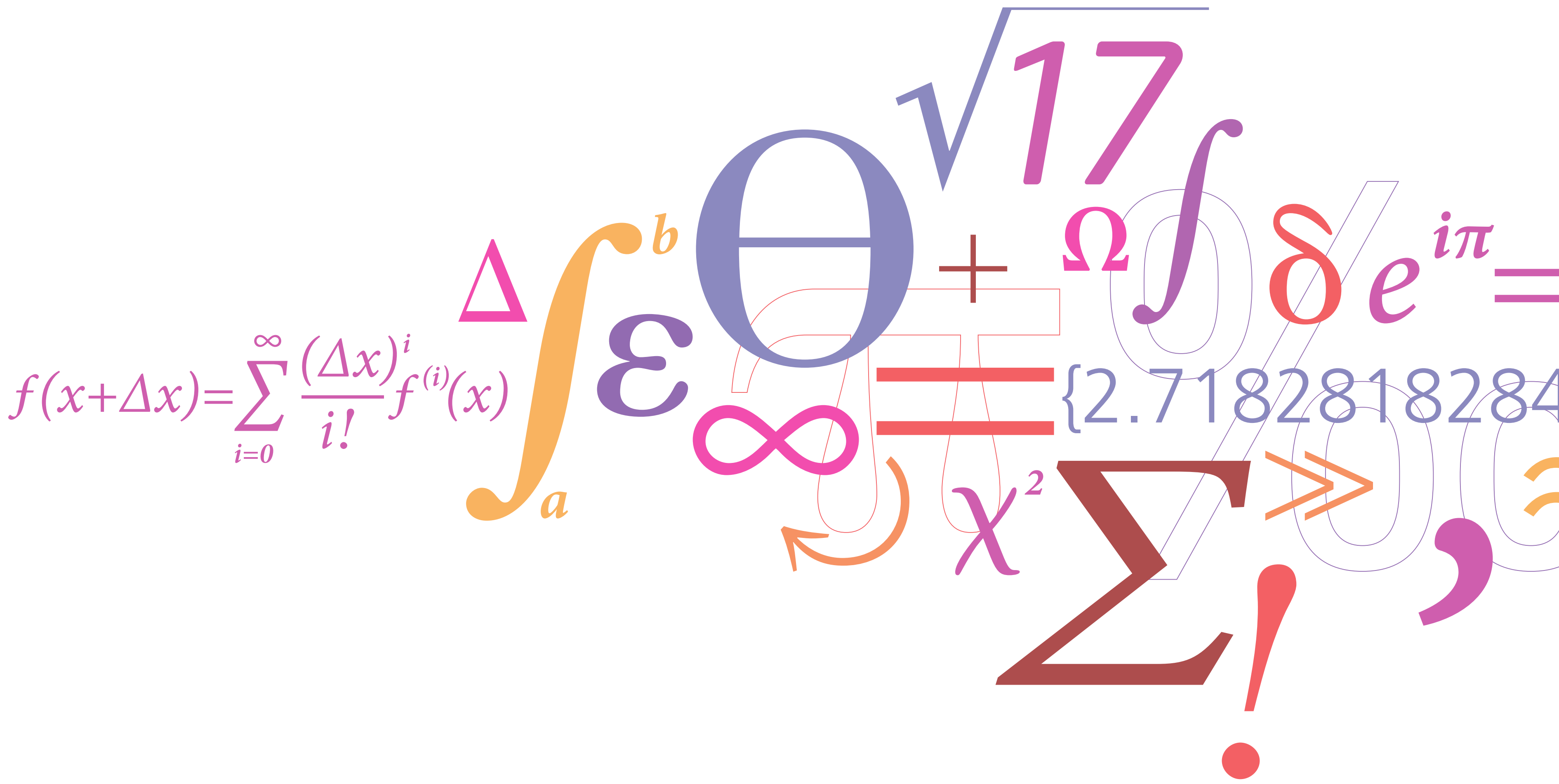
Semi-Supervised Neural Machine Translation

for small bilingual datasets



Task

German

Die Premierminister Indiens und Japans trafen sich in Tokio.

English

India and Japan prime ministers meet in Tokyo

- Google Translates processes 100 billion words each day. And have 500 million users.
- Even in EU and UN automated translation tools are used.
- Most internet content is in English. However, in India only 20% speaks English.
- Essential for modern politics and education.

Task

Europarl v7

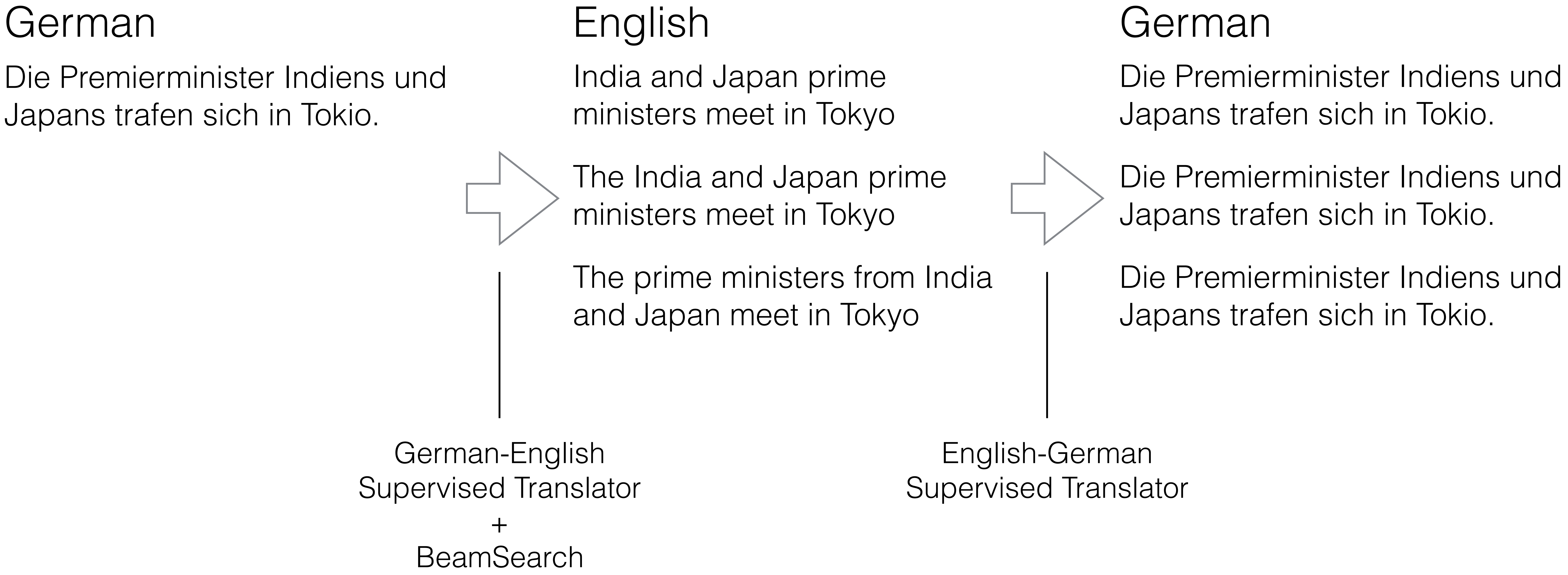
Language Pair	Sentences
French-English	2 007 723
Danish-English	1 968 800
German-English	1 920 209
Polish-English	632 565
Romanian-English	399 375

- Neural Machine Translation requires huge bilingual datasets. For some language pairs that doesn’t exists.
- Monolingural datasets are very large. Sources are Wikipedia, National News, etc.
- Combine monolingural data with bilingual data for better translation.

Approach

Semi-Supervised Neural Machine Translation is very ambitious, thus the approach has to pragmatic.

Approach



German

Die Premierminister Indiens und Japans trafen sich in Tokio.

English

India and Japan prime ministers meet in Tokyo

The India and Japan prime ministers meet in Tokyo

The prime ministers from India and Japan meet in Tokyo

German

Die Premierminister Indiens und Japans trafen sich in Tokio.

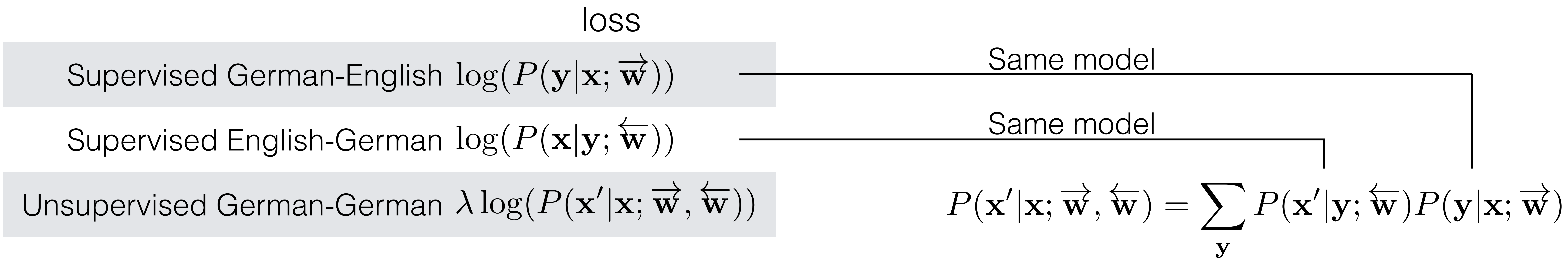
Die Premierminister Indiens und Japans trafen sich in Tokio.

Die Premierminister Indiens und Japans trafen sich in Tokio.

German-English
Supervised Translator
+
BeamSearch

English-German
Supervised Translator

Approach



	loss
Supervised German-English	$\log(P(\mathbf{y}\vert\mathbf{x};\overrightarrow{\mathbf{w}}))$
Supervised English-German	$\log(P(\mathbf{x}\vert\mathbf{y};\overleftarrow{\mathbf{w}}))$
Unsupervised German-German	$\lambda\log(P(\mathbf{x}'\vert\mathbf{x};\overrightarrow{\mathbf{w}},\overleftarrow{\mathbf{w}}))$

Same model

Same model

$$P(\mathbf{x}'|\mathbf{x};\overrightarrow{\mathbf{w}},\overleftarrow{\mathbf{w}}) = \sum_{\mathbf{y}} P(\mathbf{x}'|\mathbf{y};\overleftarrow{\mathbf{w}})P(\mathbf{y}|\mathbf{x};\overrightarrow{\mathbf{w}})$$

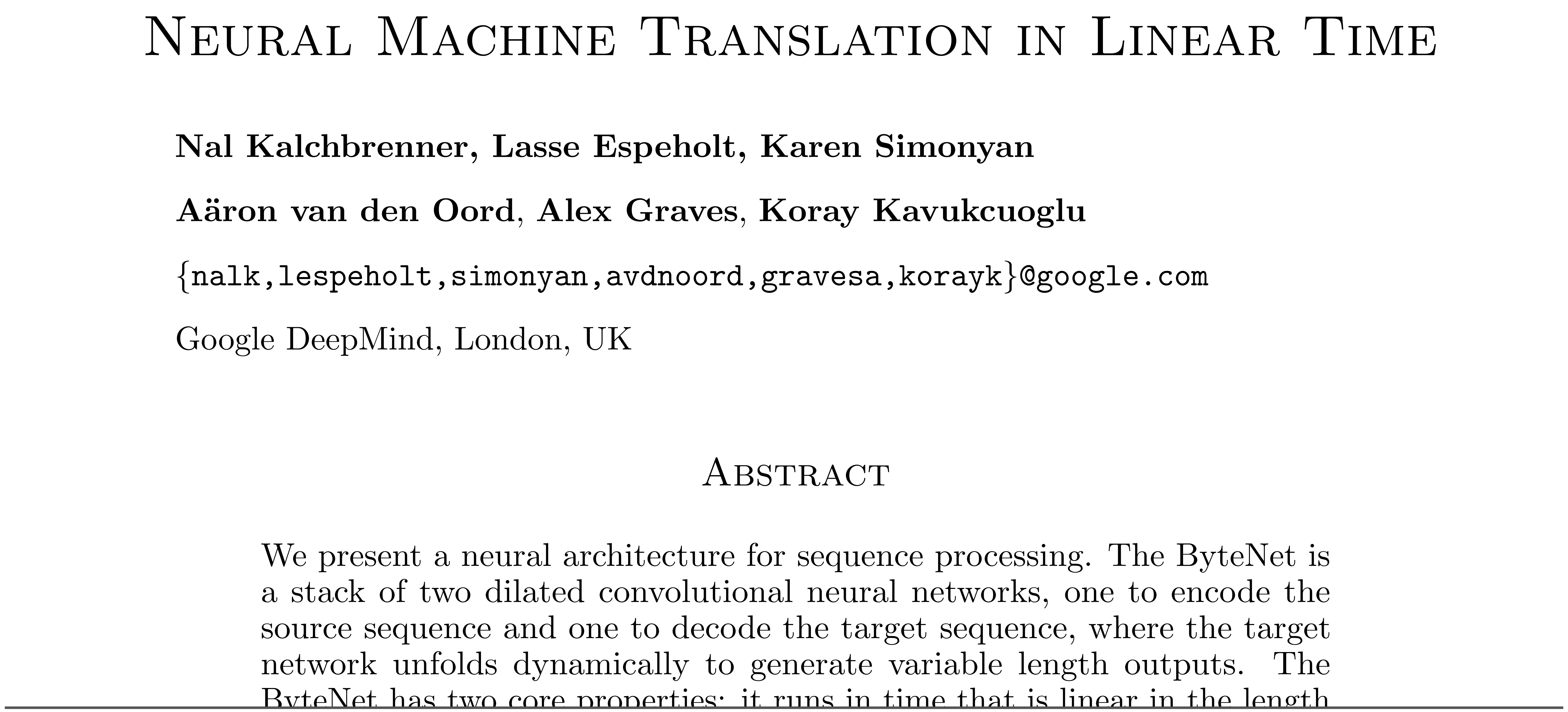
$$\mathcal{L} = -\big(\log(P(\mathbf{y}|\mathbf{x};\overrightarrow{\mathbf{w}})) + \log(P(\mathbf{x}|\mathbf{y};\overleftarrow{\mathbf{w}})) + \lambda\log(P(\mathbf{x}'|\mathbf{x};\overrightarrow{\mathbf{w}},\overleftarrow{\mathbf{w}}))\big)$$

Approach

	loss
Supervised German-English	$\log(P(\mathbf{y}\|\mathbf{x};\overrightarrow{\mathbf{w}}))$
Supervised English-German	$\log(P(\mathbf{x}\|\mathbf{y};\overleftarrow{\mathbf{w}}))$
Unsupervised German-German	$\lambda\log(P(\mathbf{x}'\|\mathbf{x};\overrightarrow{\mathbf{w}},\overleftarrow{\mathbf{w}}))$

- Model with fast training time.
- Model with fast inference, important as this is a part of the training too.

ByteNet



Neural Machine Translation in Linear Time

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan

Aäron van den Oord, **Alex Graves**, **Koray Kavukcuoglu**

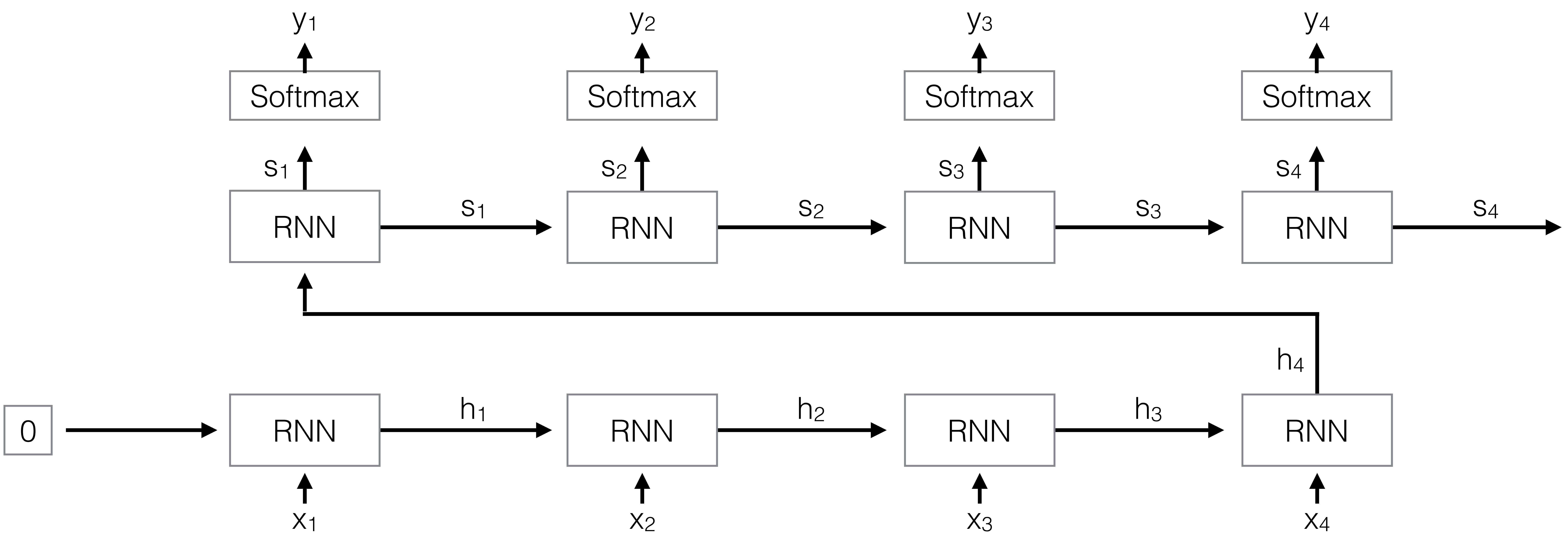
{nalk,lespeholt,simonyan,avdnoord,gravesa,korayk}@google.com

Google DeepMind, London, UK

Abstract

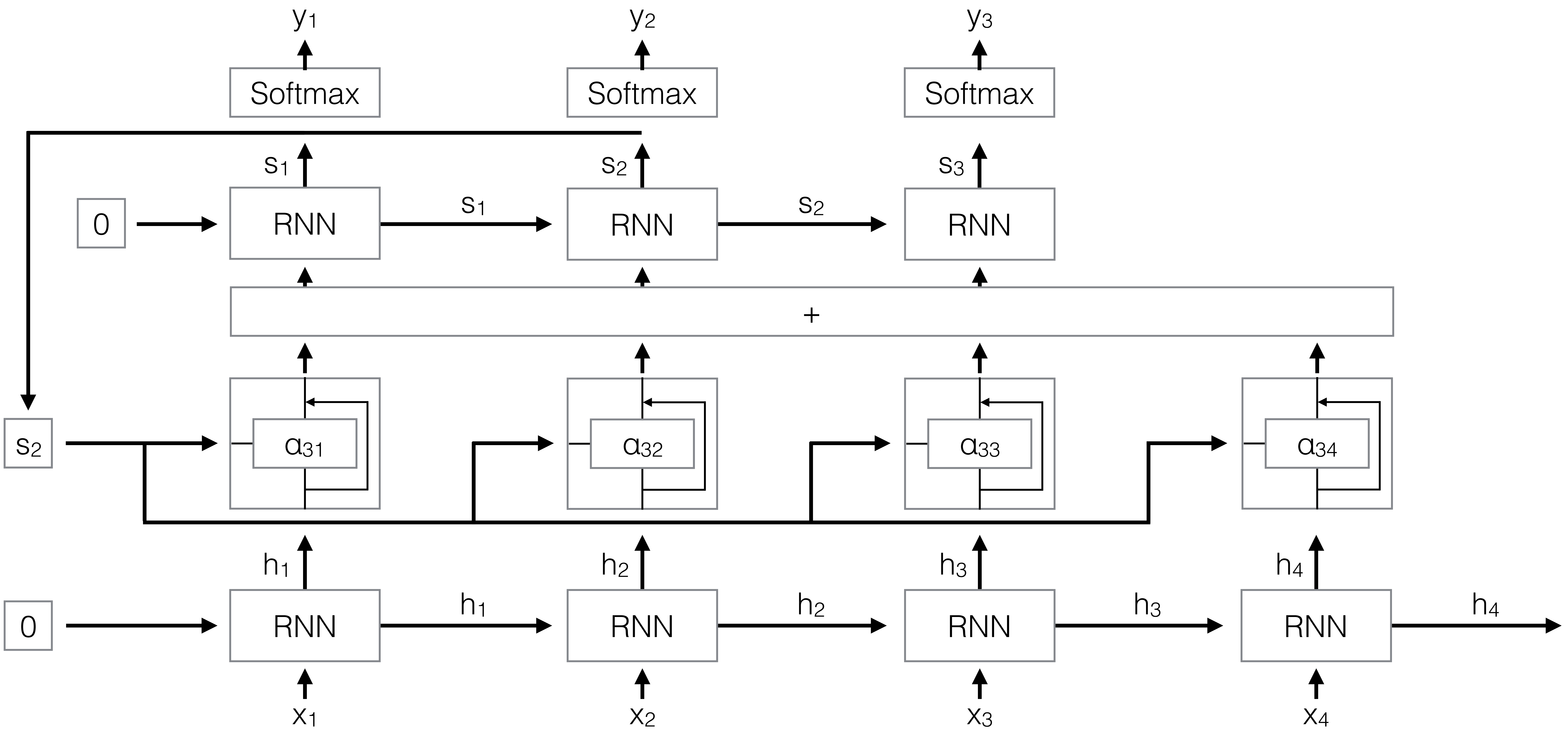
We present a neural architecture for sequence processing. The ByteNet is a stack of two dilated convolutional neural networks, one to encode the source sequence and one to decode the target sequence, where the target network unfolds dynamically to generate variable length outputs. The ByteNet has two core properties: it runs in time that is linear in the length

Sutskever



y_1
y_2
y_3
y_4
Softmax
Softmax
Softmax
Softmax
s_1
s_2
s_3
s_4
RNN
s_1
RNN
s_2
RNN
s_3
RNN
s_4
h_4
0
RNN
h_1
RNN
h_2
RNN
h_3
RNN
x_1
x_2
x_3
x_4

Bahdanau

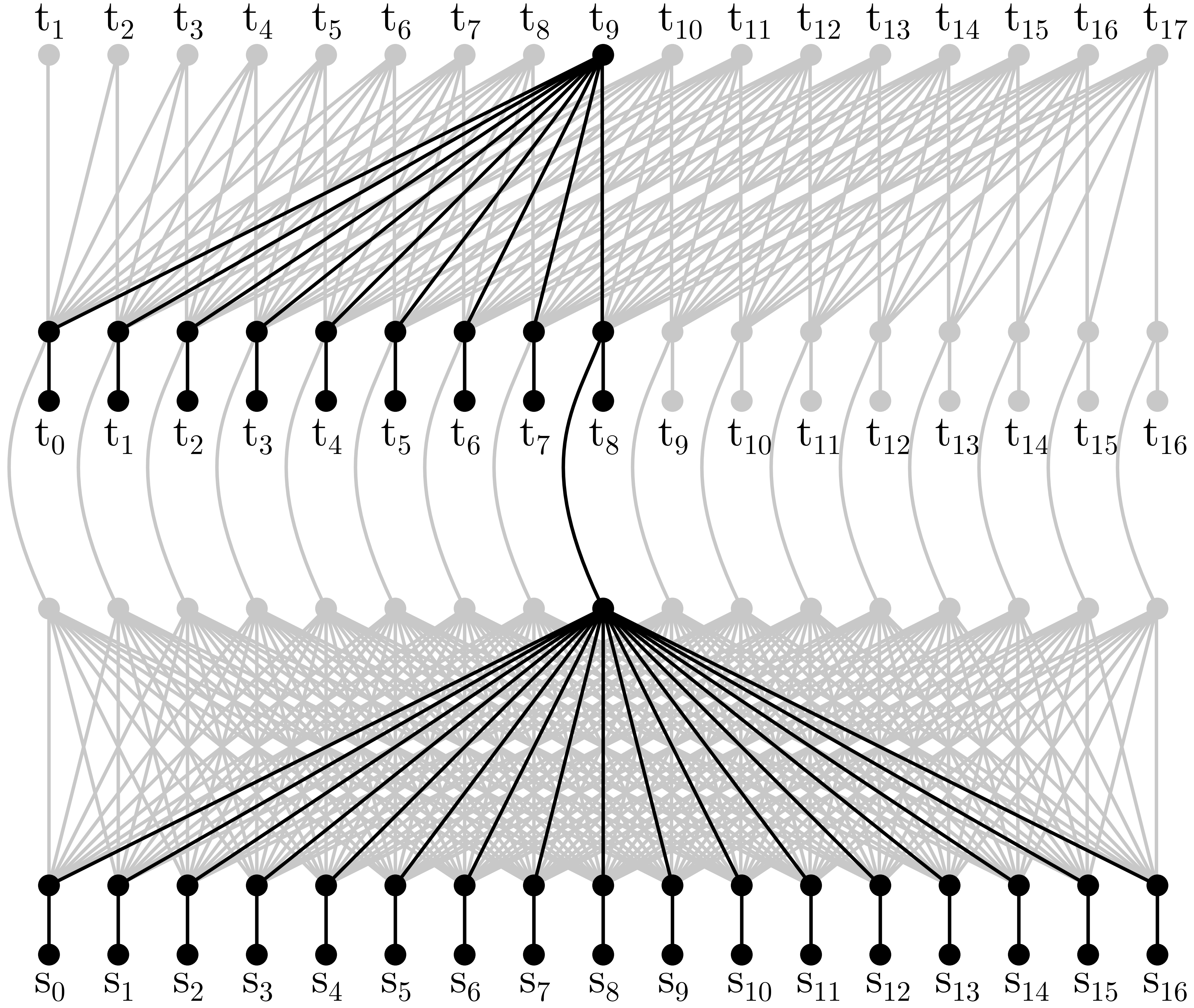


y1
y2
y3
Softmax
Softmax
Softmax
s1
s2
s3
0
RNN
s1
RNN
s2
RNN
+
s2
α31
α32
α33
α34
h1
h2
h3
h4
0
RNN
h1
RNN
h2
RNN
h3
RNN
h4
x1
x2
x3
x4

ByteNet

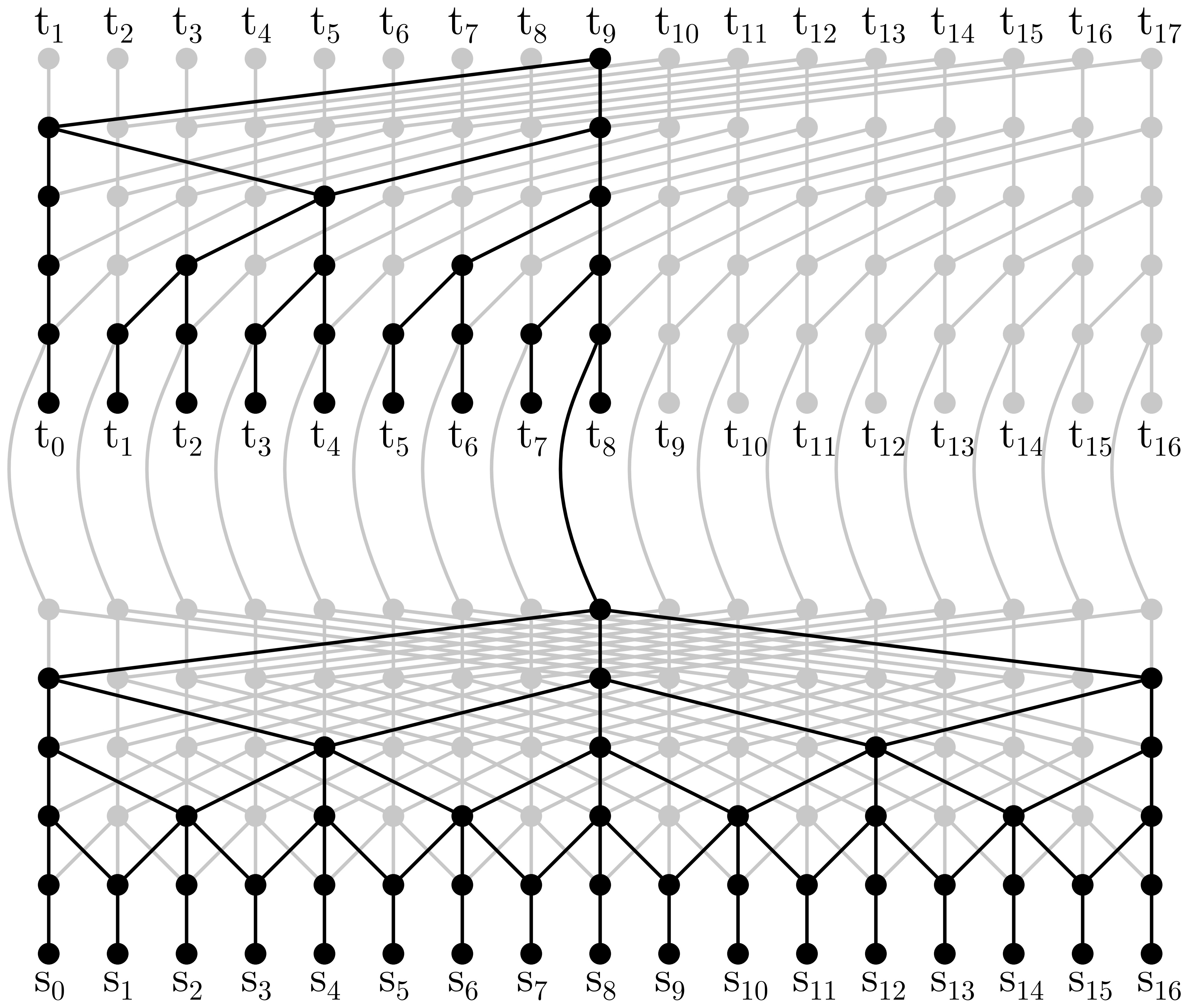
Desirables	Sutskever	Bahdanau	ByteNet
Running time should be linear. This is critical when using a character input.	×		×
Parallelizing over both source and target sequences is necessary for fast learning.			×
The source representation should be linear with source sequence length. Called *Resolution Persistent.*		×	×
The path between source and target should be constant.			×

ByteNet


t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16} t_{17}
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16}
s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11} s_{12} s_{13} s_{14} s_{15} s_{16}

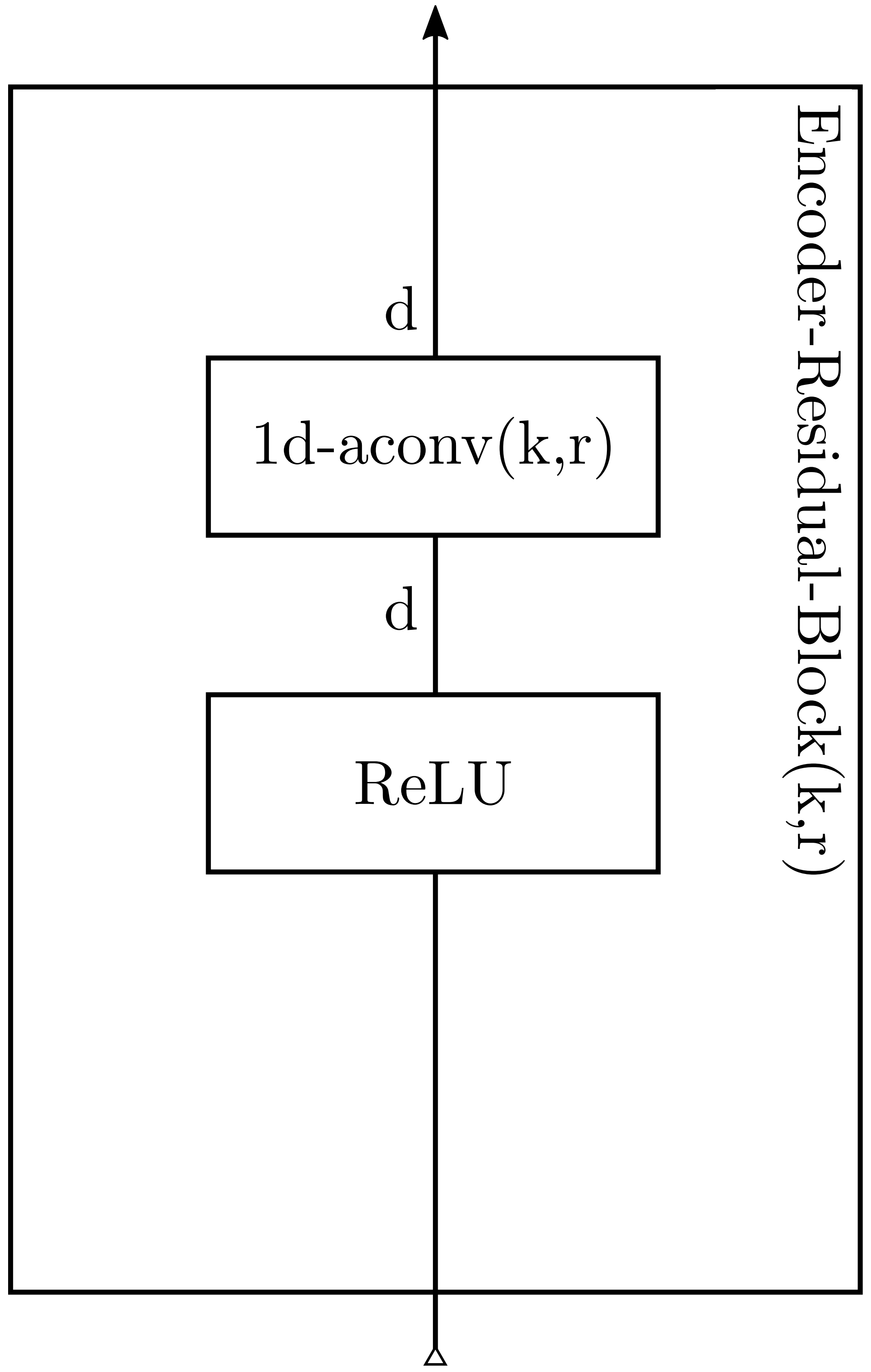
- A convolution approach runs in linear time and is easy to parallelize.
- Non-local connections are often found in text.
- Very wide convolutions are needed for no-local connections. That is very expensive.

ByteNet


t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16} t_{17}
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16}
s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11} s_{12} s_{13} s_{14} s_{15} s_{16}

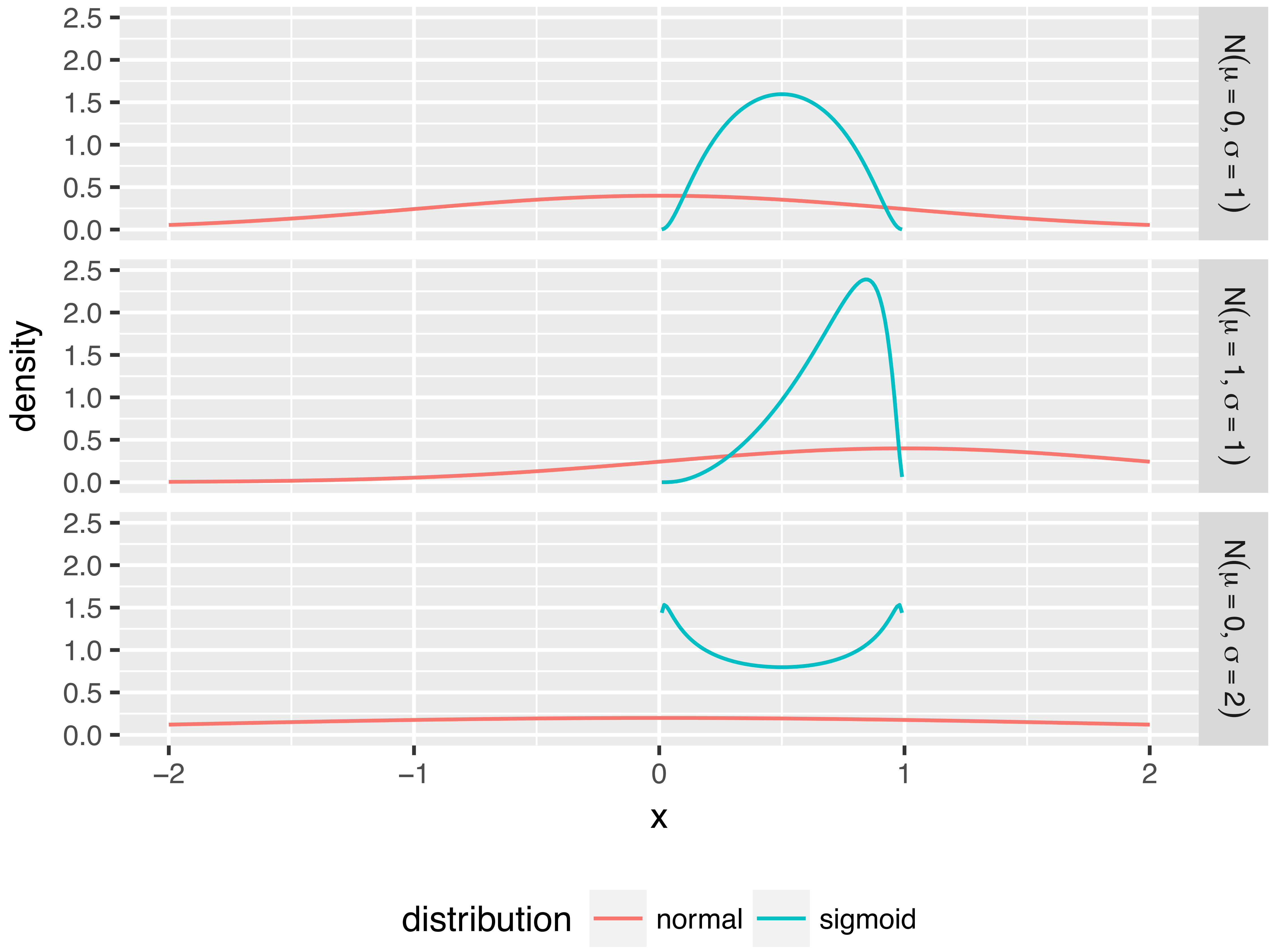
- A convolution approach runs in linear time and is easy to parallelize.
- Hierical Dilated Convolution gives an exponential width for a linear cost.

ByteNet



- Each “•” is more than just a dilated convolution.
- First step, is to add A ReLU activation function.
- The first layer is an embedding, thus ReLU before the convolution is reasonable.

Batch Normalization

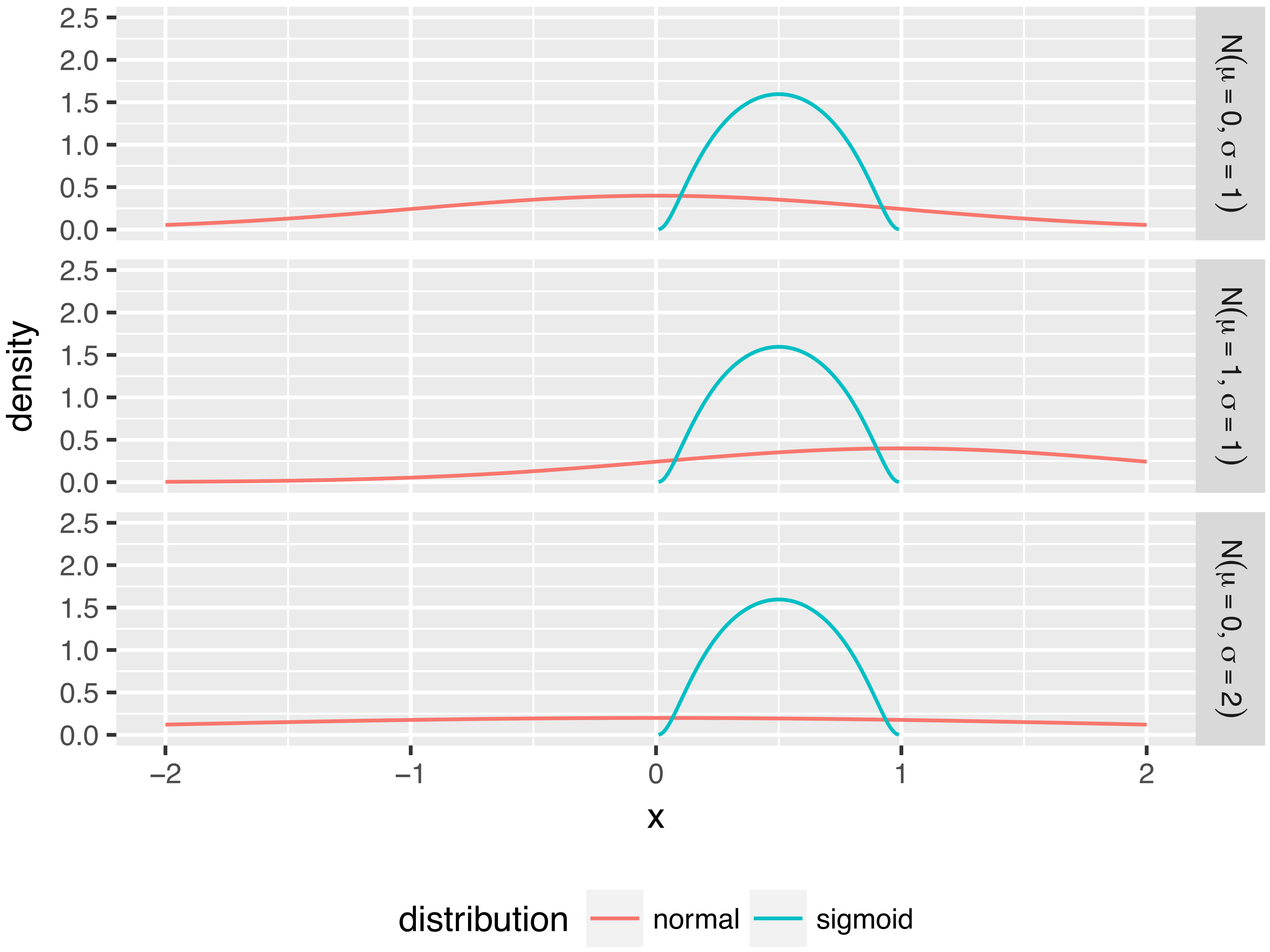


- Changes in the parameters in combination with the activation function, causes an *internal covariate shift*.

$$z_{h_\ell}^{(i)} = \sum_{h_{\ell-1}}^{H_{\ell-1}} w_{h_{\ell-1},h_\ell} a_{h_\ell -1}^{(i)}$$

$$a_{h_\ell}^{(i)} = \theta\left(z_{h_\ell}^{(i)}\right)$$

Batch Normalization



- Solution is to normalize the input to the activation function.

$$z^{(i)}_{h_\ell} = \sum_{h_{\ell-1}}^{H_{\ell-1}} w_{h_{\ell-1},h_\ell} a^{(i)}_{h_\ell - 1}$$

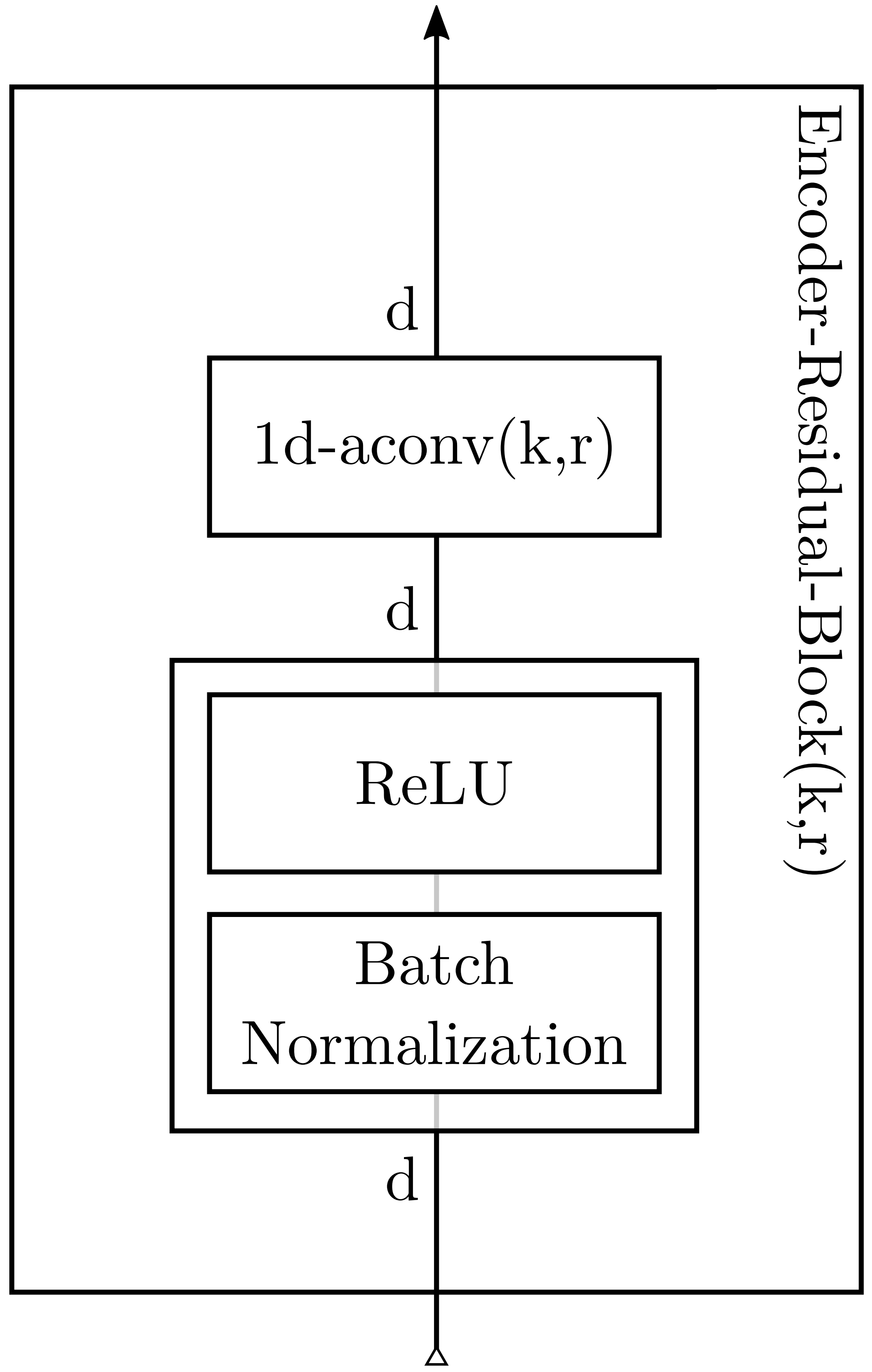
$$\hat{z}^{(i)}_{h_\ell} = \gamma_{h_\ell} \frac{z^{(i)}_{h_\ell} - \mu^{\mathcal{B}}_{h_\ell}}{\sqrt{\sigma^{2,\mathcal{B}}_{h_\ell} + \epsilon}} + \beta_{h_\ell}$$

$$a^{(i)}_{h_\ell} = \theta\left(\hat{z}^{(i)}_{h_\ell}\right)$$

$$\mu^{\mathcal{B}}_{h_\ell} = \frac{1}{n} \sum_{i=1}^{n} z^{(i)}_{h_\ell}$$

$$\sigma^{2,\mathcal{B}}_{h_\ell} = \frac{1}{n} \sum_{i=1}^{n} (z^{(i)}_{h_\ell} - \mu^{\mathcal{B}}_{h_\ell})^2$$

ByteNet



- Each “•” is more than just a dilated convolution.
- Add Batch Normalization before ReLU activation.

Residual Connection

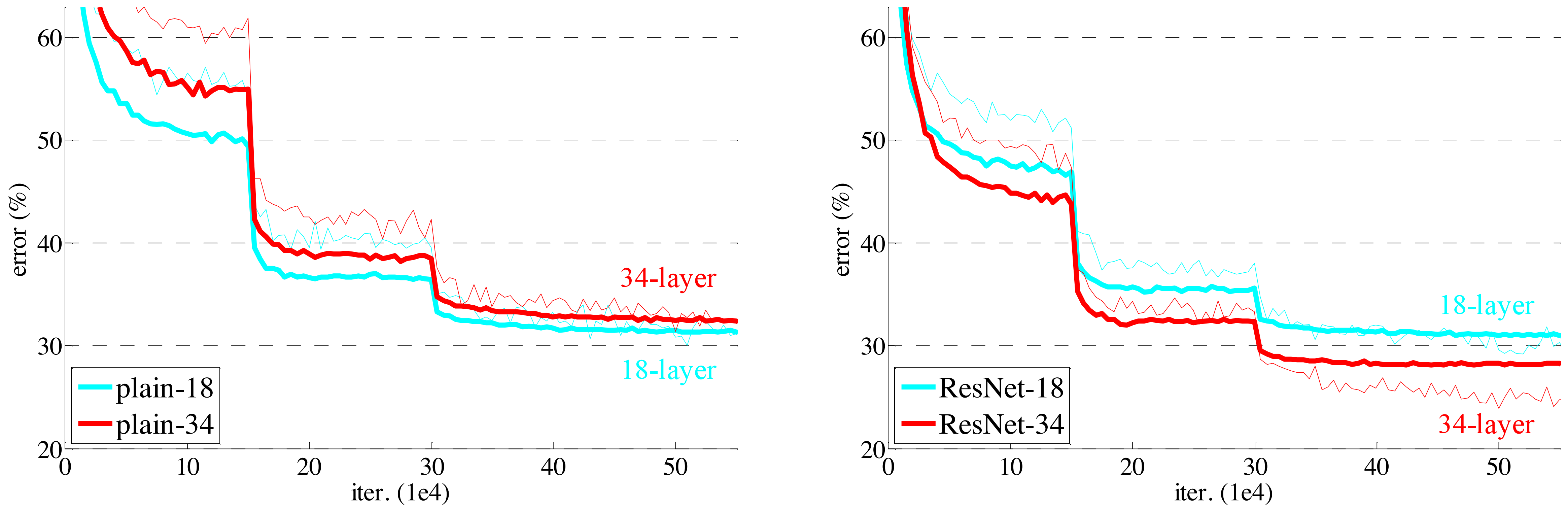
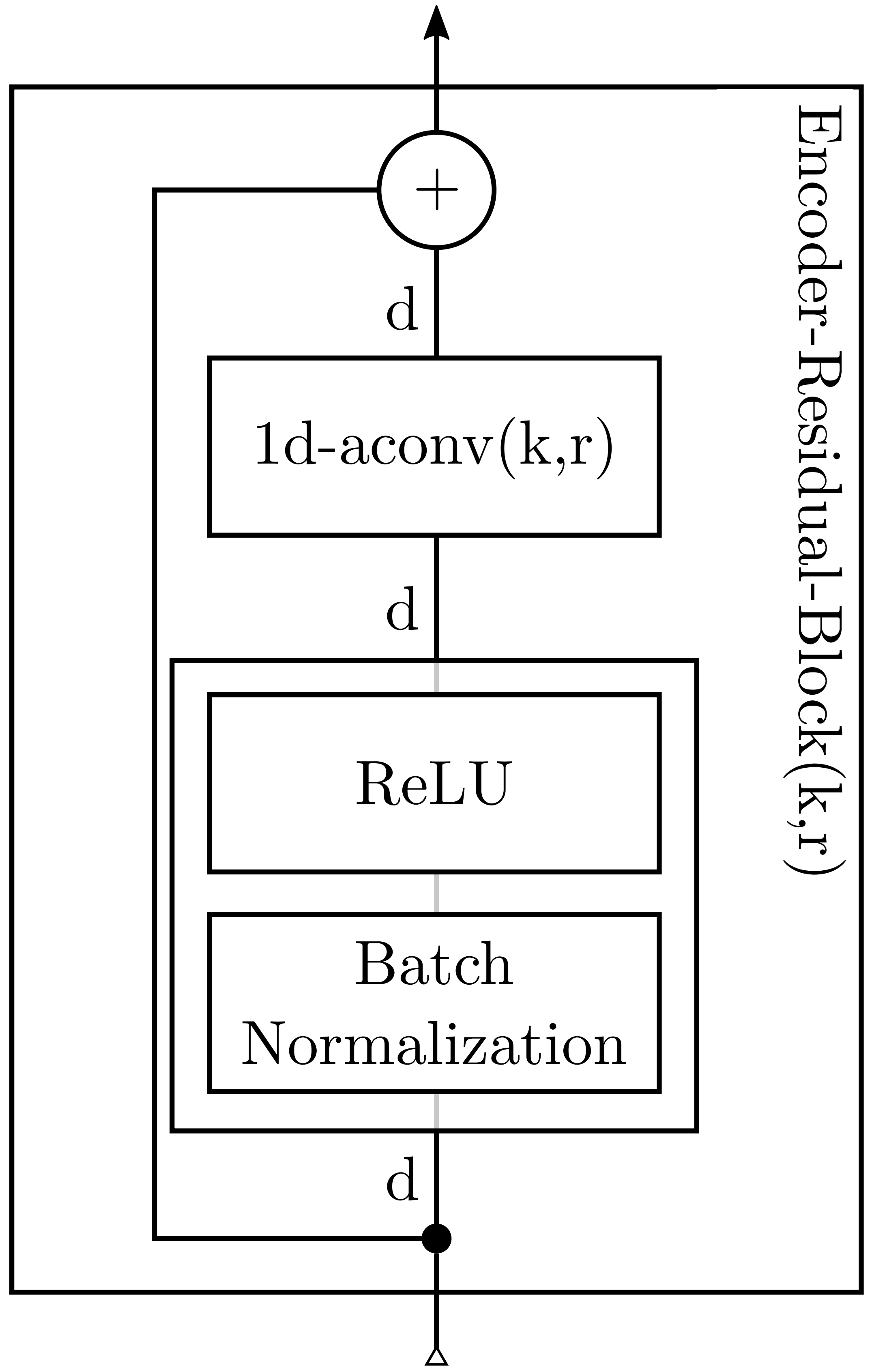


Figure 4. Training on ImageNet. Thin curves denote training error, and bold curves denote validation error of the center crops.

Source: Deep Residual Learning for Image Recognition

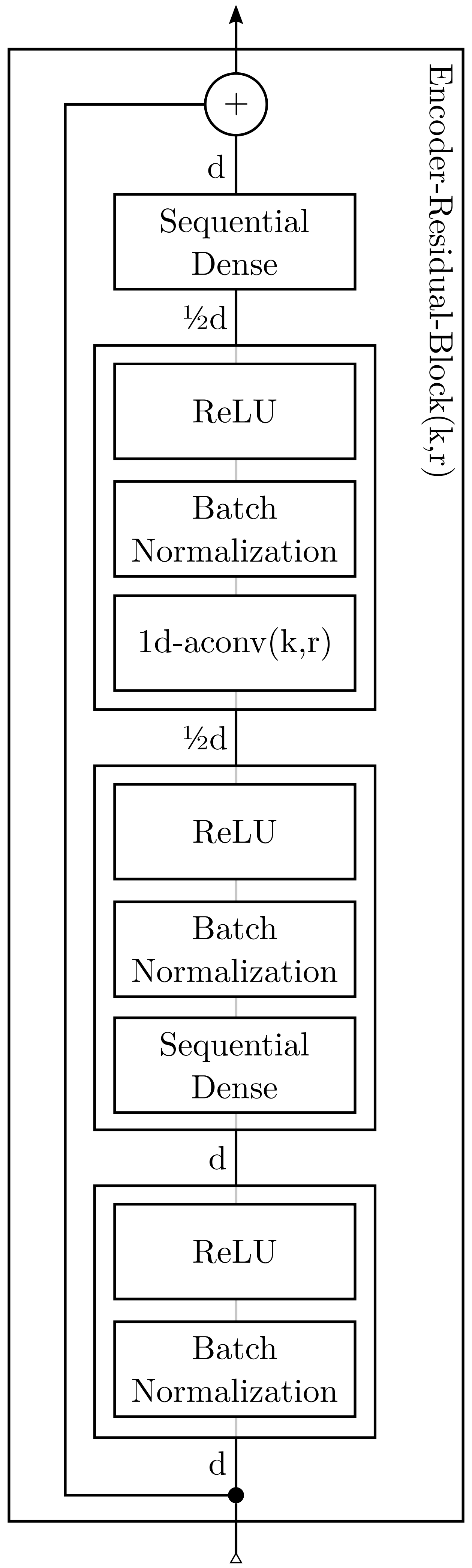
- Too many layers can cause bad training error.
- Idea: $\mathcal{F}(\mathbf{x}) = 0$ is easier to solve than $\mathcal{H}(\mathbf{x}) = \mathbf{x}$, especially with ReLU activation.
- To make the identity function easily obtainable use $\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x}$.

ByteNet



- Each “•” is more than just a dilated convolution.
- Add residual connection for for the entire block.

ByteNet



- Each “•” is more than just a dilated convolution.
- Add compression and decompression layer.
- Weights with no compression:

$$k \cdot d^2$$

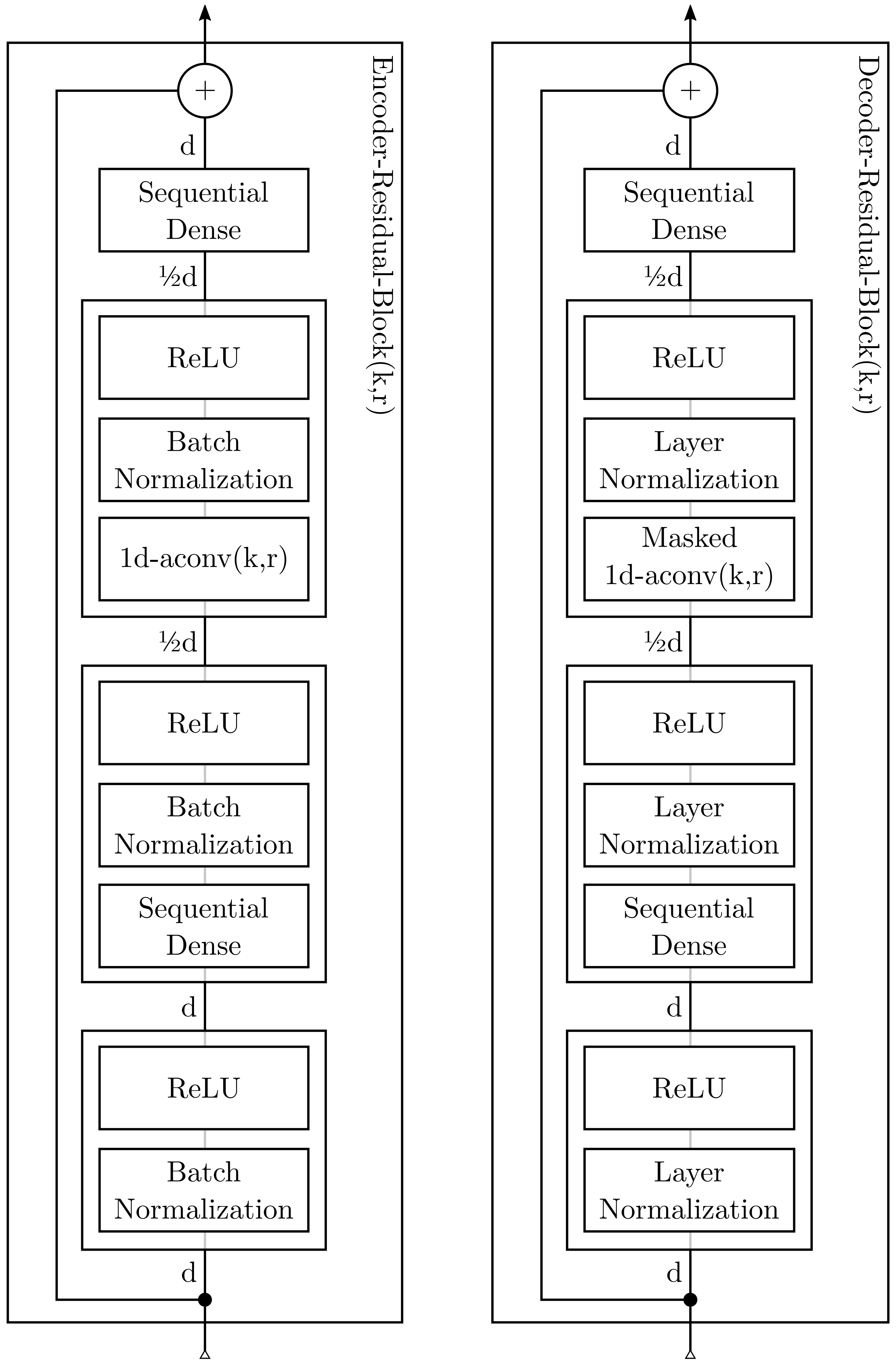
- Weights with compression:

$$k \cdot \left(\frac{d}{2}\right)^2 + \frac{1}{2}d^2 + \frac{1}{2}d^2$$

- Reduction:

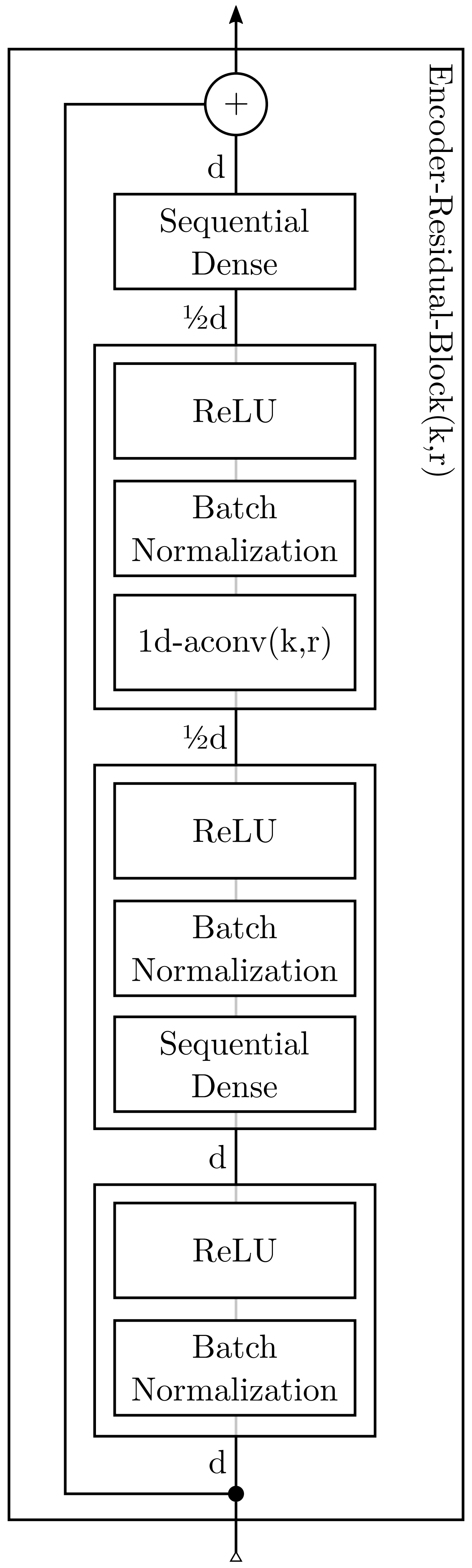
$$\frac{k+4}{4k} \xrightarrow{k=5} 45\%$$

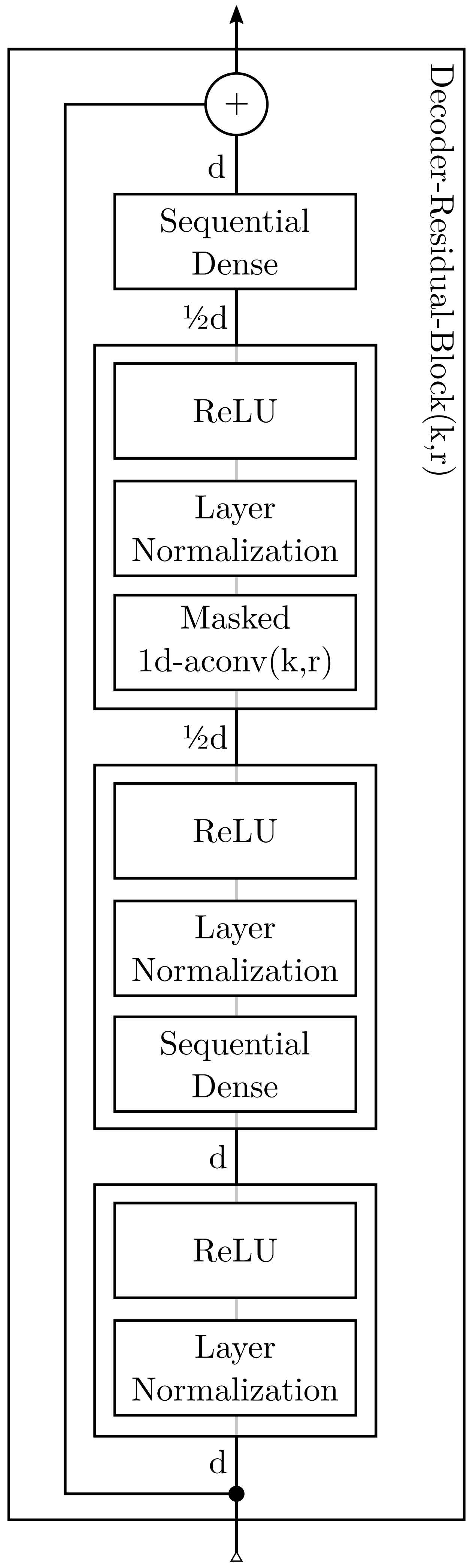
ByteNet

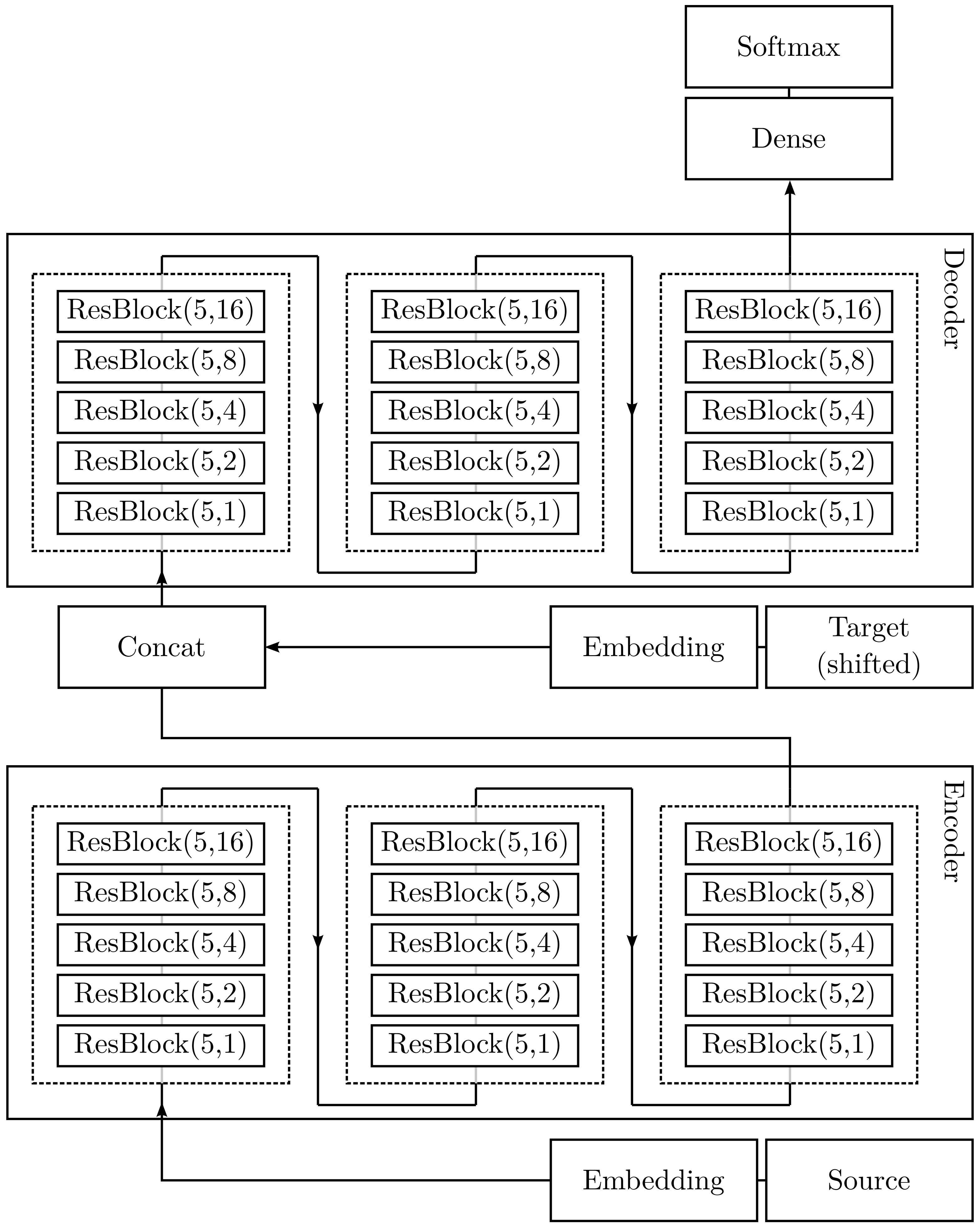


- Small changes in the decoder, such it doesn't get information from the future
  - Use Layer Normalization.
  - Use masked dilated convolution.

ByteNet


+
d
Sequential Dense
½d
ReLU
Batch Normalization
1d-aconv(k,r)
½d
ReLU
Batch Normalization
Sequential Dense
d
ReLU
Batch Normalization
d
Encoder-Residual-Block(k,r)


+
d
Sequential Dense
½d
ReLU
Layer Normalization
Masked 1d-aconv(k,r)
½d
ReLU
Layer Normalization
Sequential Dense
d
ReLU
Layer Normalization
d
Decoder-Residual-Block(k,r)


Softmax
Dense
Decoder
ResBlock(5,16)
ResBlock(5,8)
ResBlock(5,4)
ResBlock(5,2)
ResBlock(5,1)
ResBlock(5,16)
ResBlock(5,8)
ResBlock(5,4)
ResBlock(5,2)
ResBlock(5,1)
ResBlock(5,16)
ResBlock(5,8)
ResBlock(5,4)
ResBlock(5,2)
ResBlock(5,1)
Concat
Embedding
Target (shifted)
Encoder
ResBlock(5,16)
ResBlock(5,8)
ResBlock(5,4)
ResBlock(5,2)
ResBlock(5,1)
ResBlock(5,16)
ResBlock(5,8)
ResBlock(5,4)
ResBlock(5,2)
ResBlock(5,1)
ResBlock(5,16)
ResBlock(5,8)
ResBlock(5,4)
ResBlock(5,2)
ResBlock(5,1)
Embedding
Source

Experiments

Synthetic Digits

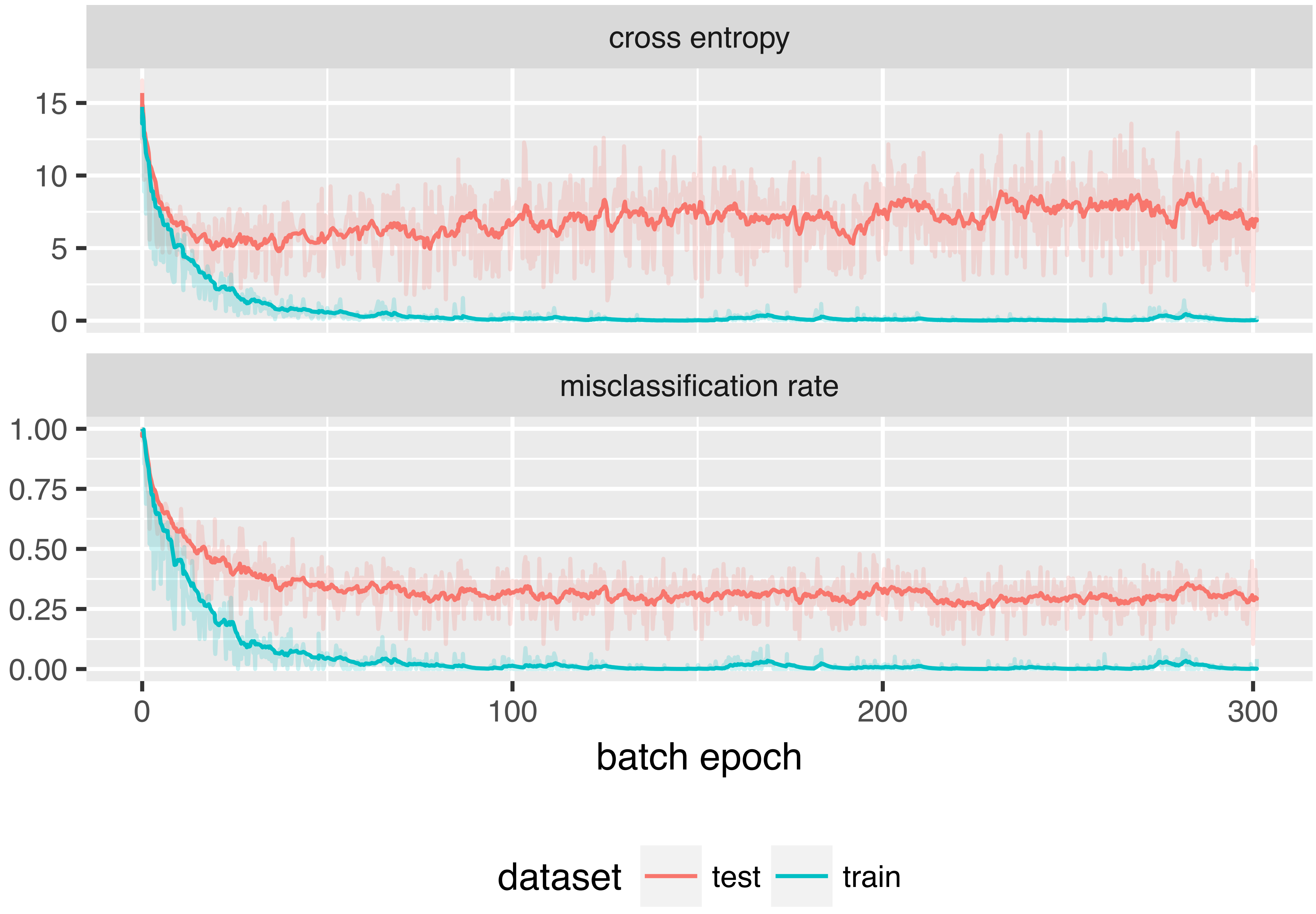
Source	Target
zero two	02
four eight	48
eight four four	844

WMT NewsTest Memorization

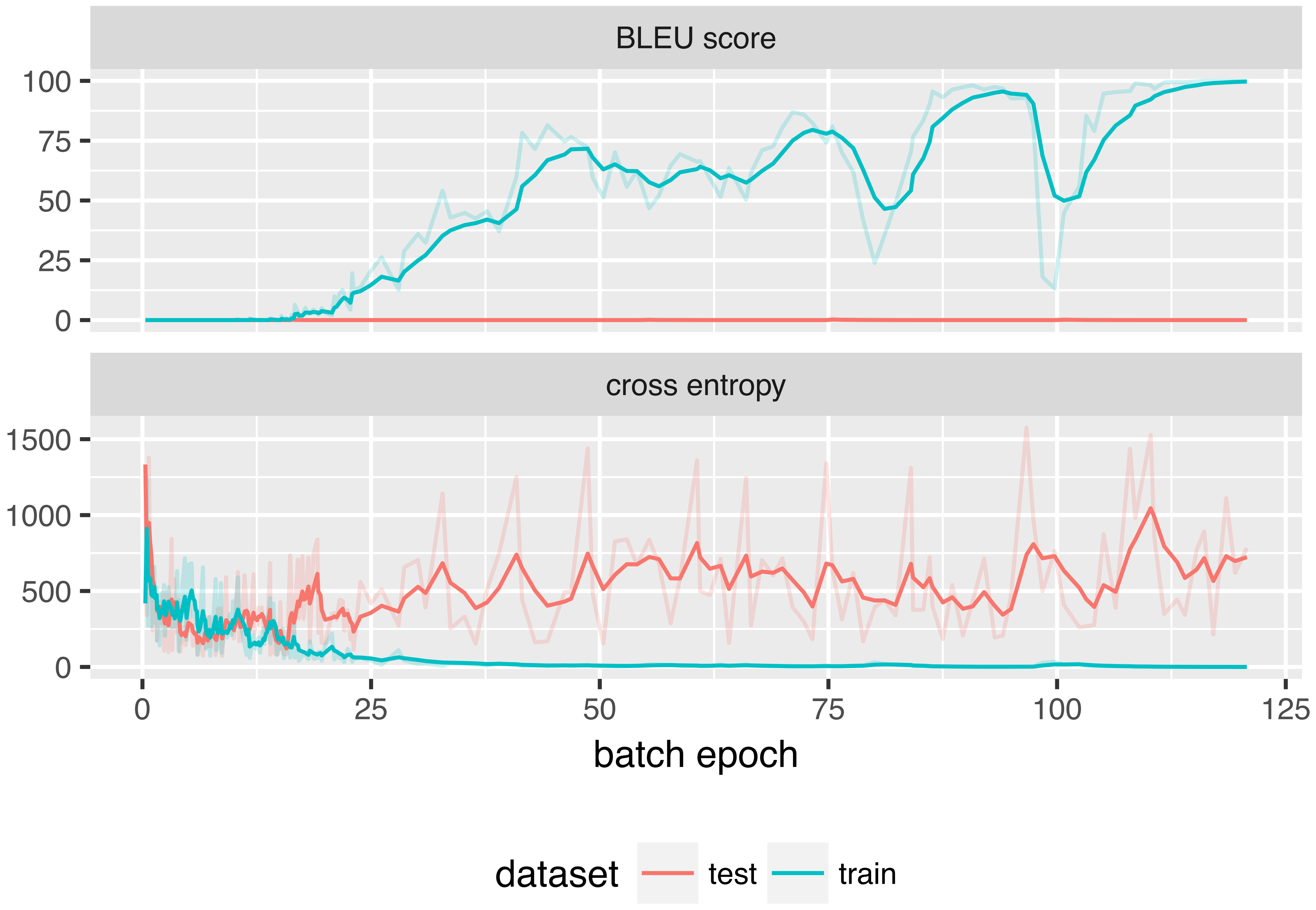
Source	Target
Die Premierminister Indiens und Japans trafen sich in Tokio.	India and Japan prime ministers meet in Tokyo
Pläne für eine stärkere kerntechnische Zusammenarbeit stehen ganz oben auf der Tagesordnung.	High on the agenda are plans for greater nuclear co-operation.

Experiments

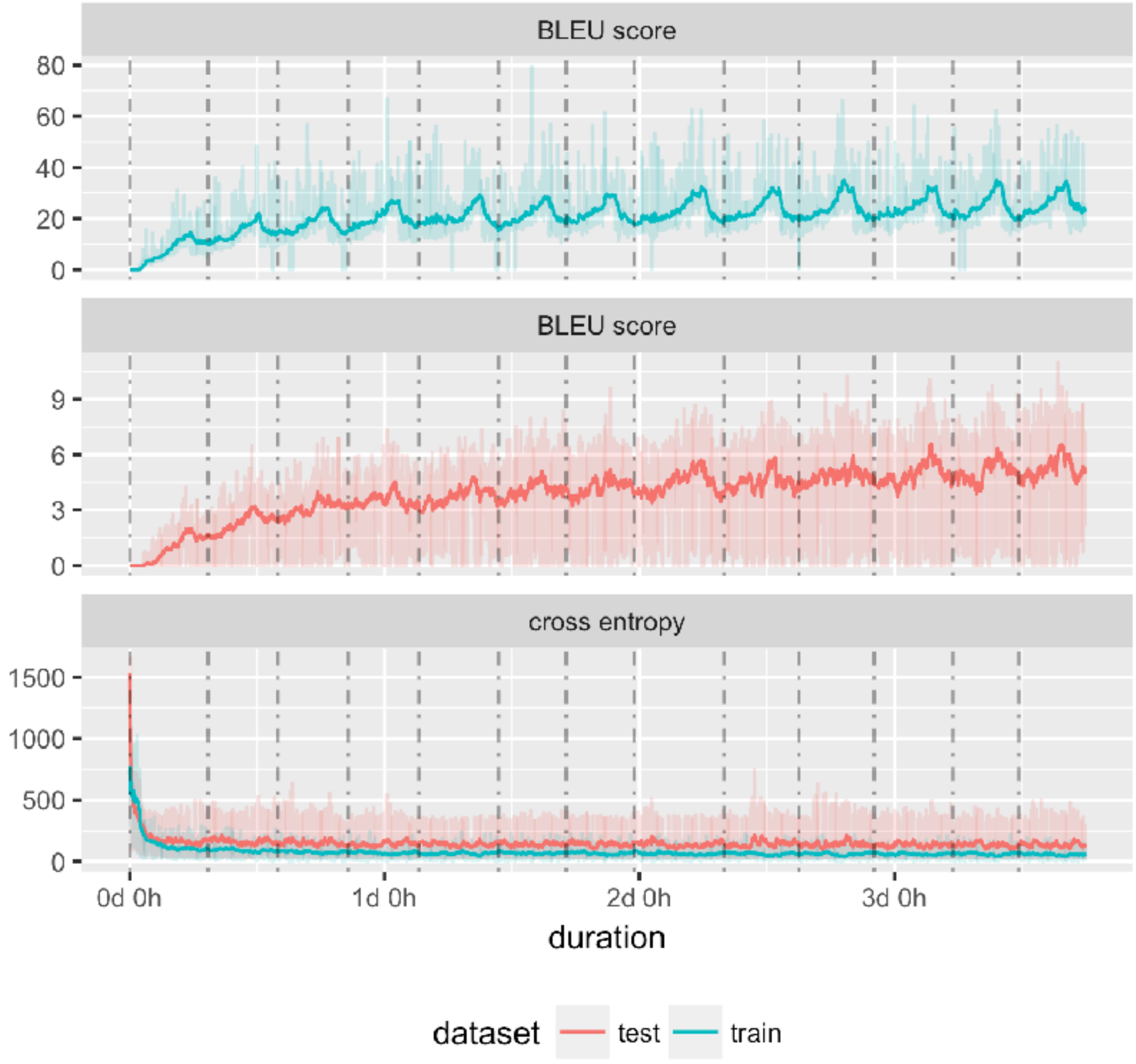
Synthetic Digits



WMT NewsTest Memorization

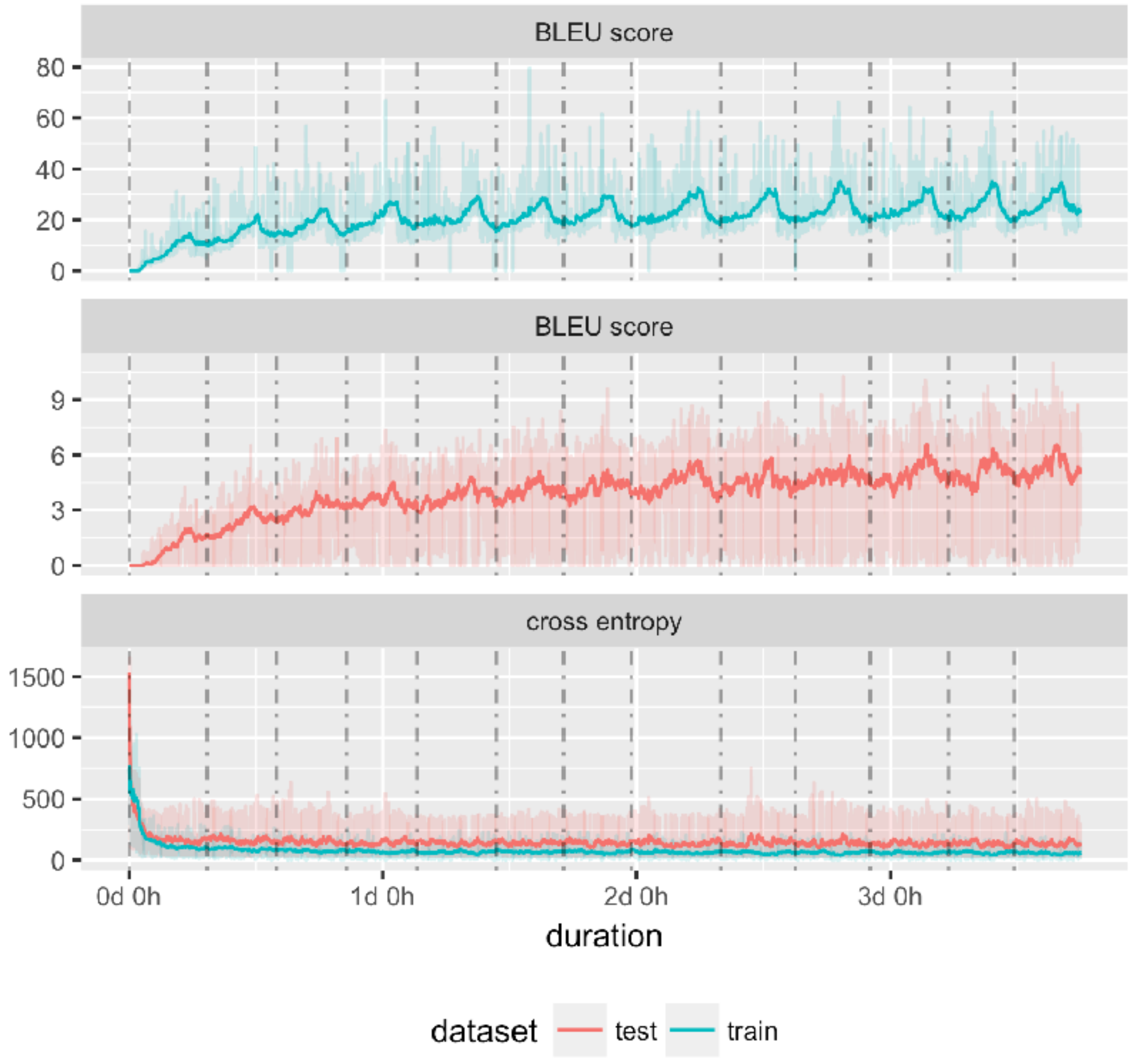


Experiments



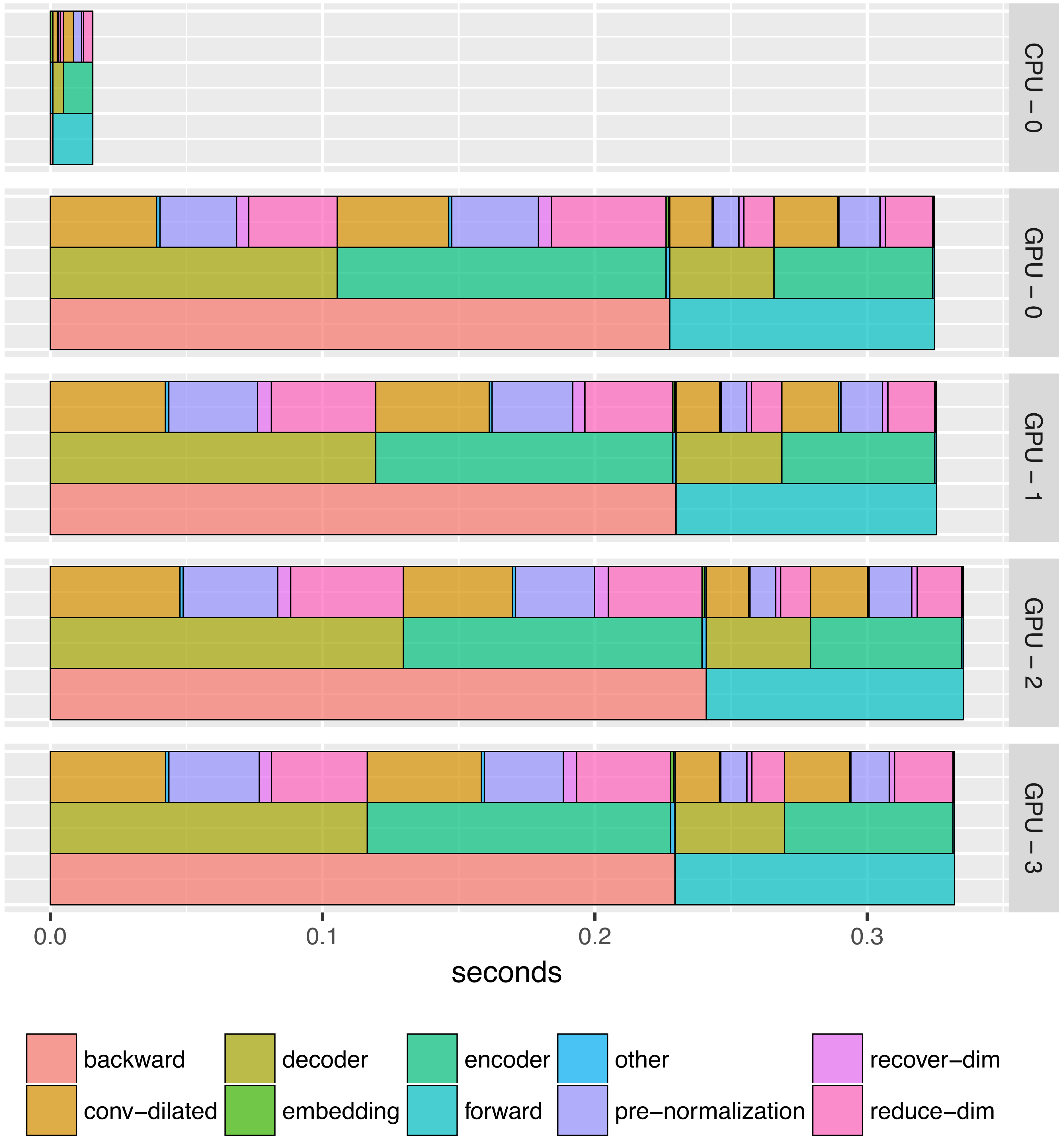
Source	Die formelle Anerkennung der Arbeitsrechte als Menschenrechte - und die Erweiterung des Schutzes der Bürgerrechte als Schutz gegen Diskriminierung bei der Schaffung von Arbeitnehmervertre- tungen - ist längst überfällig.
Target	The formal recognition of labor rights as human rights - and the extension of civil rights protections to prevent discrimination against labor organizing - is long overdue.
translation	The formal recognition of human rights as human rights - and the extension of the protection of civil liberties as a protection against discrimination against employment organizations - is long overdue.
BLEU	45.97

Experiments



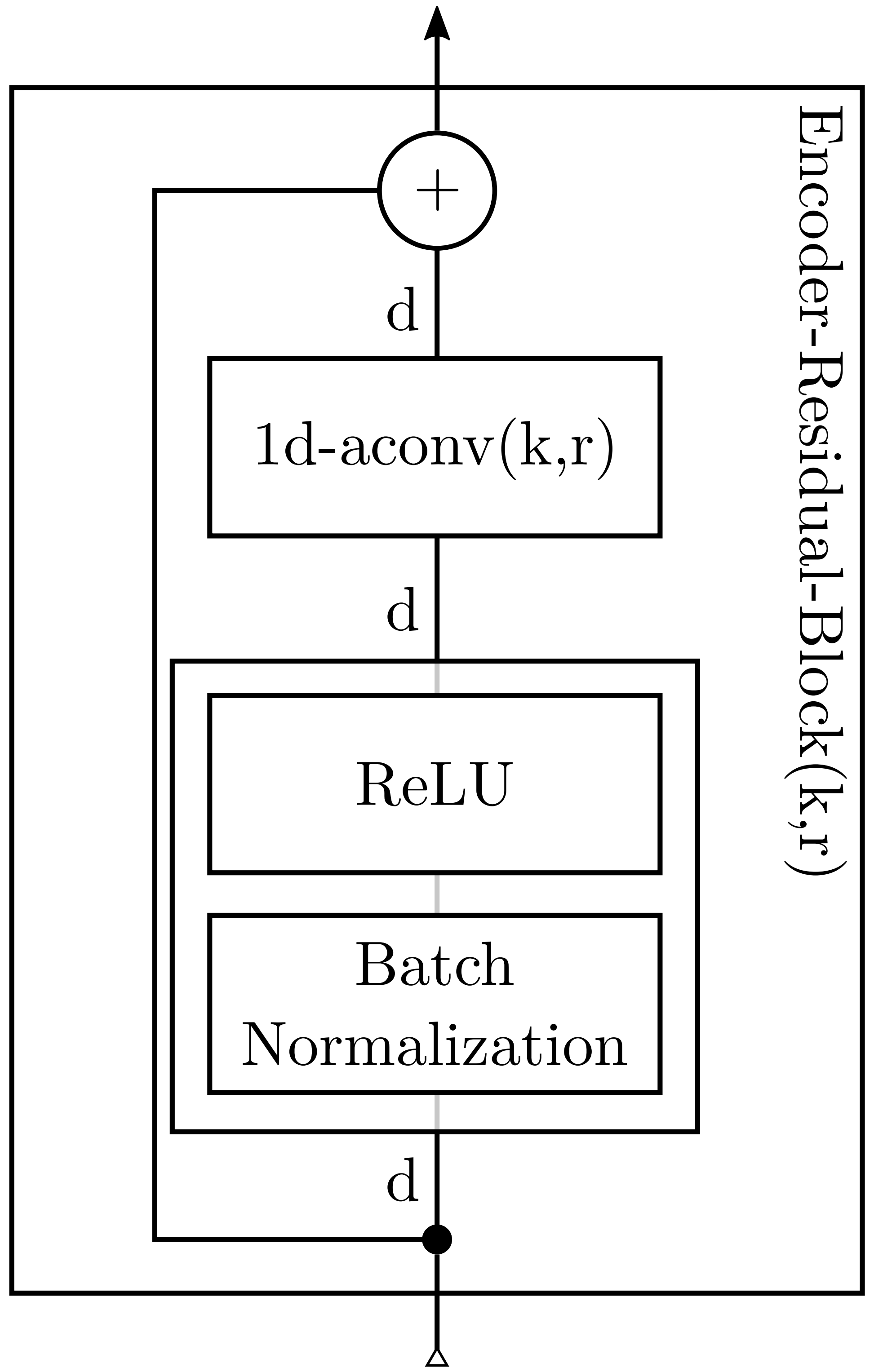
Source	”Diese Krankheit wird am besten von gynäkologischen Onkologen behandelt, und diese sind meistens in größeren Städten zu finden,” sagte sie.
Target	”This disease is best treated by gynaecological oncology surgeons and they’re mostly based in major cities,” she said.
translation	’This disease is best achieved by organic chocolate industries, and these are indeed more than ”more cities’.
BLEU	0.00

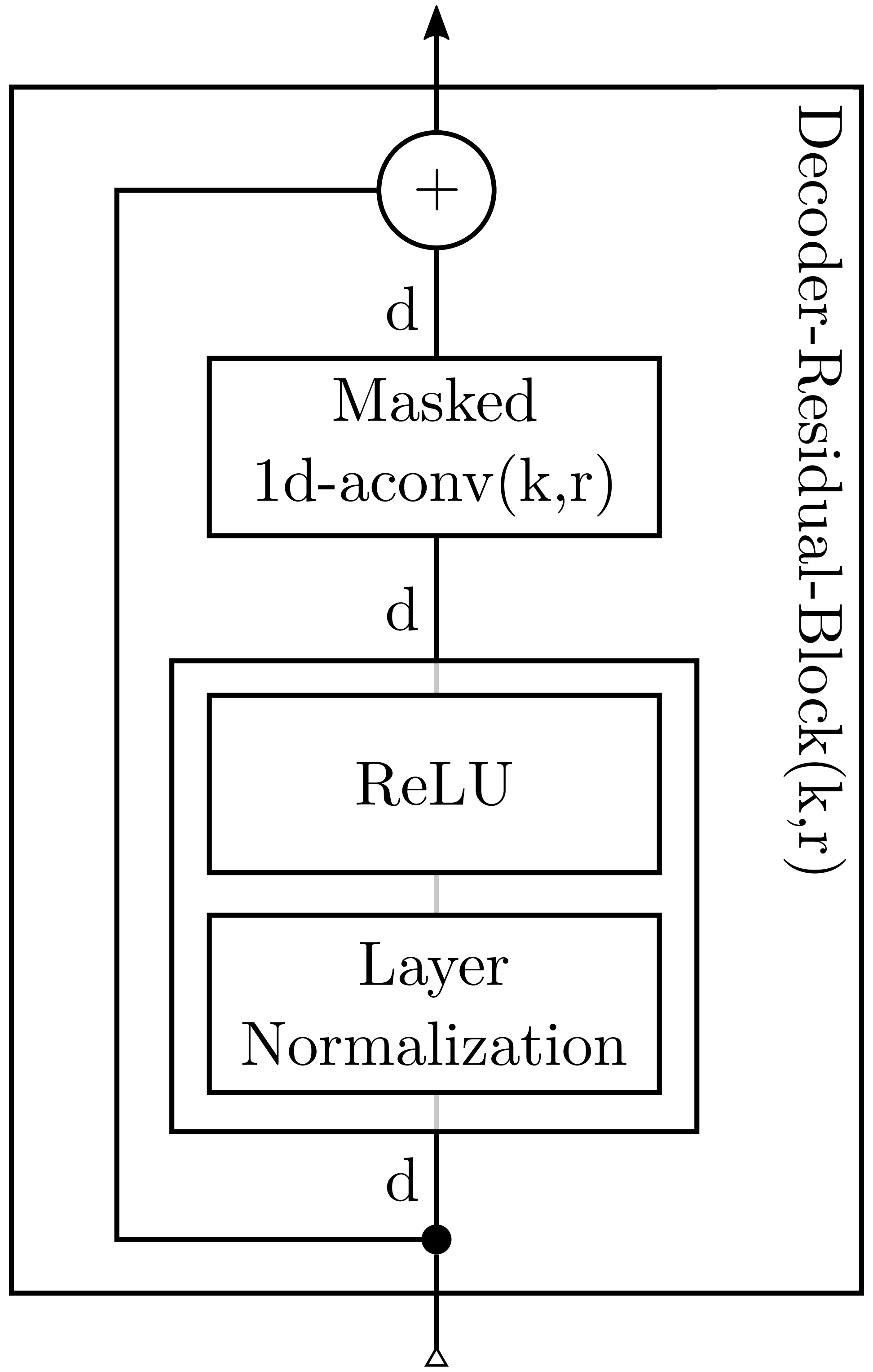
Experiments



- Training time for ByteNet is too slow.
- This is a 4x reduced model, the full model will be even worse.
- Semi-Supervised learning will take even longer time.
- Profiling indicates that many small operations and normalization is very time consuming.
- 37% of the time is spend on data-transfer.

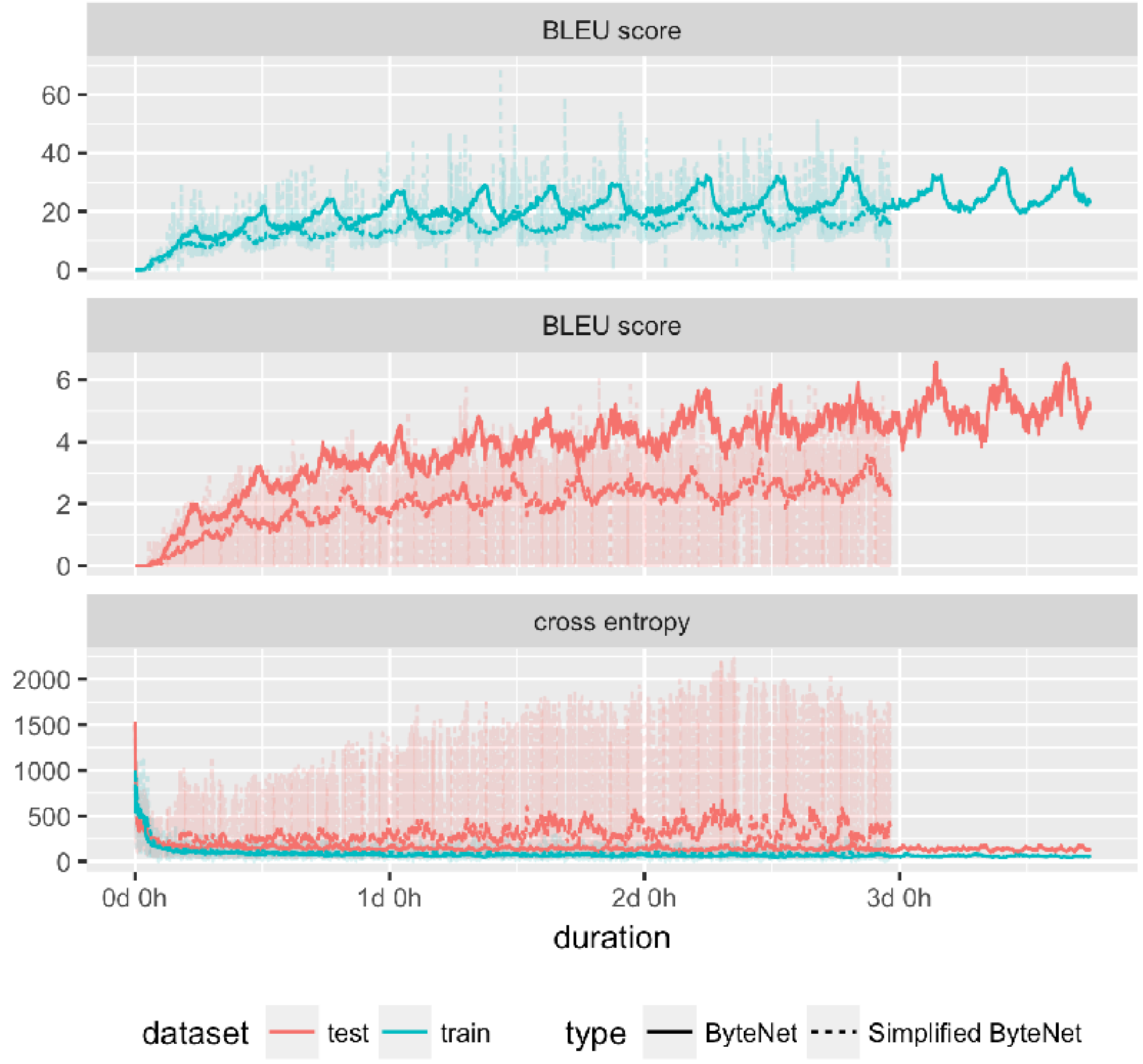
Simplified ByteNet





- Remove compression and decompression layer.
- Reduce dimensionality (d) from 400 to 200.
- Has 5/9 times fewer weights, has 1/3 normalization layers.
- The BLEU score won't be as good, but the model will train faster.

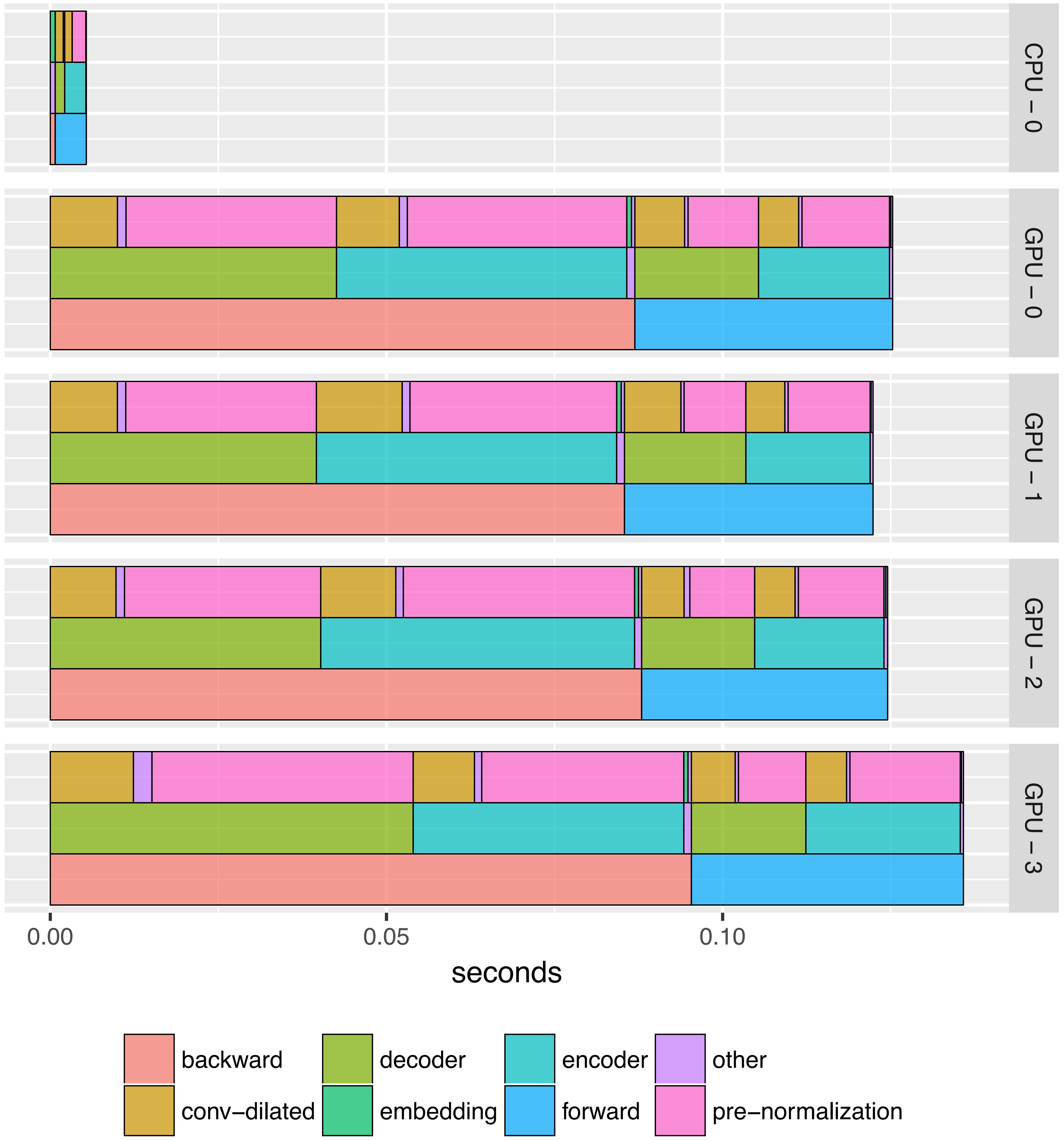
Experiments



Source	Die Premierminister Indiens und Japans trafen sich in Tokio.
Target	India and Japan prime ministers meet in Tokyo
translation	Mr Pieper and the independence of Singapore in London.
BLEU	0.00

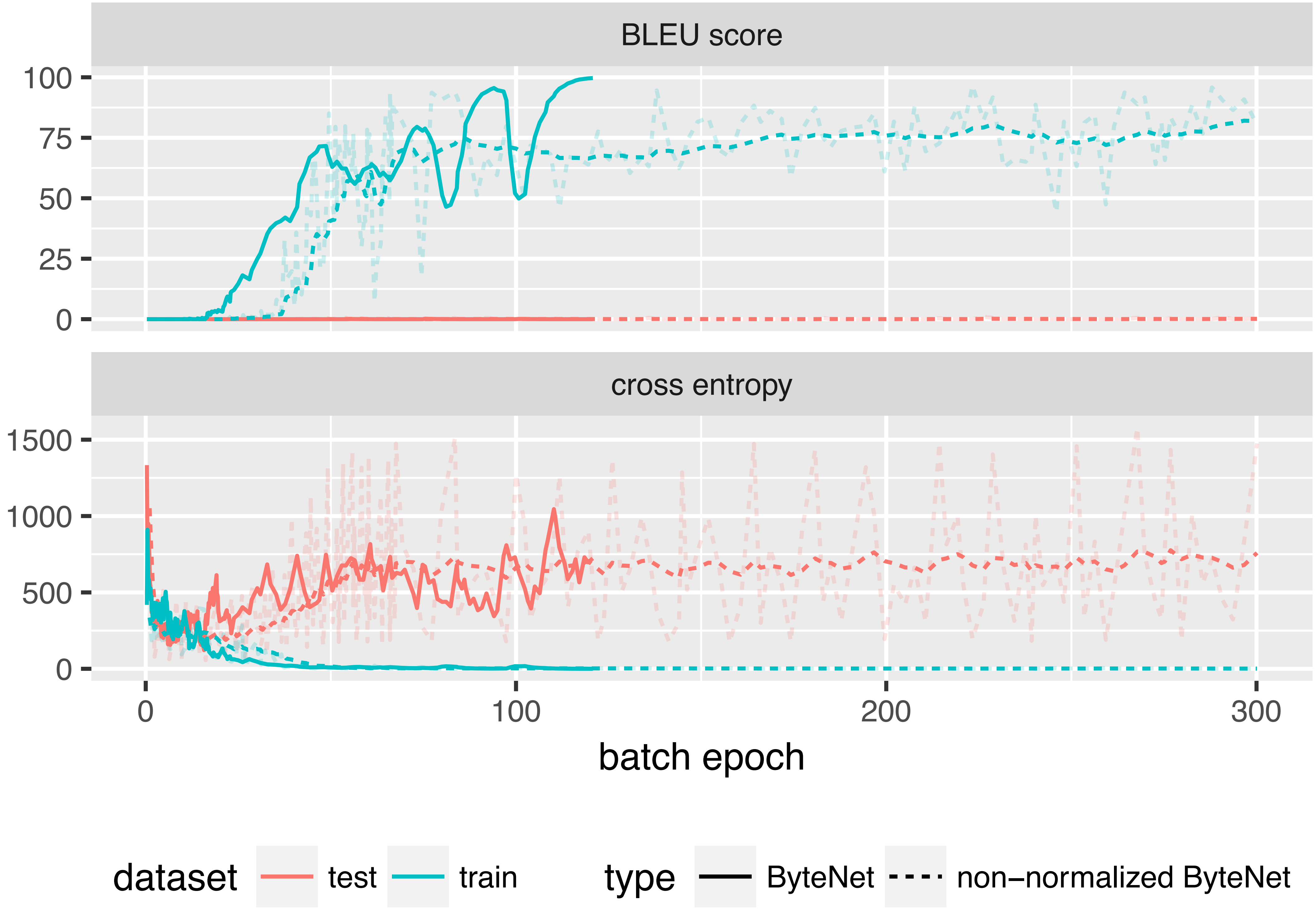
- Only marginally faster iterations.
- Translations are no longer meaningful.
- In terms of time, it learns slower thus there is no value in this approach.

Experiments



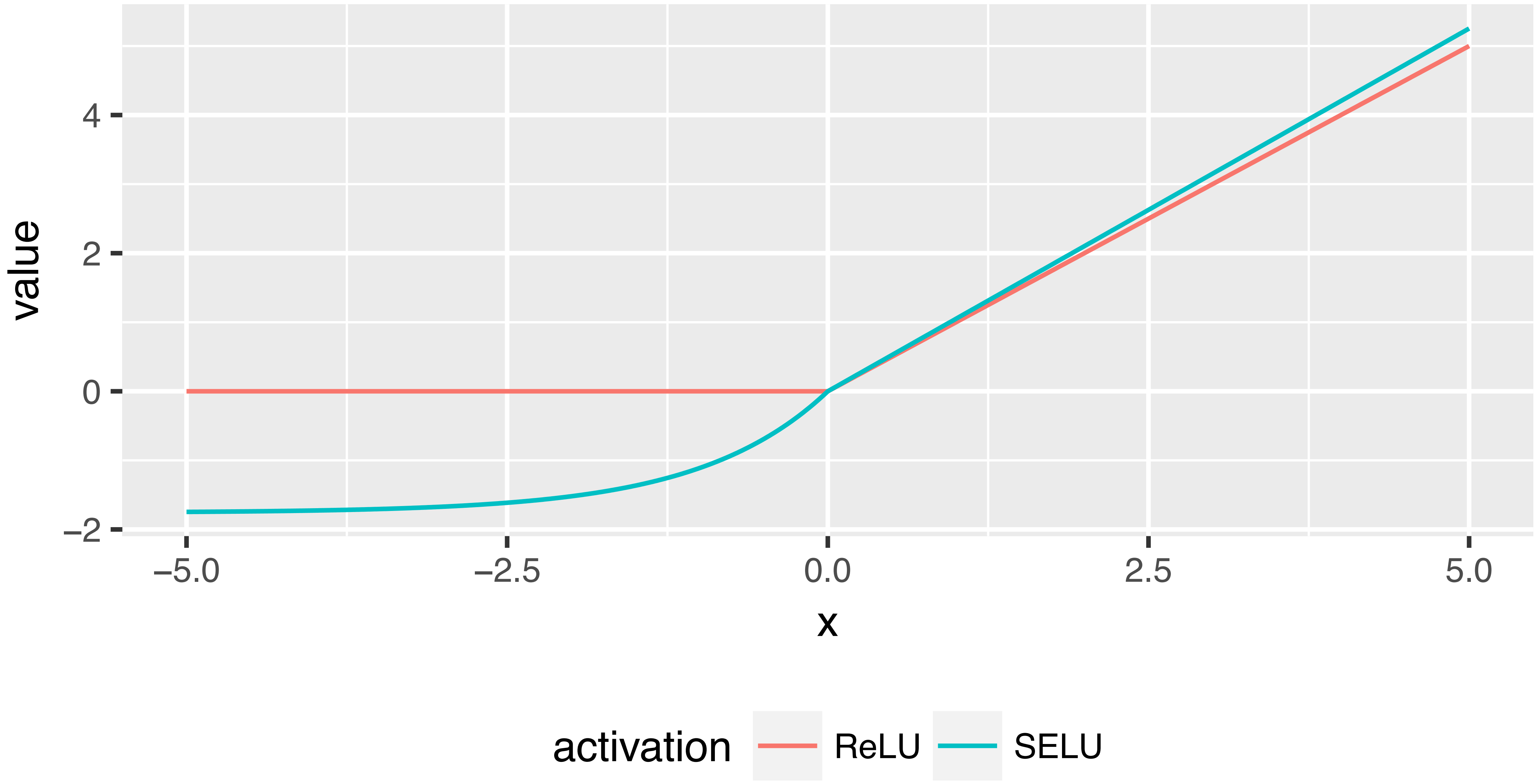
- Again, normalization is the most time consuming factor.
- Dilated convolution takes almost no time and this isn't implemented efficiently.

NoNorm. ByteNet



- Without normalization the ByteNet model converges very slowly.
- Likely a vanishing gradient problem.
- How can the normalization be removed, but convergence speed maintained.

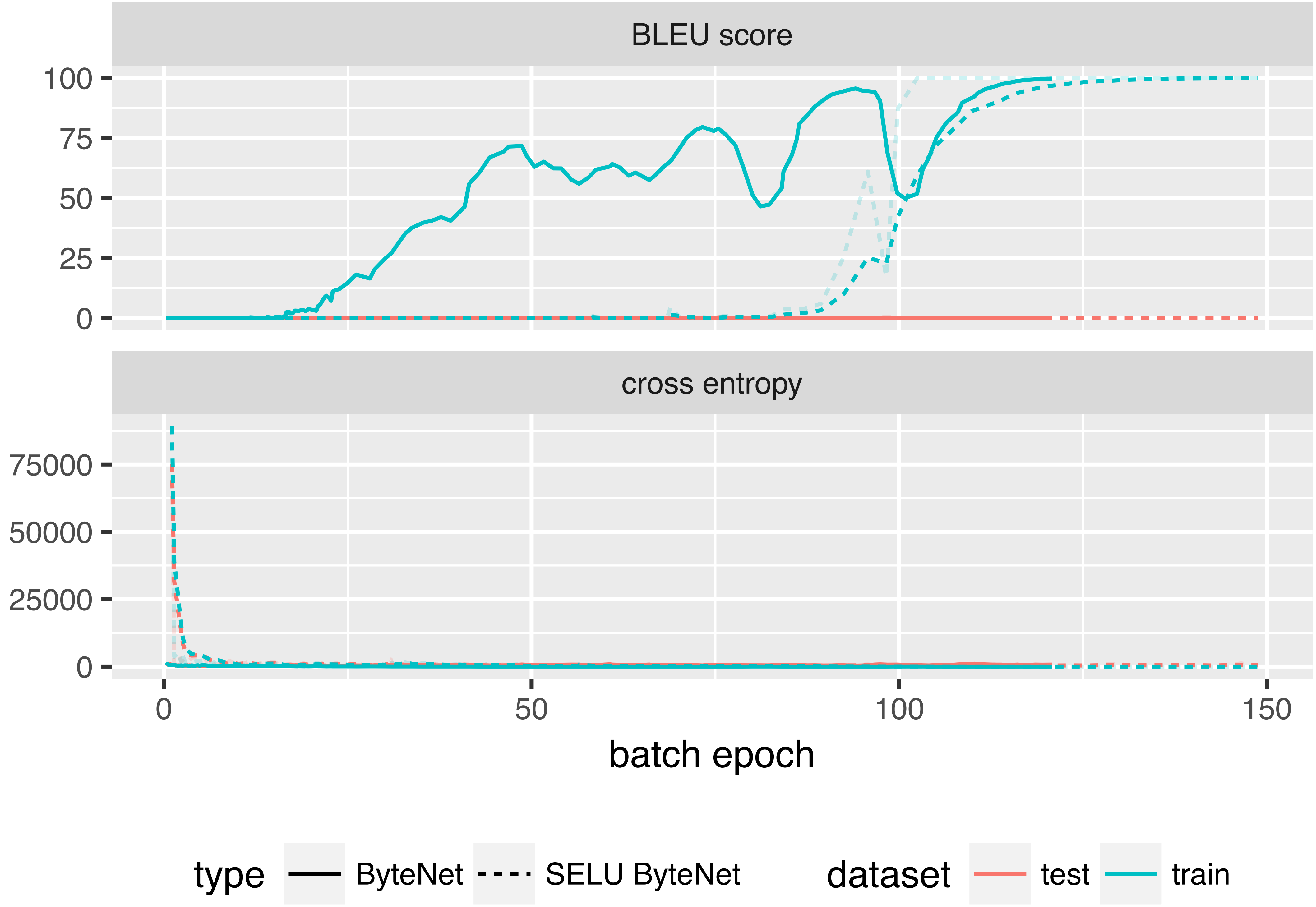
SELU ByteNet



$$\mathrm{SELU}(x) = \lambda \begin{cases} x & x > 0 \\ \alpha(\exp(x) - 1) & x \leq 0 \end{cases}, \quad \text{where:} \quad \begin{array}{l} \alpha = 1.6732632423 \\ \lambda = 1.0507009873 \end{array}$$

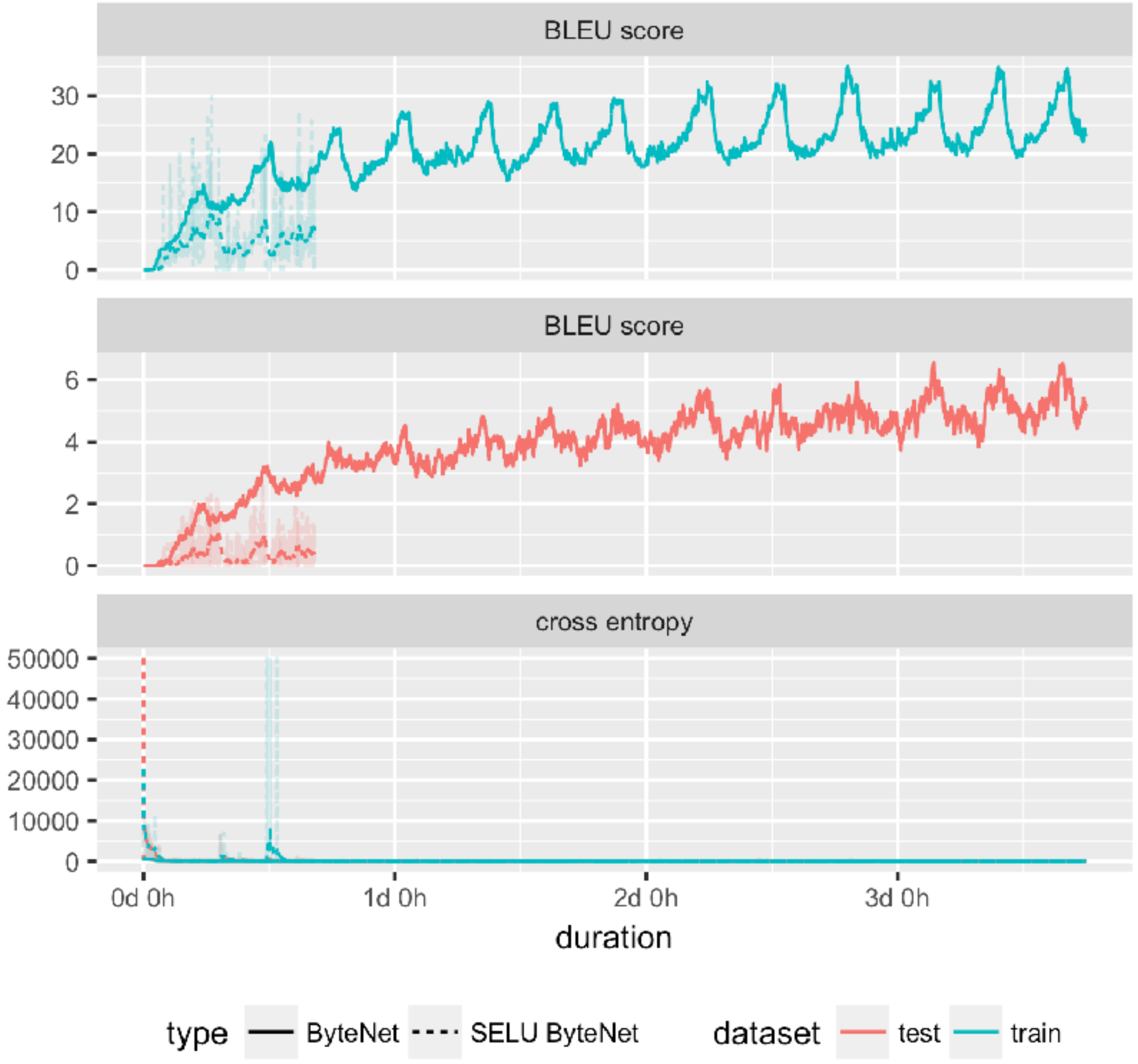
- By assuming independence the Lyapunov variant of CLT is used to derive a activation function that causes the network to self-normalize.
- The SELU paper shows a big improvement on the MNIST and CIFAR10 dataset using a CNN classifier.

SELU ByteNet



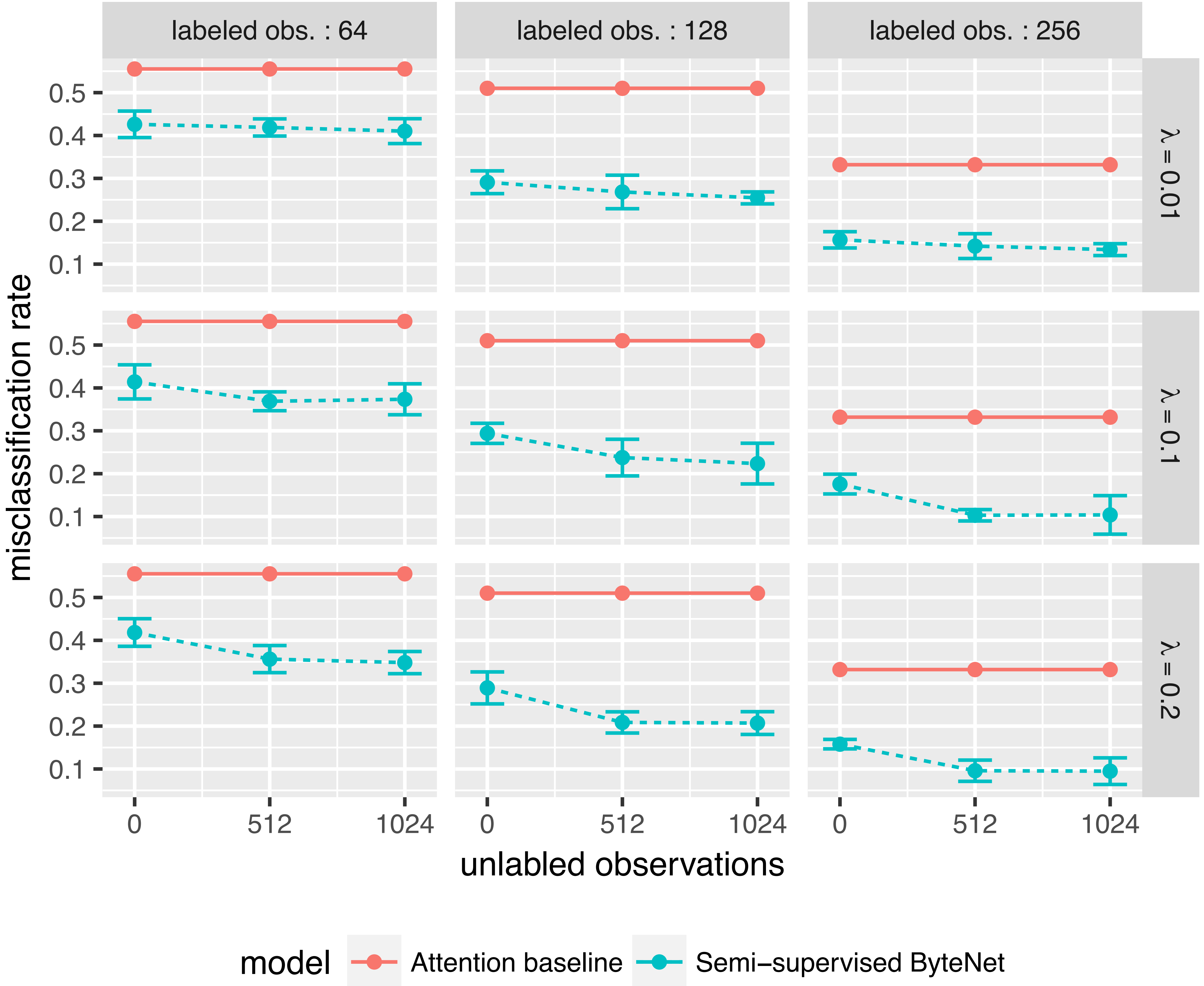
- Convergences is somewhat similar to normal ByteNet.
- Big improvement over NoNorm. ByteNet.

Experiments



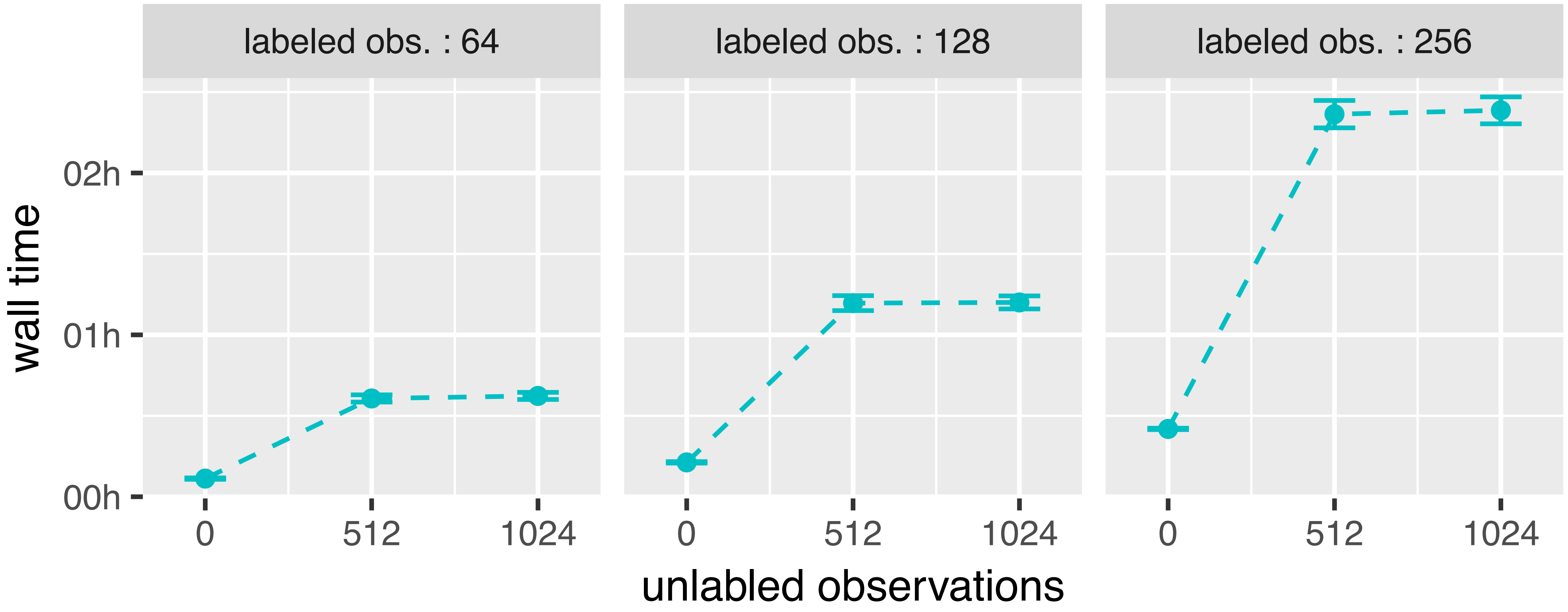
- Much faster, but convergence is almost non existing.
- It appears there are some expoding gradient issues. This should not happen with SELU ByteNet.
- Perhaps gradient clipping/normalization can be used to improve the gradient issues.

Semi-Supervised ByteNet



- On the Synthetic Digits problem, Semi-Supervised works reasonably well.

- Semi-Supervised BytNet is approximately 5 times slower.



Conclusion

ByteNet achieves a BLEU score of 7.44, a 4x more complex model and more epochs would could reach a BLEU score of 23.

The ByteNet model is too slow to be useful for semi-supervised neural machine translation.

Conclusion

Translation Model	Real train time	GPU train time	BLEU
Transformer (*)	**3 days on 8 GPU**	**24 days**	**28.4**
SliceNet	6 days on 32 GPUs	192 days	26.1
GNMT + Mixture of Experts	1 day on 64 GPUs	64 days	26.0
ConvS2S	18 days on 1 GPU	18 days	25.1
GNMT	1 day on 96 GPUs	96 days	24.6
ByteNet	**8 days on 32 GPUs**	**256 days**	**23.8**
MOSES (phrase-based baseline)	N/A	N/A	20.6

source: Google Research Blog

(*) A small Transformer model (not shown above) gets 24.9 BLEU
after 1 day of training on a single GPU.

Conclusion

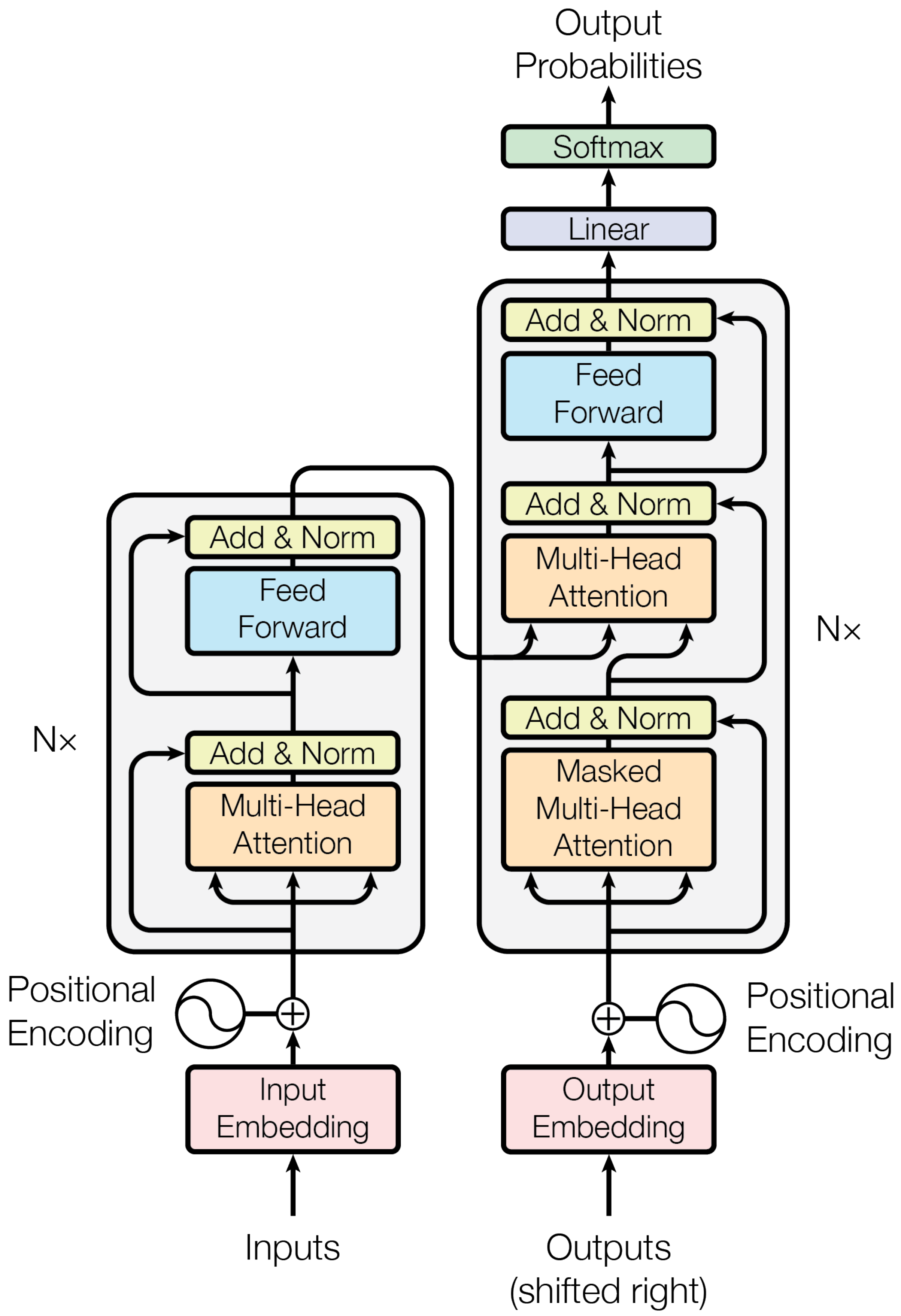
The ByteNet paper did not disclose the required running time. Given its poor performance and its forces on performance, this is scientifically inadequate.

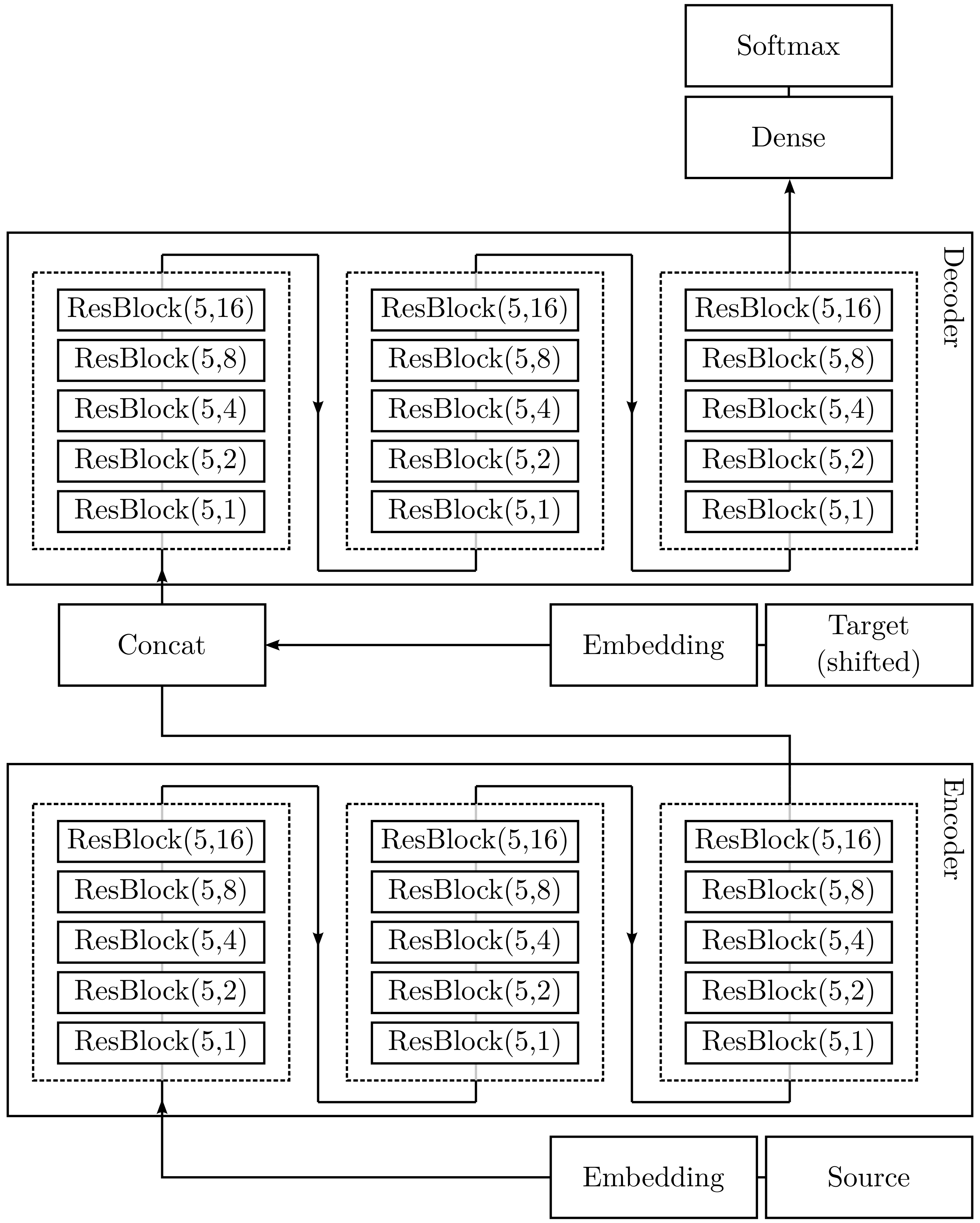
Conclusion

The Semi-Supervised ByteNet model works on the Synthetic Digits problem.

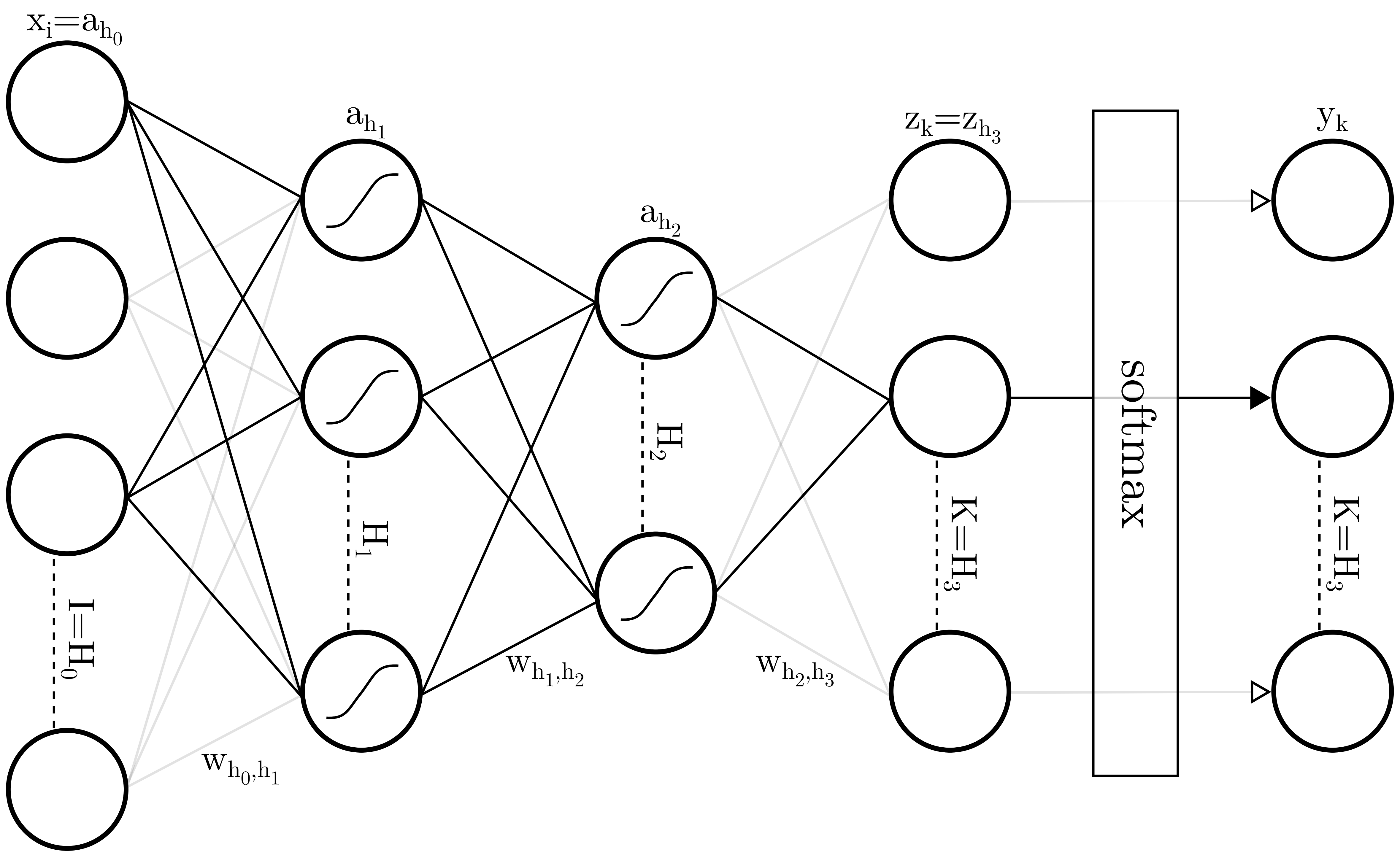
The Transformer model is likely to work well with the semi-supervised approach and should train within reasonable time.

Transformer vs ByteNet





Feed Forward Neural Network



$$z_{h_\ell} = \sum_{h_{\ell-1}=1}^{H_{\ell-1}} w_{h_{\ell-1},h_\ell} a_{h_{\ell-1}} + b_{h_\ell}$$

$$a_{h_\ell} = \theta(z_{h_\ell})$$

$$y_k = \frac{\exp(z_k)}{\sum_{k'=1}^{K} \exp(z_{k'})}$$

$$\mathcal{L} = -\sum_{k=1}^{K} t_k \ln(y_k)$$

Feed Forward Neural Network

$$\frac{\partial \mathcal{L}}{\partial w_{h_{\ell-1},h_\ell}} = \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} \frac{\partial z_{h_\ell}}{\partial w_{h_{\ell-1},h_\ell}} = \delta_{h_\ell} a_{h_{\ell-1}}$$

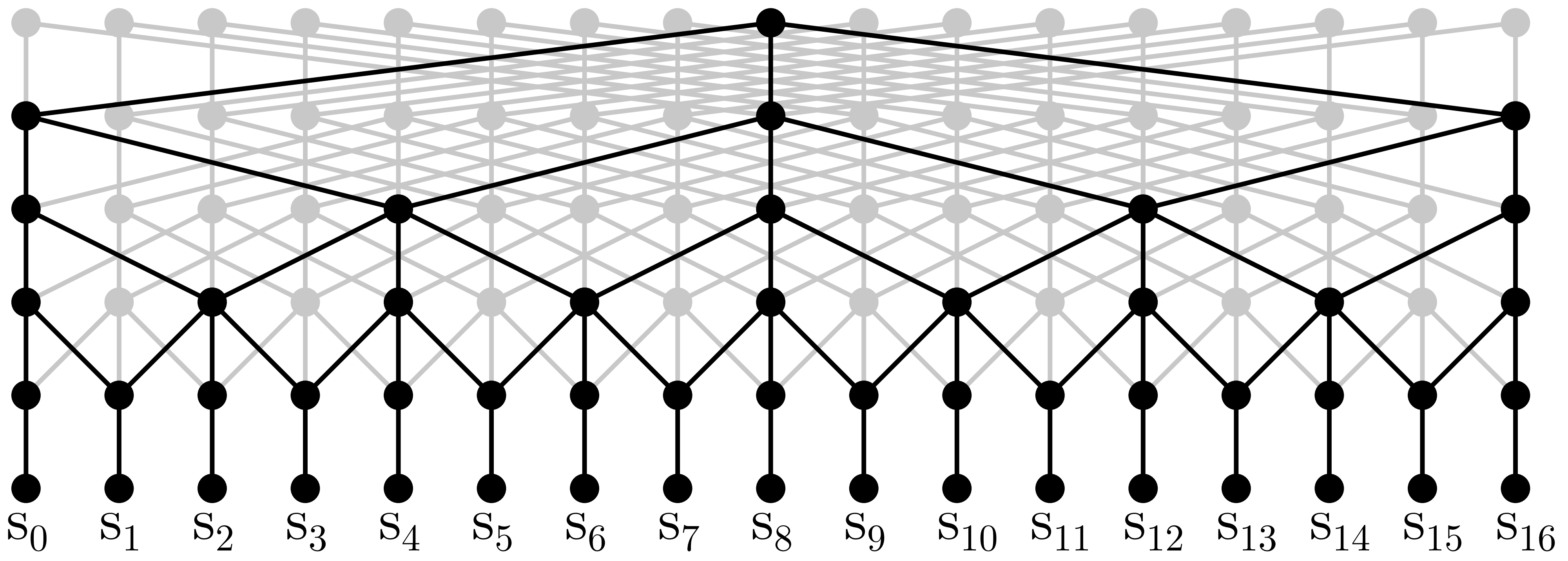
$$\frac{\partial \mathcal{L}}{\partial b_{h_\ell}} = \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} \frac{\partial z_{h_\ell}}{\partial b_{h_\ell}} = \delta_{h_\ell}$$

$$\begin{aligned}\delta_{h_\ell} = \frac{\partial \mathcal{L}}{\partial z_{h_\ell}} &= \frac{\partial \mathcal{L}}{\partial a_{h_\ell}} \frac{\partial a_{h_\ell}}{\partial z_{h_\ell}} \\ &= \theta'(z_\ell) \sum_{h_{\ell+1}}^{H_{\ell+1}} \frac{\partial \mathcal{L}}{\partial z_{\ell+1}} \frac{\partial z_{\ell+1}}{\partial a_\ell} \\ &= \theta'(z_\ell) \sum_{h_{\ell+1}}^{H_{\ell+1}} \delta_{h_{\ell+1}} w_{h_\ell,h_{\ell+1}}\end{aligned}$$

$$\delta_k = \frac{\partial \mathcal{L}}{\partial z_k} = \sum_{k'=1}^{K} \frac{\partial \mathcal{L}}{\partial y_{k'}} \frac{\partial y_{k'}}{\partial z_k} = y_k - t_k$$

$$\begin{aligned} z_{h_\ell} &= \sum_{h_{\ell-1}=1}^{H_{\ell-1}} w_{h_{\ell-1},h_\ell} a_{h_{\ell-1}} + b_{h_\ell} \\ a_{h_\ell} &= \theta(z_{h_\ell}) \\ y_k &= \frac{\exp(z_k)}{\sum_{k'=1}^{K} \exp(z_{k'})} \\ \mathcal{L} &= -\sum_{k=1}^{K} t_k \ln(y_k)\end{aligned}$$

Convolution



$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_{h_{\ell-1},h_\ell}(i)} &= \sum_t \frac{\partial \mathcal{L}}{\partial z_{h_\ell}(t)} \frac{z_{h_\ell}(t)}{\partial w_{h_{\ell-1},h_\ell}(i)} \\ &= \sum_t \delta_{h_\ell}(t) a_{h_{\ell-1}}(t + r\, i)\end{aligned}$$

$$\begin{aligned}\delta_{h_\ell}(t) &= \theta'(z_{h_\ell}(t)) \sum_{h_{\ell+1}=1}^{H_{\ell+1}} \sum_i \frac{\partial \mathcal{L}}{\partial z_{h_{\ell+1}}(t + r\, i)} \frac{\partial z_{h_{\ell+1}}(t + r\, i)}{\partial a_{h_\ell}(t)} \\ &= \theta'(z_{h_\ell}(t))(\delta_{\ell+1} *_r \mathrm{rot}(w_{:,h_{\ell+1}}))(t)\end{aligned}$$

$$z_{h_\ell}(t) = (a_{\ell-1} *_r w_{:,h_\ell})(t) = \sum_{h_{\ell-1}}^{H_{\ell-1}} \sum_i a_{h_{\ell-1}}(t + r\, i) w_{h_{\ell-1},h_\ell}(i)$$

$$a_{h_\ell}(t) = \theta(z_{h_\ell}(t))$$

Optimization

- ByteNet has 27 million parameters. Hessian matrices are thus too expensive.
- Optimize Iteratively using a gradient decent.

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \boldsymbol{\Delta}_t, \quad \boldsymbol{\Delta}_t = \alpha \frac{\partial \mathcal{L}(\mathbf{w}_{t-1})}{\partial \mathbf{w}_{t-1}}$$

- When using mini-batch stochastic gradient decent, the objective changes from $\mathcal{L}$ to $\mathbb{E}[\mathcal{L}]$.

mini-batch stochastic gradient decent

- Consider datasets with 256 and 16 observations. Either reduce standard deviation by a factor of 4 or increase iterations by 16.
- Easy to parallelize over small datasets.
- The noise added by sampling can have a regularizing effect.

Adam Optimization

- Scale step-size with “signal-to-noise”.

$$\mathbf{g}_t = \frac{\partial \mathcal{L}(\mathbf{w}_{t-1})}{\partial \mathbf{w}_{t-1}}$$

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2$$

- Running estimates of first and second order moment.
- Initialize $\mathbf{m}_0$ and $\mathbf{v}_0$ to zero.

- Modify estimates such they are bias-corrected given the zero-initialization.

$$\hat{\mathbf{m}}_t = \frac{1}{1-\beta_1^t}\mathbf{m}_t, \quad \hat{\mathbf{v}}_t = \frac{1}{1-\beta_2^t}\mathbf{v}_t$$

- The final update equations are then:

$$\mathbf{\Delta}_t = \alpha \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}\mathbf{g}_t$$

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \mathbf{\Delta}_t$$

He-Uniform Initialization

- The forward and backward pass are given as :

$$z_{h_\ell} = \sum_{h_{\ell-1}=1}^{H_{\ell-1}} w_{h_{\ell-1},h_\ell} a_{h_{\ell-1}} + b_{h_\ell}, a_{h_\ell} = \theta(z_{h_\ell})$$

$$\delta_{h_\ell} = \theta'(z_\ell) \sum_{h_{\ell+1}}^{H_{\ell+1}} \delta_{h_{\ell+1}} w_{h_\ell,h_{\ell+1}}$$

- Good initialization should have the properties:

$$\mathbb{E}[z_{h_\ell}] = 0, \mathbb{E}[\delta_{h_\ell}] = 0$$
$$\mathrm{Var}[z_{h_\ell}] = c, \mathrm{Var}[\delta_{h_\ell}] = c$$

- Assuming $w_{h_{\ell-1},h_\ell}$ is symmetrically distributed around zero, yields:

$$\begin{aligned}\mathrm{Var}[z_{h_\ell}] &= H_{\ell-1}\mathrm{Var}[w_{h_{\ell-1},h_\ell}]\mathbb{E}[a^2_{h_{\ell-1}}] \\ &= H_{\ell-1}\mathrm{Var}[w_{h_{\ell-1},h_\ell}]\frac{1}{2}\mathrm{Var}[z_{h_{\ell-1}}] \\ &= \mathrm{Var}[z_{h_1}]\prod_{\ell=2}^{L} H_{\ell-1}\frac{1}{2}\mathrm{Var}[w_{h_{\ell-1},h_\ell}]\end{aligned}$$

- Thus, the initialization should be:

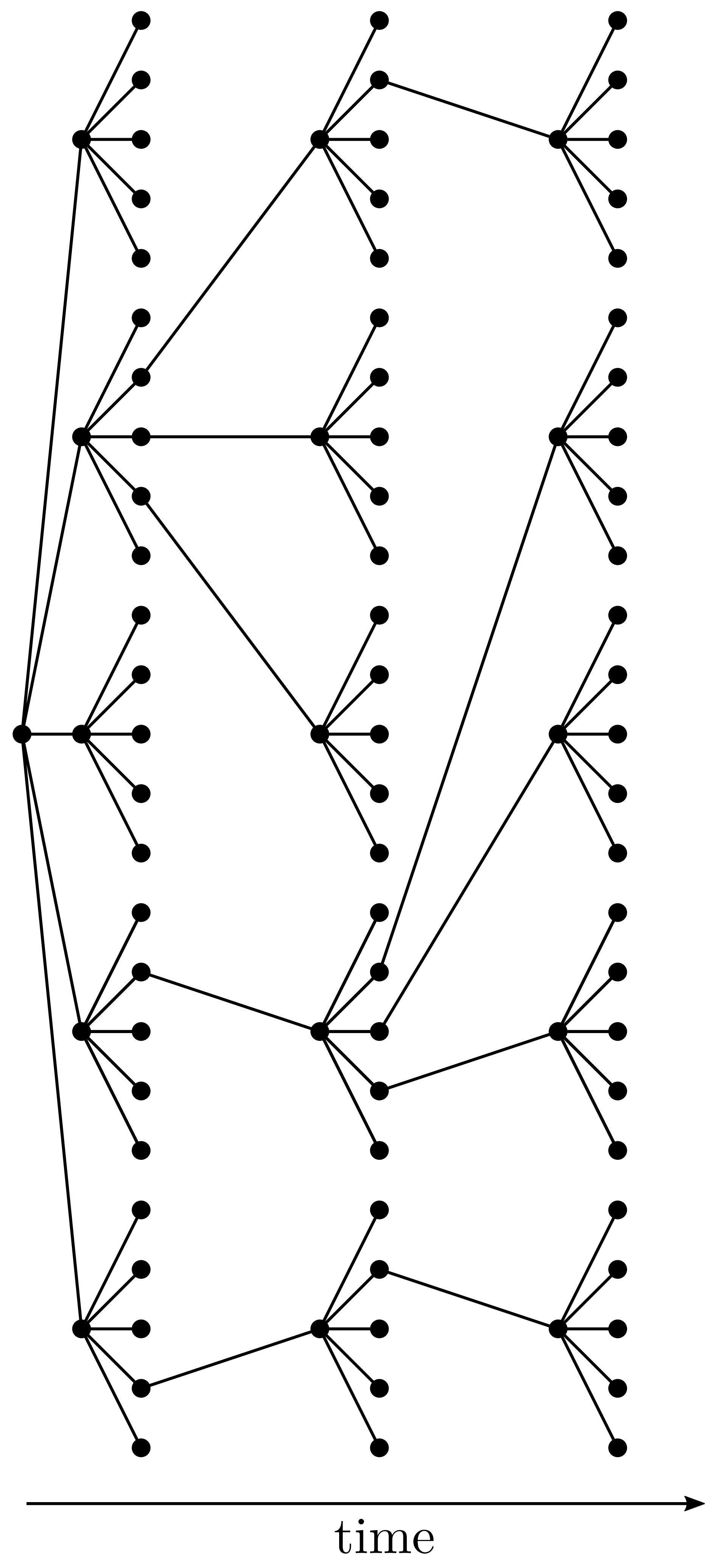
$$\mathrm{Var}[w_{h_{\ell-1},h_\ell}] = \frac{2}{H_{\ell-1}}$$

$$w_{h_{\ell-1},h_\ell} \sim U\left[-\sqrt{\frac{6}{H_{\ell-1}}}, \sqrt{\frac{6}{H_{\ell-1}}}\right]$$

BeamSearch

Algorithm 2 BeamSearch algorithm, specialized for scoring by sequence probability.

```
function BeamSearch($P(\mathbf{y}|\mathbf{x}), \mathbf{x}, b$)
    Initialize paths from top $b$ selection from $P(y_1|\{x_1, \ldots, x_{|S|}\})$
    $i \leftarrow 2$
    repeat
        for $\{y_1, \ldots, y_{i-1}\}$ in paths do
            Compute probabilities of new paths $\{y_1, \ldots, y_i\}$ conditioned on the old
            path $\{y_1, \ldots, y_{i-1}\}$.
        Update paths by selecting the top $b$ new paths by the joint probability
        $P(\{y_1, \ldots, y_i\}|\{x_1, \ldots, x_{|S|}\})$.
        $i \leftarrow i + 1$
    until <EOS> $\in$ paths
    return paths
```



Layer Normalization

Activation:

$$z_{h_\ell} = \sum_{h_{\ell-1}}^{H_{\ell-1}} w_{h_{\ell-1},h_\ell} a_{h_\ell - 1}$$

$$\hat{z}_{h_\ell} = \gamma_{h_\ell} \frac{z_{h_\ell} - \mu_\ell}{\sqrt{\sigma_\ell^2 + \epsilon}} + \beta_{h_\ell}$$

$$a_{h_\ell} = \theta\left(\hat{z}_{h_\ell}\right)$$

Statistics:

$$\mu_\ell = \frac{1}{H_\ell} \sum_{h_\ell}^{H_\ell} z_{h_\ell}$$

$$\sigma_\ell^2 = \frac{1}{H_\ell} \sum_{h_\ell}^{H_\ell} (z_{h_\ell} - \mu_\ell)^2$$

- In Batch Normalization the idea is that the weights causes an internal covariate shift. Thus the normalization happens over the time axis as well.
- Normalization over time can be an issue for sequences that aren't equally long.
- Layer Normalization is a more direct approach that directly normalizes on the input to the activation function.
- Both Batch and Layer Normalization are invariant to weight matrix re-scaling.