Semi-Supervised Sequence Modeling with Cross-View Training: Appendix

Kevin Clark[1] **Minh-Thang Luong**[2] **Christopher D. Manning**[1] **Quoc V. Le**[2]
[1]Computer Science Department, Stanford University [2]Google Brain
kevclark@cs.stanford.edu, thangluong@google.com, manning@cs.stanford.edu, qvl@google.com

This document contains the appendix for the paper *Semi-Supervised Sequence Modeling with Cross-View Training*.

1 Detailed Results

We provide a more detailed version of the test set results in the paper, adding two decimals of precision, standard deviations of the 5 runs for each model, and more prior work, in Table 1.

2 Model Details

Our models use two layer CNN-BiLSTM encoders (Chiu and Nichols, 2016; Ma and Hovy, 2016; Lample et al., 2016) and task-specific prediction modules. See Section 3 of the paper for details. We provide a few minor details not covered in the paper below.

Sequence Tagging. For Chunking and Named Entity Recognition, we use a BIOES tagging scheme. We apply label smoothing (Szegedy et al., 2016; Pereyra et al., 2017) with a rate of 0.1 to the target labels when training on the labeled data.

Dependency Parsing. We omit punctuation from evaluation, which is standard practice for the PTB-SD 3.3.0 dataset. ROOT is represented with a fixed vector h_{ROOT} instead of using a vector from the encoder, but otherwise dependencies coming from ROOT are scored the same way as the other dependencies.

Machine Translation. We apply dropout to the output of each LSTM layer in the decoder. Our implementation is heavily based off of the Google NMT Tutorial[1] (Luong et al., 2017). We attribute our significantly better results to using pre-trained word embeddings, a character-level CNN, a larger model, stronger regularization, and better hyperparameter tuning. Target words occurring 5 or fewer times in the train set are replaced with a `UNK` token (but not during evaluation). We use a beam size of 10 when performing beam search. We found it slightly beneficial to apply label smoothing with a rate of 0.1 to the teacher's predictions (unlike our other tasks, the teacher only provides hard targets to the students for translation).

Multi-Task Learning. Several of our datasets are constructed from the Penn Treebank. However, we treat them as separate rather than providing examples labeled across multiple tasks to our model during supervised training. Furthermore, the Penn Treebank tasks do not all use the same train/dev/test splits. We ensure the training split of one task never overlaps the evaluation split of another by discarding the overlapping examples from the train sets.

Other Details. We apply dropout (Hinton et al., 2012) to the word embeddings and outputs of each Bi-LSTM. We use an exponential-moving-average (EMA) of the model weights from training for the final model; we found this to slightly improve accuracy and significantly reduce the variance in accuracy between models trained with different random initializations. The model is trained using SGD with momentum (Polyak, 1964; Sutskever et al., 2013). Word embeddings are initialized with GloVe vectors (Pennington et al., 2014) and fine-tuned during training. The full set of model hyperparameters are listed in Table 2.

Baselines. Baselines were run with the same architecture and hyperparameters as the CVT model. For the "word dropout" model, we randomly replace words in the input sentence with a `REMOVED` token with probability 0.1 (this value worked well on the dev sets). For Virtual Adversarial Training, we set the norm of the perturbation to be 1.5 for CCG, 1.0 for Dependency Parsing, and 0.5 for the other tasks (these values worked best on the

[1] https://github.com/tensorflow/nmt

dev sets). Otherwise, the implementation is as described in (Miyato et al., 2017a); we base our implementation off of their code.[2]. We were unable to successfully apply VAT to machine translation, perhaps because the student is provided hard targets for that task. For ELMo, we applied dropout to the ELMo embeddings before they are incorporated into the rest of the model. When training the multi-task ELMo model, each prediction module has its own set of softmax-normalized weights (s_j^{task} in (Peters et al., 2018)) for the ELMo emeddings going into the task-specific prediction modules. All tasks share the same s_j weights for the ELMo embeddings going into the shared Bi-LSTM encoder.

2.1 CVT for Image Recognition

Although the focus of our work is on NLP, we also applied CVT to image recognition and found it performs competitively with existing methods. Most of the semi-supervised image recognition approaches we compare against rely on the inputs being continuous, so they would be difficult to apply to text. More specifically, consistency regularization methods (Sajjadi et al., 2016; Laine and Aila, 2017; Miyato et al., 2017b) rely on adding continuous noise and applying image-specific transformations like cropping to inputs, GANs (Salimans et al., 2016; Wei et al., 2018) are very difficult to train on text due to its discrete nature, and mixup (Zhang et al., 2018; Verma et al., 2018) requires a way of smoothly interpolating between different inputs.

Approach. Our image recognition models are based on Convolutional Neural Networks, which produce a set of features $H(x_i) \in \mathbb{R}^{n \times n \times d}$ from an image x_i. The first two dimensions of H index into the spatial coordinates of feature vectors and d is the size of the feature vectors. For shallower CNNs, a particular feature vector corresponds to a region of the input image. For example, $H_{0,0}$ would be a d-dimensional vector of features extracted from the upper left corner. For deeper CNNs, a particular feature vector would be extracted from the whole image, but still only use a "region" of the representations from an earlier layer. The CNNs in our experiments are all in the first category.

The primary prediction layers of our CNNs take as input the mean of H over the first two dimensions, which results in a d-dimensional vector that is fed into a softmax layer:

$$p_\theta(y|x_i) = \text{softmax}(W\texttt{global\_average\_pool}(H) + b)$$

We add n^2 auxiliary prediction layers to the top of the CNN. The jth layer takes a single feature vector as input:

$$p_\theta^j(y|x_i) = \text{softmax}(W^j H_{\lfloor j/n \rfloor, j \bmod n} + b_j)$$

Data. We evaluated our models on the CIFAR-10 (Krizhnevsky and Hinton, 2009) dataset. Following previous work, we make the datasets semi-supervised by only using the provided labels for a subset of the examples in the training set; the rest are treated as unlabeled examples.

Model. We use the convolutional neural network from Miyato et al. (2017b), adapting their TensorFlow implementation[3]. Their model contains 9 convolutional layers and 2 max pooling layers. See Appendix D of Miyato et al.'s paper for more details.

We add 36 auxiliary softmax layers to the 6×6 collection of feature vectors produced by the CNN. Each auxiliary layer sees a patch of the image ranging in size from 21×21 pixels (the corner) to 29×29 pixels (the center) of the 32×32 pixel images. For some experiments, we combine CVT with standard consistency regularization by adding a perturbation (e.g., a small random vector) to the student's inputs when computing $\mathcal{L}_{\text{CVT}}$.

Results. The results are shown in Table 3. Unsurprisingly, adding continuous noise to the inputs works much better with images, where the inputs are naturally continuous, than with language. Therefore we see much better results from VAT on semi-supervised CIFAR-10 compared to on our NLP tasks. However, we still find incorporating CVT improves over models without CVT. Our CVT + VAT models are competitive with current start-of-the-art approaches. We found the gains from CVT are larger when no data augmentation is applied, perhaps because random translations of the input expose the model to different "views" in a similar manner as with CVT.

3 Negative Results

We briefly describe a few ideas we implemented that did not seem to be effective in initial experi-

[2] https://github.com/tensorflow/models/tree/master/research/adversarial_text

[3] https://github.com/takerum/vat_tf

ments. Note these findings are from early one-off experiments. We did not pursue them further after our first attempts did not pan out, so it is possible that some of these approaches could be effective with the proper adjustments and tuning.

- **Hard vs soft targets:** Classic self-training algorithms train the student model with one-hot "hard" targets corresponding to the teacher's highest probability prediction. In our experiments, this decreased performance compared to using soft targets. This finding is consistent with research on knowledge distillation (Hinton et al., 2015; Furlanello et al., 2018) where soft targets also work notably better than hard targets.

- **Confidence thresholding:** Classic self-training often only trains the student on a subset of the unlabeled examples on which the teacher has confident predictions (i.e., the output distribution has low entropy). We tried both "hard" (where the student ignores low-confidence examples) and "soft" (where examples are weighted according to the teacher's confidence) versions of this for training our models, but they did not seem to improve performance.

- **Mean Teacher:** The Mean Teacher method (Tarvainen and Valpola, 2017) tracks an exponential moving average (EMA) of model weights, which are used to produce targets for the students. The idea is that these targets may be better quality due to a self-ensembling effect. However, we found this approach to have little to no benefit in our experiments, although using EMA model weights at test time did improve results slightly.

- **Purely supervised CVT:** Lastly, we explored adding cross-view losses to purely supervised classifiers. We hoped that adding auxiliary softmax layers with different views of the input would act as a regularizer on the model. However, we found little to no benefit from this approach. This negative result suggests that the gains are from CVT are from the improved semi-supervised learning mechanism, not the additional prediction layers regularizing the model.

References

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *EMNLP*.

Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan Salakhutdinov. 2017. Good semi-supervised learning that requires a bad gan. In *NIPS*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *ICML*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Alex Krizhnevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *EACL*.

Samuli Laine and Timo Aila. 2017. Temporal ensembling for semi-supervised learning. In *ICLR*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *ACL*.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *HLT-NAACL*.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *TACL*.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower sequence labeling with task-aware neural language model. *arXiv preprint arXiv:1709.04109*.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *IWSLT*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNN-CRF. In *ACL*.

Xuezhe Ma and Eduard Hovy. 2017. Neural probabilistic model for non-projective mst parsing. In *IJCNLP*.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *ACL*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017a. Adversarial training methods for semi-supervised text classification. In *ICLR*.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2017b. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*.

Sungrae Park, Jun-Keon Park, Su-Jin Shin, and Il-Chul Moon. 2017. Adversarial dropout for supervised and semi-supervised learning. *arXiv preprint arXiv:1707.03631*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *ICLR*.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Boris T Polyak. 1964. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *NIPS*.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *NIPS*.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate sequence labeling with iterated dilated convolutions. In *EMNLP*.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *ICML*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

Antti Tarvainen and Harri Valpola. 2017. Weight-averaged consistency targets improve semi-supervised deep learning results. In *Workshop on Learning with Limited Labeled Data, NIPS*.

Vikas Verma, Alex Lamb, Christopher Beckham, Aaron Courville, Ioannis Mitliagkis, and Yoshua Bengio. 2018. Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer. *arXiv preprint arXiv:1806.05236*.

Xiang Wei, Zixia Liu, Liqiang Wang, and Boqing Gong. 2018. Improving the improved training of Wasserstein GANs. In *ICLR*.

Huijia Wu, Jiajun Zhang, and Chengqing Zong. 2017. Shortcut sequence tagging. *arXiv preprint arXiv:1701.00576*.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *ICLR*.

Method	CCG Acc.	Chunking F1	NER F1	FGN F1	POS Acc.	Dependency Parsing UAS	Dependency Parsing LAS	Translation BLEU
LSTM-CNN-CRF (Ma and Hovy, 2016)			91.21		97.55			
LSTM-CNN (Chiu and Nichols, 2016)			91.62 ± 0.33	86.28 ± 0.26				
ID-CNN-CRF (Strubell et al., 2017)			90.65 ± 0.15	86.84 ± 0.19				
Tri-Trained LSTM (Lewis et al., 2016)	94.7							
Shortcut LSTM (Wu et al., 2017)	95.08				97.53			
JMT* (Hashimoto et al., 2017)		95.77			97.55	94.67	92.90	
LM-LSTM-CNN-CRF (Liu et al., 2017)		95.96 ± 0.08	91.71 ± 0.10		97.53 ± 0.03			
TagLM† (Peters et al., 2017)		96.37 ± 0.05	91.93 ± 0.19					
ELMo† (Peters et al., 2018)			92.22 ± 0.10					
NPM (Ma and Hovy, 2017)						94.9	93.0	
Deep Biaffine (Dozat and Manning, 2017)						95.74	94.08	
Stack Pointer (Ma et al., 2018)						95.87	94.19	
Stanford (Luong and Manning, 2015)								23.3
Google (Luong et al., 2017)								26.1
Supervised	94.94 ± 0.02	95.10 ± 0.06	91.16 ± 0.09	87.48 ± 0.08	97.60 ± 0.02	95.08 ± 0.03	93.27 ± 0.03	28.88 ± 0.12
Virtual Adversarial Training*	95.07 ± 0.04	95.06 ± 0.06	91.75 ± 0.10	87.91 ± 0.11	97.64 ± 0.03	95.44 ± 0.06	93.72 ± 0.07	–
Word Dropout*	95.20 ± 0.04	95.79 ± 0.08	92.14 ± 0.11	88.06 ± 0.09	97.66 ± 0.01	95.56 ± 0.05	93.80 ± 0.08	29.33 ± 0.10
ELMo*	95.79 ± 0.04	96.50 ± 0.03	92.24 ± 0.09	88.49 ± 0.12	97.72 ± 0.01	96.22 ± 0.05	94.44 ± 0.06	29.34 ± 0.11
ELMo + Multi-task*†	95.91 ± 0.05	96.83 ± 0.03	92.32 ± 0.12	88.37 ± 0.16	**97.79 ± 0.03**	96.40 ± 0.04	94.79 ± 0.05	–
CVT*	95.65 ± 0.04	96.58 ± 0.04	92.34 ± 0.06	88.68 ± 0.14	97.70 ± 0.03	95.86 ± 0.03	94.06 ± 0.02	**29.58 ± 0.07**
CVT + Multi-Task*†	95.97 ± 0.04	96.85 ± 0.05	92.42 ± 0.08	88.42 ± 0.13	97.76 ± 0.02	96.44 ± 0.04	94.83 ± 0.06	–
CVT + Multi-Task + Large*†	**96.05 ± 0.03**	**96.98 ± 0.05**	**92.61 ± 0.09**	**88.81 ± 0.09**	97.74 ± 0.02	**96.61 ± 0.04**	**95.02 ± 0.04**	–

Table 1: Results on the test sets for all tasks. We report the means and standard deviations of 5 runs. The +Larger model has four times as many hidden units as the others, making it similar in size to the models when ELMo is included. For dependency parsing, we omit results from Choe and Charniak (2016), Kuncoro et al. (2017), and Liu and Zhang (2017) because these train constituency parsers and convert the system outputs to dependency parses. They produce higher scores, but have access to more information during training and do not apply to datasets without constituency annotations. * denotes semi-supervised and † denotes multi-task.

Parameter	Value
Word Embeddings Initializiation	300d GloVe 6B
Character Embedding Size	50
Character CNN Filter Widths	[2, 3, 4]
Character CNN Num Filters	300 (100 per filter width)
Encoder LSTM sizes	1024 for the first layer, 512 for the second one
Encoder LSTM sizes, “Large” model	4096 for the first layer, 2048 for the second one
LSTM projection layer size	512
Hidden layer sizes	512
Dropout	0.5 for labeled examples, 0.8 for unlabeled examples
EMA coefficient	0.998
Learning rate	$0.5/(1 + 0.005t^{0.5})$ (t is number of SGD updates so far)
Momentum	0.9
Batch size	64 sentences

Table 2: Hyperparameters for the model.

Method	CIFAR-10	CIFAR-10+
	4000 labels	
GAN (Salimans et al., 2016)	–	18.63 ± 2.32
Stochastic Transformations (Sajjadi et al., 2016)	–	11.29 ± 0.24
Π model (Laine and Aila, 2017)	16.55 ± 0.29	12.36 ± 0.31
Temporal Ensemble (Laine and Aila, 2017)	–	12.16 ± 0.24
Mean Teacher (Tarvainen and Valpola, 2017)	–	12.31 ± 0.28
Complement GAN (Dai et al., 2017)	14.41 ± 0.30	–
VAT (Miyato et al., 2017b)	13.15	10.55
VAdD (Park et al., 2017)	–	11.68 ± 0.19
VAdD + VAT (Park et al., 2017)	–	$\mathbf{10.07 \pm 0.11}$
SNGT + Π model (Luong et al., 2017)	13.62 ± 0.17	11.00 ± 0.36
SNGT + VAT (Luong et al., 2017)	$\mathbf{12.49 \pm 0.36}$	$\mathbf{9.89 \pm 0.34}$
Consistency + WGAN (Wei et al., 2018)	–	$\mathbf{9.98 \pm 0.21}$
Manifold Mixup (Verma et al., 2018)	–	$\mathbf{10.26 \pm 0.32}$
Supervised	23.61 ± 0.60	19.61 ± 0.56
VAT (ours)	13.29 ± 0.33	10.90 ± 0.31
CVT, no input perturbation	14.63 ± 0.20	12.44 ± 0.27
CVT, random input perturbation	13.80 ± 0.30	11.10 ± 0.26
CVT, adversarial input perturbation	$\mathbf{12.01 \pm 0.11}$	$\mathbf{10.11 \pm 0.15}$

Table 3: Error rates on semi-supervised CIFAR-10. We report means and standard deviations from 5 runs. CIFAR-10+ refers to results where data augmentation (random translations of the input image) was applied. For some of our models we add a random or adversarially chosen perturbation to the student model's inputs, which is done in most consistency regularization methods.