Robust Classifiers

Gurparkash Singh
160050112 (CSE)

Prof. Amit Sethi
Prof. Suyash Awate

B.Tech. Project 2 (CS496)

Introduction

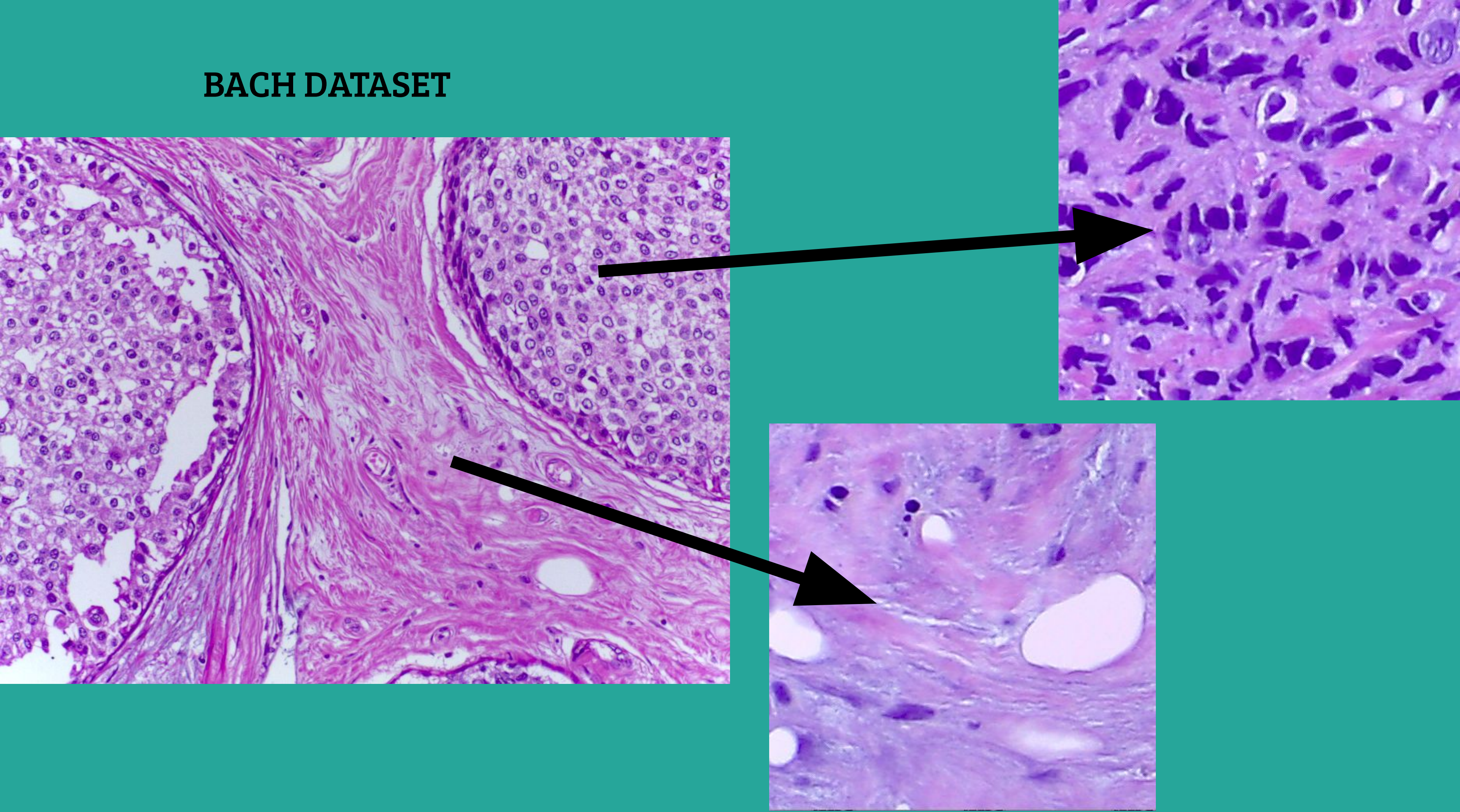
- Aim of AI : Automation of tasks done by humans
  - So far, done manually by medical experts
  - Can use Machine Learning to automate these tasks

- Deep Learning Algorithms require large and reliable datasets

- A problem with Histopathology Datasets : Noisy Labels
  - Large images => Cropped Datasets introduce noise in labels

- Building generalized models for medical tasks is a difficult problem!

BACH DATASET



Problem Description

- Variance in Visual Data handled by Color Normalization & Augmentation
- No fixed method to deal with Noisy Labels

- Neural Networks are extremely good at function approximation
- Also fit to the noise in the Training Data

- Goal is to train robust classifiers
- Performance should not be affected by noise in Training Data

- Inspiration from Curriculum Learning and Adversarial Autoencoders

Lessons from Curriculum Learning

- Neural Networks first fit to the non-noisy or “easy” data-points
- Afterwards, they start fitting to the noisy samples or “difficult” data-points

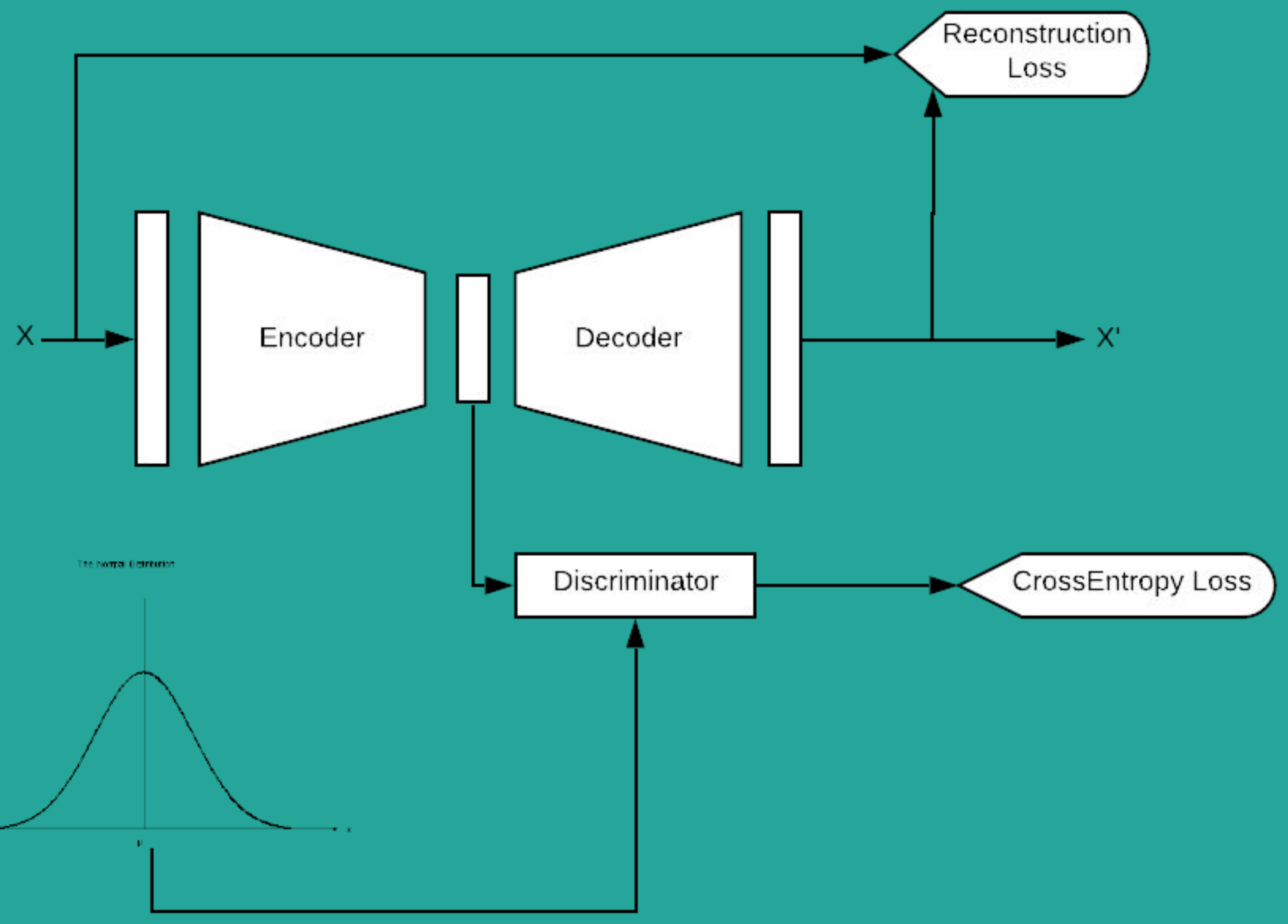
- Motivation for learning in an incremental way
- Start with updating weights for the easiest points
- Slowly keep adding more and more points to the set

- How to separate easy and difficult points?
- Loss Values!
- Easy samples will have lower loss value.
- Use a threshold on noise values and only train on points below that
- As the model trains, more and more points will come below the threshold
- Hopefully, at the end only the noisy points will be left out

Adversarial Autoencoders

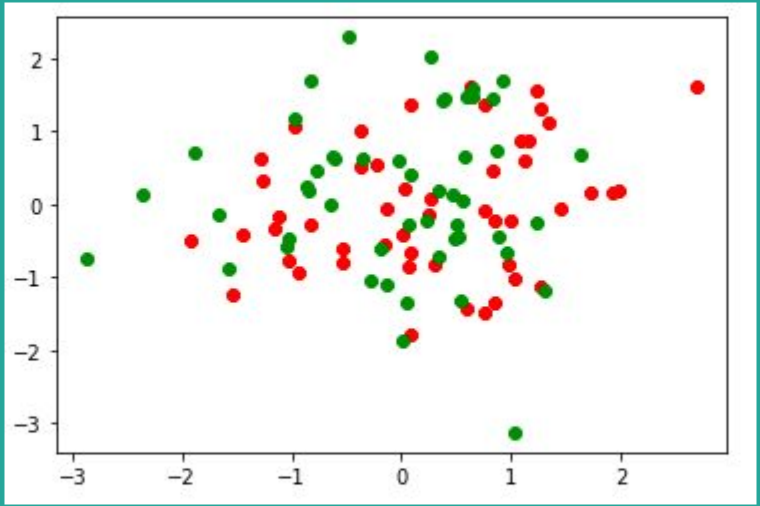
- Unsupervised Learning Algorithms
- Used to fit features to a prior

- Three components : Encoder, Decoder, Discriminator
    - Encoder outputs features
    - Decoder takes features and tries to reconstruct the input image
    - Discriminator tries to distinguish between prior samples and features



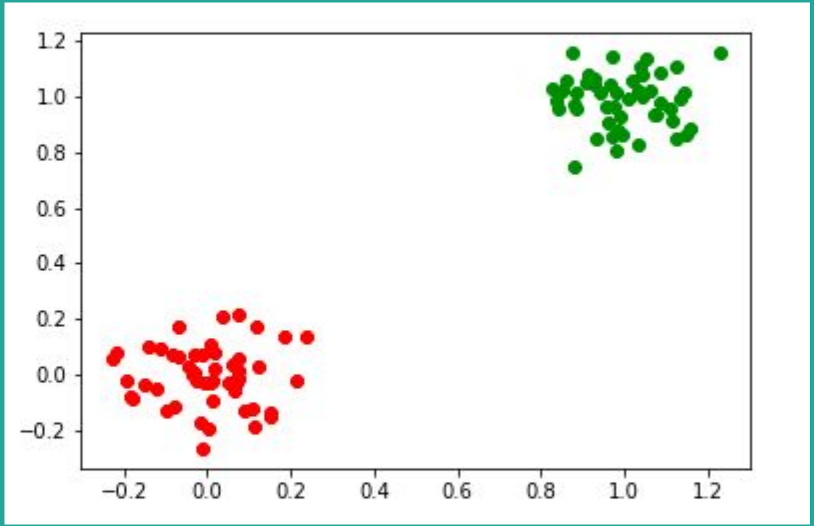
Reconstruction Loss
X
Encoder
Decoder
X'
Discriminator
CrossEntropy Loss

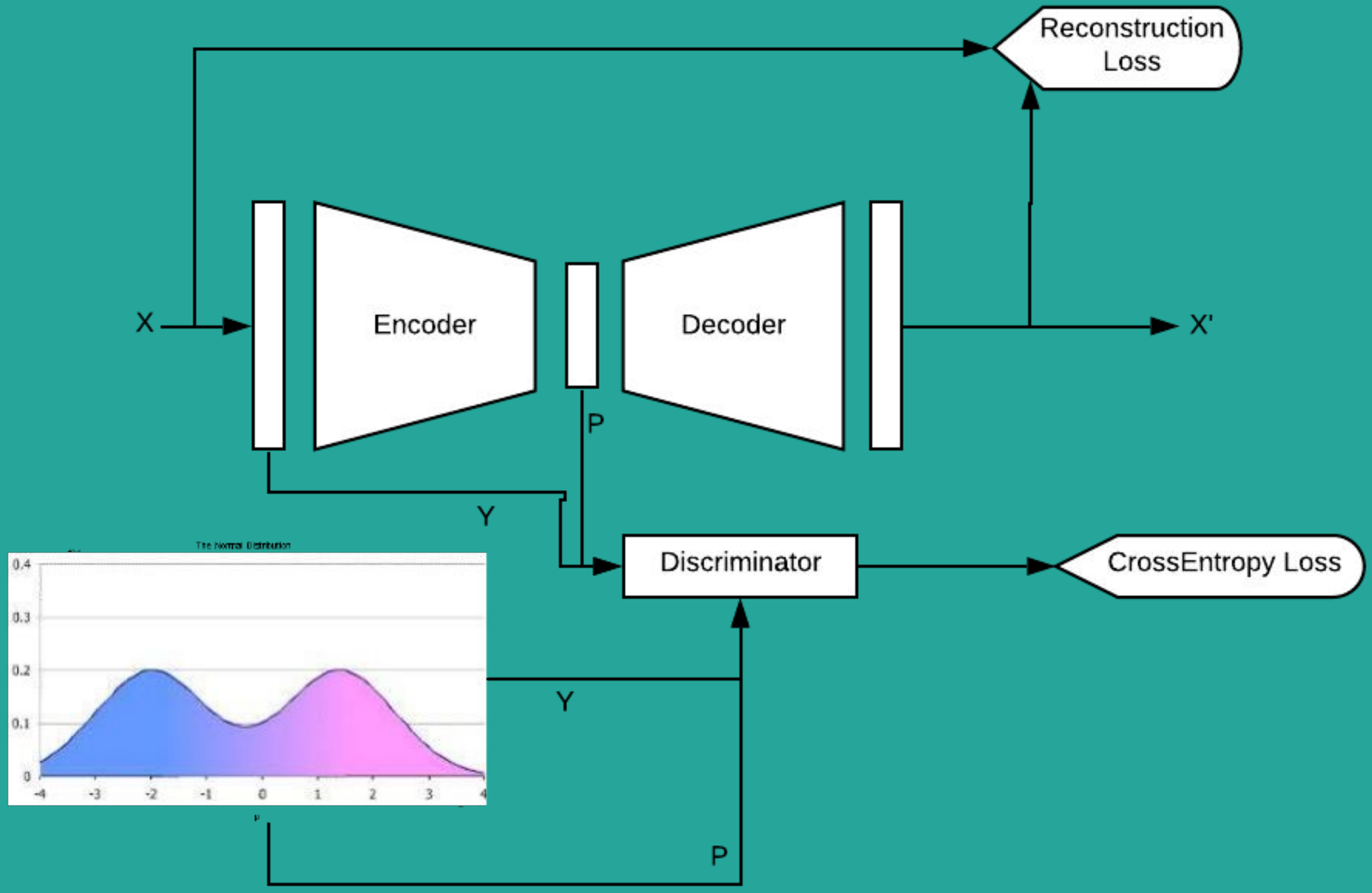
- Encoder trained in an Adversarial way
- Tries to reduce the accuracy of the discriminator
- If successful, features would be indistinguishable from the prior



Supervised AAE

- Can fit the features to a multi-modal prior using the labels
- Separate modes of distribution for separate classes
- We can hope to train a classifier on top of these features



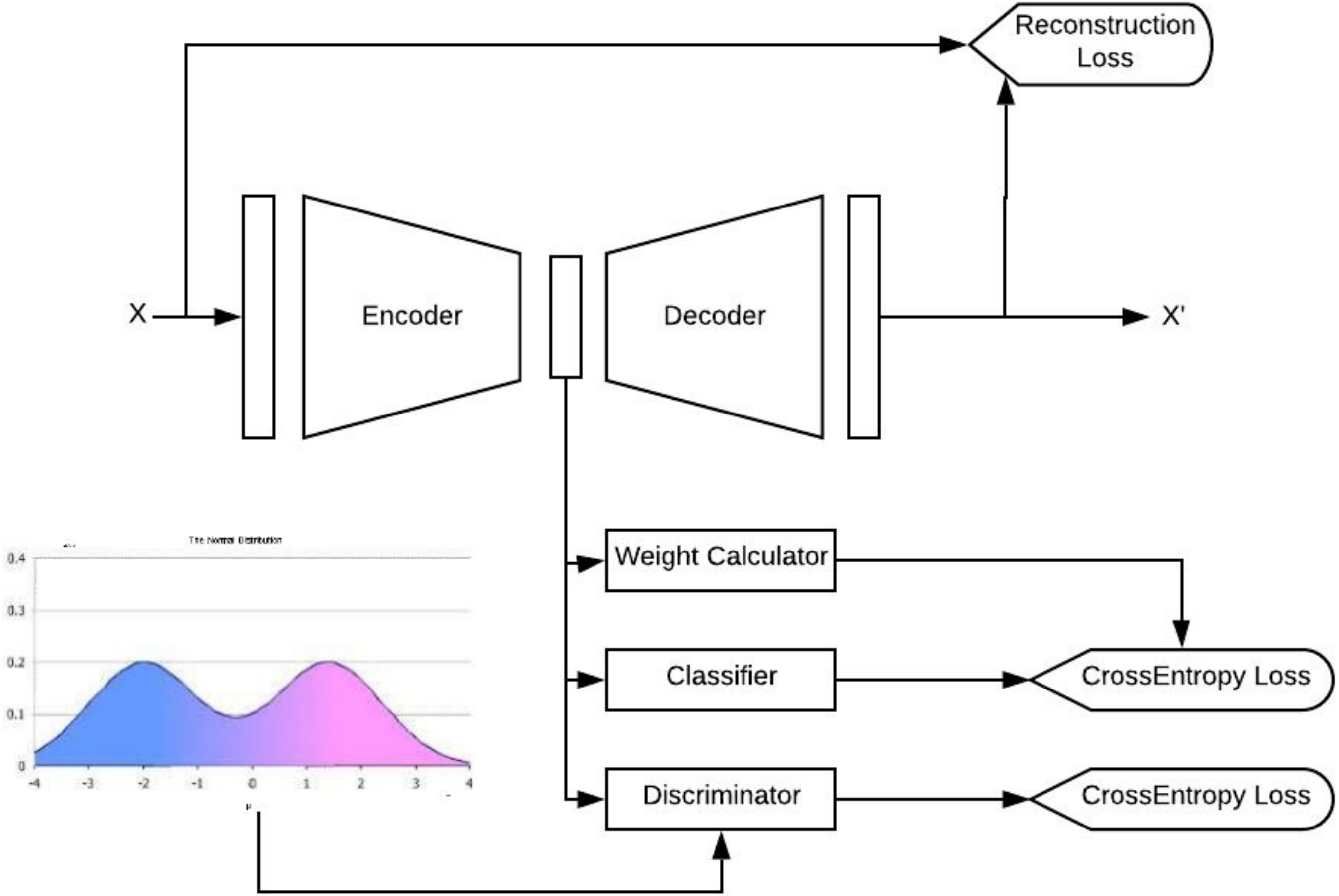

Reconstruction Loss
X
Encoder
Decoder
X'
P
Y
The Normal Distribution
0.4
0.3
0.2
0.1
0
-4
-3
-2
-1
0
1
2
3
4
Discriminator
CrossEntropy Loss
Y
P

Hypothesis

- The non-noisy data points will fit our prior better
- Can use the value of PDF to separate the noisy and non-noisy samples
- Higher value of PDF => more likely to be non-noisy
- Can we use features from AAE to train a classifier?

Problems with AAE Features

- Static! No incremental training as discussed before
- A classifier will eventually overfit to the features
- Can not train classifier separately AFTER generating features
- Solution : Add a classifier to our network and train it together

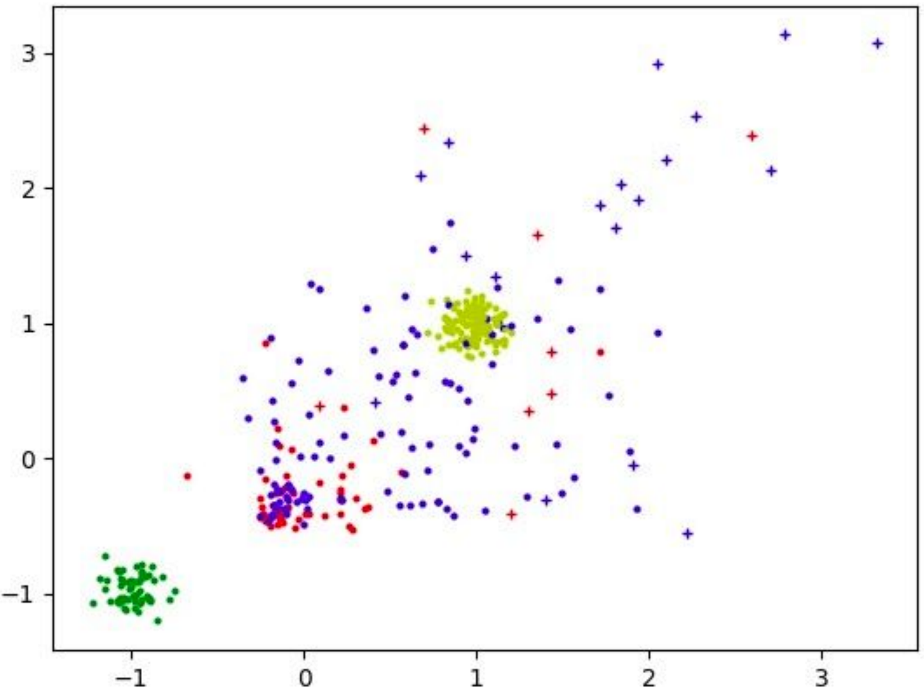

Reconstruction Loss
X
Encoder
Decoder
X'
Weight Calculator
Classifier
Discriminator
CrossEntropy Loss
CrossEntropy Loss

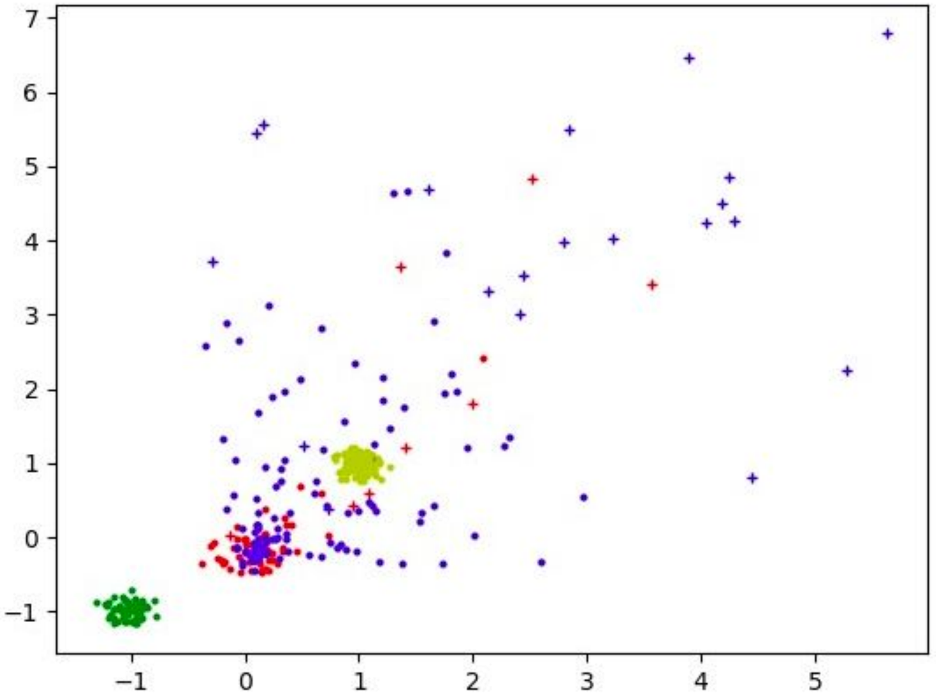
Weight Calculation

- Use the value of the PDF
    - Value of the PDF for the features
- Normalize the weights for stable training
    - Normalize the values for each batch
    - They will sum to 1 => more stable training
    - Regularization Properties
- Use binary weights
    - We can use a threshold value to make the weights binary
    - Can be used in addition to normalization

Experiments & Results

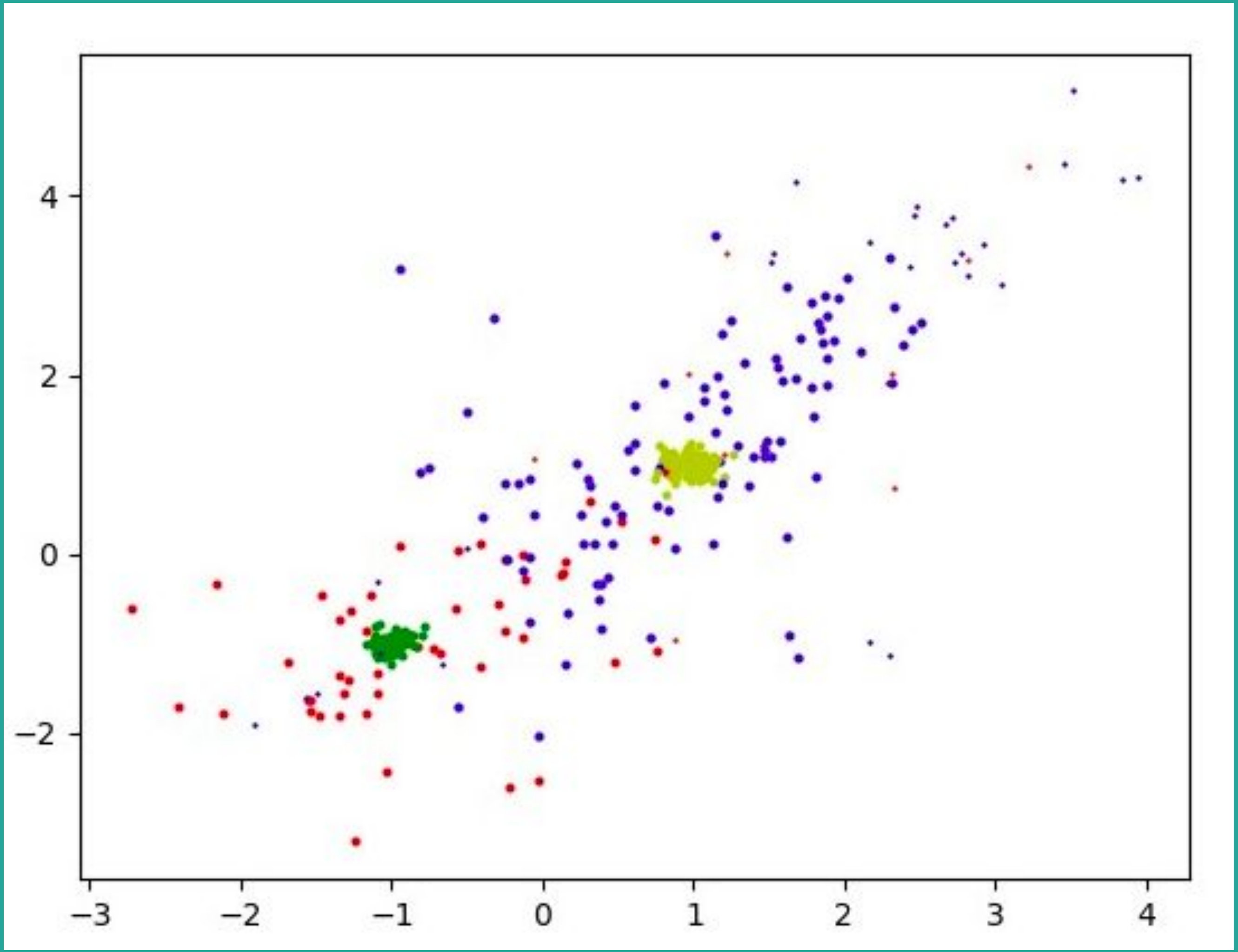
- As expected training classifier on AAE in not sufficient
  - Features not enough properly separated
  - A model eventually overfits to static features
  - Even without noisy dataset, accuracy ~75%
  - Noisy labels makes the performance even worse

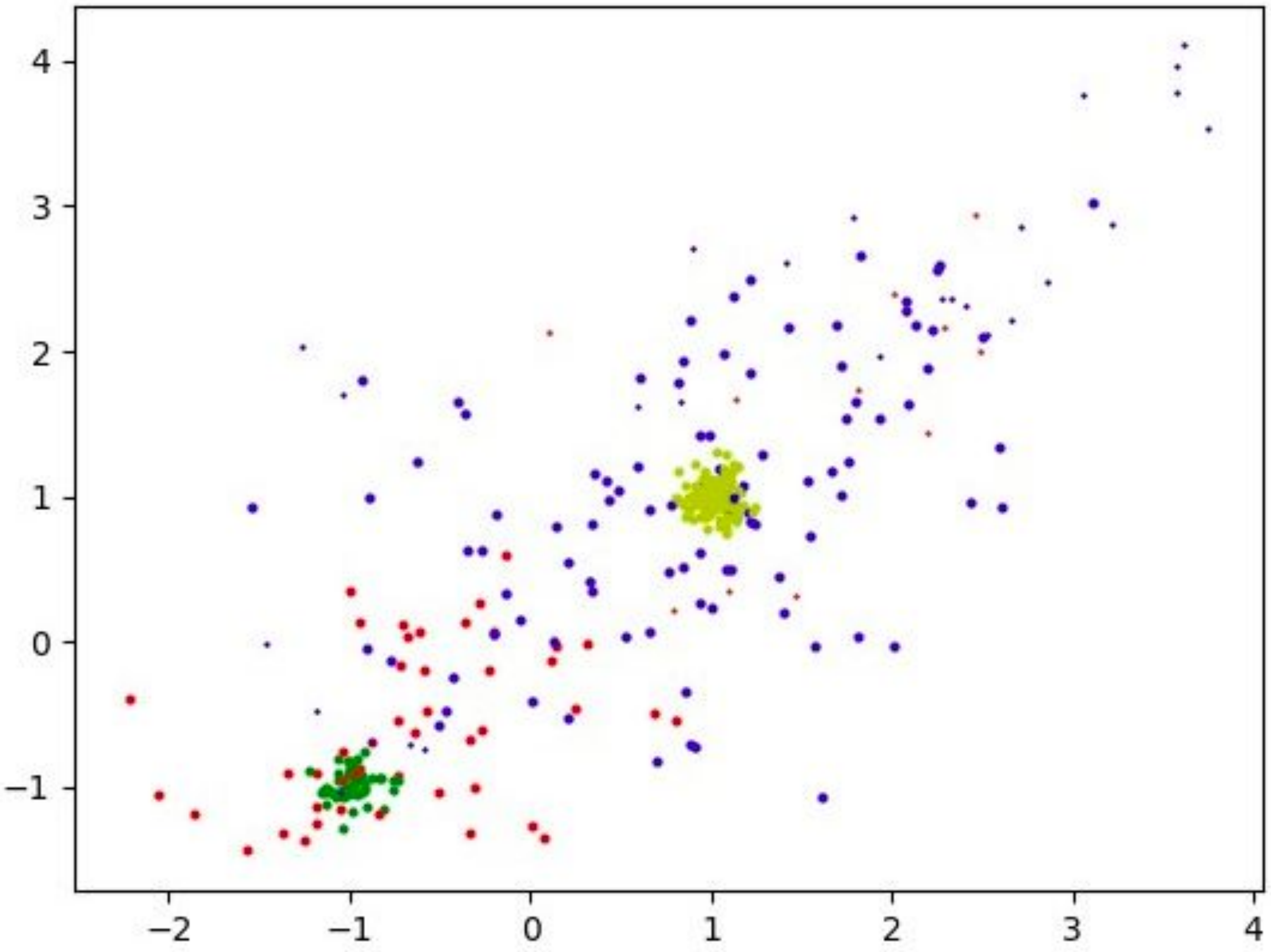

3
2
1
0
−1
−1
0
1
2
3


7
6
5
4
3
2
1
0
−1
−1
0
1
2
3
4
5

Performance of Our Model

- Co training a classifier with an AAE helps separate the features better.
- Accuracy on eval set remains constant even with increasing noise.
- Results tabulated and compared with conv-nets later.

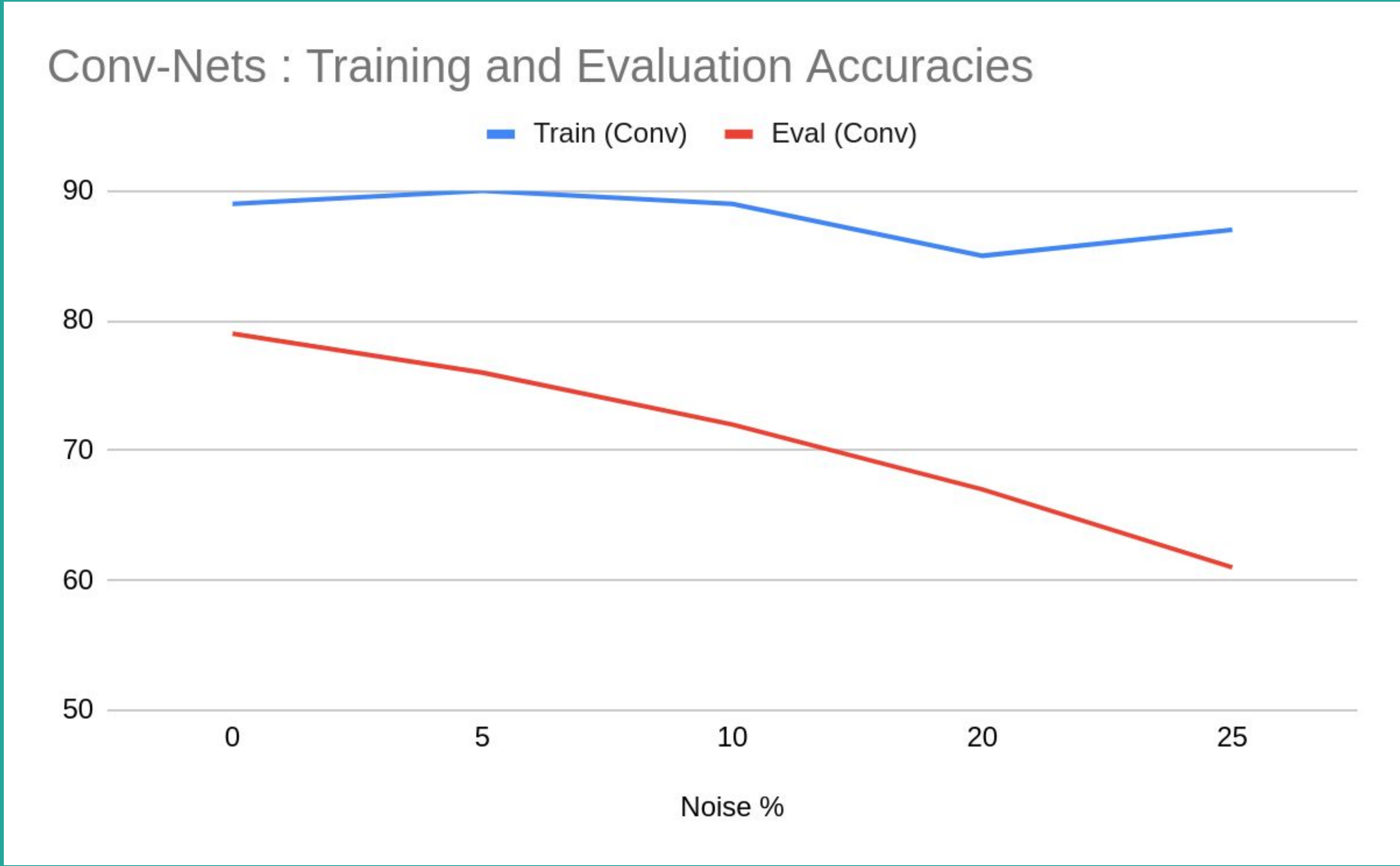

4
2
0
−2
−3
−2
−1
0
1
2
3
4

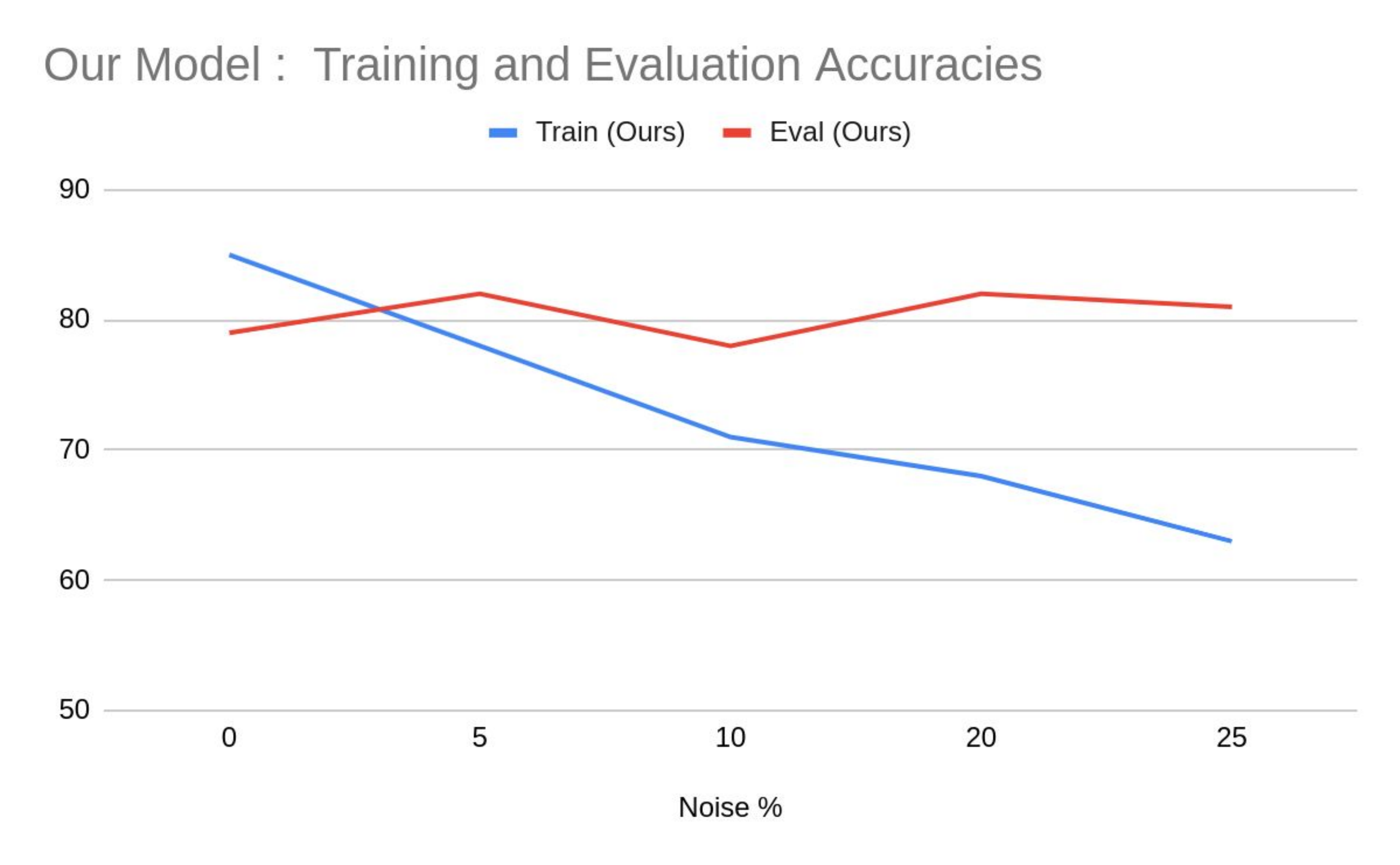


4
3
2
1
0
−1
−2
−1
0
1
2
3
4

Results & Comparison with Conv-Nets

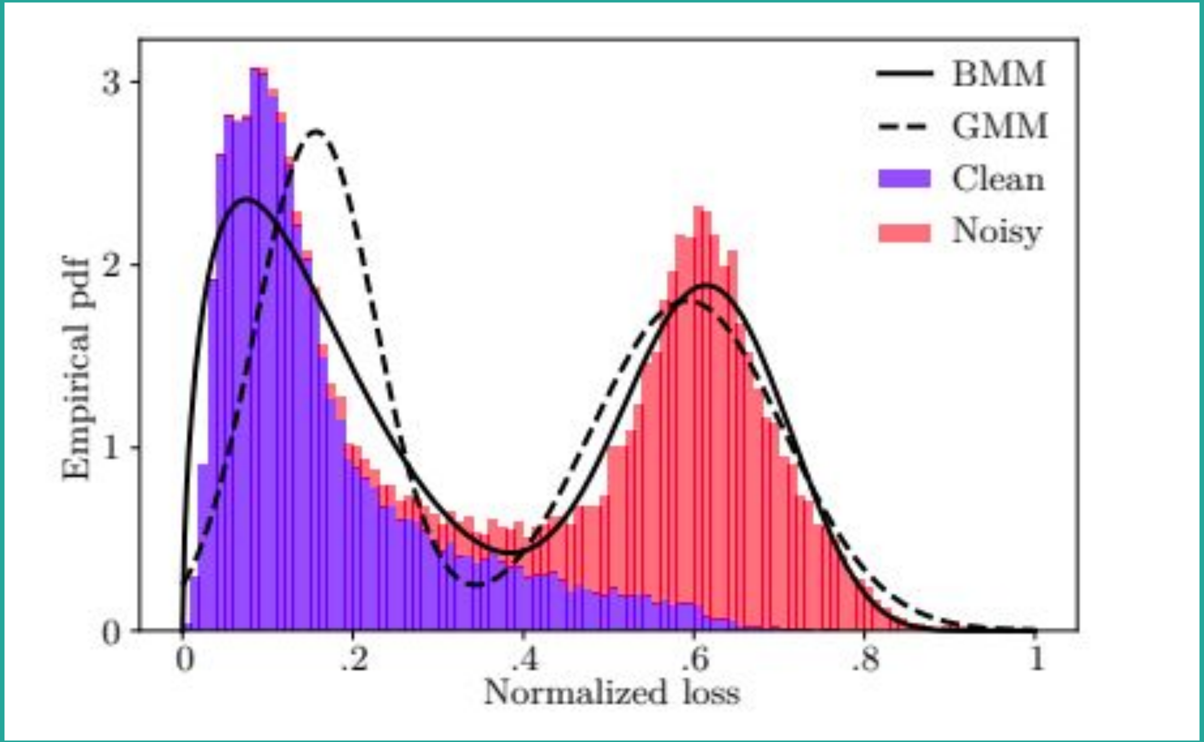
Noise %	Train (Conv)	Eval (Conv)	Train (Ours)	Eval (Ours)
00	89	79	85	79
05	90	76	78	82
10	89	72	71	78
20	85	67	68	82
25	87	61	63	81


Conv-Nets : Training and Evaluation Accuracies
Train (Conv)
Eval (Conv)
90
80
70
60
50
0
5
10
20
25
Noise %


Our Model : Training and Evaluation Accuracies
Train (Ours)
Eval (Ours)
90
80
70
60
50
0
5
10
20
25
Noise %

Can we still do better?

- Our approach either gives low weight or no weight to noisy samples.
- We can use the noisy samples to do even better.
- This idea is explored in “Unsupervised Label Noise Modeling and Loss Correction” by Eric Arazo et. al.
- The logic is that a model will first fit to the non-noisy samples
- After that it will start overfitting
- In the training process, we can use the current predictions of our network
- Before overfitting, our model should correctly label the noisy samples
- This can be used as a weighting mechanism
- Currently trying to combine Bootstrapping Loss with our weighting method


BMM
GMM
Clean
Noisy
Empirical pdf
Normalized loss
0
1
2
3
0
.2
.4
.6
.8
1

Thank You!