END-TO-END TRAINING APPROACHES FOR DISCRIMINATIVE SEGMENTAL MODELS

Hao Tang, Weiran Wang, Kevin Gimpel, Karen Livescu

Toyota Technological Institute at Chicago

{haotang, weiranwang, kgimpel, klivescu}@ttic.edu

ABSTRACT

Recent work on discriminative segmental models has shown that they can achieve competitive speech recognition performance, using features based on deep neural frame classifiers. However, segmental models can be more challenging to train than standard frame-based approaches. While some segmental models have been successfully trained end to end, there is a lack of understanding of their training under different settings and with different losses.

We investigate a model class based on recent successful approaches, consisting of a linear model that combines segmental features based on an LSTM frame classifier. Similarly to hybrid HMM-neural network models, segmental models of this class can be trained in two stages (frame classifier training followed by linear segmental model weight training), end to end (joint training of both frame classifier and linear weights), or with end-to-end fine-tuning after two-stage training.

We study segmental models trained end to end with hinge loss, log loss, latent hinge loss, and marginal log loss. We consider several losses for the case where training alignments are available as well as where they are not.

We find that in general, marginal log loss provides the most consistent strong performance without requiring ground-truth alignments. We also find that training with dropout is very important in obtaining good performance with end-to-end training. Finally, the best results are typically obtained by a combination of two-stage training and fine-tuning.

***Index Terms*—** Discriminative segmental models, end-to-end training

1. INTRODUCTION

End-to-end training has proved to be successful, for example, in connectionist temporal classification (CTC) [1], encoder-decoders [2], hidden Markov model (HMM) based hybrid systems [3], deep segmental neural networks (DSNN) [4], and segmental recurrent neural networks (SRNN) [5]. All of these models have a feature encoder and an output model for generating label sequences. The feature encoder can be a recurrent or a feedforward neural network, and the output model can be a recurrent neural decoder, such as a long short-term memory network (LSTM), or a probabilistic graphical model, such as an HMM, a conditional random field (CRF), or a semi-Markov CRF. The actual definition of end-to-end training is rarely made explicit in the literature. In this work, we define **end-to-end training** as optimizing the encoder parameters and the output model parameters jointly. The alternative, which we refer to as **two-stage training**, optimizes the feature encoder and output model parameters separately in two stages.

These two families of training approaches differ in terms of annotation requirements, computational and learning efficiency, and the loss functions customarily used for each. Two-stage training typically requires frame-level labels for the first stage, but may therefore require fewer samples to learn from [6]. End-to-end training avoids the cascading errors of pipelines, but results in hard-to-optimize objectives that are sensitive to initialization. It is also possible to perform end-to-end fine-tuning after two-stage training, which has been found useful in past work [7].

In this work, we study training approaches for segmental models. Segmental models have been shown to be successful when trained end to end from scratch [5]. We focus on a particular class of segmental models, with LSTMs as encoders, and linear segmental models as output models. For models trained in two stages, there is often an extra restriction on the representation of the encoded features. For example, they may be log probabilities of triphone states in HMM hybrid systems [8]. Systems trained end to end (encoder-decoders, DSNNs, and SRNNs) are not so constrained. To enable fair comparison, we use model architectures that seamlessly permit both kinds of training without requiring any change to the model parameterization. The only difference is that two-stage training leads to interpretable encoded features, but the functional architectures are identical.[1]

In order to thoroughly compare two-stage and end-to-end training, we consider a variety of loss functions and training settings. When end-to-end systems were first proposed, such as CTC-LSTMs, encoder-decoders, and SRNNs, they were tied to specific loss functions, such as CTC, per-output cross

[1] We note that though our model class is suitable for studying end-to-end systems in various aspects, using better encoders, such as SRNNs, might lead to better absolute performance.

entropy, and marginal log loss. However, these systems can be trained with different loss functions; e.g., encoder-decoder systems can be trained with hinge loss [9]. It is thus important to isolate the effect of training loss functions from models. For our model class, the definition of encoder and output model is completely independent of the definition of loss functions. This allows us to compare training losses while keeping everything else fixed.

Two-stage training typically uses fine-grained labels for training the first stage, such as segmentations. For some datasets, such as TIMIT, we have the luxury to use manually annotated segmentations, but for most of the datasets, we do not. If needed, segmentations are typically inferred by force aligning labels to frames. For our model class, the system can be trained with or without segmentations depending on the choice of loss function.

In the following sections, we explicitly define our model class and loss functions, in particular, hinge loss and log loss for cases where we have ground truth segmentations, and latent hinge loss and marginal log loss when we do not. We perform experiments studying two-stage and end-to-end training in different settings with different losses. On a phoneme recognition task, we show that end-to-end training from scratch with marginal log loss achieves the best result in the setting without ground truth segmentations, while two-stage training followed by end-to-end fine-tuning with log loss achieves the best result in the setting with ground truth segmentations.We also find that dropout is crucial for combating overfitting.

2. DISCRIMINATIVE SEGMENTAL MODELS

Speech recognition, or sequence prediction in general, can be formulated as a search problem. The search space is a set of paths, each of which is composed of segments. Each segment is associated with a weight, and in turn each path is associated with a weight. Prediction becomes finding the highest weighted path in the search space. We formalize this below.

Let $\mathcal{X}$ be the input space, a set of sequences of frames, e.g., MFCCs or Mel filter bank outputs. Let L be the label set, e.g., a phone set for phoneme recognition. A **segment** is a tuple (s, t, y), where s is the start time, t is the end time, and $y \in L$ is the label. Two segments e_1, e_2 are connected if the end time of e_1 is the same as the start time of e_2. A **path** is a sequence of connected segments. A path $p = ((s_1, t_1, y_1), \dots, (s_n, t_n, y_n))$ can also be seen as a label sequence $y = (y_1, \dots, y_n)$ and a segmentation $z = ((s_1, t_1), \dots, (s_n, t_n))$, or simply $p = (y, z)$.

Let E be the set of all possible segments. A **segmental model** is a tuple $(\theta, \Lambda, \phi_\Lambda)$, where $\theta \in \mathbb{R}^d$ is a parameter vector, $\phi_\Lambda : \mathcal{X} \times E \to \mathbb{R}^d$ is a feature function that uses a feature encoder parameterized by the set of parameters Λ. We will give definitions of feature encoders and feature functions in later sections. With a slight abuse of notation, for a path $p = (y, z)$, let $\phi_\Lambda(x, p) = \phi_\Lambda(x, y, z) = \sum_{e \in p} \phi_\Lambda(x, e)$. Prediction can be formulated as

$$\operatorname*{argmax}_{p \in \mathcal{P}} \theta^\top \phi_\Lambda(x, p) = \operatorname*{argmax}_{p \in \mathcal{P}} \sum_{e \in p} \theta^\top \phi_\Lambda(x, e), \tag{1}$$

where $\mathcal{P}$ is the set of all paths. Though the output contains both a label sequence and a segmentation, the segmentation is often disregarded during evaluation.

Learning a segmental model amounts to finding parameters θ and Λ that minimize a specified loss function. Learning can be divided into two cases, one with access to ground truth segmentations, and one without. When we have ground truth segmentations, we receive a dataset $S = \{(x_1, y_1, z_1), \dots, (x_m, y_m, z_m)\}$ and learning aims to solve

$$\operatorname*{argmin}_{\theta, \Lambda} \frac{1}{m} \sum_{i=1}^{m} \ell(\theta, \Lambda; x_i, y_i, z_i). \tag{2}$$

When we do not have ground truth segmentations, we have a dataset $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$, and learning becomes solving

$$\operatorname*{argmin}_{\theta, \Lambda} \frac{1}{m} \sum_{i=1}^{m} \ell(\theta, \Lambda; x_i, y_i). \tag{3}$$

3. LOSS FUNCTIONS

Since segmental models fall under structured prediction, any general loss function for structured prediction is applicable to segmental models. In particular, we investigate hinge loss and log loss for the case with ground truth segmentations, and latent hinge loss and marginal log loss for the case without ground truth segmentations. All loss definitions $\ell(\theta, \Lambda)$ below are given in terms of a single training sample (x, y, z) where $x \in \mathcal{X}$, $(y, z) \in \mathcal{P}$.

Hinge loss is defined as

$$\max_{(y', z') \in \mathcal{P}} \Big[\text{cost}((y, z), (y', z')) - \theta^\top \phi_\Lambda(x, y, z) + \theta^\top \phi_\Lambda(x, y', z') \Big], \tag{4}$$

where cost is a function that measures the distance between two paths. **Log loss** is defined as

$$-\log p(y, z | x) \tag{5}$$

where

$$p(y, z | x) = \frac{1}{Z} \exp(\theta^\top \phi_\Lambda(x, y, z)) \tag{6}$$

and $Z = \sum_{(y', z') \in \mathcal{P}} \exp(\theta^\top \phi_\Lambda(x, y', z'))$. Both hinge loss and log loss require segmentations. Hinge loss has an explicit cost function , while log loss does not. In fact during prediction, hinge loss is always an upper bound of the cost function. Hinge loss is non-smooth due to the $\max$ operation, while log loss is smooth. Both hinge loss and log loss are convex in θ, yet non-convex in Λ if a neural network is used.

Latent hinge loss is defined as

$$\max_{(y',z')\in\mathcal{P}} \Big[\text{cost}((y,\tilde{z}),(y',z')) - \max_{z''} \theta^\top \phi_\Lambda(x,y,z'') + \theta^\top \phi_\Lambda(x,y',z')\Big] \quad (7)$$

where $\tilde{z} = \text{argmax}_{z''\in\mathcal{Z}(y)} \theta^\top \phi_\Lambda(x,y,z'')$ and $\mathcal{Z}(y)$ is the set of possible segmentations of y. **Marginal log loss** is defined as

$$-\log p(y|x) = -\log \sum_{z\in\mathcal{Z}(y)} p(y,z|x). \quad (8)$$

Both latent hinge loss and marginal log loss do not require ground truth segmentations. During prediction, latent hinge loss is also an upper bound of the cost function. Latent hinge loss is non-smooth, while marginal log loss is smooth. Both latent hinge loss and marginal log loss are non-convex in both θ and Λ.

Hinge loss training for segmental models first appeared in [10], log loss in [11], and marginal log loss in [12]. For training first-pass segmental models, [13] is the first to use hinge loss, [11] is the first to use log loss, and [14] is the first to use marginal log loss. For training first-pass segmental models end to end, [4] is the first to use marginal log loss. Other loss functions, such as empirical Bayes risk and structured ramp loss, have been used in [15] for training segmental models.

The above loss functions can be optimized with stochastic gradient descent or its variants. We propagate gradients back through the feature function ϕ, allowing all parameters to be updated jointly.

4. FEATURE FUNCTIONS

Here we define explicitly feature functions we will use in the experiments. These feature functions first appeared in [16].

We assume there is a **feature encoder**, for example, an LSTM, which produces $h_1, \dots, h_T$ given input $x_1, \dots, x_T$. For any $t \in \{1, \dots, T\}$, we project h_t to a $|L|$-dimensional vector and pass the resulting vector through a log-softmax layer and get k_t. In other words, $k_{t,i} = \sum_j W_{ij} h_{t,j} - \log \sum_\ell \exp(\sum_j W_{\ell j} h_{t,j})$, where W is the projection matrix. In this case, the set of parameters Λ includes the projection matrix W and parameters in the LSTM.

The following is a list of features, and the final feature function produces a concatenation of feature vectors produced by the individual feature functions. The average of frames over a segment is defined as

$$\phi_{\text{avg}}(x,(s,t,y)) = \frac{1}{t-s}\sum_{i=s}^{t-1} k_i \otimes \mathbf{1}_y \quad (9)$$

where $\otimes$ is the tensor product, and $\mathbf{1}_y$ is a $|L|$-dimensional one-hot vector for the label y. The frame sample at the r-th percentile is defined as

$$\phi_{\text{at-}r}(x,(s,t,y)) = k_{\lfloor s+rd \rfloor} \otimes \mathbf{1}_y \quad (10)$$

where $d = t-s+1$. The frame at the left boundary is defined as

$$\phi_{\text{left-}r}(x,(s,t,y)) = k_{s-r} \otimes \mathbf{1}_y \quad (11)$$

and similarly, the frame at the right boundary is

$$\phi_{\text{right-}r}(x,(s,t,y)) = k_{t+r} \otimes \mathbf{1}_y. \quad (12)$$

Additionally, we have features that do not depend on the feature encoder. The length score is defined as

$$\phi_{\text{len}}(x,(s,t,y)) = \mathbf{1}_d \otimes \mathbf{1}_y \quad (13)$$

where $d = t-s+1$. Finally, there is a bias for each individual label

$$\phi_{\text{bias}}(x,(s,t,y)) = \mathbf{1}_y. \quad (14)$$

Gradients are propagated through vectors $k_1, \dots, k_T$ to the feature encoder. Parameters of the entire segmental model, including the feature encoder, can be updated jointly.

5. EXPERIMENTS

We conduct phonetic recognition experiments on TIMIT, a 6-hour phonetically transcribed dataset. We follow the conventional setting, training models on the 3696-utterance training set, and evaluate on the 192-utterance core test set. We use the rest of the 400 utterances in the test set as the development set. Following the convention, we collapse 61 phones down to 48 for training, and further collapse them to 39 phones for evaluation.

The feature encoder we use is a 3-layer bidirectional LSTM with 256 cells per layer. The outputs of the third layer are projected from 256 dimensions to 48 and pass through a log-softmax layer so that the final output are log probabilities. Inputs to the encoder are 39-dimensional MFCCs, normalized per dimension by subtracting the mean and dividing by the standard deviation calculated from the training set.

5.1. Two-Stage Training

Since TIMIT is phonetically transcribed, we have access to phone labels for each individual frame. We first train LSTM

Table 1. Frame error rates for different encoder architectures.

	feat	dev	test
CNN	MFCC+fbank	22.27	23.03
LSTM 256x3	MFCC	22.60	
LSTM 256x3 +dropout	MFCC	21.09	21.36

frame classifiers with cross entropy loss at each frame. This LSTM will serve as our feature encoder later on, and training such LSTM corresponds to the first stage of two-stage learning. LSTM parameters are initialized uniformly in the range $[-0.1, 0.1]$. Biases for forget gates are initialized to one [17], while other biases are initialized to zero. Dropout for LSTMs [18] is applied at all input layers and the last output layer with a dropout rate of 50%. We compare AdaGrad with step sizes in $\{0.01, 0.02, 0.04\}$ and RMSProp with step size 0.001 and decay 0.9. Mini-batch size is always one utterance. Both optimizers are run for 50 epochs. We choose the best performing model according to the frame error rate on the development set, also known as early stopping. No gradient clipping is used during training. For comparison, following [13] we train a convolutional neural network (CNN) consisting of 5 layer convolution followed by 3 fully-connected layers. Frame classification results are shown in Table 1. We observe that the best performing LSTM achieves a comparable frame error rate as the CNN. With dropout, the frame error rate is further lowered.

After obtaining LSTM frame classifiers, we proceed to the second stage, training segmental models with features based on LSTM log probabilities. Segmental models are trained with the four loss functions for 50 epochs with early stopping. Overlap cost [15] is used in hinge loss and latent hinge loss. A maximum duration of 30 frames is imposed. We use feature functions described in Section 4. No regularizer is used except early stopping. We compare AdaGrad with step sizes in $\{0.1, 0.2, 0.4\}$ and RMSProp with step size 0.001 and decay 0.9. Phonetic recognition results for hinge loss are shown in Table 2. We observe that LSTMs perform better in frame classification, but give little improvement over CNNs in phonetic recognition. Recognition results for the rest of the losses are in Table 3. Note that even though latent hinge loss and marginal log loss do not require segmentations during training, we do use ground truth segmentations for training the frame classifier. It is not a common setting, and is done purely for comparison purposes. We observe that, except latent hinge, other losses perform equally well, with log loss having a slight edge over the others.

5.2. End-to-End Training with Warm Start

After two-stage training, we fine tune the encoder and segmental model jointly to further lower the training loss. The four losses are compared with and without dropout. When dropout is used, dropout rate 50% is chosen to match the rate during frame classifier training. The input layers and the output layer are scaled by 0.5 when no dropout is used. First, we initialize the models with the one trained with hinge loss above. We run AdaGrad with step size 0.001 for 10 epochs with early stopping. Results are shown in Table 4. We observe healthy reductions in phone error rates by fine tuning the two-stage system across all loss functions. We also find that fine-tuning without dropout tends to be better than with dropout. Though fine-tuning with hinge loss leads to the most error reduction, we note that the two-stage system is trained with hinge loss. At least we are centain that the two-stage system trained with hinge loss is a descent initialization for other losses.

Minimizing other losses from a model trained with hinge loss is less than ideal. We repeat the above experiments by warm-starting from a model trained with the loss function that we are going to minimize. Results are shown in Table 5. We observe significant gains for log loss and marginal log loss if initialized with the matching loss function. Similarly, the gains with dropout in these cases are smaller than without dropout.

Table 2. Phone error rates for segmental models trained with hinge loss using log probabilities generated from various encoders in Table 1.

	feat	dev	test
CNN	MFCC+fbank	21.4	22.5
LSTM 256x3	MFCC	23.1	
LSTM 256x3 +dropout	MFCC	21.4	22.1

Table 3. Phone error rates for segmental models trained in two stages with different losses.

	dev	test
hinge	21.4	22.1
log loss	21.2	21.9
latent hinge	23.5	24.6
marginal log loss	21.6	22.5

5.3. End-to-End Training from Scratch

Next, we train the same architecture end to end from scratch. We make sure that all the models are initialized identically to the two-stage systems. The four losses are used for training with dropout rates in $\{0, 0.1, 0.2, 0.5\}$. Ground truth segmentations are used when training with hinge loss and log loss, and are disregarded when training with latent hinge loss and marginal log loss. The optimizers we use here are SGD with

Table 4. Phone error rates for segmental models trained end to end initialized from the two-stage system trained with hinge loss.

	dropout	dev	test
hinge	0	19.4	20.7
	0.5	20.8	
log loss	0	20.2	21.7
	0.5	21.1	
latent hinge	0	19.3	21.0
	0.5	20.8	
marginal log loss	0	20.7	22.2
	0.5	20.9	

Table 5. Phone error rates for segmental models trained end to end initialized from two-stage systems trained with corresponding loss functions.

	dropout	dev	test
hinge	0	19.4	20.7
	0.5	20.8	
log loss	0	18.8	19.7
	0.5	20.3	
latent hinge	0	20.0	21.2
	0.5	22.1	
marginal log loss	0	19.2	20.8
	0.5	21.0	

Table 6. Phone error rates for segmental models trained end to end with dropout.

	dropout	dev	test
hinge	0	23.1	
	0.1	22.4	
	0.2	22.3	23.7
	0.5	28.9	
log loss	0	24.8	
	0.1	22.4	
	0.2	20.8	22.2
	0.5	22.3	
latent hinge		failed	
marginal log loss	0	25.3	
	0.1	22.1	
	0.2	20.0	22.0
	0.5	22.0	

Table 7. Average cross entropy over frames before and after end-to-end fine-tuning.

LSTM	train CE	dev CE	dev PER
256x3 (best train)	0.0569	2.2395	
256x3 (best dev)	0.4179	0.9442	
256x3 +dropout	0.4595	0.7466	21.4
256x3 +dropout +e2e	0.3864	0.6928	19.4

step sizes in $\{0.1, 0.5\}$, momentum 0.9, and gradient clipping at norm 5, AdaGrad with step sizes in $\{0.01, 0.02, 0.04\}$ and no clipping, and RMSProp with step size 0.001, decay 0.9, and no clipping. We run earch optimizer for 50 epochs with early stopping. Results are shown in Table 6. First, all optimizers above fail to minimize latent hinge loss. All of them get stuck in local optima, and fail to produce reasonable forced alignments. Even though all loss functions in end-to-end training are nonconvex, latent hinge loss is more sensitive to initialization than other losses. The second observation is that adding dropout improves performance. However, using the same dropout rate as the two-stage system results in worse performance. Finally, though behind the best fine-tuned model, marginal log loss with dropout 0.2 slightly edges over other losses.

6. DISCUSSION

We have seen that end-to-end training initialized with a two-stage system leads to the best results. Since in end-to-end training, the meaning of the intermediate representations is not enforced anymore, it is unclear how the intermediate representations deviate from the learned ones. To answer this, we measure per-frame cross entropy for the LSTM frame classifier after end-to-end training. Results are shown in Table 7. First, the per-frame cross entropy for the best performing LSTM on the training set can be as low as 0.06, which shows that a 3-layer bidirectional LSTM with 256 cells per layer is able to essentially memorize the entire TIMIT dataset. However, it is severely overfitting. Early stopping and dropout help balance cross entropies on the training set and development set. In addition, the cross entropies on both sets drop after end-to-end training. It shows that the meaning of the intermediate representations is still maintained by the LSTMs after end-to-end training.

Next, since the system trained with marginal log loss does not use the ground truth segmentations, and since the evaluation measure (phone error rate) does not consider segmentations, we do not know if the system is able to discover reasonble phone boundaries without supervision. We approach this question by aligning the label sequences to the acoustics, and compare the resulting segmentations against the manually annotated segmentations. The alignment quality for different tolerance values is shown in Table 8. Though the results are behind models trained specifically to align [19], the segmen-

Table 8. Forced alignment quality on the test set as a percentage of correctly positioned phone boundaries within a predefined tolerance, measured with the best-performing segmental model trained with marginal log loss.

$t \leq 10$ms	$t \leq 20$ms	$t \leq 30$ms	$t \leq 40$ms
64.5	86.8	94.7	96.7

tal model trained with marginal log loss is not supervised with any ground truth segmentations. Limiting the maximum duration to 30 frames also affects the alignment performance.

Since most speech datasets do not have manually annotated segmentations, it is desirable to train without manual alignments. As we now know, the alignments produced by our system trained with marginal log loss are of good quality. Therefore, we can use the forced alignments to train a two-stage system followed by end-to-end fine-tuning. We follow the exact same procedure as in the previous two-stage experiments by training an LSTM frame classifier with the forced alignments, followed by training a segmental model with hinge loss. The frame error rate on the development set of the LSTM classifier is 21.68% against the forced alignments and 28.91% against the ground-truth segmentations. Though the frame error rate is significantly worse than when training with ground-truth segmentations, this two-stage system achieves a phone error rate of 21.0% on the development set. We then fine-tune the entire system with hinge loss. The final system achieves 18.6% phone error rate on the development set, and 20.1% on the test set, a significant improvement from the model trained end-to-end with marginal log loss, while not relying on ground truth segmentations.

In terms of efficiency in training, all four losses require forward-backward-like algorithms for computing gradients. Hinge loss requires one pass on the entire search space, log loss requires two passes on the entire search space, latent hinge requires one pass on the entire search space and one on the segmentation space, and marginal log loss requires two passes on the entire search space and two passes on the segmentation space. The average number of hours per epoch spent on computing gradients, excluding LSTM computations, is shown in Table 9. To put them into context, feeding forward and backpropagation for LSTMs takes 1.65 hours per epoch. The timing is done on a single 3.4GHz four-core CPU. The number of hours is consistent with the number of passes required to compute gradients. Note that the time spent on LSTMs can be halved without incurring a performance loss by applying frame skipping [20, 21] as shown for segmental models in [22].

Table 9. Average number of hours per epoch spent on computing gradients excluding LSTM computations.

hinge	log loss	latent hinge	marginal log loss
0.52	1.08	0.73	2.10

7. CONCLUSION

In this work, we study end-to-end training in the context of segmental models. The model class of choice includes a 3-layer bidirectional LSTM as feature encoder and a segmental model using the features to produce label sequences. This model class is suitable for studying end-to-end training, due to its flexibility to be trained either in a two stage manner, or end to end. The hypothesis is that training such systems in two stages is easier than end-to-end training from scratch. On the other hand, end-to-end training can better optimize the loss function, but it might be sensitive to initialization.

Our model definition is separated from the definition of loss functions, giving us the flexibility to choose loss functions based on the training settings. We consider two common training settings, one with ground truth segmentations and one without. Hinge loss and log loss require segmentations by definition, while latent hinge loss and marginal log loss do not.

We show that in the case where we have ground truth segmentations, two-stage training followed by end-to-end training is significantly better than two-stage training alone (improving upon it by 10% relative) and end-to-end training from scratch. In addition, we find that end-to-end training with marginal log loss from scratch achieves competitive results. As a byproduct, the system is able to generate high-quality forced alignments. To remove the dependency on ground truth segmentations, we train another model on the forced alignments in two stages followed by end-to-end fine-tuning, improving upon end-to-end training from scratch by 8.6% relative. The final product is a strong system trained end to end without requiring ground truth segmentations.

8. ACKNOWLEDGEMENT

This research was supported by a Google faculty research award and NSF grant IIS-1433485. The opinions expressed in this work are those of the authors and do not necessarily reflect the views of the funding agency. The GPUs used for this research were donated by NVIDIA.

9. REFERENCES

[1] Alex Graves, Abdel-Rahman Mohamed, and Geoffrey Hinton, “Speech recognition with deep recurrent neural

networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[2] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112.

[3] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahrmani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," *Annual Conference of the International Speech Communication Association*, 2016.

[4] Ossama Abdel-Hamid, Li Deng, Dong Yu, and Hui Jiang, "Deep segmental neural networks for speech recognition.," in *Annual Conference of International Speech Communication Association*, 2013, pp. 1849–1853.

[5] Liang Lu, Lingpeng Kong, Chris Dyer, Noah A. Smith, and Steve Renals, "Segmental recurrent neural networks for end-to-end speech recognition," in *Annual Conference of the International Speech Communication Association*, 2016.

[6] Shai Shalev-Shwartz and Amnon Shashua, "On the sample complexity of end-to-end training vs. semantic abstraction training," *CoRR*, vol. abs/1604.06915, 2016.

[7] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey, "Sequence-discriminative training of deep neural networks.," in *Annual Conference of the International Speech Communication Association*, 2013.

[8] Alex Graves, Navdeep Jaitly, and Abdel-Rahman Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013, pp. 273–278.

[9] Sam Wiseman and Alexander M. Rush, "Sequence-to-sequence learning as beam-search optimization," *CoRR*, vol. abs/1606.02960, 2016.

[10] Shi-Xiong Zhang and Mark Gales, "Structured SVMs for automatic speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 3, pp. 544–555, 2013.

[11] Sunita Sarawagi and William W Cohen, "Semi-Markov conditional random fields for information extraction," in *Advances in Neural Information Processing Systems*, 2004, pp. 1185–1192.

[12] Geoffrey Zweig and Patrick Nguyen, "A segmental CRF approach to large vocabulary continuous speech recognition," in *IEEE Workshop on Automatic Speech Recognition & Understanding*, 2009, pp. 152–157.

[13] Hao Tang, Weiran Wang, Kevin Gimpel, and Karen Livescu, "Discriminative segmental cascades for feature-rich phone recognition," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2015.

[14] Geoffrey Zweig, "Classification and recognition with direct segment models," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012, pp. 4161–4164.

[15] Hao Tang, Kevin Gimpel, and Karen Livescu, "A comparison of training approaches for discriminative segmental models," in *Annual Conference of the International Speech Communication Association*, 2014.

[16] Yanzhang He and Eric Fosler-Lussier, "Efficient segmental conditional random fields for phone recognition," in *Annual Conference of the International Speech Communication Association*, 2012, pp. 1898–1901.

[17] Rafal Jozefowicz, Wojciceh Zaremba, and Ilya Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning*, 2015.

[18] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals, "Recurrent neural network regularization," *CoRR*, vol. abs/1409:2320, 2014.

[19] Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer, and Dan Chazan, "A large margin algorithm for speech-to-phoneme and music-to-score alignment," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 8, pp. 2373–2382, 2007.

[20] Yajie Miao, Jinyu Li, Yongqian Wang, Shixiong Zhang, and Yifan Gong, "Simplifying long short-term memory acoustic models for fast training and decoding," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015.

[21] Vincent Vanhoucke, Matthieu Devin, and Georg Heigold, "Multiframe deep neural networks for acoustic modeling," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 7582–7585.

[22] Hao Tang, Weiran Wang, Kevin Gimpel, and Karen Livescu, "Efficient segmental cascades for speech recognition," in *Annual Conference of the International Speech Communication Association*, 2016.