Convolutional Encoders for Neural Machine Translation

Andrew Lamb
Department of Computer Science
Stanford University
Stanford, CA 94305
lamb@cs.stanford.edu

Michael Xie
Department of Computer Science
Stanford University
Stanford, CA 94305
xie@cs.stanford.edu

Abstract

We propose a general Convolutional Neural Network (CNN) encoder model for machine translation that fits within in the framework of Encoder-Decoder models proposed by Cho, et. al. [1]. A CNN takes as input a sentence in the source language, performs multiple convolution and pooling operations, and uses a fully connected layer to produce a fixed-length encoding of the sentence as input to a Recurrent Neural Network decoder (using GRUs or LSTMs). The decoder, encoder, and word embeddings are jointly trained to maximize the conditional probability of the target sentence given the source sentence. Many variations on the basic model are possible and can improve the performance of the model.

1 Introduction

Neural Machine Translation is the application of deep neural network techniques to the machine translation task, first introduced by Kalchbrenner and Blunsom (2013), Sutskever et al. (2014) and Cho et al. (2014b). The proposed models follow a general paradigm, the *encoder-decoder* model, which consists of an encoder neural network that learns to transform variable length source sentences into an encoding, usually of fixed length, and a decoder neural network that learns a target language model and outputs a result conditioned on the encoding. Traditional machine translation systems include many separate components - for example, one component may be dedicated to aligning sentences - while neural models use an end-to-end model for the translation task.

The basic encoder-decoder models use Gated Recurrent Unit Neural Networks (GRUs) for both encoding and decoding. From the encoding standpoint, the GRU model can "forget" information as it progresses through an input sentence and does not encode phrase-level or structural information well. Additionally, an encoding produced by the GRU model is often disproportionately influenced by words appearing later in the sentence.

Convolutional Neural Networks (CNNs) have shown great success in computer vision tasks and sentence classification. In these models, images are passed through multiple convolution and pooling layers, and different convolution filters learn to detect different patterns in images. We apply this intuition to machine translation, and extend earlier work that used convolutions to model sentences [2]. In this context, a sentence is modeled as a matrix by concatenating its word embeddings as columns, and multiple convolution and pooling operations encode regional (n-gram) information and increasingly global information about the sentence. Intuitively, each filter in the convolution has a view of the entire source sequence, from which it picks features. We can thus form fixed length encodings that have features that consider global information during the encoding process. In this project, we explore a few new variations of the convolutional encoder model.

2 Background/Related Work

The basic RNN encoder-decoder model proposed by Cho, et. al. in [1] encodes a source sentence with a RNN model, and uses the last hidden state as the input for the RNN decoder network that outputs the translation in the target language. Approaches for remedying the limitations of RNN encoder-decoder models include RNN models that learn to jointly align and translate [3], 1-dimensional convolution models [4] that stop the convolution operations early to create a variable length encoding, and a bottlenecking encoder that creates an encoding with multiple vectors. The basic RNN encoder-decoder proposed by Cho, et. al., their Gated Recursive Convolutional encoder [5], their search based encoder-decoder model [3], and Kalchbrenner and Blunsom's convolutional encoders [4] provide the background work on neural network approaches to machine translation.These provide extensions that are also applicable to the model that we propose. Single dimensional convolution models involve a matrix multiplication with concatenated word vectors, which can be expressed in the 2-dimensional framework by performing a 1×1 convolution with the identity filter and framing the bi-gram convolutions as fully connected pooling layers. Another approach includes a gated recursive 1-dimensional Convolutional Neural Network, which shares weights between layers [5]. This can also trivially be expressed in the general convolutional encoder framework by the fully connected pooling layer with the constraint that the weights are the same throughout the layers.

3 Problem Statement

We translate from English to French, using the data from the WMT 2014 Translation Task. For training data we use Europarl v7, the Common Crawl corpus, the UN corpus, and the News Commentary corpus. For the development set, we use newstest-2013. The 30,000 most frequent words are used (about 97% of the data). BLEU scores are used to measure of translation quality. Our objective is to minimize the cross entropy loss between the generated target language sequence and the ground truth translation.

We compare our results to the basic encoder-decoder created by Cho, et. al. in [1].

4 Technical Approach and Models

We model a sentence as a $d \times k$ matrix, where word embeddings of the source language have dimension d and the sentence is of length k. We zero pad the ends of sentences to a fixed length ($\tilde{k}$ = 32) and discard longer sentences, although we can translate any longer sentences in chunks. Zero-padding is generally preserved in the convolutions and pooling stages, so that the variable input sentence length is also encoded. Thus, we have an input example to the encoder that is $d \times \tilde{k}$ where d and $\tilde{k}$ are both constants. We perform a convolution followed by a non-linearity on this input. Here, we allow the convolution to be a general 2-dimensional convolution.

We define the convolutional encoder as follows, with input $X \in \mathbb{R}^{d \times \tilde{k}}$:

$$h_0 = p_0(f_0(conv(W_0, X) + b_0))$$

$$h_i = p_i(f_i(conv(W_i, h_{i-1}) + b_i))$$

$$h_f = W_f h_m + b_f$$

for $i = 1, ..., m$, $f_i : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$ is an elementwise nonlinearity, $p_i : \mathbb{R}^{m \times n} \to \mathbb{R}^{u \times v}$ where $u \leq m$ and $v \leq n$ is a pooling operation, and $conv(W, X)$ is the convolution of input X with filters W. The output h_f is then the input to an RNN decoder. This is our initial proposed model. Adding gating functionality to the pooling operation will involve adding weights to this definition. All models were implemented as new layers using Theano in the GroundHog framework created by Cho, et. al [1].

5 Experiments and Results

5.1 Basic GRU Encoder-Decoder Model

To familiarize ourselves with the encoder-decoder architecture, and form a baseline BLEU score, we initially trained the model outlined by Cho, et. al. in [1]. This model uses a GRU encoder and a GRU decoder with one hidden layer, while jointly learning 30000 100-dimensional word embeddings. Three separate sets of 30000 embeddings were learned for the separate tasks of gating, reseting, and the actual semantic representation of the words. After training for approximately three days on Stanford's Rye cluster, the model received a BLEU score of 8.57 on the development set. We tracked training loss over time, and experimented with different optimization functions (Adadelta, Momentum) and learning rates. As seen in Figure 1, the model may need to be run with more parameters in order to fit the data. In later iterations, switching to Momentum did allow us to decrease the training loss further.

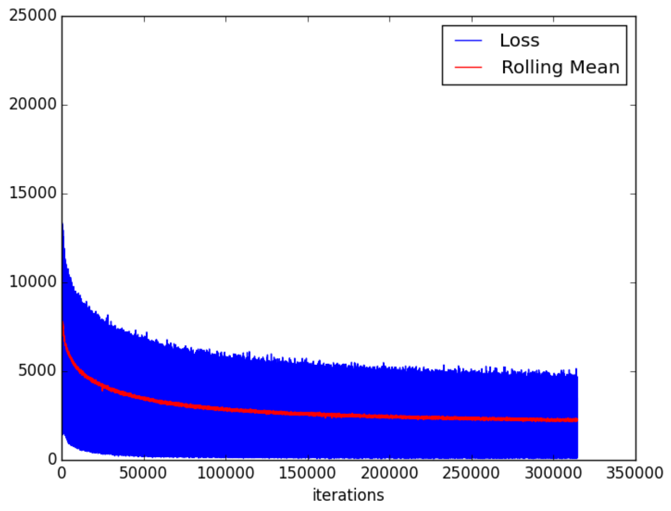

Figure 1: The GRU encoder-decoder model learns quickly initially, but training loss plateaus.

The model was able to output sentences that were usually grammatically correct and conveyed the semantic meaning of the source sentence:

```
Input:  Conference on Facilitating the Entry into Force [eol]
Target: Confrence en vue de faciliter l\&apos; entre en vigueur [eol]
Output: Confrence sur les modalits d\&apos; ouverture de la Force [eol]

Input:  The Agency believes that the current minimal level of service
        provision must be maintained , with future efforts focused on
        improvements in quality and efficiency . [eol]
Target: L\&apos; Office considre que le niveau minimal des services
        fournis devrait łtre maintenu et que les efforts futurs
        devraient tendre  amliorer la qualit et l\&apos;
        efficacit . [eol]
Output: L\&apos; avis que l\&apos; on estime qu\&apos;
        il est ncessaire de conserver le nombre minimum de services
        de personnel , mais nous permettre de travailler plus
        avant et efficaces . [eol]

Input:  When UNICEF decentralized the UNK to the regional
        offices , a new procedure was put into place as indicated
        in paragraph 6 above . [eol]
Target: Lorsque l\&apos; UNICEF a confi l\&apos;
        examen des programmes et du budget aux bureaux
        rgionaux , la nouvelle procdure dcrite plus haut
        au paragraphe 6 a t mise en place . [eol]
Output: Lorsque l\&apos; UNICEF a renforc les UNK  l\&apos;
        aide aux bureaux rgionaux , une nouvelle procdure s\&apos;
        tait prvue au paragraphe 6 ci-dessus . [eol]
```

5.2 Small Filters Model

We implemented a convolutional encoder layer with 3 convolution and average pooling layers with one fully connected layer and integrated the encoder with the implementation of an RNN/LSTM decoder by Cho, et. al. in the GroundHog repository [1]. In the 3×3 filters example, we use an average pooling operation over 2×2 region with stride 2 to decrease each dimension by a factor of 2. This model treats the sentence as an image and was tested mainly for its lower parameter size and to see if there are local interactions between dimensions of the word vectors. We tested a model with filters of dimensions 17×9. This model was not able to create encoding vectors that allow good translations and did not have enough representational power to fit the problem, as seen by the training cost.

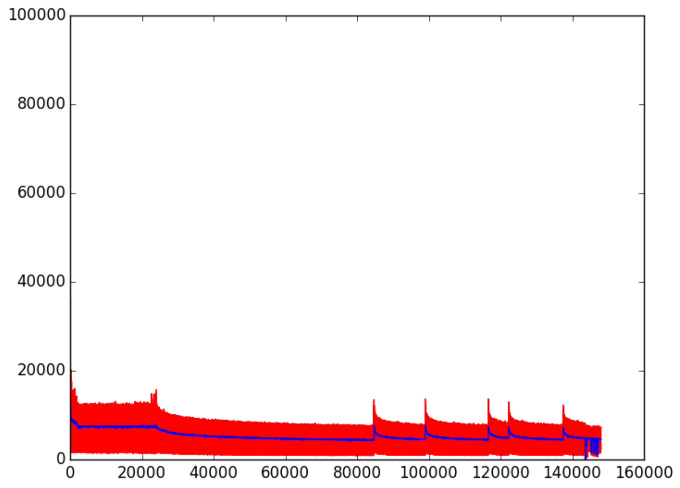

Figure 2: The small convolutional filters model had difficulty fitting the data.

5.3 One-Layer N-Gram Filters model

We also trained a one layer N-Gram convolutional filters model with 100 $d \times 2$ convolutional filters as an initial experiment, where $d = 1000$ is the word embedding size after up-sampling from the true 100-dimensional embedding. As seen in (3), the model converged to a training loss of 7000, which is much larger than that of the GRU encoder decoder and suggests that the representational power of the one layer model is lacking.

5.4 N-Gram Filters Model

Alternatively, using a filter of size $d \times n$ allows generalization of a 1 dimensional n-gram convolution over the entire space of the word embedding. Each filter outputs a feature vector that can then be concatenated together to form the input for the next convolutional layer. For example, using a filter of size $d \times 2$ with a column of zero padding on the input to create a $d \times (k + 1)$ input creates a

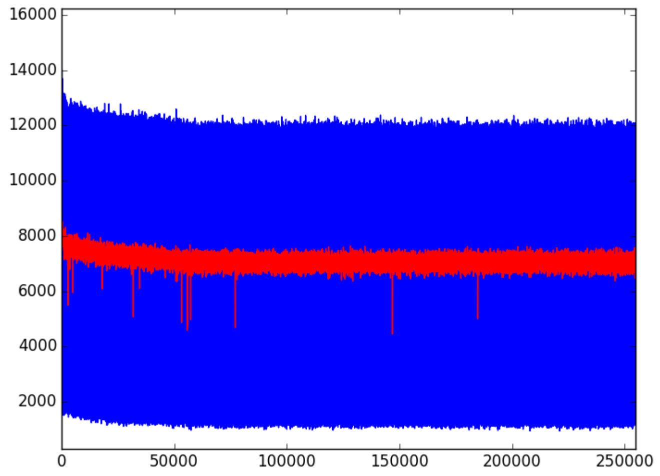

Figure 3: The one-layer N-Gram filters model had difficulty fitting the data.

$1 \times k$ feature vector. Each pooling layer increases the effective region size of the next convolution, composing the increasingly global information from regional information. Additionally, we can learn gating weights for the pooling operation, so that instead of a max operation, the model selects the weighting of the contributions of the elements being pooled, similar to that of Meng et al. [6].

We tested a 3 layer 2-gram filter model. The model used 100 dimensional randomly initialized word embeddings that were up-sampled to 1000 dimensional embeddings as the input. With a batch size of 64, the model accepted as input for every batch a tensor of size $64 \times 1 \times d \times k$ where $d = 1000$ and $k = 32$. We used three convolutional and max pooling layers, each with 500 $d \times 2$ filters. We add one column of zero padding to the end of the input before convolving. The output of each filter in the first layer is a 1×32 vector, and thus the output of the first convolution with 500 filters is a tensor of size $64 \times 500 \times 1 \times k$. We transpose the middle two dimensions to create an tensor of size $64 \times 1 \times 500 \times k$. We then apply a max pooling operation across the last dimension over a region of size 2 and a stride of 2 to create an input to the next convolutional layer of size $64 \times 1 \times 500 \times \frac{k}{2}$. We apply the same sequence of operations for 3 layers in total, followed by a fully connected layer that maps the tensor, reshaped to size $64 \times (\frac{500k}{8})$ into a 1000-dimensional encoding as input for each step of a GRU decoder. Each layer uses 500 filters, which is a hyperparameter. We use a ReLU nonlinearity in after each convolutional step.

Intuitively, each filter learns specific features of bigrams and pooled n-grams and activates throughout the length of the sentence during the convolution. After concatenating, the filters in the next convolutional layer have access to a 500 dimensional feature vector of each location in the sequence. A possible extension of this is to deterministically flip half of the vectors outputted by the filters so that the filters in the subsequent layer have access to information from the beginning and end of the sentence. Another possible extension is to reformulate pooling as gating so that the model can decide which features to keep or discard.

The N-Gram filters model with bigram filters was able to achieve a lower training cost than the small filters, and it is clear that having larger representational power helps the encoding. In around 300000 to 400000 iterations for 10 days, the bigram filters model converged to a training loss of around 4000, which is slightly above that of the basic GRU encoder-decoder at the same number of iterations. It achieves a very low BLEU score of 0.04, and almost all of the BLEU score is attributed to unigram matches. This suggests that the model did not learn the language model well, which could mean that the GRU decoder did not have enough iterations to fit the language model. From experience with training the GRU encoder-decoder, we found that there was a point near a cost of

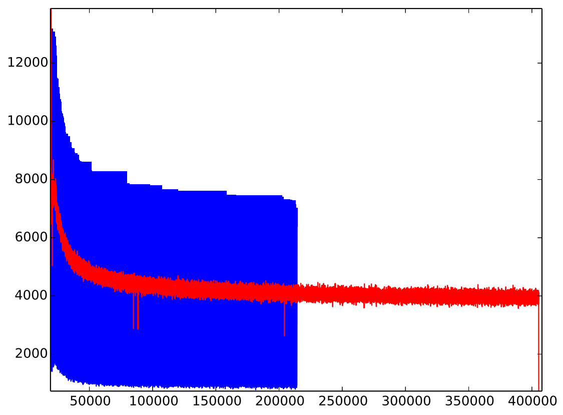

Figure 4: Training loss for N-Gram Filters Model for 400000 iterations.

3000 to 3500 in which the GRU encoder-decoder became significantly better semantically and in terms of BLEU score. The training of the N-Gram filters model takes significantly longer than the GRU encoder. Because the hyperparameters such as the number of filters and the n-gram size were somewhat arbitrarily set to be 500 throughout the model and a n-gram size of 2, it may be that with better choices of hyperparameters that the model will reach the point in which it improves dramatically in performance. Due to the large training time and limited computing resources, an extensive hyperparameter search was inhibitory. Comparing the number of parameters, the GRU encoder has 1 million parameters in its hidden state weight matrix and keeps 3 separate 100-dimensional encodings for each word for gating, reseting, and semantic representation. For 30000 words, this is about 9 million parameters in total. The N-Gram filters model has 3 convolutional layers with weight matrices that combine to a total of 1.5 million parameters, with a fully connected layer with 500000 parameters. With the 30000 word embeddings, this is a total of 5 million parameters. The CNN model that we trained has a smaller number of parameters than the GRU model, which may explain the lesser performance.

We can observe some interesting characteristics of the N-Gram filters model. Both the small filters and the N-Gram filters model exhibit a phenomenon in which it memorizes certain phrases that it outputs on a consistent basis throughout a batch. We can see this from some examples from the same batch below.

```
Input:  I realise that this is going to be difficult . <eol>
Target: Je reconnais que c&apos; est l une tche difficile . <eol>
Input:  I realise that this is going to be difficult . <eol>
Output: outre notamment l&apos; obligation de faire face aux
        politiques de l&apos; environnement de l&apos; humanit . <eol>

Input:  Note : Be sure to include the period ( . <eol>
Target: Remarque : UNK  inclure le point ( . <eol>
Input:  Note : Be sure to include the period ( . <eol>
Output: outre l&apos; importance du nombre de ces programmes
        de l&apos; ducation . <eol>

Input:  The old paragraphs 6 and 12 would be renumbered accordingly . <eol>
Target: Les actuels paragraphes 6  12 seront UNK en consquence . <eol>
Input:  The old paragraphs 6 and 12 would be renumbered accordingly . <eol>
Output: outre l&apos; assistance technique de l&apos; organe de
        l&apos; organe de gestion de l&apos; environnement . <eol>
```

This suggests that the input sentences, which are very different from each other and randomly selected, are being mapped to similar encoding vectors depending on the state of the convolutional encoder. The output also tends to repeat phrases in the same sentence, which is counterintuitive to the language model, as seen below.

```
Input:  On the return of the owners , however , the Government had had
        to build centres to accommodate the people whose homes had
        been destroyed . <eol>
Target: Mais  la suite du retour des propritaires des lieux , le
        Gouvernement a d faire construire des centres pour hberger
        les personnes dont le logement avait t dtruit . <eol>
Input:  On the return of the owners , however , the Government had had
        to build centres to accommodate the people whose homes had
        been destroyed . <eol>
Output: Pour conclure , je voudrais dire que les citoyens d&apos;
        Europe orientale et orientale du pays d&apos; Europe orientale
        ; <eol>
```

This could again suggest that not only was the convolutional encoder not yet outputting distinguising encodings, but the GRU decoder also did not train for enough iterations to fit the language model.

5.5 Pre-trained Word Vectors

In order to speed up training by decreasing the number of parameters in the model, we used pretrained word vectors from the Glove word embeddings by Pennington et. al. [7]. We tested the N-Gram filters model with pre-trained word vectors. Due to the high memory usage of the pretrained word embedding matrix, we used 50-dimensional word embeddings trained from the Wikipedia Gigaword dataset. However, due to the word embeddings and the training datasets used not matching, many input sentences had a large amount of unknown tokens, which debilitated training. Larger dimension word embeddings trained on similar datasets were not able to fit inside device memory for the machines we used.

6 Conclusion

Our encoding model extends single layer convolutional models to multiple layers [4] and applies models that have been used for general sentence modelling to the machine translation task [2], [8]. The promise of our convolutional encoders is that they generalize convolutional approaches and extend the one-layer convolututional encoder approach proposed by [2] to multi-layer deep convolutional encoders. Convolutional encoders are fast due to the convolution operator being very parallelizable in GPUs.

One of the drawbacks of the encoder-decoder translation model is the fixed-size encoding, which likely limits the amount of information a model can contain. The search based model in [3] can be added to our multi-layer convolutional model, which would likely improve performance. The fix proposed in [4], where the convolution is stopped short would clearly also integrate naturally into our model. While the models that were trained did not surpass the performance of a simple GRU encoder, it is possible that with more training, matching the number of parameters of the GRU model, and doing a more extensive hyperparameter search over the structure of the CNN, including creating a multi-path CNN with multiple sizes of N-Gram filters acting on the input as in [2], could allow the model to be able to fit the problem.

References

[1] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[2] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[4] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *EMNLP*, pages 1700–1709, 2013.

[5] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[6] Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. Encoding source language with convolutional neural network for machine translation. *CoRR*, abs/1503.01838, 2015.

[7] Jeffrey Pennington, Richard Socher, and Christopher Manning and. Glove: Global vectors for word representation. *Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2014.

[8] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188, 2014.