**1. Abstract**

Getting labeled data is a difficult and expensive task, which is the reason why Semi-Supervised Machine Learning (SSML) has been gaining more attention in the field of Artificial Intelligence. In this project, we focus on implementing and testing a promising interpolation consistency poisoning attack on the unlabeled dataset of SSML models. Given 2 images x, x\* with different true labels y and y\*, x is labeled and x\* is unlabeled, the attack attempts to make SSML models wrongly classify x\* as y classify by injecting a set of N samples to the unlabeled dataset to “connect” x and x\*. The three SSML models that we are testing this poisoning attack on are: Fixmatch, Mixmatch, and Pseudo-Labeling. The three datasets we are using are CIFAR10, Food101, and iNaturalist. So far, none of the attack attempts we tried were successful, but we have made some very interesting observations about the behavior of the attack relative to different attack configurations. Some of the most significant observations are: when the model’s performance is bad, the attacks also seem to perform badly and randomly, when the labeled set is relatively large (5% of the entire training set and up), the attacks seem to perform worse than when the labeled set is small (0.2% of the entire training set and below).

**2. Introduction [TODO]**

Supervised learning in machine learning area will be less efficient when there exists large amount of data losing labels, that’s why the Semi-Supervised Machine Learning (SSML) becomes so popular nowadays. Recently, there are a lot of semi-supervised models applied on unlabeled datasets, for example, Fixmatch, Mixmatch, UDA… and researchers are trying to increase the model performance. When paying more attention on the model’s accuracy rate, in the meanwhile, we should also care about the robustness of the SSML models, which is also crucial.

In this project, we focus on exclusively poison the unlabeled dataset applying SSML models, which will be powerful compared with simply adding additional human review to the unlabeled dataset. Once there are two different images, one is with label and the other one is without label, we will try to apply different attack strategies to inject the original dataset. And then apply use different models (Fixmatch, Mixmatch, Pseudo-Labeling) to test for the performance of the poisoning attack under three datasets (CIFAR10, Food101, and iNaturalist).

What we want to do next is to try different poison size to change the original dataset in order to maximize attack success rate, and change different attack strategies to test the performance on all three models, and try to combine the model and test something else.

**3. Model (technical preliminaries)**

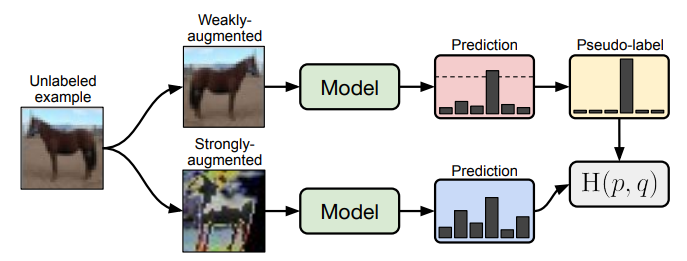
3.1 Interpolation consistency poisoning [TODO]

* [TODO] Talk about how neural networks are Lipschitz-continuous with a low constant, etc.
* [TODO] Talk about how each poison sample is a superposition of x and x\* of different scale based on the parameter alpha and the density function.

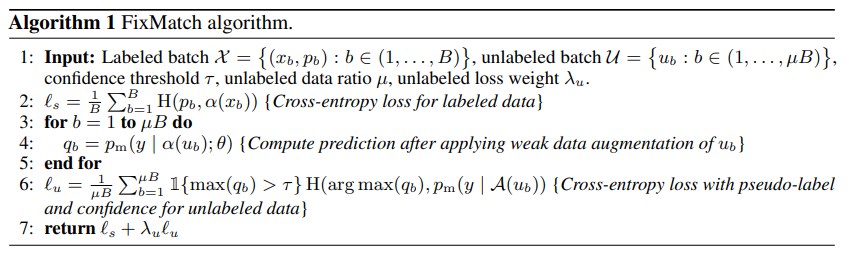
3.2 Fixmatch

The main intuition behind the Fixmatch algorithm is to train labeled and unlabeled samples concurrently while augmenting each unlabeled sample X both weakly (creating X’) and strongly (creating X’’) with the intention to use the predicted label of X’ as the true label of X’’ to penalize wrong prediction of X’’ when the confidence of the predicted label of X’ is high enough. Note that even the labeled samples are weakly augmented and labeled samples and unlabeled samples have different cost functions, and a strongly augmented unlabeled sample is only included in the cost function when the predicted label of its weakly augmented version has a high enough confidence.

The diagram below shows how an unlabeled sample is transformed and included in the cost function.



The pseudo-code below describes how the FixMatch algorithm works.



3.3 Mixmatch [TODO]

3.4 Pseudo-Labeling

Pseudo-Labeling works by applying pseudo-labels to samples in the unlabeled set by using a model trained on the combination of the labeled samples and any previously pseudo-labeled samples and iteratively repeating this process in a self-training cycle.

The main process are as below: (1)Trained on the labeled samples. (2) Use trained model to predict the unlabeled samples. (3) Combine the labeled and unlabeled samples, and then retrain the model with the combination sampleds.

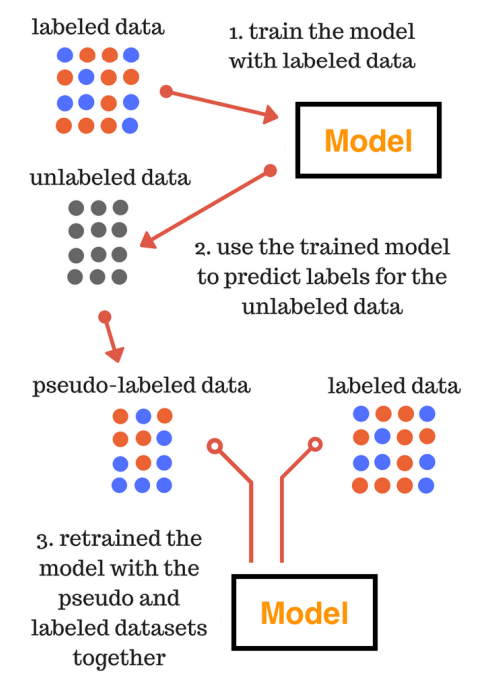
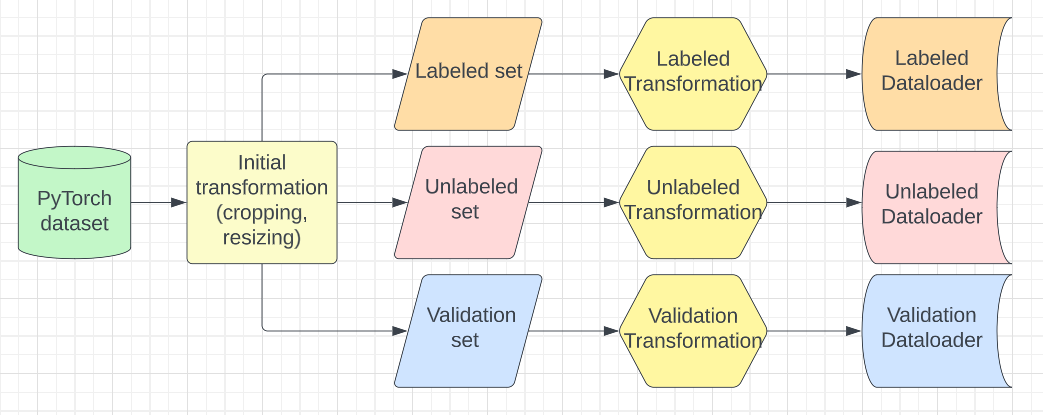


Photo source: <https://datawhatnow.com/pseudo-labeling-semi-supervised-learning/>

**4. Algorithms or Implementation (details of implementation)**

4.1 Data Pipeline

Since the time we have for this project and the amount of computational power we have access to are limited, we started by implementing a data pipeline that can get a subset of a PyTorch dataset and split it into 3 datasets: unlabeled dataset, labeled dataset, and test set. Each of the three mentioned datasets was created such that they contain an equal number of sample classes to avoid bias during training. Also note that although all 3 datasets that we use (CIFAR10, Food101, and iNaturalist) have a label for every image, we disregard the labels of images in the “Unlabeled set” to simulate a semi-supervised learning situation.

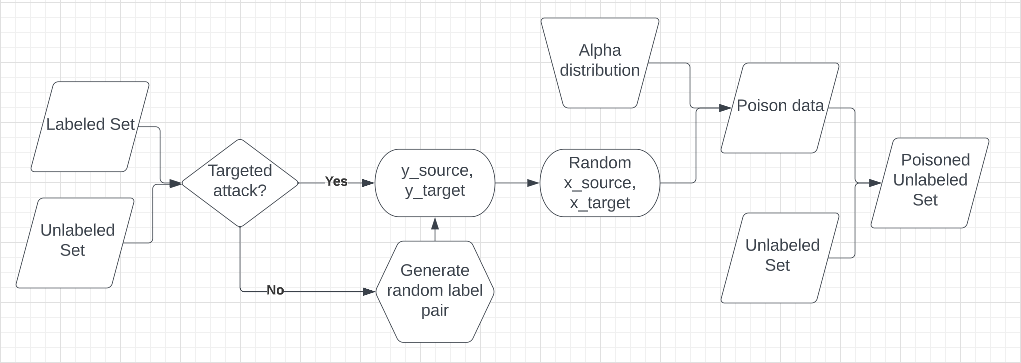


Note that since we use three learning methods (Fixmatch, Mixmatch, and Pseudo-Labeling), each has different transformations needed for the algorithm to work properly, and poison must be added to the unlabeled dataset before transformation, we separated the transformations from the labeled, unlabeled, and test set splitting and applying them to our data loader at the time of training, after the poison samples are injected.

4.2 Interpolation consistency poisoning algorithm

Utilizing OOP, we implemented a class called “Poisoner” that can take a labeled (Pytorch) dataset and an unlabeled (Pytorch) dataset along with other configurational parameters (number of poison samples, target or untargeted attack, etc.) as inputs and return a poison dataset containing the poison samples obtained using the interpolation consistency method mentioned in the previous section. This poison dataset can then be concatenated with the unlabeled dataset at the time of training to create a poisoned unlabeled set.

The overall flow of our implementation can be visualized in the diagram below:



Below is an example of poison images generated. Notice that the image with the dog (white) is the source image and the image with the cat (black) is the target image. The higher the alpha values, the more the poison image resembles target image, which in this case is a dog.



4.3 Fixmatch

Since Fixmatch is still a new model, we cannot import it from the official PyTorch library, but had to use an official implementation for PyTorch from [this repository](https://github.com/kekmodel/FixMatch-pytorch). We then modify this implementation of Fixmatch to accept extra parameters related to our poisoning attack, including subset size, poison size, resume training by reloading poison, and the label pairs for a targeted attack option. before it can be ready to be used with our attacks. We also had to modify its input data pipeline to use our custom data loaders (with poison injected) and custom transformations (for the new datasets).

4.4 Mixmatch [TODO]

4.5 Pseudo-Labeling

We use the Pseudo-Labeling

**5. Results/Evaluation**

Since a lot of time was needed to develop a reliable data pipeline, poisoning class, making sure the learning models still work well when the unlabeled dataset is combined with the poison dataset, and training the learning models, we have only managed to test our implementation of the poisoning dataset on the CIFAR10 dataset with Fixmatch as the learning model.

In the table below, we show some basic results of our attack attempts with different parameter configurations using Fixmatch as the learning model and CIFAR10 as the base dataset. Note that the “target image” here is the image x\* we want the model to wrongly classify as the label y of the “source image”. “Desired label’s rank” is the rank of confidence of the label y when trying to classify the “target image” x\*.

| ID | Subset size | Labeled set size | Number of poison samples | Source image label | Target image label | Success | Target’s actual prediction | Desired label’s rank |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 10,000 | 500 | 10 | bird | cat | No | horse | 8 |
| 2 | 10,000 | 500 | 10 | horse | truck | No | cat | 7 |
| 3 | 50,000 | 2,500 | 10 | airplane | dog | No | dog | 9 |
| 4 | 50,000 | 250 | 100 | truck | deer | No | deer | 8 |
| 5 | 50,000 | 100 | 100 | cat | dog | No | dog | 2 |
| 6 | 50,000 | 50 | 100 | cat | dog | No | deer | 3 |

Table 1: Result of 3 attack attempts using Fixmatch as the learning algorithm and CIFAR10 as the base dataset.

Although the results in the table seem fairly disappointing that none of the attack attempts we tried succeeded, we noticed some very interesting observations:

* When the subset of CIFAR10 we use for training is small (around 10,000 samples, experiments id 1 and 2), Fixmatch performance was quite bad (converged around 71% on the validation set). This poor performance also seems to affect the success rate as the predicted label of the target image is just a random label that is neither that of the source nor the target image.
* The attack doesn’t seem to work well when the Labeled Set is large (around 5% of the entire training set, experiments id 3 and 4), as the model’s performance was quite good on the test set (87% and above)
* The attack seems to work better when the Labeled Set is small (around 0.1% - 0.2% of the entire training set, experiments 5 and 6), as although these attacks still didn’t succeed, the desired label’s rank is only second or third. Also, for both experiments 5 and 6, all the injected poison samples of alpha values up to around 0.65 (closer to the target image) are classified as the target’s label, which is what we desire in a successful attack. If we can somehow increase this alpha threshold (to 1.00), we would have a successful attack.
* When the Labeled Set is very small (0.1% of the entire training set, experiment 6), the model’s performance degraded significantly (around 35% accuracy on the validation set), but the attack performance did not seem to degrade much, as described above.

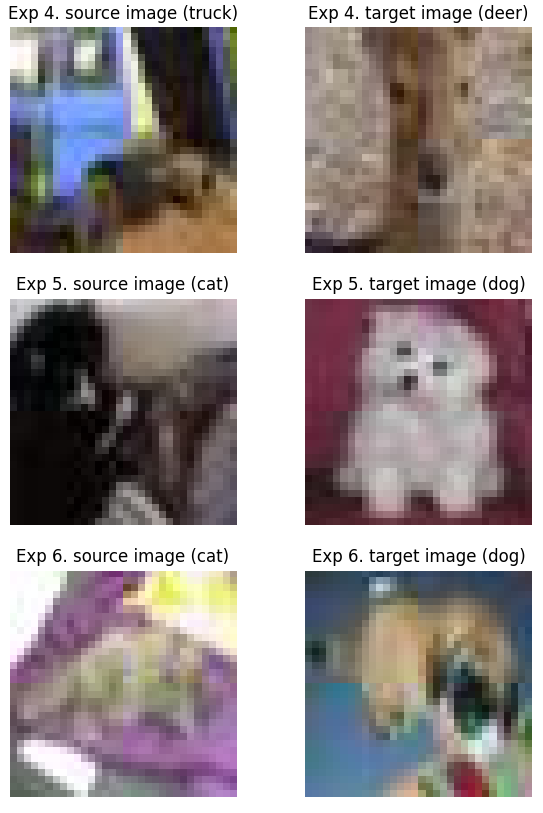


Figure 1: The pairs of source and target images for experiments ID 4 to 6

**6. Related Work [TODO]**

**7. Conclusion, Limitations, and Future Work**

7.1 Conclusion

* We haven’t had a successful attack attempt yet, but some good observations were made regarding the behavior of an attack regarding different parameters such as subset size, number of poison samples, and number of labeled samples.

7.2 Limitations

* Training takes a very long time (3 hours with Colab Pro to train and test each attack attempt for Fixmatch)
* If we use a small subset of the data, the performance of our model degrades significantly, which also reduces the attack success rate. This is a big limitation as going into this project, we know we wouldn’t have enough time nor computational power to carry out anywhere close to the extensive experiments mentioned in the original paper (which they said took hundreds of GPU-days of compute time), so our plan was to use a small subset of each subset without realizing that this would affect the attack success rate.
* [Add more stuff here]

7.3 Future Work

* Further testing optimal poison size and labeled set size to maximize attack success rate.
* Test the attack on Mixmatch and Pseudo-Labeling using CIFAR10 as the base dataset.
* Test the attack on all combinations of the 3 learning models and the 3 datasets.
* [Add more stuff here]

**8. References**

* <https://github.com/kekmodel/FixMatch-pytorch>
* <https://github.com/iBelieveCJM/pseudo_label-pytorch.git>
* <https://www.usenix.org/system/files/sec21-carlini-poisoning.pdf>
* <https://arxiv.org/abs/2001.07685v2> [Cite the diagram and pseudo code in section Models → FixMatch]