Appendix: FlexMatch: Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling

Bowen Zhang*
Tokyo Institute of Technology
bowen.z.ab@m.titech.ac.jp

Yidong Wang*
Tokyo Institute of Technology
wang.y.ca@m.titech.ac.jp

Wenxin Hou
Microsoft
wenxinhou@microsoft.com

Hao Wu
Tokyo Institute of Technology
wu.h.aj@m.titech.ac.jp

Jindong Wang†
Microsoft Research Asia
jindwang@microsoft.com

Manabu Okumura†
Tokyo Institute of Technology
oku@pi.titech.ac.jp

Takahiro Shinozaki†
Tokyo Institute of Technology
shinot@ict.e.titech.ac.jp

1 Experimental Results

1.1 Hyperparameter setting

For reproduction, we show the detailed hyperparameter setting for each method in Table 1 and 2, for algorithm-dependent and algorithm-independent hyperparameters, respectively.

Table 1: Algorithm dependent parameters.

Algorithm	PL (Flex-PL)	UDA (Flex-UDA)	FixMatch (FlexMatch)
Unlabeled Data to Labeled Data Ratio (CIFAR-10/100, STL-10, SVHN)	1	7	7
Unlabeled Data to Labeled Data Ratio (ImageNet)	-	-	1
Pre-defined Threshold (CIFAR-10/100, STL-10, SVHN)	0.95	0.8	0.95
Pre-defined Threshold (ImageNet)	-	-	0.7
Temperature	-	0.5	-

Table 2: Algorithm independent parameters.

Dataset	CIFAR-10	CIFAR-100	STL-10	SVHN	ImageNet
Model	WRN-28-2 [1]	WRN-28-8	WRN-37-2 [2]	WRN-28-2	ResNet-50 [3]
Weight Decay	5e-4	1e-3	5e-4	5e-4	3e-4
Batch Size	64				128
Learning Rate	0.03				
SGD Momentum	0.9				
EMA Momentum	0.999				
Unsupervised Loss Weight	1				

1.2 Class-wise accuracy improvement.

As introduced in the paper, CPL has its ability of improving performance on those hard-to-learn classes by taking into consider the model's learning status. A detailed class-wise accuracy comparison

*Equal contribution.
†Corresponding author.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

is listed in Table 3, where the final accuracies of class 2, 3 and 5 with originally bad performance are improved.

Table 3: Class-wise accuracy comparison on CIFAR-10 40-label split.

Class Number	0	1	2	3	4	5	6	7	8	9
FixMatch	0.964	0.982	**0.697**	0.852	0.974	0.890	0.987	0.970	0.982	0.981
FlexMatch	0.967	0.980	**0.921**	0.866	0.957	0.883	0.988	0.975	0.982	0.968

1.3 Median error rates

We also report the median error rates of the last 20 checkpoints by allowing all methods to run the same iterations, following existing work [4]. There are 1000 iterations between every two checkpoints. The results in Table 4 show that our CPL method can dramatically improve the performance of existing SSL algorithms and the FlexMatch achieves the best accuracy. These conclusions are in consistency with the results of Table1 in the main text, showing the effectiveness of our proposed CPL algorithm.

Table 4: Median error rates of the last 20 checkpoints.

Dataset	CIFAR-10			CIFAR-100			STL-10			SVHN	
Label Amount	40	250	4000	400	2500	10000	40	250	1000	40	1000
PL	77.42±1.19	48.33±2.43	15.64±0.29	90.01±0.21	58.38±0.42	37.64±0.16	**76.44**±0.67	56.90±2.32	33.57±0.40	69.05±6.77	**9.99**±0.35
Flex-PL	**76.09**±2.25	**47.53**±2.25	**15.30**±0.24	**86.60**±0.48	**56.72**±0.54	**36.20**±0.20	76.84±1.04	**53.71**±2.69	**33.19**±0.25	**67.20**±3.99	15.10±1.33
UDA	10.96±3.68	**5.46**±0.07	4.60±0.05	**51.97**±1.38	29.92±0.35	23.64±0.33	**41.11**±5.21	**10.74**±1.39	8.00±0.58	**5.31**±4.39	**1.97**±0.04
Flex-UDA	**5.77**±0.52	5.48±0.33	**4.52**±0.07	59.51±2.70	**29.33**±0.23	**23.38**±0.19	61.16±4.34	10.88±0.54	**7.16**±0.20	6.21±2.84	2.13±0.09
FixMatch	7.99±0.59	**5.12**±0.33	**4.46**±0.11	48.95±1.19	29.19±0.25	23.06±0.12	44.70±6.58	12.34±2.13	7.38±0.26	**3.92**±1.18	**2.06**±0.01
FlexMatch	**5.19**±0.05	5.33±0.12	4.47±0.09	**45.91**±1.76	**28.11**±0.20	**23.04**±0.28	**44.69**±7.49	**9.27**±0.49	**6.15**±0.25	20.81±5.26	12.90±2.68

1.4 Detailed results

To comprehensively evaluate the performance of all methods in a classification setting, we further report the precision, recall, f1 score and AUC (area under curve) results on CIFAR-10 dataset. As shown in Table 5, we see that in addition to the reduced error rates, CPL also has the best performance on precision, recall, F1 score, and AUC. These metrics, together with error rates (accuracy), shows the strong performance of our proposed method.

Table 5: Precision, recall, f1 score and AUC results on CIFAR-10.

Label Amount	40 labels				4000 labels			
Criteria	Precision	Recall	F1 Score	AUC	Precision	Recall	F1 score	AUC
PL	0.2539	0.2552	0.2493	0.6542	0.8498	0.8509	0.8500	0.9833
Flex-PL	**0.2865**	**0.2865**	**0.2663**	**0.6718**	**0.8544**	**0.8545**	**0.8542**	**0.9843**
UDA	0.8759	0.8408	0.8086	0.9775	0.9557	0.9559	0.9557	0.9985
Flex-UDA	**0.9482**	**0.9485**	**0.9482**	**0.9974**	**0.9576**	**0.9577**	**0.9576**	**0.9986**
Fixmatch	0.9333	0.9290	0.9278	0.9910	0.9571	0.9571	0.9569	**0.9984**
Flexmatch	**0.9506**	**0.9507**	**0.9506**	**0.9975**	**0.9580**	**0.9581**	**0.9580**	**0.9984**

2 TorchSSL: A PyTorch-based SSL Codebase

The PyTorch [5] framework has gained increasing attention in the deep learning research community. However, the main existing SSL codebase [6] is based on TensorFlow. For the convenience and customizability, we re-implement and open source a PyTorch-based SSL toolbox, named *TorchSSL* [3] as shown in Figure 1. TorchSSL contains eight popular semi-supervised learning methods: Π-Model [8], Pseudo-Labeling [9], VAT [10], Mean Teacher [11], MixMatch [12], ReMixMatch [13], UDA [14],

[3]Our toolbox is partially based on [7].

and FixMatch [4], along with our proposed method FlexMatch. Most of our implementation details are based on [6]. More importantly, in addition to the basic SSL methods and components, we implement several techniques to make the results stable under PyTorch framework. For instance, we add synchronized batch normalization [15] to avoid the performance degradation caused by multi-GPU training with small batch size, and a batch norm controller to prevent performance crashes for some algorithms, which is not officially supported in PyTorch.

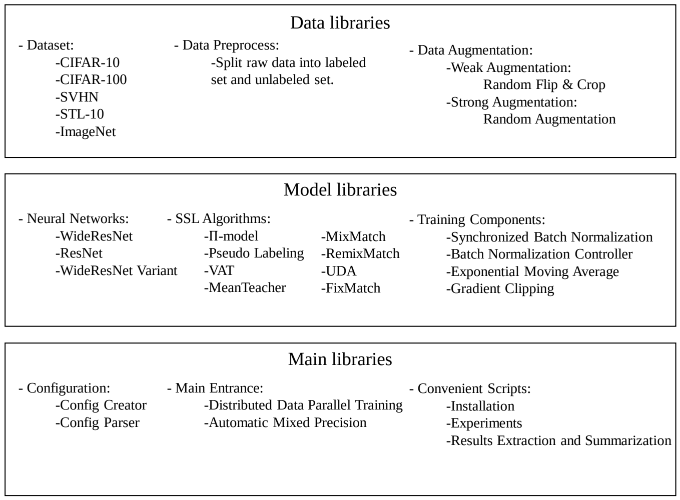

Figure 1: Components of TorchSSL.

2.1 BatchNorm Controller

We observed that Mean Teacher can be very unstable if we update BatchNorm for both labeled data and unlabeled data in turn. Other algorithms such as Π-Model and MixMatch also show the similar instability. Therefore, we use BatchNorm Controller to update BatchNorm only for labeled data if labeled data and unlabeled data are forwarded separately. The code of BatchNorm Controller is as follows. We record the BatchNorm statistics before the forward propagation of unlabeled data and restore them after the propagation is done.

2.2 Benchmark results

We comprehensively run all algorithms in our TorchSSL on four common datasets in SSL: CIFAR-10, CIFAR-100, SVHN, and STL-10, and report the best error rates in Table 6, 7, 8, and 9, respectively. These benchmark results provide a reference of using this toolbox.

Table 6: Benchmark results on CIFAR-10. The error bars are obtained from three trials.

Algorithms	Error Rate (40 labels)	Error Rate (250 labels)	Error Rate (4000 labels)
Π-Model [8]	74.34±1.76	46.24±1.29	13.13±0.59
Pseudo-Labeling [9]	74.61±0.26	46.49±2.20	15.08±0.19
VAT [10]	74.66±2.12	41.03±1.79	10.51±0.12
Mean Teacher [11]	70.09±1.60	37.46±3.30	8.10±0.21
MixMatch [12]	36.19±6.48	13.63±0.59	6.66±0.26
ReMixMatch [13]	9.88±1.03	6.30±0.05	4.84±0.01
UDA [14]	10.62±3.75	5.16±0.06	4.29±0.07
FixMatch [4]	7.47±0.28	4.86±0.05	4.21±0.08
FlexMatch	4.97±0.06	4.98±0.09	4.19±0.01

Table 7: Benchmark results on CIFAR-100.

Algorithms	Error Rate (400 labels)	Error Rate (2500 labels)	Error Rate (10000 labels)
Π-Model [8]	86.96±0.80	58.80±0.66	36.65±0.00
Pseudo-Labeling [9]	87.45±0.85	57.74±0.28	36.55±0.24
VAT [10]	85.20±1.40	46.84±0.79	32.14±0.19
Mean Teacher[11]	81.11±1.44	45.17±1.06	31.75±0.23
MixMatch [12]	67.59±0.66	39.76±0.48	27.78±0.29
ReMixMatch [13]	42.75±1.05	26.03±0.35	20.02±0.27
UDA [14]	46.39±1.59	27.73±0.21	22.49±0.23
FixMatch [4]	46.42±0.82	28.03±0.16	22.20±0.12
FlexMatch	39.94±1.62	26.49±0.20	21.90±0.15

Table 8: Benchmark results on STL-10.

Algorithms	Error Rate (40 labels)	Error Rate (250 labels)	Error Rate (1000 labels)
Π-Model [8]	74.31±0.85	55.13±1.50	32.78±0.40
Pseudo-Labeling [9]	74.68±0.99	55.45±2.43	32.64±0.71
VAT [10]	74.74±0.38	56.42±1.97	37.95±1.12
Mean Teacher [11]	71.72±1.45	56.49±2.75	33.90±1.37
MixMatch [12]	54.93±0.96	34.52±0.32	21.70±0.68
ReMixMatch [13]	32.12±6.24	12.49±1.28	6.74±0.14
UDA [14]	37.42±8.44	9.72±1.15	6.64±0.17
FixMatch [4]	35.97±4.14	9.81±1.04	6.25±0.33
FlexMatch	29.15±4.16	8.23±0.39	5.77±0.18

Table 9: Benchmark results on SVHN.

Algorithms	Error Rate (40 labels)	Error Rate (250 labels)	Error Rate (1000 labels)
Π-Model [8]	67.48±0.95	13.30±1.12	7.16±0.11
Pseudo-Labeling [9]	64.61±5.60	15.59±0.95	9.40±0.32
VAT [10]	74.75±3.38	4.33±0.12	4.11±0.20
Mean Teacher [11]	36.09±3.98	3.45±0.03	3.27±0.05
MixMatch [12]	30.60±8.39	4.56±0.32	3.69±0.37
ReMixMatch [13]	24.04±9.13	6.36±0.22	5.16±0.31
UDA [14]	5.12±4.27	1.92±0.05	1.89±0.01
FixMatch [4]	3.81±1.18	2.02±0.02	1.96±0.03
FlexMatch	8.19±3.20	6.59±2.29	6.72±0.30

References

[1] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[2] Tianyi Zhou, Shengjie Wang, and Jeff Bilmes. Time-consistent self-supervision for semi-supervised learning. In *ICML*, pages 11523–11533. PMLR, 2020.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *NeurIPS*, 33, 2020.

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32:8026–8037, 2019.

[6] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. fixmatch. `https://github.com/google-research/fixmatch`, 2020.

[7] Lee Doyup and Cheon Yeongjae. Fixmatch-pytorch. `https://github.com/LeeDoYup/FixMatch-pytorch`, 2020.

[8] Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. Semi-supervised learning with ladder networks. In *NeurIPS*, pages 3546–3554, 2015.

[9] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 2013.

[10] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE TPAMI*, 41(8):1979–1993, 2018.

[11] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, pages 1195–1204, 2017.

[12] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *NeurIPS*, page 5050–5060, 2019.

[13] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *ICLR*, 2019.

[14] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. *NeurIPS*, 33, 2020.

[15] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *CVPR*, pages 7151–7160, 2018.