JASPER

UCS749: SPEECH PROCESSING AND SYNTHESIS

Raghav B. Venkataramaiyer

CSED TIET Patiala India.

September 17, 2024

OUTLINE

METADATA

TAGS	Interspeech 2019
KEYWORDS	Computer Science - Computation and Language, Computer Science - Machine Learning, Computer Science - Sound, Electrical Engineering and Systems Science - Audio and Speech Processing
AUTHOR	Li, J., Lavrukhin, V., Ginsburg, B., Leary, R., Kuchaiev, O., Cohen, J. M., Nguyen, H., [...]
URL	[arXiv], [Papers With Code]

OUTLINE

DATASETS

- LibriSpeech (ICASSP '15)
- WSJ: LDC93S6A (WSJ0), LDC94S13A (WSJ1)
- 2000hr Fisher+Switchboard (F+S): LDC2004S13, LDC2005S13, LDC97S62

IDEAS AND STRATEGY

- Time-delay Networks (TDNN) [PDF]
- Connectionist Temporal Classification (CTC) ICML '06 [ACM]
- Wav2Letter (ICLR '17)
- Gated Convnets (for ASR) [arXiv]

Normalisation
- Batch Norm; See also:[arXiv];
- Weight Norm; See also: [arXiv];
- Layer Norm; See also: [arXiv];

Activation
- ReLU
- Clipped ReLU
- Leaky ReLU

Gated Units
- Gated Linear Units
- Gated Activation Units

OUTLINE

Contribution

- Jasper Model; and Dense Residual Topology;
- Evidence-based insights on convergence and non-convergence;
- NovoGrad Solver (like Adam);
- WER improvement on LibriSpeech test-clean.

JASPER MODEL

All figures and tables repeated here from the paper [LLGL+19].
Model search across

3 TYPES OF NORMALISATION Batch Norm, Weight Norm, Layer Norm;

3 TYPES OF ACTIVATIONS ReLU, clipped ReLU, Leaky ReLU;

2 TYPES OF GATES Gated Linear Units, Gated Activation Units;

ARCHITECTURE $B \times R$ parameterisation, Dense Residual.

JASPER $B \times R$ ARCHITECTURE

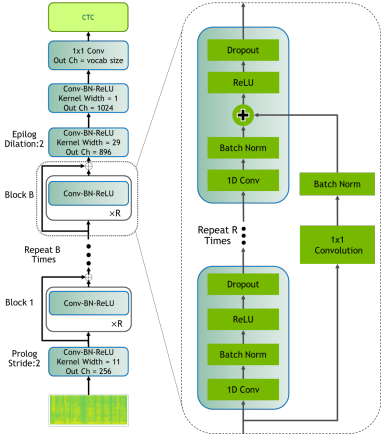
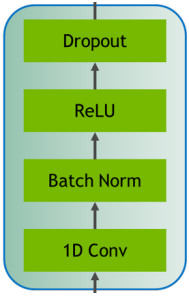


FIGURE: Jasper $B \times R$ model: B: number of blocks; R: number of sub-blocks.

TABLE: Jasper 10×5

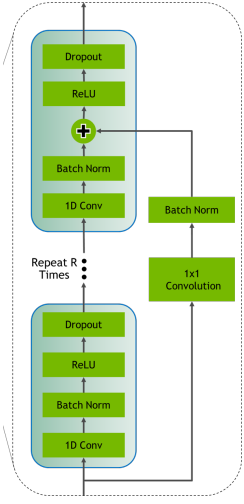
#B	#S	B	K	#C (out)	Dropout
1	1	Conv1	11 (s=2)	256	0.2
2	5	B1	11	256	0.2
2	5	B2	13	384	0.2
2	5	B3	17	512	0.2
2	5	B4	21	640	0.3
2	5	B5	25	768	0.3
1	1	Conv2	29 (D=2)	896	0.4
1	1	Conv3	1	1024	0.4
1	1	Conv4	1	#graphemes	0

FUNDAMENTAL CONV BLOCK



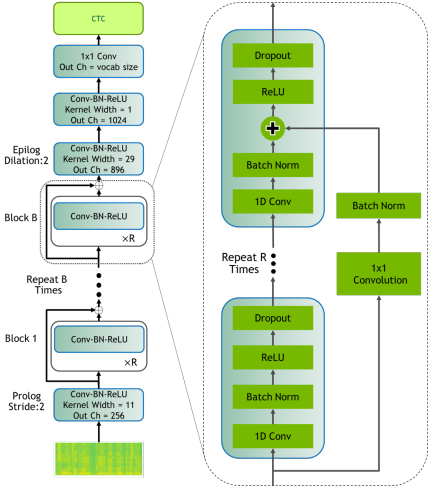
The fundamental conv block with
Conv $\mapsto$ Batch Norm $\mapsto$ ReLU $\mapsto$ Dropout
progression

JASPER BLOCK



The fundamental Conv Block (aka Sub-Block) is repeated R times, with a residual connection from inputs to the final block as in figure, to create a Jasper Block.

$B \times R$ AGAIN



The Jasper Block is repeated B times to create the $B \times R$ architecture.

JASPER DENSE RESIDUAL ARCHITECTURE

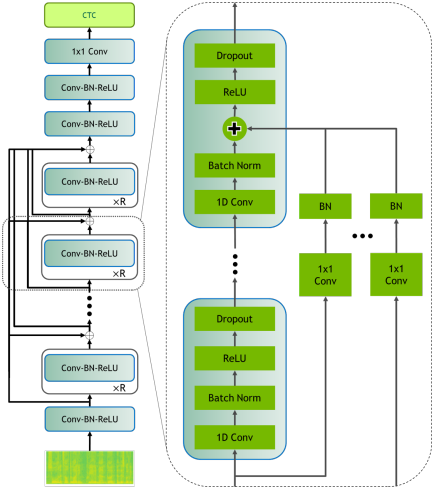


FIGURE: Jasper Dense Residual Model

- The dense residual architecture builds on top of jasper residual architecture; and
- provides skip connections to the final block from each of the previous blocks.
- *I.e.* within a block, the final sub-block receives a skip-connection from each of the previous sub-blocks; and
- on the whole, the final block receives skip connections from each of the previous blocks.

OUTLINE

OUTLINE

COVARIATE SHIFT

One of the challenges of deep learning is that the gradients with respect to the weights in one layer are highly dependent on the outputs of the neurons in the previous layer especially if these outputs change in a highly correlated way.
— *Layer Normalisation Paper*

BATCH NORM

Consider a mini-batch $\mathcal{B}$ of size m. Since the normalization is applied to each activation independently, let us focus on a particular activation $x^{(k)}$ and omit k for clarity. We have m values of this activation in the mini-batch,

$$\mathcal{B} = \{x_{1...m}\}.$$

Let the normalized values be $\widehat{x}_{1...m}$, and their linear transformations be $y_{1...m}$. We refer to the transform

$$\text{BN}_{\gamma,\beta} : x_{1...m} \rightarrow y_{1...m}$$

as the *Batch Normalizing Transform*. We present the BN Transform in Algorithm 1. In the algorithm, ϵ is a constant added to the mini-batch variance for numerical stability.

FIGURE: Courtesy: Batch Norm Paper

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

WEIGHT NORM

Each neuron on an artificial neural network may be represented as,

$$y = \phi(\mathbf{w} \cdot \mathbf{x} + b)$$

where,
$\mathbf{x}$ is k dimensional vector of input features,
$\mathbf{w}$ is k dimensional vector of learnable weights,
b is a (learnable) scalar bias term, and
ϕ denotes element-wise non-linearity.

WEIGHT NORM

The key idea in weight normalisation is to re-parameterise the weight vector, as

$$\mathbf{w} = \frac{g}{\|\mathbf{v}\|}\mathbf{v}$$

so that,
$\mathbf{v}$ is k dimensional vector of learnable weights,
g is a learnable scalar parameter, and
$\|\mathbf{w}\| = g$, independent of $\mathbf{v}$.

WEIGHT NORM

Instead of working with g directly, we may also use an exponential parameterisation for the scale,

$$g = e^s$$

where, s is a log-scale learnable scalar parameter.
For more details, please see §2.1 and §2.2 of the weight norm paper.

Summarising Batch Norm

The l^{th} layer in a feed forward neural network with inputs $\mathbf{h}^l$ and weight matrix W^l and non-linear activation f, may be written as,

$$a_i^l = \mathbf{w}_{:,i}^{l\top} \mathbf{h}^l \qquad h_i^{l+1} = f(a_i^l + b_i^l)$$

A Batch Norm may be summarised as,

$$h_i^{l+1} = f(\hat{a}_i^l + b_i^l) \qquad \hat{a}_i^l = \frac{g_i^l}{\sigma_i^l}(a_i^l - \mu_i^l)$$

$$\mu_i^l = \mathop{\mathbb{E}}_{\mathbf{x}\sim P(\mathbf{x})}\left[a_i^l\right] \qquad \sigma_i^l = \sqrt{\mathop{\mathbb{E}}_{\mathbf{x}\sim P(\mathbf{x})}\left[\left(a_i^l - \mu_i^l\right)^2\right]}$$

A Batch Norm may be summarised as,

$$h_i^{l+1} = f(\hat{a}_i^l + b_i^l)$$
$$\hat{a}_i^l = \frac{g_i^l}{\sigma_i^l}(a_i^l - \mu_i^l)$$
$$\mu_i^l = \mathop{\mathbb{E}}_{\mathbf{x}\sim P(\mathbf{x})}\left[a_i^l\right]$$
$$\sigma_i^l = \sqrt{\mathop{\mathbb{E}}_{\mathbf{x}\sim P(\mathbf{x})}\left[\left(a_i^l - \mu_i^l\right)^2\right]}$$

The authors say,

- It is typically impractical to [exactly] compute the expectations [in the adjoining eqn's]; since
- it would require forward passes through the whole training dataset with the current set of weights;
- Instead, μ and σ are estimated using the empirical samples from the current mini-batch.

In certain cases, the covariate shift is was observed to be more profound at layer level by the authors. The authors say,

> *[...] Notice that changes in the output of one layer will tend to cause highly correlated changes in the summed inputs to the next layer, especially with ReLU units whose outputs can change by a lot.*

We, thus, compute the layer normalization statistics over all the hidden units in the same layer as follows:

$$\mu_i^l = \mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l$$

$$\sigma_i^l = \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)}$$

i.e. μ^l and σ^l are same for all neurons in the same layer.

COMPARING NORMALISATION MECHANISMS

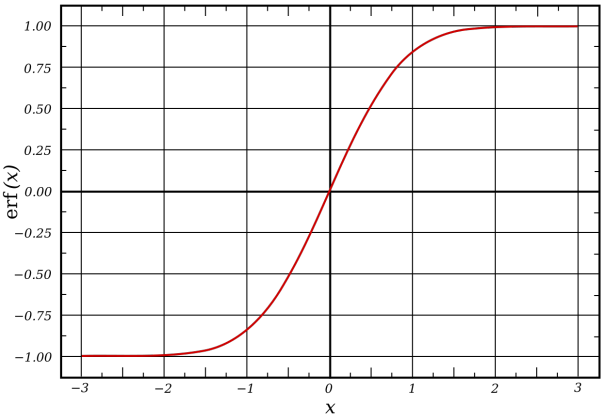
	Weight matrix re-scaling	Weight matrix re-centering	Weight vector re-scaling	Dataset re-scaling	Dataset re-centering	Single training case re-scaling
Batch norm	Invariant	No	Invariant	Invariant	Invariant	No
Weight norm	Invariant	No	Invariant	No	No	No
Layer norm	Invariant	Invariant	No	Invariant	No	Invariant

Table 1: Invariance properties under the normalization methods.

FIGURE: Courtesy: Layer Norm Paper

OUTLINE

ERROR FUNCTION



$$\operatorname{erf} z = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} \mathrm{d}t$$

FIGURE: Image Courtesy: Wikipedia

sigmoid (logistic Function)

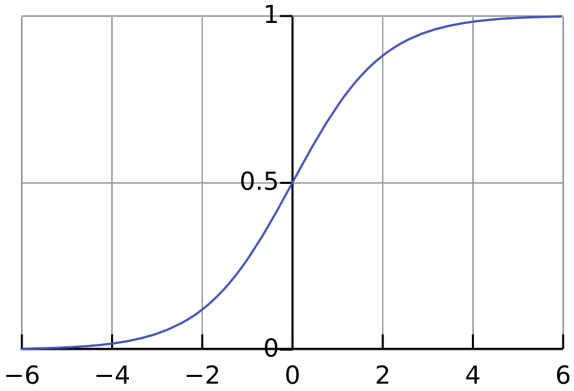


Figure: Image Courtesy: Wikipedia

$$\begin{aligned}\sigma(x) &= \frac{1}{1+e^{-x}} \\ &= \frac{e^{x}}{1+e^{x}} \\ &= 1-\sigma(-x)\end{aligned}$$

OTHER SIGMOIDAL FUNCTIONS

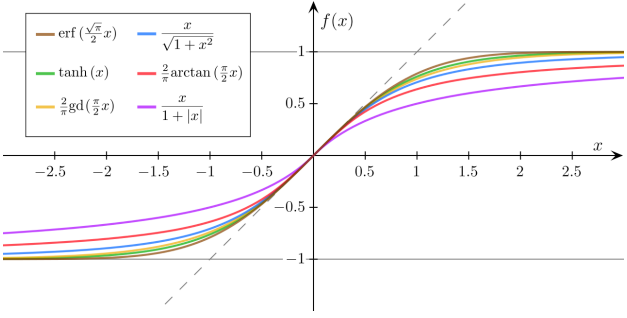


FIGURE: Image Courtesy: Wikipedia

- Hyperbolic Tangent

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

OTHER SIGMOIDAL FUNCTIONS

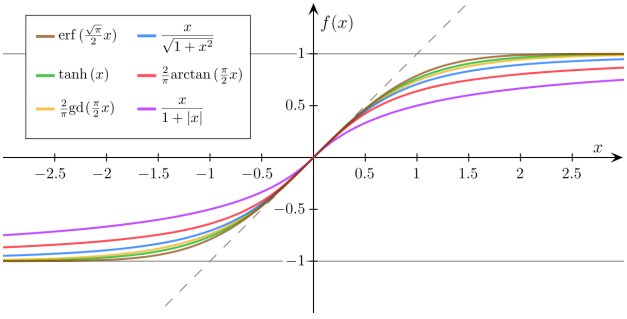


FIGURE: Image Courtesy: Wikipedia

- Arc Tangent

$$\begin{aligned} & y = \arctan x \\ \iff\ & x = \tan y; \\ & y \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \end{aligned}$$

OTHER SIGMOIDAL FUNCTIONS

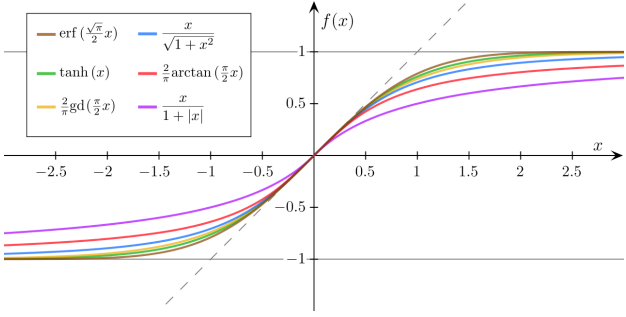


FIGURE: Image Courtesy: Wikipedia

- Gudermannian Function

$$\begin{aligned}\mathrm{gd}(x) &= \int_0^x \frac{\mathrm{d}t}{\cosh\, t} \\ &= 2\arctan\left(\tanh\left(\frac{x}{2}\right)\right)\end{aligned}$$

other sigmoidal Functions

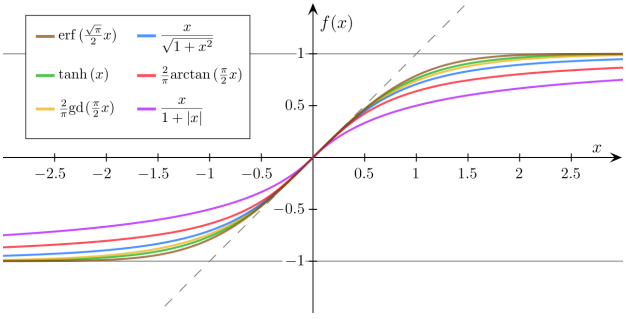


Figure: Image Courtesy: Wikipedia

- Algebraic Functions

$$
\begin{aligned}
f(x) &= \frac{x}{\left(1+|x|^k\right)^{1/k}} \\
&= \frac{x}{(1+|x|)}; && k=1 \\
&= \frac{x}{\sqrt{1+x^2}}; && k=2
\end{aligned}
$$

OUTLINE

RECTIFIER

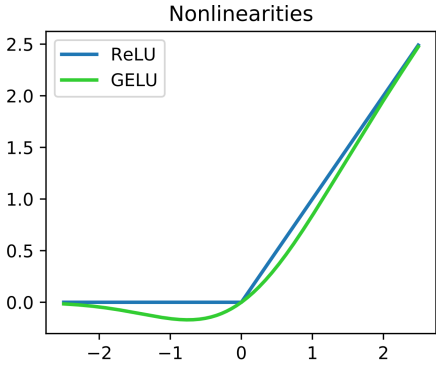


FIGURE: Image Courtesy: Wikipedia

- ReLU (Rectified Linear Unit)

$$\begin{aligned}\mathrm{ReLU}(x) &= x^+ \\ &= \max(0, x) \\ &= \frac{x + |x|}{2} \\ &= \begin{cases} x; & \text{if } x > 0, \\ 0; & \text{otherwise.} \end{cases}\end{aligned}$$

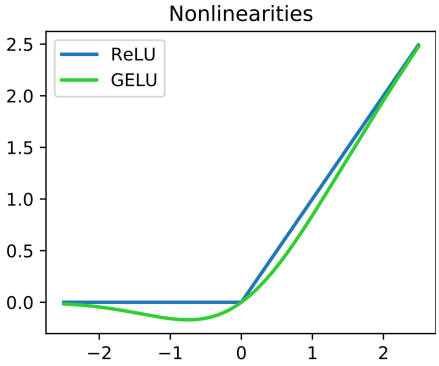


FIGURE: Image Courtesy: Wikipedia

- Clipped ReLU

$$\text{cReLU}(x; a) = \max(0, \min(a, x))$$

e.g. ReLU6 in [Pytorch], [Keras]

RECTIFIER

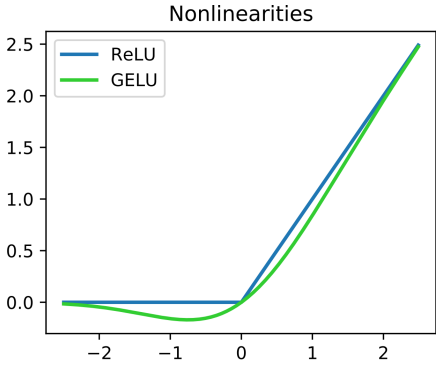


FIGURE: Image Courtesy: Wikipedia

- Parametric and Leaky ReLU

$$\text{PReLU}(x; a) = \begin{cases} x; & \text{if } x > 0, \\ ax; & \text{otherwise.} \end{cases}$$

$$\text{LeakyReLU}(x) = \text{PReLU}(x, 0.01)$$

RECTIFIER

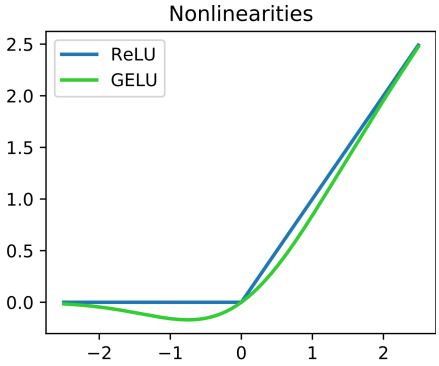


FIGURE: Image Courtesy: Wikipedia

- GELU (Gaussian-error linear unit)

$$GELU(x) = x \cdot \Phi(x)$$

$$\frac{\partial}{\partial x} GELU(x) = x \cdot \Phi'(x) + \Phi(x)$$

where $\Phi(x) = Pr(X \leqslant x)$ is the cumulative Gaussian distribution.

OUTLINE

VANISHING/EXPLODING GRADIENT PROBLEM

Hochreiter's work formally identified a major reason: Typical deep NNs suffer from the now famous problem of vanishing or exploding gradients. With standard activation functions (Sec. 1), cumulative backpropagated error signals (Sec. 5.5.1) either shrink rapidly, or grow out of bounds. In fact, they decay exponentially in the number of layers or CAP depth (Sec. 3), or they explode. This is also known as the long time lag problem.

See also: Deep Learning by Jurgen Schmidhuber

GATING HISTORY

- Gating was introduced in the LSTM paper in ’97, in order to address vanishing/exploding gradient problem.
- Simply put, gating mechanism is element-wise multiplication of input vector with a gate-activation vector.

- The gate, in turn, is activated by looking at the input vector itself. For example, a basic gate would be formulated as,

$$\mathbf{y} = \mathbf{g} \otimes \mathbf{x}$$
$$\mathbf{g} = \sigma_{\otimes}(W\mathbf{x} + \mathbf{b})$$

where,
$\sigma_{\otimes}(\mathbf{x})$ is the element-wise sigmoid activation of input vector $\mathbf{x}$; and
$\otimes$ represents element-wise multiplication.

For a more involved use-case, let an RNN be defined for T time steps, with

- Given inputs as $\{\mathbf{z}_1, \ldots, \mathbf{z}_T\}$;
- Cell States, $\{\mathbf{c}_1, \ldots, \mathbf{c}_T\}$;
- Hidden States, $\{\mathbf{h}_1, \ldots, \mathbf{h}_T\}$;
- Given initial states as $\mathbf{c}_0, \mathbf{h}_0$;
- Neural Network $\Phi(\mathbf{z}, \mathbf{c}, \mathbf{h})$ to compute pre gate activation;

GATING HISTORY (LSTM)

$\forall t \in \{1, \ldots, T\}$,

$$
\begin{aligned}
\mathbf{x} &\leftarrow \Phi(\mathbf{z}_t, \mathbf{c}_{t-1}, \mathbf{h}_{t-1}) \\
\mathbf{i} &\leftarrow \sigma_{\otimes}(W_i \mathbf{x} + U_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f} &\leftarrow \sigma_{\otimes}(W_f \mathbf{x} + U_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o} &\leftarrow \sigma_{\otimes}(W_o \mathbf{x} + U_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\mathbf{g} &\leftarrow \tanh_{\otimes}(W_g \mathbf{x} + U_g \mathbf{h}_{t-1} + \mathbf{b}_g) \\
\mathbf{c}_t &\leftarrow \mathbf{f} \otimes \mathbf{c}_{t-1} + \mathbf{i} \otimes \mathbf{g} \\
\mathbf{h}_t &\leftarrow \mathbf{o} \otimes \tanh_{\otimes} \mathbf{c}_t
\end{aligned}
$$

GATING HISTORY (GRU)

$\forall t \in \{1, \ldots, T\}$,

$$
\begin{aligned}
\mathbf{x} &\leftarrow \Phi(\mathbf{z}_t, \mathbf{c}_{t-1}, \mathbf{h}_{t-1}) \\
\mathbf{r} &\leftarrow \sigma_{\otimes}(W_r \mathbf{x} + U_r \mathbf{h}_{t-1} + \mathbf{b}_r) \\
\tilde{\mathbf{h}} &\leftarrow \tanh_{\otimes}(W_h \mathbf{x} + U_h(\mathbf{r} \otimes \mathbf{h}_{t-1}) + \mathbf{b}_h) \\
\mathbf{c}_t &\leftarrow \sigma_{\otimes}(W_c \mathbf{x} + U_c \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{h}_t &\leftarrow \mathbf{c}_t \otimes \mathbf{h}_{t-1} + (1 - \mathbf{c}_t) \otimes \tilde{\mathbf{h}}
\end{aligned}
$$

GATING HISTORY (READ MORE)

See also:

- [Medium] Gating Mechanisms (Blog by Eugene Shevchenko);
- [arXiv] Jacobian Spectrum of Gates (Fig.1; Theory of Gating)

GATED LINEAR UNIT

In the context of speech processing, let

$$\tilde{X} = W * X;$$
$$\tilde{X} \in \mathbb{R}^{n \times (\cdot)},$$
$$W \in \mathbb{R}^{n \times m \times k},$$
$$X \in \mathbb{R}^{m \times (\cdot)}$$

represent a 1-D convolution operation with kernel size k, input filters m and output filters n.

GATED LINEAR UNIT

A gated linear unit (GLU) wraps a convolution layer with a linear activation and sigmoid gate as follows,

$$h_l(X) = (W * X + B) \otimes \sigma_{\otimes}(V * X + C)$$

Since the element-wise multiplication is a symmetric operation, this may as well be interpreted as a linear gate over a sigmoid activation.

GATED LINEAR UNIT

With hardware acceleration, this operation may be implemented with single parallelised convolution operations with double filter size, namely $W \in \mathbb{R}^{2n \times m \times k}$, and bias $B \in \mathbb{R}^{2n \times (\cdot)}$, as follows,

$$\tilde{X} = W * X + B$$
$$h_l(X) = \tilde{X}_{:n} \otimes \sigma_{\otimes}(\tilde{X}_{n:})$$

See also: Gated Conv-Net Paper [arXiv]

GATED ACTIVATION UNIT

A gated activation unit (GLU) wraps a convolution layer with a hyperbolic tangent activation and sigmoid gate as follows,

$$\tilde{X} = W * X + B$$
$$h_l(X) = \tanh_{\otimes}(\tilde{X}_{:n}) \otimes \sigma_{\otimes}(\tilde{X}_{n:})$$

Since the element-wise multiplication is a symmetric operation, this may equally well be interpreted as a hyperbolic tangent gate and sigmoid activation.
See also: Conditional PixelCNN Paper [NeurIPS '16]

OUTLINE

WORD ERROR RATE

- Word Error Rate is inspired by “word recognition” accuracy measure in *cognitive psychology*, which is “the ability of a reader to recognize written words correctly and virtually effortlessly.”

WORD ERROR RATE

- The experiments generally test the ability to recognise "isolated words," without additional contextual information. (*Trivia*: testing whose ability, the reader's or the model's?)

WORD ERROR RATE

- WER is a special type of normalised edit distance; computed as
  - the normalised number operations
  - required to transform reference *(target)* to hypothesis *(prediction)*.
  - The set of operations consist of substitution, deletion and insertion.

WORD ERROR RATE

Formally, if Y is the reference set and Y' is the hypothesis,

$$\mathrm{WER}(Y \rightarrow Y') = \frac{|Y' \setminus Y| + [|Y| - |Y'|]_+}{|Y|}$$

where $\setminus$ is the set difference operator.

- Intuitively, we resolve for two cases, *i.e.* either Y' is larger than Y or otherwise.
- In case of former, $Y' \setminus Y$ would include the set of substitutions as well as insertions.
- In case of latter, $Y' \setminus Y$ would include the set of substitutions only; hence, the additional term of difference in size is added to account for the number of deletions.
- The denominator is a normalisation factor.

OUTLINE

EXERCISE

1. If the following equation describes the jasper model,

$$\theta_* = \arg\min_{\theta} \mathop{\mathbb{E}}_{(X,Y)\sim\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}} [\Delta(Y, \mathcal{F}(X;\theta))]$$

Define X, Y, $\mathcal{F}$, θ, θ_*.

EXERCISE

2. Formally define $\mathrm{MSE}(\mathbf{y}, \widetilde{\mathbf{y}})$ as mean-squared error measure between target and prediction vectors.

EXERCISE

3. What are the conditions under which $\mathrm{MSE}(\mathbf{y}, \widetilde{\mathbf{y}}) \equiv \Delta_E(\mathbf{y}, \widetilde{\mathbf{y}}) \equiv \|\mathbf{y} - \widetilde{\mathbf{y}}\|_F^2$?

EXERCISE

4. How is a CTC Loss different from MSE Loss as a training objective?

EXERCISE

5. In order to learn a model for sequence-to-sequence mapping like speech to text, recommend whether to use Softmax-Cross-Entropy Classification Loss or Connectionist Temporal Classification Loss. Also justify your recommendation.