Distributed Semi-Supervised Sequence Modeling

Karan Malhotra [1] **Shubham Bansal** [2] **Chaikesh Chouragade** [3] **Advisor: Dr. Partha Pratim Talukdar** [4]

Abstract

Cross View Training(CVT) is a semi supervised learning technique that has shown improvements for several NLP tasks. For labeled examples, standard supervised learning is performed and for unlabeled examples, CVT teaches the auxiliary prediction modules that see only restricted views of the input to match the prediction of the primary module which has the full input view. Since our approach requires distributing CVT, so to better understand it we first implement CVT for text chunking task in a distributed setting. Recent LSTM-CTC based end-to-end ASR(Automatic Speech Recognition) have been shown to work extremely well when there is an abundance of supervised training data, matching and exceeding the performance of hybrid DNN systems. But, labeling speech utterances is very expensive and time consuming task. So, we propose a method to apply CVT with LSTM-CTC based ASR model to leverage unlabeled raw speech data and train a robust ASR with limited labeled speech data. As this approach requires large amount of unlabeled data, it necessitates training of model in distributed settings. For this, we explore different settings of distributed training for the above mentioned tasks.

1. Introduction

Models such as deep neural networks(DNNs) and deep recurrent networks works best when trained on large amounts of labeled data. But getting labels is labour intensive and expensive, thus motivating the need of effective semi-supervised learning techniques that leverage unlabeled examples. Methods which are widely successful such as pretraining has a key disadvantage that the first representation learning phase does not take advantage of labeled data - the model attempts to learn generally effective representations rather than ones that are targeted towards a particular tasks.

On the other hand the conventional semi-supervised algorithms such as self-training doesn't suffer from such shortcomings, so in this project we implement a recently proposed self training method called Cross-View Training (CVT)(1). CVT is inspired by multiview learning(3) and where model is trained in such a way that it produces consistent predictions across different views of input. The algorithm uses two prediction modules namely primary prediction module and auxiliary prediction modules. Primary prediction module has a full view of the input while the auxiliary have a restricted view of the input. For example, one auxiliary prediction module for sequence tagging is attached to only the forward LSTM in the models first BiLSTM layer, so it makes predictions without seeing any tokens to the right of the current one. In short, CVT combines the idea of representation learning on unlabeled data with classic self-training. Figure-1 is an example of CVT applied to named entity recognition task:

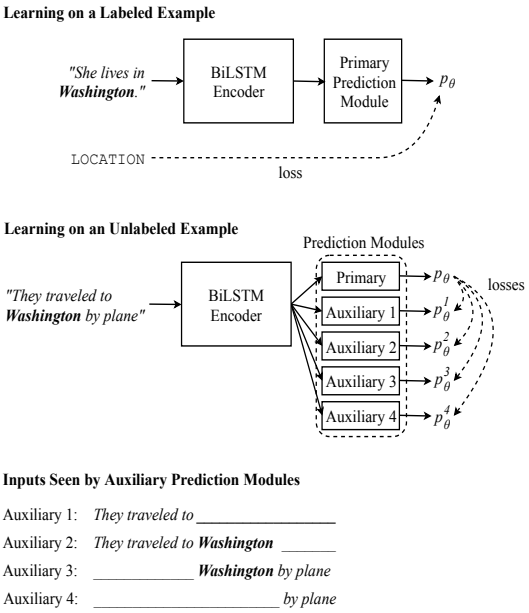

Figure 1. An overview of Cross-View Training

Figure-1 gives an idea, how CVT can be applied on the tasks with neural architectures. The model is trained with standard supervised learning on labeled examples. On unlabeled examples, auxiliary prediction modules with different views of the input are trained to agree with the primary prediction module.

CVT can be applied on various NLP sequence modelling tasks like sequence tagging , machine translation , text chunking etc, but for this project only text chunking task has been implemented in both local and distributed mode. Large amount of unlabelled data to be used, justify the choice of distributed implementation.

The task of Automatic Speech Recognition(ASR) is very similar to that of machine translation task and as more End-to-End model architectures are being proposed in recent years for ASRs, the need for data to train them also increases. But labeling speech utterances is labour intensive and expensive. We propose a method to apply CVT in case of LSTM-CTC based ASR model architecture(2) to leverage unlabeled raw speech data and train a robust ASR with limited amount of labeled speech data. This configuration is also being implemented in both local and distributed mode.

The rest of the report is organized as follows. In Section-2 we describe some of the related work, in Section-3 our methodologies , Section-4 consists of details of dataset used, Section-5 describes experiments, Section-6 describe results, Section-7 and Section-8 discuss about Observations and Conclusion respectively.

2. Related Work

Yarowsky et al.(5) has shown effectiveness of self-training for several NLP tasks. Although, this process has shown value for some tasks, it is somewhat tautological in the sense that for unlabeled data, the model already produces the predictions it is being trained on. Recent research by Sajjadi et al.(4) addresses this by adding noise to the model input, training the model so it is robust to input perturbation. Further, Kevin et al.(1), taking motivation for multiview learning from Xu et al. (3) proposed CVT to train the model to produce consistent predictions across different views of the unlabeled input using auxiliary modules. CVT(1) showed improvement in performance for several NLP taks and in our project, we have propose CVT for End-to-End ASR under distributed settings.

Long Short Term Memory(LSTM) based ASR when trained with Connectionist Temporal Classification (CTC) first proposed by Alex Graves et al.(2) have recently been shown to work extremely well when there is an abundance of supervised training data, matching and exceeding the performance of hybrid DNN systems.

However, these system lags comparable hybrid DNN system when trained on smaller supervised training sets. LSTM-CTC systems are much simpler to train, they can be trained as an End-to-End speech recognition system as opposed to the iterative multi-process approach to hybrid DNN ASR system training. Hence, we propose to improve the training with small supervised training sets by leveraging unlabeled data with the help of CVT(1).

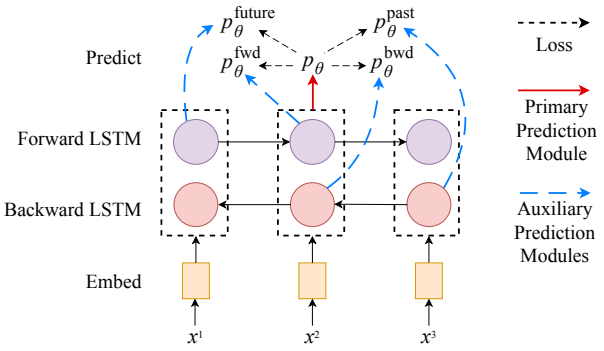

Figure 2. Auxiliary prediction module for text chunking.

3. Methods

3.1. Cross View Training(CVT)

For labeled examples, CVT minimizes standard cross-entropy loss:

$$\mathcal{L}_{\text{sup}}(\theta) = \frac{1}{|\mathcal{D}_l|} \sum_{x_i, y_i \in \mathcal{D}_l} CE(y_i, p_\theta(y|x_i))$$

CVT trains the auxiliary prediction modules to match the primary prediction module on the unlabeled data by minimizing

$$\mathcal{L}_{\text{CVT}}(\theta) = \frac{1}{|\mathcal{D}_{ul}|} \sum_{x_i \in \mathcal{D}_{ul}} \sum_{j=1}^{k} D(p_\theta(y|x_i), p_\theta^j(y|x_i))$$

It combines the supervised and CVT losses into the total loss, $\mathcal{L} = \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{CVT}}$, and minimize it with stochastic gradient descent. In particular, we alternate minimizing $\mathcal{L}_{\text{sup}}$ over a minibatch of labeled examples and minimizing $\mathcal{L}_{\text{CVT}}$ over a minibatch of unlabeled examples.

3.2. CVT for Text Chunking

Text Chunking task divides the text into phrases such that syntactically related words become member of same phrase.

It uses a two-layer CNN-BiLSTM sentence encoder. It takes as input a sequence of words $x_i = [x_i^1, x_i^2, ..., x_i^T]$. First, each word is represented as the sum of an embedding vector and the output of a character-level Convolutional Neural Network, resulting in a sequence of vectors $v = [v^1, v^2, ..., v^T]$. The encoder applies a two-layer bidirectional LSTM to these representations. The output of the Bi-LSTM is the concatenation of forward and backward vectors: $h_1 = [\overrightarrow{h}_1^1 \oplus \overleftarrow{h}_1^1, ..., \overrightarrow{h}_1^T \oplus \overleftarrow{h}_1^T]$. The second Bi-LSTM layer works the same, producing outputs h_2, except it takes h_1 as input instead of v.

In text chunking, each token x_i^t has a corresponding label y_i^t. The primary prediction module for sequence tagging produces a probability distribution over classes for the t^{th} label using a one-hidden-layer neural network applied to the corresponding encoder outputs:

$$p(y^t|x_i) = \text{NN}(h_1^t \oplus h_2^t)$$
$$= \text{softmax}(U \cdot \text{ReLU}(W(h_1^t \oplus h_2^t)) + b)$$

The auxiliary prediction modules (Figure-2) take $\overrightarrow{h}_1(x_i)$ and $\overleftarrow{h}_1(x_i)$, the outputs of the forward and backward LSTMs in the first Bi-LSTM layer, as inputs. Following four auxiliary prediction modules are addded to the model:

$$p_\theta^{\text{fwd}}(y^t|x_i) = \text{NN}^{\text{fwd}}(\overrightarrow{h}_1^t(x_i))$$
$$p_\theta^{\text{bwd}}(y^t|x_i) = \text{NN}^{\text{bwd}}(\overleftarrow{h}_1^t(x_i))$$
$$p_\theta^{\text{future}}(y^t|x_i) = \text{NN}^{\text{future}}(\overrightarrow{h}_1^{t-1}(x_i))$$
$$p_\theta^{\text{past}}(y^t|x_i) = \text{NN}^{\text{past}}(\overleftarrow{h}_1^{t+1}(x_i))$$

3.3. LSTM-CTC based ASR

The LSTM-CTC model is for labelling sequence data with recurrent nets that removes the need for pre-segmented training data and post-processed outputs, and models all aspects of the sequence within a single network architecture. The basic idea is to interpret the network outputs as a probability distribution over all possible label sequences, conditioned on a given input sequence. Given this distribution, an objective function can be derived that directly maximizes the probabilities of the correct labellings. Since the objective function is differentiable, the network can then be trained with standard backpropagation through time.
The objective function used for this task is called connectionist temporal classification (CTC). Figure-3 shows the model architecture for LSTM-CTC:

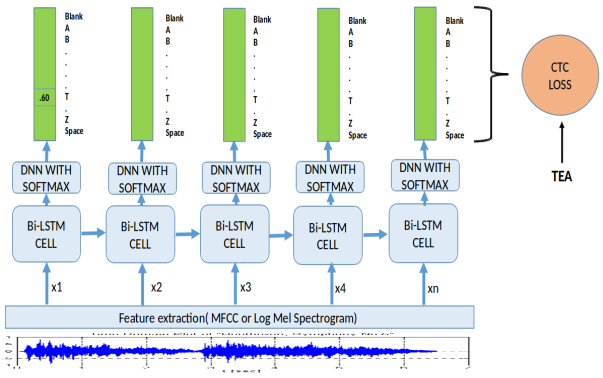

Figure 3. LSTM-CTC Architecture

In the above figure the input $x_1, x_2.....x_n$ is a sequence of MFCCs corresponding to a speech recording which is converted to a high level feature representation using a BiLSTM layer.
On top of this a neural network outputs **Z** which encodes distribution over symbols $\{A, B, ...Z, SPACE, BLANK\}$ i.e
Encode:$\mathbf{X} \Rightarrow \mathbf{Z}$
Define a mapping from distribution to text : $f(\mathbf{Z}) = \mathbf{C}$. For example,

$$f(HH\_EEE\_\_\_L\_\_LLL\_\_\_OOOO) = HELLO$$
$$f(HHH\_\_E\_\_\_LLLL\_LLL\_\_\_OO) = HELLO$$

In training : summation of probability of all **Z** that gives us **C** for each of given pair $(\mathbf{X}, \mathbf{C})$ in training data.

Conditional independence assumption: Here also, we assume frame independence assumption i.e

$$P(\mathbf{Z} = HH\_EEE\_\_\_L\_\_LLL\_\_\_OOOO|\mathbf{X})$$

can be written as product of individual probabilities

$$P(H|\mathbf{X})P(H|\mathbf{X})P(BLANK|\mathbf{X})...P(O|\mathbf{X})$$

Maximizing the likelihood:Update network parameters θ to maximize likelihood of training data as below:

$$\theta^* = argmax_\theta \sum_i Log\left(P\left(\boldsymbol{C}_i|\boldsymbol{X}_i\right)\right)$$

, which can be further written as

Inner sum can be computed efficiently using dynamic programming using forward-backward algorithm.
The Below figure is an illustration of the forward backward algorithm applied to the labelling CAT. Black circles represent labels, and white circles represent blanks. Arrows signify allowed transitions. Forward variables are updated in the direction of the arrows, and backward variables are updated against them.

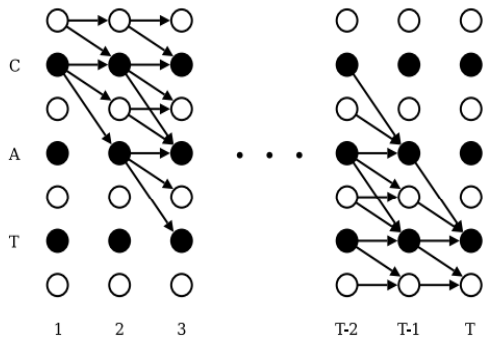

Figure 4. Forward Backward Algorithm

In testing : We use Use Beamsearch decoding as shown in Figure 5.

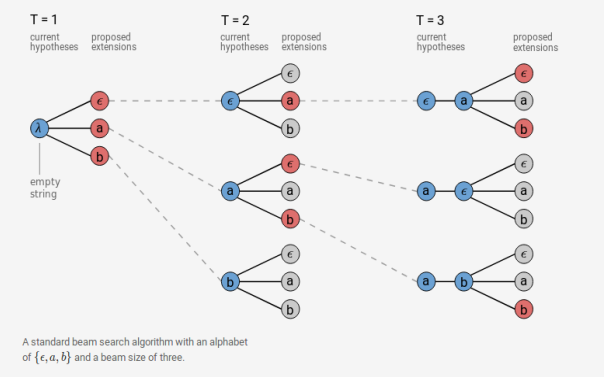

Figure 5. Beam Search Decoding

3.4. CVT for ASR(Proposed Novelty)

[] recently proposed Cross View Training for NLP tasks like text chunking, NER and Machine Translation and shown that this type of semi supervised methods allows models to effectively leverage their own predictions on unlabeled data. In this project we propose application of CVT for LSTM-CTC based ASR architecture so that it can leverage from large amount of raw speech.
In our proposed approach, we have one primary and two auxiliary modules. For labeled data set (**X,C**) the primary module will try to minimize the following standard CTC loss :

$$L_{sup}(\mathbf{X},\mathbf{C}) = -Log\left(\sum_{f(\mathbf{Z})=\mathbf{C}_i} P(\mathbf{Z}|\mathbf{X}_i)\right)$$

For unlabeled data,since, we have no target label sequence, we produce hard targets ($\hat{\mathbf{C}}$) for the auxiliary modules by running the primary module with beam search decoding. First, we produce two restricted views of input **X**.

$$\mathbf{H}^f = Softmax(Linear_f(BLSTM_f(\mathbf{X})))$$
$$\mathbf{H}^b = Softmax(Linear_b(BLSTM_b(\mathbf{X})))$$

Hence, we could define unsupervised CVT loss as below:

$$L_{CVT}(\mathbf{X},\hat{\mathbf{C}}) = CTCloss(\mathbf{H}_b,\hat{\mathbf{C}}) + CTCloss(\mathbf{H}_f,\hat{\mathbf{C}})$$

Finally, during optimization, we alternate minimizing L_{sup} over a minibatch of labeled examples and minimizing L_{CVT} over a minibatch of unlabeled examples.

4. Dataset

- **TEXT CHUNKING**:

  Supervised: CoNLL-2000 , this data consists of the same partitions of the Wall Street Journal corpus (WSJ).
  Training data consists of 211727 tokens and test data consists of 47377 tokens.

  Unsupervised: 1 Billion Word Language Model corpus.

- **LSTM CTC ASR**:

  LibriSpeech Corpus is a Large-scale (1000 hours) corpus of labeled read English speech.
  But for this project, 150hrs(13G) out of 1000hrs were utilized. We have considered 50 hrs as labeled and 100 hrs as unlabeled .

5. Experiments

We have used **Tensorflow distributed framework** which enables us to create a cluster of servers and then distribute computation graph among them. Also, it provides with very abstract way to train in both synchronous and asynchronous manner without worrying about communication among servers and distribution of data. Between graph replication approach has been used where every worker has the copy of computation graph. For all our experiments, we have kept a single parameter servers and number of worker nodes are varied. Cluster details are as follows:
CLUSTER: LEAP LAB, EE, IISc
Parameter server: [10.1.1.254:2225] (node 0)
Worker 0: [10.1.1.253:2223] (node 1)
Worker 1: [10.1.1.252:2224] (node 2)
Worker 2: [10.1.1.251:2222] (node 3)

5.1. Text Chunking

5.1.1. Model Details:

For encoder, each word is represented as the sum of an embedding vector and the output of a character-level Convolutional Neural Network. Hyperparameters used for CNN and Bi-LSTM layers in encoder has been given in table 1.

Table 1. Hyperparameter for text chunking

Hyperparameter	value
Character embedding size	50
CNN filter widths	2,3,4
No of filters	100 (for each width)
1st LSTM hidden units	1024
2nd LSTM hidden units	512

5.1.2. EXPERIMENTAL SETTINGS:

We have modified the baseline CVT approach to experiment with it under local and distributed settings. For all our distributed experiments in text chunking, we used 2 worker nodes and 1 parameter server. We implemented Bulk Synchronous Parallel(BSP), Stale Synchronous Parallel(SSP) and Asynchronous Parallel(ASP). For SSP, we have varied staleness between 8 and 16. Figure 5 and Figure 6 shows convergence of Supervised and Unsupervised loss respectively with time.

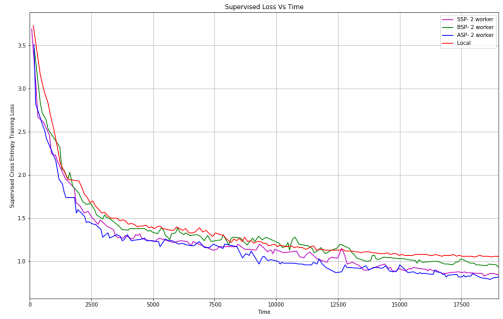

Figure 6. Supervised Loss for Text Chunking

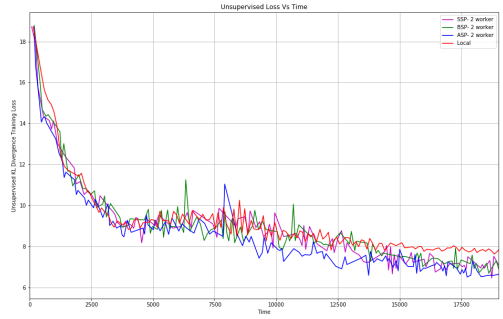

Figure 7. Unsupervised Loss for Text Chunking

5.2. CVT for LSTM-CTC ASR

5.2.1. MODEL DETAILS:

Raw audio files are featurized using Mel-frequency cepstral coefficients(MFCC). For Encoder, we have used 2 layer BiLSTM network. The encoder parameters are being shared among primary and auxiliary modules. Further, we have a DNN with softmax that gives us probability distribution over symbols $\{A, B, ...Z, SPACE, BLANK\}$.
The distribution for entire sequence is being used to calculate L_{sup} for labeled data and L_{CVT} for unlabeled data as described in Section 3.4 and alternatively minimized. Details about model and optimizer hyperparameters have been provided in Table 2.

Table 2. Hyperparameter for CVT in LSTM-CTC ASR

Hyperparameter	value
Dimension of MFCC	39
No of LSTM layers	2
No of hidden units	200
No of Auxiliary modules	2
Learning rate for L_{sup}	0.001
Learning rate for L_{CVT}	0.00002
Optimizer used	Momentum

5.2.2. EXPERIMENTAL SETTINGS:

Firstly, the LSTM-CTC ASR was built from scratch without using CVT algorithm and then it was modified to work with CVT , both in local mode.
Then for distributed settings, we have implemented ASP,BSP and SSP mode. Single Parameter server has been used for all our experiments. With 2 workers, staleness of 5 was observed to perform better. Figure 8 and Figure 9 shows the convergence of supervised and unsupervised training loss respectively for all the three distributed modes.

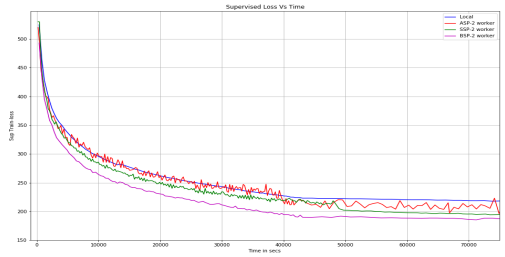

Figure 8. Supervised Training CTC-Loss for LSTM-CTC in case of 2-Workers

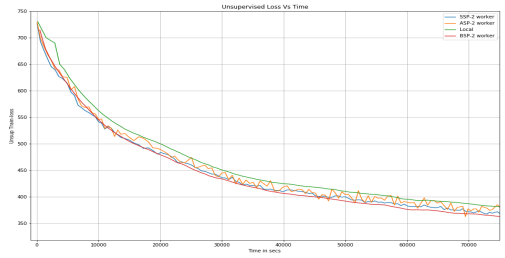

Figure 9. Unsupervised Training for LSTM-CTC in case of 2-Workers

With three workers,the ASP when implemented gave NANs as CTC-loss output, as the CTC-loss in general is very sensitive and asynchronous mode causes highly noisy updates . So, in this configuration only BSP and SSP were explored. Figure 10 shows the convergence of supervised training loss for all the three distributed modes.

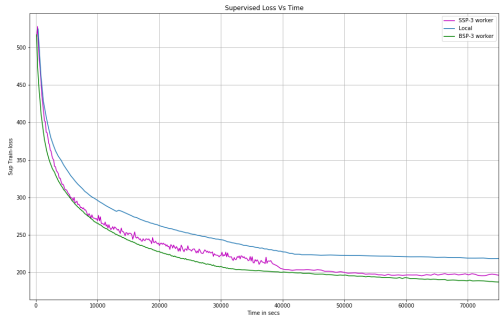

Figure 10. Supervised Training CTC-Loss for LSTM-CTC in case of 3-Workers

The Figure 11 shows the comparison between the convergence of supervised training loss with number of workers varied for all the distribution modes.

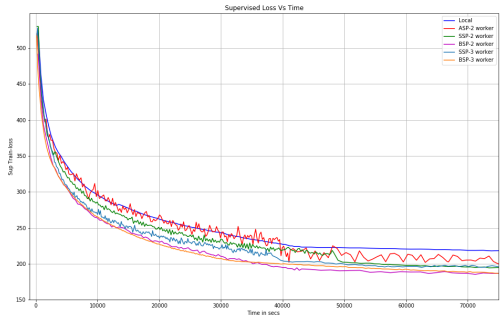

Figure 11. Comparison of 2 Worker and 3 worker case

6. Results

6.1. Text Chunking

Table 3 compares the test F1-score for text chunking task with and without CVT.

6.2. CVT for LSTM-CTC ASR

Table 4 shows the Convergence time for Supervised loss and Test CER obtained on standard Librispeech test-clean dataset when model is trained with and without CVT in local mode. Table 5 compares the Convergence time for Supervised loss and Test CER obtained for different distribution modes.

Table 3. F1-score with different training modes for text chunking

Mode of training	Test F1-Score
Supervised training	94.7
Semi-Supervised training(with CVT)	95.5

Table 4. Test CER and convergence time for LSTM-CTC ASR with different training modes

Mode of training	Convergence time	Test CER
Supervised training	31hrs	28.2
Semi-Supervised training(with CVT)	65hrs	27.1

Table 5. Test CER and convergence time for LSM-CTC ASR for different distribution modes

Distributed settings	Convergence time	Test CER
BSP 3-Worker, Supervised	8hrs	30.1
ASP 2-worker, SSL(with CVT)	30hrs	30.2
BSP 2-worker, SSL(with CVT)	22 hrs	28.5
SSP 2-worker, SSL(with CVT)	25hrs	29.1
BSP 3-worker, SSL(with CVT)	20hrs	28.6
SSP 3-worker, SSL(with CVT)	23hrs	29.3

7. Observations

- Text Chunking, as originally shown by (1), with CVT, gave marginal improvement in F1-score with CVT for us. Also, **under distributed settings, as shown in Figure 6 and 7, we achieved best convergence with ASP mode followed by SSP and then BSP**. This might be due to more frequent updates with ASP mode and **updates being less noisy or loss function being less sensitive to noisy updates**.
- **Our proposed CVT approach for LSTM-CTC ASR model gave 3.9 percent relative improvement in Test CER over standard Supervised approach**.(local mode)
- In Distributed settings , it was observed that **CTC loss was very sensitive to noisy updates and with three workers doing asynchronous gradient updates caused NAN values for CTC-Loss**. Moreover, for two worker case, we **achieved best performance with BSP followed by SSP and then ASP**.
- Overall BSP with three workers gave the best test CER in case of distributed mode.

- It was also observed that **distributing ASR caused around 1 to 1.5% degradation in performance compared to local settings**. Though, we did not find any significant degradation in performance of text chunking in distributed mode.

8. Conclusion and Future Work

From our experiments for the task of text chunking with CVT, we observed that the ASP performs best among all distributed setting. As the loss function in case of text chunking task is less sensitive, ASP results in faster convergence inspite of noisy gradient updates. However, in case of LSTM-CTC based ASR with CVT the loss function is quite sensitive, so the performance of ASP degrades due to too many noisy updates. It is evident from our experiment that the BSP achives the best convergence among all distributed setting in this case but distributing ASR gives a marginal degradation performance when compared to local mode.

Also, In our experiments, due to lack of computational resources we have used only 100 hours of unlabeled data and 50 hrs of labeled data. By using more unlabeled data we hope to improve the performance further. Incorporating more recently proposed Encoder Decoder structure and their hybrid models with LSTM-CTC , with CVT could help further lowering the test CER. Moreover, inclusion of a strong language model could also give a significant improvement.The model parallel approach coul be explored in these tasks and their performance can be compared with implemented data parallel approaches.

References

[1] Kevin Clark, Minh-Thang Luong, Christopher D. Manning and Quoc V. Le. “Semi-Supervised Sequence Modeling with Cross-View Training”.In EMNLP 2018

[2] “Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks.”In ICML 2014

[3] Chang Xu, Dacheng Tao, and Chao Xu. 2013.“A survey on multi-view learning.”

[4] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. 2016.“Regularization with stochastic transformations and perturbations for deep semisupervised learning”. In NIPS

[5] David Yarowsky. 1995. “Unsupervised word sense disambiguation rivaling supervised methods”. In ACL.