

ConMatch: Semi-Supervised Learning with Confidence-Guided Consistency Regularization

Jiwon Kim^{1,2***} and Youngjo Min^{1**}, Daehwan Kim^{3**}, Gyuseong Lee¹,
Junyoung Seo¹, Kwangrok Ryoo¹, and Seungryong Kim¹

¹ Korea University

² NAVER AI Lab

³ Samsung Electro-Mechanics

In this supplementary document, we provide additional experimental results and implementation details to complement the main paper. The source code and a pre-trained model will be released in the near future.

1 Implementation Details - Hyper-parameters

For a fair comparison, we basically followed the default hyperparameters of our baselines, i.e., FixMatch [5] and FlexMatch [6] as demonstrated in the Sec 4.2 of the main paper. In addition to this, we also provide the best hyperparameters for the specific loss functions and each training stage, as shown in Table 1.

Table 1. Additional hyper-parameter list of ConMatch

Stage	hyper-parameters	
Feature encoder pre-training	λ_{sup}	1.0
	λ_{un}	1.0
Confidence estimator pre-training	λ_{conf}	0.1
	$\lambda_{\text{conf-sup}}$	1.0
Fine-tuning	λ_{sup}	1.0
	λ_{un}	1.0
	λ_{ccr}	1.0
	λ_{conf}	1.0
	$\lambda_{\text{conf-sup}}$	1.0

2 Additional Experimental Results

Results of ConMatch with FixMatch Baseline Since our framework basically works as a plug-and-play module, it can be combined with various semi-supervised learners. We experimented FlexMatch [6]-based ConMatch (ConMatch with FlexMatch) on general semi-supervised learning benchmarks in Table 1 and Table 2 of the main paper. In this supplementary material, we additionally provide the benchmark results of FixMatch [5]-based ConMatch (ConMatch with FixMatch) on SVHN and STL-10 datasets. On SVHN with 40 labels

* Work done in Korea University

** Equal contribution

Table 2. Comparison on error rates on SVHN [4] and STL-10 [1] benchmarks on 3 different folds.

Method	SVHN		STL-10
	40	250	1,000
FixMatch [5]	3.96±2.17	2.48±0.38	7.98±1.50
FlexMatch [6]	4.97±0.06	4.98±0.09	5.77±0.18
ConMatch w/FixMatch [5]	3.25±1.06	1.86±0.06	5.90±0.17
ConMatch w/FlexMatch [6]	3.14±0.57	3.13±0.72	5.26±0.04

and 250 labels, ConMatch with FixMatch records the state-of-the-art accuracy of 96.75% and 98.14% respectively as displayed in Table 2. We could observe that ConMatch with FixMatch shows similar or even better results than ConMatch with FlexMatch on SVHN dataset. This is mainly due to the weakness of FlexMatch [6] on the *unbalanced* dataset. While training feature encoder of FlexMatch on SVHN, class-wise imbalance of SVHN leads the classes with fewer samples to have low thresholds. Such low thresholds allow noisy pseudo-labeled samples to be learned throughout the training process and eventually reduce the accuracy of model’s prediction. A feature encoder of FixMatch, on the other hand, is not affected by class-wise imbalance of dataset, since it fixes its threshold at 0.95 to filter out noisy samples. Consequently, this performance gap between two encoders caused by class-wise imbalance of dataset allows ConMatch with FixMatch to achieve similar or even higher accuracy score than ConMatch with FlexMatch.

Class-wise Results with Other SSL Techniques In Table 2 and Table 3 of the main paper, we demonstrated that semi-supervised learners [5, 6] combined with ConMatch outperform their baselines by a significant margin in most SSL benchmark settings. Additionally, to provide a detailed analysis for our performance results on CIFAR-10, we also performed a per-class quantitative evaluation as shown in Table 3. ConMatch w/ [5] achieves performance improvement over FixMatch in all the classes except for the *dog* class. Although FixMatch seems to predict *dog* class better than any other models, its prediction for the *cat* class, which is very similar to *dog*, obtains significantly low accuracy score. This result shows that Fixmatch suffers from a confirmation bias, and its prediction accuracy for *dog* class is a distorted value. ConMatch w/ [5] on the other hand, records high accuracy score for all classes including *cat* class. This verifies that ConMatch w/ [5] effectively alleviate the confirmation bias by making use of confidence estimator. Furthermore, ConMatch w/ [6] outperforms its baseline [6] on all classes. Based on all these class-wise evaluation and comparison results, we could confirm the effectiveness of combining ConMatch with existing semi-supervised learners.

Convergence Time. In this section, we analyze the effect of performance-boosting of ConMatch on end-to-end training. Specifically, we analyze the time taken to achieve the best accuracy of FixMatch [5], 86.19% in CIFAR-10 40 labels for a fair comparison between FixMatch [5], FlexMatch [6], and ConMatch

Table 3. Per-class quantitative evaluation on ConMatch and base semi-supervised learners [5, 6] on CIFAR-10 with 40 labels. The best results are in bold.

Methods	air.	auto.	bird	cat	deer	dog	frog	horse	ship	truck	total
FixMatch [5]	94.3	97.6	72.6	28.1	96.6	94.1	98.1	95.6	97.5	96.6	87.11
FlexMatch [6]	96.8	98.1	92.3	88.0	95.8	88.7	98.3	97.1	98.4	97.2	95.07
ConMatch w/ [5]	97.2	97.7	90.9	86.0	96.9	88.6	98.7	97.2	97.6	97.9	94.87
ConMatch w/ [6]	97.1	98.5	92.9	88.1	96.0	91.8	98.9	97.6	98.7	97.4	95.70

Table 4. Comparison of Convergence Time between FixMatch and ConMatch w/FixMatch Convergence time means the time taken to reach the highest accuracy of FixMatch, 86.19%, in CIFAR-10 40 labels. Runtime is measured in the same setting using RTX 3090 Ti. The best results are in bold.

Method	Iter	Convergence Time
FixMatch	261.9k	1,699 mins
ConMatch w/FixMatch	27.1k	117 mins
FlexMatch	44.6k	138 mins
ConMatch w/FlexMatch	20.3k	89 mins

due to their different convergence speeds. As seen from Table 4, FixMatch [5] takes 261.9k iters to converge, 1,699 minutes, while ConMatch w/FixMatch [5] converges at 27.1k iters, 117 minutes, which means that ConMatch can boost convergence about 14.5 times faster. Additionally, ConMatch w/FlexMatch [6] converges at 20.5k iters, 89 minutes, which is 1.5 times faster than FlexMatch [6] which converges at 44.6k iters, 138 minutes.

3 Ablation Studies

Warm-up Iteration As mentioned in Sec.3.5 of the main paper, we adopt the warm-up stage when training, to stabilize the confidence estimator at the early stage of training and to boost the convergence of training. Therefore, it can be the question that how many iterations of pre-training are we needed, so we perform an ablation study of these warm-up iterations at CIFAR-10 40 labels setting, and the results are shown in Table 5. After warm-up iteration, we perform the pre-training stage for confidence estimator for 10,000 iterations equally for all ablation settings except for the end-to-end training strategy. While without warm-up stage training results the accuracy of 93.24%, pre-training the feature extractor 10,000 iterations and the confidence estimator 10,000 iterations results 94.87% accuracy, showing 1.63% improvement compared to the end-to-end training strategy.

Designing Confidence Estimator Network The overall confidence estimation network, consisting of two sub-networks, is built as the collection of linear layer and ReLU activations. To input the logit values and features into the

Table 5. Ablation study on warm_up iterations for feature encoder. The best results are in bold.

Warm_up iter.	CIFAR-10
	40
0	93.24
1000	93.78
4000	94.87
5000	93.29
10000	93.10
20000	92.34

confidence estimator, it includes feature projection embedding and logit projection embedding. For the first variant of the confidence estimator, as shown in Fig. 1(b), we reduce the channel dimension of each linear layer of the confidence estimator by a factor of two, which is called *reduced* version. It shows 93.26% accuracy with the same evaluation setting, which is 0.91% increased results compared to the basic architecture as shown in Table 7 (I) and (II). Given the distributions of all classes from the classifier, we find that passing a part of it, especially masking the lower values of logit makes the estimator more robust and working well, shown in Fig. 1(c). The performance gap between (II) and (III) proves that the top-k approach is effective for the confidence estimator. We also find that the best top-k value is set by k=5 and that *masking* top-k, logit dimension is same as the number of class but masking class value, not belonging to top-k, is better than *reducing* top-k, logit dimension is same as top-k. For the final variant of the confidence estimator, we test the effect of BatchNorm [2] that came after the linear layer and a detailed architecture for this version is shown in Fig. 1(d). As shown in Table 7, (IV) record the best accuracy, 2.52%, 1.61%, and 0.93% better than (I), (II) and (III), respectively.

Similarity Functions in Non-parametric Approach In this section, we analyze the similarity function of the confidence estimation module in a non-parametric approach. We assume that the confidence of a strongly-augmented sample can be represented by the similarity between a weakly-augmented sample $\alpha(r)$ and a strongly-augmented sample $\mathcal{A}(r)$ in a probabilistic way. As seen in Eq.(4), we define a similarity function as a cross-entropy loss between the logit of weakly-augmented sample and the logit of the strongly-augmented sample. The similarity function can be substituted in other ways. Specifically, it can be formulated by L2 distance like $\|p_{\text{model}}(y|\mathcal{A}(r); \theta) - p_{\text{model}}(y|\alpha(r); \theta_{\text{freeze}})\|$ or feature cosine-similarity like $\frac{p_{\text{model}}(y|\mathcal{A}(r); \theta) \cdot p_{\text{model}}(y|\alpha(r); \theta_{\text{freeze}})}{\|p_{\text{model}}(y|\mathcal{A}(r); \theta)\| \|p_{\text{model}}(y|\alpha(r); \theta_{\text{freeze}})\|}$ where p_{model} is considered as logit and feature, respectively. Table 7 shows the results of replacing the similarity function with the L2 distance of logits and feature cosine-similarity. Through the result, we can find that the similarity function based on cross-entropy is the most effective confidence measure, compared to others.

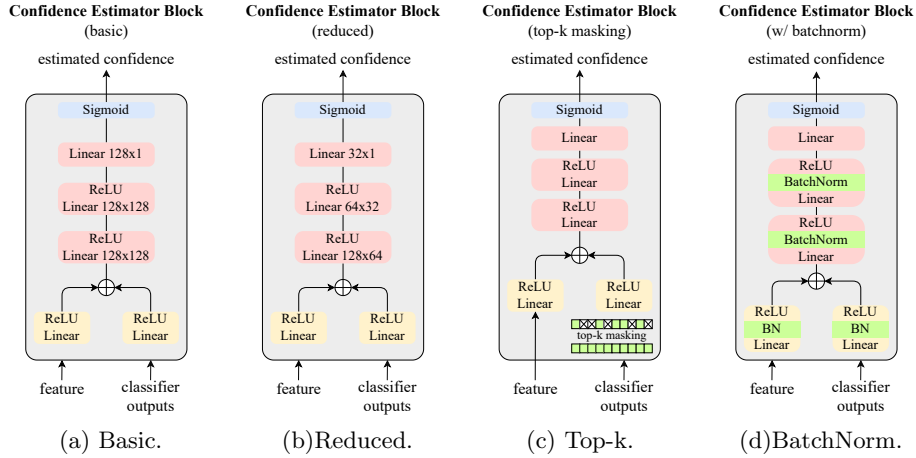


Fig. 1. Network Architecture Variants for Confidence estimator: (a) Basic-The number of parameters of linear layers is always constant. (b) Reduced-The number of parameters decreases to a multiplier of 2 according to the depth of the linear layers. (c) Top-k - The top-k values of the classifier outputs are only used for confidence estimator, by masking class values that do not belong to the top-k. (d) BatchNorm.-Batch normalization layer is followed by the linear layer.

Table 6. Ablation study on network architecture variants for confidence estimator.The best results are in bold.

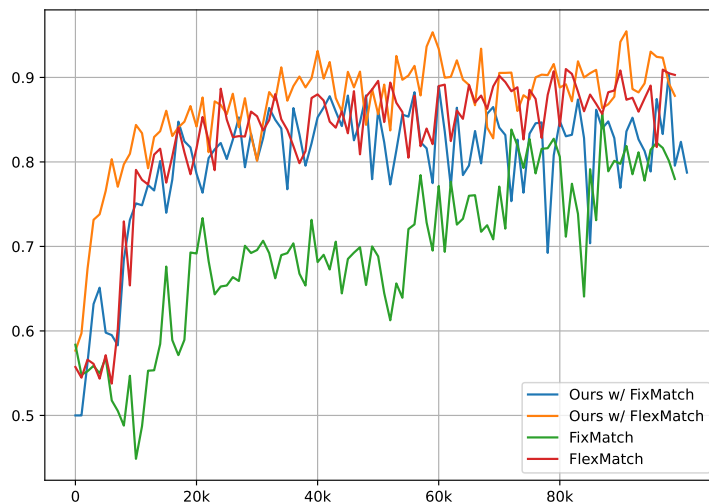
	Network Arch. for Confidence estimator				CIFAR-10
	Basic	Reduced	Top-K	BatchNorm	
(I)	✓	✗	✗	✗	92.35
(II)	✗	✓	✗	✗	93.26
(III)	✗	✓	✓	✗	93.94
(IV)	✗	✓	✓	✓	94.87

4 Visualization

Visualization of AUC-ROC curve of Confidence. The area under the ROC curve (AUC) is used for measuring the accuracy of a classifier and the better confidence prediction is closer to 1. On confidence estimation, it is also used to evaluate the capability of the confidence measure to distinguish correct class assignments from erroneous ones. As shown in Fig. 2, for overall training iterations, ConMatch w/ [5] and ConMatch w/ [6] show the higher AUC-ROC value compared to each baseline, FixMatch [5], and FlexMatch [6]. This explains that the confidence estimator measures the confidences of augmented instances more properly than FixMatch and FlexMatch, which use max class probability as the confidence measure. This will be also proven in the following section.

Table 7. Ablation study on similarity functions in non-parametric approach. The best results are in bold.

Similarity Function	Target	CIFAR-10
		40
Cross Entropy	logit	4.89
L2 Norm	logit	4.93
Cosine Similarity	feature	5.02

**Fig. 2. AUC-ROC Curve of ConMatch with various baselines** The orange line means FlexMatch w/ ConMatch, the blue line means FixMatch w/ ConMatch, the red line means FlexMatch, and the green line means original FixMatch, respectively.

Visualization on Transition of Confidence. To explain the effectiveness of ConMatch, we show the feature distribution of ConMatch w/FixMatch [5], ConMatch w/FlexMatch [6], compared to [5] and [6] at different training iterations in Table 8, Table 9, Table 10 and Table 11, respectively. For visualizing the high dimensional features, the complex feature matrices are transformed into two-dimensional points based on t-SNE [3]. As training progresses, we can confirm that t-SNE visualization of labeled and weakly-augmented samples are clustered into distinct groups, relatively faster than baseline [5] by comparing Table 8 and Table 9. And we can also find that confident strongly-augmented samples are also well-organized near confident weakly-augmented samples.

The color of dots means the confidence of each sample. The darkest color dot means the high-confidence instances, and the empty dot means the low- or zero-confidence instances, whose confidence values are relatively lower. For ConMatch, we consider the confidence estimator output as the confidence of each instance, and for the FixMatch [5] and FlexMatch [6], we consider the maximum

probability of the output distribution as the confidence of each instance. Both methods, ConMatch and FixMatch / FlexMatch show that the high-confident strongly-augmented instances are closer to high-confident weakly-augmented instances than the low-confident strongly-augmented instances. About FixMatch and FlexMatch, they measure the confidence of the insufficiently clustered strongly augmented samples as relatively high around 0.5, expressed as the *light-blue* color of figures. Otherwise, different to FixMatch and FlexMatch, ConMatch shows sharp changes in confidence values, distinguishing the high- and low-confidence instances definitely so the insufficiently clustered samples have low confidence values around zero, expressed as the *empty* circle of figures, even from the early stage of training. By this visualization, we can confirm that our confidence estimator effectively measures the confidence of samples and makes possible the training between the strongly augmented images.

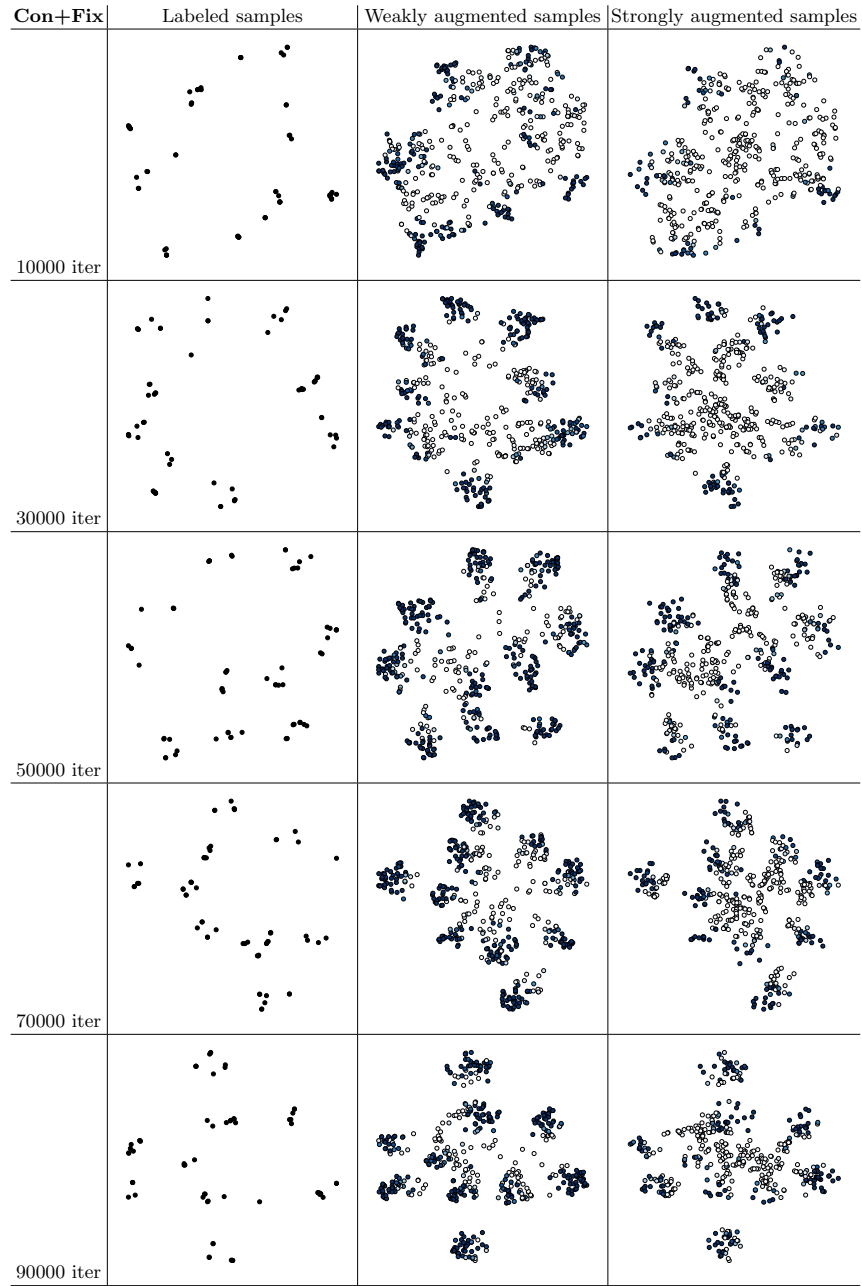


Table 8. Visualization on the transitions feature distribution with confidence in **Con-Match w/FixMatch** (denoted as blue dots in colors) of training samples from CIFAR-10 in the feature space at different training stages. Data projection in 2-D space is attained by t-SNE based on the feature representation. The dark blue dots represent the most confident samples, and the empty dots represent the least confident samples.

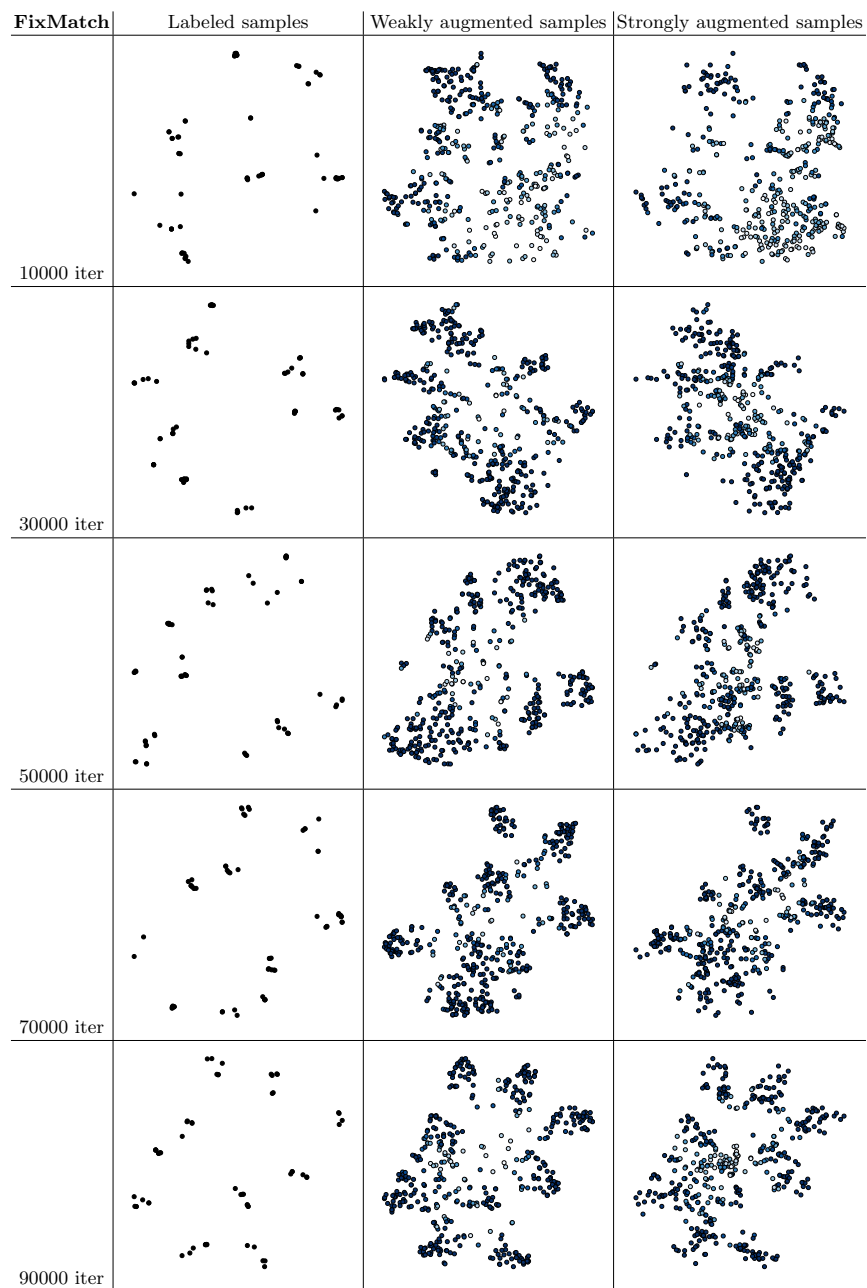


Table 9. Visualization on the transitions feature distribution with confidence in **FixMatch**. The details are same as above.

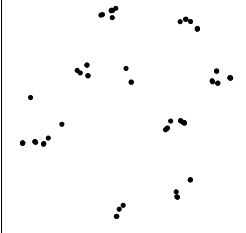
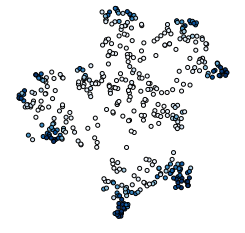
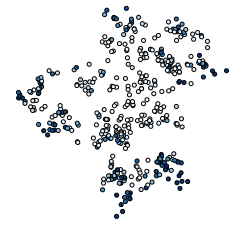
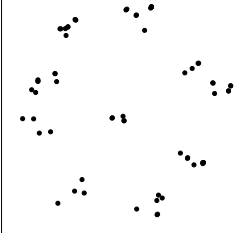
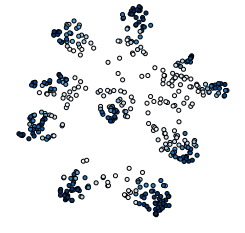
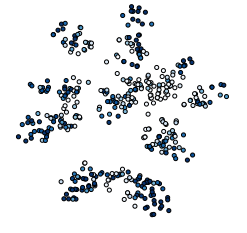

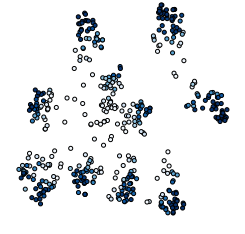
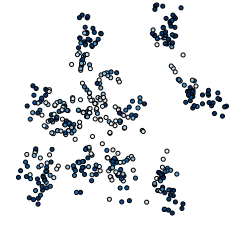
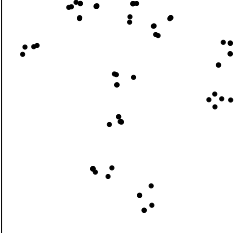
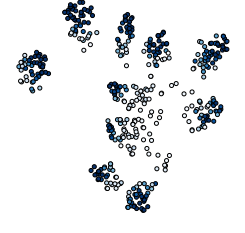
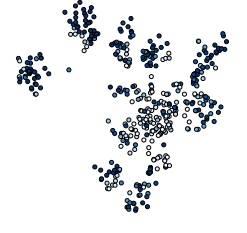

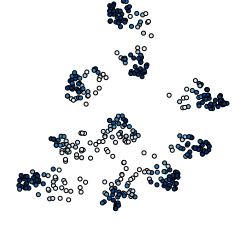
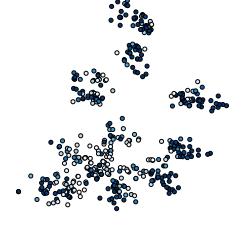
Con+Flex	Labeled samples	Weakly augmented samples	Strongly augmented samples
10000 iter			
30000 iter			
50000 iter			
70000 iter			
90000 iter			

Table 10. Visualization on the transitions feature distribution with confidence in **Con-Match w/FlexMatch**. The details are same as above.


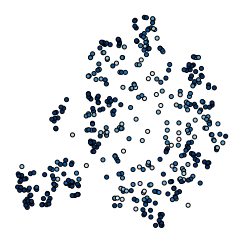
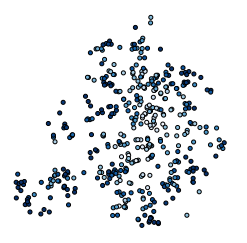
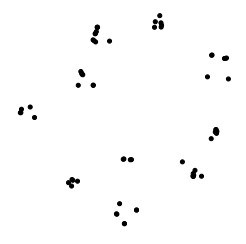
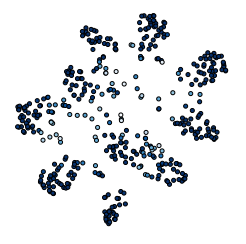
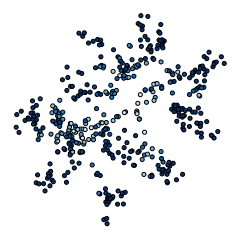
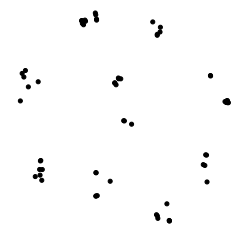
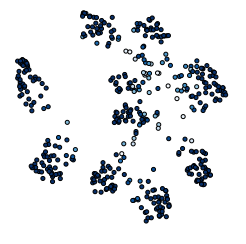
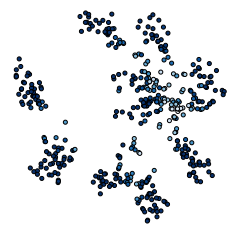
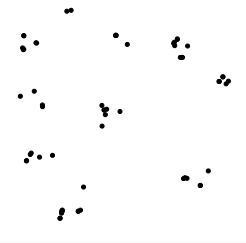
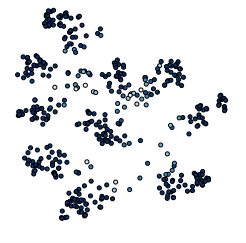
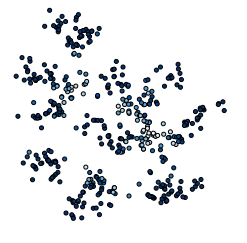

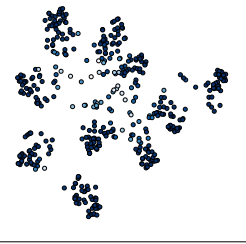
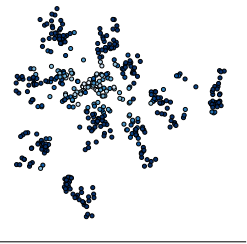
FlexMatch	Labeled samples	Weakly augmented samples	Strongly augmented samples
10000 iter			
30000 iter			
50000 iter			
70000 iter			
90000 iter			

Table 11. Visualization on the transitions feature distribution with confidence in **FlexMatch**. The details are same as above.

References

1. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: AISTATS (2011)
2. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
3. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. JMLR (2008)
4. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
5. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C.A., Cubuk, E.D., Kurakin, A., Li, C.L.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In: NeurIPS (2020)
6. Zhang, B., Wang, Y., Hou, W., Wu, H., Wang, J., Okumura, M., Shinozaki, T.: Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. In: NeurIPS (2021)