INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Virtual Adversarial Training Applied to Neural Higher-Order Factors for Phone Classification

Martin Ratajczak[1], *Sebastian Tschiatschek*[2], *Franz Pernkopf*[1]

[1]Graz University of Technology, Signal Processing and Speech Communication Laboratory
[2]ETH Zurich, Learning & Adaptive Systems Group

martin.ratajczak@tugraz.at, sebastian.tschiatschek@inf.ethz.ch, pernkopf@tugraz.at

Abstract

We explore virtual adversarial training (VAT) applied to neural higher-order conditional random fields for sequence labeling. VAT is a recently introduced regularization method promoting local distributional smoothness: It counteracts the problem that predictions of many state-of-the-art classifiers are unstable to adversarial perturbations. Unlike random noise, adversarial perturbations are minimal and bounded perturbations that flip the predicted label. We utilize VAT to regularize neural higher-order factors in conditional random fields. These factors are for example important for phone classification where phone representations strongly depend on the context phones. However, without using VAT for regularization, the use of such factors was limited as they were prone to overfitting. In extensive experiments, we successfully apply VAT to improve performance on the TIMIT phone classification task. In particular, we achieve a phone error rate of 13.0%, exceeding the state-of-the-art performance by a wide margin.

Index Terms: Virtual adversarial training, local distributional smoothing, deep higher-order factors, neural higher-order conditional random field, phone classification

1. Introduction

In *sequence labeling*, an output label sequence $\mathbf{y}$ is assigned to some given input sequence $\mathbf{x}$. First-order linear-chain conditional random fields (LC-CRFs) are established models for sequence labeling [1], e.g. speech recognition [2]. Due to several advantages, LC-CRFs often achieve better performance compared to their generative counterparts [1], i.e. hidden Markov models (HMMs). Compared to HMMs and first-order LC-CRFs, *higher-order LC-CRFs (HO-LC-CRFs)* are more expressive by allowing for arbitrary input-independent (such factors depend on the output labels only) [3] and input-dependent (such factors depend on both the input and output variables) higher-order factors [4, 5]. That is, both types of higher-order factors can depend on more than two output labels. Such factors are for example important for phone classification where phone representations strongly depend on the context phones.

Unfortunately, the model complexity of higher-order CRFs increases exponentially with the number of the output variables considered in higher order factors [6]. Furthermore, the overfitting problem is more serious. Besides many other approaches it has been suggested to parametrize the factors in LC-CRFs by *neural models* for first-order factors [7, 8, 9, 10, 11] and for *higher-order* factors [12, 13]. However, without using an appropriate regularizer, the use of such factors was limited because they were prone to overfitting [12, 13]. For instance, using only ℓ_2 regularization, the higher order factors easily overfit and their expressive power cannot be leveraged.

We address the overfitting problem by applying virtual adversarial training (VAT) [14] to HO-LC-CRFs. VAT is a recently introduced regularization method promoting local distributional smoothness, i.e. it counteracts the problem that predictions of many state-of-the-art classifiers are vulnerable to adversarial perturbations. In contrast to random noise, adversarial perturbations are minimal and bounded perturbations that flip the predicted label. These perturbations are determined as the solution of an optimization problem. In contrast to adversarial training (AT) [15], VAT is not using label information but depends only on the posterior distribution in the proximity of the input samples. We use VAT to regularize neural higher-order factors in conditional random fields.

Additionally we used the following two *techniques* to improve our results: Firstly, batch normalization [16] which proved itself to be useful in conjunction with VAT in several image recognition tasks [14]. Secondly, a novel modeling approach for neural higher-order factors: A fully-connected multi-layer sub-network is convolved over several segments sharing its weights. The resulting hidden activations are combined by additional hidden layers followed by the output layer predicting a sub-sequence of labels.

Our main contributions are: (i) We utilize VAT to regularize neural higher-order factors in CRFs. Without using this novel regularizer, the use of such factors was limited due to overfitting. (ii) For modeling the neural higher-order factors, we introduce a multi-layer sub-network convolved over several segments. The resulting hidden activations are concatenated and combined by additional hidden layers. This network (without sequential modeling) is already achieving a phone error rate (PER) of 16.8% on the TIMIT phone classification task. (iii) In extensive experiments, we successfully apply VAT and the novel modeling to improve performance on phone classification. In particular, we achieve 13.0% PER — the best reported performance on this task.

This paper is structured as follows: In Section 2 and 3 we recap neural HO-LC-CRFs and VAT, respectively. In Section 4 we introduce our novel modeling of the higher-order factors. In Section 5 we evaluate our model on the TIMIT phone classification task. Finally, we conclude the paper in Section 6.

http://dx.doi.org/10.21437/Interspeech.2016-832

2. Neural Higher-Order Conditional Random Fields

We briefly recap neural HO-LC-CRFs (NHO-LC-CRFs) for sequence labeling [12, 13]. The NHO-LC-CRF defines a conditional distribution

$$p^{\text{CRF}}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \prod_{n=1}^{N} \Phi_t(\mathbf{y}_{t-n+1:t}; \mathbf{x}), \tag{1}$$

for an output sequence $\mathbf{y}$ of length T given an input sequence $\mathbf{x}$ of length T, where $\Phi_t(\mathbf{y}_{t-n+1:t}; \mathbf{x})$ are non-negative factors that can depend on the label sub-sequence $\mathbf{y}_{t-n+1:t}$ and the full input sequence $\mathbf{x}$, and where $Z(\mathbf{x})$ is an input-dependent normalization computed as

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{t=1}^{T} \prod_{n=1}^{N} \Phi_t(\mathbf{y}_{t-n+1:t}; \mathbf{x}). \tag{2}$$

The factors in $(\mathrm{N}-1)^{\text{th}}$-order CRFs can depend on label sub-sequences of maximal span N. All factors $\Phi_t(\mathbf{y}_{t-n+1:t}; \mathbf{x})$ are assumed to be given in log-linear form, i.e.

$$\Phi_t(\mathbf{y}_{t-n+1:t}; \mathbf{x}) = \exp\left(\sum_k \mathbf{w}_k^{t,n} \mathbf{f}_k(\mathbf{y}_{t-n+1:t}; t, \mathbf{x})\right), \tag{3}$$

where $\mathbf{f}_k(\mathbf{y}_{t-n+1:t}; t, \mathbf{x})$ are arbitrary vector-valued and (possibly) position-dependent feature functions and $\mathbf{w}_k^{t,n}$ are weights. These feature functions can for example be simple indicator functions, linear functions, or functions computed using neural networks as in this work. We distinguish the following types of feature functions:

n-gram input-independent features. These features are observation- and position-independent, i.e. $\mathbf{f}_k(\mathbf{y}_{t-n+1:t}; t, \mathbf{x}) = \mathbf{f}^n(\mathbf{y}_{t-n+1:t})$. Each entry of the vectors corresponds to the indicator function of an instance of a label sub-sequence $\mathbf{a}_i$, i.e. $\mathbf{f}^n(\mathbf{y}_{t-n+1:t}) = [\mathbf{1}(\mathbf{y}_{t-n+1:t} = \mathbf{a}_1), \mathbf{1}(\mathbf{y}_{t-n+1:t} = \mathbf{a}_2), \ldots]^T$. Typically $\mathbf{a}_1, \mathbf{a}_2, \ldots$ enumerate all possible label sub-sequences of length n. For the case $n = 2$, these functions are also called transition functions. In Figure 1, these functions correspond to the factors Φ^n.

m-n-gram input-dependent MLP features. These features generalize local factors to longer label sub-sequences. In this way, these feature functions can depend on the label sub-sequence $\mathbf{y}_{t-n+1:t}$ and an input sub-sequence of $\mathbf{x}$. In this paper, we use these features in the form $\mathbf{f}^{m\text{-}n}(\mathbf{y}_{t-n+1:t}; t, \mathbf{x}) = [\mathbf{1}(\mathbf{y}_{t-n+1:t} = \mathbf{a}_1)\, \mathbf{g}^m(\mathbf{x}, t), \ldots]^T$, where we use MLP networks for $\mathbf{g}^m(\mathbf{x}, t)$, enabling us to model complex interactions among the input variables. Hence, these functions map an input sub-sequence into a new feature space, i.e. $\mathbf{g}^m(\mathbf{x}, t)$ considers only a contextual window of the input around position t. Specifically, the hidden activations of the last layer $\mathbf{h}^m(\mathbf{x}, t)$ of an MLP network are used, i.e. $\mathbf{g}^m(\mathbf{x}, t) = \mathbf{h}^m(\mathbf{x}, t)$. We call these features *m-n-gram MLP features*. They correspond to the factors $\Phi^{m\text{-}n}$ in Figure 1, assuming that they only depend on input-output sub-sequences. These features extract an input sub-sequence of length m aligned and centered with the considered output sub-sequence.

2.1. Parameter Learning

The parameters $\mathbf{w} = \{\mathbf{w}_k^{t,n} \mid \forall k, t, n\}$ are optimized to maximize the conditional log-likelihood of the training-data, i.e.

$$\mathcal{F}(\mathbf{w}) = \sum_{j=1}^{J} \log p^{\text{CRF}}(\mathbf{y}^{(j)} \mid \mathbf{x}^{(j)}), \tag{4}$$

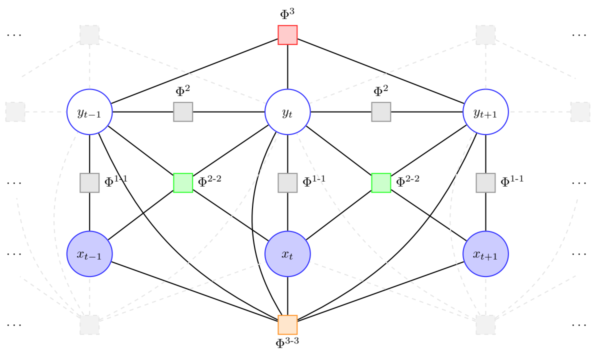

Figure 1: Factor graph of HO-LC-CRFs using input-dependent uni-gram factors $\Phi^{1\text{-}1}$ and bi-gram transition factors Φ^2 (typical) and additionally 3-gram factors Φ^3 as well as input-dependent factors $\Phi^{2\text{-}2}$ and $\Phi^{3\text{-}3}$.

where $((\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(J)}, \mathbf{y}^{(J)}))$ is a collection of J input-output sequence pairs drawn i.i.d. from some unknown data distribution. The partial derivatives of (4) with respect to the weights $\mathbf{w}_k^{t,n} = \mathbf{w}_k^n$ (parameters are shared across time) can be computed as described in [3, 4]. We compute the conditional log-likelihood by computing the forward recursion as described in Section 2.2. Then we utilize back-propagation [17] and automatic differentiation [18] as common in neural networks to compute the gradients and update the weights.

2.2. Forward Algorithm for 2nd-order CRFs

The main ingredient needed for applying the back-propagation algorithm is the forward recursion and the computation of the normalization constant. For a given input-output sequence pair $(\mathbf{x}, \mathbf{y})$, the forward recursion is given in terms of quantities $\alpha_t(\mathbf{y}_{t-1:t})$ that are updated according to

$$\begin{aligned}\alpha_t(\mathbf{y}_{t-1:t}) =& \Phi_t(y_t; \mathbf{x}) \Phi_t(\mathbf{y}_{t-1:t}; \mathbf{x}) \times \\ & \sum_{y_{t-2}} \Phi_t(\mathbf{y}_{t-2:t}; \mathbf{x}) \alpha_{t-1}(\mathbf{y}_{t-2:t-1}).\end{aligned} \tag{5}$$

The recursion is initialized as $\alpha_2(\mathbf{y}_{1:2}) = \Phi_2(y_2; \mathbf{x}) \Phi_1(\mathbf{y}_{1:2}; \mathbf{x}) \Phi_1(y_1; \mathbf{x})$. Finally, the normalization constant can be computed as $Z(\mathbf{x}) = \sum_{\mathbf{y}_{T-1:T}} \alpha_T(\mathbf{y}_{T-1:T})$. The most probable label sequence can be found by the Viterbi algorithm generalized to HO-LC-CRFs: The summation in the forward recursion is replaced by the maximum operation. At the end of the recursion, we identify the most probable state at the last position and apply back-tracking. For details and for time complexities we refer to [3, 4].

2.3. Pre-Training of the m-n-gram MLPs

Before training the CRF, we pre-trained MLP networks for classifying label sub-sequences given corresponding input sub-sequences. For this purpose we defined new datasets $\mathcal{D}_{m\text{-}n}$ which, informally, contain all possible sub-sequences of the training data of length n (on the output side) and the corresponding input sub-sequences of length m. Formally,

$$\mathcal{D}_{m\text{-}n} = \left\{ (\mathbf{x}_{t-n-w+1:t+w}^{(j)}, \mathbf{y}_{t-n+1:t}^{(j)}) \right\}_{j=1\ldots J,\ t=n\ldots T_j},$$

where T_j is the length of the j^{th} input sequence and $w = \frac{m-n}{2}$ (we assume that $m \geq n$ and that w is an integer). Whenever a subscript to $\mathbf{x}$ becomes negative or is larger than the length of the input sequence, we fill in zeros. For pre-training the m-n gram MLP networks, we maximized the log conditional likelihood $\sum_{(\mathbf{x}^r,\mathbf{y}^r)\in\mathcal{D}_{m\text{-}n}} p^{\text{MLP}}(\mathbf{y}^r|\mathbf{x}^r)$. Note that we train one network for each factor $\Phi^{m\text{-}n}$.

3. Virtual Adversarial Training

For pre-training of the m-n-gram MLP features, we make use of a novel regularization technique named *virtual adversarial training* (VAT) [14]. The simple yet well motivated idea behind VAT is to promote local smoothness of the posterior distribution $p(\mathbf{y}|\mathbf{x})$ with respect to $\mathbf{x}$, i.e. the posterior distribution should only vary minimally for small and bounded perturbations of the input $\mathbf{x}$. By incorporating this idea into the training of the m-n-gram MLP features, the corresponding classifiers (and thus also the MLP features) become more robust and, as substantiated in experiments, generalize better.

We now introduce VAT formally. We compute the adversarial perturbation $\boldsymbol{\delta}^r$ by maximizing the Kullback-Leibler divergence KL $(\cdot||\cdot)$ of the posterior distribution for unperturbed and perturbed inputs, i.e.

$$\boldsymbol{\delta}^r = \arg\max_{||\boldsymbol{\delta}||\leq\epsilon} \text{KL}\left(p(\mathbf{y}|\mathbf{x}^r)||p(\mathbf{y}|\mathbf{x}^r+\boldsymbol{\delta})\right), \qquad (6)$$

where $\epsilon > 0$ limits the maximal perturbation and $p = p^{\text{MLP}}$. Informally, $\boldsymbol{\delta}^r$ is a minimal and bounded perturbation, i.e. the perturbed input $\mathbf{x}^r + \boldsymbol{\delta}$ is within a radius of ϵ around $\mathbf{x}^r$ and maximally changes the posterior distribution. The smaller $\text{KL}\left(p(\mathbf{y}|\mathbf{x}^r)||p(\mathbf{y}|\mathbf{x}^r+\boldsymbol{\delta}^r)\right)$, the smoother is the posterior distribution around $\mathbf{x}^r$. Now we intend to use the Kullback-Leibler divergence as regularizer. Thus instead of maximizing the conditional likelihood $\sum_r \log p(\mathbf{y}^r|\mathbf{x}^r)$, where r is the index of the r^{th} training example in $\mathcal{D}_{m\text{-}n}$, we maximize the regularized objective

$$\sum_r \log p(\mathbf{y}^r|\mathbf{x}^r) - \lambda \sum_r \text{KL}\left(p(\mathbf{y}|\mathbf{x}^r)||p(\mathbf{y}|\mathbf{x}^r+\boldsymbol{\delta}^r)\right), \quad (7)$$

where λ is a trade-off parameter. Optimization of (6) and (7) can be performed efficiently [14].

VAT has two hyper-parameters that need to be adjusted, the trade-off parameter λ and the radius ϵ. The trade-off parameter weights the (competing) goals of maximizing likelihood with the goal of having smooth posteriors around the training samples. The radius ϵ specifies the *range* within which the posterior should be smooth around the training data.

Note that the term $\text{KL}\left(p(\mathbf{y}|\mathbf{x}^r)||p(\mathbf{y}|\mathbf{x}^r+\boldsymbol{\delta}^r)\right)$ in (7) does not depend on the true label of $\mathbf{x}^r$ but only on the distribution over the labels. This makes VAT applicable in semi-supervised scenarios, e.g. we could basically make use of untranscribed utterances for improved pre-training of the m-n-gram MLPs. In contrast to VAT, AT makes use of label information as follows: Instead of identifying a local perturbation that maximally changes the posterior distribution in terms of Kullback-Leibler divergence, it identifies a local perturbation that minimizes the posterior probability of the correct label, i.e.

$$\boldsymbol{\delta}^{r,\text{AT}} = \arg\min_{||\boldsymbol{\delta}||\leq\epsilon} p(\mathbf{y}^r|\mathbf{x}^r+\boldsymbol{\delta}). \qquad (8)$$

Furthermore, the objective that is maximized for AT is

$$\sum_r \log p(\mathbf{y}^r|\mathbf{x}^r) + \lambda \sum_r \log p(\mathbf{y}^r|\mathbf{x}^r+\boldsymbol{\delta}^{r,\text{AT}}). \qquad (9)$$

4. Convolution over Segments

We introduce a slight modification to the network structure of the m-n-gram MLP networks. This modification can be seen as convolution along the segments of the input sequence $\mathbf{x}$, i.e. a sub-network is processing the input vectors at segment level and acts as a filter. The output of the sub-networks of m input segments is further passed through one or more hidden layers and, finally, mapped to the posterior of the output label sub-sequence of length n by a softmax layer. A similar structure has been successfully used for phone recognition [19]. However, in [19], the convolution has been applied on frame level and it was not used for neural higher-order factors.

5. Experiments

We evaluated the performance of the proposed models on the TIMIT phone classification task.

5.1. TIMIT Data Set

The TIMIT data set [20] contains recordings of 5.4 hours of English speech from 8 major dialect regions of the United States. The recordings were manually segmented at phone level. We use this segmentation for phone classification. Note that phone classification should not be confused with phone recognition [21] where no segmentation is provided. We collapsed the original 61 phones into 39 phones. All frames of MFCC, delta and double-delta coefficients of a phonetic segment are mapped into one feature vector. Details on pre-processing and data set are presented in [22]. The task is, given an utterance and a corresponding segmentation, to infer the phone within every segment. The development set is used for parameter tuning. The performance measure is the phone error rate (PER) in [%].

5.2. Experimental Setup

For training the neural networks we used the ADAM optimizer [23] with a batch size of 100, an initial learning rate of 0.002, learning rate decay of 0.9 and no ℓ_2-norm regularizer for all models with batch normalization and for the baseline networks (MLP+L2) we used 10, 0.0001, 0.9 and 0.0001, respectively. In both cases, the number of epochs was 100. Optimization of the HO-LC-CRF weights was in all cases performed using stochastic gradient ascent using a batch size of one sample, an initial learning rate of 0.001, learning rate decay of 0.998, a momentum of 0.0001 and a maximal number of epochs of 500. An ℓ_2-norm regularizer on the model weights was used with a fixed regularization factor of 0.001. We utilized early stopping based on the development data set. Further, the last hidden activations of the pre-trained networks were normalized to zero mean and unit standard deviation.

5.3. Labeling Results for Proposed Models

Network Architectures and Regularizers. We trained MLP networks for several m-n-gram settings, i.e for m input segments and label sub-sequence of length n. The activation functions in the hidden layers and output layer are rectifier and softmax functions, respectively. In the following the numbers of hidden units per hidden layer is denoted by a separating hyphen. In all cases we trained the following network sizes {150, 500-150, 500-300-150, 500-300-200-150, 500-400-300-200-150}. We compared objective functions with three different regularizers: ℓ_2-norm (L2), adversarial training (AT) and virtual ad-

Table 1: *TIMIT Phone Classification:* Labeling results for (top) isolated phone classification using $m = 3$ input segments and one output label $n = 1$ and (bottom) for NHO-LC-CRFs. Performance measure: Phone error rate (PER) in [%].

Neural Network	MLP+L2		MLP-BN+AT		MLP-BN+VAT		MLPCS-BN+VAT	
Hidden Layer Sizes	valid	test	valid	test	valid	test	valid	test
150	21.56	22.37	22.74	23.04	20.82	21.71	18.09	18.34
500-150	**19.45**	**20.16**	**17.90**	**18.91**	17.92	**18.16**	16.69	17.39
500-300-150	19.96	20.44	18.06	18.50	18.11	18.79	16.78	17.12
500-300-200-150	20.92	21.74	18.00	**18.27**	**17.81**	**18.63**	**16.39**	**16.80**
500-400-300-200-150	20.90	21.65	18.48	19.42	18.16	18.54	16.70	17.43

NHO-LC-CRF	MLP+L2		MLP-BN+AT		MLP-BN+VAT		MLPCS-BN+VAT	
Higher-Order Factors	valid	test	valid	test	valid	test	valid	test
$\Phi^1, \Phi^2, \Phi^{1\text{-}1}$	19.83	20.55	20.40	21.21	19.66	20.32	-	-
$+\Phi^{3\text{-}1}$	17.43	18.16	16.98	17.42	17.08	17.40	15.47	16.13
$+\Phi^{2\text{-}2}$	15.62	16.46	14.90	15.75	15.07	15.63	14.62	14.91
$+\Phi^{4\text{-}2}$	15.26	15.79	14.29	14.72	14.47	14.88	14.00	14.30
$+\Phi^3, \Phi^{3\text{-}3}$	14.59	15.12	13.72	14.21	13.93	14.31	13.50	13.21
$+\Phi^{5\text{-}3}$	14.66	14.81	13.68	14.02	**13.78**	**13.90**	**13.22**	**13.04**

versarial training (VAT) based on finite difference approximation. After preliminary experiments we fixed the optimal perturbation strength ϵ to 2.1 and the trade-off parameter λ to 1.0 for AT and VAT. We used three different neural network architectures: Multilayer-perceptron network (MLP), MLP network with batch normalization (MLP-BN) and MLP with convolution over segments and with batch normalization (MLPCS-BN). In the case of MLPCS-BN we used the network sizes from above for the sub-networks, and we added an additional hidden layer with 150 hidden units combining the concatenated outputs of the sub-networks.

Isolated Classification. In Table 1 top panel we report labeling results for isolated phone classification of the 3-1-gram setting for the mentioned hidden layer sizes and for four different network architectures and regularizers: MLP+L2, MLP-BN+AT, MLP-BN+VAT and MLPCS-BN+VAT. The standard MLP network with ℓ_2-norm regularizer (MLP+L2) starts to overfit for more than two hidden layers and gives the poorest results. Including BN+AT and BN+VAT improved the PER, significantly. The best PER result we achieved with a single network was 16.8% including additional convolutions over segments in the MLPs.

Sequence Labeling. As described in Section 2.3, we discriminatively pre-trained m-n-gram MLP factors. After pre-training, we normalized the activations of the last hidden layer of the MLP networks to zero mean and unit variance and used them as features to train NHO-LC-CRFs. For training of NHO-LC-CRFs we considered again four different configurations of the neural network architectures and regularizers: MLP+L2, MLP-BN+AT, MLP-BN+VAT and MLPCS-BN+VAT. We chose the best hidden layer sizes of the corresponding m-n-gram MLP factors based on the validation PER of the single networks. Incrementally, we included more m-n-gram MLP factors to the NHO-LC-CRFs. In Table 1 bottom panel we report labeling results. Additional higher-order factors (plus sign indicates additional factors to the model of previous line) improved consistently the PER. As for the single networks BN+AT improved the results as well as BN+VAT. We achieved our best PER result of 13.0% including additional convolutions over segments in the neural higher-order factors (MLPCS-BN+VAT).

Summary. Finally, we compared our best result to other state-of-the-art methods based on MFCC features as shown in Table 2 and to deep scattering spectrum [24], a method based on more advanced preprocessing which in combination with support vector machines achieves state-of-the-art performance of 15.9%. The NHO-LC-CRF using standard MFCC features achieves a performance of 15.8%. In Table 1 bottom panel we improved this baseline and achieved a performance of 14.8% by using ADAM optimizer [23] and including additional higher-order factors (MLP+L2). NHO-LC-CRF including BN, VAT and convolution over segments (MLPCS-BN+VAT) achieved 13.0% PER — the best reported performance on this task.

Table 2: *TIMIT Phone Classification:* Summary of labeling results. Performance measure: Phone error rate (PER) in [%].

Model	PER [%]
GMMs ML [25]	25.9
HCRFs [26]	21.5
Large-Margin GMM [25]	21.1
Heterogeneous Measurements [22]	21.0
CNF [12]	20.7
NHO-LC-CRF [13]	17.7
Deep Scattering Spectrum [24]	**15.9**
NHO-LC-CRF (disc. pre-training) [12]	**15.8**
Proposed models in this paper:	
MLPCS-BN+VAT (Isolated)	**16.8**
NHO-LC-CRF + MLP-BN+VAT	**13.9**
NHO-LC-CRF + MLPCS-BN+VAT	**13.0**

6. Conclusion

We utilized virtual adversarial training to regularize the neural higher-order factors in HO-LC-CRFs for sequence labeling. Without using this novel regularization, the use of such factors was limited due to overfitting. Furthermore, we introduced a novel approach for modeling higher-order factors which can be seen as convolution along the segments of the input sequence. In experiments, we demonstrated excellent performance of our approach on the TIMIT phone classification task, reporting a new state-of-the-art performance of 13.0% phone error rate.

7. References

[1] J. Lafferty, A. McCallum, and F. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in *International Conference on Machine Learning (ICML)*, 2001, pp. 282–289.

[2] A. Gunawardana, M. Mahajan, A. Acero, and J. C. Platt, “Hidden conditional random fields for phone classification,” in *Interspeech*, 2005, pp. 1117–1120.

[3] N. Ye, W. S. Lee, H. L. Chieu, and D. Wu, “Conditional random fields with high-order features for sequence labeling,” in *Neural Information Processing Systems (NIPS)*, 2009, pp. 2196–2204.

[4] X. Qian, X. Jiang, Q. Zhang, X. Huang, and L. Wu, “Sparse higher order conditional random fields for improved sequence labeling,” in *International Conference on Machine Learning (ICML)*, 2009, pp. 849–856.

[5] T. Lavergne, A. Allauzen, J. M. Crego, and F. Yvon, “From n-gram-based to CRF-based Translation Models,” in *Workshop on Statistical Machine Translation*, 2011, pp. 542–553.

[6] L. Stewart, X. He, and R. S. Zemel, “Learning flexible features for conditional random fields.” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 8, pp. 1415–1426, 2008.

[7] H. Larochelle and Y. Bengio, “Classification using discriminative restricted Boltzmann machines,” in *International Conference on Machine Learning (ICML)*, 2008, pp. 536–543.

[8] J. Peng, L. Bo, and J. Xu, “Conditional neural fields,” in *Neural Information Processing Systems (NIPS)*, 2009, pp. 1419–1427.

[9] R. Prabhavalkar and E. Fosler-Lussier, “Backpropagation training for multilayer conditional random field based phone recognition.” in *International Conference on Acoustics Speech and Signal Processing (ICASSP)*, 2010, pp. 5534–5537.

[10] L. van der Maaten, M. Welling, and L. K. Saul, “Hidden-unit conditional random fields.” in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 479–488.

[11] M. Ratajczak, S. Tschiatschek, and F. Pernkopf, “Sum-product networks for structured prediction: Context-specific deep conditional random fields,” in *International Conference on Machine Learning (ICML) Workshop on Learning Tractable Probabilistic Models Workshop*, 2014.

[12] ——, “Neural higher-order factors in conditional random fields for phoneme classification,” in *Interspeech*, 2015.

[13] ——, “Structured regularizer for neural higher-order sequence models,” in *European Conference on Machine Learning (ECML)*, 2015.

[14] T. Miyato, M. Koyama, K. Nakae, and S. Ishii, “Distributional smoothing with virtual adversarial training,” in *International Conference on Learning Representations (ICLR)*, 2016.

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *ArXiv e-prints*, 2015.

[16] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *ArXiv e-prints*, 2015.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1,” D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362.

[18] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, “Theano: a CPU and GPU math expression compiler,” in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, jun 2010, oral Presentation.

[19] L. Tóth, “Combining time- and frequency-domain convolution in convolutional neural network-based phone recognition,” in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014*, 2014, pp. 190–194.

[20] V. Zue, S. Seneff, and J. R. Glass, “Speech database development at MIT: Timit and beyond,” *Speech Communication*, vol. 9, no. 4, pp. 351–356, 1990.

[21] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition,” *Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[22] A. K. Halberstadt and J. R. Glass, “Heterogeneous acoustic measurements for phonetic classification,” in *EUROSPEECH*, 1997, pp. 401–404.

[23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *International Conference on Learning Representations (ICLR)*, 2015.

[24] J. Andn and S. Mallat, “Deep scattering spectrum,” *CoRR*, vol. abs/1304.6763, 2013.

[25] F. Sha and L. Saul, “Large margin Gaussian mixture modeling for phonetic classification and recognition,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2006, pp. 265–268.

[26] Y.-H. Sung, C. Boulis, C. Manning, and D. Jurafsky, “Regularization, adaptation, and non-independent features improve hidden conditional random fields for phone classification,” in *Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2007, pp. 347–352.