Deep Architectures for Neural Machine Translation

Citation for published version:

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the Second Conference on Machine Translation, Volume 1: Research Papers

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Deep Architectures for Neural Machine Translation

Antonio Valerio Miceli Barone† Jindřich Helcl⋆ Rico Sennrich† Barry Haddow† Alexandra Birch†
†School of Informatics, University of Edinburgh
⋆Faculty of Mathematics and Physics, Charles University
amiceli, bhaddow}@inf.ed.ac.uk
rico.sennrich, a.birch}@ed.ac.uk
helcl@ufal.mff.cuni.cz

Abstract

It has been shown that increasing model depth improves the quality of neural machine translation. However, different architectural variants to increase model depth have been proposed, and so far, there has been no thorough comparative study.

In this work, we describe and evaluate several existing approaches to introduce depth in neural machine translation. Additionally, we explore novel architectural variants, including deep transition RNNs, and we vary how attention is used in the deep decoder. We introduce a novel “BiDeep” RNN architecture that combines deep transition RNNs and stacked RNNs.

Our evaluation is carried out on the English to German WMT news translation dataset, using a single-GPU machine for both training and inference. We find that several of our proposed architectures improve upon existing approaches in terms of speed and translation quality. We obtain best improvements with a BiDeep RNN of combined depth 8, obtaining an average improvement of 1.5 BLEU over a strong shallow baseline.

We release our code for ease of adoption.

1 Introduction

Neural machine translation (NMT) is a well-established approach that yields the best results on most language pairs (Bojar et al., 2016; Cettolo et al., 2016). Most systems are based on the sequence-to-sequence model with attention (Bahdanau et al., 2015) which employs single-layer recurrent neural networks both in the encoder and in the decoder.

Unlike feed-forward networks where depth is straightforwardly defined as the number of non-input layers, recurrent neural network architectures with multiple layers allow different connection schemes (Pascanu et al., 2014) that give rise to different, orthogonal, definitions of depth (Zhang et al., 2016) which can affect the model performance depending on a given task. This is further complicated in sequence-to-sequence models as they contain multiple sub-networks, recurrent or feed-forward, each of which can be deep in different ways, giving rise to a large number of possible configurations.

In this work we focus on stacked and deep transition recurrent architectures as defined by Pascanu et al. (2014). Different types of stacked architectures have been successfully used for NMT (Zhou et al., 2016; Wu et al., 2016). However, there is a lack of empirical comparisons of different deep architectures. Deep transition architectures have been successfully used for language modeling (Zilly et al., 2016), but not for NMT so far. We evaluate these architectures, both alone and in combination, varying the connection scheme between the different components and their depth over the different dimensions, measuring the performance of the different configurations on the WMT news translation task.¹

Related work includes that of Britz et al. (2017), who have performed an exploration of NMT architectures in parallel to our work. Their experiments, which are largely orthogonal to ours, focus on embedding size, RNN cell type (GRU vs. LSTM), network depth (defined according to the architecture of Wu et al. (2016)), attention mechanism and beam size. Gehring et al. (2017) recently proposed a NMT architecture based on convolutions over fixed-sized windows.

¹http://www.statmt.org/wmt17/translation-task.html
rather than RNNs, and they reported results for
different model depths and attention mechanism
configurations. A similar feedforward architec-
ture which uses multiple pervasive attention mech-
anism rather than convolutions was proposed by
Vaswani et al. (2017), who also report results for
different model depths.

2 NMT Architectures

All the architectures that we consider in this work
are GRU (Cho et al., 2014a) sequence-to-sequence
transducers (Sutskever et al., 2014; Cho et al.,
2014b) with attention (Bahdanau et al., 2015). In
this section we describe the baseline system and
the variants that we evaluated.

2.1 Baseline Architecture

As our baseline, we use the NMT architecture im-
plemented in Nematus, which is described in more
depth by Sennrich et al. (2017b). We augment it
with layer normalization (Ba et al., 2016), which
we have found to both improve translation quality
and make training considerably faster.

For our discussion, it is relevant that the base-
line architecture already exhibits two types of
depth:

- recurrence transition depth in the decoder
  RNN which consists of two GRU transitions
  per output word with an attention mechanism
  in between, as described in Firat and Cho
  (2016).

- feed-forward depth in the attention network
  that computes the alignment scores and in the
  output network that predicts the target words.
  Both these networks are multi-layer percep-
trons with one tanh hidden layer.

2.2 Deep Transition Architectures

In a deep transition RNN (DT-RNN), at each time
step the next state is computed by the sequen-
tial application of multiple transition layers, effec-
tively using a feed-forward network embedded in-
side the recurrent cell. In our experiments, these
layers are GRU transition blocks with independ-
dently trainable parameters, connected such that
the "state" output of one of them is used as the
"state" input of the next one. Note that each of
these GRU transition is not individually recurrent,
recurrence only occurs at the level of the whole
multi-layer cell, as the "state" output of the last

2.2.1 Deep Transition Encoder

As in a baseline shallow Nematus system, the en-
coder is a bidirectional recurrent neural network.
Let $L_s$ be the encoder recurrence depth, then for
the $i$-th source word in the forward direction the
forward source word state $\hat{h}_{i,L_s}$ is com-
puted as:

$$\hat{h}_{i,1} = \text{GRU}_1 \left( x_i, \hat{h}_{i-1,L_s} \right)$$
$$\hat{h}_{i,k} = \text{GRU}_k \left( 0, \hat{h}_{i,k-1} \right) \text{ for } 1 < k \leq L_s$$

where the input to the first GRU transition is the
word embedding $x_i$, while the other GRU transi-
tions have no external inputs. Recurrence occurs
as the previous word state $\hat{h}_{i-1,L_s}$ enters the com-
putation in the first GRU transition for the current
word.

The reverse source word states are computed simi-
larly and concatenated to the forward ones to
form the bidirectional source word states $C \equiv \{ \hat{h}_{i,L_s}, \hat{h}_{i,L_s} \}$.

2.2.2 Deep Transition Decoder

The deep transition decoder is obtained by exten-
sing the baseline decoder in a similar way. Recall
that the baseline decoder of Nematus already has
a transition depth of two, with the first GRU tran-
sition receiving as input the embedding of the pre-
vious target word and the second GRU transition
receiving as input a context vector computed by
the attention mechanism. We extend this decoder

![Deep transition decoder](image_url)
architecture to an arbitrary transition depth $L_t$ as follows:

$$s_{j,1} = \text{GRU}_1 (y_{j-1}, s_{j-1,1})$$
$$s_{j,2} = \text{GRU}_2 (\text{ATT}(C, s_{j,1}), s_{j,1})$$
$$s_{j,k} = \text{GRU}_k (0, s_{j,k-1}) \text{ for } 2 < k \leq L_t$$

where $y_{j-1}$ is the embedding of the previous target word and \(\text{ATT}(C, s_{i,1})\) is the context vector computed by the attention mechanism. GRU transitions other than the first two do not have external inputs. The target word state vector $s_j \equiv s_j,L_t$ is then used by the feed-forward output network to predict the current target word. A diagram of this architecture is shown in Figure 1.

The output network can be also made deeper by adding more feed-forward hidden layers.

### 2.3 Stacked architectures

A stacked RNN is obtained by having multiple RNNs (GRUs in our experiments) run for the same number of time steps, connected such that at each step the bottom RNN takes "external" inputs from the outside, while each of the higher RNN takes as its "external" input the "state" output of the one below it. Residual connections between states at different depth (He et al., 2016) are also used to improve information flow. Note that unlike deep transition GRUs, here each GRU transition block constitutes a cell that is individually recurrent, as it has its own state that is carried over between time steps.

#### 2.3.1 Stacked Encoder

In this work we consider two types of bidirectional stacked encoders: an architecture similar to Zhou et al. (2016) which we denote here as alternating encoder (Figure 2), and one similar to Wu et al. (2016) which we denote as biunidirectional encoder (Figure 3).

Our contribution is the empirical comparison of these architectures, both in isolation and in combination with the deep transition architecture.

We do not consider stacked unidirectional encoders (Sutskever et al., 2014) as bidirectional encoders have been shown to outperform them (e.g. Britz et al. (2017)).

**Alternating Stacked Encoder** The forward part of the encoder consists of a stack of GRU recurrent neural networks, the first one processing words in the forward direction, the second one in the backward direction, and so on, in alternating directions. For an encoder stack depth $D_s$, and a source sentence length $N$, the forward source word state $\overrightarrow{w}_i \equiv \overrightarrow{w}_{i,D_s}$ is computed as:

$$\overrightarrow{w}_{i,1} = \overrightarrow{h}_{i,1} = \text{GRU}_1 (x_i, \overrightarrow{h}_{i-1,1})$$
$$\overrightarrow{h}_{i,2k} = \text{GRU}_{2k} (\overrightarrow{w}_{i,2k-1}, \overrightarrow{h}_{i,2k})$$

for $1 < 2k \leq D_s$

$$\overrightarrow{h}_{i,2k+1} = \text{GRU}_{2k+1} (\overrightarrow{w}_{i,2k}, \overrightarrow{h}_{i,2k+1})$$

for $1 < 2k + 1 \leq D_s$

$$\overrightarrow{w}_{i,j} = \overrightarrow{h}_{i,j} + \overrightarrow{w}_{i,j-1}$$

for $1 < j \leq D_s$

where we assume that $\overrightarrow{h}_{0,k}$ and $\overrightarrow{h}_{N+1,k}$ are zero vectors. Note the residual connections: at each level above the first one, the word state of the previous level $\overrightarrow{w}_{i,j-1}$ is added to the recurrent state of the GRU cell $\overrightarrow{h}_{i,j}$ to compute the word state for the current level $\overrightarrow{w}_{i,j}$.

The backward part of the encoder has the same structure, except that the first level of the stack processes the words in the backward direction and the subsequent levels alternate directions.

The forward and backward word states are then concatenated to form bidirectional word states $C \equiv \{\overrightarrow{w}_{i,D_s}, \overleftarrow{w}_{i,D_s}\}$. A diagram of this architecture is shown in Figure 2.

**Biunidirectional Stacked Encoder** In this encoder the forward and backward parts are shallow, as in the baseline architecture. Their word states are concatenated to form shallow bidirectional word states $w_i \equiv [\overrightarrow{w}_{i,1} \overleftarrow{w}_{i,1}]$ that are then used as inputs for subsequent stacked GRUs which operate only in the forward sentence direction, hence the name "biunidirectional". Since residual connections are also present, the higher depth
GRUs have a state size twice that of the base ones. This architecture has shorter maximum information propagation paths than the alternating encoder, suggesting that it may be less expressive, but it has the advantage of enabling implementations with higher model parallelism. A diagram of this architecture is shown in Figure 3.

In principle, alternating and biunidirectional stacked encoders can be combined by having $D_{sa}$ alternating layers followed by $D_{sb}$ unidirectional layers.

### 2.3.2 Stacked Decoder

A stacked decoder can be obtained by stacking RNNs which operate in the forward sentence direction. A diagram of this architecture is shown in Figure 4.

Note that the base RNN is always a conditional GRU (cGRU, Firat and Cho, 2016) which has transition depth at least two due to the way that the context vectors generated by the attention mechanism are used in Nematus. This opens up the possibility of several architectural variants which we evaluated in this work:

**Stacked GRU** The higher RNNs are simple GRUs which receive as input the state from the previous level of the stack, with residual connections between the levels.

\[
\begin{align*}
    s_{j,1,1} &= \text{GRU}_{1,1} (y_{j-1}, s_{j-1,1,2}) \\
    c_{j,1} &= \text{ATT}(C, s_{j,1,1}) \\
    s_{j,1,2} &= \text{GRU}_{1,2} (c_{j,1}, s_{j,1,1}) \\
    r_{j,1} &= s_{j,1,2} \\
    s_{j,k,1} &= \text{GRU}_{k} (r_{j,k-1,1}, s_{j-1,k,1}) \\
    r_{j,k} &= s_{j,k,1} + r_{j,k-1} \\
    &\text{for } 1 < k \leq D_t
\end{align*}
\]

Note that the higher levels have transition depth one, unlike the base level which has two.

**Stacked rGRU** The higher RNNs are GRUs whose "external" input is the concatenation of the state below and the context vector from the base RNN. Formally, the states $s_{j,k,1}$ of the higher RNNs are computed as:

\[
\begin{align*}
    s_{j,k,1} &= \text{GRU}_{k} ([r_{j,k-1,1}, c_{j,1}], s_{j-1,k,1}) \\
    &\text{for } 1 < k \leq D_t
\end{align*}
\]

This is similar to the deep decoder by Wu et al. (2016).

**Stacked cGRU** The higher RNNs are conditional GRUs, each with an independent attention mechanism. Each level has two GRU transitions per step $j$, with a new context vector $c_{j,k}$ computed in between:

\[
\begin{align*}
    s_{j,k,1} &= \text{GRU}_{k,1} (r_{j,k-1,1}, s_{j-1,k,1}) \\
    c_{j,k} &= \text{ATT}(C, s_{j,k,1}) \\
    s_{j,k,2} &= \text{GRU}_{k,2} (c_{j,k}, s_{j,1,1}) \\
    &\text{for } 1 < k \leq D_t
\end{align*}
\]

Note that unlike the stacked GRU and rGRU, the higher levels have transition depth two.

**Stacked crGRU** The higher RNNs are conditional GRUs but they reuse the context vectors from the base RNN. Like the cGRU there are two GRU transition per step, but they reuse the context vector $c_{j,1}$ computed at the first level of the stack:

\[
\begin{align*}
    s_{j,k,1} &= \text{GRU}_{k,1} (r_{j,k-1,1}, s_{j-1,k,1}) \\
    s_{j,k,2} &= \text{GRU}_{k,2} (c_{j,1}, s_{j,1,1}) \\
    &\text{for } 1 < k \leq D_t
\end{align*}
\]
2.4 BiDeep architectures

We introduce the BiDeep RNN, a novel architecture obtained by combining deep transitions with stacking.

A BiDeep encoder is obtained by replacing the $D_s$ individually recurrent GRU cells of a stacked encoder with multi-layer deep transition cells each composed by $L_s$ GRU transition blocks.

For instance, the BiDeep alternating encoder is defined as follows:

$$
t = DTGRU_1 (x_1, \hat{h}_{i-1,1})$$

$$\hat{h}_{i,2k} = DTGRU_{2k} \left( \hat{w}_{i,2k-1}, \hat{h}_{i+1,2k} \right) \quad \text{for } 1 < 2k \leq D_s$$

$$\hat{h}_{i,2k+1} = DTGRU_{2k+1} \left( \hat{w}_{i,2k}, \hat{h}_{i-1,2k+1} \right) \quad \text{for } 1 < 2k + 1 \leq D_s$$

$$\hat{w}_{i,j} = \hat{h}_{i,j} + \hat{w}_{i,j-1} \quad \text{for } 1 < j \leq D_s$$

where each multi-layer cell $DTGRU_k$ is defined as:

$$v_{k,1} = GRU_{k,1} (in_k, state_k)$$

$$v_{k,t} = GRU_{k,t} (0, v_{k,t-1}) \quad \text{for } 1 < k \leq L_s$$

$$DTGRU_k (in_k, state_k) = v_{k,L_s}$$

It is also possible to have different transition depths at each stacking level.

BiDeep decoders are similarly defined, replacing the recurrent cells (GRU, rGRU, cGRU or crGRU) with deep transition multi-layer cells.

3 Experiments

All experiments were performed with Nematus (Sennrich et al., 2017b), following Sennrich et al. (2017a) in their choice of preprocessing and hyperparameters. For experiments with deep models, we increase the depth by a factor of 4 compared to the baseline for most experiments; in preliminary experiments, we observed diminishing returns for deeper models.

We trained on the parallel English–German training data of WMT-2017 news translation task, using newstest2013 as validation set. We used early-stopping on the validation cross-entropy and selected the best model based on validation BLEU.

We report cross-entropy (CE) on newstest2013, training speed (on a single Titan X (Pascal) GPU), and the number of parameters. For translation quality, we report case-sensitive, detokenized BLEU, measured with mteval-v13a.pl, on newstest2014, newstest2015, and newstest2016.

We release the code under an open source license, including it in the official Nematus repository.

3.1 Layer Normalization

Our first experiment is concerned with layer normalization. We are interested to see how essential layer normalization is for our deep architectures, and compare the effect of layer normalization on a baseline system, and a system with an alternating encoder with stacked depth 4. Results are shown in Table 1. We find that layer normalization is similarly effective for both the shallow baseline model and the deep encoder, yielding an average improvement of 0.8–1 BLEU, and reducing training time substantially. Therefore we use it for all the subsequent experiments.

3.2 Deep Encoders

In Table 2 we report experimental results for different architectures of deep encoders, while the decoder is kept shallow.

We find that all the deep encoders perform substantially better than baseline (+0.5–+1.2 BLEU), with no consistent quality differences between each other. In terms of number of parameters and training speed, the deep transition encoder performs best, followed by the alternating stacked encoder and finally the bidirectional encoder (note that we trained on a single GPU, the bidirectional encoder may be comparatively faster on multiple GPUs due to its higher model parallelism).

3.3 Deep Decoders

Table 3 shows results for different decoder architectures, while the encoder is shallow. We find that the deep decoders all improve the cross-entropy, but the BLEU results are more varied: deep output$^4$ decreases BLEU scores (but note that the baseline

$^4$https://github.com/EdinburghNLP/nematus

$^6$https://github.com/Avmb/deep-nmt-architectures

$^d$deep feed-forward output with shallow RNNs in both the encoder and decoder
Table 1: Layer normalization results. English→German WMT17 data.

<table>
<thead>
<tr>
<th>encoder</th>
<th>CE 2014</th>
<th>BLEU 2015</th>
<th>BLEU 2016</th>
<th>parameters (M)</th>
<th>training speed (words/s)</th>
<th>early stop (10^4 minibatches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>49.98</td>
<td>21.2</td>
<td>23.8</td>
<td>28.4</td>
<td>98.0</td>
<td>3350</td>
</tr>
<tr>
<td>+layer normalization</td>
<td>47.53</td>
<td>21.9</td>
<td>24.7</td>
<td>29.3</td>
<td>98.1</td>
<td>2900</td>
</tr>
<tr>
<td>alternating (depth 4)</td>
<td>49.25</td>
<td>21.8</td>
<td>24.6</td>
<td>28.9</td>
<td>135.8</td>
<td>2150</td>
</tr>
<tr>
<td>+layer normalization</td>
<td>46.29</td>
<td>22.6</td>
<td>25.2</td>
<td>30.5</td>
<td>135.9</td>
<td>1600</td>
</tr>
</tbody>
</table>

Table 2: Deep encoder results. English→German WMT17 data. Parameters and speed are highlighted for the deep recurrent models.

<table>
<thead>
<tr>
<th>encoder</th>
<th>depth s. bidir.</th>
<th>depth s. forw.</th>
<th>depth trans.</th>
<th>CE 2014</th>
<th>BLEU 2015</th>
<th>BLEU 2016</th>
<th>parameters (M)</th>
<th>training speed (words/s)</th>
<th>early stop (10^4 minibatches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>shallow</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>47.53</td>
<td>22.6</td>
<td>25.2</td>
<td>30.5</td>
<td>173.7</td>
<td>1500</td>
</tr>
<tr>
<td>alternating</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>46.29</td>
<td>22.4</td>
<td>25.4</td>
<td>30.0</td>
<td>135.9</td>
<td>1500</td>
</tr>
<tr>
<td>biunidirectional</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>46.79</td>
<td>22.9</td>
<td>25.4</td>
<td>30.2</td>
<td>117.0</td>
<td>1900</td>
</tr>
<tr>
<td>deep transition</td>
<td>1</td>
<td>-</td>
<td>4</td>
<td>46.54</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.4 Deep Encoders and Decoders

Table 4 shows results for models where both the encoder and the decoder are deep, in addition to the results of the best deep encoder (the deep transition encoder) + shallow decoder reported here for ease of comparison.

Compared to deep transition encoder alone, we generally see improvements in cross-entropy, but not in BLEU. We evaluate architectures similar to Zhou et al. (2016) (alternating encoder + stacked GRU decoder) and (Wu et al., 2016) (biunidirectional encoder + stacked rGRU decoder), though they are not straight replications since we used GRU cells rather than LSTMs and the implementation details are different. We find that the former architecture performs better in terms of BLEU scores, model size and training speed.

The other variants of alternating encoder + stacked or deep transition decoder perform similarly to alternating encoder + stacked rGRU decoder, but do not improve BLEU scores over the best deep encoder with shallow decoder. Applying the BiDeep architecture while keeping the total depth the same yields small improvements over the best deep encoder (+0.2 BLEU on average), while the improvement in cross-entropy is stronger. We conjecture that deep decoders may be better at handling subtle target-side linguistic phenomena that are not well captured by the 4-gram precision-based BLEU evaluation.

Finally, we evaluate a subset of architectures with a combined depth that is 8 times that of the baseline. Among the large models, the BiDeep model yields substantial improvements (average +0.6 BLEU over the best deep encoder, +1.5 BLEU over the shallow baseline), in addition to cross-entropy improvements. The stacked-only model, on the other hand, performs similarly to the smaller models, despite having even more parameters than the BiDeep model. This shows that it is useful to combine deep transitions with stacking, as they provide two orthogonal kinds of depth that are both beneficial for neural machine translation.

3.5 Error Analysis

One theoretical difference between a stacked RNN and a deep transition RNN is that the distance in the computation graph between timesteps is increased for deep transition RNNs. While this allows for arguably more expressive computations to be represented, in principle it could reduce the ability to remember information over long dis-
Table 3: Deep decoder results. English→German WMT17 data. Parameters and speed are highlighted for the deep recurrent models.

<table>
<thead>
<tr>
<th>decoder</th>
<th>high RNN stacked</th>
<th>decoder RNN depth</th>
<th>output</th>
<th>CE 2014</th>
<th>B 2015</th>
<th>BLEU 2016</th>
<th>params. (M)</th>
<th>training speed (words/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>shallow</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>47.53</td>
<td>21.9</td>
<td>24.7</td>
<td>29.3</td>
<td>98.1</td>
</tr>
<tr>
<td>stacked</td>
<td>GRU</td>
<td>4</td>
<td>1</td>
<td>46.73</td>
<td>21.8</td>
<td>24.6</td>
<td>29.5</td>
<td>117.0</td>
</tr>
<tr>
<td>stacked</td>
<td>rGRU</td>
<td>4</td>
<td>1</td>
<td>46.72</td>
<td>22.1</td>
<td>25.0</td>
<td>29.4</td>
<td>135.9</td>
</tr>
<tr>
<td>stacked</td>
<td>cGRU</td>
<td>4</td>
<td>1</td>
<td>44.76</td>
<td>22.8</td>
<td>25.5</td>
<td>29.6</td>
<td>145.4</td>
</tr>
<tr>
<td>stacked</td>
<td>crGRU</td>
<td>4</td>
<td>1</td>
<td>45.88</td>
<td>22.5</td>
<td>24.7</td>
<td>29.7</td>
<td>164.3</td>
</tr>
<tr>
<td>deep transition</td>
<td>-</td>
<td>1</td>
<td>3</td>
<td>45.98</td>
<td>22.4</td>
<td>24.9</td>
<td>30.0</td>
<td>117.0</td>
</tr>
<tr>
<td>deep output</td>
<td>-</td>
<td>1</td>
<td>4</td>
<td>47.21</td>
<td>21.5</td>
<td>24.2</td>
<td>28.7</td>
<td>98.9</td>
</tr>
</tbody>
</table>

Table 4: Deep encoder–decoder results. English→German WMT17 data. Transition depth 4/2 means 4 in the base RNN of the stack and 2 in the higher RNNs. The last two models are large and their results are highlighted separately.

Figure 5: Subject-verb agreement accuracy as a function of distance between subject and verb.
stacked encoders (Wu et al., 2016), both in accuracy and (single-GPU) speed. We showed that deep transition architectures, which we first applied to NMT, perform comparably to the stacked ones in terms of accuracy (BLEU, cross-entropy and long-distance syntactic agreement), and better in terms of speed and number of parameters.

We found that depth improves BLEU scores especially in the encoder. Decoder depth, however, still improves cross-entropy if not strongly BLEU scores. The best results are obtained by our BiDeep architecture which combines both stacked depth and transition depth in both the (alternating) encoder and the decoder, yielding better accuracy for the same number of parameters than systems with only one kind of depth.

We recommend to use combined architectures when maximum accuracy is the goal, or use deep transition architectures when speed or model size are a concern, as deep transition performs very positively in the quality/speed and quality/size trade-off.

While this paper only reports results for one translation direction, the effectiveness of the presented architectures across different data conditions and language pairs was confirmed in follow-up work. For the shared news translation task of this year’s Conference on Machine Translation (WMT17), we built deep models for 12 translation directions, using a deep transition architecture or a stacked architecture (alternating encoder and rGRU decoder), and observe improvements for the majority of translation directions (Sennrich et al., 2017a).

Acknowledgments

The research presented in this publication was conducted in cooperation with Samsung Electronics Polska sp. z o.o. - Samsung R&D Institute Poland. This project received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreements 645452 (QT21), 644402 (HimL) and 688139 (SUMMA).

References


