# Semi-Supervised Sequence Modeling With Cross-View Training

## 집현전 2기 중급반

6조 이다영, 양재우, 김태형

# 0. Abstract

- CVT는 labeled + unlabeled data를 사용해 LM의 표현을 향상시키는 semi-supervised learning algorithm.
- CVT는 multi-task learning과 결합할 때 특히 효과적!

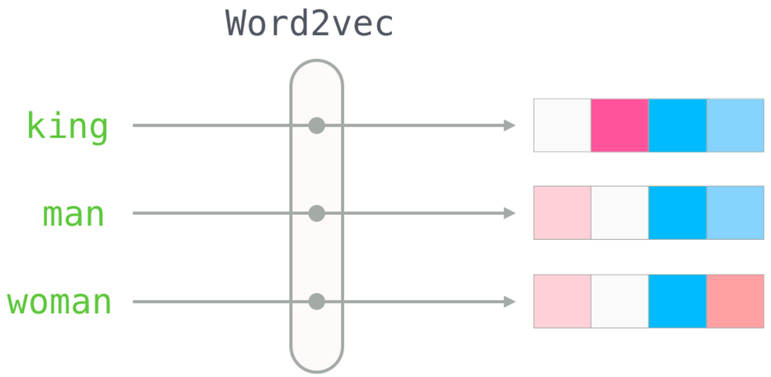
# Introduce

# 1. Introduce

- 딥러닝 모델은 많은 양의 labeled deta로 훈련했을 때 제일 잘 작동하지만, 라벨을 붙이는데는 돈이 많이 든다.
- 그래서 unlabeled example을 활용하는 semi-supervised 학습 기술인 word vector를 훈련하는 방식을 사용함.

# 1. Introduce

**Unsupervised representation learning algorithm**



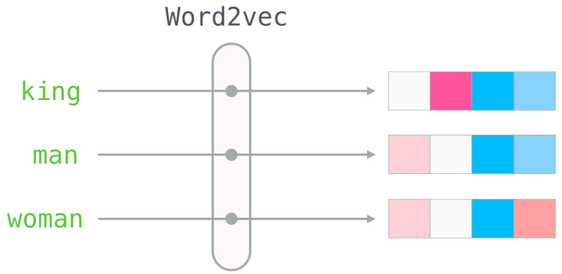

ELMo

많은 양의 unlabeled text를 활용할 수 있음 => supervised NLP Model 정확도 향상!

# 1. Introduce

- 주요 단점: 첫 번째 표현 학습 단계는 라벨링된 데이터의 장점을 얻을 수가 없다.
- 아무래도 모델은 특정 작업을 목표로 하는 표현 대신, 일반적으로 효과적인 표현을 학습하기 위해 훈련하기 때문.
- self-training 같은 semi-supervised 학습 알고리즘은 labeled + unlabeled 작업에 대해 지속적으로 학습함.
- NLP에서도 효과적이나, 보통 neural model에서는 사용을 잘 안한다.

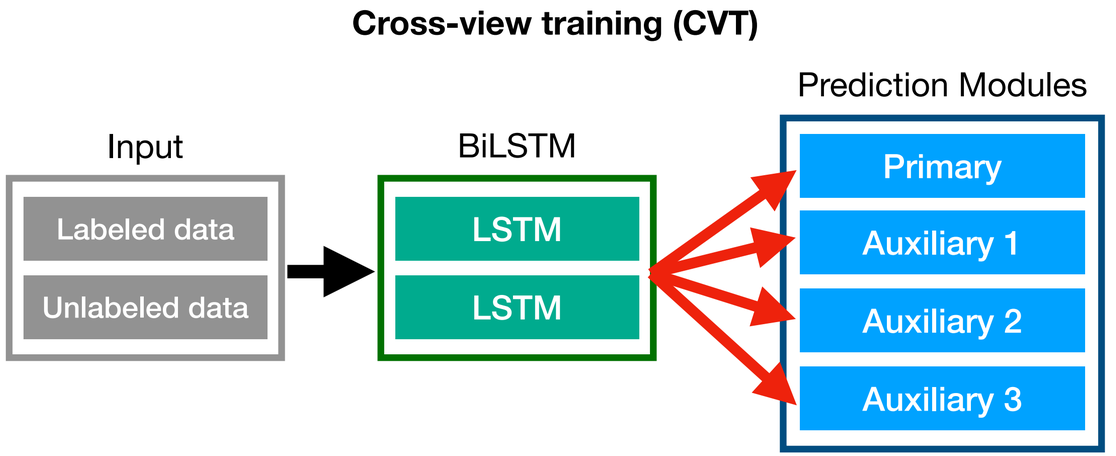
**Unsupervised representation learning algorithm**




ELMo

# 1. Introduce

- 뉴럴 시퀀스 모델에서 잘 작동하는 새로운 self-training 알고리즘: CVT
- CVT는 기존 self-training이 가지는 고질적인 문제점(student가 이미 모델이 훈련되고 있는 예측을 만들어냄)을 해결한다.

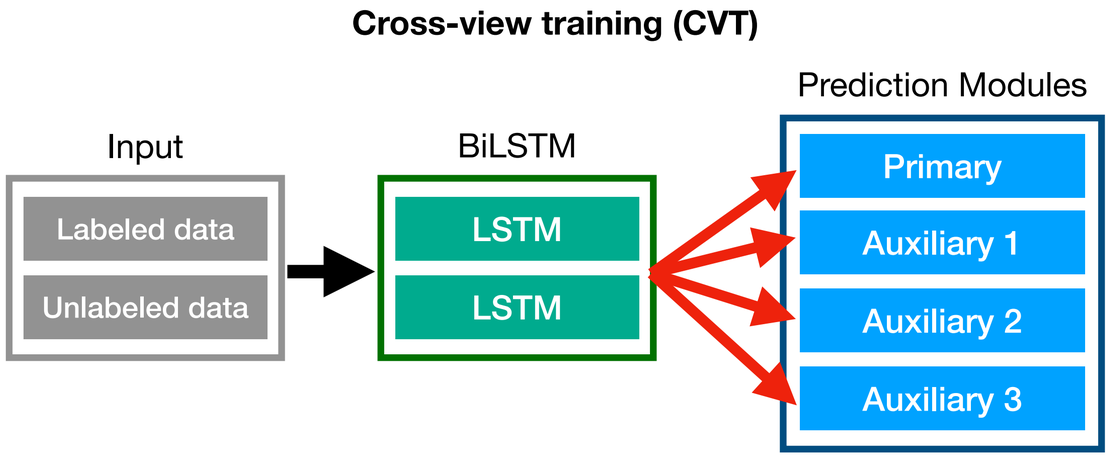


# 1. Introduce

- CVT는 이에 대한 해결책으로, multi-view learning으로부터 영감을 얻어, input의 다른 view들을 걸쳐 일관된 결과를 생성하기 위해 모델을 학습.

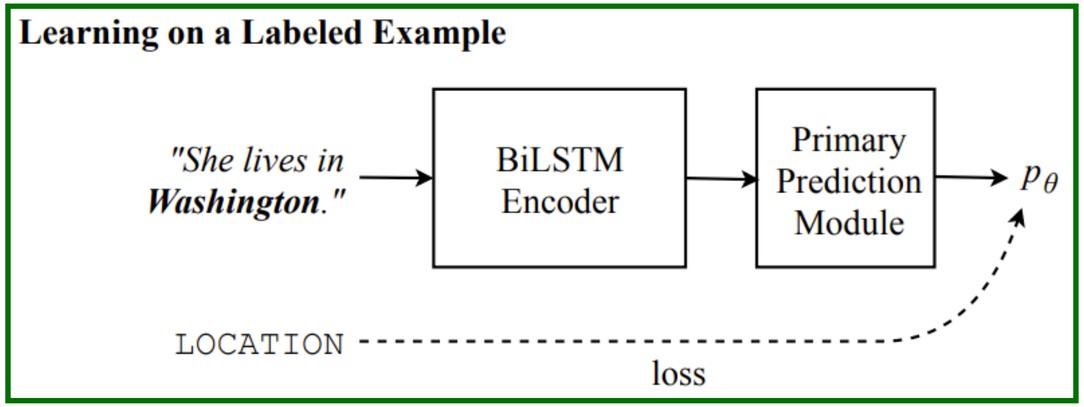
- student로서 전체 모델을 학습하는 방법이 아닌, CVT에 vector 표현을 예측으로 바꿔주는 뉴럴 네트워크인 auxiliary prediction module을 추가하고, student로서 이들을 훈련시킴.

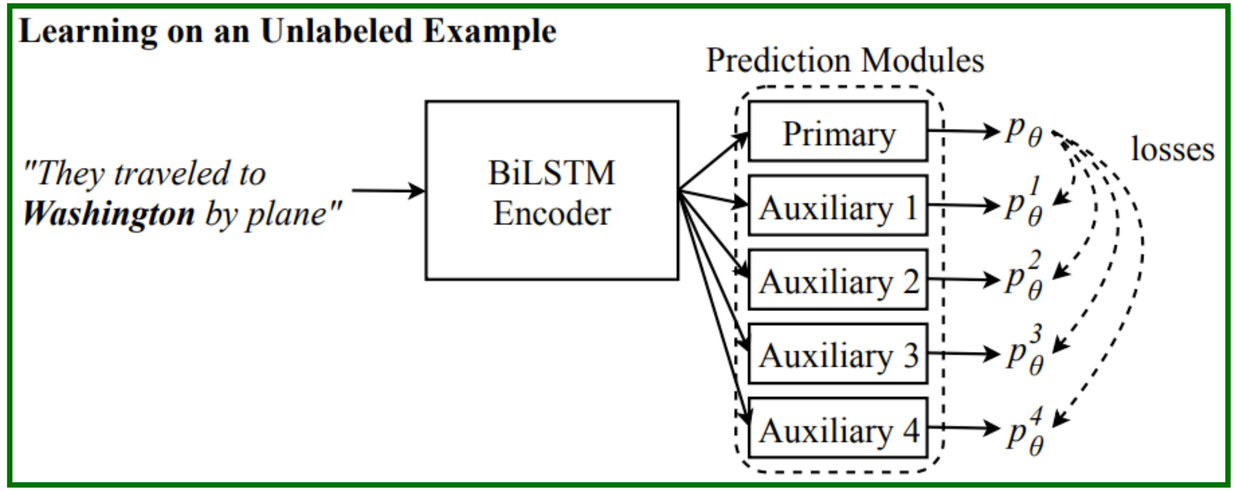
- 각 student 예측 모듈의 input은 전체 input의 제한된 뷰에 해당하는 모델의 중간 표현의 하위집합.



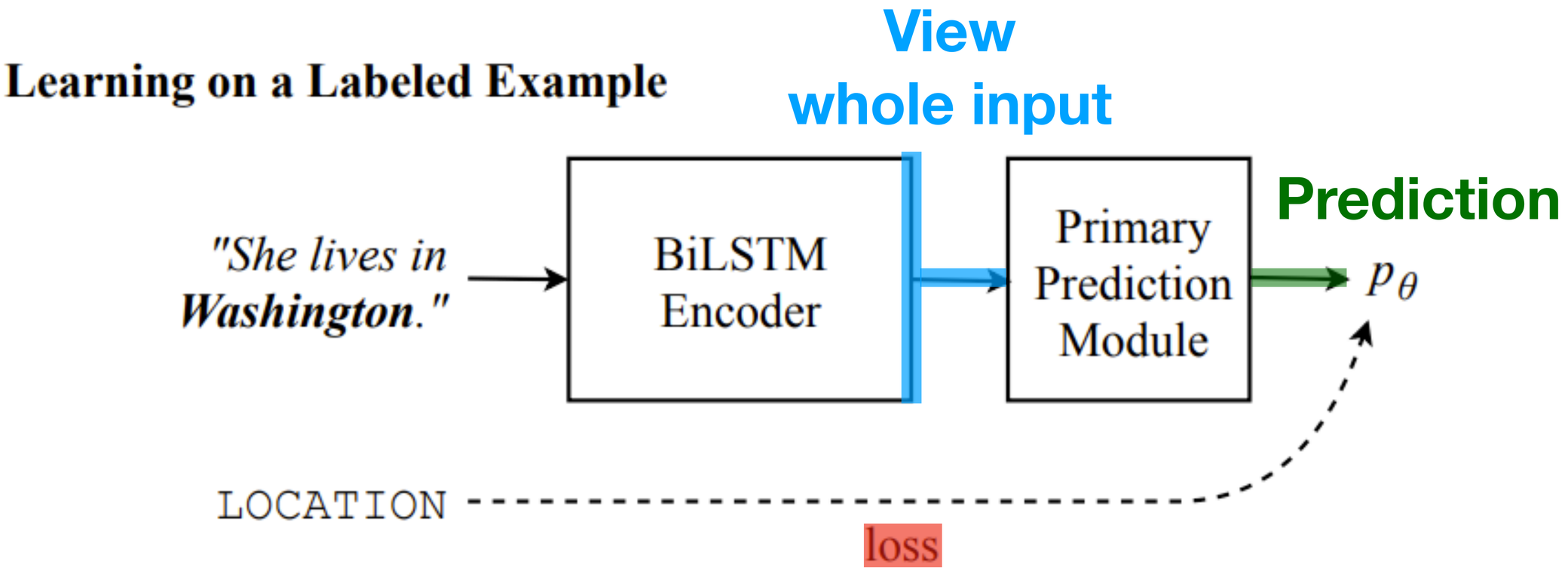
# Cross-View Training

# 2.0 Overview


Learning on a Labeled Example
"She lives in Washington."
BiLSTM Encoder
Primary Prediction Module
$p_\theta$
LOCATION
loss


Learning on an Unlabeled Example
Prediction Modules
"They traveled to Washington by plane"
BiLSTM Encoder
Primary
Auxiliary 1
Auxiliary 2
Auxiliary 3
Auxiliary 4
$p_\theta$
$p_\theta^1$
$p_\theta^2$
$p_\theta^3$
$p_\theta^4$
losses

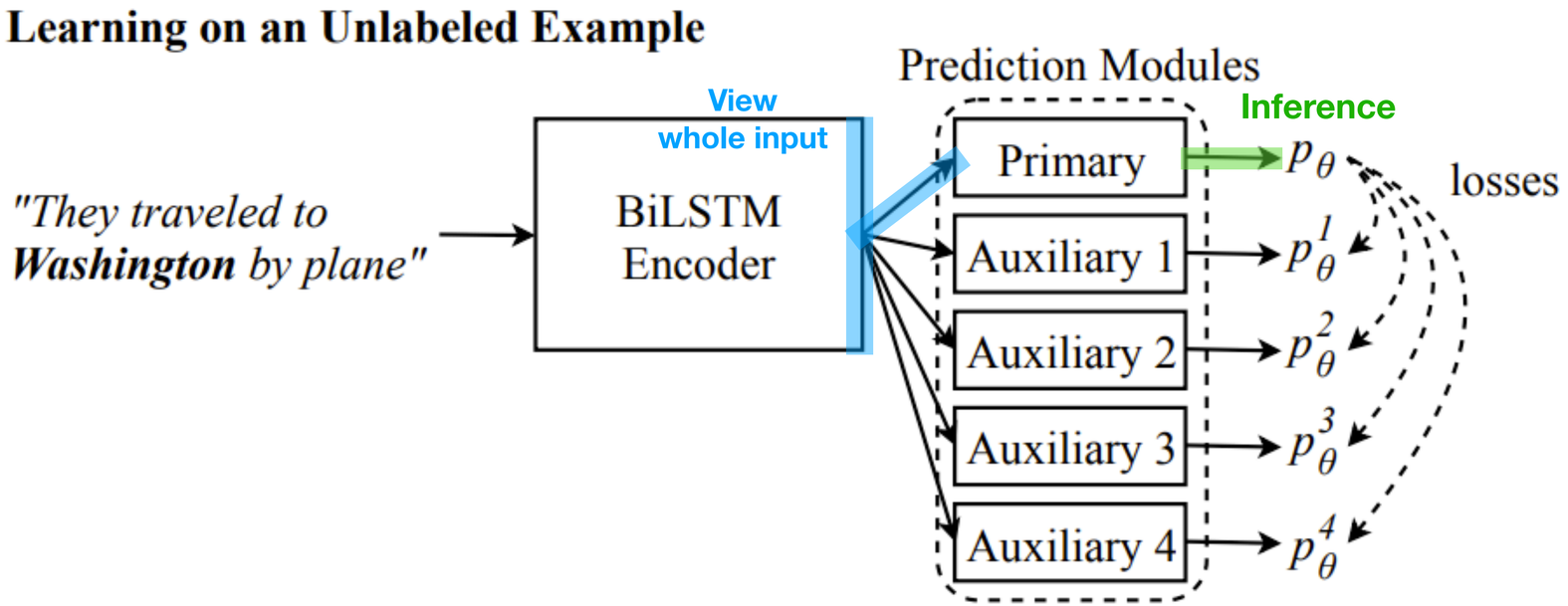
# 2.1 Method




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

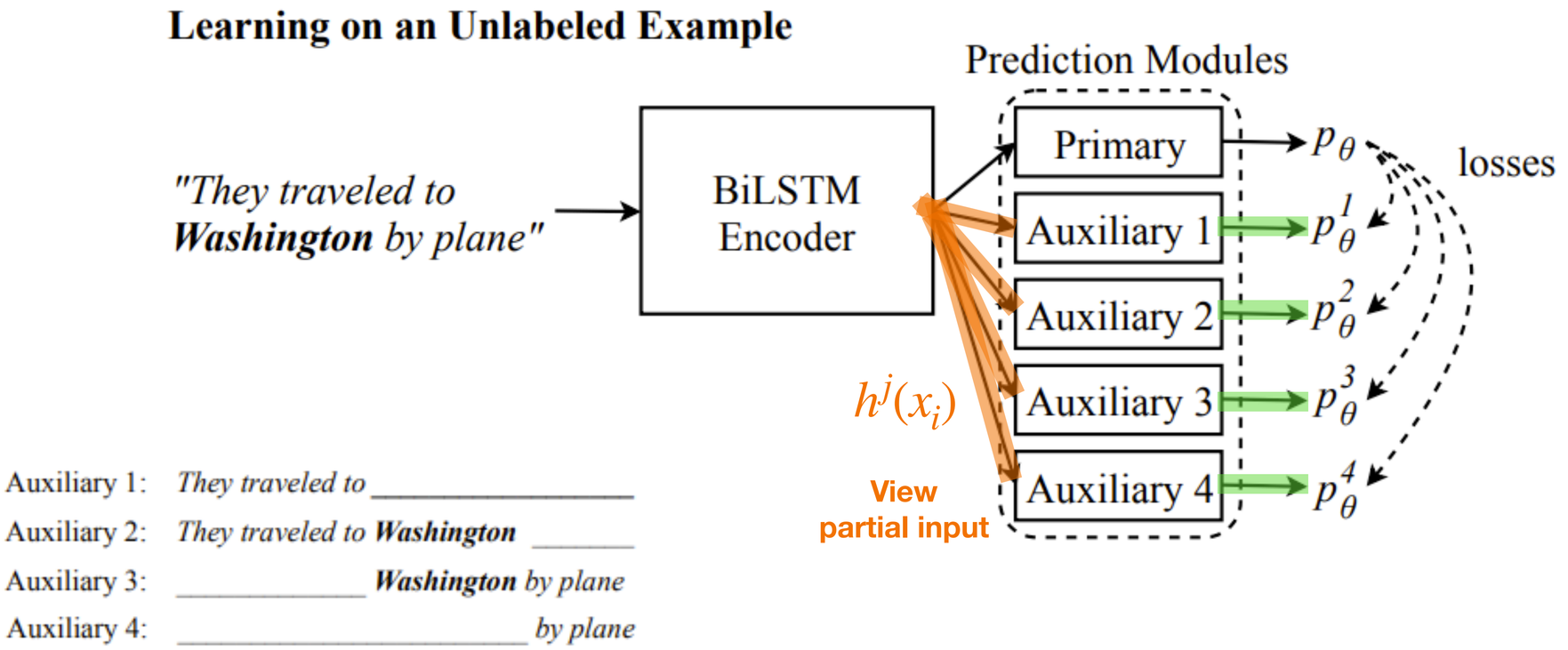
c.f. D_l = 라벨이 붙은 데이터 셋

# 2.1 Method



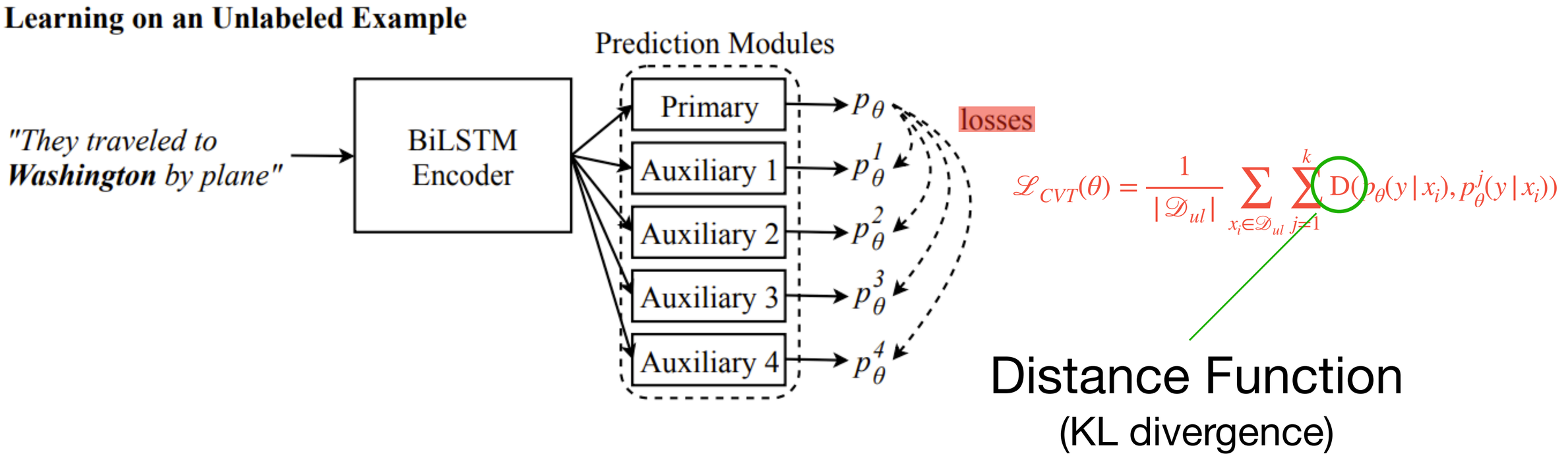

1. inference를 먼저 수행해 soft target을 만든다.

# 2.1 Method




2. Auxiliary prediction module을 사용. 단 훈련할 때만

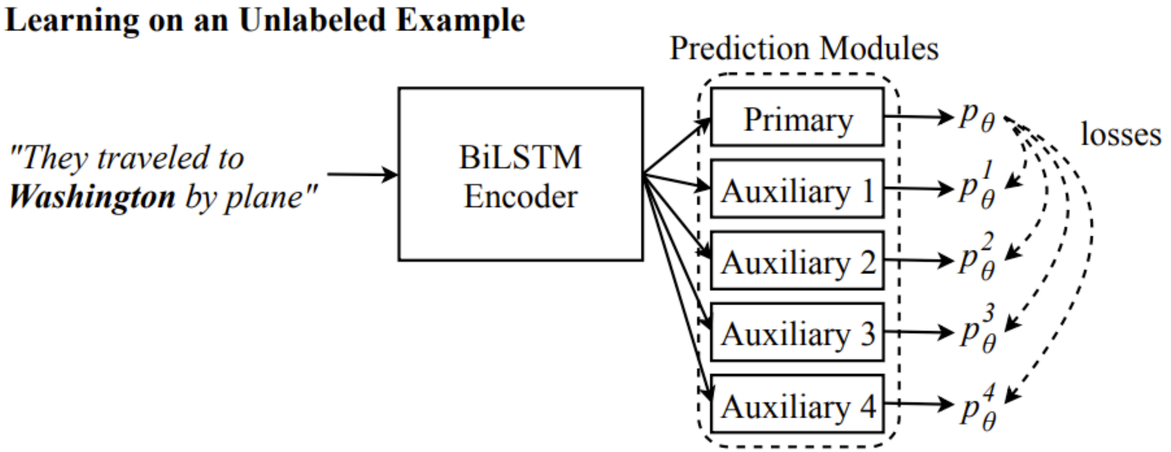
# 2.1 Method




3. Primary 모듈과 일치하도록 공동으로(jointly) 훈련.

# 2.1 Method

- 트레이닝하는 동안 primary module의 prediction은 고정됨.
- Auxiliary module을 학습 => 입력으로 받아들이는 표현이 개선되어 모델의 입력 중 일부를 사용할 수 없는 경우에도 예측을 만드는 에 유용하다.
- 따라서 동일한 공유 표현을 기반으로 구축된 primary prediction module이 개선된다.



# 2.1 Method

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{CVT}}$$

Supervised loss와 cvt loss를  total loss로 결합 후
SGD로 최소화함.

# 2.1 Method

- 대부분의 neural network에서, prediction module들을 더 추가하는 것은 모델이 표현을 만드는 부분(RNN, CNN) 대비 계산 비용이 싸다.
- 따라서 앞서 제안한 방법은 대부분의 작업에 대한 다른 self-training 접근 방식에 비해 훈련 시간에 overhead를 거의 주지 않는다.
- CVT는 inference 시간이나 fully-trained 모델의 파라미터 수를 바꾸지 않는다 => auxiliary prediction module은 훈련할 때만 사용되기 때문!

# 2.2 Combining CVT with multi-task learning

- CVT는 shared BiLSTM encoder 위에 다른 작업에 대한 prediction module을 추가해 multi-task learning과 쉽게 결합할 수 있음.
- Supervised learning 동안, 랜덤하게 과제를 선택하고 과제에 대한 labeled data의 미니배치를 사용해 L_sup를 업데이트.
- Unlabeled data: 한 번에 모든 과제에 걸쳐 L_cvt 공동으로 최적화 => primary prediction module inference => auxiliary prediction module로 예측 정보 학습(마찬가지로 labeled, unlabeled 번갈아서 수행)

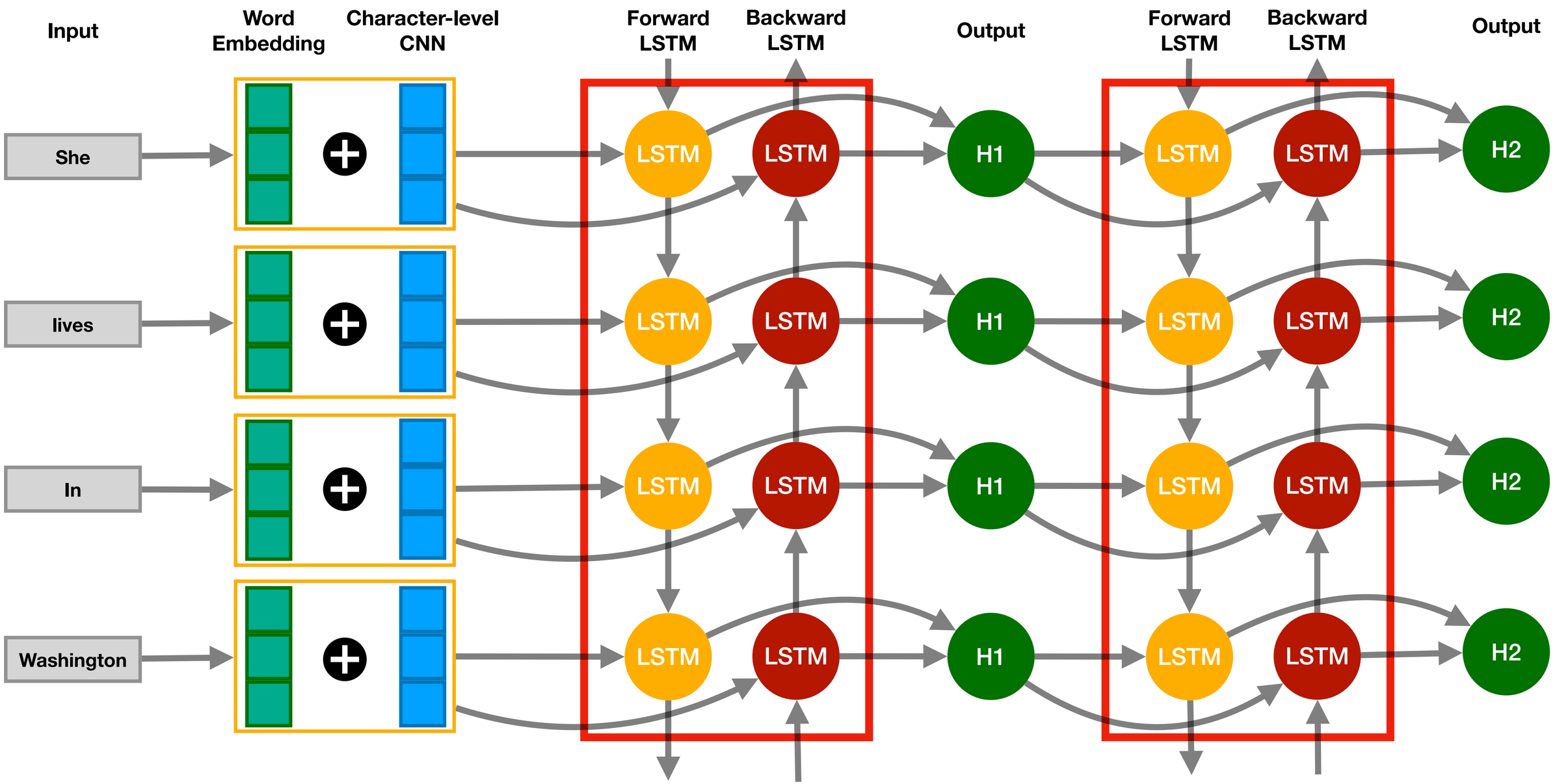
# 2.2 Combining CVT with multi-task learning

- 많은 과제에 걸쳐 labeled된 데이터들은 multi-task 시스템에서 학습하는 데 유용하지만, 대부분의 데이터 셋에는 하나의 작업만 labeling됨.

- Multi-task cvt의 장점: 모델이 unlabeled data로부터 모든 과제에 대한 labeled data를 만들어냄 => 효율성과 훈련시간 상당히 개선!

# Cross-View Training Models

# 3.1 Bi-LSTM sentence encoder



# 3.2 CVT for sequence tagging

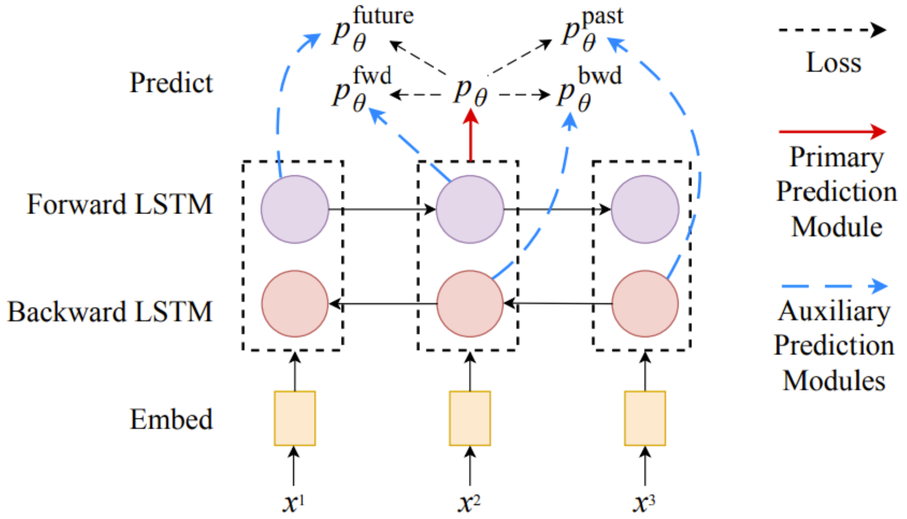
Input token $x_i^t$ 는 대응되는 라벨인 $y_i^t$를 가짐

**Primary prediction module** encorder의 두 레이어의 출력 $h_1^t, h_2^t$ 를 input으로 받아 하나의 hidden layer로 결과값을 직접 학습

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

**Auxiliary prediction module** encorder의 두 레이어의 출력 중 첫 번째 결과인 $h_1$만을 사용함.
$h_2$는 $h_1$을 input으로 받음 => 전체 문장에 대한 정보를 이미 봐버렸기 때문에 제한된 뷰를 본다고 할 수가 없다!

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



# 3.3 CVT for dependency parsing

문장 안에 있는 단어들은 그래프에서 노드로 취급되며, 각각의 단어는 다른 단어로 가는 정확히 하나의 edge(u, t, r)만을 받는다.
u: head(source node), t: dependent(destination node), r: relation(edge type)

**Primary prediction module**

1) 인코더로부터 생성된 분석할 head와 dependent에 대한 표현을 각각 분리된 hidden layer에 입력
2) 이 두 노드를 연결할 후보 edge들에 대해 점수를 매기기 위해 bilinear classifier 사용
3) softmax 레이어를 통과해 확률을 만듦.

Head 단어 전체 표현 / dependenct 단어 전체 표현

$$p_\theta((u,t,r)|x_i) \propto e^{s(h_1^u(x_i)\oplus h_2^u(x_i),\, h_1^t(x_i)\oplus h_2^t(x_i),\, r)}$$

$$s(z_1, z_2, r) = \mathrm{ReLU}(W_{\text{head}} z_1 + b_{\text{head}})\,(W_r + W)\,\mathrm{ReLU}(W_{\text{dep}} z_2 + b_{\text{dep}})$$

모든 관계에서 공유되는 가중치 행렬

후보 관계에 특화된 가중치 행렬

# 3.3 CVT for dependency parsing

**Auxiliary prediction module** Sequence tagging 모델과 유사하게 4개의 auxiliary prediction module을 추가.
인코더의 h1 단방향만 보고 edge 점수를 계산하므로 head, dependent의 일부만 보고 판단하게 된다.

$$p_\theta^{\text{fwd-fwd}}((u,t,r)|x_i) \propto e^{s^{\text{fwd-fwd}}(\overrightarrow{h}_1^u(x_i), \overrightarrow{h}_1^t(x_i), r)}$$

$$p_\theta^{\text{fwd-bwd}}((u,t,r)|x_i) \propto e^{s^{\text{fwd-bwd}}(\overrightarrow{h}_1^u(x_i), \overleftarrow{h}_1^t(x_i), r)}$$

$$p_\theta^{\text{bwd-fwd}}((u,t,r)|x_i) \propto e^{s^{\text{bwd-fwd}}(\overleftarrow{h}_1^u(x_i), \overrightarrow{h}_1^t(x_i), r)}$$

$$p_\theta^{\text{bwd-bwd}}((u,t,r)|x_i) \propto e^{s^{\text{bwd-bwd}}(\overleftarrow{h}_1^u(x_i), \overleftarrow{h}_1^t(x_i), r)}$$

# 3.4 CVT for seq2seq learning

Attention을 사용하는 encoder seq2seq모델 사용. input token $x_i^t$ 는 대응되는 라벨인 $y_i^t$를 가짐

**Primary prediction module** Bilinear 어텐션 메커니즘을 사용하는 것 외에는 기본 어텐션 메커니즘과 동일

**Auxiliary prediction module**

- 두 개의 auxiliary decoder를 추가함.
- auxiliary decoder는 primary decoder와 임베딩, LSTM 파라미터를 공유
- 어텐션 메커니즘과 softmax layer 파라미터는 다름
  - 첫 번째로 attention dropout 적용해 input view 제한하고, attention weight의 일부를 무작위로 0으로 조정(Knol
  - 두 번째로 현재 단어 대신 target sequence의 다음 단어를 예측하기 위해 훈련.

$$p_\theta^{\text{future}}(y_i^t \mid y_i^{<t}, x_i) = \text{softmax}(W_s^{\text{future}} a_{t-1}^{\text{future}})$$

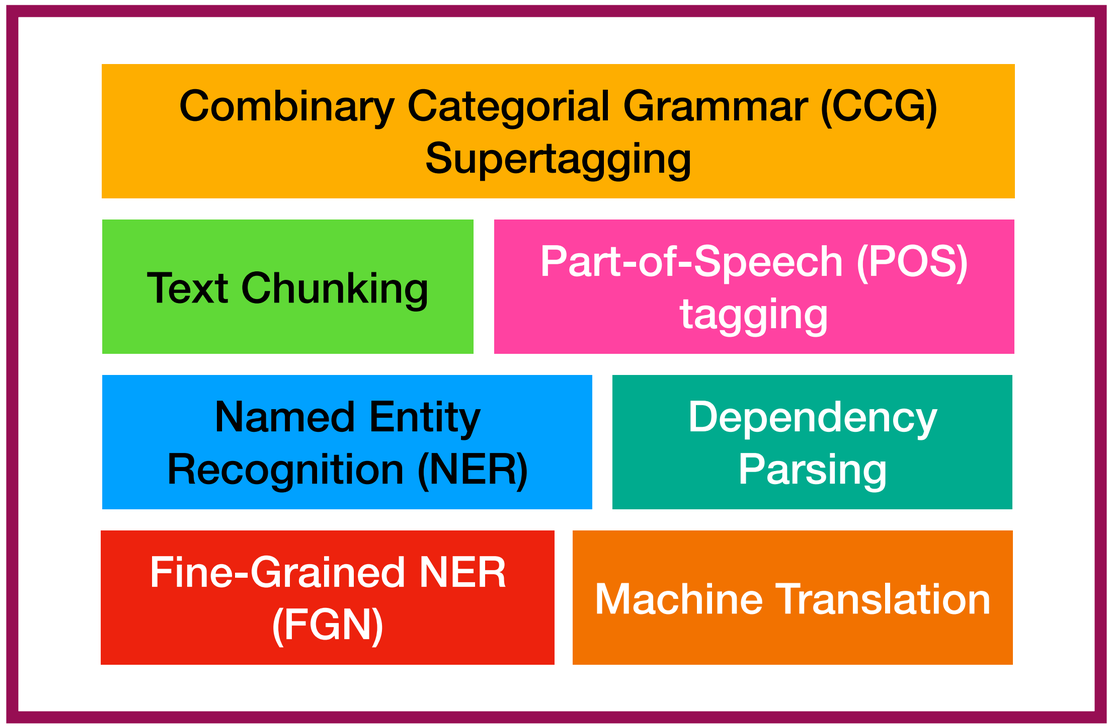
  - 라벨이 붙지 않은 예제에서는 target sequence가 없음 => input sequnce에서 primary decoder로 beam search를 실행해 hard target 생성

# Experiments

# 4.0 Experiments

## 7가지 task에 대한 strong baseline과 비교

### Labeled example




### Unlabeled example

1 Billion Word Language Model Benchmark

# 4.1 Model details and baselines

## Model details

- Unlabeled example에 대한 soft target을 만들기 위해 primary prediction module이 돌아갈 때를 제외하고는 dropout을 트레이닝할 때 적용함.

- Auxiliary prediction module 외에, 하위 집합이 아닌 전체 input을 보는 또 다른 input을 추가하는 것이 결과를 약간 개선한다는 것을 알게 됨(primary와는 다르게 해당 표현에 dropout이 적용됨)

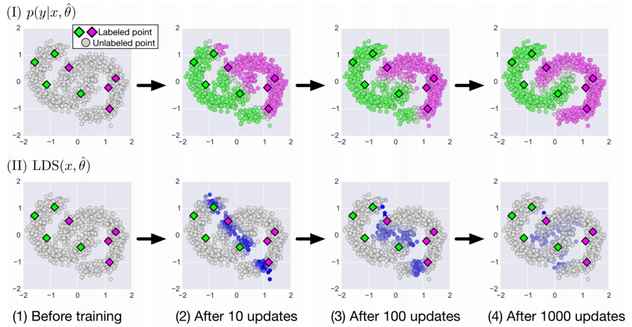
| Parameter | Value |
|---|---|
| Word Embeddings Initializiation | 300d GloVe 6B |
| Character Embedding Size | 50 |
| Character CNN Filter Widths | [2, 3, 4] |
| Character CNN Num Filters | 300 (100 per filter width) |
| Encoder LSTM sizes | 1024 for the first layer, 512 for the second one |
| Encoder 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 6: Hyperparameters for the model.

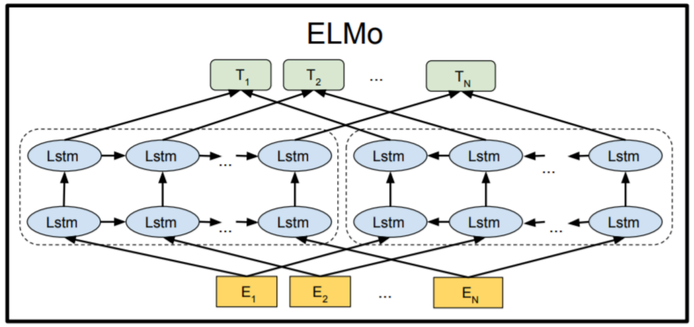
# 4.1 Model details and baselines

## Baselines


Word
dropout



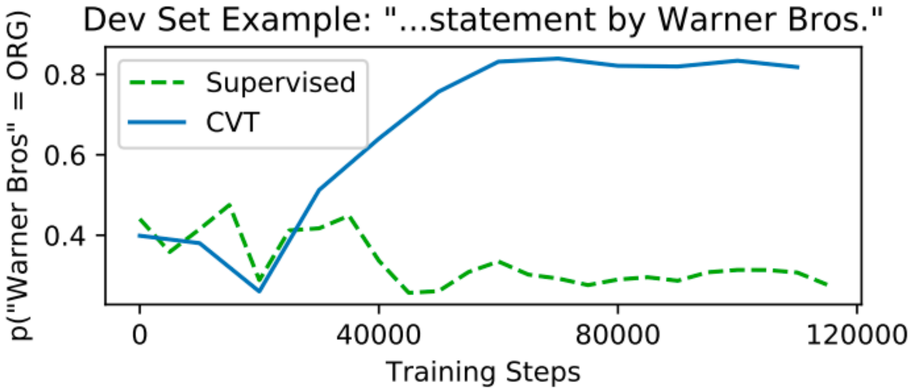

Virtual
Adversarial Training




ELMo

# 4.2 Result

| Method | CCG Acc. | Chunk F1 | NER F1 | FGN F1 | POS Acc. | Dep. Parse UAS | Dep. Parse LAS | Translate BLEU |
|---|---|---|---|---|---|---|---|---|
| Shortcut LSTM (Wu et al., 2017) | 95.1 | | | | 97.53 | | | |
| ID-CNN-CRF (Strubell et al., 2017) | | | 90.7 | 86.8 | | | | |
| JMT† (Hashimoto et al., 2017) | | 95.8 | | | 97.55 | 94.7 | 92.9 | |
| TagLM* (Peters et al., 2017) | | 96.4 | 91.9 | | | | | |
| ELMo* (Peters et al., 2018) | | | 92.2 | | | | | |
| Biaffine (Dozat and Manning, 2017) | | | | | | 95.7 | 94.1 | |
| Stack Pointer (Ma et al., 2018) | | | | | | 95.9 | 94.2 | |
| Stanford (Luong and Manning, 2015) | | | | | | | | 23.3 |
| Google (Luong et al., 2017) | | | | | | | | 26.1 |
| Supervised | 94.9 | 95.1 | 91.2 | 87.5 | 97.60 | 95.1 | 93.3 | 28.9 |
| Virtual Adversarial Training* | 95.1 | 95.1 | 91.8 | 87.9 | 97.64 | 95.4 | 93.7 | – |
| Word Dropout* | 95.2 | 95.8 | 92.1 | 88.1 | 97.66 | 95.6 | 93.8 | 29.3 |
| ELMo (our implementation)* | 95.8 | 96.5 | 92.2 | 88.5 | 97.72 | 96.2 | 94.4 | 29.3 |
| ELMo + Multi-task*† | 95.9 | 96.8 | 92.3 | 88.4 | **97.79** | 96.4 | 94.8 | – |
| CVT* | 95.7 | 96.6 | 92.3 | 88.7 | 97.70 | 95.9 | 94.1 | **29.6** |
| CVT + Multi-task*† | 96.0 | 96.9 | 92.4 | 88.4 | 97.76 | 96.4 | 94.8 | – |
| CVT + Multi-task + Large*† | **96.1** | **97.0** | **92.6** | **88.8** | 97.74 | **96.6** | **95.0** | – |



# 4.2 Result

## CVT + Multi-Task

- 기계번역을 제외한 모든 task를 수행하기 위해 단일 공유 encoder CVT 모델을 훈련.
- Multi-task 학습은 세분화된 NER을 제외한 모든 과제에서 때로는 큰 폭으로 결과를 개선
- Many-task nlp에서 모델이 크고 많은 양의 데이터에 대해 교육된 경우, 파라미터 공유만으로 효과적인 many-task 학습에 충분할 수 있다는 결과 보여줌.

**A Joint Many-Task Model:**
**Growing a Neural Network for Multiple NLP Tasks**

**Kazuma Hashimoto,* Caiming Xiong,† Yoshimasa Tsuruoka, and Richard Socher**
The University of Tokyo
{hassy, tsuruoka}@logos.t.u-tokyo.ac.jp
Salesforce Research
{cxiong, rsocher}@salesforce.com

**Abstract**

Transfer and multi-task learning have traditionally focused on either a single source-target pair or very few, similar tasks. Ideally, the linguistic levels of morphology, syntax and semantics would benefit each other by being trained in a single model. We introduce a joint many-task model together with a strategy for successively growing its depth to solve increasingly complex tasks. Higher layers include shortcut connections to lower-level task predictions to reflect linguistic hierarchies. We use a simple regularization term to allow for optimizing all model weights to improve one task's loss without exhibiting catastrophic interference of the other tasks. Our single end-to-end model obtains state-of-the-art or competitive results on five different tasks from tagging, parsing, relatedness, and entailment tasks.

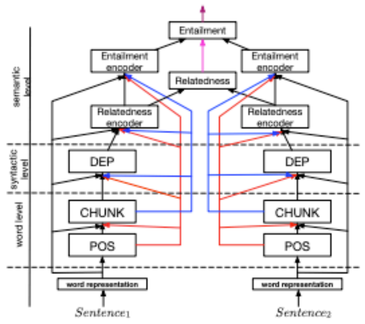



Figure 1: Overview of the joint many-task model predicting different linguistic outputs at successively deeper layers.

different tasks either entirely separately or at the same depth (Collobert et al., 2011).

# 4.2 Result

## CVT + Multi-Task

- CVT가 모델이 하나의 과제를 학습하는 동안 다른 과제를 '잊어버리는 것' 에 대한 위험을 경감시킨다고 믿음.
- Multi-task cvt 동안, 모델은 모든 과제에 걸쳐서 라벨이 붙지 않은 예제를 예측하며, 인공적으로 모든 과제에 labeled data를 만들기 때문에, 모델이 한 번에 하나의 작업만 보지 않는다.
- Without forgetting algorithm 학습과 유사하다.

**Overcoming catastrophic forgetting in neural networks**

James Kirkpatrick[a], Razvan Pascanu[a], Neil Rabinowitz[a], Joel Veness[a], Guillaume Desjardins[a], Andrei A. Rusu[a], Kieran Milan[a], John Quan[a], Tiago Ramalho[a], Agnieszka Grabska-Barwinska [a], Demis Hassabis[a], Claudia Clopath[b], Dharshan Kumaran[a], and Raia Hadsell[a]

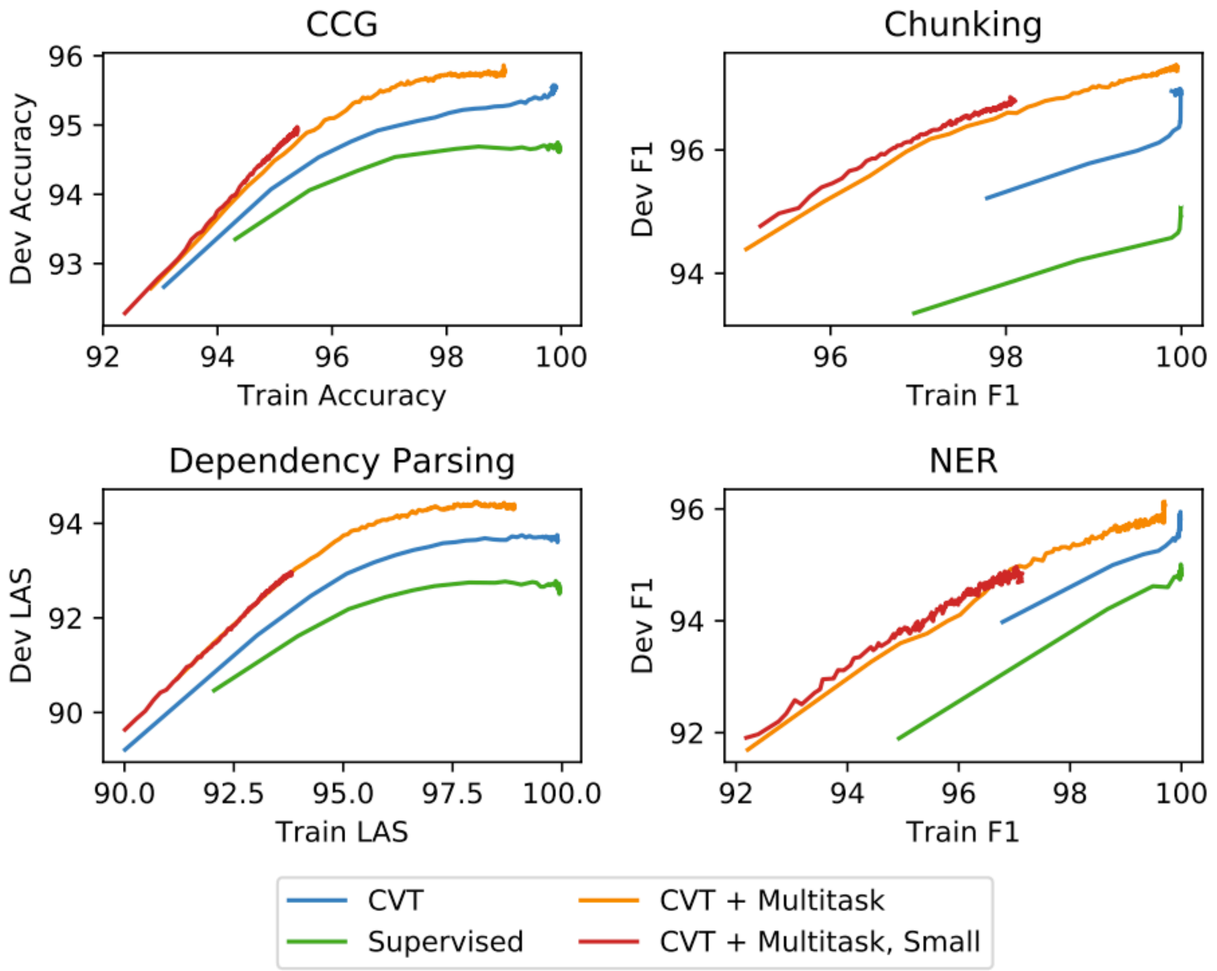
[a]DeepMind, London, N1C 4AG, United Kingdom
[b]Bioengineering department, Imperial College London, SW7 2AZ, London, United Kingdom

**Abstract**

The ability to learn tasks in a sequential fashion is crucial to the development of artificial intelligence. Neural networks are not, in general, capable of this and it has been widely thought that *catastrophic forgetting* is an inevitable feature of connectionist models. We show that it is possible to overcome this limitation and train networks that can maintain expertise on tasks which they have not experienced for a long time. Our approach remembers old tasks by selectively slowing down learning on the weights important for those tasks. We demonstrate our approach is scalable and effective by solving a set of classification tasks based on the MNIST hand written digit dataset and by learning several Atari 2600 games sequentially.

# 4.2 Result

## Model Generalization



# 4.2 Result

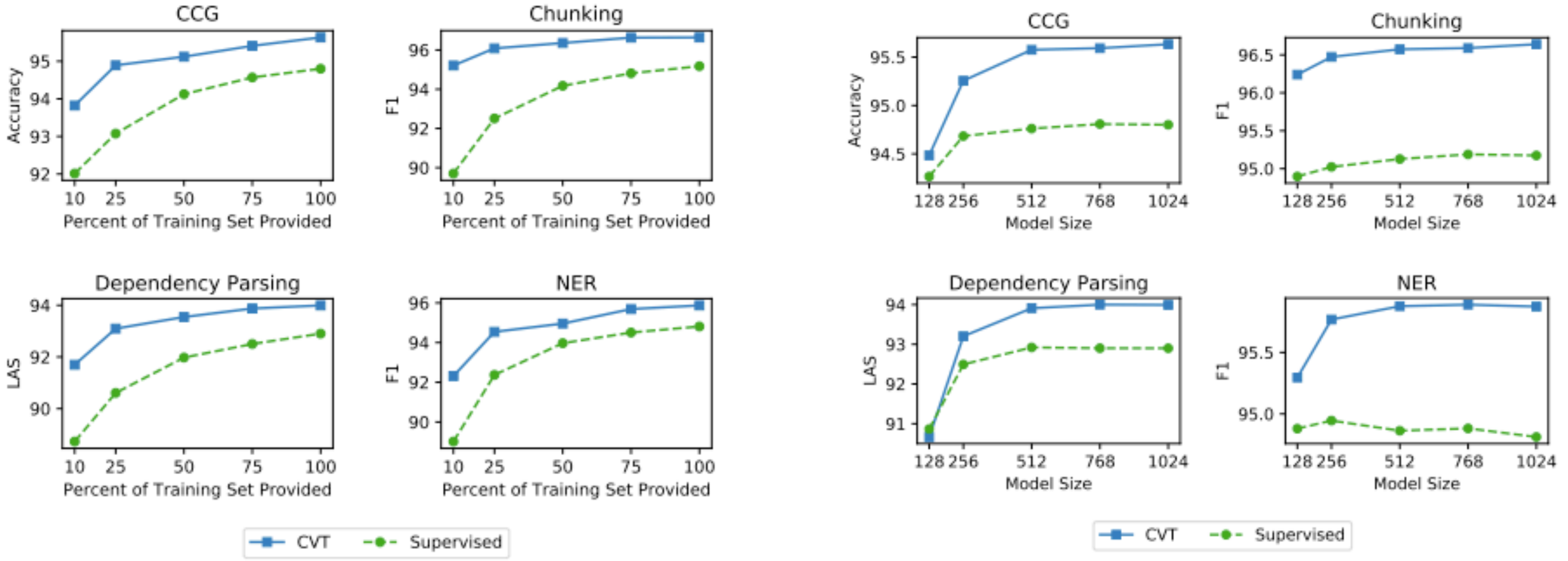
## Auxiliary Prediction Module Ablation

| Model | CCG | Chnk | NER | FGN | POS |
|---|---|---|---|---|---|
| Supervised | 94.8 | 95.5 | 95.0 | 86.0 | 97.59 |
| CVT | 95.6 | 97.0 | 95.9 | 87.3 | 97.66 |
| no fwd/bwd | –0.1 | –0.2 | –0.2 | –0.1 | –0.01 |
| no future/past | –0.3 | –0.4 | –0.4 | –0.3 | –0.04 |

Sequence tagging task 에서 auxiliary prediction module은 매우 중요함!

# 4.2 Result

## Training Models on Small Datasets




CVT가 labeled data를 25%만 사용 > 100% 사용하는 fully supervised model

# 4.2 Result

## Training Larger Models

- 이전 작업에서 대부분의 sequence tagger와 dependency parser는 작은 LSTM을 사용 => 대형 모델의 경우 성능 면에서 득이 되지 않음.
- Supervised 접근 또한 모델 사이즈를 키우는 것으로 큰 이득이 없다.
- CVT accuracy를 사용하면 모델 크기에 따라 더 잘 스케일링된다는 것을 발견.
- 적절한 semi-supervised learning 방법은 제한된 양의 labeled data를 사용하는 NLP task에서 더 크고 더욱 정교한 모델 개발을 가능하게 한다는 것을 제안.

# 4.2 Result

## Generalizable Representation

- 다섯 가지 태스크에 대한 CVT + Multi-task 학습을 진행 => encoder freezing => 여섯 번째 과제에서 prediction module만 학습.
- Prediction module만 학습하는 것은 매우 빠름.
- Multi-task 표현 위에 있는 예측 모듈만 훈련하는 것은 ELMo 임베딩, 심지어 바닐라 supervised model을 능가.

| Model | CCG | Chnk | NER | FGN | POS | Dep. |
|---|---|---|---|---|---|---|
| Supervised | 94.8 | 95.6 | 95.0 | 86.0 | 97.59 | 92.9 |
| CVT-MT frozen | 95.1 | 96.6 | 94.6 | 83.2 | 97.66 | 92.5 |
| ELMo frozen | 94.3 | 92.2 | 91.3 | 80.6 | 97.50 | 89.4 |

# Conclusion

# 5. Conclusion

- Semi-supervised 학습을 위한 새로운 방법인 CVT를 제안.
- 모델이 unlabeled data에 대한 자체적인 예측을 효과적으로 활용할 수 있으며, input 중 일부를 사용할 수 없는 경우에도 정확한 예측을 산출해내는 효과적인 표현을 만들 수 있도록 교육이 가능
- CVT가 multi-task learning과 결합했을 때, 7가지 NLP task에서 훌륭한 결과를 달성.