**Sequence to Sequence Learning with Neural Networks**

Ilya Sutskever, Oriol Vinyals, Quoc V. Le

**Abstract**

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT’14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM’s BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous best result on this task. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM’s performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.심층 신경망(DNN)은 어려운 학습 작업에서 우수한 성능을 달성한 강력한 모델입니다. DNN은 레이블이 지정된 큰 훈련 세트를 사용할 수 있을 때마다 잘 작동하지만 시퀀스를 시퀀스에 매핑하는 데 사용할 수는 없습니다. 이 논문에서 우리는 시퀀스 구조에 대한 최소한의 가정을 하는 시퀀스 학습에 대한 일반적인 종단 간 접근 방식을 제시합니다. 우리의 방법은 다층 LSTM(Long Short-Term Memory)을 사용하여 입력 시퀀스를 고정 차원의 벡터에 매핑한 다음 다른 깊은 LSTM을 사용하여 벡터에서 대상 시퀀스를 디코딩합니다. 주요 결과는 WMT'14 데이터 세트의 영어에서 프랑스어로의 번역 작업에서 LSTM으로 생성된 번역이 전체 테스트 세트에서 BLEU 점수 34.8을 달성했다는 것입니다. 단어. 또한 LSTM은 긴 문장에 어려움이 없었습니다. 비교를 위해 구문 기반 SMT 시스템은 동일한 데이터 세트에서 BLEU 점수 33.3을 달성했습니다. 앞서 언급한 SMT 시스템에서 생성된 1000개 가설의 순위를 재지정하기 위해 LSTM을 사용했을 때 BLEU 점수는 36.5로 증가했으며 이는 이 작업에 대한 이전 최고의 결과에 가깝습니다. LSTM은 또한 어순에 민감하고 능동태와 수동태에 상대적으로 변하지 않는 합리적인 구와 문장 표현을 학습했습니다. 마지막으로, 우리는 모든 소스 문장(목표 문장은 제외)에서 단어의 순서를 반대로 하면 LSTM의 성능이 현저하게 향상된다는 것을 발견했습니다. 그렇게 하면 소스와 대상 문장 사이에 많은 단기 종속성이 도입되어 최적화 문제가 더 쉬워지기 때문입니다.

**1. Introduction**

Deep Neural Networks (DNNs) are extremely powerful machine learning models that achieve excellent performance on difficult problems such as speech recognition [13, 7] and visual object recognition [19, 6, 21, 20]. DNNs are powerful because they can perform arbitrary parallel computation for a modest number of steps. A surprising example of the power of DNNs is their ability to sort N N-bit numbers using only 2 hidden layers of quadratic size [27]. So, while neural networks are related to conventional statistical models, they learn an intricate computation. Furthermore, large DNNs can be trained with supervised backpropagation whenever the labeled training set has enough information to specify the network’s parameters. Thus, if there exists a parameter setting of a large DNN that achieves good results (for example, because humans can solve the task very rapidly), supervised backpropagation will find these parameters and solve the problem.

심층 신경망(DNN)은 음성 인식 및 시각적 개체 인식과 같은 어려운 문제에서 우수한 성능을 달성하는 매우 강력한 기계 학습 모델입니다. DNN은 적당한 수의 단계에 대해 임의의 병렬 계산을 수행할 수 있기 때문에 강력합니다. DNN의 능력에 대한 놀라운 예는 2차 크기의 은닉 레이어 2개만 사용하여 N개의 N비트 숫자를 정렬하는 기능입니다[27]. 따라서 신경망은 기존의 통계 모델과 관련이 있지만 복잡한 계산을 배웁니다. 또한 레이블이 지정된 훈련 세트에 네트워크의 매개변수를 지정하기에 충분한 정보가 있을 때마다 대형 DNN을 지도 역전파(supervised backpropagation)로 훈련할 수 있습니다. 따라서 좋은 결과를 얻는 대규모 DNN의 매개변수 설정이 있는 경우(예: 인간이 작업을 매우 빠르게 해결할 수 있기 때문에), 감독 역전파는 이러한 매개변수를 찾아 문제를 해결합니다.

Despite their flexibility and power, DNNs can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality. It is a significant limitation, since many important problems are best expressed with sequences whose lengths are not known a-priori. For example, speech recognition and machine translation are sequential problems. Likewise, question answering can also be seen as mapping a sequence of words representing the question to a 1 sequence of words representing the answer. It is therefore clear that a domain-independent method that learns to map sequences to sequences would be useful.

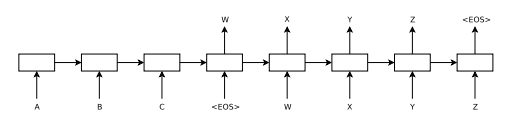
유연성과 능력에도 불구하고 DNN은 입력과 목표가 고정 차원의 벡터로 현명하게 인코딩될 수 있는 문제에만 적용될 수 있습니다. 많은 중요한 문제가 길이가 사전에 알려지지 않은 시퀀스로 가장 잘 표현되기 때문에 이는 상당한 제한입니다. 예를 들어, 음성 인식과 기계 번역은 순차적인 문제입니다. 마찬가지로 질문 답변은 질문을 나타내는 단어 시퀀스를 답변을 나타내는 단어 시퀀스 1에 매핑하는 것으로 볼 수도 있습니다. 따라서 시퀀스를 시퀀스에 매핑하는 방법을 배우는 도메인 독립적인 방법이 유용할 것이 분명합니다.

Sequences pose a challenge for DNNs because they require that the dimensionality of the inputs and outputs is known and fixed. In this paper, we show that a straightforward application of the Long Short-Term Memory (LSTM) architecture [16] can solve general sequence to sequence problems. The idea is to use one LSTM to read the input sequence, one timestep at a time, to obtain large fixeddimensional vector representation, and then to use another LSTM to extract the output sequence from that vector (fig. 1). The second LSTM is essentially a recurrent neural network language model [28, 23, 30] except that it is conditioned on the input sequence. The LSTM’s ability to successfully learn on data with long range temporal dependencies makes it a natural choice for this application due to the considerable time lag between the inputs and their corresponding outputs (fig. 1).

시퀀스는 입력과 출력의 차원을 알고 고정해야 하기 때문에 DNN에 문제가 됩니다. 이 논문에서 우리는 LSTM(Long Short-Term Memory) 아키텍처[16]의 간단한 적용이 일반적인 시퀀스 대 시퀀스 문제를 해결할 수 있음을 보여줍니다. 아이디어는 하나의 LSTM을 사용하여 입력 시퀀스를 한 번에 한 타임 스텝씩 읽고 큰 고정 차원 벡터 표현을 얻은 다음 다른 LSTM을 사용하여 해당 벡터에서 출력 시퀀스를 추출하는 것입니다(그림 1). 두 번째 LSTM은 입력 시퀀스에 따라 조건이 지정된다는 점을 제외하고 본질적으로 순환 신경망 언어 모델입니다. 장거리 시간 종속성을 가진 데이터에 대해 성공적으로 학습할 수 있는 LSTM의 기능은 입력과 해당 출력 사이의 상당한 시간 지연으로 인해 이 애플리케이션에 자연스러운 선택이 됩니다(그림 1).

There have been a number of related attempts to address the general sequence to sequence learning problem with neural networks. Our approach is closely related to Kalchbrenner and Blunsom [18] who were the first to map the entire input sentence to vector, and is related to Cho et al. [5] although the latter was used only for rescoring hypotheses produced by a phrase-based system. Graves [10] introduced a novel differentiable attention mechanism that allows neural networks to focus on different parts of their input, and an elegant variant of this idea was successfully applied to machine translation by Bahdanau et al. [2]. The Connectionist Sequence Classification is another popular technique for mapping sequences to sequences with neural networks, but it assumes a monotonic alignment between the inputs and the outputs [11].

신경망의 일반적인 시퀀스-투-시퀀스 학습 문제를 해결하기 위한 많은 관련 시도가 있었습니다. 우리의 접근 방식은 전체 입력 문장을 벡터에 매핑한 최초의 Kalchbrenner 및 Blunsom과 밀접한 관련이 있으며 Cho et al. 후자는 구문 기반 시스템에 의해 생성된 가설을 다시 채점하는 데에만 사용되었습니다. Graves는 신경망이 입력의 다른 부분에 집중할 수 있도록 하는 새로운 차별화 가능한 주의 메커니즘을 도입했으며 이 아이디어의 우아한 변형이 Bahdanau et al.에 의해 기계 번역에 성공적으로 적용되었습니다. [2]. 연결주의자 시퀀스 분류는 신경망을 사용하여 시퀀스를 시퀀스에 매핑하는 또 다른 인기 있는 기술이지만 입력과 출력 사이의 단조로운 정렬을 가정합니다.



.

Figure 1: Our model reads an input sentence “ABC” and produces “WXYZ” as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.그림 1: 우리 모델은 입력 문장 "ABC"를 읽고 "WXYZ"를 출력 문장으로 생성합니다. 모델은 문장 끝 토큰을 출력한 후 예측을 중지합니다. LSTM은 입력 문장을 반대로 읽습니다. 그렇게 하면 최적화 문제를 훨씬 쉽게 만드는 데이터에 많은 단기 종속성이 도입되기 때문입니다.

The main result of this work is the following. On the WMT’14 English to French translation task, we obtained a BLEU score of 34.81 by directly extracting translations from an ensemble of 5 deep LSTMs (with 384M parameters and 8,000 dimensional state each) using a simple left-to-right beamsearch decoder. This is by far the best result achieved by direct translation with large neural networks. For comparison, the BLEU score of an SMT baseline on this dataset is 33.30 [29]. The 34.81 BLEU score was achieved by an LSTM with a vocabulary of 80k words, so the score was penalized whenever the reference translation contained a word not covered by these 80k. This result shows that a relatively unoptimized small-vocabulary neural network architecture which has much room for improvement outperforms a phrase-based SMT system.

이 작업의 주요 결과는 다음과 같습니다. WMT'14 영어에서 프랑스어로 번역 작업에서 간단한 왼쪽에서 오른쪽으로 빔 검색 디코더를 사용하여 5개의 심층 LSTM(각각 384M 매개변수 및 8,000차원 상태 포함)의 앙상블에서 번역을 직접 추출하여 BLEU 점수 34.81을 얻었습니다. 이것은 대규모 신경망을 사용한 직접 번역으로 달성한 단연 최고의 결과입니다. 비교를 위해 이 데이터 세트에 대한 SMT 기준선의 BLEU 점수는 33.30입니다. 34.81 BLEU 점수는 80k 단어의 어휘를 가진 LSTM에 의해 달성되었으므로 참조 번역에 이러한 80k 단어에 포함되지 않는 단어가 포함될 때마다 점수가 감점되었습니다. 이 결과는 개선의 여지가 많은 상대적으로 최적화되지 않은 작은 어휘 신경망 아키텍처가 구문 기반 SMT 시스템보다 성능이 우수함을 보여줍니다.

Finally, we used the LSTM to rescore the publicly available 1000-best lists of the SMT baseline on the same task [29]. By doing so, we obtained a BLEU score of 36.5, which improves the baseline by 3.2 BLEU points and is close to the previous best published result on this task (which is 37.0 [9]).

마지막으로 LSTM을 사용하여 동일한 작업에 대해 공개적으로 사용 가능한 1000개의 SMT 기준선 목록을 다시 채점했습니다. 그렇게 함으로써 우리는 36.5의 BLEU 점수를 얻었는데, 이는 기준선을 3.2 BLEU 포인트 향상시키고 이 작업에 대해 이전에 발표된 최고 결과(37.0)에 가깝습니다.

Surprisingly, the LSTM did not suffer on very long sentences, despite the recent experience of other researchers with related architectures [26]. We were able to do well on long sentences because we reversed the order of words in the source sentence but not the target sentences in the training and test set. By doing so, we introduced many short term dependencies that made the optimization problem much simpler (see sec. 2 and 3.3). As a result, SGD could learn LSTMs that had no trouble with long sentences. The simple trick of reversing the words in the source sentence is one of the key technical contributions of this work

놀랍게도, LSTM은 관련 아키텍처를 가진 다른 연구자들의 최근 경험에도 불구하고 매우 긴 문장에서 어려움을 겪지 않았습니다. 우리는 훈련 및 테스트 세트에서 대상 문장이 아니라 소스 문장에서 단어의 순서를 반전했기 때문에 긴 문장을 잘 할 수 있었습니다. 그렇게 함으로써 우리는 최적화 문제를 훨씬 더 단순하게 만드는 많은 단기 종속성을 도입했습니다(섹션 2 및 3.3 참조). 결과적으로 SGD는 긴 문장에도 문제가 없는 LSTM을 학습할 수 있었습니다. 소스 문장의 단어를 뒤집는 간단한 트릭은 이 작업의 주요 기술 기여 중 하나입니다.

A useful property of the LSTM is that it learns to map an input sentence of variable length into a fixed-dimensional vector representation. Given that translations tend to be paraphrases of the source sentences, the translation objective encourages the LSTM to find sentence representations that capture their meaning, as sentences with similar meanings are close to each other while different 2 sentences meanings will be far. A qualitative evaluation supports this claim, showing that our model is aware of word order and is fairly invariant to the active and passive voice.

LSTM의 유용한 속성은 가변 길이의 입력 문장을 고정 차원 벡터 표현으로 매핑하는 방법을 학습한다는 것입니다. 번역이 원본 문장의 의역을 하는 경향이 있다는 점을 감안할 때 번역 목표는 LSTM이 의미를 포착하는 문장 표현을 찾도록 권장합니다. 비슷한 의미를 가진 문장은 서로 가깝지만 다른 2개의 문장 의미는 멀기 때문입니다. 정성적 평가는 우리 모델이 어순을 인식하고 능동태와 수동태에 상당히 불변함을 보여줌으로써 이 주장을 뒷받침합니다.

**2. The model**

The Recurrent Neural Network (RNN) [31, 28] is a natural generalization of feedforward neural networks to sequences. Given a sequence of inputs , a standard RNN computes a sequence of outputs by iterating the following equation:

RNN(Recurrent Neural Network) [31, 28]은 피드포워드 신경망을 시퀀스로 자연스럽게 일반화한 것입니다. 입력 시퀀스가 주어지면 표준 RNN은 다음 방정식을 반복하여 출력 시퀀스를 계산합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

The RNN can easily map sequences to sequences whenever the alignment between the inputs the outputs is known ahead of time. However, it is not clear how to apply an RNN to problems whose input and the output sequences have different lengths with complicated and non-monotonic relationships.

RNN은 입력 출력 간의 정렬이 미리 알려질 때마다 시퀀스를 시퀀스에 쉽게 매핑할 수 있습니다. 그러나 복잡하고 비단조적인 관계로 입력 시퀀스와 출력 시퀀스의 길이가 다른 문제에 RNN을 적용하는 방법은 명확하지 않습니다.

The simplest strategy for general sequence learning is to map the input sequence to a fixed-sized vector using one RNN, and then to map the vector to the target sequence with another RNN (this approach has also been taken by Cho et al. [5]). While it could work in principle since the RNN is provided with all the relevant information, it would be difficult to train the RNNs due to the resulting long term dependencies (figure 1) [14, 4, 16, 15]. However, the Long Short-Term Memory (LSTM)[16] is known to learn problems with long range temporal dependencies, so an LSTM may succeed in this setting.

일반적인 시퀀스 학습을 위한 가장 간단한 전략은 하나의 RNN을 사용하여 입력 시퀀스를 고정 크기 벡터에 매핑한 다음 벡터를 다른 RNN으로 대상 시퀀스에 매핑하는 것입니다(이 접근법은 Cho et al. ]). RNN에는 모든 관련 정보가 제공되므로 원칙적으로 작동할 수 있지만 결과적인 장기 종속성으로 인해 RNN을 훈련하기 어려울 것입니다(그림 1). 그러나 LSTM(Long Short-Term Memory)[16]은 장거리 시간 종속성 문제를 학습하는 것으로 알려져 있으므로 LSTM이 이 설정에서 성공할 수 있습니다.

The goal of the LSTM is to estimate the conditional probability where is an input sequence and is its corresponding output sequence whose length T’ may differ from T. The LSTM computes this conditional probability by first obtaining the fixeddimensional representation v of the input sequence given by the last hidden state of the LSTM, and then computing the probability of with a standard LSTM-LM formulation whose initial hidden state is set to the representation of :

LSTM의 목표는 조건부 확률 를 추정하는 것입니다. 여기서 는 입력 시퀀스이고 는 T와 길이가 다를 수 있는 해당 출력 시퀀스입니다. LSTM은 먼저 LSTM의 마지막 숨겨진 상태에 의해 주어진 입력 시퀀스 의 고정 차원 표현 v를 얻은 다음 초기 상태로 설정된 표준 LSTM-LM 공식으로의 확률을 계산하여 이러한 조건부 확률을 계산합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

In this equation, each distribution is represented with a softmax over all the words in the vocabulary. We use the LSTM formulation from Graves [10]. Note that we require that each sentence ends with a special end-of-sentence symbol “<EOS>”, which enables the model to define a distribution over sequences of all possible lengths. The overall scheme is outlined in figure 1, where the shown LSTM computes the representation of “A”, “B”, “C”, “<EOS>” and then uses this representation to compute the probability of “W”, “X”, “Y”, “Z”, “<EOS>”.

이 방정식에서 각 분포는 어휘의 모든 단어에 대한 softmax로 표현됩니다. 우리는 Graves[10]의 LSTM 공식을 사용합니다. 각 문장은 특별한 문장 끝 기호 "<EOS>"로 끝나야 합니다. 이 기호를 사용하면 모델이 가능한 모든 길이의 시퀀스에 대한 분포를 정의할 수 있습니다. 전체 체계는 그림 1에 요약되어 있습니다. 여기서 표시된 LSTM은 "A", "B", "C", "<EOS>"의 표현을 계산한 다음 이 표현을 사용하여 "W", "X"의 확률을 계산합니다. ", "Y", "Z", "<EOS>".

Our actual models differ from the above description in three important ways. First, we used two different LSTMs: one for the input sequence and another for the output sequence, because doing so increases the number model parameters at negligible computational cost and makes it natural to train the LSTM on multiple language pairs simultaneously [18]. Second, we found that deep LSTMs significantly outperformed shallow LSTMs, so we chose an LSTM with four layers. Third, we found it extremely valuable to reverse the order of the words of the input sentence. So for example, instead of mapping the sentence a, b, c to the sentence α, β, γ, the LSTM is asked to map c, b, a to α, β, γ, where α, β, γ is the translation of a, b, c. This way, a is in close proximity to α, b is fairly close to β, and so on, a fact that makes it easy for SGD to “establish communication” between the input and the output. We found this simple data transformation to greatly improve the performance of the LSTM.

우리의 실제 모델은 세 가지 중요한 면에서 위의 설명과 다릅니다. 첫째, 우리는 두 가지 다른 LSTM을 사용했습니다. 하나는 입력 시퀀스용이고 다른 하나는 출력 시퀀스용입니다. 그렇게 하면 계산 비용을 무시할 수 있지만 숫자 모델 매개변수가 증가하고 여러 언어 쌍에 대해 LSTM을 동시에 훈련하는 것이 자연스럽기 때문입니다[18]. 둘째, 깊은 LSTM이 얕은 LSTM보다 성능이 훨씬 뛰어나서 4개의 레이어가 있는 LSTM을 선택했습니다. 셋째, 입력 문장의 단어 순서를 반대로 하는 것이 매우 가치 있음을 발견했습니다. 따라서 예를 들어 문장 a, b, c를 문장 α, β, γ에 매핑하는 대신 LSTM에 c, b, a를 α, β, γ로 매핑하도록 요청합니다. 여기서 α, β, γ는 번역입니다. 의, b, c. 이런 식으로 a는 α에 가깝고 b는 β에 상당히 가깝습니다. 이런 식으로 SGD가 입력과 출력 사이에 "통신을 설정"하기 쉽게 만드는 사실입니다. 우리는 이 간단한 데이터 변환이 LSTM의 성능을 크게 향상시키는 것을 발견했습니다.

**3.Experiments**

We applied our method to the WMT’14 English to French MT task in two ways. We used it to directly translate the input sentence without using a reference SMT system and we it to rescore the n-best lists of an SMT baseline. We report the accuracy of these translation methods, present sample translations, and visualize the resulting sentence representation.

WMT'14 English to French MT task에 두 가지 방법으로 우리의 방법을 적용했습니다. 참조 SMT 시스템을 사용하지 않고 입력 문장을 직접 번역하는 데 사용했으며 SMT 기준선의 n-best 목록을 다시 채점했습니다. 우리는 이러한 번역 방법의 정확성을 보고하고 샘플 번역을 제시하며 결과 문장 표현을 시각화합니다.

**3.1 Dataset Details**

We used the WMT’14 English to French dataset. We trained our models on a subset of 12M sentences consisting of 348M French words and 304M English words, which is a clean “selected” subset from [29]. We chose this translation task and this specific training set subset because of the public availability of a tokenized training and test set together with 1000-best lists from the baseline SMT [29]. As typical neural language models rely on a vector representation for each word, we used a fixed vocabulary for both languages. We used 160,000 of the most frequent words for the source language and 80,000 of the most frequent words for the target language. Every out-of-vocabulary word was replaced with a special “UNK” token.WMT'14 English to French 데이터셋을 사용했습니다. 우리는 [29]에서 깨끗한 "선택된" 부분 집합인 3억 4800만 프랑스어 단어와 3억 400만 영어 단어로 구성된 1200만 문장의 부분 집합에 대해 모델을 훈련했습니다. 우리는 이 번역 작업과 이 특정 훈련 세트 하위 집합을 선택했습니다. 기본 SMT의 1000개 목록과 함께 토큰화된 훈련 및 테스트 세트의 공개 가용성 때문입니다. 일반적인 신경 언어 모델은 각 단어에 대한 벡터 표현에 의존하기 때문에 두 언어에 대해 고정 어휘를 사용했습니다. 가장 많이 나오는 단어 중 160,000개를 출발어에, 가장 많이 나오는 단어 80,000개를 도착어에 사용했습니다. 어휘에 없는 모든 단어는 특별한 "UNK" 토큰으로 대체되었습니다.

**3.2 Decoding and Rescoring**

The core of our experiments involved training a large deep LSTM on many sentence pairs. We trained it by maximizing the log probability of a correct translation T given the source sentence S, so the training objective is

우리 실험의 핵심은 많은 문장 쌍에 대해 큰 deep LSTM을 훈련하는 것이었습니다. 우리는 소스 문장 S가 주어졌을 때 정확한 번역 T의 로그 확률을 최대화하여 훈련시켰습니다. 그래서 훈련 목표는 다음과 같습니다.



where S is the training set. Once training is complete, we produce translations by finding the most likely translation according to the LSTM:여기서 S는 훈련 세트입니다. 교육이 완료되면 LSTM에 따라 가장 가능성이 높은 번역을 찾아 번역을 생성합니다.



We search for the most likely translation using a simple left-to-right beam search decoder which maintains a small number B of partial hypotheses, where a partial hypothesis is a prefix of some translation. At each timestep we extend each partial hypothesis in the beam with every possible word in the vocabulary. This greatly increases the number of the hypotheses so we discard all but the B most likely hypotheses according to the model’s log probability. As soon as the “<EOS>” symbol is appended to a hypothesis, it is removed from the beam and is added to the set of complete hypotheses. While this decoder is approximate, it is simple to implement. Interestingly, our system performs well even with a beam size of 1, and a beam of size 2 provides most of the benefits of beam search (Table 1).

우리는 부분 가설이 일부 번역의 접두사인 부분 가설의 소수 B를 유지하는 간단한 왼쪽에서 오른쪽으로의 빔 검색 디코더를 사용하여 가장 가능성이 높은 번역을 검색합니다. 각 단계에서 우리는 어휘에서 가능한 모든 단어로 빔의 각 부분 가설을 확장합니다. 이것은 가설의 수를 크게 증가시키므로 모델의 로그 확률에 따라 가장 가능성이 높은 B를 제외한 모든 가설을 폐기합니다. "<EOS>" 기호가 가설에 추가되는 즉시 빔에서 제거되고 완전한 가설 세트에 추가됩니다. 이 디코더는 대략적이지만 구현하기 쉽습니다. 흥미롭게도 우리 시스템은 빔 크기가 1인 경우에도 잘 수행되며 크기가 2인 빔은 대부분의 빔 검색 이점을 제공합니다(표 1).

We also used the LSTM to rescore the 1000-best lists produced by the baseline system [29]. To rescore an n-best list, we computed the log probability of every hypothesis with our LSTM and took an even average with their score and the LSTM’s score.

우리는 또한 LSTM을 사용하여 베이스라인 시스템에서 생성된 1000개의 최고 목록을 다시 채점했습니다[29]. n-best 목록을 다시 채점하기 위해 LSTM을 사용하여 모든 가설의 로그 확률을 계산하고 해당 점수와 LSTM 점수로 짝수 평균을 취했습니다.

**3.3 Reversing the Source Sentences**

While the LSTM is capable of solving problems with long term dependencies, we discovered that the LSTM learns much better when the source sentences are reversed (the target sentences are not reversed). By doing so, the LSTM’s test perplexity dropped from 5.8 to 4.7, and the test BLEU scores of its decoded translations increased from 25.9 to 30.6.LSTM은 장기 종속성 문제를 해결할 수 있지만 소스 문장이 반전될 때(목표 문장이 반전되지 않음) LSTM이 훨씬 더 잘 학습한다는 것을 발견했습니다. 이를 통해 LSTM의 테스트 복잡도는 5.8에서 4.7로 떨어졌고 디코딩된 번역의 테스트 BLEU 점수는 25.9에서 30.6으로 높아졌습니다.

While we do not have a complete explanation to this phenomenon, we believe that it is caused by the introduction of many short term dependencies to the dataset. Normally, when we concatenate a source sentence with a target sentence, each word in the source sentence is far from its corresponding word in the target sentence. As a result, the problem has a large “minimal time lag” [17]. By reversing the words in the source sentence, the average distance between corresponding words in the source and target language is unchanged. However, the first few words in the source language are now very close to the first few words in the target language, so the problem’s minimal time lag is greatly reduced. Thus, backpropagation has an easier time “establishing communication” between the source sentence and the target sentence, which in turn results in substantially improved overall performance.

이 현상에 대한 완전한 설명은 없지만 데이터 세트에 대한 많은 단기 종속성이 도입되었기 때문에 발생한다고 생각합니다. 일반적으로 원본 문장을 대상 문장과 연결할 때 원본 문장의 각 단어는 대상 문장의 해당 단어에서 멀리 떨어져 있습니다. 결과적으로 문제는 "최소 시차"가 큽니다[17]. 원본 문장의 단어를 반대로 하면 원본 언어와 대상 언어의 해당 단어 사이의 평균 거리가 변경되지 않습니다. 그러나 소스 언어의 처음 몇 단어는 이제 대상 언어의 처음 몇 단어에 매우 가깝기 때문에 문제의 최소 시간 지연이 크게 줄어듭니다. 따라서 역전파는 소스 문장과 대상 문장 간의 "통신 설정" 시간을 더 쉽게 가지며 결과적으로 전반적인 성능이 크게 향상됩니다.

Initially, we believed that reversing the input sentences would only lead to more confident predictions in the early parts of the target sentence and to less confident predictions in the later parts. However, LSTMs trained on reversed source sentences did much better on long sentences than LSTMs 4 trained on the raw source sentences (see sec. 3.7), which suggests that reversing the input sentences results in LSTMs with better memory utilization.처음에 우리는 입력 문장을 반대로 하면 목표 문장의 초기 부분에서 더 확신에 찬 예측을 하고 후반 부분에서 덜 확신할 수 있는 예측으로 이어질 것이라고 믿었습니다. 그러나 역 소스 문장에 대해 훈련된 LSTM은 원시 소스 문장에 대해 훈련된 LSTM 4보다 긴 문장에서 훨씬 더 나은 결과를 얻었으며(섹션 3.7 참조), 이는 입력 문장을 역전시키면 더 나은 메모리 사용률을 갖는 LSTM이 생성됨을 시사합니다.

**3.4 Training details**

We found that the LSTM models are fairly easy to train. We used deep LSTMs with 4 layers, with 1000 cells at each layer and 1000 dimensional word embeddings, with an input vocabulary of 160,000 and an output vocabulary of 80,000. Thus the deep LSTM uses 8000 real numbers to represent a sentence. We found deep LSTMs to significantly outperform shallow LSTMs, where each additional layer reduced perplexity by nearly 10%, possibly due to their much larger hidden state. We used a naive softmax over 80,000 words at each output. The resulting LSTM has 384M parameters of which 64M are pure recurrent connections (32M for the “encoder” LSTM and 32M for the “decoder” LSTM). The complete training details are given below:우리는 LSTM 모델이 훈련하기가 상당히 쉽다는 것을 발견했습니다. 우리는 4개의 레이어가 있는 심층 LSTM을 사용했으며, 각 레이어에는 1000개의 셀이 있고 1000차원 단어 임베딩이 있으며 입력 어휘는 160,000이고 출력 어휘는 80,000입니다. 따라서 deep LSTM은 8000개의 실수를 사용하여 문장을 나타냅니다. 깊은 LSTM이 얕은 LSTM을 훨씬 능가하는 것으로 나타났습니다. 여기서 각 추가 레이어는 훨씬 더 큰 숨겨진 상태로 인해 당혹도를 거의 10% 감소시켰습니다. 우리는 각 출력에서 ​​80,000 단어가 넘는 순진한 softmax를 사용했습니다. 결과 LSTM에는 384M 매개변수가 있으며 그 중 64M은 순수 순환 연결입니다("인코더" LSTM의 경우 32M, "디코더" LSTM의 경우 32M). 전체 교육 세부 정보는 다음과 같습니다.

• We initialized all of the LSTM’s parameters with the uniform distribution between -0.08 and 0.08• -0.08에서 0.08 사이의 균일한 분포로 LSTM의 모든 매개변수를 초기화했습니다.

• We used stochastic gradient descent without momentum, with a fixed learning rate of 0.7. After 5 epochs, we begun halving the learning rate every half epoch. We trained our models for a total of 7.5 epochs.• 고정 학습률이 0.7인 모멘텀 없는 확률적 경사 하강법을 사용했습니다. 5 epoch 이후, 우리는 epoch의 절반마다 학습률을 절반으로 줄이기 시작했습니다. 우리는 총 7.5 Epoch 동안 모델을 훈련했습니다.

• We used batches of 128 sequences for the gradient and divided it the size of the batch (namely, 128).

• 그라디언트에 대해 128개 시퀀스의 배치를 사용하고 배치 크기(즉, 128)로 나눴습니다.

• Although LSTMs tend to not suffer from the vanishing gradient problem, they can have exploding gradients. Thus we enforced a hard constraint on the norm of the gradient [10, 25] by scaling it when its norm exceeded a threshold. For each training batch, we compute , where g is the gradient divided by 128. If we set .

• LSTM은 그라디언트 소실 문제를 겪지 않는 경향이 있지만 그라디언트가 폭발할 수 있습니다. 따라서 우리는 기울기의 규범이 임계값을 초과할 때 크기를 조정하여 기울기[10, 25]의 규범에 대한 엄격한 제약을 적용했습니다. 각 훈련 배치에 대해 를 계산합니다. 여기서 g는 기울기를 128로 나눈 값입니다. s>5이면 로 설정합니다.

• Different sentences have different lengths. Most sentences are short (e.g., length 20-30) but some sentences are long (e.g., length > 100), so a minibatch of 128 randomly chosen training sentences will have many short sentences and few long sentences, and as a result, much of the computation in the minibatch is wasted. To address this problem, we made sure that all sentences in a minibatch are roughly of the same length, yielding a 2x speedup.• 문장마다 길이가 다릅니다. 대부분의 문장은 짧지만(예: 길이 20-30) 일부 문장은 길기 때문에(예: 길이 > 100) 무작위로 선택된 128개의 훈련 문장으로 구성된 미니배치는 많은 짧은 문장과 적은 긴 문장을 가지며 결과적으로 많은 미니 배치의 계산이 낭비됩니다. 이 문제를 해결하기 위해 우리는 미니배치의 모든 문장이 대략 같은 길이인지 확인하여 2배의 속도 향상을 얻었습니다.

**3.5 Parallelization**

A C++ implementation of deep LSTM with the configuration from the previous section on a single GPU processes a speed of approximately 1,700 words per second. This was too slow for our purposes, so we parallelized our model using an 8-GPU machine. Each layer of the LSTM was executed on a different GPU and communicated its activations to the next GPU / layer as soon as they were computed. Our models have 4 layers of LSTMs, each of which resides on a separate GPU. The remaining 4 GPUs were used to parallelize the softmax, so each GPU was responsible for multiplying by a 1000 × 20000 matrix. The resulting implementation achieved a speed of 6,300 (both English and French) words per second with a minibatch size of 128. Training took about a ten days with this implementation.단일 GPU에서 이전 섹션의 구성을 사용하여 deep LSTM을 C++로 구현하면 초당 약 1,700단어의 속도로 처리됩니다. 이것은 우리의 목적에 비해 너무 느렸으므로 8-GPU 머신을 사용하여 모델을 병렬화했습니다. LSTM의 각 레이어는 다른 GPU에서 실행되었으며 계산되는 즉시 활성화를 다음 GPU/레이어에 전달했습니다. 우리 모델에는 4개의 LSTM 레이어가 있으며 각 레이어는 별도의 GPU에 있습니다. 나머지 4개의 GPU는 softmax를 병렬화하는 데 사용되었으므로 각 GPU는 1000 × 20000 행렬을 곱하는 역할을 했습니다. 그 결과 구현은 128개의 미니배치 크기로 초당 6,300단어(영어 및 프랑스어 모두)의 속도를 달성했습니다. 이 구현으로 교육에 약 10일이 걸렸습니다.

**3.6 Experimental Results**

We used the cased BLEU score [24] to evaluate the quality of our translations. We computed our BLEU scores using multi-bleu.pl1 on the tokenized predictions and ground truth. This way of evaluating the BELU score is consistent with [5] and [2], and reproduces the 33.3 score of [29]. However, if we evaluate the best WMT’14 system [9] (whose predictions can be downloaded from statmt.org\matrix) in this manner, we get 37.0, which is greater than the 35.8 reported by statmt.org\matrix.

우리는 BLEU 점수[24]를 사용하여 번역 품질을 평가했습니다. 우리는 토큰화된 예측과 정답에 대해 multi-bleu.pl1을 사용하여 BLEU 점수를 계산했습니다. 이러한 BELU 점수 평가 방식은 [5], [2]와 일치하며, [29]의 33.3점을 재현한다. 그러나 이러한 방식으로 최고의 WMT'14 시스템[9](예측은 statmt.org\matrix에서 다운로드할 수 있음)을 평가하면 statmt.org\matrix에서 보고한 35.8보다 큰 37.0을 얻습니다.

The results are presented in tables 1 and 2. Our best results are obtained with an ensemble of LSTMs that differ in their random initializations and in the random order of minibatches. While the decoded translations of the LSTM ensemble do not outperform the best WMT’14 system, it is the first time that a pure neural translation system outperforms a phrase-based SMT baseline on a large scale MT task by a sizeable margin, despite its inability to handle out-of-vocabulary words. The LSTM is within 0.5 BLEU points of the best WMT’14 result if it is used to rescore the 1000-best list of the baseline system.

결과는 표 1과 2에 나와 있습니다. 무작위 초기화와 미니 배치의 무작위 순서가 다른 LSTM 앙상블에서 최상의 결과를 얻을 수 있습니다. LSTM 앙상블의 디코딩된 번역이 최고의 WMT'14 시스템보다 성능이 좋지는 않지만 순수 신경 번역 시스템이 무능력에도 불구하고 대규모 MT 작업에서 구문 기반 SMT 기준선을 상당한 차이로 능가하는 것은 처음입니다. 어휘가 아닌 단어를 처리합니다. LSTM은 기준 시스템의 1000-best 목록을 다시 채점하는 데 사용되는 경우 최상의 WMT'14 결과의 0.5 BLEU 포인트 내에 있습니다.

테이블이(가) 표시된 사진

자동 생성된 설명

Table 1: The performance of the LSTM on WMT’14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.표 1: WMT'14 영어에서 프랑스어로 테스트 세트(ntst14)에 대한 LSTM의 성능. 빔 크기가 2인 5개의 LSTM 앙상블은 빔 크기가 12인 단일 LSTM보다 저렴합니다.

테이블이(가) 표시된 사진

자동 생성된 설명

Table 2: Methods that use neural networks together with an SMT system on the WMT’14 English to French test set (ntst14).

표 2: WMT'14 영어에서 프랑스어로 테스트 세트(ntst14)에서 SMT 시스템과 함께 신경망을 사용하는 방법.

**3.7 Performance on long sentences**

We were surprised to discover that the LSTM did well on long sentences, which is shown quantitatively in figure 3. Table 3 presents several examples of long sentences and their translations.우리는 LSTM이 그림 3에 정량적으로 표시된 긴 문장에서 잘 작동한다는 사실에 놀랐습니다. 표 3은 긴 문장과 그 번역의 몇 가지 예를 보여줍니다.

**3.8 Model Analysis**

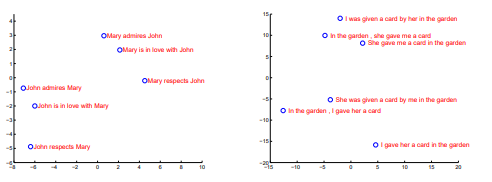


Figure 2: The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words model. Notice that both clusters have similar internal structure.

그림 2: 그림은 그림의 문구를 처리한 후 얻은 LSTM 숨겨진 상태의 2차원 PCA 투영을 보여줍니다. 구는 의미별로 클러스터링되어 있으며, 이러한 예에서는 주로 단어 순서의 기능이며, 이는 단어 모음 모델로 포착하기 어렵습니다. 두 클러스터의 내부 구조가 비슷합니다.

One of the attractive features of our model is its ability to turn a sequence of words into a vector of fixed dimensionality. Figure 2 visualizes some of the learned representations. The figure clearly shows that the representations are sensitive to the order of words, while being fairly insensitive to the replacement of an active voice with a passive voice. The two-dimensional projections are obtained using PCA.

우리 모델의 매력적인 기능 중 하나는 일련의 단어를 고정된 차원의 벡터로 바꾸는 기능입니다. 그림 2는 학습된 표현 중 일부를 시각화합니다. 그림은 표현이 단어의 순서에 민감한 반면 능동태를 수동태로 바꾸는 데는 상당히 둔감함을 분명히 보여줍니다. 2차원 투영은 PCA를 사용하여 얻습니다.

테이블이(가) 표시된 사진

자동 생성된 설명

Table 3: A few examples of long translations produced by the LSTM alongside the ground truth translations. The reader can verify that the translations are sensible using Google translate.

표 3: 실측 번역과 함께 LSTM에 의해 생성된 긴 번역의 몇 가지 예. 독자는 Google 번역을 사용하여 번역이 합리적인지 확인할 수 있습니다.

텍스트, 하늘, 지도이(가) 표시된 사진

자동 생성된 설명

Figure 3: The left plot shows the performance of our system as a function of sentence length, where the x-axis corresponds to the test sentences sorted by their length and is marked by the actual sequence lengths. There is no degradation on sentences with less than 35 words, there is only a minor degradation on the longest sentences. The right plot shows the LSTM’s performance on sentences with progressively more rare words, where the x-axis corresponds to the test sentences sorted by their “average word frequency rank”.

그림 3: 왼쪽 플롯은 문장 길이의 함수로 시스템의 성능을 보여줍니다. 여기서 x축은 길이별로 정렬된 테스트 문장에 해당하고 실제 시퀀스 길이로 표시됩니다. 35단어 미만의 문장에는 저하가 없으며 가장 긴 문장에는 약간의 저하만 있습니다. 오른쪽 그림은 점점 더 희귀한 단어가 포함된 문장에 대한 LSTM의 성능을 보여줍니다. 여기서 x축은 "평균 단어 빈도 순위"로 정렬된 테스트 문장에 해당합니다.

**4. Related work**

There is a large body of work on applications of neural networks to machine translation. So far, the simplest and most effective way of applying an RNN-Language Model (RNNLM) [23] or a 7 Feedforward Neural Network Language Model (NNLM) [3] to an MT task is by rescoring the nbest lists of a strong MT baseline [22], which reliably improves translation quality.

신경망을 기계 번역에 적용하는 작업이 많이 있습니다. 지금까지 RNN-Language Model(RNNLM)[23] 또는 7 Feedforward Neural Network Language Model(NNLM)[3]을 MT 작업에 적용하는 가장 간단하고 효과적인 방법은 강력한 MT의 nbest 목록을 재채점하는 것입니다. 기준선 [22], 번역 품질을 안정적으로 향상시킵니다.

More recently, researchers have begun to look into ways of including information about the source language into the NNLM. Examples of this work include Auli et al. [1], who combine an NNLM with a topic model of the input sentence, which improves rescoring performance. Devlin et al. [8] followed a similar approach, but they incorporated their NNLM into the decoder of an MT system and used the decoder’s alignment information to provide the NNLM with the most useful words in the input sentence. Their approach was highly successful and it achieved large improvements over their baseline.

보다 최근에 연구자들은 NNLM에 소스 언어에 대한 정보를 포함하는 방법을 조사하기 시작했습니다. 이 작업의 예는 Auli et al. [1], NNLM을 입력 문장의 주제 모델과 결합하여 재채점 성능을 향상시킵니다. Devlin et al. [8] 유사한 접근 방식을 따랐지만 NNLM을 MT 시스템의 디코더에 통합하고 디코더의 정렬 정보를 사용하여 입력 문장에서 가장 유용한 단어를 NNLM에 제공했습니다. 그들의 접근 방식은 매우 성공적이었고 기준선보다 큰 개선을 이루었습니다.

Our work is closely related to Kalchbrenner and Blunsom [18], who were the first to map the input sentence into a vector and then back to a sentence, although they map sentences to vectors using convolutional neural networks, which lose the ordering of the words. Similarly to this work, Cho et al. [5] used an LSTM-like RNN architecture to map sentences into vectors and back, although their primary focus was on integrating their neural network into an SMT system. Bahdanau et al. [2] also attempted direct translations with a neural network that used an attention mechanism to overcome the poor performance on long sentences experienced by Cho et al. [5] and achieved encouraging results. Likewise, Pouget-Abadie et al. [26] attempted to address the memory problem of Cho et al. [5] by translating pieces of the source sentence in way that produces smooth translations, which is similar to a phrase-based approach. We suspect that they could achieve similar improvements by simply training their networks on reversed source sentences.우리 작업은 Kalchbrenner와 Blunsom[18]과 밀접하게 관련되어 있습니다. 이들은 입력 문장을 벡터로 매핑한 다음 다시 문장으로 다시 매핑했지만 단어의 순서를 잃는 컨볼루션 신경망을 사용하여 문장을 벡터로 매핑합니다. . 이 작업과 유사하게 Cho et al.는 LSTM과 유사한 RNN 아키텍처를 사용하여 문장을 벡터로 매핑하고 그 반대로 매핑했지만 주요 초점은 신경망을 SMT 시스템에 통합하는 것이었습니다. Bahdanau et al. 또한 Cho et al이 경험한 긴 문장의 열악한 성능을 극복하기 위해 주의 메커니즘을 사용하는 신경망으로 직접 번역을 시도했습니다. 고무적인 결과를 얻었다. 마찬가지로, Pouget-Abadie et al. [26] Cho et al.의 메모리 문제를 해결하려고 시도했습니다. [5] 구문 기반 접근 방식과 유사한 부드러운 번역을 생성하는 방식으로 원본 문장의 조각을 번역합니다. 우리는 그들이 단순히 역 소스 문장에 대해 네트워크를 훈련함으로써 유사한 개선을 달성할 수 있다고 생각합니다.

End-to-end training is also the focus of Hermann et al. [12], whose model represents the inputs and outputs by feedforward networks, and map them to similar points in space. However, their approach cannot generate translations directly: to get a translation, they need to do a look up for closest vector in the pre-computed database of sentences, or to rescore a sentence.종단 간 교육도 Hermann et al.의 초점입니다. 모델이 피드포워드 네트워크에 의한 입력과 출력을 나타내고 공간의 유사한 지점에 매핑합니다. 그러나 그들의 접근 방식은 번역을 직접 생성할 수 없습니다. 번역을 얻으려면 미리 계산된 문장 데이터베이스에서 가장 가까운 벡터를 찾거나 문장을 다시 채점해야 합니다.

**5. Conclusion**

In this work, we showed that a large deep LSTM, that has a limited vocabulary and that makes almost no assumption about problem structure can outperform a standard SMT-based system whose vocabulary is unlimited on a large-scale MT task. The success of our simple LSTM-based approach non MT suggests that it should do well on many other sequence learning problems, provided they have enough training data.

이 작업에서 우리는 어휘가 제한적이고 문제 구조에 대한 가정을 거의 하지 않는 대규모 deep LSTM이 대규모 MT 작업에서 어휘가 무제한인 표준 SMT 기반 시스템보다 성능이 우수함을 보여주었습니다. MT가 아닌 간단한 LSTM 기반 접근 방식의 성공은 훈련 데이터가 충분하다면 다른 많은 시퀀스 학습 문제에서도 잘 수행되어야 함을 시사합니다.

We were surprised by the extent of the improvement obtained by reversing the words in the source sentences. We conclude that it is important to find a problem encoding that has the greatest number of short term dependencies, as they make the learning problem much simpler. In particular, while we were unable to train a standard RNN on the non-reversed translation problem (shown in fig. 1), we believe that a standard RNN should be easily trainable when the source sentences are reversed (although we did not verify it experimentally).우리는 원문에서 단어를 반대로 하여 얻은 개선의 정도에 놀랐습니다. 우리는 학습 문제를 훨씬 간단하게 만들기 때문에 단기 종속성이 가장 많은 문제 인코딩을 찾는 것이 중요하다고 결론지었습니다. 특히, non-reversed translation 문제에 대해 표준 RNN을 훈련할 수 없었지만(그림 1 참조), 표준 RNN은 원본 문장이 뒤집힐 때 쉽게 훈련할 수 있어야 한다고 믿습니다(비록 검증하지는 않았지만 실험적으로).

We were also surprised by the ability of the LSTM to correctly translate very long sentences. We were initially convinced that the LSTM would fail on long sentences due to its limited memory, and other researchers reported poor performance on long sentences with a model similar to ours [5, 2, 26]. And yet, LSTMs trained on the reversed dataset had little difficulty translating long sentences.

우리는 또한 매우 긴 문장을 정확하게 번역하는 LSTM의 능력에 놀랐습니다. 우리는 처음에 LSTM이 제한된 메모리로 인해 긴 문장에서 실패할 것이라고 확신했고 다른 연구자들은 우리와 유사한 모델을 사용하여 긴 문장에서 성능이 좋지 않다고 보고했습니다. 그러나 역 데이터셋으로 훈련된 LSTM은 긴 문장을 번역하는 데 어려움이 거의 없었습니다.

Most importantly, we demonstrated that a simple, straightforward and a relatively unoptimized approach can outperform an SMT system, so further work will likely lead to even greater translation accuracies. These results suggest that our approach will likely do well on other challenging sequence to sequence problems.

가장 중요한 것은 간단하고 간단하며 상대적으로 최적화되지 않은 접근 방식이 SMT 시스템보다 성능이 우수하므로 추가 작업으로 번역 정확도가 훨씬 더 높아질 수 있다는 점입니다. 이러한 결과는 우리의 접근 방식이 다른 도전적인 시퀀스 대 시퀀스 문제에서 잘 작동할 가능성이 있음을 시사합니다.

**6 Acknowledgements**

We thank Samy Bengio, Jeff Dean, Matthieu Devin, Geoffrey Hinton, Nal Kalchbrenner, Thang Luong, Wolfgang Macherey, Rajat Monga, Vincent Vanhoucke, Peng Xu, Wojciech Zaremba, and the Google Brain team for useful comments and discussions.

유용한 논평과 토론을 해주신 Samy Bengio, Jeff Dean, Matthieu Devin, Geoffrey Hinton, Nal Kalchbrenner, Thang Luong, Wolfgang Macherey, Rajat Monga, Vincent Vanhoucke, Peng Xu, Wojciech Zaremba 및 Google Brain 팀에 감사드립니다.