A. Weight Initialization

We derive a weight initialization scheme tailored to the GLU activation function similar to Glorot & Bengio (2010); He et al. (2015b) by focusing on the variance of activations within the network for both forward and backward passes. We also detail how we modify the weight initialization for dropout.

A.1. Forward Pass

Assuming that the inputs $\mathbf{x}_l$ of a convolutional layer l and its weights W_l are independent and identically distributed (i.i.d.), the variance of its output, computed as $\mathbf{y}_l = W_l\mathbf{x}_l + \mathbf{b}_l$, is

$$Var\left[y_l\right] = n_l Var\left[w_l x_l\right] \tag{3}$$

where n_l is the number inputs to the layer. For one-dimensional convolutional layers with kernel width k and input dimension c, this is kc. We adopt the notation in (He et al., 2015b), i.e. y_l, w_l and x_l represent the random variables in $\mathbf{y}_l$, W_l and $\mathbf{x}_l$. With w_l and x_l independent from each other and normally distributed with zero mean, this amounts to

$$Var\left[y_l\right] = n_l Var\left[w_l\right] Var\left[x_l\right]. \tag{4}$$

$\mathbf{x}_l$ is the result of the GLU activation function $\mathbf{y}^a_{l-1}\sigma(\mathbf{y}^b_{l-1})$ with $\mathbf{y}_{l-1} = (\mathbf{y}^a_{l-1}, \mathbf{y}^b_{l-1})$, and $\mathbf{y}^a_{l-1}, \mathbf{y}^b_{l-1}$ i.i.d. Next, we formulate upper and lower bounds in order to approximate $Var[x_l]$. If $\mathbf{y}_{l-1}$ follows a symmetric distribution with mean 0, then

$$\begin{aligned} Var\left[x_l\right] &= Var\left[y^a_{l-1}\sigma(\mathbf{y}^b_{l-1})\right] && (5)\\ &= E\left[\left(y^a_{l-1}\sigma(\mathbf{y}^b_{l-1})\right)^2\right] - E^2\left[y^a_{l-1}\sigma(\mathbf{y}^b_{l-1})\right] && (6)\\ &= Var[y^a_{l-1}]E\left[\sigma(\mathbf{y}^b_{l-1})^2\right]. && (7)\end{aligned}$$

A lower bound is given by $(1/4)Var[y^a_{l-1}]$ when expanding (6) with $E^2[\sigma(\mathbf{y}^b_{l-1})] = 1/4$:

$$\begin{aligned} Var\left[x_l\right] &= Var\left[y^a_{l-1}\sigma(\mathbf{y}^b_{l-1})\right] && (8)\\ &= Var\left[y^a_{l-1}\right]E^2\left[\sigma(\mathbf{y}^b_{l-1})\right] + \\ &\quad\; Var\left[y^a_{l-1}\right]Var\left[\sigma(\mathbf{y}^b_{l-1})\right] && (9)\\ &= \frac{1}{4}Var\left[y^a_{l-1}\right] + Var\left[y^a_{l-1}\right]Var\left[\sigma(\mathbf{y}^b_{l-1})\right] && (10)\end{aligned}$$

and $Var[y^a_{l-1}]Var[\sigma(\mathbf{y}^b_{l-1})] > 0$. We utilize the relation $\sigma(x)^2 \leq (1/16)x^2 - 1/4 + \sigma(x)$ (Appendix B) to provide an upper bound on $E[\sigma(x)^2]$:

$$\begin{aligned} E[\sigma(x)^2] &\leq E\left[\frac{1}{16}x^2 - \frac{1}{4} + \sigma(x)\right] && (11)\\ &= \frac{1}{16}E[x^2] - \frac{1}{4} + E[\sigma(x)] && (12)\end{aligned}$$

With $x \sim \mathcal{N}(0, std(x))$, this yields

$$\begin{aligned} E\left[\sigma(x)^2\right] &\leq \frac{1}{16}E\left[x^2\right] - \frac{1}{4} + \frac{1}{2} && (13)\\ &= \frac{1}{16}Var\left[x\right] + \frac{1}{4}. && (14)\end{aligned}$$

With (7) and $Var[y^a_{l-1}] = Var[y^b_{l-1}] = Var[y_{l-1}]$, this results in

$$Var\left[x_l\right] \leq \frac{1}{16}Var\left[y_{l-1}\right]^2 + \frac{1}{4}Var\left[y_{l-1}\right]. \tag{15}$$

We initialize the embedding matrices in our network with small variances (around 0.01), which allows us to dismiss the quadratic term and approximate the GLU output variance with

$$Var[x_l] \approx \frac{1}{4}Var[y_{l-1}]. \tag{16}$$

If L network layers of equal size and with GLU activations are combined, the variance of the final output $\mathbf{y}_L$ is given by

$$Var[y_L] \approx Var[y_1]\left(\prod_{l=2}^{L}\frac{1}{4}n_l Var[w_l]\right). \tag{17}$$

Following (He et al., 2015b), we aim to satisfy the condition

$$\frac{1}{4}n_l Var\left[w_l\right] = 1, \forall l \tag{18}$$

so that the activations in a network are neither exponentially magnified nor reduced. This is achieved by initializing W_l from $\mathcal{N}(0, \sqrt{4/n_l})$.

A.2. Backward Pass

The gradient of a convolutional layer is computed via back-propagation as $\Delta\mathbf{x}_l = \hat{W}_l\mathbf{y}_l$. Considering separate gradients $\Delta\mathbf{y}^a_l$ and $\Delta\mathbf{y}^b_l$ for GLU, the gradient of $\mathbf{x}$ is given by

$$\Delta\mathbf{x}_l = \hat{W}^a_l\Delta\mathbf{y}^a_l + \hat{W}^b_l\Delta\mathbf{y}^b_l. \tag{19}$$

$\hat{W}$ corresponds to W with re-arranged weights to enable back-propagation. Analogously to the forward pass, Δx_l, $\hat{w}_l$ and Δy_l represent the random variables for the values in $\Delta\mathbf{x_l}$, $\hat{W}_l$ and $\Delta\mathbf{y_l}$, respectively. Note that W and $\hat{W}$ contain the same values, i.e. $\hat{w} = w$. Similar to (3), the variance of Δx_l is

$$Var[\Delta x_l] = \hat{n}_l\Big(Var[w^a_l]Var[\Delta y^a_l] + Var[w^b_l]Var[\Delta y^b_l]\Big). \tag{20}$$

Here, $\hat{n}_l$ is the number of inputs to layer $l+1$. The gradients for the GLU inputs are:

$$\Delta\mathbf{y}^a_l = \Delta\mathbf{x}_{l+1}\sigma(\mathbf{y}^b_l) \quad \text{and} \tag{21}$$

$$\Delta\mathbf{y}^b_l = \Delta\mathbf{x}_{l+1}\mathbf{y}^a_l\sigma'(\mathbf{y}^b_l). \tag{22}$$

The approximation for the forward pass can be used for $Var[\Delta y^a_l]$, and for estimating $Var[\Delta y^b_l]$ we assume an upper bound on $E[\sigma'(y^b_l)^2]$ of $1/16$ since $\sigma'(y^b_l) \in [0, \frac{1}{4}]$. Hence,

$$Var[\Delta y^a_l] - \frac{1}{4}Var[\Delta x_{l+1}] \leq \frac{1}{16}Var[\Delta x_{l+1}]Var[y^b_l])] \tag{23}$$

$$Var[\Delta y^b_l] \leq \frac{1}{16}\Delta Var[\Delta x_{l+1}]Var[y^a_l] \tag{24}$$

We observe relatively small gradients in our network, typically around 0.001 at the start of training. Therefore, we approximate by discarding the quadratic terms above, i.e.

$$Var[\Delta y_l^a] \approx \frac{1}{4} Var[\Delta x_{l+1}] \tag{25}$$

$$Var[\Delta y_l^b] \approx 0 \tag{26}$$

$$Var[\Delta x_l] \approx \frac{1}{4} \hat{n}_l Var[w_l^a] Var[\Delta x_{l+1}] \tag{27}$$

As for the forward pass, the above result can be generalized to backpropagation through many successive layers, resulting in

$$Var[\Delta x_2] \approx Var[\Delta x_{L+1}] \left(\prod_{l=2}^{L} \frac{1}{4} \hat{n}_l Var[w_l^a] \right) \tag{28}$$

and a similar condition, i.e. $(1/4)\hat{n}_l Var[w_l^a] = 1$. In the networks we consider, successions of convolutional layers usually operate on the same number of inputs so that most cases $n_l = \hat{n}_l$. Note that W_l^b is discarded in the approximation; however, for the sake of consistency we use the same initialization for W_l^a and W_l^b.

For arbitrarily large variances of network inputs and activations, our approximations are invalid; in that case, the initial values for W_l^a and W_l^b would have to be balanced for the input distribution to be retained. Alternatively, methods that explicitly control the variance in the network, e.g. batch normalization (Ioffe & Szegedy, 2015) or layer normalization (Ba et al., 2016) could be employed.

A.3. Dropout

Dropout retains activations in a neural network with a probability p and sets them to zero otherwise (Srivastava et al., 2014). It is common practice to scale the retained activations by $1/p$ during training so that the weights of the network do not have to be modified at test time when p is set to 1. In this case, dropout amounts to multiplying activations $\mathbf{x}$ by a Bernoulli random variable r where $\Pr[r = 1/p] = p$ and $\Pr[r = 0] = 1 - p$ (Srivastava et al., 2014). It holds that $E[r] = 1$ and $Var[r] = (1-p)/p$. If x is independent of r and $E[x] = 0$, the variance after dropout is

$$Var[xr] = E[r]^2 Var[x] + Var[r] Var[x] \tag{29}$$

$$= \left(1 + \frac{1-p}{p}\right) Var[x] \tag{30}$$

$$= \frac{1}{p} Var[x] \tag{31}$$

Assuming that a the *input* of a convolutional layer has been subject to dropout with a retain probability p, the variations of the forward and backward activations from§A.1 and§A.2 can now be approximated with

$$Var[x_{l+1}] \approx \frac{1}{4p} n_l Var[w_l] Var[x_l] \quad \text{and} \tag{32}$$

$$Var[\Delta x_l] \approx \frac{1}{4p} n_l Var[w_l^a] Var[\Delta x_{l+1}]. \tag{33}$$

This amounts to a modified initialization of W_l from a normal distribution with zero mean and a standard deviation of $\sqrt{4p/n}$. For layers without a succeeding GLU activation function, we initialize weights from $\mathcal{N}(0, \sqrt{p/n})$ to calibrate for any immediately preceding dropout application.

B. Upper Bound on Squared Sigmoid

The sigmoid function $\sigma(x)$ can be expressed as a hyperbolic tangent by using the identity $\tanh(x) = 2\sigma(2x) - 1$. The derivative of tanh is $\tanh'(x) = 1 - \tanh^2(x)$, and with $\tanh(x) \in [0,1], x \geq 0$ it holds that

$$\tanh'(x) \leq 1, x \geq 0 \tag{34}$$

$$\int_0^x \tanh'(x) \mathrm{d}x \leq \int_0^x 1 \mathrm{d}x \tag{35}$$

$$\tanh(x) \leq x, x \geq 0 \tag{36}$$

We can express this relation with $\sigma(x)$ as follows:

$$2\sigma(x) - 1 \leq \frac{1}{2} x, x \geq 0 \tag{37}$$

Both terms of this inequality have rotational symmetry w.r.t 0, and thus

$$\left(2\sigma(x) - 1\right)^2 \leq \left(\frac{1}{2} x\right)^2 \quad \forall x \tag{38}$$

$$\Leftrightarrow \sigma(x)^2 \leq \frac{1}{16} x^2 - \frac{1}{4} + \sigma(x). \tag{39}$$

C. Attention Visualization

Figure 3 shows attention scores for a generated sentence from the WMT'14 English-German task. The model used for this plot has 8 decoder layers and a 80K BPE vocabulary. The attention passes in different decoder layers capture different portions of the source sentence. Layer 1, 3 and 6 exhibit a linear alignment. The first layer shows the clearest alignment, although it is slightly off and frequently attends to the corresponding source word of the previously generated target word. Layer 2 and 8 lack a clear structure and are presumably collecting information about the whole source sentence. The fourth layer shows high alignment scores on nouns such as "festival", "way" and "work" for both the generated target nouns as well as their preceding words. Note that in German, those preceding words depend on gender and object relationship of the respective noun. Finally, the attention scores in layer 5 and 7 focus on "built", which is reordered in the German translation and is moved from the beginning to the very end of

the sentence. One interpretation for this is that as generation progresses, the model repeatedly tries to perform the re-ordering. "aufgebaut" can be generated after a noun or pronoun only, which is reflected in the higher scores at positions 2, 5, 8, 11 and 13.

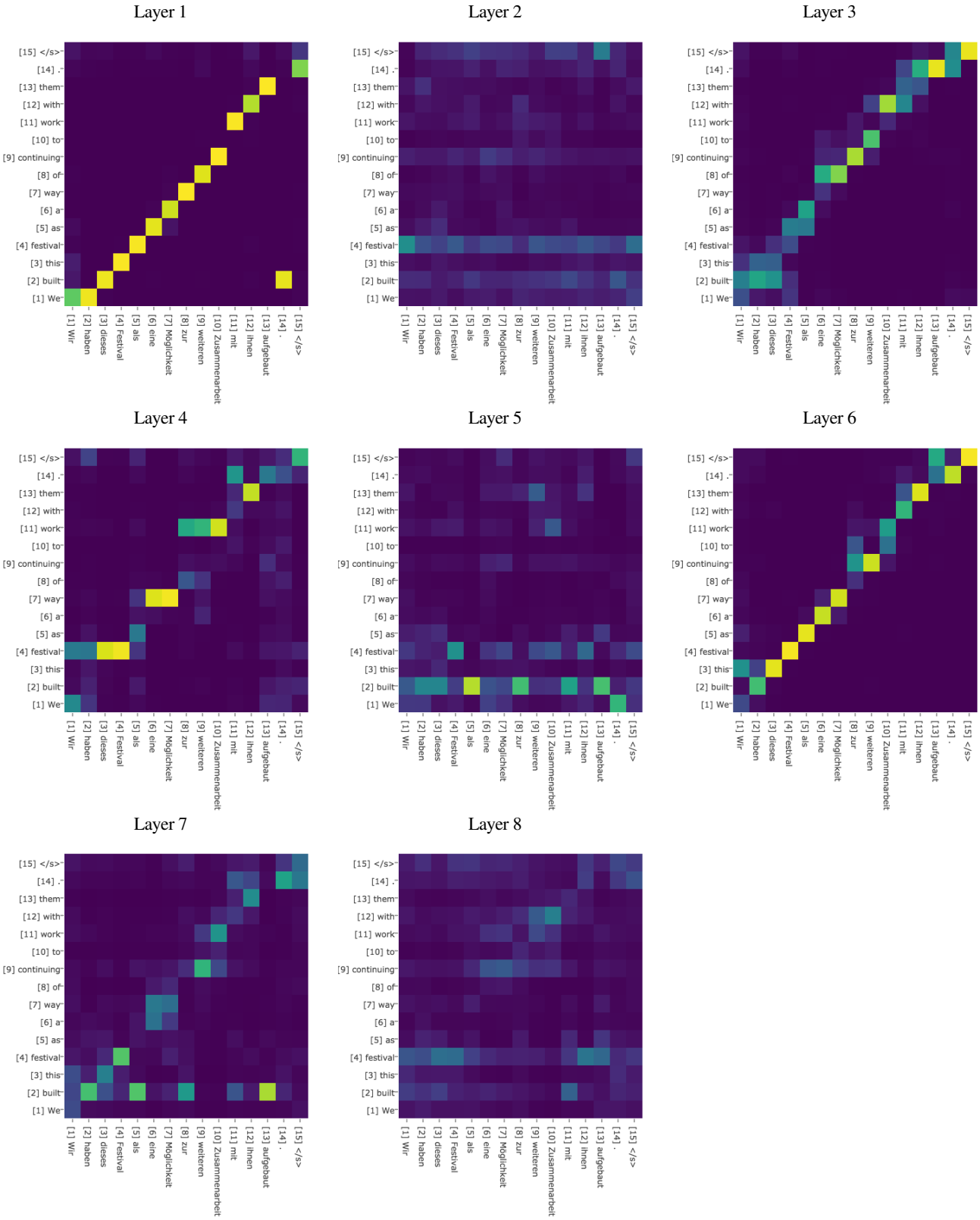

Figure 3. Attention scores for different decoder layers for a sentence translated from English (y-axis) to German (x-axis). This model uses 8 decoder layers and a 80k BPE vocabulary.