statement
2. Discussion of Several Methods
3. Our Approach
4. Experiments
5. Other Methods tried before

Problem Statement

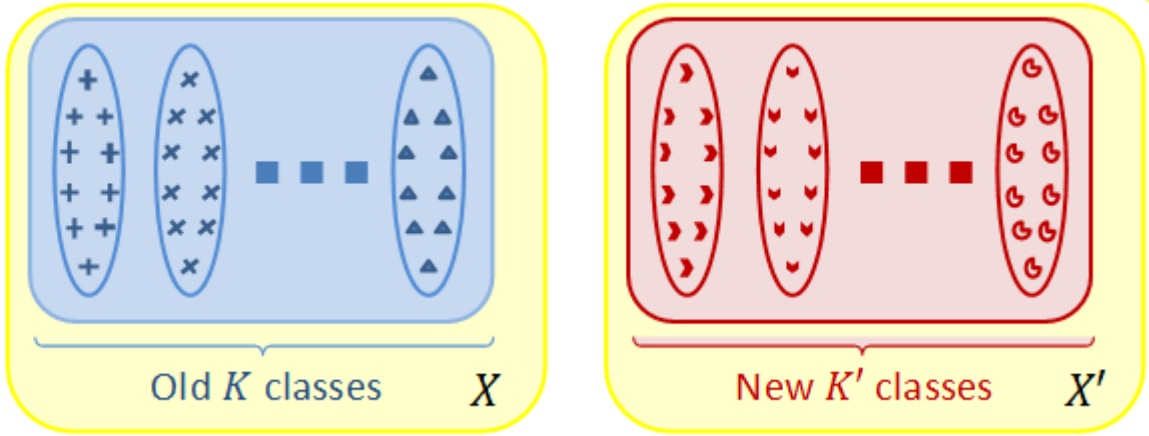
- Have the old model already trained

Training dataset for old model



- How to incrementally train the new model.

Training dataset for new model (Incremental training)

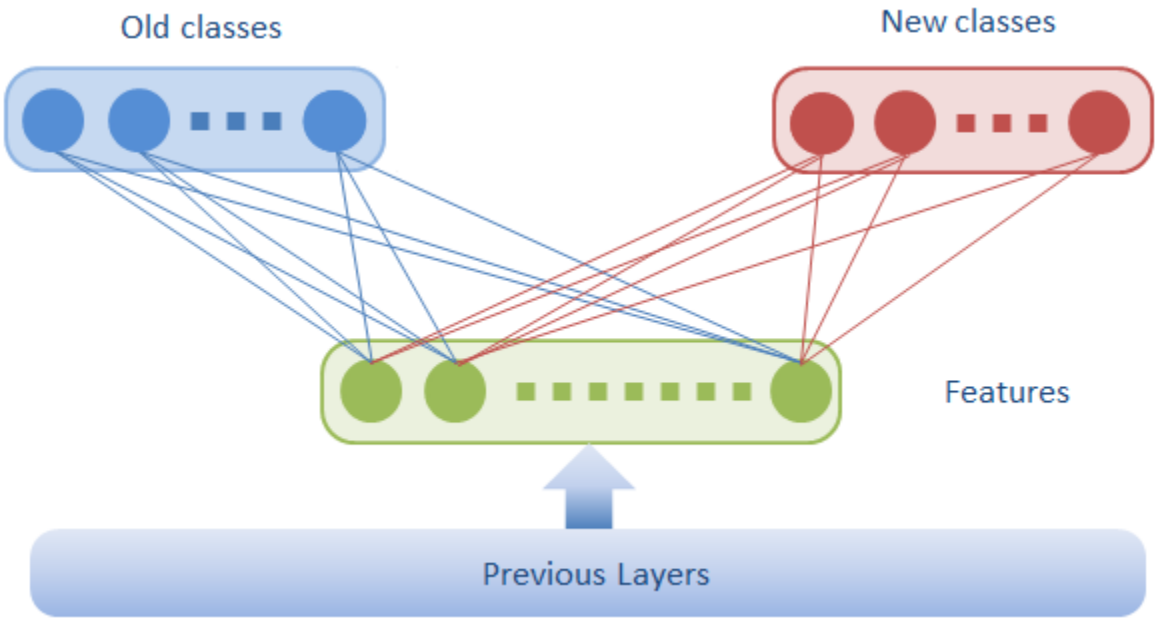


Overview

1. Problem Statement
2. Discussion of Several Methods
3. Our Approach
4. Experiments
5. Other Methods tried before

Recall Flat Expansion

- Add new classes directly, represented by red nodes
- Randomly initialize weights of the red connections
- Fine-tuning all weights on old & new training datasets



Two directions

A. Modify network structure / design loss function accordingly:

Super-class, sub-class; Attributes learning; Semantic embedding

1.Basically does not solve incremental learning problem
2.(More importantly) Focus on accuracy rather than efficiency

B. Focus on training data:

Curriculum Learning -> Adaptively adjust training dataset distribution

Extend -> AdaBoosting, Hard Negative Mining (just slightly better)
Extend -> Ensemble (work well)

Overview

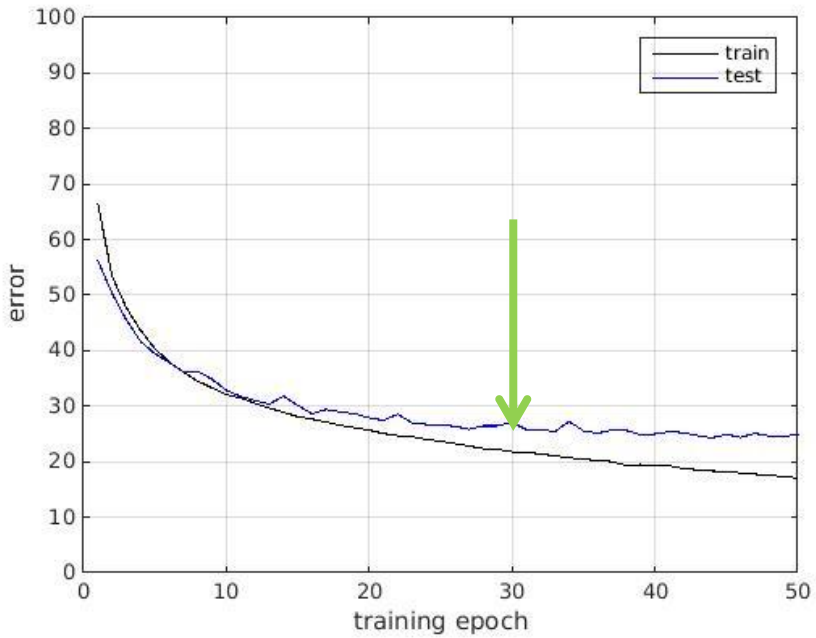
1. Problem Statement
2. Discussion of Several Methods
3. Our Approach
4. Experiments
5. Other Methods tried before

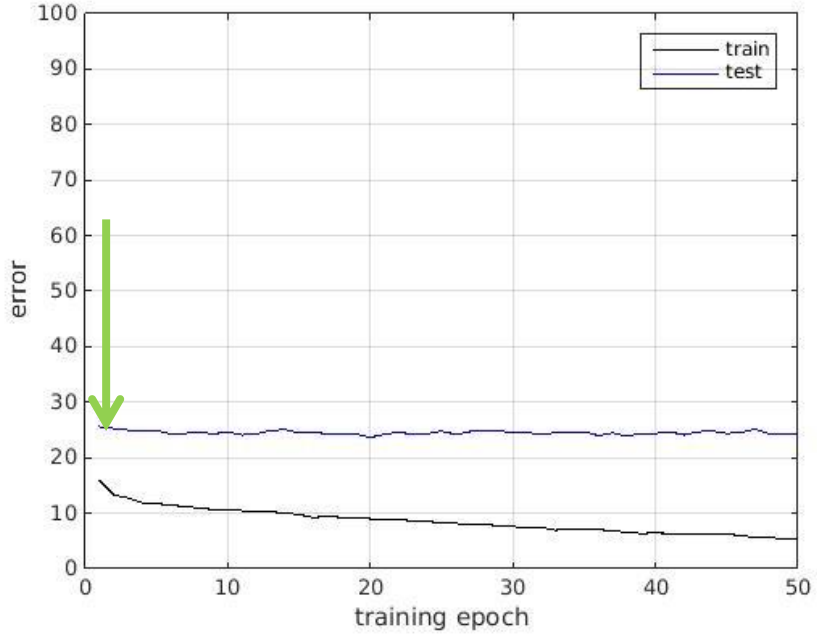
Justify: Observation

- **Accuracy** of new class; **Accuracy** of old class
- **Efficiency** - Training time: measured by the length of training point sequence, equals to "number of images used in each training iteration" * "number of iterations used for training”

- Observation: the straightforward approach -> Flat Expansion
- Conclusion: comparing with training from scratch, Flat Expansion already do very well –

30 epoch (0.2711 Blue line) *VS* 1 epoch (0.2558 Blue line)





Justify: Motivation

- Training weak classifier is already very fast. So we focus on reducing the time for improving weak classifier to strong classifier

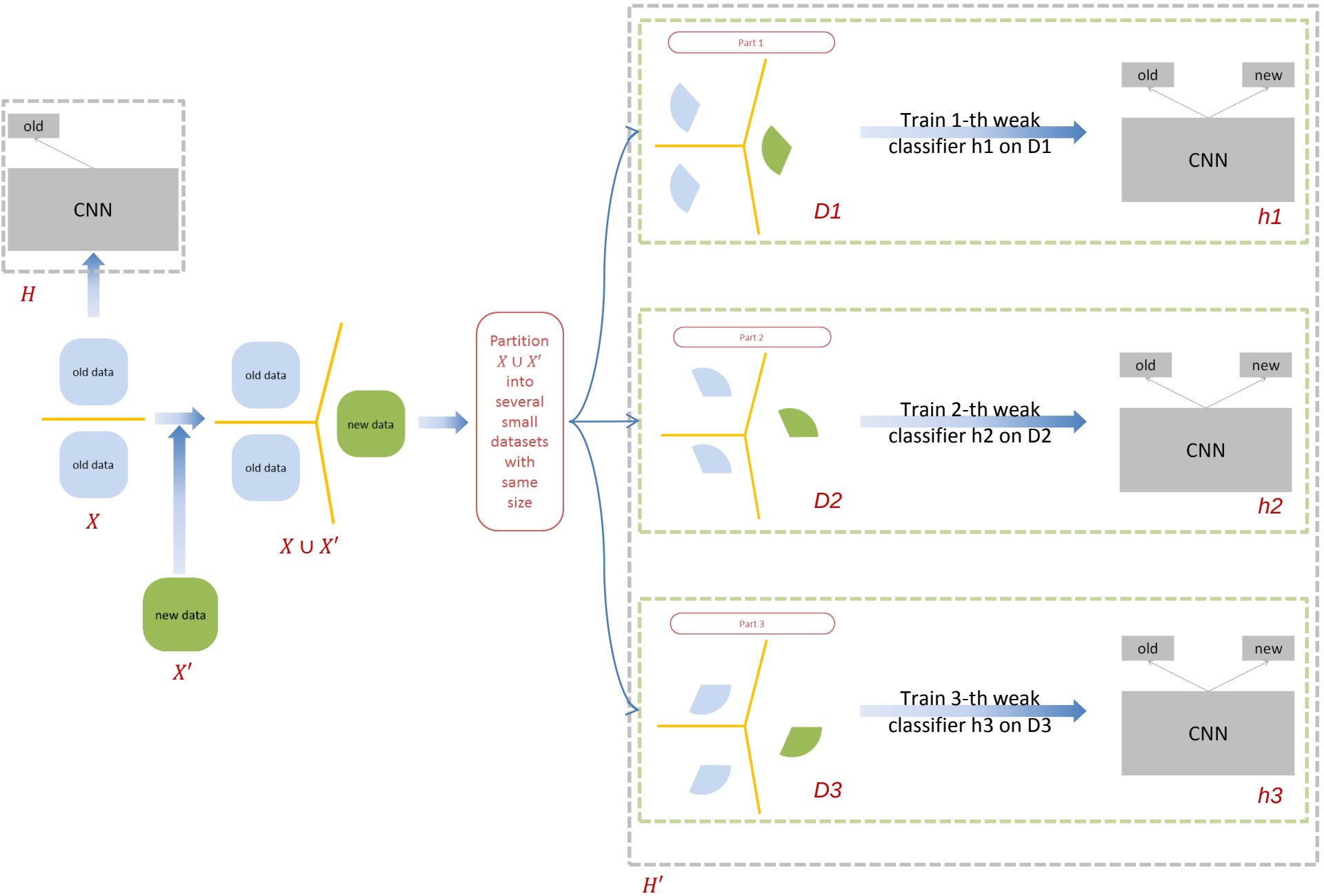
- Basic idea:

1. Subsampling whole training datasets into several parts.
2. These several parts (sub dataset) have smaller number of training data. Use each small training dataset to obtain a weak classifier respectively.
3. Then ensemble these weak CNN classifiers into strong classifier to obtain comparable accuracy, or even higher accuracy.

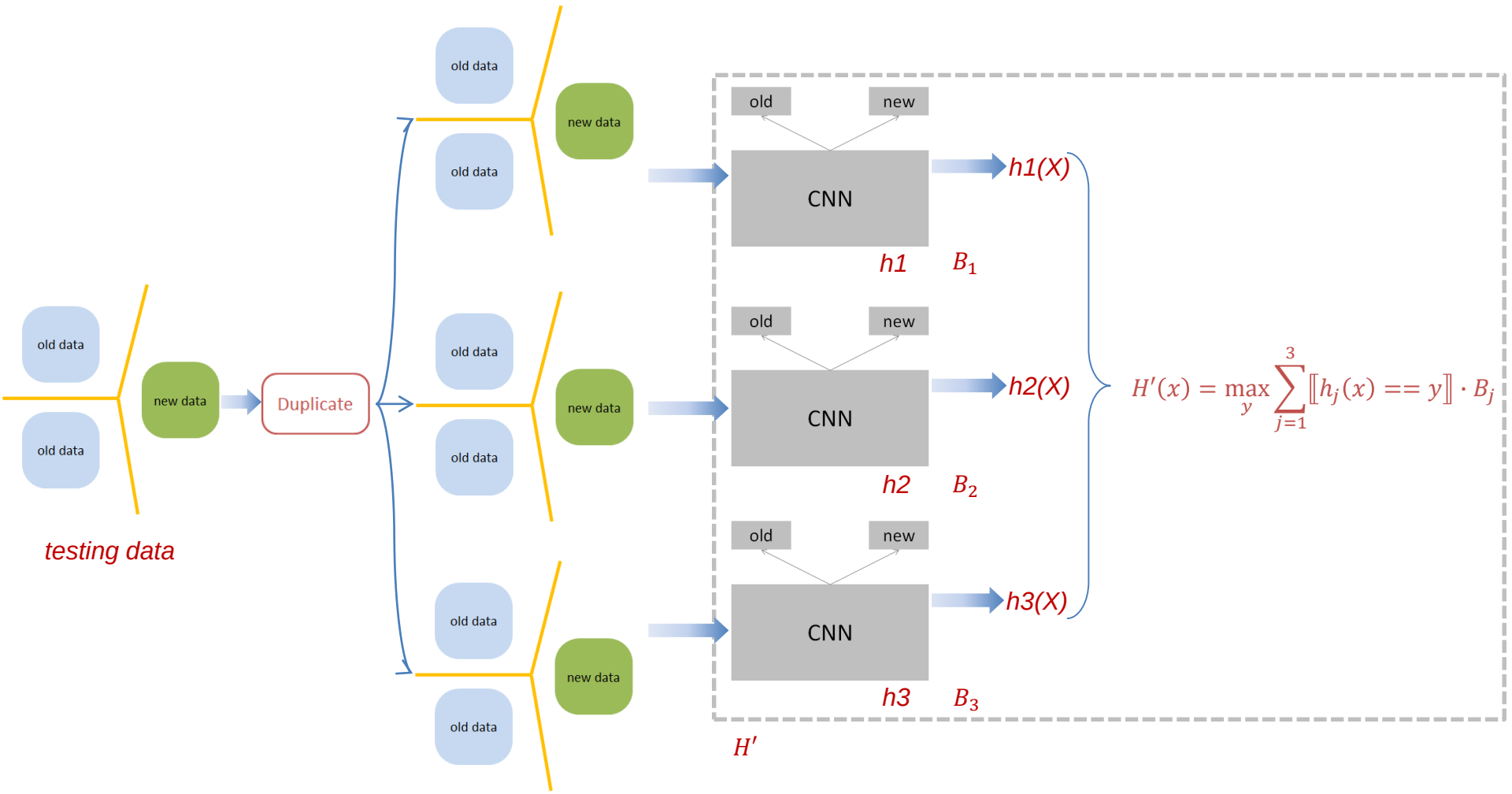
- Why should work:

1. Training weak classifiers can be done quickly
2. Training dataset becomes smaller -> converge faster
3. Training each weak classifier is independent and can be done in parallel

Combine with Ensemble – Training



Combine with Ensemble – Testing (output of H')



Currently, just set B1=B2=B3=1 to do average voting.

Overview

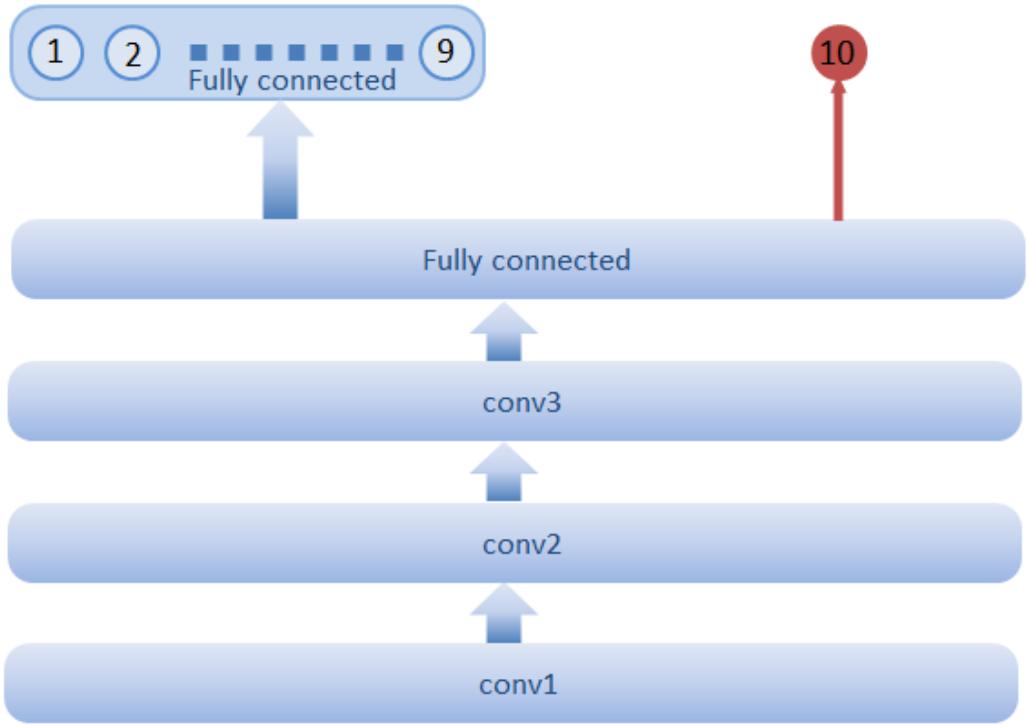
1. Problem Statement
2. Discussion of Several Methods
3. Our Approach
4. Experiments
5. Other Methods tried before

Experiments

- CIFAR-10: 10 classes. Each class has 5000 training images and 1000 testing images.

- Old model: Train on class 1~9, 5000 images per class
- New model: Add the new class –> class 10 with 5000 images. Now train the new model on totally 50000 images.

- Rotate the new class from 1 to 10. At last average accuracy.
- Learning rate: 0.0001; each iteration takes 100 images.



Experiments

- Flat expansion: 5000 images per class (old & new)
- Ensemble-1000: 1000 images per class (old & new), ensemble 5 nets
- Ensemble-2000: 2000 images per class (old & new), ensemble 5 nets

- Testing dataset is same: 1000 images per class

- “Testing error rate” by “Training iterations”. Lower means better accuracy.

training dataset for flat expansion

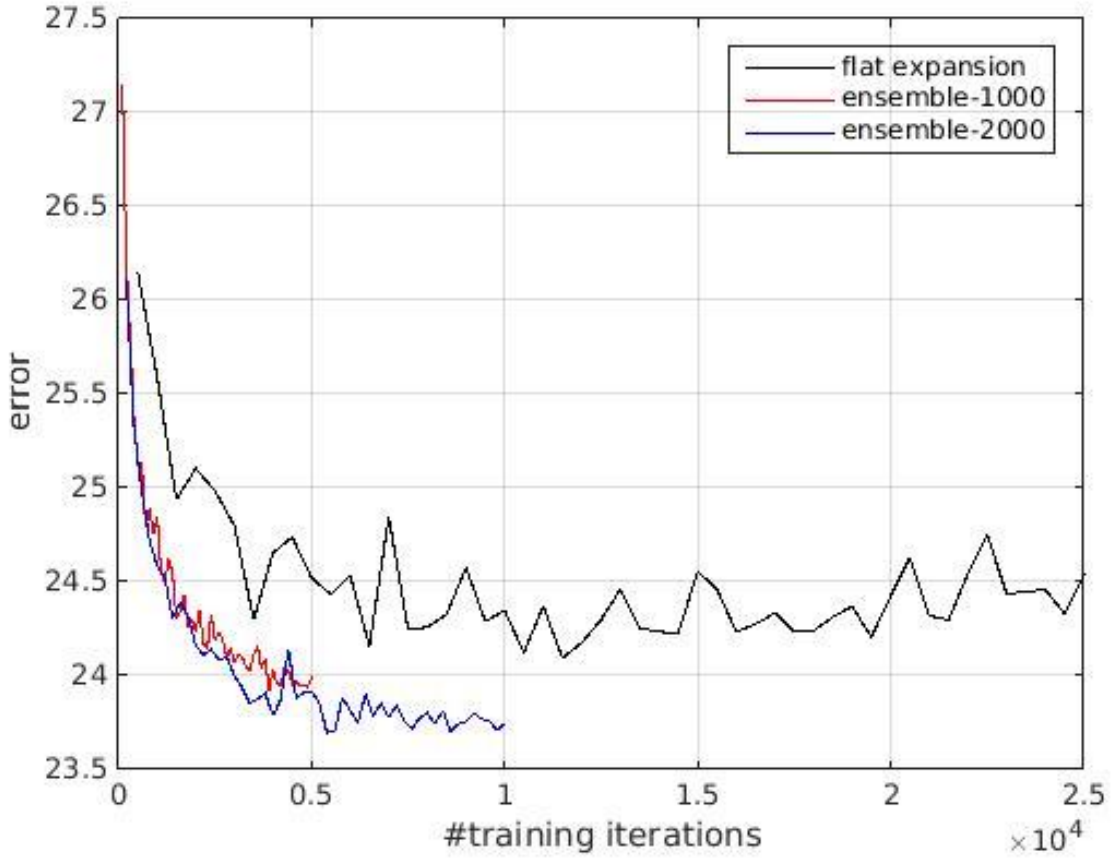
5000

training dataset for ensemble-1000

1000	1000	1000	1000	1000

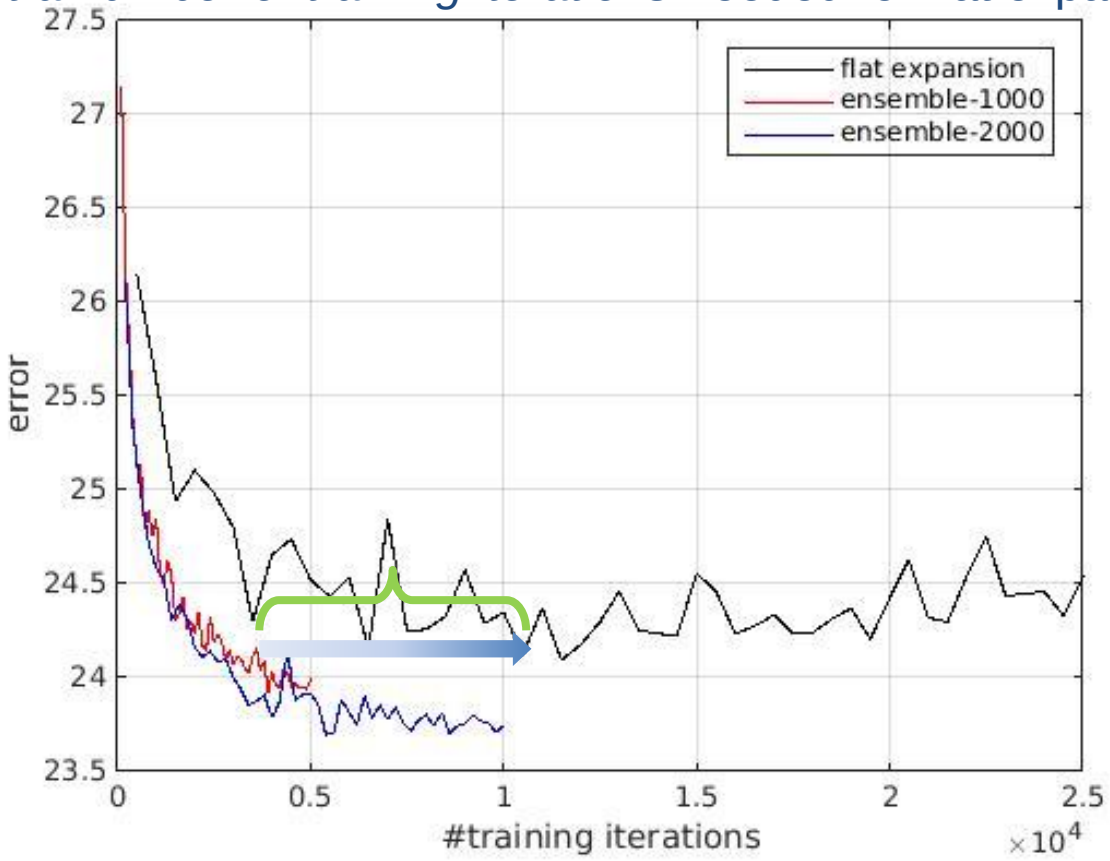
training dataset for ensemble-2000

2000	2000	2000.1
2000.2	2000	2000



Experiments - Other criteria

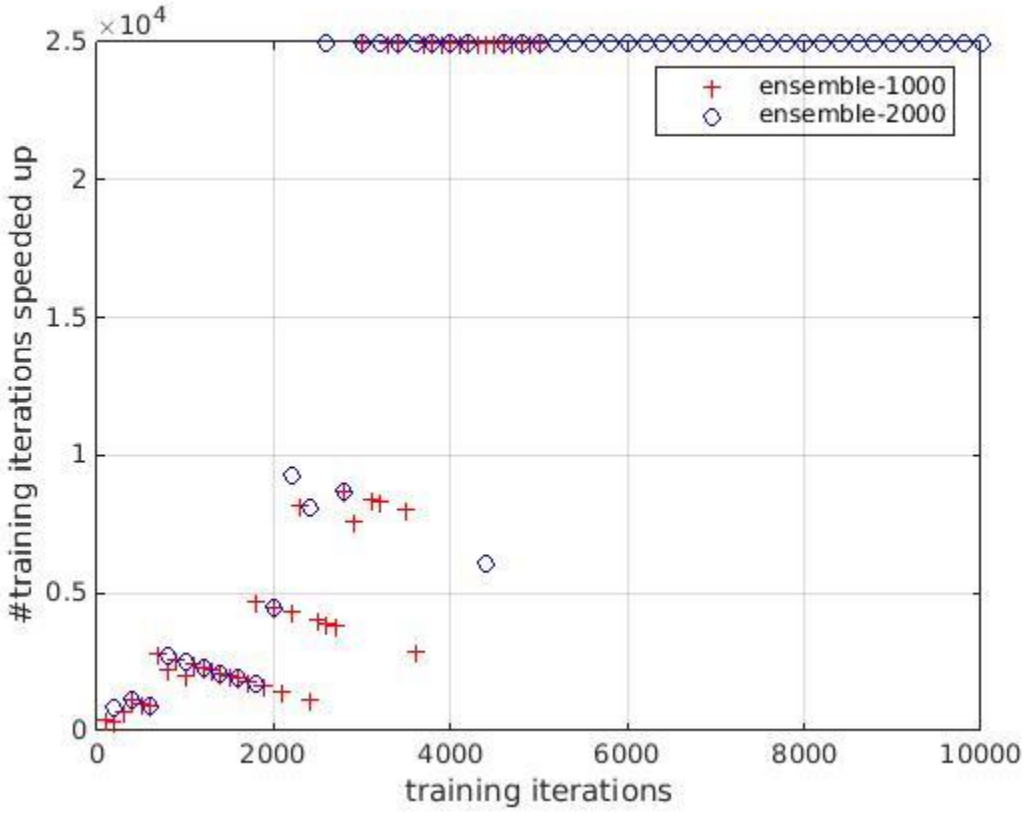
- Comparing with the our model, in order to reach smaller error rate, how many *extra number of training iterations* needed for flat expansion?



- We compute this *extra number of training iterations* for each model of ensemble-1000 and ensemble-2000. And plot in next page.

Experiments

- Vertical axis: Comparing with the our model, in order to reach smaller error rate, how many extra number of training iterations needed for flat expansion?
- Horizontal axis: How many training iterations used for training our model
- Positive value indicates our approach runs faster. Higher is better.

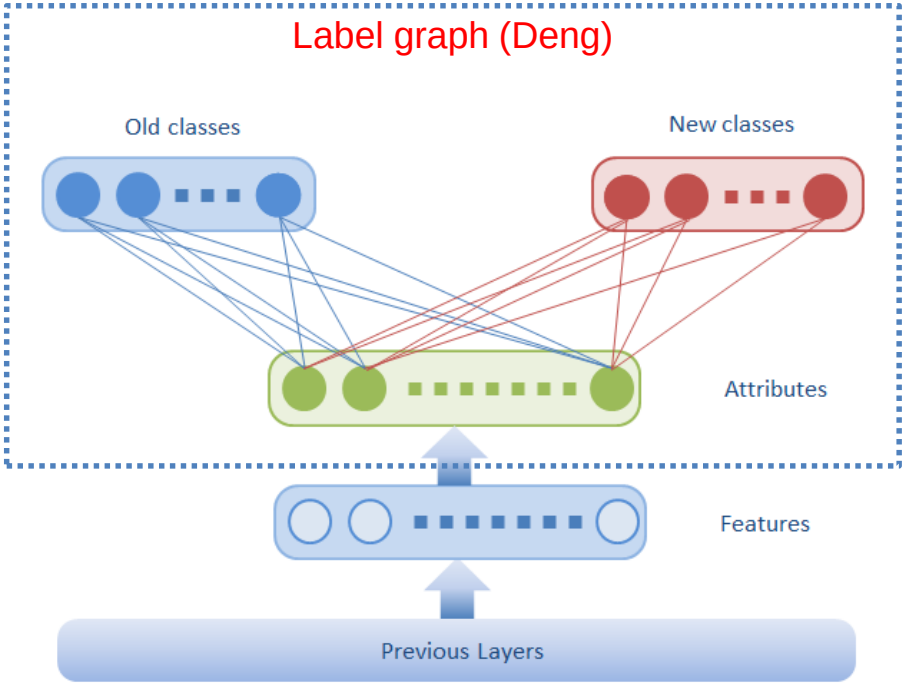


Thank you!

Overview

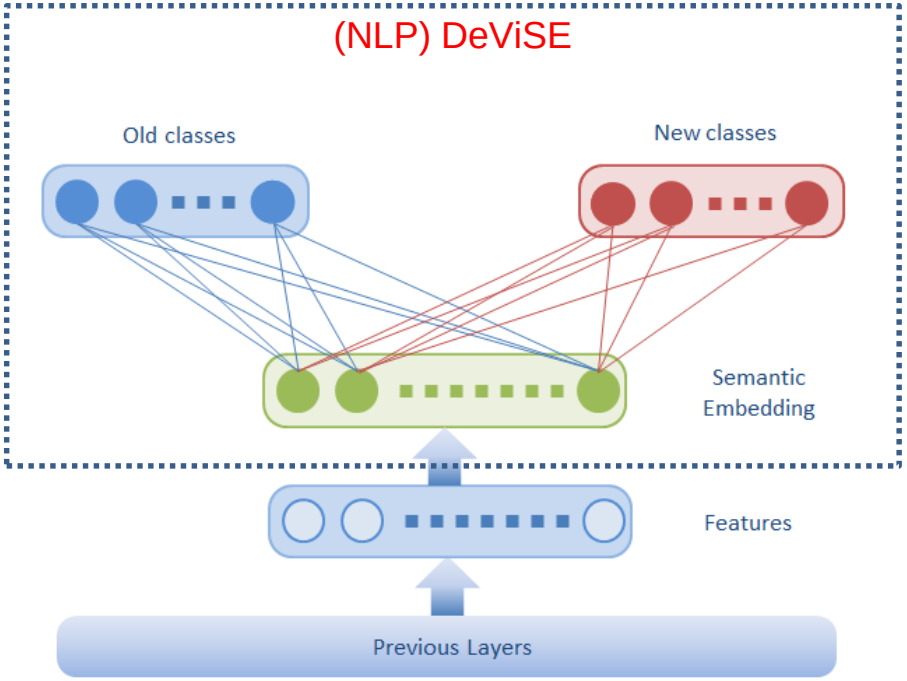
1. Problem Statement
2. Discussion of Several Methods
3. Our Approach
4. Experiments
5. Other Methods tried before (If need discuss)

B. Attribute Learning



1. #Attributes fixed
2. Organize attributes nodes and classes nodes as a label graph
3. Incremental learning problem becomes adding new nodes in the label graph
4. May improve accuracy but no help or even hurt training speed
5. Incremental problem still remains in the graph

C. Semantic Embedding

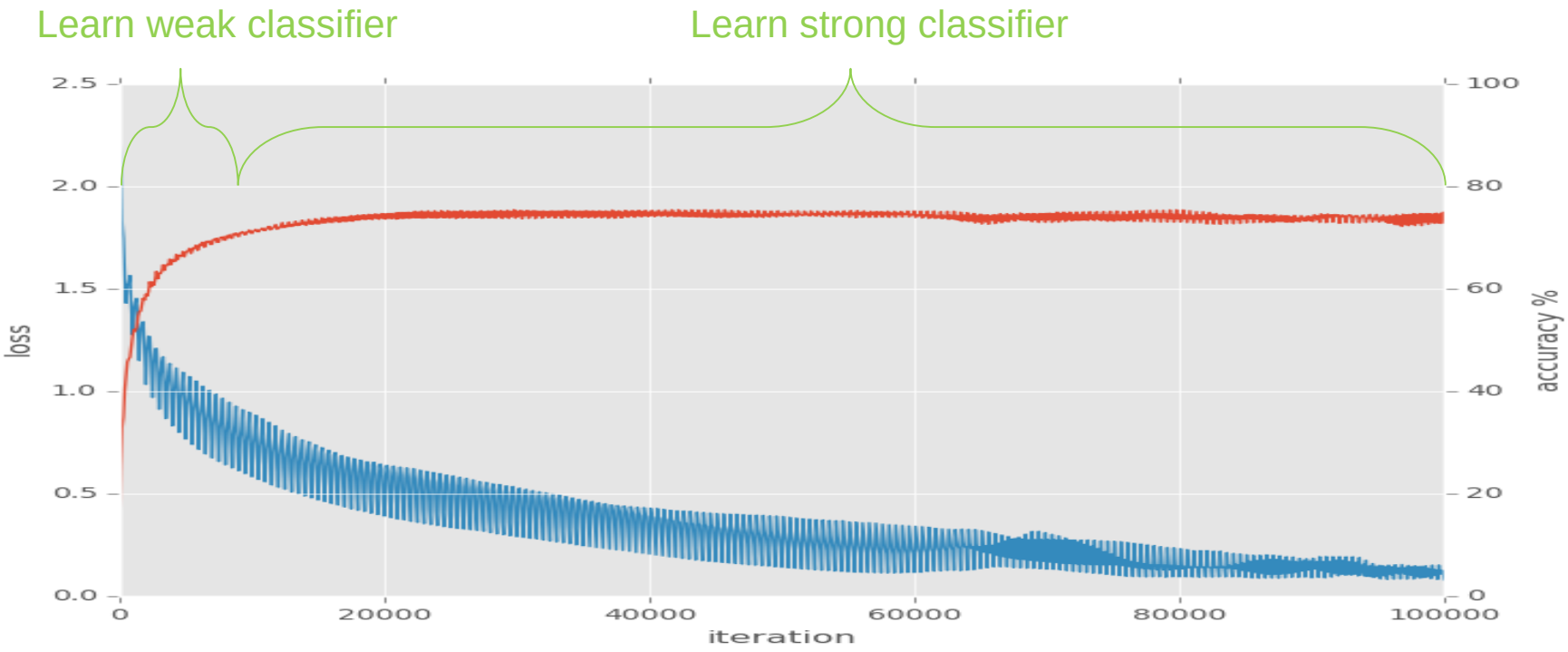


1. Semantic Embedding means using NLP tools to map class label as a fixed size semantic vector
2. Similarly as attribute learning based method, replace attributes with semantic embedding
3. May improve accuracy but no help or even hurt training speed
4. Incremental problem still remains in the graph

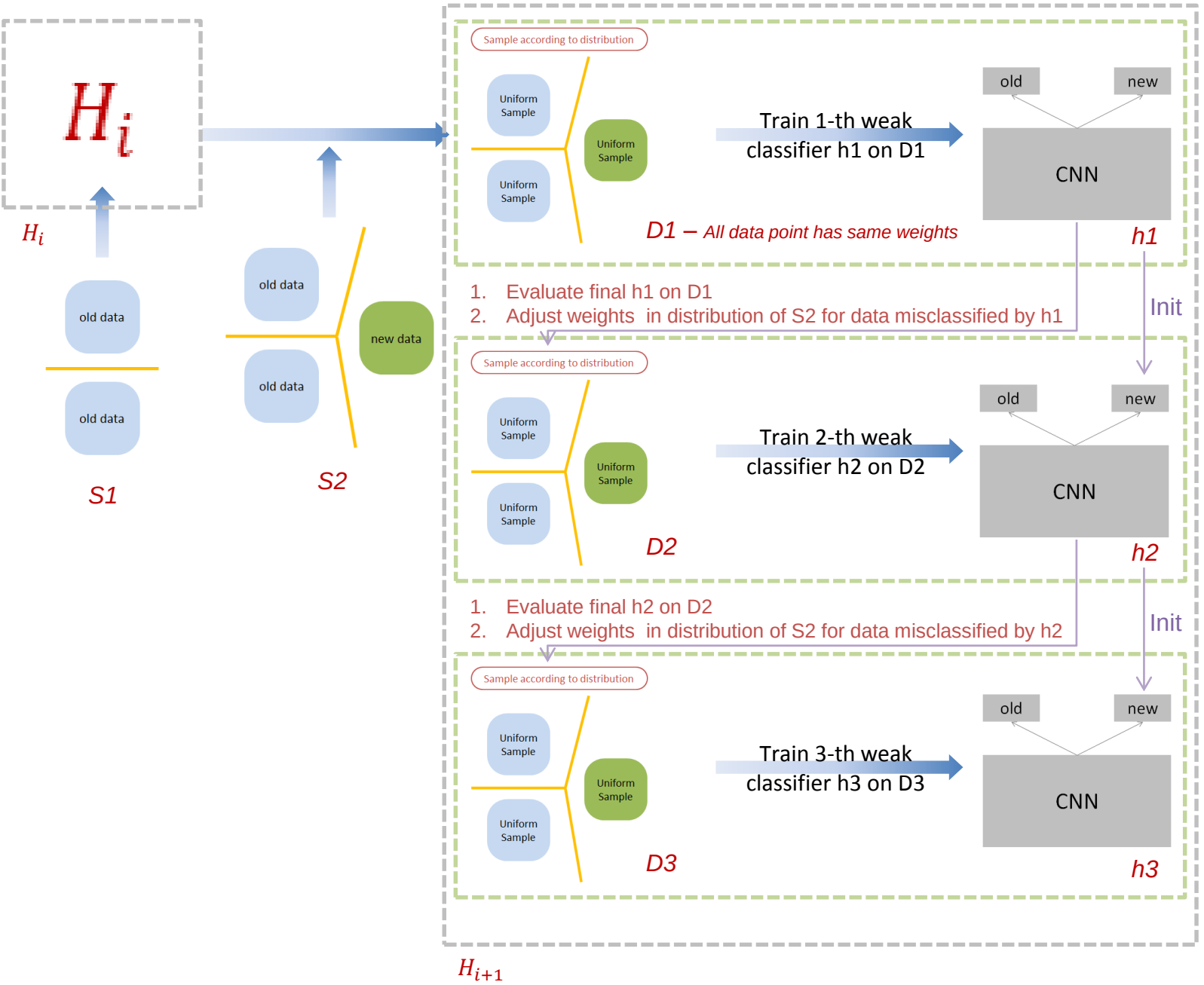
Compare with AdaBoosting

- How to keep accuracy while further speed up flat expansion in the incremental learning procedure?

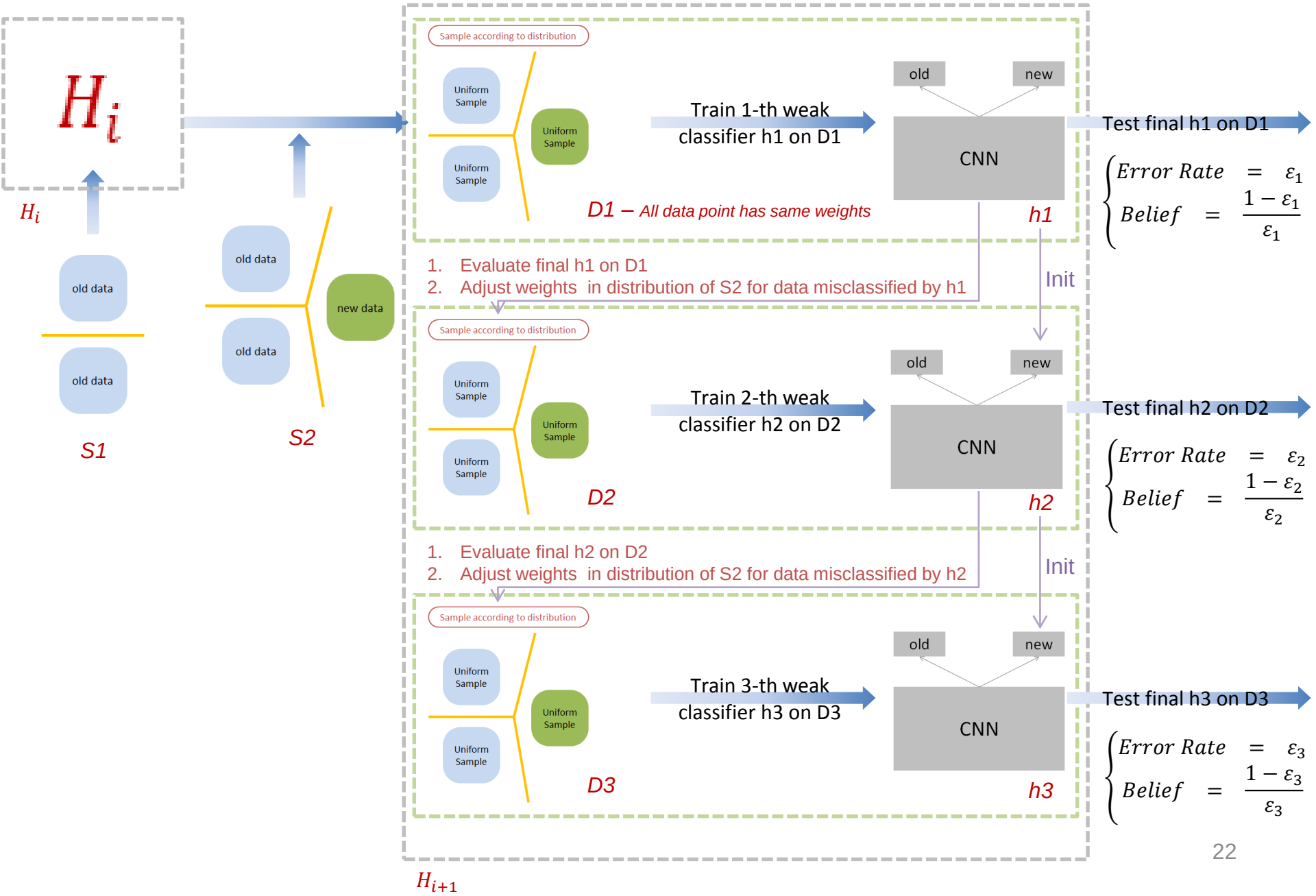
- For CNNs, learning weak classifier is quick, while most of time takes on learning strong classifier
- Boosting: using ensemble of weak classifiers as a strong classifier
- AdaBoosting involves adjusting data distribution, which is the key of our adapted curriculum learning approach



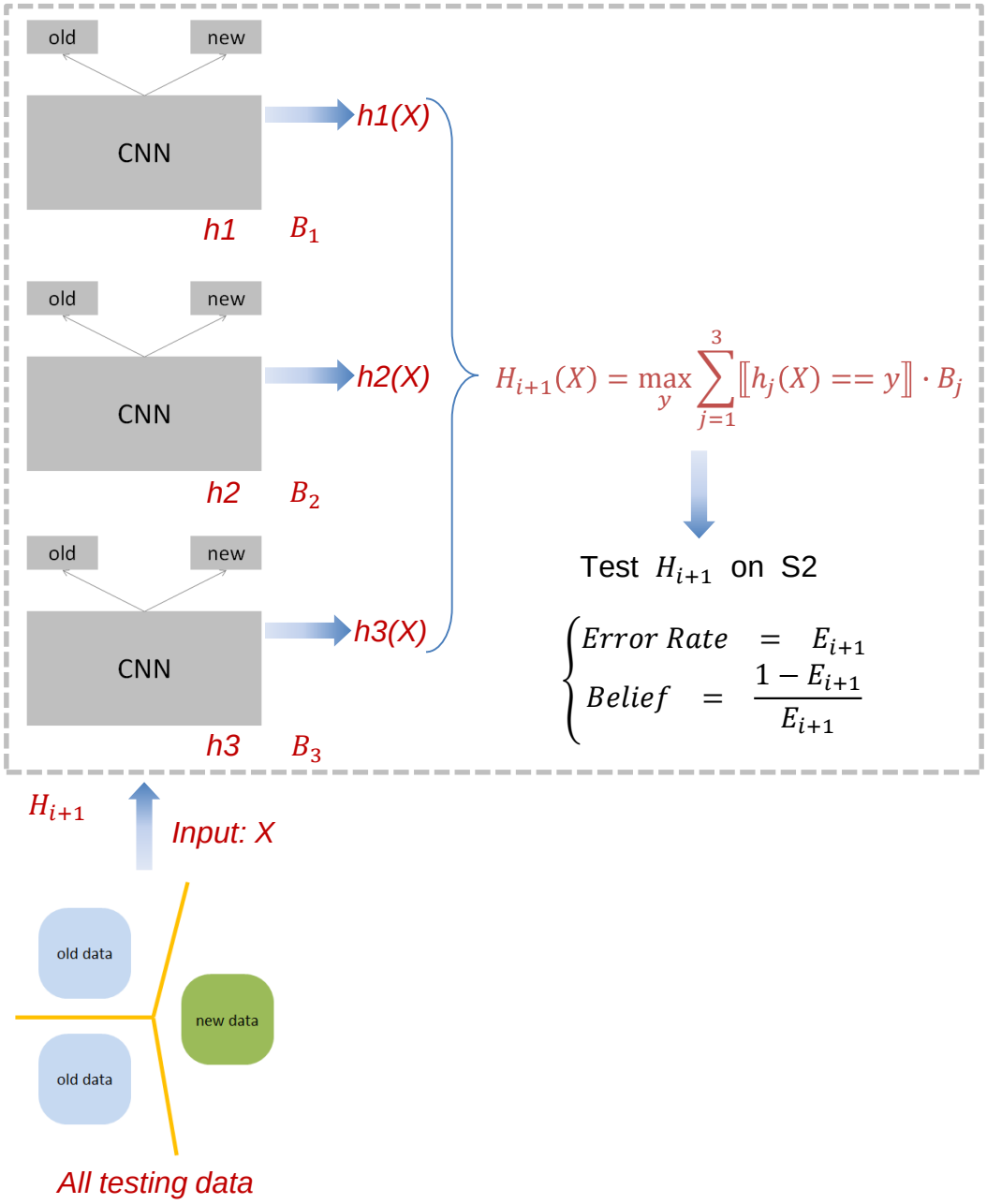
Our Approach AdaBoosting – 1. Training



Our Approach AdaBoosting – 2. Validation

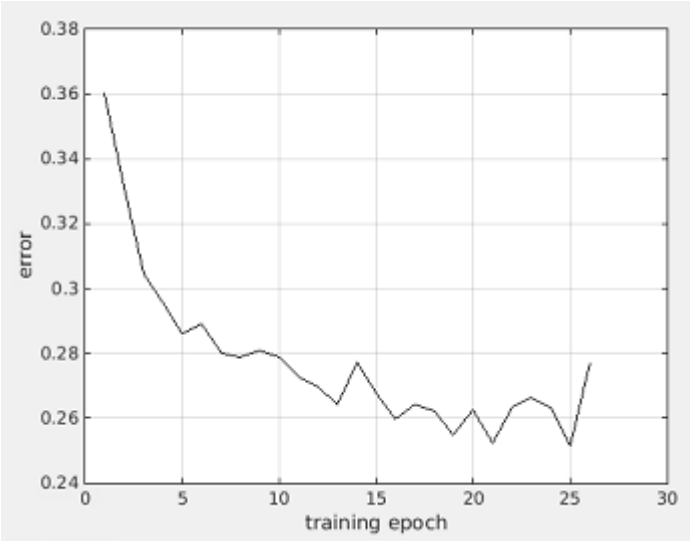


AdaBoosting – Testing (output of H_{i+1})



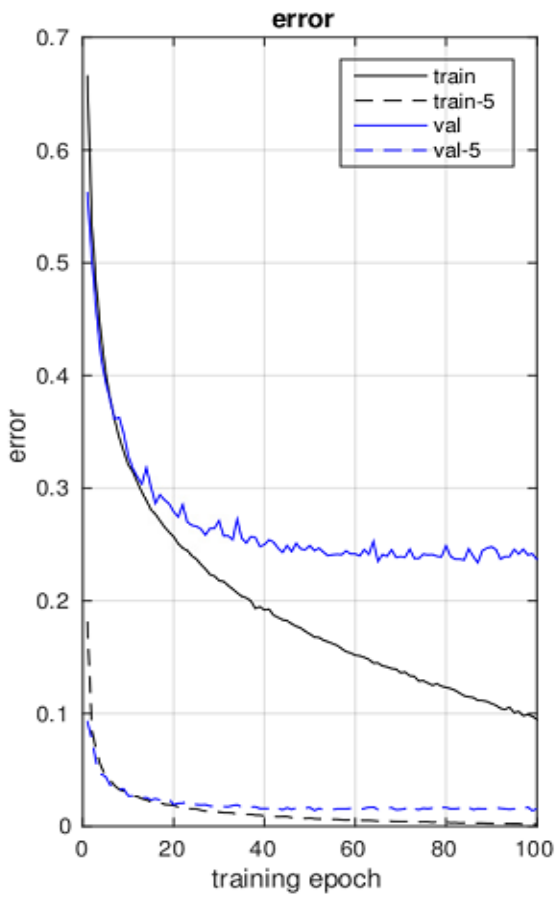
Experiments – Ours

- Old model is trained by 25000 images for classes 1-5 and the incremental learning goal is to train CNN when 25000 new images for classes 6-10 are added into the training dataset
- Training dataset during incremental training: 5000 images for each class. Totally 50, 000 images.
- Init by model already trained for classes 1~5
- Net0: 21 epochs; Net1: 3 epochs; Net2: 1 epoch; Net3: 1 epoch
- Final testing error rate: 24.81%

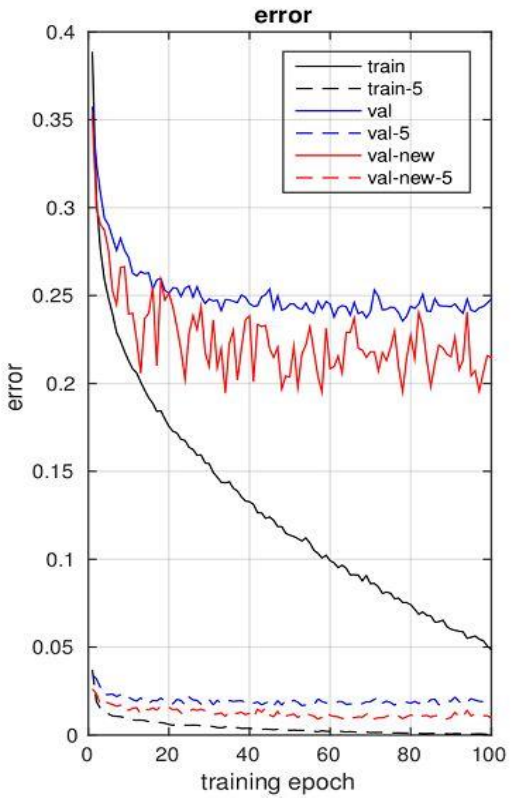


Experiments - Baselines

- Baseline1 (Training from scratch):
- Training dataset: 5000 images for each class. Totally 50, 000 images.
- Learning rate = 0.0001



- Baseline2 (Flat expansion - Init by model already trained for class 1~5):
- Training dataset: 5000 images for each class. Totally 50, 000 images.
- Learning rate = 0.0001



Experiments - Comparisons

- Set learning rate as 0.0001
- Testing set: 10000 images. Class 1~10. 1000 images for each class.
- Evaluation: # training epochs needed to get testing error rate < 0.25
- Training from scratch: 39 epochs
- Flat expansion: 32 epochs
- Ours: 26 epochs (21-24-25-26)

Analysis & Comparison

- Why should work: It is efficient to learn weak classifiers while it takes too long to learn strong classifier. Boosting guarantees final accuracy. Also this approach can learn new data for both old and new classes.

- Differences with AdaBoosting: In AdaBoosting, weak classifiers are independent and learn from scratch; while in our each incremental procedure, weak classifiers are related and have dependency.

- Differences with Hard Negative Mining: Hard Negative Mining only uses the final classifier and updates distribution after each epoch; our approach combines all weak classifiers and updates distribution after weak classifier is learned.