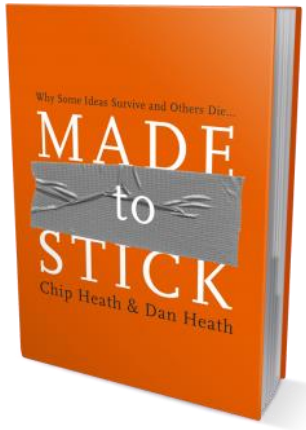
MixMatch - an SSL implementation

Team ~
- Kapil Rajesh Kavitha
- Harshavardhan P

Intro

- Semi-supervised learning
- Prevailing approaches in semi-supervised learning:
  - Entropy minimization
  - Consistency regularization
  - Generic regularization



MixMatch

- Incorporates other approaches
- MixMatch operates on a batch X comprising labeled examples alongside an equally sized unlabeled batch U.
MixMatch
- MixMatch generates a processed batch of augmented labeled examples X' and a batch of augmented unlabeled examples with "guessed" labels, denoted as U'. Subsequently, both U' and X' are utilized in the computation of distinct loss terms for labeled and unlabeled instances.
- Combined loss L:

$$\mathcal{X}', \mathcal{U}' = \text{MixMatch}(\mathcal{X}, \mathcal{U}, T, K, \alpha)$$

$$\mathcal{L}_{\mathcal{X}} = \frac{1}{|\mathcal{X}'|} \sum_{x,p \in \mathcal{X}'} \text{H}(p, \text{p}_{\text{model}}(y \mid x; \theta))$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{L|\mathcal{U}'|} \sum_{u,q \in \mathcal{U}'} \|q - \text{p}_{\text{model}}(y \mid u; \theta)\|_2^2$$

$$\mathcal{L} = \mathcal{L}_{\mathcal{X}} + \lambda_{\mathcal{U}} \mathcal{L}_{\mathcal{U}}$$

Supporting techniques

- **Data augmentation** on both labeled and unlabeled data
- **Label guessing for generating “guessed labels”**
- **Sharpening to reduce entropy of distribution**

$$\bar{q}_b = \frac{1}{K} \sum_{k=1}^{K} \mathrm{p}_{\mathrm{model}}(y \mid \hat{u}_{b,k}; \theta)$$

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

MixUp

- :Both labeled examples and unlabeled examples are mixed with label guesses.
- For a pair of two examples with corresponding label probabilities (x1, p1) and (x2, p2), (x', p') is computed using

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

  - To apply MixUp:

$$\hat{\mathcal{X}} = \left((\hat{x}_b, p_b); b \in (1, \ldots, B)\right)$$
$$\hat{\mathcal{U}} = \left((\hat{u}_{b,k}, q_b); b \in (1, \ldots, B), k \in (1, \ldots, K)\right)$$

Implementation details

Data Augmentations

The following data augmentations were implemented:

- Random Pad and Crop: takes an input image ‘x’, applies random padding of 4 pixels to all sides of the image, then randomly crops it to the specified output size.
- Random Horizontal Flip: performs a horizontal flip on the input image with a 50% probability.
- Gaussian Noise: takes an input image x and adds Gaussian noise to it. Random samples from a standard normal distribution for each channel, and these samples are multiplied by 0.15 to control the intensity of the noise.
- Resize Image: Resizes the image to (3, 32, 32) which is an acceptable size for the model.

Implementation details

Model used:
The model used was a WideResNet model with depth 28. The model was obtained from the implementation used by Google.

Optimizer

The Adam optimizer was used for the WideResNet model, as it has the ability to decay the weights - which has been mentioned in the Google implementation.

Exponential Moving Average (EMA) Optimizer

An EMA optimizer was also implemented as part of our code. Another model was created on which EMA was applied.
It is a technique used to smooth out noisy data and highlight trends over time, and is often applied to the parameters of a neural network during training. The purpose is to maintain a moving average of the model's parameters to provide a more stable and less noisy estimate. It creates a more stable version of the model, and makes the model more robust.

Hyperparameters and Tuning

Hyperparameters:

- Temperature (T) : 0.5
- Number of augmentations for unlabeled data (K) : 2
- Unsupervised loss weight (λu) : 0.75
- Weight decay : 0.0008

Tuning

Tuning was performed on the number of labeled samples in each dataset, and the model performance was noted for different number of labeled samples.

Experimentation

Experimentation

Datasets in paper

- **CIFAR10**
- **CIFAR100**
- **SVHN**
- **STL10 (extra and no extra)**

Additional datasets

- **Fashion-MNIST**
- **MNIST**

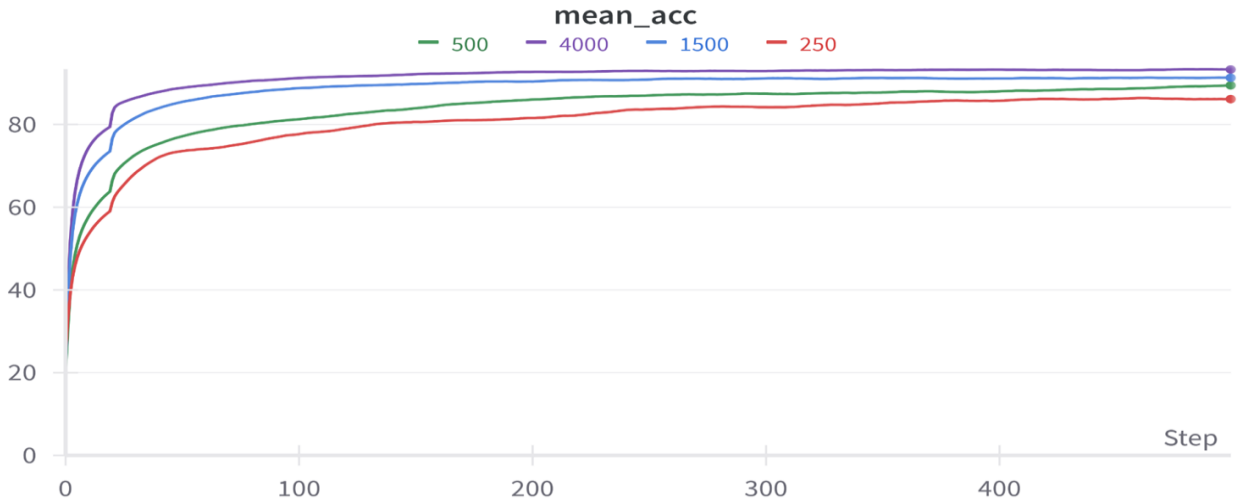
LINK FOR ALL WANDB PROJECTS:

harsha_cvit – Weights & Biases (wandb.ai)

CIFAR-10

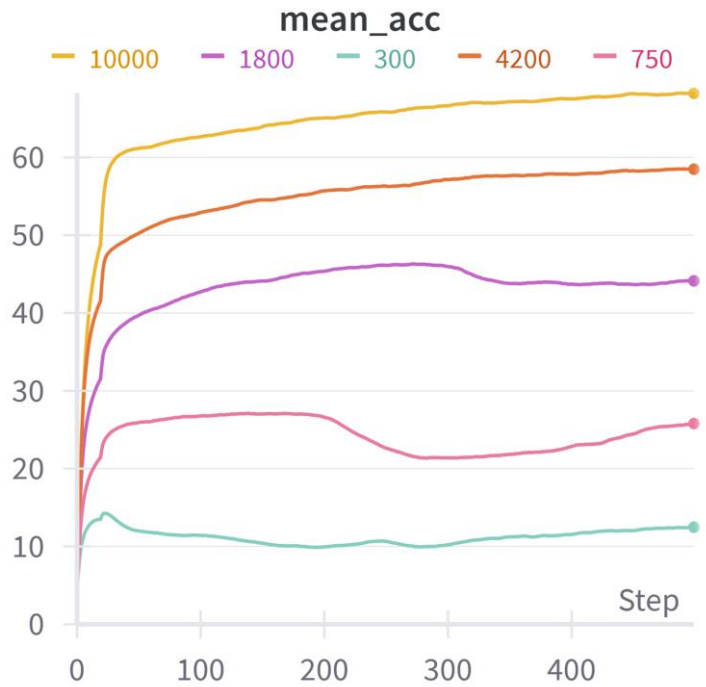
CIFAR-10 was run for 4 values of number of labeled data points (250, 500, 1500, 4000)

We observe that the performance of the model increases as the number of labeled examples increases for the given data, but the model performs sufficiently well for all 4 values.



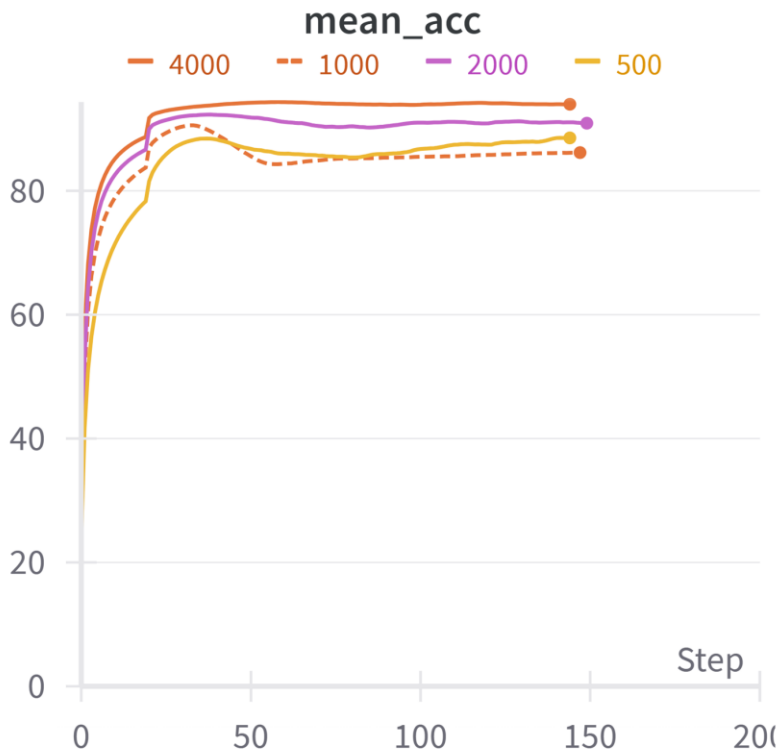
CIFAR-100

CIFAR-100 was run for 5 values of labeled data (300, 750, 1800, 4200, 10000). The best performance was observed for the models with 4200 and 10000 labels. Due to the higher possibility of incorrect labels which would affect the performance of a semi-supervised model, the model's performance was not substantially improved by the MixMatch approach for 300 labels.



SVHN - no extra set

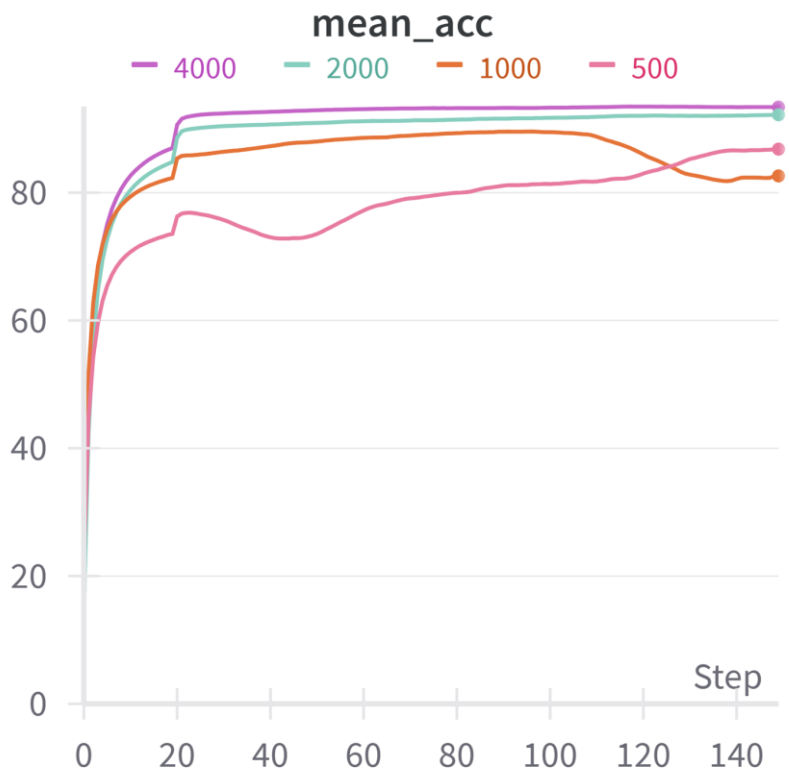
All 4 runs showed good performances, with accuracies ranging from ~85% to ~93% for the four of them. The best performance was shown when 4000 labeled data points are passed to the model.



SVHN - with extra set

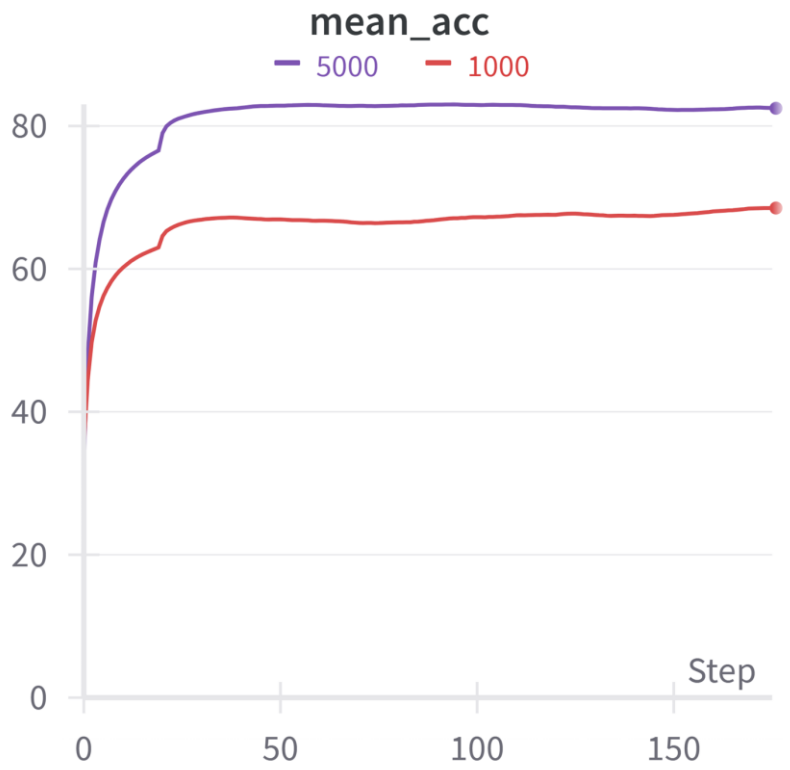
With extra set, due to the much larger number of unlabeled samples, we modify the model input parameters: we set the mixing ratio (alpha) as 0.25 and lower weight decay to 0.000002 to account for the larger amount of available data.

Here the performance of the model when 2000 and 4000 labeled inputs are passed stands out even further, as its performance is substantially better than the model when lesser number of labeled inputs are passed.

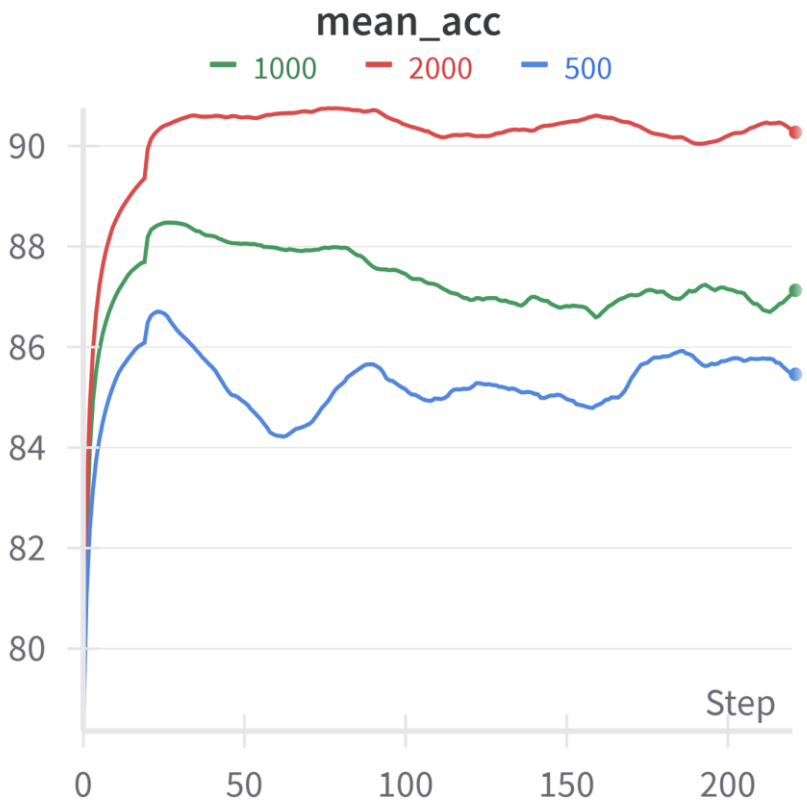


STL10

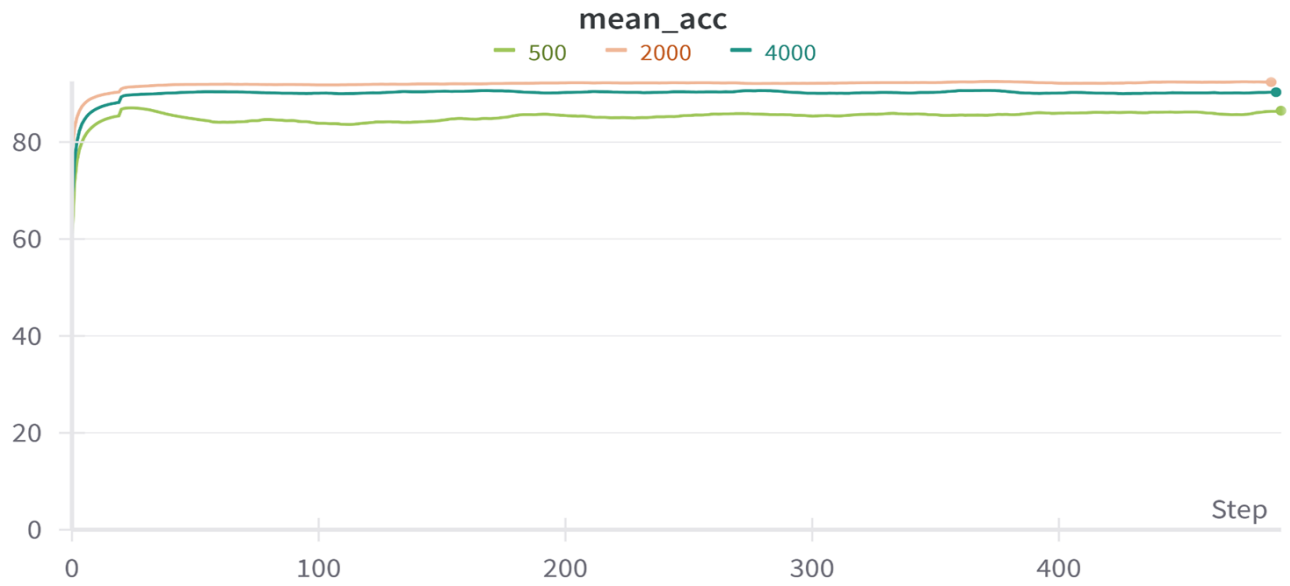
The performance of our MixMatch implementation on the STL-10 dataset falls short of the performance of the Google paper. This is because of the need to resize the 96x96 input images to 32x32 so that they fit in our model, due to the inability of being able to compute with a larger number of parameters in our WideResNet model.



Additional : MNIST



Additional : Fashion-MNIST

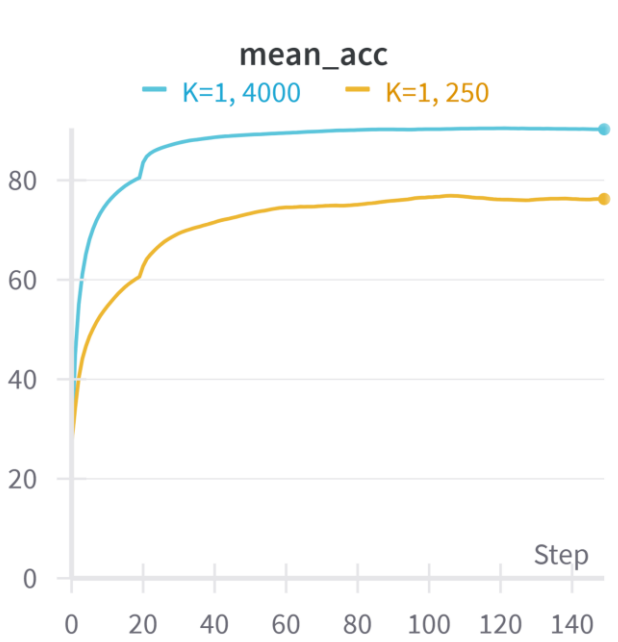


Ablation Studies

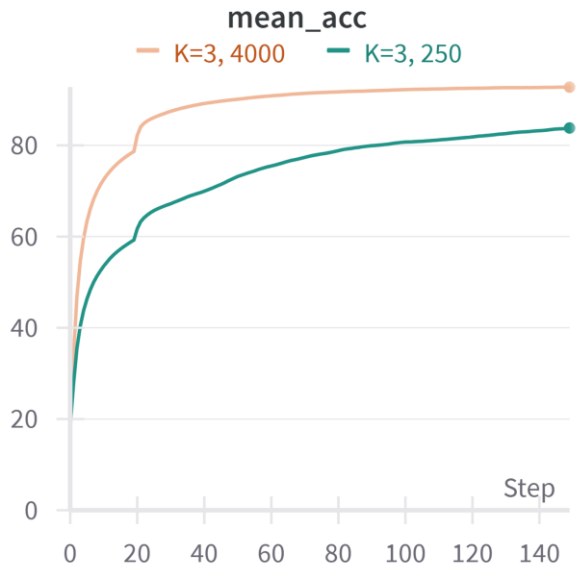
In this section, we aim to study the impact of the different parts of the MixMatch algorithm on its success. All of these experiments were run on the CIFAR-10 dataset.

Number of augmentations

K = 1

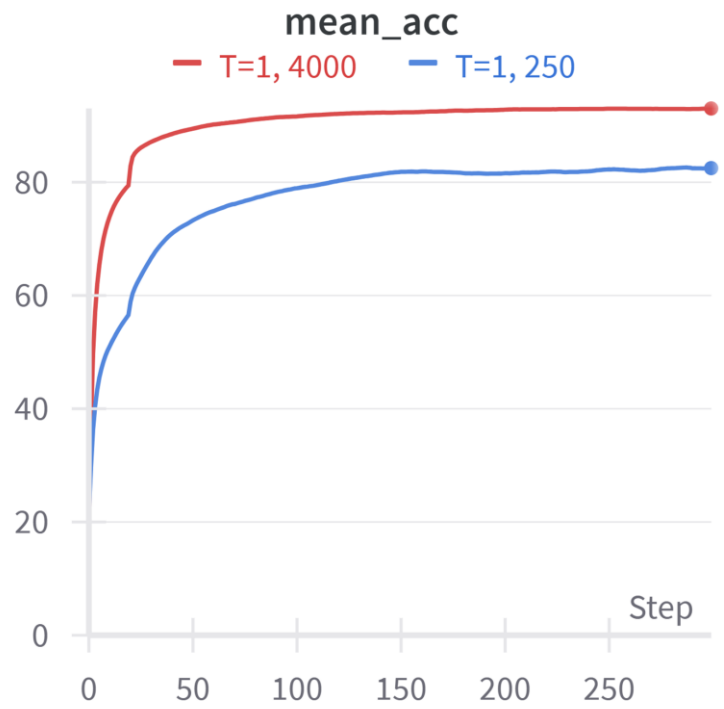


K = 3



Ablation Studies

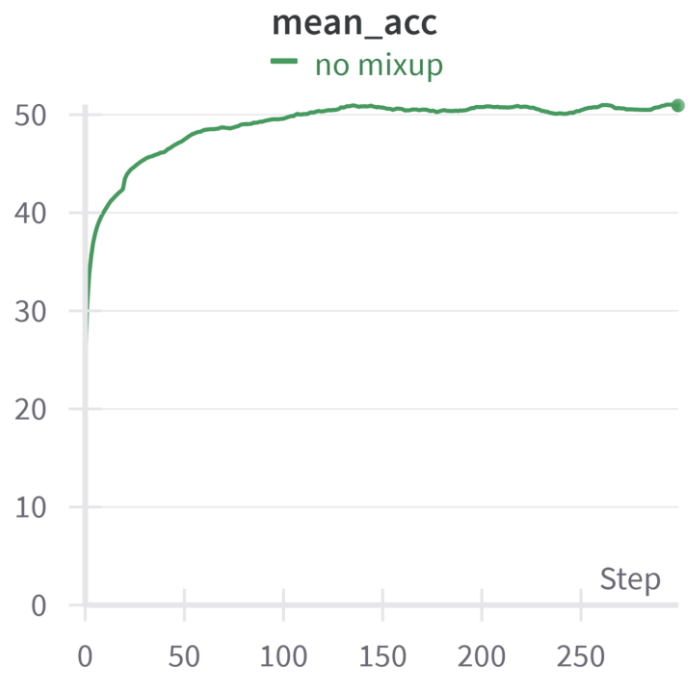
Temperature (T) = 1
The impact of T=1 appears to be more significant when the number of labels is lesser, and it becomes less significant on the increase in the number of labels, as the model has more amount of reliable information to learn from.



Ablation Studies

MixMatch with no MixUp

This experiment was performed on 250 labels. The impact of no MixUp on the MixMatch algorithm is very significant as there is a drop in accuracy of above 30%, as the training process may have become much less stable.



Comparison

In the cases of CIFAR10, CIFAR100 and SVHN, our implementation of MixMatch performs comparatively well to the Google implementation, with a ~4-5% difference in error rates.
Only the STL-10 dataset saw a dip in performance. This is due to usage of WideResNet models with different depths, and the need to resize images in the STL-10 dataset to fit such a model.

The MNIST and Fashion-MNIST models seemed to display performances relatively close to the benchmark scores, indicating the strength of the approach.

Issues faced

- Lack of compute and relatively high running time, which also was the reason for not being able to perform hyperparameter tuning significantly.
- The lack of compute gave us an issue with not being able to run models with more than 1.4M parameters, so we couldn’t train the CIFAR-100 dataset on a larger model or efficiently train the STL-10 dataset.

THANK YOU