Robust classification of Histology Images using Semi-supervision in Adverserial Autoencoders

A dissertation

submitted in partial fulfillment of requirements of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering

by

POORVI HEBBAR

(***170050094***)

Under the supervision of

Prof. Amit Sethi

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Powai, Mumbai – 400076

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Poorvi Hebbar
Roll no. 170050094
IIT Bombay

Acknowledgments

I would like to express my sincere gratitude to Prof. Amit Sethi for his valuable guidance and support throughout this project. I would like to thank him for his insightful suggestions which helped me in smooth and successful completion of the project. I am very thankful to Nikhil Cherian Kurian for his constant support and motivation at each and every step of this work. I am also thankful to Shreekanya Kodate and Gurprakash for always motivating me and helping me in case of any doubts. I also thank Shashikant for helping whenever needed. I thank all the members of MeDAL lab and my colleagues for making my stay in IIT Bombay a lifetime experience with tons of memories. Finally, I would like to thank my family and friends for their constant moral support. This work is dedicated to all of them.

Poorvi Hebbar

Abstract

Deep learning (DL) thrives on the availability of a large number of high quality images with reliable labels. Due to the large size of whole slide images in digital pathology, patches of manageable size are often mined for use in DL models. These patches are often variable in quality, weakly supervised, individually less informative, and noisily labelled. To improve classification accuracy even with these noisy input and labels in histopathology, we propose a novel method for robust feature generation using an adversarial autoencoder (AAE) . The likelihood of the features in the latent space of AAE has been utilized as a criterion to weigh the training samples. Different weighing schemes have been proposed for the framework and test the methods on two publicly available histopathology datasets. Consistent improvement in AUC scores using these methods is observed, and can be said that robust supervision strategies should be further explored for computational pathology. The model performance should not depend on the prior being used, but observations say that it might depend on the choice of priors, experiment has been done using different priors. One of the main objective of the project is to reduce the amount of labelled data for training, this can be achieved by exploring the semi-supervision ability of AAE.

Contents

Chapter 1

Introduction

Supervised Deep Learning (DL) models have consistently shown promising results for automated image analysis in medical image analysis over the last eight years [1] [2] [3]. However, the success of DL models depends on the availability of large datasets of high quality and correctly labelled images for training. When the quality of the training images or the accuracy of their labels degrade, the accuracy of the DL models trained using them reduces drastically [4]. Consequently, for automated medical image analysis in general, and computational pathology in particular, medical experts on a research team need to carefully label, annotate, and curate whole slide images (WSIs) to prepare training and testing datasets. This data preparation process is time-consuming and expensive.

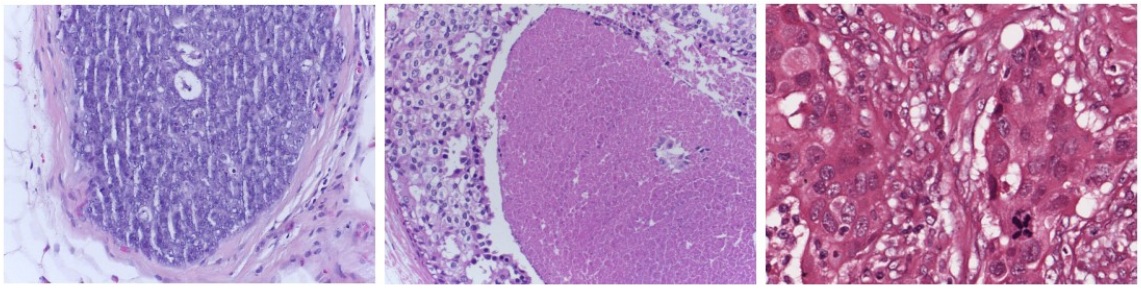


Figure 1.1: Diversity in the quality of histopathology images: (a) Venetian blinds artifact due to improper cut, (b) tissue degradation due to improper fixation, and (c) over-staining of eosin dye.

On the other hand, weakly supervised labels for WSIs (or large images, in general) are much easier to obtain. Such weak supervision disregards the heterogeneity in the quality and anatomical content of patches, For example, image quality can vary with tissue

preparation, staining, and slide preparation methods [5], as shown in Figure 1.1. Additionally, intra-tumoral heterogeneity is a natural phenomenon. Spatial heterogeneity in anatomy and quality combined with the large size of WSIs means that weakly supervised label propagation from the WSI to its patches leads to mislabeling of a certain unknown proportion of the individual patches.

Although weakly supervised learning techniques such as multiple instance learning (MIL) try to address these issues [6], the expressiveness of the generated features is sometimes not strong from the classification perspective as these methods extracts aggregate bag level features consisting of multiple instances [7]. Such bag formation from several instances can also reduce the number of training points available for supervision.

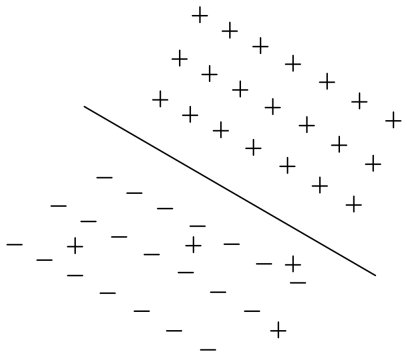


Figure 1.2: Dataset with noisy labels

Here, we address the problem of learning robust models for image classification in the face of label noise and weak supervision. We use adversarial autoencoders to get sample-wise weights for each training image. We assume and confirm that the samples that can deteriorate the model training will fall into the lesser likelihood regions of the class specific distribution priors. This eliminates the need for additional optimization steps to calculate the sample-wise weights . We have first tested the model for two-class classification and then extended it to 4-class classification problem with variable priors.

Chapter 2

Prior Work

Previously in RnD-1, we had worked on an adversarial autoencoders and used it to classify noisy datasets.

2.1 Adversarial Autoencoder

Unlike other techniques that involve additional optimization steps or predefined curricula, we utilize the likelihood of the features of a sample in the latent space of AAE to derive its dynamic weight. We hypothesized that the less informative or the noisy samples will fall into the low likelihood regions of the regularized latent space.

2.1.1 Architecture description

As shown in figure 2.1, our model has an encoder block that acts as a feature generator for the task-specific classifier as well as for the AAE. The adversarial training for the generated features in the d-dimensional (d=32 in our experiments) latent space is performed with the aid of the discriminator block. The discriminator compares the features generated by the encoder with a random vector sampled from its corresponding class specific prior distribution, which we assume to be a d-dimensional Gaussian distribution. Since the task-specific classifier also needs to be optimized, the encoder block in this adversarial task has to generate samples that are optimized both for the classifier as well as to fool the discriminator. Here the role of decoder block in our architecture is to ensure that all images belonging to a particular class are pulled towards a mean feature vector of its Gaussian prior. During the training phase, feature generator and discriminator will perform the following min-max game to generate the samples:

$$\arg\max_{Disc} \arg\min_{Enc,Dec,Cls} [C(X, L_X + \lambda_1 R(X) - \lambda_2 D(X, P)] \tag{2.1}$$

where C is the classification loss, R is the reconstruction loss, and D is the discriminator

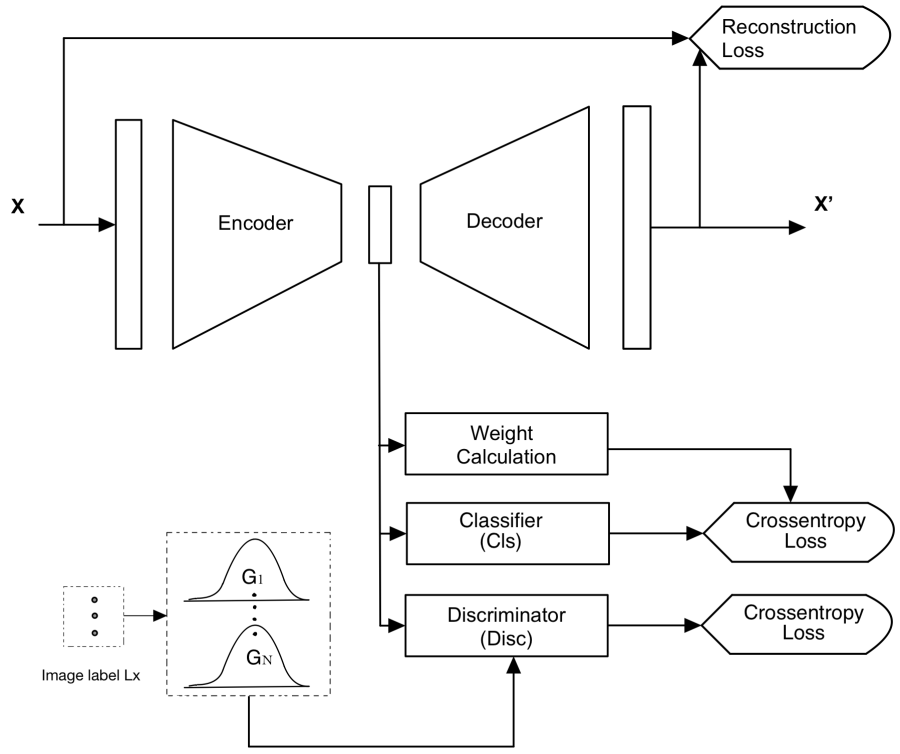


Figure 2.1: Adversarial auto-encoder based architecture used for robust classification.

loss. These losses are sample-wise weighted cross-entropy, mean square error, and cross entropy respectively in the scheme. Further, X is a training sample and L_X is its label, and P is a sample from the prior distribution. Hyperparameters λ_1 and λ_2 were decided based on validation. When the discriminator reports low confidence in distinguishing a real Gaussian sample against the feature vector, the adversarial training is declared successful. Training the classifier separately on top of a well-trained AAE generator gave poor classifier performance because such a feature generator was agnostic to the classification task a priori.

For convolution and transpose-convolution (upsampling) layers, we used kernels of size 5×5 with a stride of 3. Batch- normalization was used after each layer of all four segments of the model. The activation function used throughout the network is leaky-ReLU with a slope of 0.2 for the negative inputs. We used Adam optimizer with a learn rate of 0.001. Data imbalance in our experiments was accounted using a proportionate weighted sampling for each mini-batch updates.

2.1.2 Weighing Schemes

We explored the use of following schemes for sample-wise weighting in the loss C to train the adversarial autoencoder based classifier (AAEC):

- **Binary weighting (BW)**: The sample's class-specific likelihood is compared to a global threshold, which is a tunable hyperparameter, to decide on its inclusion or exclusion (binary weight).
- **Binary normalized weighting (BNW)**: Binary weighting is computed separately within each training minibatch by normalizing the likelihood within the batch and comparing that to a threshold.
- **Normalised weighting (NW)**: Continuous weights are obtained by normalizing the likelihood within each batch.

2.1.3 Flowgraph

The figure 2.4 shows the details steps involved in training the model. The experiments have been performed on 2 types of dataset, one in which we deliberately add noise (in steps of 5% from 0 to 20), so that we know the noise content in the dataset and tested with dataset containing unknown level of noise (this is the first step of data preparation). As in a medical images dataset, the samples of positive cases will be lower than negative cases, there will be an imbalance in the dataset. If there is huge imbalance in th dataset, it might create a problem for the model as it will only learn about negative samples and still give high accuracy, so it is important to balance such dataset. The next steps have been clearly shown in the flowgraph and have been already discussed along the objective loss function in 2.1. The same steps have been used for 4class classification also.

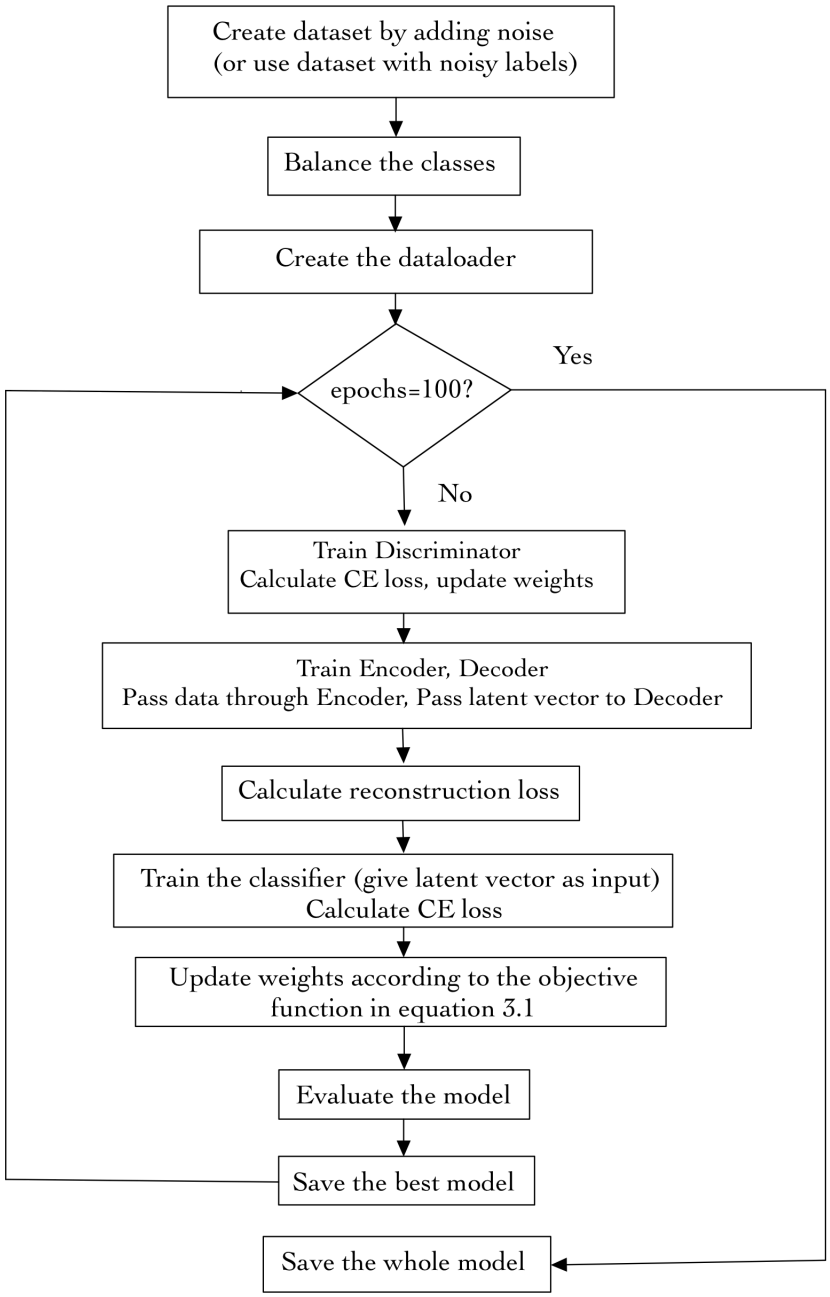


Figure 2.2: Flowgraph showing steps involved in training the model.

2.2 Experiments for Binary Datasets

2.2.1 BACH Dataset (DCIS vs IDC)

conv = [3,4,8,16]
fc = [32,16,2]
Noise=20%

Architecture	Accuracy	Loss	ROC-AUC Score
Simple CNN	57.8125	0.66742	0.682
Resnet	60.1562	0.6953	0.826
Binary Weights	82.03125	0.68844	0.836
Normalized Weights	76.5625	0.576888	**0.855**
Binary Normalized Weights	81.25	0.49741	0.842
Binary followed by no Weights	82.03125	0.56707	0.810
Normalized followed by no Weights	82.8125	0.62552	0.833
Binary Normalized followed by no Weights	78.125	0.54860	0.841

Table 2.1: Comparison of Weighing Schemes

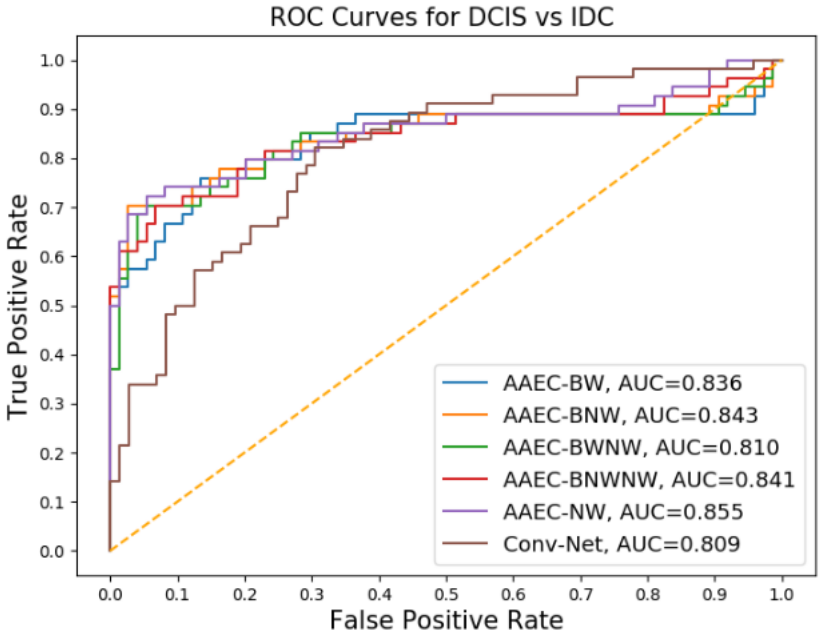


Figure 2.3: Insitu (DCIS) vs Invasive (IDC)

2.2.2 Breakhis Dataset (Tumor vs Non-Tumor)

Label noise level - Training Scheme	0%	5%	10%	15%	20%
Conv-Net	**0.903**	**0.851**	0.802	0.790	0.796
AAEC-BWNW	0.815	0.802	0.802	0.806	0.763
AAEC-BW	0.823	0.819	0.819	0.805	0.802
AAEC-BNWNW	0.810	0.808	0.808	0.797	0.790
AAEC-BNW	0.836	0.828	**0.827**	0.808	0.803
AAEC-NW	0.838	0.829	0.826	**0.821**	**0.814**

Table 2.2: ROC-AUC scores for different noise levels tumor vs. non-tumor

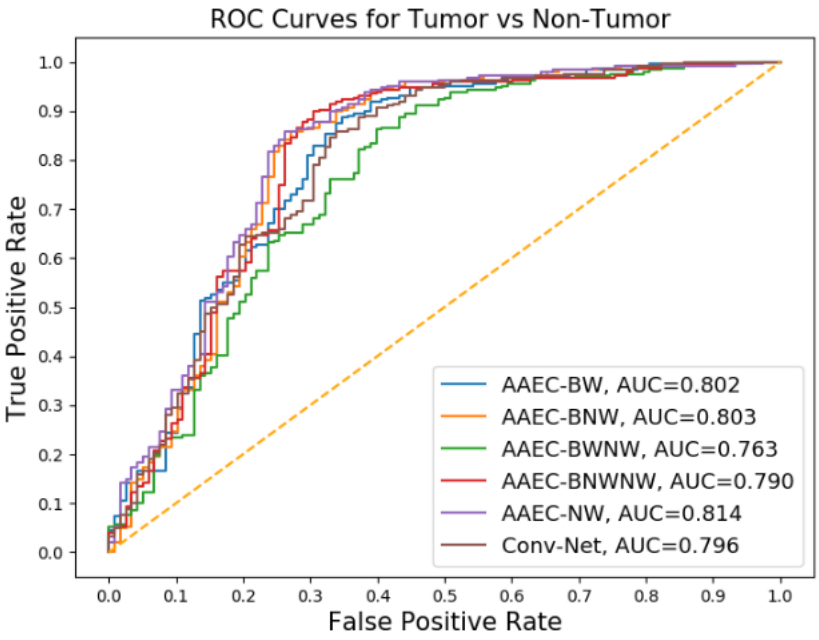


Figure 2.4: Tumour vs Non-Tumour: 20% noise

2.3 4-Class Classification Problems

The binary classification task was achieved successfully with high accuracy. To test the generalization of the model for more than two class classification, we have extended it to Four class classification problem. The next idea was to use multivariate Gaussian with unit mean in 4 different quadrants 2.5 of unit hyperspehere. But as can be observed, there

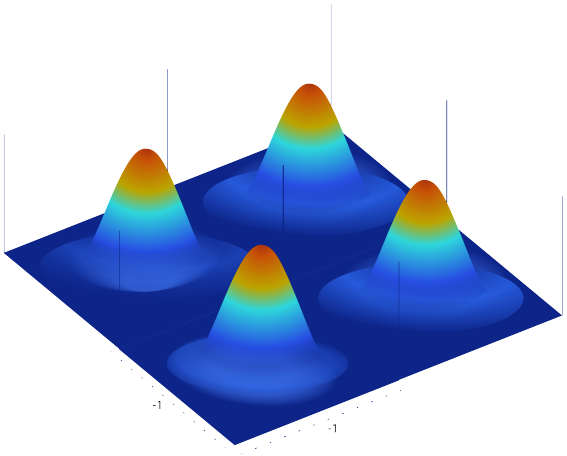
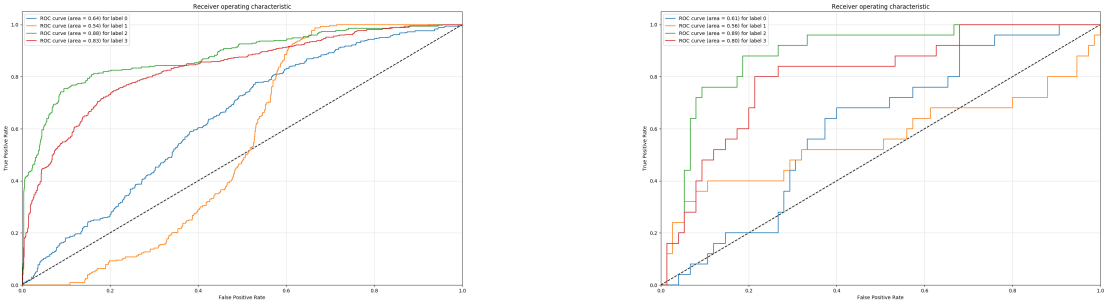


Figure 2.5: Gaussian lying in 4 quadrants conditioned on label information.



(a) clean labels

(b) noisy labels

Figure 2.6: ROC curve for BACH dataset containing 4 classes, clean and noisy labels using single multivariate Gaussian

are very low ROCs for a few labels and few classes are not well recognised in testing.

Chapter 3

Objective

3.1 Motivation

4-class classification results using the prior model was unsatisfactory. When our model was extended to 4-class classification with multivariate gaussian priors, we found that ROCs for a few labels were very low and they were not being recognised in testing. So this was one of the main motivation for the RnD-2 work.

Fully supervised algorithms don't work well on datasets with lots of unlabelled data, and/or noisy labels and/or outlier samples. So, semi-supervision is needed to address the problem of noisy labels effectively. Also, Unlabelled dataset might have outliers. Out-of-distribution samples contained in the unlabelled data also needs to be addressed. So, if we transfer the knowledge gained from labelled data to unlabelled dataset without making sure the outlier samples are properly filtered out, the overall accuracy will suffer.

3.2 Aim

- Robust classification on noisy labeled data and a large pool of unlabelled data using semi supervision
- Negative Learning for segregating clean dataset from noisy samples
- Improve efficacy of semi-supervision using energy based model.
- Address a more complex open-set scenario, where out of distribution samples are contained in unlabelled data
- Compare results with the temporal baseline model

Chapter 4

Literature Survey

4.1 Multi-Task Semi-Supervised Adversarial Autoencoding for Speech Emotion Recognition(SER)

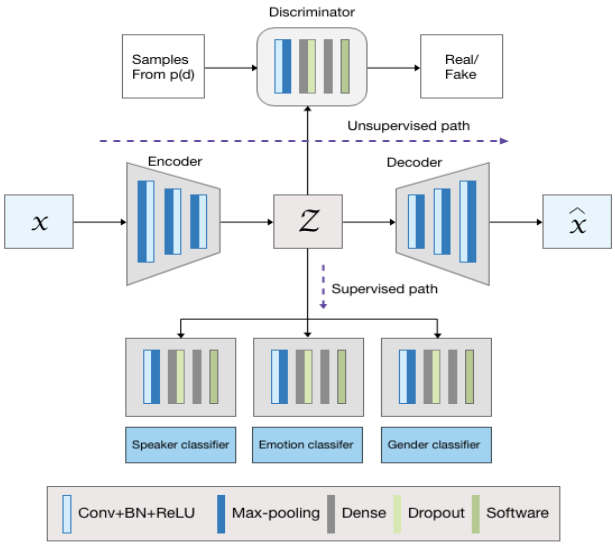


Figure 4.1: Illustration of the proposed multitask framework using a semi-supervised adversarial autoencoder (AAE).

Due to scarcity in the dataset available for speech emotion recognition, the authors have implemented auxiliary tasks(multi-task learning) for which abundant data is available along with AAE to achieve the main task of emotion recognition. The proposed model achieves state-of-the-art performance.

The unsupervised learning capability of the AAE along with supervised training of three different classifiers(Speaker, emotion and gender) creates a semi-supervised model. The training of the model is done in 3 phases [8] :

1. **Reconstruction Phase** : the autoencoder updates the encoder E_θ and the decoder D_δ and minimises the reconstruction error by encoding x into latent representation z.

2. **Regularisation Phase** : The discriminator is first trained to distinguish between real(from prior distribution $\mathcal{N}((d; 0, I))$ and fake(latent vector) samples. Goal is to fool the discriminator(D_ω) by learning to encode data that D_ω perceives as real. Discriminator weights and biases are kept fix and encoder is updated.

3. **Classification Phase** : The latent code generated is given as input to the classifiers(C_ϕ) and cross-entropy loss is minimized. The encoder is updated again.

The objective function is as follows :

$$\begin{aligned} L_{enc} = \min_{\theta}(&E_{x\sim p_x}[\log(1 - D_\omega(E_\theta(x)))] \\ &+ E_{x\sim p_x}[\beta\mathcal{L}_{AE}(x, D_\delta(E_\theta(x)))] \\ &+ E_{x,y\sim p_{X,Y}}[\mathcal{L}_c(E_\theta(x), y; C_\phi)]) \end{aligned} \tag{4.1}$$

The most important point to be noted is that the encoder is being updated in all the three phases and the prior used is Gaussian with zero mean and standard deviation of 1, which is of the same dimensions as that of the latent vector. This paper proves that there is no need to condition on the modes of the Gaussian.

4.1.1 Observations

In semi-supervision Shreekanya and I tried to first exploit the unsupervised learning property of the AAE and trained the model without the classifier. So, in this manner, the encoder, decoder and the discriminator will be ready with initial weights and biases. In unsupervised learning, we trained on the clean BACH dataset to get these initial weigths. Then the same model with a classifier head was trained again, now, we have some learned features by encoder, the only random weights are that of the classifier. But the reconstruction using 2-classes was good but that of 4-classes was worse, hence we did not get any significant improvement in the accuracy for classification. The reconstructed images have been shown below.

4.1.2 Results

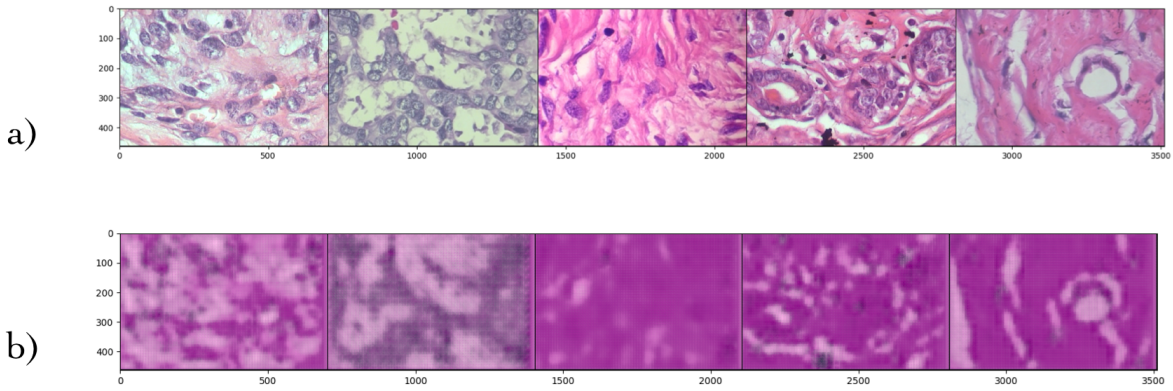


Figure 4.2: Reconstruction using Unsupervised Learning for binary classification

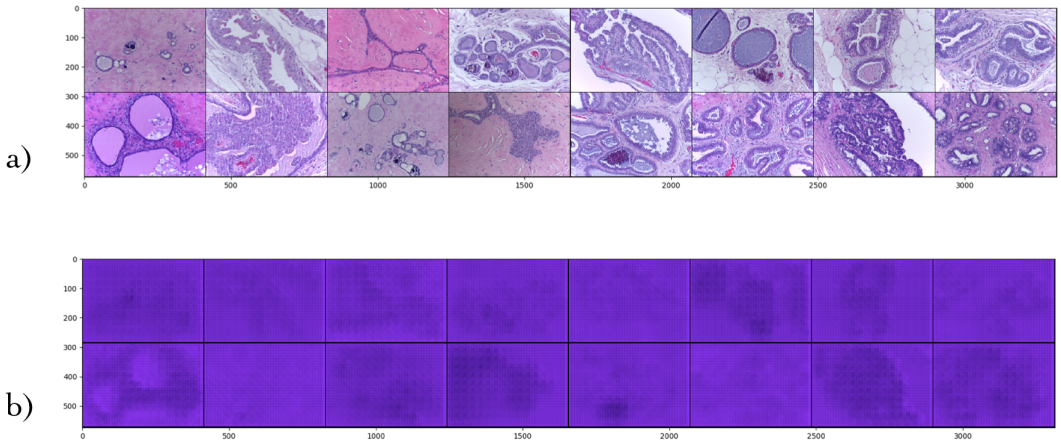


Figure 4.3: Reconstruction using Unsupervised Learning for 4-class classification

4.2 Using Pre-training can Improve Model Robustness and Uncertainty (ICML 2019)

The authors have demonstrated through extensive experiments on adversarial examples (label corruption, class imbalance, out-of-distribution detection, and confidence calibration), large gains from pre-training and complementary effects with task-specific methods. Sometimes pre-training may not improve performance on traditional classification metrics and segmentation tasks but it improves model robustness and uncertainty estimates.

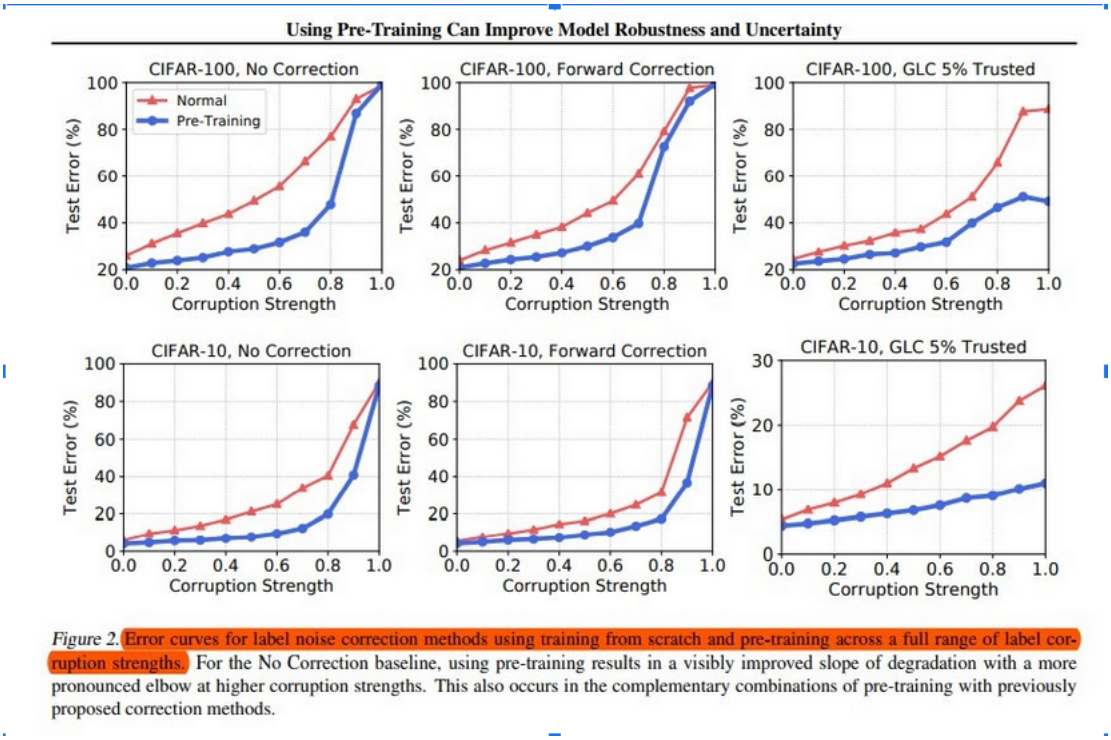


Figure 2. Error curves for label noise correction methods using training from scratch and pre-training across a full range of label corruption strengths. For the No Correction baseline, using pre-training results in a visibly improved slope of degradation with a more pronounced elbow at higher corruption strengths. This also occurs in the complementary combinations of pre-training with previously proposed correction methods.

For pretraining, Downsampled ImageNet, which is the 1,000-class ImageNet dataset resized to 32 × 32 resolution is being used for all the experiments. For ablation experiments, they remove 153 CIFAR-10-related classes from the dataset. Wide-ResNet model has been used in all the experiments, SGD with Nesterov momentum and different learning rates for pre-training and fine-tuning, datasets on which results have been shown are CIFAR-10 and CIFAR-100.

4.3 DivideMix: Learning With Noisy Labels as Semi-Supervised Learning (ICLR 2020)

In this work, we propose DivideMix, a novel framework for learning with noisy labels by leveraging semi-supervised learning techniques. In particular, DivideMix models the per-sample loss distribution with a mixture model to dynamically divide the training data into a labeled set with clean samples and an unlabeled set with noisy samples, and trains the model on both the labeled and unlabeled data in a semi-supervised manner. To avoid confirmation bias, we simultaneously train two diverged networks where each network uses the dataset division from the other network. During the semi-supervised training phase, we improve the MixMatch strategy by performing label co-refinement and label

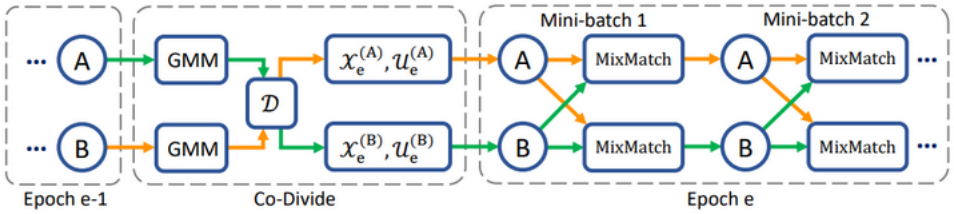


Figure 1: DivideMix trains two networks (A and B) simultaneously. At each epoch, a network models its per-sample loss distribution with a GMM to divide the dataset into a labeled set (mostly clean) and an unlabeled set (mostly noisy), which is then used as training data for the other network (*i.e.* co-divide). At each mini-batch, a network performs semi-supervised training using an improved MixMatch method. We perform label co-refinement on the labeled samples and label co-guessing on the unlabeled samples.

co-guessing on labeled and unlabeled samples, respectively.

Algorithm 1: DivideMix. Line 4-8: co-divide; Line 17-18: label co-refinement; Line 20: label co-guessing.

Input: $\theta^{(1)}$ and $\theta^{(2)}$, training dataset $(\mathcal{X}, \mathcal{Y})$, clean probability threshold τ, number of augmentations M, sharpening temperature T, unsupervised loss weight λ_u, Beta distribution parameter α for MixMatch.
$\theta^{(1)}, \theta^{(2)} = \text{WarmUp}(\mathcal{X}, \mathcal{Y}, \theta^{(1)}, \theta^{(2)})$ *// standard training (with confidence penalty)*
while $e < \text{MaxEpoch}$ **do**
  $\mathcal{W}^{(2)} = \text{GMM}(\mathcal{X}, \mathcal{Y}, \theta^{(1)})$ *// model per-sample loss with $\theta^{(1)}$ to obtain clean proabability for $\theta^{(2)}$*
  $\mathcal{W}^{(1)} = \text{GMM}(\mathcal{X}, \mathcal{Y}, \theta^{(2)})$ *// model per-sample loss with $\theta^{(2)}$ to obtain clean proabability for $\theta^{(1)}$*
  for $k = 1, 2$ **do** *// train the two networks one by one*
    $\mathcal{X}_e^{(k)} = \{(x_i, y_i, w_i) | w_i \geq \tau, \forall (x_i, y_i, w_i) \in (\mathcal{X}, \mathcal{Y}, \mathcal{W}^{(k)})\}$ *// labeled training set for $\theta^{(k)}$*
    $\mathcal{U}_e^{(k)} = \{x_i | w_i < \tau, \forall (x_i, w_i) \in (\mathcal{X}, \mathcal{W}^{(k)})\}$ *// unlabeled training set for $\theta^{(k)}$*
    for iter $= 1$ **to** num_iters **do**
      From $\mathcal{X}_e^{(k)}$, draw a mini-batch $\{(x_b, y_b, w_b); b \in (1, ..., B)\}$
      From $\mathcal{U}_e^{(k)}$, draw a mini-batch $\{u_b; b \in (1, ..., B)\}$
      for $b = 1$ **to** B **do**
        for $m = 1$ **to** M **do**
          $\hat{x}_{b,m} = \text{Augment}(x_b)$ *// apply m^{th} round of augmentation to x_b*
          $\hat{u}_{b,m} = \text{Augment}(u_b)$ *// apply m^{th} round of augmentation to u_b*
        end
        $p_b = \frac{1}{M} \sum_m \text{p}_{\text{model}}(\hat{x}_{b,m}; \theta^{(k)})$ *// average the predictions across augmentations of x_b*
        $\bar{y}_b = w_b y_b + (1 - w_b) p_b$
        // refine ground-truth label guided by the clean probability produced by the other network
        $\hat{y}_b = \text{Sharpen}(\bar{y}_b, T)$ *// apply temperature sharpening to the refined label*
        $\bar{q}_b = \frac{1}{2M} \sum_m \left(\text{p}_{\text{model}}(\hat{u}_{b,m}; \theta^{(1)}) + \text{p}_{\text{model}}(\hat{u}_{b,m}; \theta^{(2)})\right)$
        // co-guessing: average the predictions from both networks across augmentations of u_b
        $q_b = \text{Sharpen}(\bar{q}_b, T)$ *// apply temperature sharpening to the guessed label*
      end
      $\hat{\mathcal{X}} = \{(\hat{x}_{b,m}, \hat{y}_b); b \in (1, ..., B), m \in (1, ..., M)\}$ *// augmented labeled mini-batch*
      $\hat{\mathcal{U}} = \{(\hat{u}_{b,m}, q_b); b \in (1, ..., B), m \in (1, ..., M)\}$ *// augmented unlabeled mini-batch*
      $\mathcal{L}_\mathcal{X}, \mathcal{L}_\mathcal{U} = \text{MixMatch}(\hat{\mathcal{X}}, \hat{\mathcal{U}})$ *// apply MixMatch*
      $\mathcal{L} = \mathcal{L}_\mathcal{X} + \lambda_u \mathcal{L}_\mathcal{U} + \lambda_r \mathcal{L}_{\text{reg}}$ *// total loss*
      $\theta^{(k)} = \text{SGD}(\mathcal{L}, \theta^{(k)})$ *// update model parameters*
    end
  end
end

4.4 MixMatch: A Holistic Approach to Semi-Supervised Learning (NeurIPS 2019)

In this work, we unify the current dominant approaches for semi-supervised learning to produce a new algorithm, MixMatch, that guesses low-entropy labels for data-augmented unlabeled examples and mixes labeled and unlabeled data using MixUp. Experimentally,

> **Contribution** Motivated by these issues, we introduce a simple and data-agnostic data augmentation routine, termed *mixup* (Section 2). In a nutshell, *mixup* constructs virtual training examples
>
> $$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \quad \text{where } x_i, x_j \text{ are raw input vectors}$$
> $$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \quad \text{where } y_i, y_j \text{ are one-hot label encodings}$$
>
> (x_i, y_i) and (x_j, y_j) are two examples drawn at random from our training data, and $\lambda \in [0, 1]$. Therefore, *mixup* extends the training distribution by incorporating the prior knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated targets. *mixup* can be implemented in a few lines of code, and introduces minimal computation overhead.

we show that MixMatch obtains state-of-the-art results on all standard image benchmarks (section 4.2), and reducing the error rate on CIFAR-10 by a factor of 4; We further show in an ablation study that MixMatch is greater than the sum of its parts; We demonstrate in section 4.3 that MixMatch is useful for differentially private learning, enabling students in the PATE framework [36] to obtain new state-of-the-art results that simultaneously strengthen both privacy guarantees and accuracy. In short, MixMatch introduces a unified loss term for unlabeled data that seamlessly reduces entropy while maintaining consistency and remaining compatible with traditional regularization techniques.

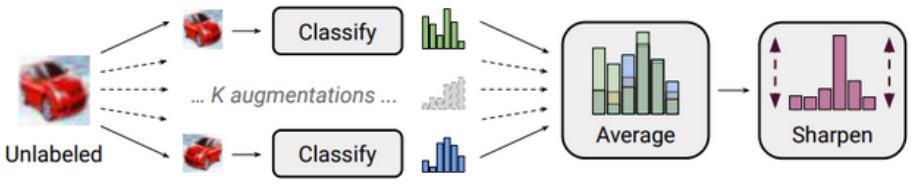


Figure 1: Diagram of the label guessing process used in MixMatch. Stochastic data augmentation is applied to an unlabeled image K times, and each augmented image is fed through the classifier. Then, the average of these K predictions is "sharpened" by adjusting the distribution's temperature. See algorithm 1 for a full description.

Chapter 5

Current Methodology

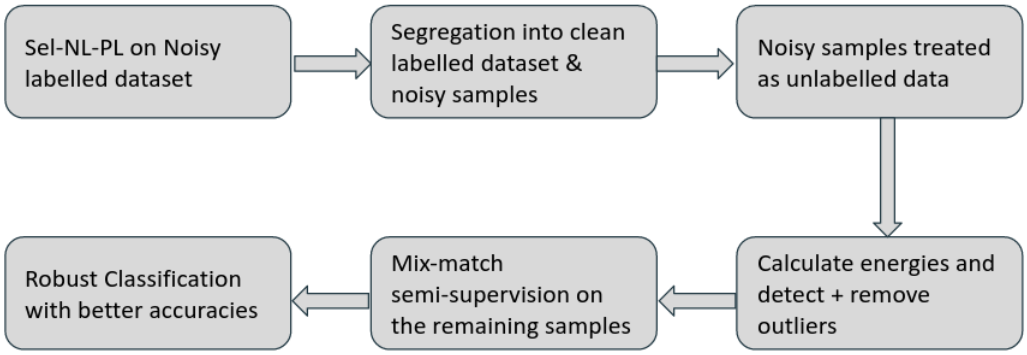
5.1 Breakhis Dataset

	Train	Test	Eval	Total
Benign	470	59	59	588
Malignant	985	124	123	1232
Total	1455	183	182	1820

The training samples were randomly selected and categorized as 500 labelled samples, 955 unlabelled samples, and 450 outliers (overstained or understained images). The first basic 2class dataset we worked on was breakhis dataset. We randomly selected and categorised the training samples into 500 labelled samples, 955 unlabelled samples and 450 of the overstained and understained samples as outliers.

5.2 Model

In the current methodology, to be explained in brief, we first use Selective NL PL to segregate the noisy labelled dataset into clean labelled and noisy samples. Noisy samples are added to the unlabelled data. We then calculate energies, detect and remove outliers. With the remaining labelled-unlabelled data pool, we apply mixmatch semisupervision to get a robust classification with much better accuracies. I will be explaining the methods and the results obtained in each in detail.



5.3 Negative Learning on Noisy dataset

5.3.1 Complementary Labels

Algorithm 1 Complementary label generation

Input: Training label $y \in \mathcal{Y}$

while iteration **do**

$\overline{y}$ = Randomly select from $\{1, ..., C\} \setminus \{y\}$

Output: Complementary label $\overline{y}$

We first find a random complementary label for every sample. So if y is actual training label, and c is the total number of classes, we randomly select a y' from the set 1 ..c-y We also define a c-dimensional probability vector p for every sample. So, for k=1..c, pk is the probability of the particular sample belonging to class c

5.3.2 Positive Learning

$$\mathcal{L}(f, y) = -\sum_{k=1}^{c} \boldsymbol{y}_k \log \boldsymbol{p}_k$$

In positive learning the main aim is to maximise py or the probability corresponding to the actual training label y. So the loss function becomes the cross entropy loss and as training continues, py tends to 1, satisfying the purpose of positive learning. Incase of clean dataset, positive learning works quite good.

5.3.3 Negative Learning

$$\mathcal{L}(f, \overline{y}) = -\sum_{k=1}^{c} \overline{\boldsymbol{y}}_k \log(1 - \boldsymbol{p}_k)$$

In negative learning, instead of saying this sample belongs the class y, we say it doesn't belong to a class y' !=y. Y' here is the randomly selected complementary label and we try to make py' tend to 0 inorder to make the model learn that the sample's target is Not y'. The loss function therefore has a (1-pk) term in order to optimise the probability of complementary label far from 1 or in other words close to 0. In a c¿2 class classification, as the probability of y'!=y being wrong is much higher than the probability of y being correct, Negative learning tends to make less errors as compared to positive learning in case of noisy dataset.

5.3.4 Results

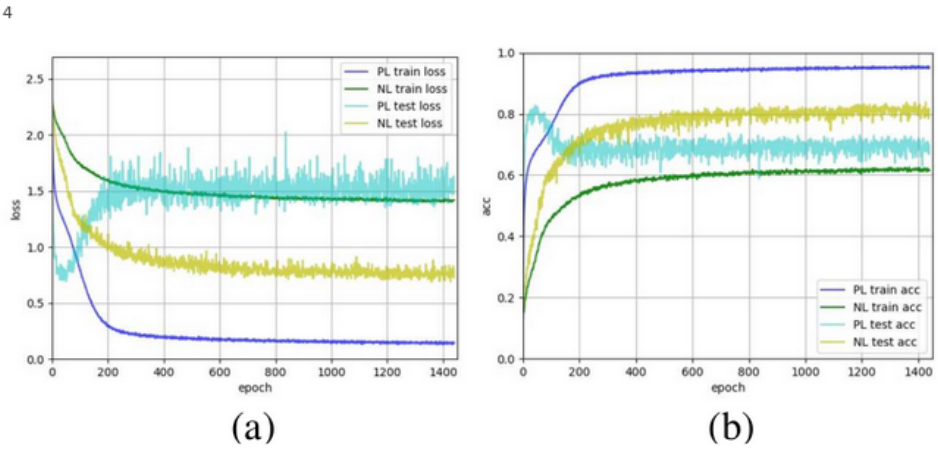


Figure 5.1: Comparison b/w NL and PL (a)Loss graph (b) Accuracy graph

In the first graph, loss vs epoch, the light blue curve corresponds to the test loss in positive learning, there is a drop at the early stage and then a constant value is reached. This is due to overfitting at the later epochs of training. As against to this, the yellow curve represents the test loss in negative learning. The curve drops gradually and overfitting is not observed as training progresses as we had expected. In these histograms, we have data vs probability. Blue is the clean data and orange represents the noisy samples. In the first figure, as we can see, the confidence of both clean and noisy data increased with PL training, whereas in the second figure corresponding to NL, the confidence of noisy

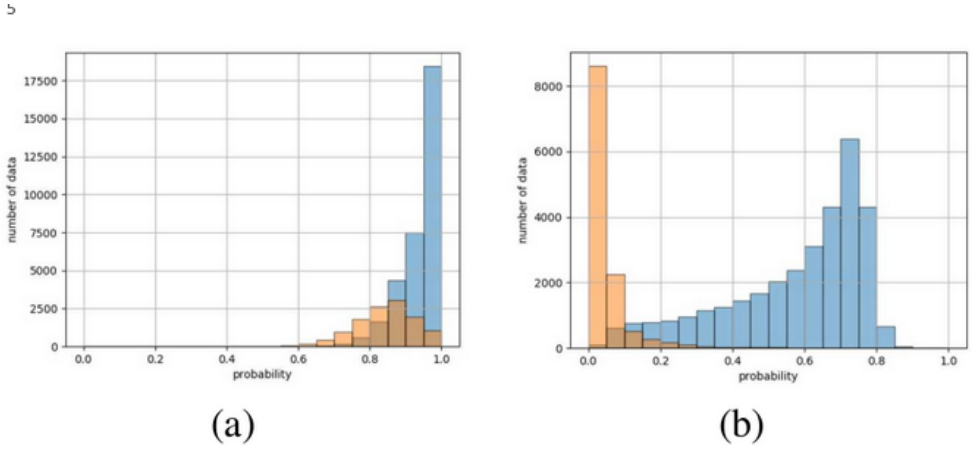


Figure 5.2: Comparison b/w NL and PL (a)Loss graph (b) Accuracy graph

data is much lower as compared to clean data. This is another evidence to overfitting not observed in NL.

5.3.5 Selective NL-PL

Coming to Selective NL-PL, we use both NL and PL. We know that NL avoids overfitting and works well when it comes to noisy data but PL is actually better when the data is clean. So, in sel NL PL, we start with Negative learning and minimize the corresponding loss till a confidence threshold of 1/c for the sample target y is reached. Once that's done, we can safely now consider the data to be almost clean and use PL to further better the results.

Algorithm 2 Overall process of SelNLPL

Input: Training data $(\boldsymbol{x}, y) \in (\mathcal{X}, \mathcal{Y})$, network $f(\boldsymbol{x}; \boldsymbol{\theta})$, total epoch T

for $i \leftarrow 1$ to T **do** ▷ NL
  Batch $\leftarrow$ Sample $\boldsymbol{x}$
  Update f by minimizing Eq. 2

for $i \leftarrow 1$ to T **do** ▷ SelNL
  Batch $\leftarrow$ Sample $\boldsymbol{x}$ if $\boldsymbol{p}_y > 1/\mathrm{c}$
  Update f by minimizing Eq. 2

for $i \leftarrow 1$ to T **do** ▷ SelPL
  Batch $\leftarrow$ Sample $\boldsymbol{x}$ if $\boldsymbol{p}_y > \gamma$
  Update f by minimizing Eq. 1

Output: Network $f(\boldsymbol{x}; \boldsymbol{\theta})$

- **Selective NL:** Improves convergence after NL; confidence threshold of 1/c.
- **Selective PL:** More accurate than NL for clean labels; confidence threshold of 0.5
- **Selective NL and PL:** used for filtering noisy data from clean data; confidences separated by a large margin.

5.3.6 Results and Observations

As we can see from figure c, after selectiveNl, i.e NL followed by confidence thresholding, the segregation of clean and noisy samples happen. We then use PL to further increase the confidence margin as can be seen in figure d. Also from the accuracy graph, one can

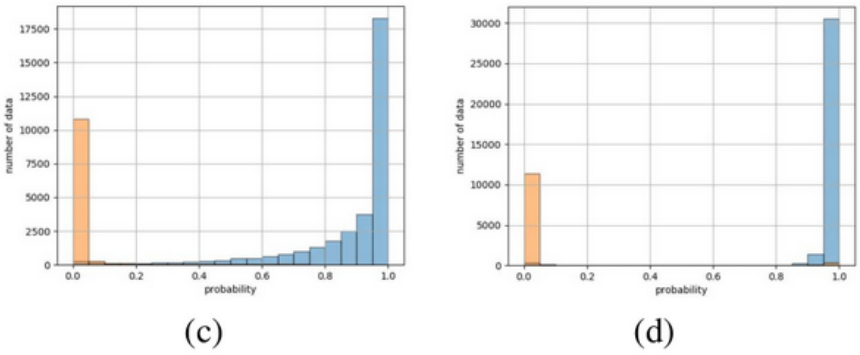


Figure 5.3: Histogram of training data after (c) Sel-NL (d) Sel-NL-PL

see that the accuracies keeps on increasing with NL, confidence thresholding or inother words selective NL and finally PL.

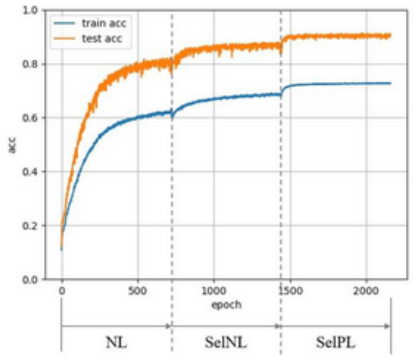


Figure 5.4: Accuracy graph. Training performed sequentially with NL, SelNL, SelPL

5.4 Pseudolabelling Method

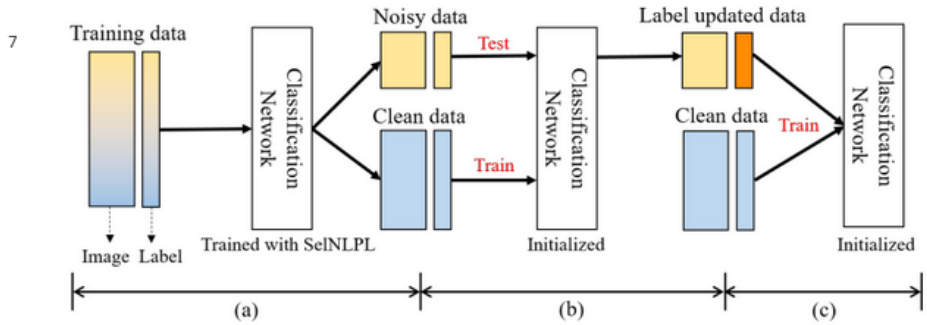
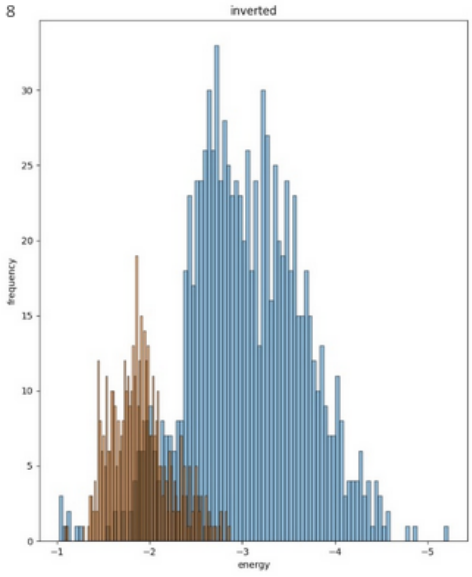


Figure 5.5: Accuracy graph. Training performed sequentially with NL, SelNL, SelPL

- Division of training data into clean or noisy data with CNN trained with SelNLPL
- Training initialized CNN with clean data from (a), then noisy data's label is updated with the soft label
- Clean data and label-updated noisy data are both used for training initialized CNN in the final step.

5.5 Energy Based OOD detection model



Energy based OOD model is used to detect outliers before semi supervision. For calculating the energy of a sample we take the logarithm of (sum of e to the power of sample logits). If y1,y2 are the logits of a particular input sample x, from the output nodes of the classifier, log(eŷ1+eŷ2) can be a measure of energy of the sample x.

Higher the value in negative scale, greater is the probability of the sample being an inlier. For instance -0.1 would imply an outlier while -2 would imply an inlier. In the graph, we have frequency vs energy of the samples, and blue represents the inliers and brown the outliers. Keeping in mind the negative sign, We can easily see that inliers have lower energies as compared to outliers.

5.6 Mixmatch Semisupervision

In every batch, every labeled data point is augmented once and every unlabeled data point is augmented K times. Augmentation here means any kind of transformation like horizontal flip, crop etc. The model is asked to predict the target class for all the K augmented entries and then their average is taken as the prediction for all the K entries. This average is sharpened to minimize the entropy and then taken as the final prediction. Augmented labeled and unlabeled data are then concatenated and shuffled to get a new set W. Say —X— is the size of the labelled data in the batch, then this labelled data is "mixed up" with first —X— entries of W to get X'. After this Unlabeled data in the batch is "mixed up" with rest of the entries of W to get U'. So if x1 and x2 are to be mixed up, we do a linear combination with a hyper parameter lambda as lambda*x1 + (1-lambda)*x2 and same with their corresponding targets.

Algorithm 1 MixMatch ingests a batch of labeled data $\mathcal{X}$ and a batch of unlabeled data $\mathcal{U}$ and produces a collection $\mathcal{X}'$ of processed labeled examples and a collection $\mathcal{U}'$ of processed unlabeled examples with "guessed" labels.

Input: Batch of labeled examples and their one-hot labels $\mathcal{X} = \big((x_b, p_b); b \in (1, \dots, B)\big)$, batch of unlabeled examples $\mathcal{U} = \big(u_b; b \in (1, \dots, B)\big)$, sharpening temperature T, number of augmentations K, Beta distribution parameter α for MixUp.
for $b = 1$ **to** B **do**
  $\hat{x}_b = \text{Augment}(x_b)$ *// Apply data augmentation to* x_b
  for $k = 1$ **to** K **do**
    $\hat{u}_{b,k} = \text{Augment}(u_b)$ *// Apply* k^{th} *round of data augmentation to* u_b
  end for
  $\bar{q}_b = \frac{1}{K} \sum_k \text{p}_{\text{model}}(y \mid \hat{u}_{b,k}; \theta)$ *// Compute average predictions across all augmentations of* u_b
  $q_b = \text{Sharpen}(\bar{q}_b, T)$ *// Apply temperature sharpening to the average prediction (see eq. (7))*
end for
$\hat{\mathcal{X}} = \big((\hat{x}_b, p_b); b \in (1, \dots, B)\big)$ *// Augmented labeled examples and their labels*
$\hat{\mathcal{U}} = \big((\hat{u}_{b,k}, q_b); b \in (1, \dots, B), k \in (1, \dots, K)\big)$ *// Augmented unlabeled examples, guessed labels*
$\mathcal{W} = \text{Shuffle}\big(\text{Concat}(\hat{\mathcal{X}}, \hat{\mathcal{U}})\big)$ *// Combine and shuffle labeled and unlabeled data*
$\mathcal{X}' = \big(\text{MixUp}(\hat{\mathcal{X}}_i, \mathcal{W}_i); i \in (1, \dots, |\hat{\mathcal{X}}|)\big)$ *// Apply* MixUp *to labeled data and entries from* $\mathcal{W}$
$\mathcal{U}' = \big(\text{MixUp}(\hat{\mathcal{U}}_i, \mathcal{W}_{i+|\hat{\mathcal{X}}|}); i \in (1, \dots, |\hat{\mathcal{U}}|)\big)$ *// Apply* MixUp *to unlabeled data and the rest of* $\mathcal{W}$
return $\mathcal{X}', \mathcal{U}'$

$$\lambda \sim \text{Beta}(\alpha, \alpha)$$
$$\lambda' = \max(\lambda, 1 - \lambda)$$
$$x' = \lambda' x_1 + (1 - \lambda') x_2$$
$$p' = \lambda' p_1 + (1 - \lambda') p_2$$

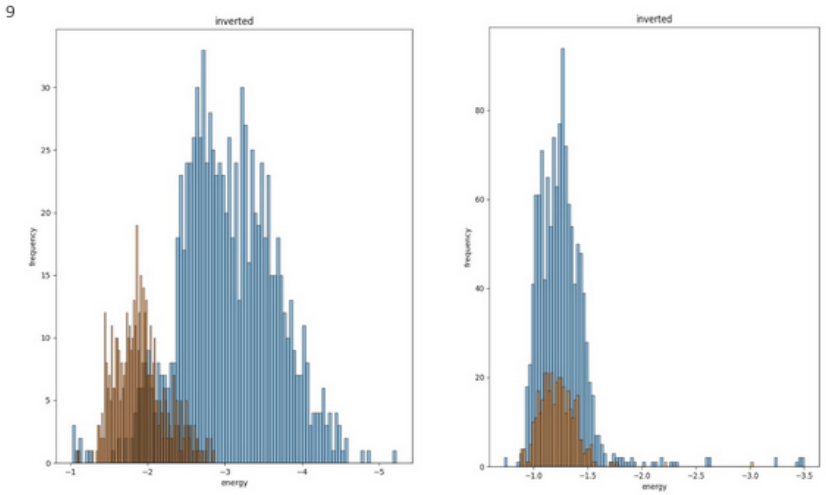
$$\text{Sharpen}(p, T)_i := p_i^{\frac{1}{T}} \Big/ \sum_{j=1}^{L} p_j^{\frac{1}{T}}$$

As lambda is ≥ 0.5, MixUp gives more importance to the first point than the second.

That indirectly implies model's prediction of X' should correspond to labeled output and model's prediction of U' should correspond to unlabeled guesses.

For losses, we use the cross entropy loss directly for the labelled data and mse loss for the unlabelled data.

5.6.1 Results



After training 3 epochs of warm-up phase

After warm-up, sample selection and mix-match

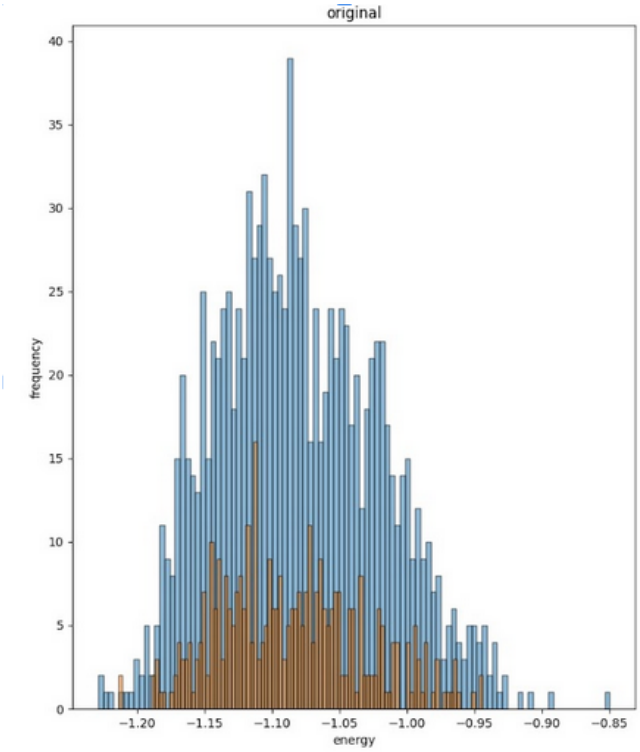
As semisupervised training progresses, energy of samples starts increasing and the energies of the inliers tend to coincide with the outliers. Accuracy for Mixmatch with outliers and single-energy threshold on Breakhis Data: Train: 89%, Val 84%, Test: 85%

When extended to 4 class, we used Et breast cancer dataset and used EH, Endometrial Hyperplasia as a open set class to include the outliers.

Normal Endometrium	Endometrial Polyp	Endometrial Hyperplasia	Endometrial Adenocarcinoma	Total
1333	636	800	535	3302

We didn't use the 4-class breakhis dataset as before as it didnt have as many outliers and noisy samples as needed. The training samples were selected as: Labeled- 200 per class (EA,EP and EH), 300 from NE; Unlabeled : 2402

Accuracies observed: Train: 73.25%; Val: 68.18%; Test: 67.01%



It was observed that the testing and validation accuracies were quite less for this dataset and as we can see from the graph, the outliers were not being able to be distinguished itself. Even after increasing the warm-up epochs, no good separation was observed.

Chapter 6

Conclusion and Future Work

EH can't be used as outliers or open-set samples. Energy separation can be made between samples which have different distributions, but EH falls in the same distribution as rest of the samples.

Currently, the next focus is on only 4 class classification. Also another suggestion from Nikhil was to consider only the samples with very low energy for further training. Accuracies would then be expected to increase as probability of choosing the inliers would be quite high in that case. So, that would be one possible way to solve the coinciding distribution problem we are facing.

References

[1] A. Madabhushi and G. Lee, "Image analysis and machine learning in digital pathology: Challenges and opportunities," 2016.

[2] S. Jha and E. J. Topol, "Adapting to artificial intelligence: radiologists and pathologists as information specialists," *Jama*, vol. 316, no. 22, pp. 2353–2354, 2016.

[3] K. Yasaka, H. Akai, A. Kunimatsu, S. Kiryu, and O. Abe, "Deep learning with convolutional neural network in radiology," *Japanese journal of radiology*, vol. 36, no. 4, pp. 257–272, 2018.

[4] D. Karimi, H. Dou, S. K. Warfield, and A. Gholipour, "Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis," *Medical Image Analysis*, vol. 65, p. 101759, 2020.

[5] S. A. Taqi, S. A. Sami, L. B. Sami, and S. A. Zaki, "A review of artifacts in histopathology," *Journal of oral and maxillofacial pathology: JOMFP*, vol. 22, no. 2, p. 279, 2018.

[6] M. Ilse, J. M. Tomczak, and M. Welling, "Attention-based deep multiple instance learning," *arXiv preprint arXiv:1802.04712*, 2018.

[7] M. Li, L. Wu, A. Wiliem, K. Zhao, T. Zhang, and B. Lovell, "Deep instance-level hard negative mining model for histopathology images," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 514–522, Springer, 2019.

[8] S. Latif, R. Rana, S. Khalifa, R. Jurdak, J. Epps, and B. W. Schuller, "Multi-task semi-supervised adversarial autoencoding for speech emotion recognition," *IEEE Transactions on Affective Computing*, 2020.