INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

slimIPL: Language-Model-Free Iterative Pseudo-Labeling

Tatiana Likhomanenko[1], *Qiantong Xu*[1], *Jacob Kahn*[1], *Gabriel Synnaeve*[2], *Ronan Collobert*[1]

[1]Facebook AI Research, USA
[2]Facebook AI Research, France

antares@fb.com

Abstract

Recent results in end-to-end automatic speech recognition have demonstrated the efficacy of pseudo-labeling for semi-supervised models trained both with Connectionist Temporal Classification (CTC) and Sequence-to-Sequence (seq2seq) losses. Iterative Pseudo-Labeling (IPL), which continuously trains a single model using pseudo-labels iteratively re-generated as the model learns, has been shown to further improve performance in ASR. We improve upon the IPL algorithm: as the model learns, we propose to iteratively re-generate transcriptions with hard labels (the most probable tokens), that is, *without* a language model. We call this approach Language-Model-Free IPL (slimIPL) and give a resultant training setup for low-resource settings with CTC-based models. slimIPL features a dynamic cache for pseudo-labels which reduces sensitivity to changes in relabeling hyperparameters and results in improved training stability. slimIPL is also highly-efficient and requires 3.5-4x fewer computational resources to converge than other state-of-the-art semi/self-supervised approaches. With only 10 hours of labeled audio, slimIPL is competitive with self-supervised approaches, and is state-of-the-art with 100 hours of labeled audio without the use of a language model both at test time and during pseudo-label generation.

Index Terms: deep learning, semi-supervised learning, pseudo-labeling, self-training, speech recognition

1. Introduction

Recent work in deep learning has shifted towards methods which can efficiently learn from large amounts of unlabeled data to improve performance and decrease costs associated with labeling. Semi-supervised learning [1] combines information from both labeled and unlabeled data; the amount of unlabeled data typically exceeds the amount of labeled data. In automatic speech recognition (ASR), while many recent semi-supervised methods outperform a supervised baseline in a low-resource setting, a gap between semi- and fully-supervised training remains. Further, not all of the approaches are equally scalable as the amount of labeled and unlabeled data increases, as is the case in recent setups such as the Libri-Light benchmark [2].

Some of the earliest and simplest semi-supervised approaches use self-training [3]. Self-training employs a base model trained with labeled data which acts as a "teacher" and is used to label unlabeled data (the resulting labels are referred as "pseudo-labels", PLs). A "student" model is then trained (typically from scratch) with both labeled and pseudo-labeled data to yield a final model. For competitive results in ASR, a language model (LM) was a key component of pseudo-labeling: it is usually combined with the acoustic model via beam-search decoding [4, 5, 6] or through shallow fusion [7, 8, 9, 10] to generate PLs. With this setting, however, acoustic models tend to over-fit to the text training set of the LM used for pseudo-labeling [5, 9].

In this work, we show that competitive pseudo-labeling approaches rely neither on beam-search decoding nor on a language model. In our setup, pseudo-labels are generated by picking hard labels – tokens with the highest acoustic model probability. Our approach is based on the recently-proposed iterative pseudo-labeling algorithm (IPL) [5]: we continuously train a single model using iteratively re-generated pseudo-labels as model learns. We call our algorithm language-model-free IPL (slimIPL) and give its overview in Section 4. We demonstrate in Section 5 that this approach is effective for models trained with Connectionist Temporal Classification (CTC) [11] in low-resource settings and is competitive with current state-of-the-art results. Ablation studies on different aspects of the proposed algorithm in Section 5.6 show slimIPL provides training stability and a robustness to hyperparameter settings.

2. Related Work

Self-training methods [3] still attract researchers: extensions to self-training are numerous and include (a) selecting particular subsets of pseudo-labeled data for student training, (b) the reiteration of the PL procedure several times to progressively-improve the teacher model, (c) the introduction of different types of noise for student model training, and (d) sampling techniques and schedules for training over labeled and pseudo-labeled datasets. Many recent works on self-training propose and validate these extensions, including those in computer vision [12, 13], natural language processing [14, 15, 16, 17, 18, 19, 20], ASR [21, 7, 22, 8, 9], and speech translation [23].

One extension to the simple pseudo-labeling method consists of continuously training a single model [24]. At the beginning of training, a model is trained only on labeled data after which training continues on data jointly-selected from both labeled and unlabeled datasets. PL re-generation occurs after some number of iterations, and a supervised loss is computed both on labeled and pseudo-labeled data for each batch. An additional parameter determines the contribution of pseudo-labeled data to the overall loss. The effectiveness of this iterative training for a single model has been validated on tasks in vision [25], natural language processing [18], and ASR [26, 5]. Below, we give an overview of the most relevant approaches in ASR to our work.

Iterative pseudo-labeling (IPL) [5] algorithm follows prior work [24] and uses augmentation of both labeled and unlabeled data, and continuously trains a single model with iterative re-generation of PLs by beam-search decoding with an LM, as the model learns. Compared to IPL, we maintain a dynamic cache with PLs and don't use any beam-search decoding or an LM.

Noisy self-training [9] performs five iterations of student network training, each time from scratch, with PLs generated by a teacher network. In this approach, as is the case with IPL, shallow fusion with an LM is used with a decoding procedure to generate PLs, while slimIPL doesn't use an LM at all.

Self-training [26] is the closest to our work: the authors con-

http://dx.doi.org/10.21437/Interspeech.2021-740

tinuously train a model with re-generated hard PLs after *each* iteration. This work is more focused on studying the impact of noise, and both SpecAugment [27] and speed perturbation are applied for labeled and unlabeled data during training. Our experiments show that re-generating PLs with hard labels after *each* iteration causes training instability resulting in divergence, whereas slimIPL exploits a dynamic cache mechanism to stabilize training.

wav2vec [28]'s unsupervised pre-training gives a significant boost in performance for low-resource settings. Training has two steps: first, pre-training on unlabeled data by masking the input audio in the latent space and solving a contrastive learning task [29]; second, fine-tuning the model using labeled audio only. Recently, one-step training [30] is proposed to directly optimize a downstream task which simplifies hyperparameter tuning. One of the known problems with contrastive training is a need of large batches [28, 30].

3. Pseudo-Labeling

Let $L = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ be a labeled dataset and $U = \{\boldsymbol{x}_j\}$ a large unlabeled dataset. We consider a semi-supervised pseudo-labeling approach where the acoustic model (AM) $\mathcal{M}_\theta$ is continuously trained on combination of a labeled set and an iteratively re-generated pseudo-labeled set. Training minimizes the following loss function: $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_L(\boldsymbol{\theta}) + \lambda\mathcal{L}_U(\boldsymbol{\theta})$, $\lambda \in \mathbb{R}^+$, where $\boldsymbol{\theta}$ are the parameters of the AM, and λ is a tunable parameter controlling the importance of unlabeled data. The losses for labeled data $\mathcal{L}_L$ and for unlabeled data $\mathcal{L}_U$ are defined as $\mathcal{L}_L(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim p(\boldsymbol{x},\boldsymbol{y})} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$, $(\boldsymbol{x}, \boldsymbol{y}) \in L$, where $p(\boldsymbol{x}, \boldsymbol{y})$ is the empirical data distribution of samples from L, $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ is the conditional distribution defined by $\mathcal{M}_\theta$, $\mathcal{L}_U(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \log p_{\boldsymbol{\theta}}(\hat{\boldsymbol{y}}|\boldsymbol{x})$, $\boldsymbol{x} \in U$, where $p(\boldsymbol{x})$ is the empirical data distribution of samples from U, and $\hat{\boldsymbol{y}}$ are the pseudo-labels for utterance $\boldsymbol{x} \in U$.

One key difference in existing pseudo-labeling approaches is how the labels assignments $\hat{\boldsymbol{y}}$ are obtained for unlabeled data $\boldsymbol{x} \in U$. In the general literature, pseudo-labeling refers to the hard label generation:

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}). \quad (1)$$

In machine translation and ASR domains, the model $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ is often sequence-to-sequence, and the solution of Eq. (1) may be approximated with a beam-search decoding algorithm [31, 18, 8, 7, 5, 9, 23, 26]. Indeed, most recent work in ASR relies on an LM $p^{lm}(\boldsymbol{y})$ to generate PLs, and instead optimizes:

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{argmax}} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) + \alpha \log p^{lm}(\boldsymbol{y}),\ \boldsymbol{x} \in U, \quad (2)$$

where α is a hyperparameter controlling the amount of language model regularization. Pseudo-labeling is also popular in computer vision [24, 32]. Variants exist, such as "soft labels" $\hat{\boldsymbol{y}} = p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$, variations on soft labeling [33, 34], "hard distillation" [35] and sampling [36, 37].

4. Language-Model-Free IPL

In the original IPL algorithm [5], PLs are generated with a beam-search decoder leveraging an LM and approximating the solution of Eq. (2). While the main motivation is to transfer the knowledge of the LM into the AM, drawbacks exist: (i) generating PLs is computationally intensive, and (ii) models easily over-fit to LM knowledge. Regularization techniques are proposed in [5] to overcome (ii), such that one can still benefit from the LM when decoding at evaluation time.

Algorithm 1: slimIPL

Data: labeled $L = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ and unlabeled $U = \{\boldsymbol{x}_j\}$
Result: Acoustic model $\mathcal{M}_\theta$
1. Train $\mathcal{M}_\theta$ on L with augmentation for M updates;
2. **while** *cache is not full at size* C **do**
  - Draw a random batch from $\boldsymbol{x} \in U$;
  - Generate its PL $\hat{\boldsymbol{y}}$ by $\mathcal{M}_\theta$ following Eq.(1);
  - Store $\{\boldsymbol{x}, \hat{\boldsymbol{y}}\}$ into the cache;
  - Train $\mathcal{M}_\theta$ on L with augmentation for 1 update;
end
3. Decrease model's $\mathcal{M}_\theta$ dropout;
repeat
  4. Train $\mathcal{M}_\theta$ on L with augmentation for N_L updates;
  5. **for** N_U *updates* **do**
    - Draw a random batch $B = \{\boldsymbol{x}, \hat{\boldsymbol{y}}\}$ from the cache;
    - **With probability** p, B is removed from cache and a new pair of random batch $\boldsymbol{x}' \in U$ and its PL $\hat{\boldsymbol{y}}'$ generated by $\mathcal{M}_\theta$ is added in;
    - Apply augmentation to batch B and make an optimization step to update $\mathcal{M}_\theta$.
  end
until *convergence or maximum iterations are reached*;

We demonstrate that PLs do not need to rely on any LM information at all. Algorithm 1 describes slimIPL. While it follows the IPL algorithm, PLs are generated by considering the top prediction according to the AM as per Eq. (1). For CTC-based acoustic models, this corresponds exactly to choosing the most likely token at each time step. Our approach also imitates self-training [26], but instead of re-generating PLs after *each* update we exploit a *dynamic cache* to use PLs generated by the *previous model states* (this can be viewed as models ensemble averaging for PL generation). This stabilizes the optimization process and avoids sudden model divergence as discussed in Section 5.6. In addition, a regularization scheme is implemented via data augmentation over the input (acoustic) data, both for labeled L and unlabeled U samples. slimIPL has several hyperparameters: (i) when PL generation begins M, (ii) the proportion of labeled and unlabeled data $\lambda = N_U/N_L$, (iii) the dynamic cache size C and the probability p of updating the cache.

5. Experiments

5.1. Data

All experiments are performed on the LibriSpeech dataset [40] (contains 960 hours of training audio with paired transcriptions: *train-clean-100*, *train-clean-360*, and *train-other-500* parts) and Libri-Light [2] labeled limited resource training subset *train-10h* originally extracted from LibriSpeech. We consider two low-resource scenarios with different amounts of labeled / unlabeled data: (i) LL-10/LS-960 uses *train-10h* as labeled data and full LibriSpeech as unlabeled; (ii) LS-100/LS-860 as labeled data and *train-clean-360* and *train-other-500* as unlabeled. The standard LibriSpeech validation sets (*dev-clean* and *dev-other*) are used to tune all hyperparameters, as well as to select the best models. Test sets (*test-clean* and *test-other*) are used only to report final word error rate (WER) performance. We keep the original 16kHz sampling rate and compute log-mel filterbanks with 80 coefficients for a 25ms sliding window, strided by 10ms. All features are normalized to have zero mean and unit variance

Table 1: *WER comparison of our supervised baselines with prior work: LL-10 (top) and LS-100 (bottom).*

Method	Stride	Tokens	Criterion	LM	Dev WER		Test WER	
					clean	other	clean	other
Libri-Light [2]	20 ms	letters	CTC	word 4-gram	34	60.9	33.5	62.1
Ours	30ms	letters	CTC	-	31.9	52.3	32.6	52.4
				word 4-gram	18.8	39.3	19.6	39.7
				+ rescoring	17.1	38.2	17.9	38.9
RWTH [38]	-	-	hybrid	word 4-gram	5.0	19.5	5.8	18.6
DeCoAR [39]	-	phn.	CTC	-	-	-	6.1	17.4
Improved T/S [9]	-	16k wp	S2S	-	5.3	16.5	5.5	16.9
Ours	30ms	letter	CTC	-	6.2	16.8	6.2	16.8
				word 4-gram	4.1	12.4	4.5	12.7
				+ rescoring	3.3	10.9	3.7	11.4

per input sequence before feeding them into the acoustic model.

5.2. Acoustic Models

We consider CTC-based models. Architectures follow [8]: the encoder is composed of a 1-D convolution with kernel size 7 and stride 3 followed by 36 4-head Transformer blocks [41]. The self-attention dimension is 768 and the feed-forward network (FFN) dimension is 3072 (with 4 heads) in each Transformer block. The output of the encoder is followed by a linear layer to the output classes. We use dropout after the convolution, dropout on the self-attention and on the FFN for all Transformer layers, and layer drop [42], dropping entire layers at the FFN level.

Tokens Letters are used for all experiments. The letter set consists of the 26 English alphabet letters augmented with the apostrophe and a word boundary token.

Data augmentation Training is performed with SpecAugment [27] only. We use two frequency masks with frequency mask parameter $F = 30$, ten time masks with maximum time-mask ratio $p = 0.1$ and time mask parameter $T = 50$; time warping is not used. For LL-10/LS-960 we found that twenty time masks with $T = 25$ improve performance.

Training For all experiments we use the Adagrad optimizer [43] and decay learning rate by 2 each time the WER reaches a plateau on the validation sets. All models architectures, as well as slimIPL are implemented within the flashlight[1] framework. Models are trained with dynamic batching (effective average batch size is 14 per GPU) and mixed-precision computations on 16 GPUs (Volta 32GB) for 350-500k updates.

5.3. Beam-search Decoding and Rescoring

In all our experimental results, we report not only WER without an LM, but also WER obtained with a one-pass beam-search decoder leveraging an LM. Following the notation introduced in Section 3, the beam-search decoder aims at maximizing:

$$\log p_{\boldsymbol{\theta}}(\hat{\boldsymbol{y}}|\boldsymbol{x}) + \alpha \log p^{lm}(\hat{\boldsymbol{y}}) + \beta|\hat{\boldsymbol{y}}|,$$

where α and β are hyperparameters to tune. We rely on the beam-search decoder from the flashlight framework following [44]: the lexicon-based beam-search decoder with a word-level LM. LibriSpeech validation sets, *dev-clean* and *dev-other*, are used to optimize the beam-search decoder hyperparameters, through random search. We also report WER obtained by rescoring the beam of hypothesis generated by the one-pass decoder. Rescoring is performed with a strong word-level Transformer LM, following the procedure described in [8]. We use open-sourced word-level LMs trained on the LibriSpeech LM corpus: 4-gram [45] and Transformer [8] LMs.

[1] `https://github.com/flashlight/flashlight`

5.4. Supervised Baselines

The dropout and layer drop parameters of our acoustic models were set to 0.5 (0.3) when trained on LL-10 (LS-100). Performance in WER is reported in Table 1. Our supervised baseline models trained on either LL-10 or LS-100 define a new state-of-the-art both on *test-clean* and *test-other* with beam-search decoding and further rescoring. On *test-other* these models are state-of-the-art even without an LM.

5.5. Semi-Supervised Experiments

slimIPL architectures are identical to their supervised counterparts, except for their dropout and layer drop values which are decreased after M (supervised-only) updates to 0.1 for both settings, see Algorithm 1 step 3. Stronger regularization (via high dropout) is critical to avoid over-fitting when training with labeled-only data. These regularization parameters are then decreased to "increase" model capacity, as more data is involved during the semi-supervised training (see ablation in Table 3 when dropout is not changed during the training). slimIPL hyperparameters were tuned by performing a search over the following ranges: M in 5k, 10k, 20k; N_U:N_L in 1:1, 1:2, 2:1, {3,4,5}:1, 10:1, 20:1; C in 10, 100, 1k; p in 0.1, 0.5, 1. The best configuration results are presented in Table 2. On the LL-10/LS-960 setup, slimIPL is the best performing pseudo-label-based technique, on both *test-clean* and *test-other*, almost closing the gap with wav2vec [28]. On LS-100/LS-860, we define a new state-of-the-art for LM-free approaches on both *test-clean* and *test-other*, while being similar to wav2vec with additional LM decoding.

5.6. Ablations

In Table 3, we study the robustness of slimIPL with respect to the choice of its hyperparameters. First, we consider the pseudo-labeling algorithm from [26], where PLs are generated from the acoustic model without dropout, from the acoustic samples without data augmentation, after *each* update (no cache, $C = no$). We found in practice that this approach is unstable, prone to unpredictable model divergence (generated transcriptions become empty), which can be occasionally recovered. In contrast, slimIPL is robust thanks to its dynamic cache strategy – in practice we observed no divergence if the cache size $C \geq 10$. For runs where the "cache-free" approach converges, we observe similar

Table 2: *Comparison with other semi- and unsupervised methods: LL-10/LS-960 (top) and LS-100/LS-860 (bottom).*

Method	Stride	Tokens	Criterion	LM	Dev WER		Test WER		Compute Resources		
					clean	other	clean	other	Train Time (Days)	# G/TPUs	G/TPU-days
Libri-Light [2]	20 ms	letters	CTC	word 4-gram	30.5	55.8	30.1	57.2	-	-	-
IPL [5]	80ms	5k wp	CTC	- + rescoring	23.8 23.5	25.7 25.5	24.6 24.4	26.5 26.0	3	64 GPUs	192
wav2vec 2.0 [28]	20ms	letter	CTC	- word 4-gram word Transf	8.1 3.4 2.9	12.0 6.9 5.7	8.0 3.8 3.2	12.1 7.3 6.1	2.3	128 GPUs	294.4
slimIPL	30ms	letter	CTC	- word 4-gram + rescoring	11.4 6.6 5.3	14 9.6 7.9	11.4 6.8 5.5	14.7 10.5 9.0	4.7	16 GPUs	75.2
IPL [5]	80ms	5k wp	CTC	- + rescoring	5.5 5.0	9.3 8.0	6.0 5.6	10.3 9.0	3	64 GPUs	192
Improved T/S [9]	-	16k wp	S2S	- LSTM	4.3 3.9	9.7 8.8	4.5 4.2	9.5 8.6	10×5	32 TPUs	1600
wav2vec 2.0 [28]	20ms	letters	CTC	- word 4-gram word Transf.	4.6 2.3 2.1	9.3 5.7 4.8	4.7 2.8 2.3	9.0 6.0 5.0	2.3	128 GPUs	294.4
slimIPL	30ms	letter	CTC	- word 4-gram + rescoring	3.7 2.7 2.2	7.3 5.5 4.6	3.8 3.1 2.7	7.5 6.2 5.2	5.2	16 GPUs	83.2

Table 3: *Ablations study on slimIPL hyperparameters reporting validation WER: LL-10/LS-960 (top) and LS100/LS-860 (bottom). "Naive" approach [26] is referred as $C = no$; "x" refers to the values from a baseline (grey) model.*

dropout	C	p	M/ WER-other	λ	no LM		4-gram LM	
					clean	other	clean	other
0.5→0.1	1000	0.1	20k/60	10/1	11.4	14.0	6.6	9.6
0.5	x	x	x	x	12.0	17.7	6.9	11.7
x	no	-	x	x	diverges			
x	10	x	x	x	17.8	20.7	9.0	12.8
x	100	x	x	x	13.0	15.3	7.4	10.0
x	x	0.5	x	x	12.3	15.8	7.0	10.1
x	x	1	x	x	13.8	17.5	7.2	10.7
x	x	x	5k/91	x	54.3	56.8	30.1	35.0
x	x	x	10k/73	x	23.7	26.5	9.6	13.8
x	x	x	40k/55	x	10.7	13.7	6.7	9.8
x	x	x	x	1/1	11.3	15.0	6.6	10.3
x	x	x	x	5/1	11.6	14.6	6.8	9.7
x	x	x	x	20/1	11.9	14.7	6.5	9.8
0.3→0.1	100	0.1	20k/33	1/1	3.6	7.5	2.9	5.9
0.3	x	x	x	x	4.1	8.1	3.1	6.3
x	no	-	x	x	3.9	7.5	3.1	5.9
x	10	x	x	x	3.9	7.7	2.9	5.9
x	1000	x	x	x	3.8	7.5	2.8	5.9
x	x	0.5	x	x	3.8	7.3	3.0	5.9
x	x	1	x	x	4.1	8.1	3.1	6.1
x	x	x	5k/80	x	3.9	7.8	2.9	5.9
x	x	x	10k/53	x	3.9	7.6	2.9	5.9
x	x	x	50k/23	x	3.9	7.8	2.9	5.9
x	x	x	x	1/2	4.2	8.5	3.1	6.4
x	x	x	x	2/1	3.7	7.3	2.7	5.6
x	x	x	x	3/1	3.7	7.3	2.8	5.6
x	x	x	x	4/1	3.7	7.3	2.7	5.5

performance with slimIPL, as shown in Table 3. In addition, lowering the cache update probability p allows to re-generate PLs more rarely in slimIPL than in cache-free approaches, which leads to faster training. slimIPL is robust to the cache size C and cache update probability p. However, for limited supervision the small cache setting $C = 10$ performs worse.

The starting update M for involving unlabeled data in the training process is critical for the low-resource labeled data setting LL-10/LS-960. In the case of LS-100/LS-860, slimIPL recovers even when starting from high WER supervised models. To prevent quick over-fitting over supervised data for LL-10/LS-960, the ratio λ between the number of unsupervised and supervised updates should be greater than 1, with little variations in WER for any $\lambda > 1$. For LS-100/LS-860 this hyperparameter is less critical, $\lambda = 4/1$ found to be optimal, and slower convergence observed for $\lambda < 1$.

Overall, with enough labeled data, slimIPL is robust to hyperparameter changes. When labeled data is limited, C, M and λ should be large enough to avoid over-fitting to labeled data.

5.7. Efficiency

Table 2 shows the reported training time of different semi- and unsupervised methods to fully converge. slimIPL has a clear advantage in training time and resource consumption. Among other things, we attribute this to the dynamic pseudo-label cache and the stability of the algorithm.

6. Conclusion

We revisit a key component of recent pseudo-labeling success in ASR, beam-search decoding with an LM, and propose slimIPL, in which a single model iteratively re-generates hard pseudo-labels with a dynamic cache to stabilize optimization. slimIPL is robust to hyperparameter changes and substantially simplifies training compared to other semi/unsupervised approaches, while delivering competitive performance for low-resource settings on LibriSpeech test sets. For inference, slimIPL is less prone to LM over-fitting than methods which use an LM for PL generation.

7. Acknowledgement

We thank Alex Rogozhnikov for insightful discussions about the algorithm and experiments, and Gil Keren for results discussions.

8. References

[1] O. Chapelle and B. Schölkopf, "Alexanderzien. semi-supervised learning," 2006.

[2] J. Kahn *et al.*, "Libri-light: A benchmark for asr with limited or no supervision," in *ICASSP*. IEEE, 2020, pp. 7669–7673.

[3] H. Scudder, "Probability of error of some adaptive pattern-recognition machines," *IEEE Transactions on Information Theory*, vol. 11, no. 3, pp. 363–371, 1965.

[4] W.-N. Hsu, A. Lee, G. Synnaeve, and A. Hannun, "Semi-supervised speech recognition via local prior matching," *arXiv preprint arXiv:2002.10336*, 2020.

[5] Q. Xu *et al.*, "Iterative pseudo-labeling for speech recognition," *Proc. Interspeech 2020*, pp. 1006–1010, 2020.

[6] ——, "Self-training and pre-training are complementary for speech recognition," *arXiv preprint arXiv:2010.11430*, 2020.

[7] J. Kahn, A. Lee, and A. Hannun, "Self-training for end-to-end speech recognition," in *ICASSP*. IEEE, 2020, pp. 7084–7088.

[8] G. Synnaeve *et al.*, "End-to-end asr: from supervised to semi-supervised learning with modern architectures," *arXiv preprint arXiv:1911.08460*, 2019.

[9] D. S. Park *et al.*, "Improved noisy student training for automatic speech recognition," *Proc. Interspeech 2020*, pp. 2817–2821, 2020.

[10] Y. Zhang *et al.*, "Pushing the limits of semi-supervised learning for automatic speech recognition," *arXiv preprint arXiv:2010.10504*, 2020.

[11] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[12] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan, "Billion-scale semi-supervised learning for image classification," *arXiv preprint arXiv:1905.00546*, 2019.

[13] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 687–10 698.

[14] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *33rd annual meeting of the association for computational linguistics*, 1995, pp. 189–196.

[15] D. McClosky, E. Charniak, and M. Johnson, "Effective self-training for parsing," in *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, 2006, pp. 152–159.

[16] R. Reichart and A. Rappoport, "Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007, pp. 616–623.

[17] Z. Huang and M. Harper, "Self-training pcfg grammars with latent annotations across languages," in *EMNLP*, 2009, pp. 832–841.

[18] J. He, J. Gu, J. Shen, and M. Ranzato, "Revisiting self-training for neural sequence generation," in *International Conference on Learning Representations*, 2020.

[19] N. Ueffing, "Using monolingual source-language data to improve mt performance," in *International Workshop on Spoken Language Translation (IWSLT) 2006*, 2006.

[20] J. Zhang and C. Zong, "Exploiting source-side monolingual data in neural machine translation," in *EMNLP*, 2016, pp. 1535–1545.

[21] S. Novotney and R. Schwartz, "Analysis of low-resource acoustic model self-training," in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[22] S. H. K. Parthasarathi and N. Strom, "Lessons from building acoustic models with a million hours of speech," in *ICASSP*. IEEE, 2019, pp. 6670–6674.

[23] J. Pino *et al.*, "Self-training for end-to-end speech translation," *Proc. Interspeech 2020*, pp. 1476–1480, 2020.

[24] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, no. 2, 2013.

[25] E. Arazo *et al.*, "Pseudo-labeling and confirmation bias in deep semi-supervised learning," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[26] Y. Chen, W. Wang, and C. Wang, "Semi-supervised asr by end-to-end self-training," *Proc. Interspeech 2020*, pp. 2787–2791, 2020.

[27] D. S. Park *et al.*, "Specaugment: A simple data augmentation method for automatic speech recognition," *Proc. Interspeech 2019*, pp. 2613–2617, 2019.

[28] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.

[29] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[30] C. Talnikar *et al.*, "Joint masked cpc and ctc training for asr," in *ICASSP*. IEEE, 2021, pp. 3045–3049.

[31] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 86–96.

[32] K. Sohn *et al.*, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[33] D. Berthelot *et al.*, "Mixmatch: A holistic approach to semi-supervised learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 5049–5059.

[34] ——, "Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring," *arXiv preprint arXiv:1911.09785*, 2019.

[35] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," *arXiv preprint arXiv:2012.12877*, 2020.

[36] K. Imamura, A. Fujita, and E. Sumita, "Enhancement of encoder and attention using target monolingual corpora in neural machine translation," in *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, 2018, pp. 55–63.

[37] S. Edunov, M. Ott, M. Auli, and D. Grangier, "Understanding back-translation at scale," in *EMNLP*, 2018, pp. 489–500.

[38] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "Rwth asr systems for librispeech: Hybrid vs attention," *Proc. Interspeech 2019*, pp. 231–235, 2019.

[39] S. Ling, Y. Liu, J. Salazar, and K. Kirchhoff, "Deep contextualized acoustic representations for semi-supervised speech recognition," in *ICASSP*. IEEE, 2020, pp. 6429–6433.

[40] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[41] A. Vaswani *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[42] A. Fan, E. Grave, and A. Joulin, "Reducing transformer depth on demand with structured dropout," in *International Conference on Learning Representations*, 2020.

[43] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of machine learning research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[44] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[45] T. Likhomanenko, G. Synnaeve, and R. Collobert, "Who needs words? lexicon-free speech recognition," *Proc. Interspeech 2019*, pp. 3915–3919, 2019.