Robustness against label noise in semi-supervised models by applying self-supervision

Abhigyan Ganguly
IIIT-D(2020011)
abhigyan20011@iiitd.ac.in

Aman Sharma
IIIT-D(2020021)
hrishav20367@iiitd.ac.in

Mann Khatri
mannk@iiitd.ac.in

1. Problem Definition

Majority of the data generated in the real world is unsupervised. Practical usage of unsupervised data requires human annotations to convert, if not all, some portions of the corpus to labelled data. These annotations are not only costly, but also prone to errors and misclassifications also known as **label noise**.
Semi-supervised algorithms use a small set of labelled data to generate high confidence predictions, which are then used as **pseudo-labels** for the next batch.
Unfortunately, these pseudo-labels when used with label noise, decrease model-performance exponentially.

2. Related Works

Self-supervised learning is a method of training machine learning models using large amounts of unlabeled data and a set of predefined techniques to generate labels for the data. The goal is to learn useful features and representations of the data that can then be used in downstream tasks. One approach to self-supervised learning is to use techniques such as rotation and exemplar-based learning, which involve generating new data by applying transformations to the original data and training a model to recognize the original data from the transformed data. These techniques can be used to improve the performance of semi-supervised learning (SSL) models, which are trained on a combination of labeled and unlabeled data. S4L(Self-Supervised Semi-Supervised Learning) is the paper that aims at improving SSL model performances by pretraining the model using self supervision techniques (S4L Rotation and S4L Exemplar). The S4L (Self-Supervised Semi-Supervised Learning) paper proposes using these self-supervised techniques to pretrain a model, which can then be fine-tuned on a smaller amount of labeled data for a specific task. This approach can help improve the performance of SSL models, especially when there is a limited amount of labeled data available.

The key differences bewteen S4L and our model is that the Self supervision techniques are used only for pre-training, while we use it in every epoch of our training. Our paper is specifically focused on improving robustness against label noise while S4L paper focused on improving model performance.

3. Methodology

Unlike other frameworks combining the realms of Semi-supervised learning and Self-supervised learning such as **S4L** and **Self-Match**, our first algorithm, RobMix, performs performs self-supervised contrastive training along with semi-supervised MixMatch for every epoch. (see Fig 1.1)

To achieve maximum efficiency, the contrastive training and MixMatch are done in parallel with a two-headed meta-learning architecture (as seen in Fig 1.2)

[t]

3.1. Model training and Meta-architecture

3.2. Hypothesis

For each epoch, the common backbone (Black-Box Net as shown in figure) relays the forward pass to both heads of the model.
The loss total loss sent back to the main branch of the network is a linear combination of both losses:
$L_{Total} = L_1 + \lambda * L_2$
where L_1 is the self supervised loss, while L_2 is the semi-supervised loss from MixMatch.

Minimising this combined loss, ultimately trains both, semi-supervised branch and the self-supervised.

4. Experimental Results and Analysis

4.1. Noisy labels

The following are results after running the model with varied levels of artificially induced label noise on datasets

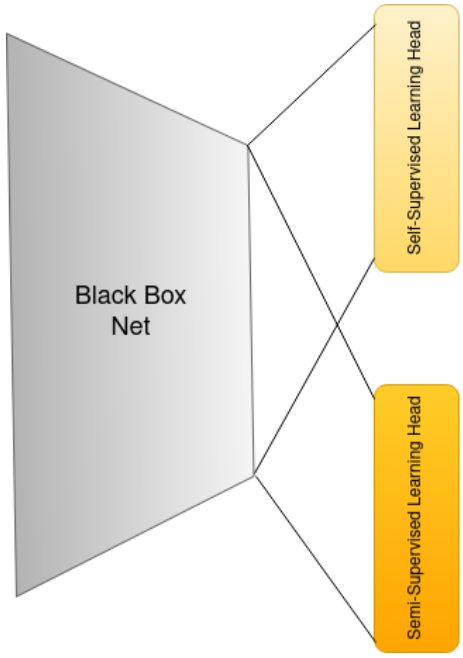
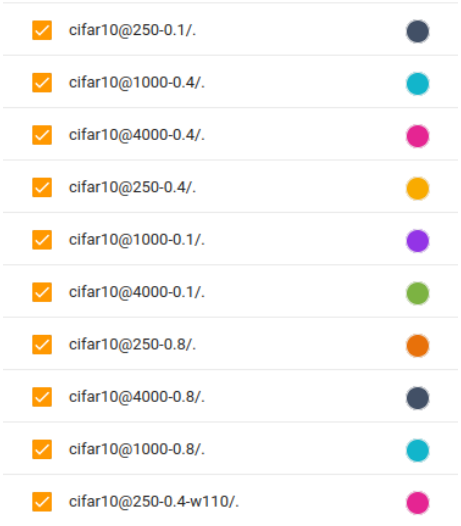
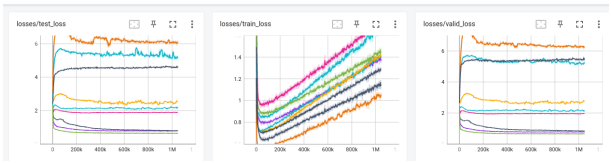


Figure 1. A two-headed meta-learning structure.

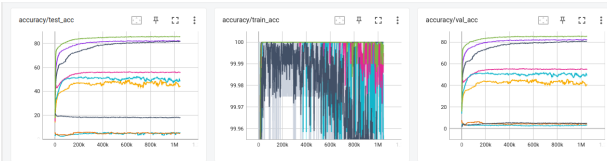
with 250, 1000 and 4000 labels



The losses for the same are as follows:



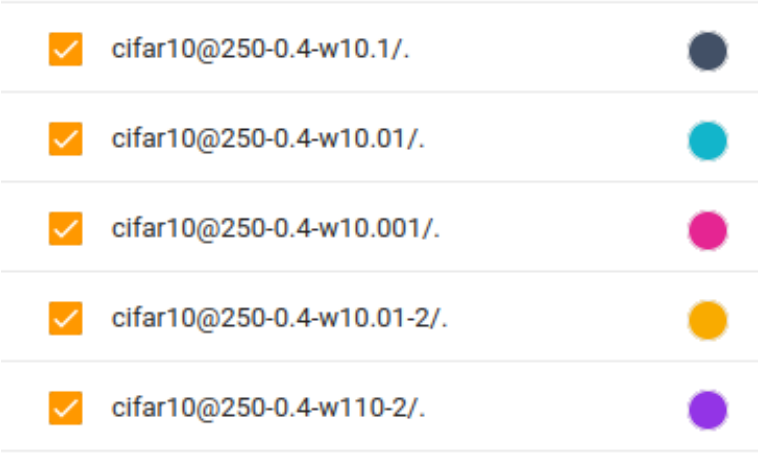
Their accuracies:

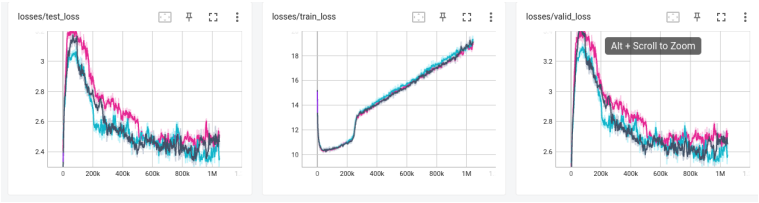


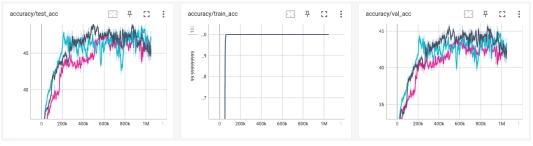
As expected, label noise plays a direct role on model performance. Model gives best performance with 10% label noise and 4000 labels. Another thing to take away from this experiment is that label noise is also directly related to the number of labels in the data. This makes sense as more labels more options for misclassification, hence a more varied label noise.

4.2. No gradient update in Self-supervised branch

For this experiment, we stop the updation of weights in the self-supervised projection head.



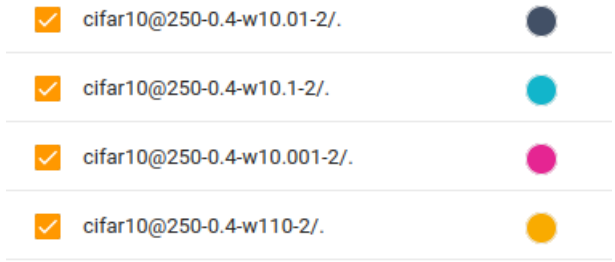


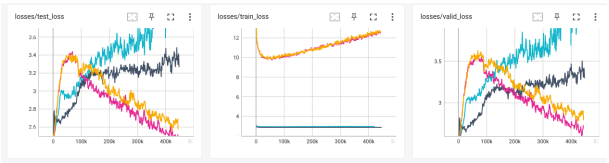


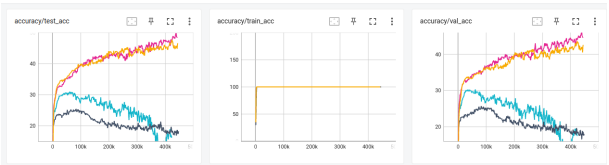
As we can see, even after stopping updation of weights in the self-supervised projection heads, the loss and accuracy graphs keep up the general trend, thus indicating that the model continues to learn.

4.3. Changing Gradient

For this experiment, we modify the gradient of self-supervised projection head.



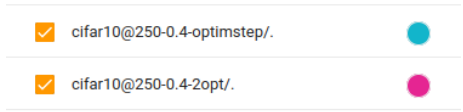




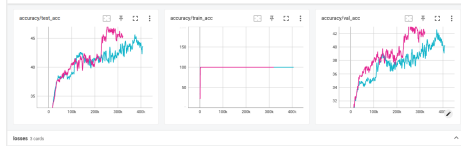
4.4. Optimiser update

For this experiment, we update the optimizers for both projection heads.

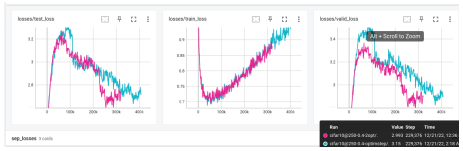
The following experiments have been done on a 250-label dataset with 40% added label noise



The model performs slightly better overall when both the optimizers get updated.Updating the optimizers of both heads can potentially improve the performance of the overall architecture because it allows each head to adapt to its specific task or subproblem more effectively. This can also be inferred from the loss and accuracy plots below.



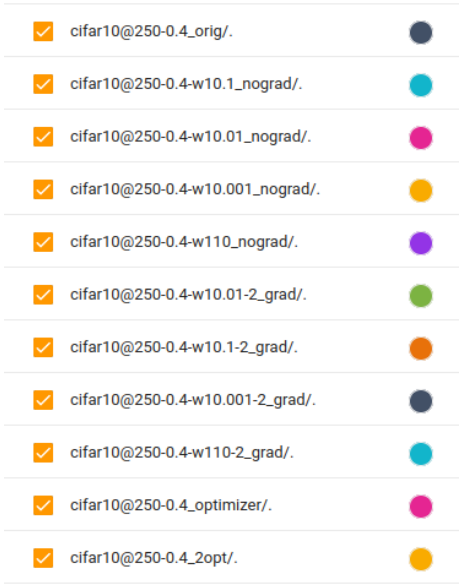
The following are the losses of the same:

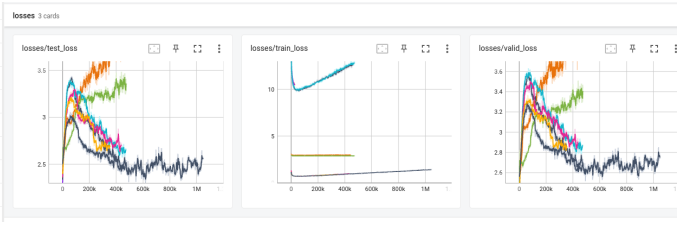


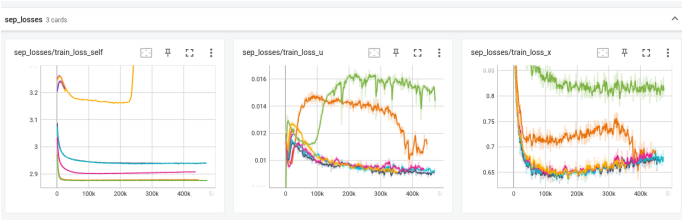
It is worth noting that this trend is not followed in the original RobMix architecture (the one without two heads). This is because, there is no reason to sync gradients here.

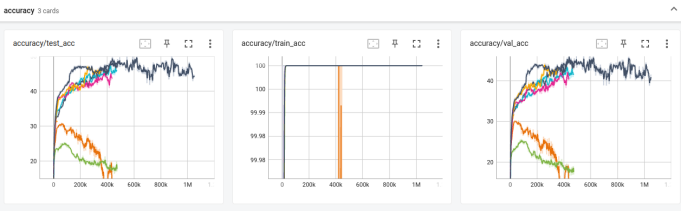
4.5. Ablation Analysis

Here, we have performed a combined observation of the ablation experiments done above.









pseudocode is given below:

Algorithm 1 RobMix

Require: $D_L : \{(x_N, y_N)\}$
Require: $D_U : \{(x_M)\}$
Require: $R_N : WideResNet(pretrained)$
  for epochs **do**
    $R_N \leftarrow SimCLR(R_N, \{x_L\} \cup \{x_M\})$
    `Change projection head of the model`
    $R_N \leftarrow MixMatch(R_N, \{x_N, y_N\}, \{x_M\})$
    `Change projection head of the model`
  end for

Algorithm 2 RobMix: Meta

Require: $D_L : \{(x_N, y_N)\}$
Require: $D_U : \{(x_M)\}$
Require: $R_N : WideResNet(pretrained)$
  for epochs **do**
    $R_N, L_1 \leftarrow SimCLRHead(R_N, \{x_L\} \cup \{x_M\})$
    $R_N, L_2 \leftarrow MixMatchHead(R_N, \{(x_N, y_N)\}, \{x_M\})$
    $L \leftarrow Combined(L_1, L_2)$
    `BackPropagate on` R_N
  end for

5. Individual Contribution

Mann – Built and ran the meta model by adding various optimizers and gradient methods.

Abhigyan – Built and ran the model in which first self learning is done then semi-supervised learning is done and made the report.

Aman – Built and ran the model in which first self learning is done then semi-supervised learning is done and made the report.