Consistency Based Regularization

Consistency

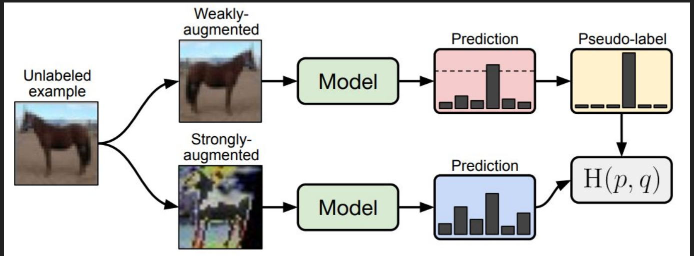
- Through a couple of papers, I’ve learned about consistency based regularization that is being used in many new SSL techniques.
- It utilizes unlabeled data by relying on the assumption that the model should output similar predictions when fed perturbed versions of the same image.
- The loss function can be seen as:

$$\sum_{b=1}^{\mu B} \|p_{\mathrm{m}}(y|\,\alpha(u_b)) - p_{\mathrm{m}}(y|\,\alpha(u_b))\|_2^2$$

- Where u_b is an unlabeled image, p_m is the models prediction, and alpha is a data augmentator.

FixMatch

- One of the newest and simplest SSL techniques, FixMatch, utilizes both consistency regularization and pseudo-labeling.
- For labeled images they use normal cross-entropy loss but for unlabeled images their pipeline is as follows:
    - They weakly augment the image and feed it into the model.
    - The prediction on the weakly augmented image becomes the label for the image.
    - They then strongly augment the image and feed it into the model.
    - They treat this as the prediction on the image and use cross-entropy with the pseudo-label as their loss.



Our Problem

- We are working in the noisy label domain, and thus SSL techniques cannot be directly applied to the data.
- We can, however, treat confident samples as labeled data and unconfident samples as unlabeled data if we were able to split them.
- Inspired by consistency based regularization, I came up with a way to select confident samples.

Consistency based on Similarity

- My idea is to be applied to a batch of samples in order to split them into a labeled and unlabeled set:
  - For a batch, we feed the images (can be MixedUp) into the model and obtain predictions.
  - We group predictions by their labels (Dogs group, Horse group, etc.)
  - For each group, we build a matrix in which we compute the class probability wise difference between the predictions for each pair of samples. (Difference shown next)
  - Each difference will be representative of how different the model believes these two samples are.

Similarity Calculation

Given two softmax probability distributions on two images (example of 4 classes):

s_1 = [0.1, 0.2, 0.5, 0.2] and s_2 = [0.0, 0.4, 0.1, 0.5]

The difference between the two distributions can be calculated as:

diff = abs(s_1 - s_2) = | [0.1, 0.2, 0.5, 0.2] - [0.0, 0.4, 0.1, 0.5] | = [0.1, 0.2, 0.4, 0.3]

The difference score can be computed as:

diff_score = $\sum diff_i$ = 0.1 + 0.2 + 0.4 + 0.3 = 1.0

Difference Meaning

- A low difference means the model predicted similarly on both images while a high difference means the model predicted far apart for each image.
- Using this idea, noisy samples should more often than not cause higher difference values as they will be different than the other images in the group.
- We can select the samples that have the most low difference values as confident, and the rest as unconfident.

Code

```
for batch of samples x,y //let x,y be all images (nx3x32x32), all labels (nx1)
    y_pred = softmax(model(x)) //y_pred is for all images = represents probability for each class
    confident_ind, unconfident_ind = [ ], [ ]
    for class_label in 0,1,2,...,10:
        class_ind = find_ind(y) //find indexes of labels with current class (length m)
        difference_matrix = build_matrix(y_pred, class_ind) //builds mxm matrix of class probability wise differences
        ind_low = find_best(difference_matrix , class_ind) //finds samples that had the most number of low differences
        ind_high = all_other(class_ind, ind_low) //all other indexes not included in low list
        confident_ind.append(ind_low)
        unconfident_ind.append(ind_high)
    return confident_ind, unconfident_ind //splits batch into confident and unconfident set
```

Difference Score Matrix

Image Index:	4	6	8	13	15
4	$\text{diff\_score}_{4,4} = 0$	$\text{diff\_score}_{4,6} = 0.1$	…0.5	…0.2	…0.15
6	…0.1	…0	…0.6	…0.34	…0.7
8	…0.5	…0.6	…0	…0.9	…1.5
13	…0.2	…0.34	…0.9	…0	…0.24
15	…0.15	…0.7	…1.5	…0.24	…0
Count <0.3: threshold hyperparameter	**3 (Most Confident)**	1 (Less Confident)	0 (Least Confident)	**2 (More Confident)**	**2 (More Confident)**

Pipeline Afterwards

- A general process I thought of:
  - Apply confident/unconfident split on a batch of samples and treat the confident as labeled and unconfident as unlabeled.
  - Use normal cross-entropy on the labeled samples.
  - Use FixMatch on the unlabeled samples:
    - Generate a pseudo-label using a weakly augmented version of the image.
    - Make a prediction using a strongly augmented version of the image.
    - Calculate cross entropy between prediction and pseudo-label.
- This applies consistency based regularization twice to the data:
  - Once to split the samples, and once to train on unlabeled samples.
- Will try implementing this over the next week.

Updates

Update To Method

- In implementation, I realized we do not need to keep track of the matrix pair scores and only need each samples count of consistency.

Image Index:	4	6	8	13	15
4	diff_score$_{4,4}$ = 0	diff_score$_{4,6}$ = 0.1	…0.5	…0.2	…0.15
6	…0.1	…0	…0.6	…0.34	…0.7
8	…0.5	…0.6	…0	…0.9	…1.5
13	…0.2	…0.34	…0.9	…0	…0.24
15	…0.15	…0.7	…1.5	…0.24	…0
Count <0.3: threshold hyperparameter	**3 (Most Confident)**	1 (Less Confident)	0 (Least Confident)	**2 (More Confident)**	**2 (More Confident)**

Image Index:	4	6	8	13	15
Count <0.3: threshold hyperparameter	3	1	0	2	2

Three Ways of Selecting Samples

- I needed a metric to decide how to split the confident samples from the unconfident samples.
- The three I came up with were:
  - The sum of confidences for a sample.
  - The average of confidences for a sample. (Realized later on this is the same as sum)
  - The number of times a score was under a threshold for a sample.

Methods: Base Algorithm

```
for batch of samples x,y //let x,y be all images (nx3x32x32), all labels (nx1)
    y_pred = softmax(model(x)) //y_pred is for all images = represents probability for each class
    confident_ind, unconfident_ind = [ ], [ ]
    for class_label in 0,1,2,...,10:
        class_ind = find_ind(y) //find indexes of labels with current class (length m)
        diff_scores = metric(y_pred[class_ind]) //calculates scores using a metric (3 different kinds)
        diff_avg_of_all = mean(diff_scores) //finds average of all scores for samples in class i
        ind_low = get_low(diff_scores, diff_avg_of_all) //finds samples that had their diff_score < diff_avg_of_all
        ind_high = get_high(diff_scores, diff_avg_of_all) //finds samples that had their diff_score >= diff_avg_of_all
        confident_ind.append(ind_low)
        unconfident_ind.append(ind_high)
    return confident_ind, unconfident_ind //splits batch into confident and unconfident set
```

Methods: Sum Based Metric (Given y_pred_c)

```
confidence = [0, …, 0] //zero initially for length y_pred_c
for i in range(len(y_pred_c)):
    for j in range(len(y_pred_c)):
            score = sum(absolute(subtract(y_pred_c[i], y_pred_c[j])))
            confidence[j] += score //simply add score to corresponding column
return confidence
```

Methods: Average Based Metric (Given y_pred_c)

```
confidence = [0, …, 0] //zero initially for length y_pred_c
for i in range(len(y_pred_c)):
    for j in range(len(y_pred_c)):
            score = sum(absolute(subtract(y_pred_c[i], y_pred_c[j])))
            confidence[j] += score //simply add score to corresponding column
confidence = confidence / len(y_pred_c) //divide each sum by length to average
return confidence
```

Methods: Threshold Based Metric (Given y_pred_c and thresh)

```
confidence = [0, …, 0] //zero initially for length y_pred_c
for i in range(len(y_pred_c)):
    for j in range(len(y_pred_c)):
            score = sum(absolute(subtract(y_pred_c[i], y_pred_c[j])))
            if score < thresh:
                confidence[j] += 1 //add 1 if under threshold
return confidence
```

Preliminary Results

- I wanted to test out these three metrics to find what would be best.
- So I trained using cross entropy normally, found the split between confident and unconfident using each metric, and found how much noise was present in each side of the split.

Sum Metric Results (10% Noise Total)

Epoch:	0	10	25	50	75	100
Number Noisy/Total Confident	2876/35231	557/34512	436/36017	393/36429	1003/40061	1523/40553
Number Noisy/Total Unconfident	1629/14769	3948/15488	4069/13983	4112/13571	3497/9939	2982/9447

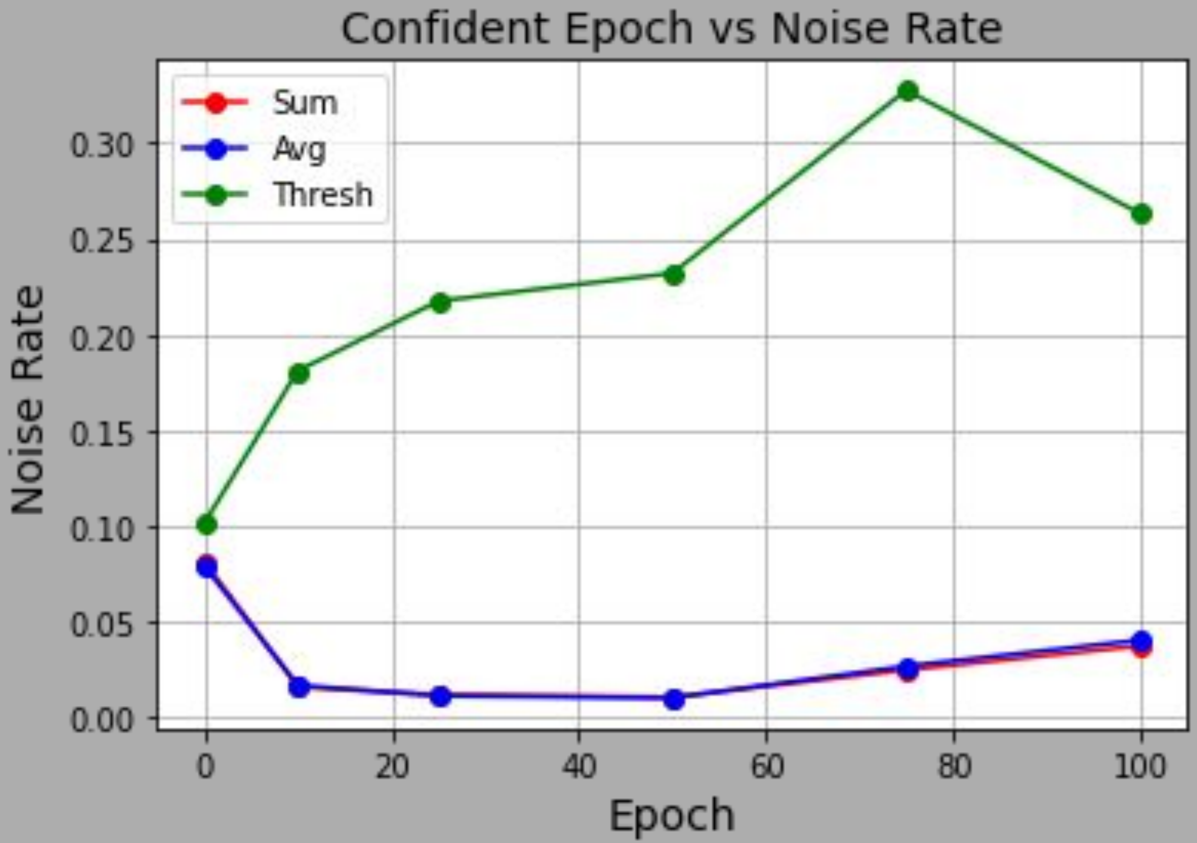
Average Metric Results (10% Noise Total)

Epoch:	0	10	25	50	75	100
Number Noisy/Total Confident	2701/34052	583/34358	413/35925	380/36430	1085/40233	1657/40824
Number Noisy/Total Unconfident	1804/15948	3922/15643	4092/14075	4125/13570	3420/9767	2848/9176

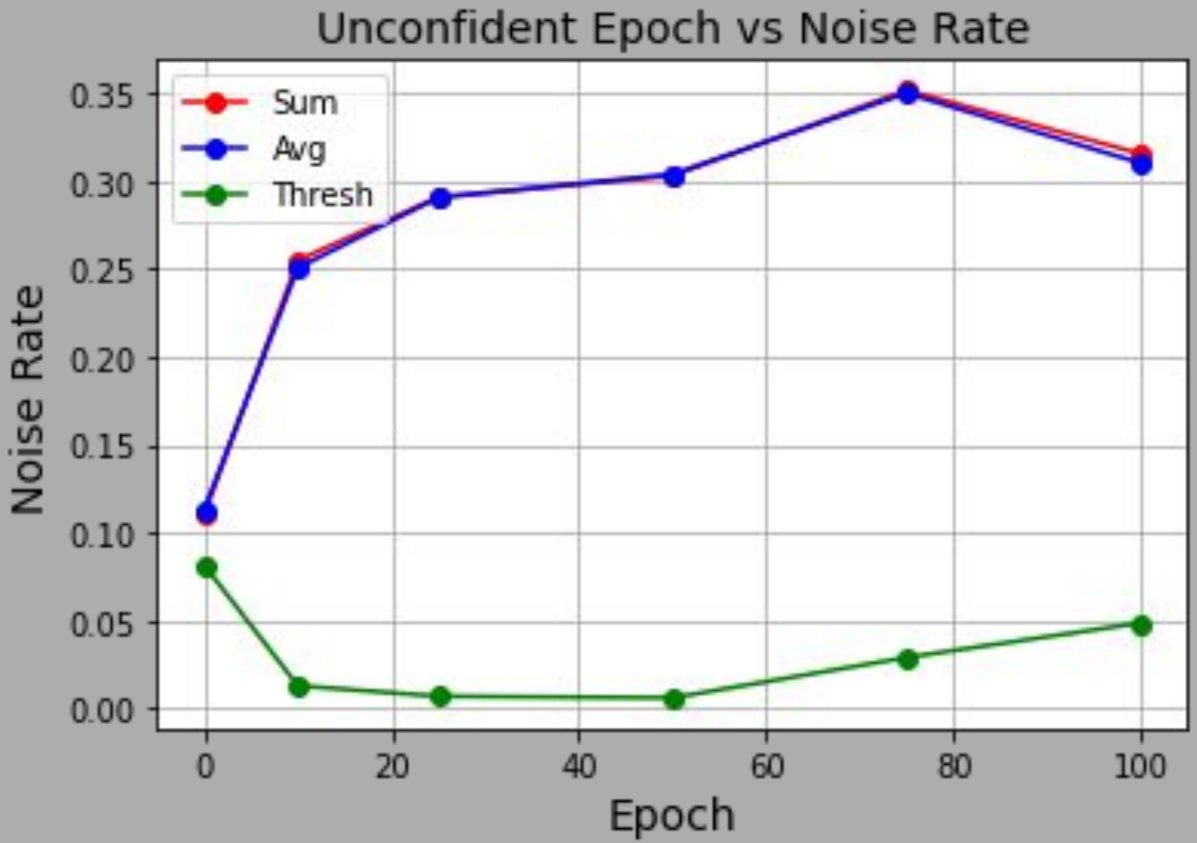
Threshold Metric Results (10% Noise Total) (threshold = 0.3)

Epoch:	0	10	25	50	75	100
Number Noisy/Total Confident	2175/21402	4159/22973	4299/19775	4320/18628	3358/10257	2549/9692
Number Noisy/Total Unconfident	2330/28598	346/27027	206/30225	185/31372	1147/39743	1956/40308

Visual


Confident Epoch vs Noise Rate
Sum
Avg
Thresh
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Noise Rate
0
20
40
60
80
100
Epoch

Visual



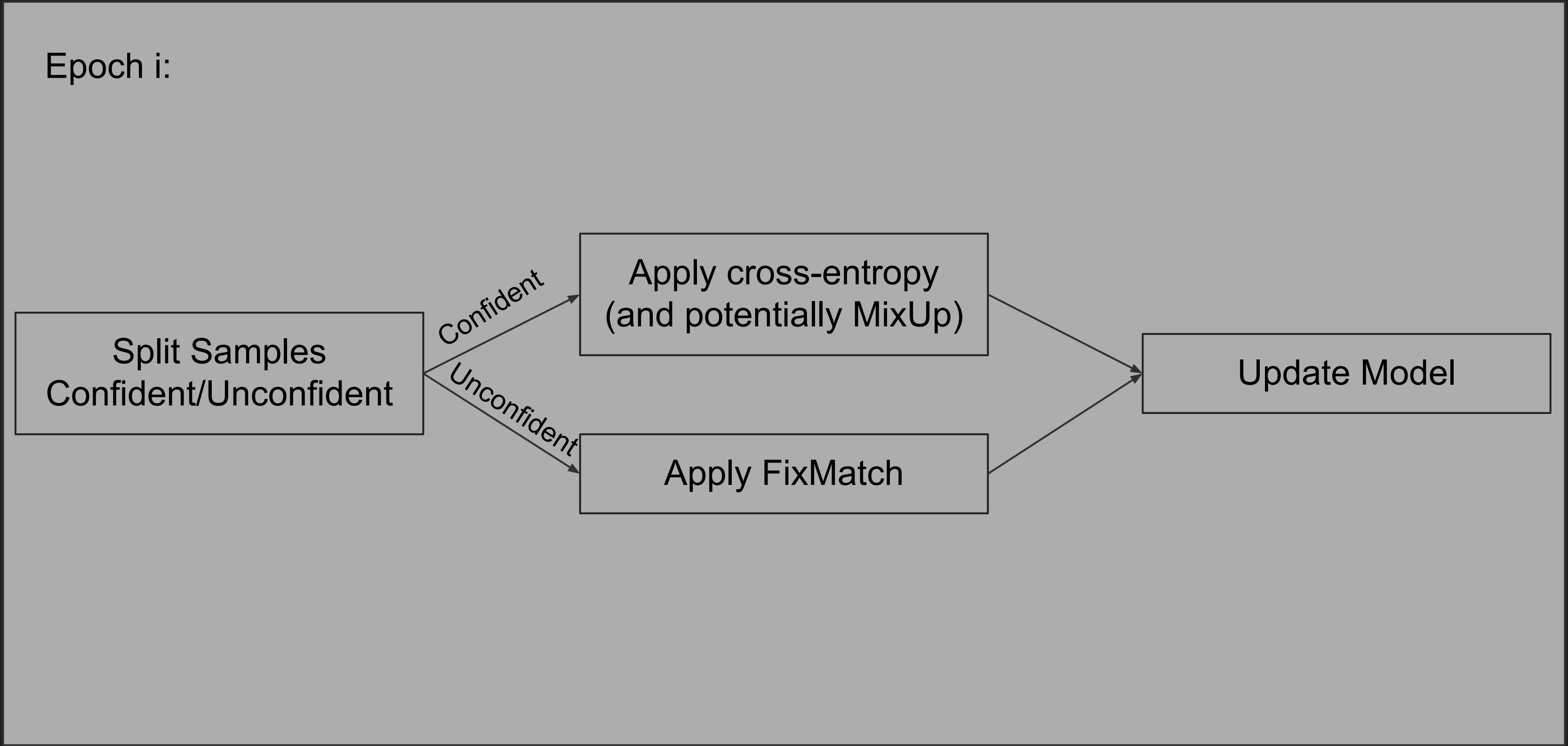
Unconfident Epoch vs Noise Rate
Sum
Avg
Thresh
Noise Rate
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
20
40
60
80
100
Epoch

Sum Metric Results (40% Noise)

Epoch:	0	10	25	50	75	100
Number Noisy/Total Confident	12246/33515	7559/31560	6601/31984	6437/31977	6201/33114	8239/34325
Number Noisy/Total Unconfident	7858/16485	12545/18440	13503/18016	13667/18023	13903/16886	11865/15675

Next Steps

- We can see that more often than not the unconfident set has a much higher noise percentage.
    - Using the average or sum metric seems to be the best option.
- Using this split, we can treat the confident samples as “clean” samples and the unconfident samples as “unlabeled” samples and apply semi-supervised learning techniques.
- In the high noise setting, there are still bad samples in the confident set but the total amount of noise is cut to 20%.
- I have yet to try the small-clean labeled set problem setting.


Epoch i:
Split Samples
Confident/Unconfident
Confident
Unconfident
Apply cross-entropy
(and potentially MixUp)
Apply FixMatch
Update Model

Updates

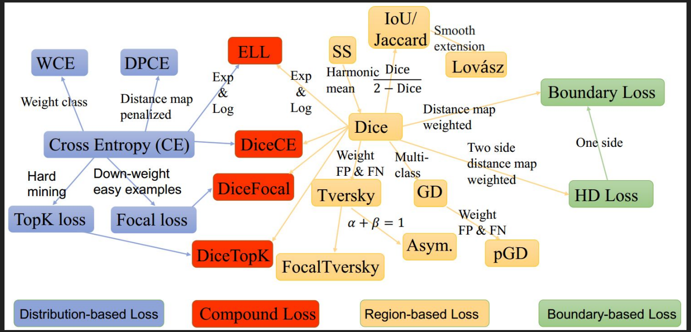
- I implemented the previous pipeline and it seems the performance is not up to par with other methods.
- Tried removing MixUp, increasing confidence threshold, and increasing warm-up period.
- I believe this is due to the model giving poor labels for the unconfident samples which impacts performance greatly.
- Will try using an ensembled label so that it can be built over time rather than a single decision made at once.

Medical Image Segmentation Questions

Problem Setting

- Is segmentation done by giving each pixel its own label on whether it is of interest or not?
- Are superpixels just generalizations of a local area that all appear the same?

Loss Functions



- What areas are best to focus on for newer learners?
- How are the pixels incorporated into loss functions?

Noise

- Two types of noise: image level and pixel level.
  - Is image level a noise caused by the entire label of the image?
  - Is pixel level dependent on what the annotator gave for each pixel of the image?