A study of latent monotonic attention variants

Albert Zeyer 1 2  Ralf Schlüter 1 2  Hermann Ney 1 2

Abstract
End-to-end models reach state-of-the-art performance for speech recognition, but global soft attention is not monotonic, which might lead to convergence problems, to instability, to bad generalisation, cannot be used for online streaming, and is also inefficient in calculation. Monotonicity can potentially fix all of this. There are several ad-hoc solutions or heuristics to introduce monotonicity, but a principled introduction is rarely found in literature so far. In this paper, we present a mathematically clean solution to introduce monotonicity, by introducing a new latent variable which represents the audio position or segment boundaries. We compare several monotonic latent models to our global soft attention baseline such as a hard attention model, a local windowed soft attention model, and a segmental soft attention model. We can show that our monotonic models perform as good as the global soft attention model. We perform our experiments on Switchboard 300h. We carefully outline the details of our training and release our code and configs.

1. Introduction
The conventional hidden Markov model (HMM), including the hybrid neural network (NN) / HMM (Bourlard & Morgan, 1990; Robinson, 1994) is a time-synchronous model, which defines a probability per input frame (either \( p(x_t | y_t) \) or \( p(y_t | x_t) \)). As such, it implicitly enforces monotonicity. Connectionist temporal classification (CTC) (Graves et al., 2006) can be seen as a special case of this (Zeyer et al., 2017). Generalisations of CTC are the recurrent neural network transducer (RNN-T) (Battenberg et al., 2017; Graves, 2012) or the recurrent neural aligner (Dong et al., 2018; Sak et al., 2017) or further extended generalized variations of the transducer (Zeyer et al., 2020).

The encoder-decoder-attention model (Bahdanau et al., 2014) is becoming one of the most competitive models for speech recognition (Chiu et al., 2018; Park et al., 2019; Tüske et al., 2020; Zeyer et al., 2018b). This model directly models the posterior probability \( p(y_N^t | x_T^1) \) for the output labels \( y_N^t \) (word sequence), given the input sequence \( x_T^1 \) (audio features). This is factorised into \( \prod_t p(y_t | y_{t-1}^t, x_T^1) \). We call this a label-synchronous model (sometimes also called direct model) because each NN output corresponds to one output label. This is in contrast to the time-synchronous models like hybrid NN/HMM, CTC and RNN-T. The encoder-decoder (label-sync.) model usually uses global soft attention in the decoder. I.e. for each output label \( i \), we access the whole input sequence \( x_T^i \). This global soft attention mechanism is inefficient (time complexity \( O(T \cdot N) \)) and can violate monotonicity.

Monotonicity is necessary to allow for online decoding. Also, for monotonic sequence-to-sequence tasks like speech recognition, a model which is enforced to be monotonic might converge faster, and should be more stable in decoding. In this work, we want to introduce monotonicity in a mathematically clean way by introducing a latent variable which presents the position in the audio (or frame-wise encoder).

2. Related work
Several approaches exists to introduce monotonicity for label-synchronous models. We categorize these different approaches and outline their advantages and disadvantages.

2.1. Soft constraints: By training, or by attention energy tendency
Monotonicity can be encouraged by the model (e.g. by modeling the attention energy computation in such a way that it tends to be monotonic) or during training (by additional losses, or some initialization with linear alignment) (Tachibana et al., 2018; Zhang et al., 2018). While these help for convergence, these are soft constraints, or just guidance, i.e. the model still can violate monotonicity.

2.2. Hybrids of attention and CTC/RNN-T
The idea is that a time-synchronous model like CTC or RNN-T enforces the monotonicity, and the attention model
contributes for better performance (Chiu et al., 2019; Hori et al., 2017; Kim et al., 2017; Miao et al., 2019; Moritz et al., 2019a;b; Watanabe et al., 2017). Often there is a shared encoder, and one CTC output layer and another common decoder with attention, and the decoding combines both the CTC and the attention decoder. The decoding can be either implemented in a time-synchronous way or in a label-synchronous way. Depending on the specific details of the model and decoding, monotonicity is strictly enforced or just strongly encouraged (but it could happen that the attention scores dominate over the CTC model or so). These models usually perform quite well. However both the model and the decoding procedure become more complicated, and often rely on several heuristics. And it can be seen as a model combination, which makes the performance comparison somewhat unfair to single models.

### 2.3. Deterministic methods

Monotonicity can be introduced in a more conceptional way via deterministic methods, which model explicitly the source position (Chiu & Raffel, 2017; Fan et al., 2018; Graves, 2013; He et al., 2019; Hou et al., 2017; Jaitly et al., 2016; Luong et al., 2015; Merboldt et al., 2019; Miao et al., 2019; Raffel et al., 2017; Tjandra et al., 2017; Zhang et al., 2019). These approaches all provide strict monotonicity by using some position scalar (e.g. for a running window over the encoder output), and it is enforced that the position can only increase. In these approaches, the position is not interpreted as a latent variable, but it is rather calculated by a deterministic function given the current state of the decoder. These can be fully differentiable (if the position is a real number scalar) or rely on heuristics/approximations (if the position is a discrete number). This includes the implicit assumption that there exists a deterministic method which determines the position for the next label. We argue that this is a too strong assumption. In practice, given some history of input (audio) and previous labels (words), the model cannot tell for sure where the next label will occur. The model eventually can only estimate a probability or score. And later in decoding, after a few more input frames or words, we might see that the model did a wrong choice for the position. The beam search should be able to correct this, which is not possible if it is simply a deterministic function. If it cannot correct it, then the performance will have a bottleneck at this deterministic function, because when its estimate is bad or wrong, the further decoding will be affected consequently. MoChA (Chiu & Raffel, 2017) is probably one of the most prominent examples in this category. As it still does soft attention within one chunk, it can compensate bad chunk positions to some degree but not fully.

### 2.4. Latent variable models

A variety of works have introduced an explicit latent variable to model the alignment. We always have and even need this for hybrid NN/HMM models, CTC and RNN-T. One example of early label-synchronous models which have such a latent variable are the segmental conditional random fields (CRF) (Lu et al., 2016; Ostendorf et al., 1996; Zweig & Nguyen, 2009). The latent variable is usually discrete, and often represents the position in the encoder, or the segment boundaries. We will categorize the model for the latent variable by dependency orders in the following.

#### 2.4.1. 0-ORDER MODELS

The probability distribution of the latent variable can be independent from its history (0 order) (Bahuleyan et al., 2017; Deng et al., 2018; Shankar & Sarawagi, 2019; Shankar et al., 2018; Wu et al., 2018). The marginalisation over the latent variable becomes simple and efficient in that case and can easily be done in both training and decoding. However, it does not allow for strong alignment models, and also does not allow to constrain for monotonicity, because that would add a dependence on the history and then it is not 0-order anymore.

#### 2.4.2. 1ST ORDER MODELS

If the probability distribution of the latent variable depends only on its predecessor, then it is of 1st order (Alkhouli et al., 2016; Beck et al., 2018a;b; Wang et al., 2017; 2018; Yu et al., 2016a;b). A first-order dependency allows to directly enforce monotonicity. The full marginalisation can still be calculated efficiently for 1st order models using dynamic programming (Wang et al., 2018; Yu et al., 2016b).

#### 2.4.3. HIGHER ORDER MODELS

The probability distribution of the latent variable can also depend on more than just the predecessor, or even on the full history (Alkhouli & Ney, 2017; Alkhouli et al., 2018; Arivazhagan et al., 2019; Ba et al., 2014; Lawson et al., 2018; Mnih et al., 2014; Vinyals et al., 2015; Xu et al., 2015). This is often the case if some recurrent decoder is used and the probability distribution depends on that state, which itself depends on the full history. The full marginalisation becomes infeasible for higher order, thus requires further approximations such as the maximum approximation (Alkhouli & Ney, 2017; Alkhouli et al., 2018).

#### 2.4.4. DECODING WITH FIRST OR HIGHER ORDER LATENT VARIABLE MODELS

Only few other work includes the latent variable in beam search during decoding (Alkhouli & Ney, 2017; Alkhouli et al., 2016; 2018; Beck et al., 2018a; Wang et al., 2018;
Yu et al., 2016a;b) while most other work uses sampling or a greedy deterministic decision during decoding, which again leads to all the problems outlined in Section 2.3. We argue that we should properly search over the latent variable space and incorporate that into the beam search decoding procedure such that the decoding can correct wrong choices on the latent variable. Also this is mostly straightforward and mathematically clean.

Note that if we simplify the beam search to simply pick the \( \arg \max \) for the latent variable in each decoder step, we get back to deterministic models. In that sense, latent variable models are a strict generalization of all the deterministic approaches.

### 3. This work: our latent variable model

We conceptually introduce monotonicity in a mathematically clean way to our label-synchronous encoder-decoder model by a discrete latent variable which represents the position (or segment boundary) in the audio (or frame-wise encoder). We now have several options for the model assumptions. For the monotonicity, we at least need a model of 1st order. We make it higher order and depend on the full history. This is a trade-off between a more powerful model and more approximations needed in decoding and training, such as the maximum approximation. The maximum approximation has successfully and widely been used for speech recognition in different scenarios such as decoding and Viterbi training for hybrid NN/HMM models (Yu & Deng, 2014). This is why we think the maximum approx. should be fine for label-sync. models as well. And on the other side, a more powerful model could improve the performance.

We argue that these models with a latent variable are conceptually better founded than the deterministic models which rely on heuristics without good mathematical justification. In addition, in such an approach, the conventional maximum approximation can be naturally introduced to achieve efficient and simple training and decoding.

As we want to understand the effect of this latent variable, and the hard monotonicity constrain on it, we want to keep the differences to our baseline encoder-decoder model with global soft attention as small as possible. This means that we still have a bidirectional long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) encoder and an offline feature extraction pipeline is offline. There is a wide range of existing work on making these missing parts online capable (Peddinti et al., 2017; Xue & Yan, 2017; Zeyer et al., 2016). In all our latent models, the added latent variable represents either:

- a single discrete position in the encoder output,
- or a segment boundary on the encoder output.

To further keep the difference to the baseline small, the probability of the latent variable is simply given by the attention weights (but masked to fulfil monotonicity). Also the remaining part of the decoder is kept exactly the same, except of how we calculate the attention context, which is not by global soft attention anymore. We do not improve on the runtime complexity in this work, to stay closer to the baseline for a better comparison. However, it is an easy further step to make the runtime linear, as we will outline later.

Also, in the existing works, the final model properties are rarely discussed beyond the model performance. We perform a systematic study of different properties of our latent variable models, considering modeling, training, and decoding aspects.

The experiments were performed with RETURNN (Zeyer et al., 2018a), based on TensorFlow (Abadi et al., 2015). All the code and the configs for all our experiments is publicly available\(^1\).

### 4. Models

We closely follow the LSTM-based global soft attention model as defined in (Zeyer et al., 2018b;c; 2019). In all cases, the model consists of an encoder, which learns a high-level representation of the input signal \( x^T \),

\[
 h_{\text{1}}^T = \text{Encoder}(x_{\text{1}}^T).
\]

We apply time downsampling, thus we have \( T' \leq T \). In our case, it is a bidirectional LSTM with time-max-pooling. The global soft attention model defines the probability

\[
p(y_i^N|\cdot) = \prod_{i=1}^{N} p(y_i|y_{1}^{i-1}, x_{\text{1}}^T)
\]

with

\[
p(y_i|y_{1}^{i-1}, x_{\text{1}}^T) = p(y_i|y_{1}^{i-1}, h_1^{T'} c_i) = p(y_i|y_{1}^{i-1}, c_i),
\]

for \( i \in \{1, \ldots, N\} \), where \( c_i \) is the attention context vector. We get the dependence on the full sequence \( c_1^T \) because we use a recurrent NN.

### 4.1. Baseline: Global soft attention

There is no latent variable. The attention context \( c_i \) is calculated as

\[
c_i = \sum_{t=1}^{T'} \alpha_{i,t} (y_{1}^{i-1}, \alpha_{1}^{i-1}, c_{1}^{i-1}) \cdot h_t,
\]

---

\(^1\)https://github.com/rwth-i6/returnn-experiments/tree/master/2021-latent-monotonic-attention
We also use a pretraining scheme such as growing the model size, which we outline in the experiments section. Decoding is done by searching for

\[
\arg \max_{N,y_1^N} \sum_{i=1}^{N} \log p(y_i|y_1^{i-1}, x_1^{T})
\]

which is approximated using label-synchronous beam search.

### 4.2. Latent models

When introducing a label-synchronous discrete latent variable \( t_i \) for all \( i \in \{1, \ldots, N\} \), we get

\[
p(y_i^N|x_1^T) = \sum_{t_1^N} p(y_i^N, t_1^N|x_1^T)
\]

\[
= \sum_{t_i^N} \prod_{i=1}^{N} p(y_i, t_i|y_1^{i-1}, t_{i-1}^{i-1}, x_T^1)
\]

\[
= \sum_{t_i^N} \prod_{i=1}^{N} p(y_i|y_1^{i-1}, t_i^1, x_T^1)p(t_i|y_1^{i-1}, t_{i-1}^{i-1}, x_T^1).
\]

Analogous to the baseline, we use

\[
p(y_i|y_1^{i-1}, t_i^1, x_T^1) = p(y_i|y_1^{i-1}, c_i^1),
\]

where the attention context \( c_i \) depends also on \( t_i \) now.

We construct the model in a way that the access to \( h_1^T \) via \( t_i \) is monotonic by \( t_i \leq t_{i+1} \) or even \( t_i < t_{i+1} \). In every case, the model is trained again by minimising the negative log probability, and using the max. approximation for \( \sum_{t_i^N} \),

\[
L = - \sum_{(x,y) \in D} \max_{t_i^N} \sum_{i=1}^{N} \log p(y_i, t_i|y_1^{i-1}, t_{i-1}^{i-1}, x_T^1).
\]

The \( \max_{t_i^N} \) is further approximated by beam search, which is also called forced alignment or Viterbi alignment. Also, in all cases, we try to be close to the global soft attention model, such that a comparison is fair and direct, and even importing model parameters is reasonable.

The probability for the latent variable \( t_i \) is simply given by the attention weights

\[
p(t_i = t|y_1^{i-1}, t_{i-1}^{i-1}, x_T^1) = \alpha_{i,t}^t
\]

with \( \alpha_{i,t}^t = 0 \quad \forall t < t_i-1 \). The attention weights in this case are masked to fulfill monotonicity \( t < t_{i-1} \) if we want strict monotonicity, else \( t \leq t_{i-1} \), and \( \alpha_{i,t}^t \propto \alpha_{i,t}^t \) renormalised accordingly. We choose this model \( p(t|\ldots) \) based on the attention weights to keep a strong similarity to the baseline model. However, because of this model, we still have \( O(T \cdot N) \) runtime complexity. In future work, we can further deviate from this, and compare to other models for \( p(t|\ldots) \) which allow for linear runtime \( O(T+N) \) or even \( O(T) \). E.g. an obvious alternative is to use a Bernoulli distribution and to define it frame-wise, not globally over all frames.

In all latent models, we define the attention context \( c_i \) as

\[
c_i = \sum_{t=w_i}^{w'_i} \alpha_{i,t}^t \cdot h_t,
\]

i.e. we use soft attention on a window \([w_i, w'_i]\) (or hard, if it is a single frame), for some \( \alpha^t \) as described in the following.

Decoding is done by searching for

\[
\arg \max_{N^T,y_1^N} \sum_{i=1}^{N} \log p(y_i, t_i|y_1^{i-1}, t_{i-1}^{i-1}, x_T^1)
\]

which is approximated using label-synchronous beam search.

### 4.3. Monotonic hard attention

By simply setting \([w_i, w'_i] = [t_i, t_i]\) in Equation (1), we have the attention only on a single frame, and simply \( \alpha_{i,t}^t \) = 1. This can be interpret as hard attention instead of soft attention, and \( t_i \) is the position in \( h_1^T \). Everything else stays as before. There are multiple motivations why to use hard attention instead of soft attention:

- The model becomes very simple.
- In the global soft attention case, we can experimentally see that the attention weights are usually very sharp, often focused almost exclusively on a single frame.

### 4.4. Monotonic local windowed soft attention

We use \([w_i, w'_i] = [t_i - D_t, t_i + D_t]\) for fixed \( D_t, D_r \). In this case, the latent variable \( t_i \) is the center window position, and \( \alpha_{i,t}^t \propto \alpha_{i,t}^t \) renormalised on the window. This model is very similar to (Merboldt et al., 2019) but with a latent variable.
4.5. Monotonic segmental soft attention

Here we make use of the latent variable $t_i$ as the segment boundary, i.e. we use $[w_i, w'_i] = [t_{i-1} + 1, t_i]$. Here, we have a separate model for $\alpha'_t$, which are separately computed MLP soft attention weights.

5. Training procedure for latent models

For one training sample $(x, y) \in \mathcal{D}$, the loss we want to minimise is

$$L = -\max_{t_i} \log p(y_i^N, t_i^N | x_1^T),$$

i.e. we have to calculate $\arg \max_{t_i} \log p(y_i^N, t_i^N | x_1^T)$, i.e. search for the best $t_i^N$, which we call alignment. This is the maximum approximation for training. This can be done using dynamic programming (beam search, Viterbi). We could also decouple the search for the best alignment from updating the model parameters, but in our experiments, we always do the search online on-the-fly, i.e. in every single mini-batch, it uses the current model for the search. We note that this beam search is exactly the same beam search implementation which we use during decoding, but we keep the ground truth $y_i^N$ fixed, and only search over the $t_i^N$.

For stable training, we use the following tricks, which are common for other models as well, but adopted now for our latent variable models.

- The maximum approximation can be problematic in the very beginning of the training, when the model is randomly initialized. In the case of HMM models, it is common to use a linear alignment in the beginning, instead of calculating the Viterbi alignment ($\arg \max_{t_i} \log p(y_i^N, t_i^N | x_1^T)$). We do the same for our latent variable models.

- We store the last best alignment (initially the linear alignment) and its score. When we calculate a new Viterbi alignment, we compare the score to the previous best score, and update the alignment if the score is better. This greatly enhances the stability early in training, esp. when we still grow the model size in pretraining.

- We can use normal global soft attention initially. Note that this is still slightly different compared to the global soft attention baseline because of the other small differences in the model, such as different (hard) weight feedback.

Given an existing alignment $t_i^N$ for this training example, the loss becomes

$$L = \sum_{i=1}^N \log p(y_i | y_i^{i-1}, c_i^1) - \log p(t_i | y_i^{i-1}, c_i^{i-1}).$$

We note that this is the same loss as before, with the additional negative log likelihood for $p(t_i | \ldots)$. We can also easily introduce a different loss scale for each log likelihood. We use the scale 0.1 by default for $p(t_i | \ldots)$.

As we do the search for the optimal $t_i^N$ on-the-fly during training, this adds to the runtime and makes it slower. However, on the other side, the model itself can be faster, depending on the specific implementation. In our case, for the hard attention model, we see about 40% longer training time. If we would use a fixed alignment $t_i^N$, it would actually be faster to train than the baseline.

6. Decoding with latent variable

The decoding without an additional latent variable is performed as usual, using label synchronous beam search with a fixed beam size (Bahdanau et al., 2014; Sutskever et al., 2014; Zeyer et al., 2018a). When we introduce the new latent variable into the decoding search procedure, the beam search procedure stays very similar. We need to search for

$$\arg \max_{N,y^N,t^N} \sum_{i=1}^N \log p(y_i | y_i^{i-1}, c_i^1) + \log p(t_i | y_i^{i-1}, c_i^{i-1}).$$

As $p(y_i | \ldots)$ depends on $t_i$ (via $c_i$), in any decoder step $i$, we first need to hypothesize possible values for $t_i$, and then possible values for $y_i$, and then repeat for the next $i$. We have two beam sizes, $K_y$ and $K_t$ for $p(y | \ldots)$ and $p(t | \ldots)$ respectively.

The expansion of the possible choices for $t_i$ is simply analogous as for the output label $y_i$. We fully expand all possible values for $t_i$, i.e. calculate the score $p(t_i = \ldots) \cdot p(y_i^{i-1}, t_i^{i-1} | x_1^T)$ for all possible $t_i$ for all current active hypotheses, just as we do for $y$. However, we have various options for when, what and how do we apply pruning. After we calculated the scores for $y_i$, based on the joint scores $p(y_i = \ldots) \cdot p(y_i^{i-1}, t_i^{i-1} | x_1^T)$, we prune to the best $K_y$ hypotheses. Now on the choice of $t_i$, we have $K \cdot T'$ active hypotheses, where $K = 1$ at the beginning, and then $K = K_y$. For pruning the hypotheses on $t_i$ based on the joint scores $p(t_i = \ldots)$:

- We either select the overall $K_t$ best, ending up with $K_t$ hypotheses.
- Or for each previous hypotheses (last choice on $y_{i-1}$), we select the best $K'_t$ choices for $t_i$, ending up with $K_t = K \cdot K'_t$ hypotheses. I.e. we only expand and don’t prune $y_{i-1}$ hypotheses away at this point, but only prune away new choices of $t_i$, i.e. take the top-$K'_t$ of possible $t_i$ values.

As a further approximation, we can also discard hypotheses in the beam which have the same history $y_1^i$ but a different
Table 1. Overall comparisons of models. We use the beam sizes $K_0 = 12$ and $K_1 = 48$ with expand on $t$ as explained in Section 6. The frame error rate (FER) is calculated on the labels, feeding in the ground truth seq., on some cross-validation set. The latent models all import the global soft att. baseline, which screws the effective num. of epochs though, as the model is changed drastically.

<table>
<thead>
<tr>
<th>Model</th>
<th>Effect. num. ep.</th>
<th>WER [%]</th>
<th>Hub5’00</th>
<th>Hub5’01</th>
<th>Hub5’00</th>
<th>Hub5’01</th>
<th>FER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Σ</td>
<td>SWB</td>
<td>CH</td>
<td>Σ</td>
<td>SWB</td>
<td>CH</td>
<td></td>
</tr>
<tr>
<td>Global soft.</td>
<td>33.3</td>
<td>15.3</td>
<td>10.1</td>
<td>20.5</td>
<td>14.9</td>
<td>8.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>66.7</td>
<td>14.3</td>
<td>9.3</td>
<td>19.3</td>
<td>14.0</td>
<td>8.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>14.4</td>
<td>9.1</td>
<td>19.7</td>
<td>13.9</td>
<td>8.0</td>
<td></td>
</tr>
<tr>
<td>Segmental</td>
<td>58.3</td>
<td>15.7</td>
<td>9.9</td>
<td>21.5</td>
<td>14.6</td>
<td>7.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>83.3</td>
<td>15.7</td>
<td>10.1</td>
<td>21.3</td>
<td>14.9</td>
<td>6.9</td>
<td></td>
</tr>
<tr>
<td>Hard att.</td>
<td>33.3</td>
<td>15.8</td>
<td>10.3</td>
<td>21.3</td>
<td>15.4</td>
<td>7.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>15.1</td>
<td>9.7</td>
<td>20.5</td>
<td>14.8</td>
<td>7.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>91.7</td>
<td>14.4</td>
<td>9.3</td>
<td>19.5</td>
<td>14.2</td>
<td>7.0</td>
<td></td>
</tr>
<tr>
<td>Local win.</td>
<td>33.3</td>
<td>15.4</td>
<td>10.2</td>
<td>20.5</td>
<td>14.9</td>
<td>8.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>14.7</td>
<td>9.4</td>
<td>20.0</td>
<td>14.1</td>
<td>7.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>83.3</td>
<td>14.4</td>
<td>9.1</td>
<td>19.7</td>
<td>13.8</td>
<td>7.6</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. From scratch vs imported initialization, for hard attention and local windowed soft attention. WER on Hub5’00.

<table>
<thead>
<tr>
<th>Model</th>
<th>Imported baseline, num. of ep.</th>
<th>Effective num. of ep.</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard att.</td>
<td>—</td>
<td>33.3</td>
<td>58.3</td>
</tr>
<tr>
<td></td>
<td>66.7</td>
<td>91.7</td>
<td>14.4</td>
</tr>
<tr>
<td>Local win.</td>
<td>—</td>
<td>33.3</td>
<td>50</td>
</tr>
</tbody>
</table>

We currently do this for the Viterbi alignment during training with ground truth, such that we only consider

$$t_i \mapsto \max_{t_i'} \max_{p(t_i| \ldots)} p(y_t^{-1}, t_i^{-1}| x_t^T).$$

We do not use this for decoding yet. In general, this can make more efficient use of the beam.

Note that this is in the worst case only slower to the baseline by some small constant factor. In practice, in the batched GPU-based beam search which we do, we do not see any difference. Once we use different models for $p(t_i| \ldots)$, we can even potentially gain a huge speedup.

7. Experiments

7.1. Dataset & global soft attention baseline

All our experiments are performed on Switchboard (Godfrey et al., 1992) with 300h of English telephone speech. Our global soft attention baseline is mostly based on (Zeyer et al., 2018b;c; 2019). We use SpecAugment (Park et al., 2019) for data augmentation. We have two 2D-convolution layers, followed by 6 layer bidirectional LSTM encoder of dimension 1024 in each direction, with two time-max-pooling layers which downsample the time dimension by a factor of 6 in total. Our output labels are 1k BPE subword units (Sennrich et al., 2015). All our experiments are without the use of an external language model. We use a pretraining/training-scheduling scheme which

- starts with a small encoder, consisting of 2 layers and 512 dimensions, and slowly grows both in depth (num-

This scheme is performed during the first 10 full epochs of training. In addition, for the first 1.5 epochs, we do learning rate warmup, and after that we do the usual learning rate scheduling (Bengio, 2012). After every 33.3 epochs, we do a reset of the learning rate. We show the performance in Table 1. We see that longer training time can yield drastic improvements. We train these experiments using a single Nvidia GTX 1080 Ti, and one epoch takes about 3h-6h, depending on the model.

7.2. Latent variable models

7.2.1. Pretraining scheme

As all our proposed models are close to the global soft attention baseline, we can import the model parameters from the baseline and fine tune. With the aforementioned methods for stable training such as reusing the previously best alignment, we are also able to train from scratch, as we show in Table 2. Note that the difference in “from scratch” training to importing a baseline is somewhat arbitrary here, because the latent variable models also use some pretraining scheme as outlined in Section 5. Thus we end up with two different kinds of pretraining schedules, and “from scratch” training uses much less global soft attention during pretraining. This merely demonstrates that from scratch training is possible, i.e. that we can switch to a fully latent model (without global soft attention) already early in training. The performance difference in these experiments might also just be due to the different effective number of epochs. Note that ultimately the effect of the global soft attention model to a latent variable model, we will always see a drop in WER, as the model is not the same. The amount of this degradation can be seen in Table 6. In the following experiments we
use the best possible scheduling, which currently imports existing global soft attention models.

### 7.2.2. Prune variants in decoding

We analyse the different kinds of pruning on the latent variable $t_i$ during decoding, and the beam size $K_t$, as explained in Section 6. Results are collected in Table 3. The results indicate that we need a large enough beam size for optimal performance. For the hard attention model, a simple deterministic decision on $t_i$ is slightly worse than doing beam search on it. It does not seem to matter too much for the local windowed attention model, which probably can recover easily if the choice on $t_i$ is slightly off. Note that this is very much dependent on the quality of the model $p(t | \ldots)$. If that model is weak, it should be safer and better to use the expand prune variant with some $K_t > K_y$, as it would not prune away possibly good $y_t$ hypotheses. However, our experiments do not demonstrate this yet, which could be simply due to an already powerful model. In our experiments $K_y = 12$, $K_t = K_y \cdot K_t' = 48$, i.e. $K_t' = 4$ seems to be enough. That means that we effectively consider always the top 4 scored choices for $t_i$. We also did experiments on overall larger beam sizes for $K_y$ as well but do not see any further improvement. We also experimented with an exponent on the probability, i.e. $p(t | \ldots)^\alpha$, but $\alpha = 1$ seems to perform best. We also tried different softmax temperature factors for $p(t | \ldots)$, but again 1 seems to be the optimum. These experiments are shown in Table 4. We note again that this behaviour is likely dependent on the specific kind of model $p(t | \ldots)$.

### 7.2.3. Overall results

We collect the final results in Table 1. We again also report the total number of epochs of the training data which were seen. We also report the frame error rate (FER), which we found interesting, as it is consistently much better for all of the latent models. Our assumption is that the alignment procedure works well and it makes a good prediction of the output label easier, given the good position information. However, we do not see this for the WER. The segmental model seems to have the highest FER, while it performs not as good as the other latent models. This could be due to some overfitting effect, but needs further studying. We assume this is due to the exposure bias in training, where it always has seen the ground truth label sequence and a good alignment. The global soft attention model has less problems with this, as there is never a fixed alignment. Overall, the WERs of the latent models are competitive to global soft attention.

### 7.2.4. Restriction on the max step size $t_i - t_{i-1}$

In all the experiments, the model $p(t | \ldots)$ operates offline on the whole input sequence, and then calculates a softmax over the time dimension. For an online capable model, we cannot use softmax for the normalisation unless we

---

**Table 3.** Comparison on different decode pruning on $t_i$, for a hard attention model. We always use the beam size $K_y = 12$. Expand as explained in Section 6, and $K_t$ is the beam size after the choice on $t_i$. In the case of expand with $K_t = K_y$, we get the deterministic arg max $y_i$ approach. WER on Hub 5’00.

<table>
<thead>
<tr>
<th>Model</th>
<th>Expand $t_i$</th>
<th>$K_t$</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard att.</td>
<td>yes</td>
<td>12</td>
<td>14.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>14.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>48</td>
<td>14.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>96</td>
<td>14.4</td>
</tr>
<tr>
<td>no</td>
<td>12</td>
<td>14.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>14.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>14.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>96</td>
<td>14.4</td>
<td></td>
</tr>
<tr>
<td>Local win.</td>
<td>yes</td>
<td>12</td>
<td>14.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>48</td>
<td>14.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>96</td>
<td>14.7</td>
</tr>
<tr>
<td>no</td>
<td>12</td>
<td>14.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>14.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>96</td>
<td>14.7</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.** Comparison on different $\alpha$ for $p(t | \ldots)^\alpha$ and softmax temperature for a hard attention model. We always use the beam size $K_y = 12$, $K_t = 48$, and expand $t_i$ (Section 6). WER on Hub 5’00.

<table>
<thead>
<tr>
<th>Softmax temp.</th>
<th>$p(t)$ scale</th>
<th>$\alpha$</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.1</td>
<td>21.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>14.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>14.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>14.6</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>1.0</td>
<td>14.5</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>1.0</td>
<td>14.6</td>
<td></td>
</tr>
</tbody>
</table>

**Table 5.** Comparison on different maximum step sizes for $p(t | \ldots)$. WER on Hub 5’00. In the unlimited case, there can be step sizes over 150 frames (9s), although the mean step size is about 4.3 ± 4.0 frames (0.26s ± 0.24s).

<table>
<thead>
<tr>
<th>Model</th>
<th>Max. step size</th>
<th>WER [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard att.</td>
<td>10 (0.6s)</td>
<td>18.1</td>
</tr>
<tr>
<td></td>
<td>20 (1.2s)</td>
<td>14.7</td>
</tr>
<tr>
<td></td>
<td>30 (1.8s)</td>
<td>14.5</td>
</tr>
<tr>
<td></td>
<td>∞</td>
<td>14.4</td>
</tr>
<tr>
<td>Local win.</td>
<td>10 (0.6s)</td>
<td>14.6</td>
</tr>
<tr>
<td></td>
<td>20 (1.2s)</td>
<td>14.5</td>
</tr>
<tr>
<td></td>
<td>30 (1.8s)</td>
<td>14.5</td>
</tr>
<tr>
<td></td>
<td>∞</td>
<td>14.5</td>
</tr>
</tbody>
</table>
restrict the maximum possible step size, which is maybe a good idea in general. Specifically, $p(t_i|\ldots) = 0$ for all $t > t_{i-1} + \Delta_i$ for some $\Delta_i$. We collected the results in Table 5. We can conclude that a maximum step size of 30 frames, which corresponds to 1.8 seconds, is enough, although this might depend on the dataset.

### 7.2.5. Further discussion

Achieving to the final results required multiple states of tuning. In Table 6, we share our development history up to the final results. The starting point does not use label smoothing, which was actually a bug and not intended. We see that label smoothing has a big positive effect. Even bigger was the effect by not shuffling the sequences, which was a bug as well. As we can see from the table, this makes an absolute WER difference of more than 1%. While it was never the intention to do that, we think it might still be interesting for the reader to follow the history of our development. We see that in our initial experiments, the training actually made the model worse, which can be explained by running into a bad local optimum due to an unstable alignment procedure. Fortunately the alignment procedure becomes more stable for the further experiments. The global soft attention baseline also used weight feedback, and it was not obvious whether the latent models should have that as well, and how exactly. We ended up in the variant that we use the hard choice on $t_i$ instead of the attention weights, and create a “hard” variant of accumulated weights, although it is not fully clear whether this is the optimal solution.

In Figure 1 we can see the different alignment behaviour and attention weights of each model. We can see that most models behave as intended. In this direct comparison, we see that the global soft attention is noticeably less sharp.

### 8. Conclusion & future work

We introduced multiple monotonic latent attention models and demonstrated competitive performance to our strong global soft attention baseline. From the low FER, we speculate that we have a stronger exposure bias problem now (not only the ground truth labels $y_t^{N-1}$ but also the time alignment $t_1^N$). This problem can be solved e.g. via more regularisation or different training criteria such as minimum WER training (Prabhavalkar et al., 2018) and we expect to get future improvements. By further tuning, eventually we expect to get consistently better than the global soft attention baseline. Future work will also include different alignment models as well as online capable encoders. Also, while we do label-synchronous decoding in this work, a latent variable model allows for time-synchronous decoding as well, which is esp. interesting the the context of online streaming.
Acknowledgements

This work has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 694537, project "SEQ-CLAS") and from a Google Focused Award. The work reflects only the authors’ views and none of the funding parties is responsible for any use that may be made of the information it contains.

References


A study of latent monotonic attention variants


