# Tensor2Tensor Transformers

## New Deep Models for NLP

Łukasz Kaiser

Joint work with Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, Jakob Uszkoreit, Ashish Vaswani.

# RNNs Everywhere

Very successful for variable-length representations

Sequences (e.g. language), images, …

Gating (LSTM, GRU) for long-range error propagation

At the core of seq2seq (w/ attention)

# But…

Sequentiality prohibits parallelization within instances

Long-range dependencies still tricky, despite gating

**Many modalities are hierarchical-ish (e.g. language)**

**RNNs (w/ sequence-aligned states) are wasteful!**

# CNNs?

Trivial to parallelize (per layer)

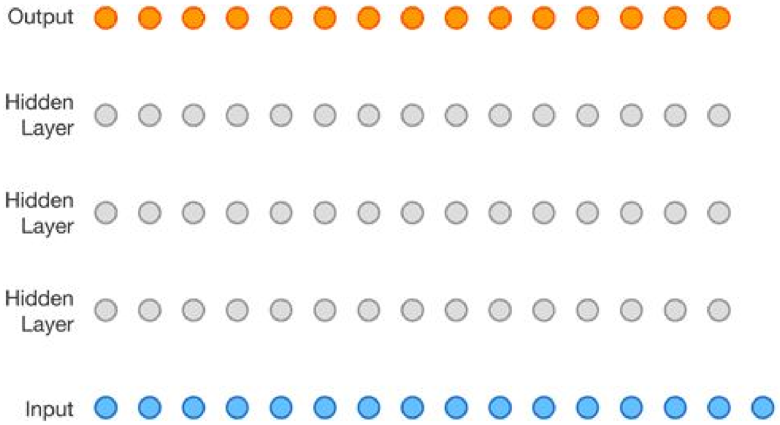
Fit intuition that most dependencies are local

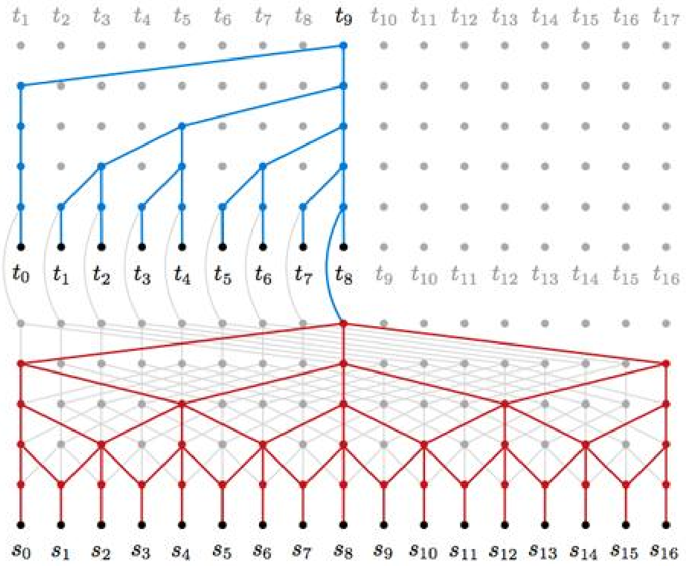
**Path length between positions can be logarithmic**

**when using dilated convolutions, left-padding for text.**

# Auto-Regressive CNNs

## WaveNet and ByteNet



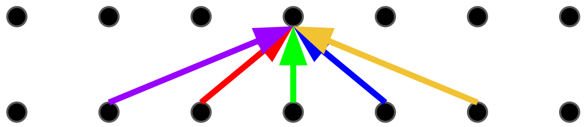
# Attention

Attention between encoder and decoder is crucial in NMT

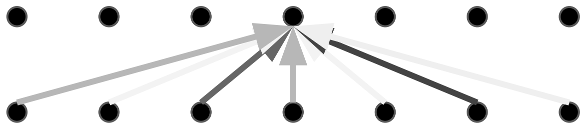
**Why not use (self-)attention for the representations?**

# Self-Attention

Convolution



Self-Attention



# Self-Attention

Constant path length between any two positions

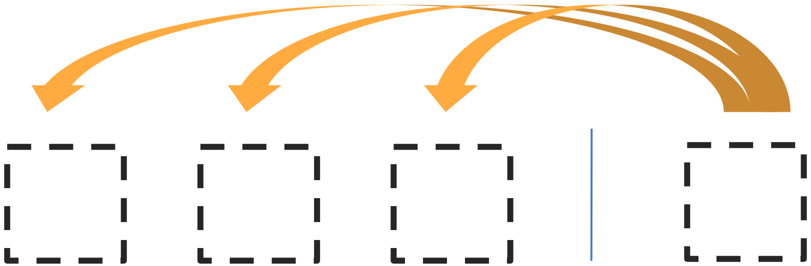
Variable-sized perceptive field

Gating/multiplication enables crisp error propagation

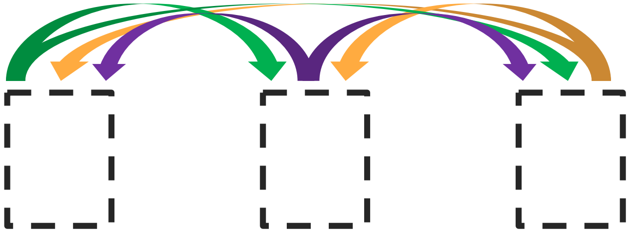
Trivial to parallelize (per layer)

**Can replace sequence-aligned recurrence entirely**

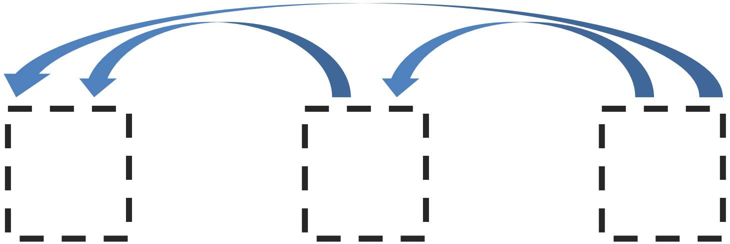
# Three ways of attention



Encoder-Decoder Attention

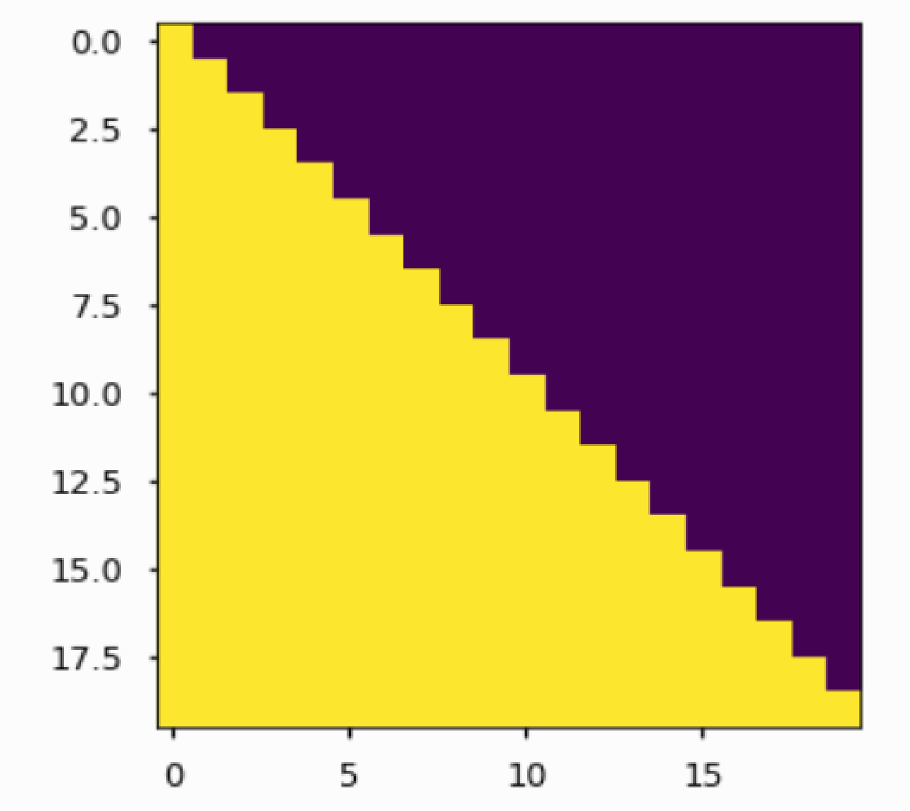


Encoder Self-Attention



MaskedDecoder Self-Attention

# Attention Mask



# The Transformer

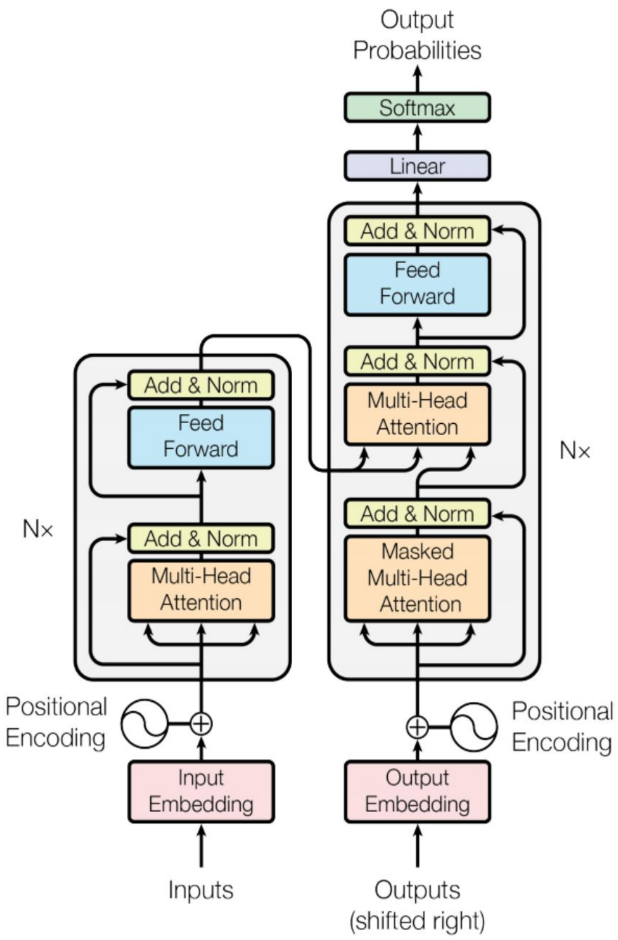
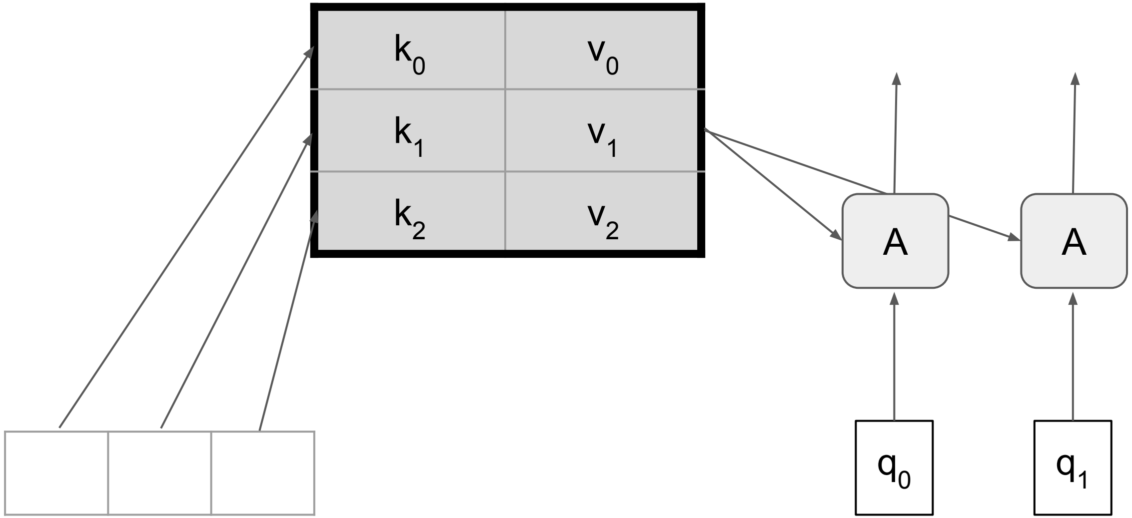



Figure 1: The Transformer - model architecture.

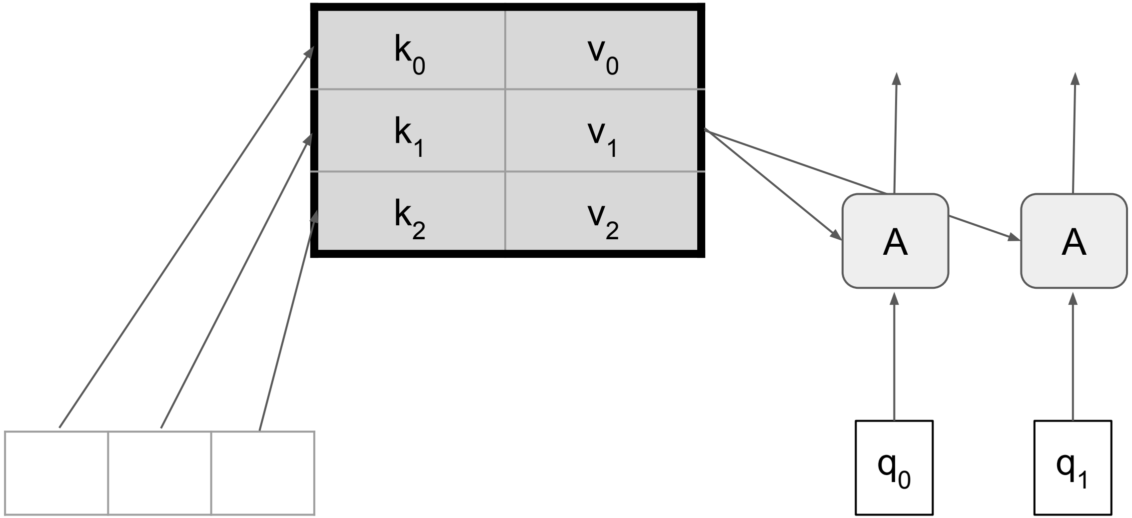
# Dot-Product Attention

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$



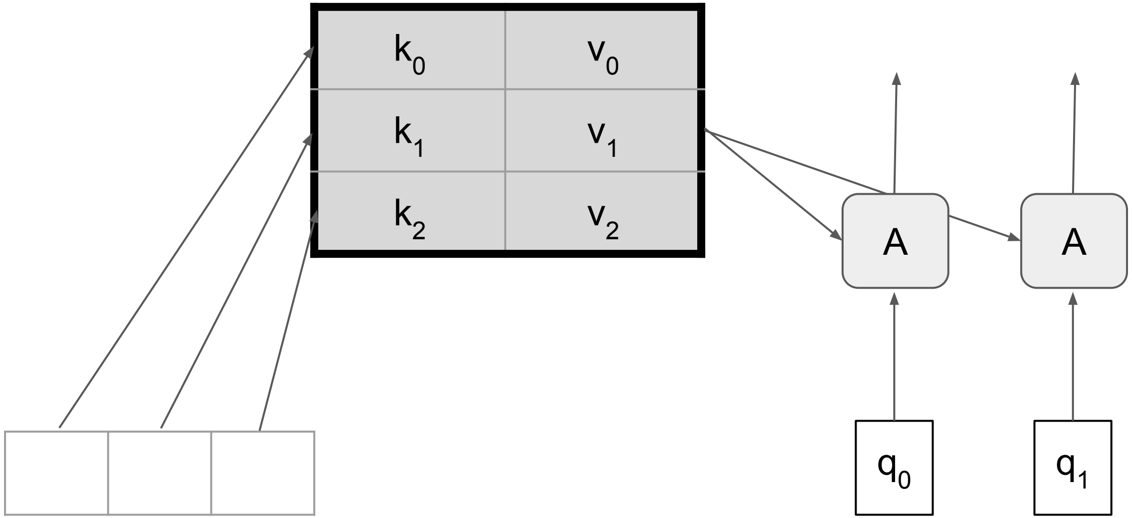
# Dot-Product Attention

$$A(Q, K, V) = softmax(QK^T)V$$



# Scaled Dot-Product Attention:

$$A(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

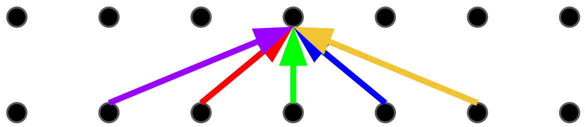


| | Ops | Activations |
|---|---|---|
| Attention (dot-prod) | $n^2 \cdot d$ | $n^2 + n \cdot d$ |
| Attention (additive) | $n^2 \cdot d$ | $n^2 \cdot d$ |
| Recurrent | $n \cdot d^2$ | $n \cdot d$ |
| Convolutional | $n \cdot d^2$ | $n \cdot d$ |

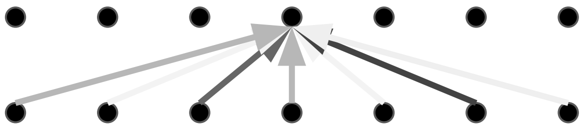
n = sequence length d = depth k = kernel size

# What's missing from Self-Attention?

Convolution



Self-Attention

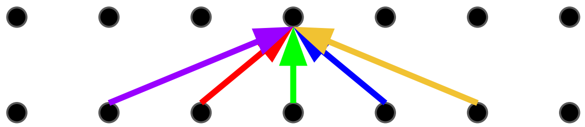


# What’s missing from Self-Attention?

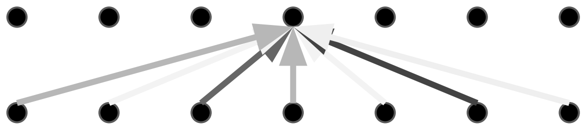
- Convolution: a different linear transformation for each relative position. Allows you to distinguish what information came from where.

- Self-Attention: a weighted average :(

Convolution



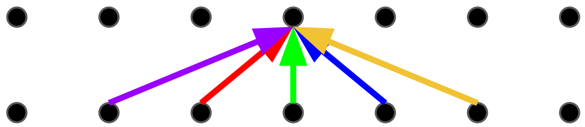
Self-Attention



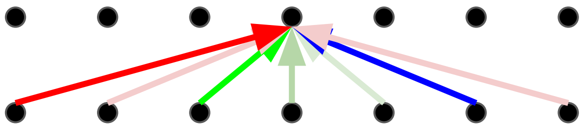
# The Fix: Multi-Head Attention

- Multiple attention layers (heads) in parallel (shown by different colors)
- Each head uses different linear transformations.
- Different heads can learn different relationships.

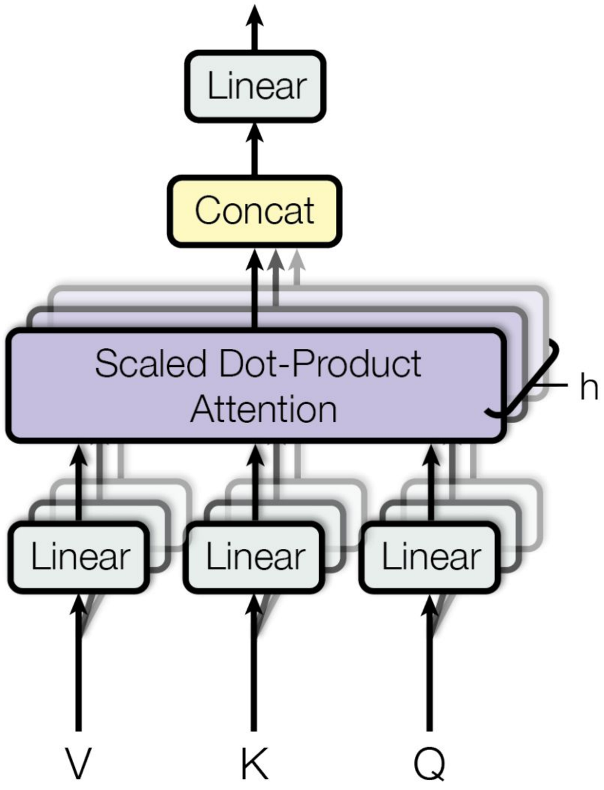
## Convolution



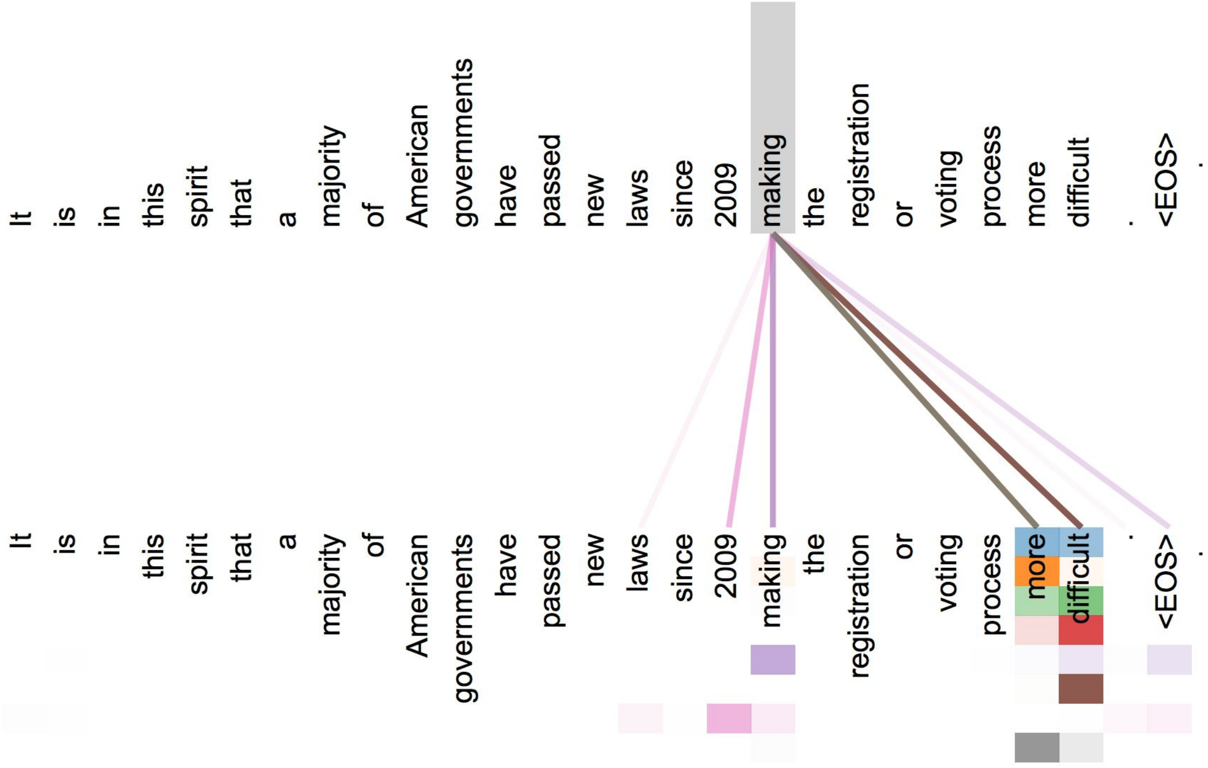
## Multi-Head Attention



# The Fix: Multi-Head Attention



# The Fix: Multi-Head Attention



| | Ops | Activations |
|---|---|---|
| Multi-Head Attention<br>with linear transformations.<br>For each of the h heads,<br>$d_q = d_k = d_v = d/h$ | $n^2 \cdot d + n \cdot d^2$ | $n^2 \cdot h + n \cdot d$ |
| Recurrent | $n \cdot d^2$ | $n \cdot d$ |
| Convolutional | $n \cdot d^2$ | $n \cdot d$ |

$n$ = sequence length $d$ = depth $k$ = kernel size

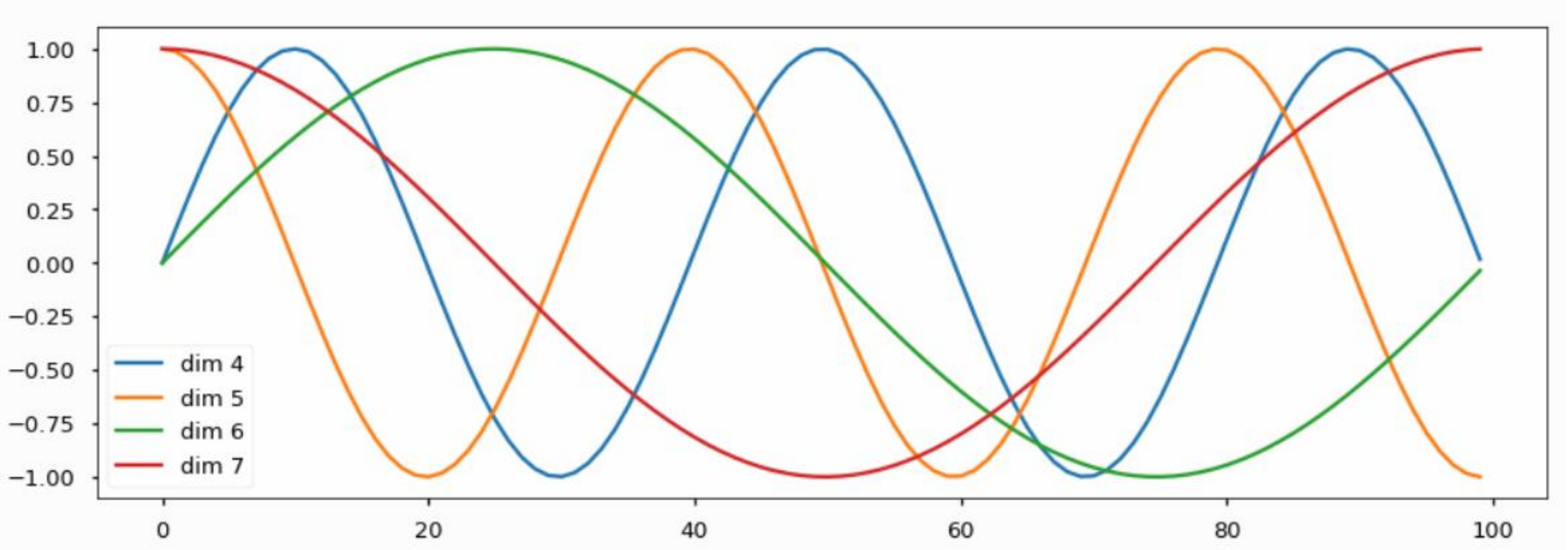
# Positional Encoding

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

For any fixed offset k,

$PE_{(pos+k)}$ can be represented as a linear function of $PE_{pos}$.

# Positional Encoding



# Why Self-Attention?

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. $n$ is the sequence length, $d$ is the representation dimension, $k$ is the kernel size of convolutions and $r$ the size of the neighborhood in restricted self-attention.

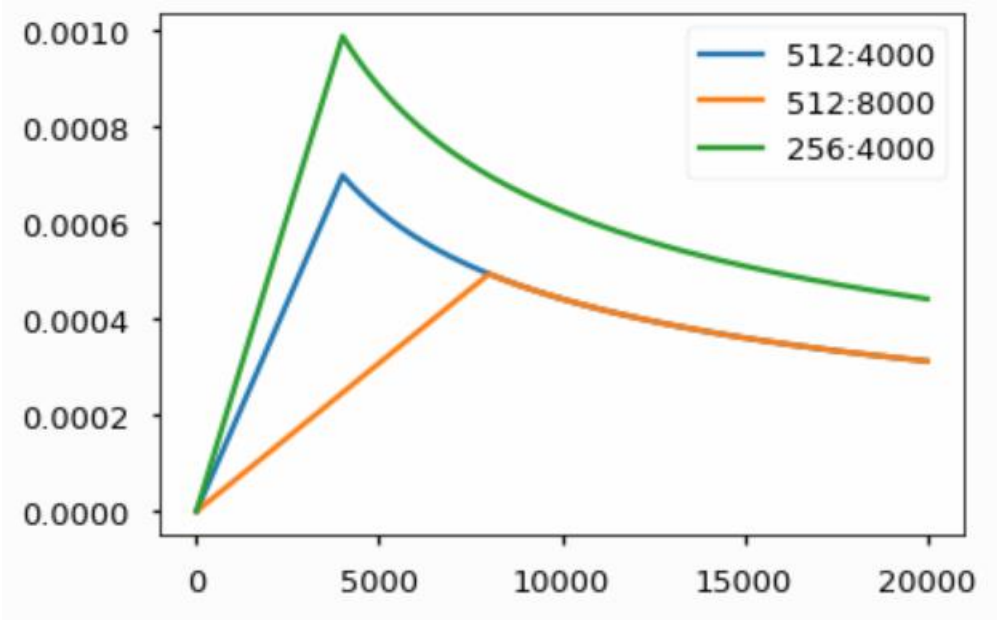
| Layer Type | Complexity per Layer | Sequential Operations | Maximum Path Length |
|---|---|---|---|
| Self-Attention | $O(n^2 \cdot d)$ | $O(1)$ | $O(1)$ |
| Recurrent | $O(n \cdot d^2)$ | $O(n)$ | $O(n)$ |
| Convolutional | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(log_k(n))$ |
| Self-Attention (restricted) | $O(r \cdot n \cdot d)$ | $O(1)$ | $O(n/r)$ |

# Training and Decoding - usual tricks

- ADAM optimizer with learning rate proportional to ($step^{-0.5}$)
- Dropout during training at every layer just before adding residual
- Label smoothing
- Auto-regressive decoding with beam search and length penalties
- Checkpoint-averaging

# Optimizer

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$



# Machine Translation Results: WMT-14

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [17] | 23.75 | | | |
| Deep-Att + PosUnk [37] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [36] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [31] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [37] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [36] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | $\mathbf{3.3 \cdot 10^{18}}$ | |
| Transformer (big) | **28.4** | **41.0** | $2.3 \cdot 10^{19}$ | |

# Ablations

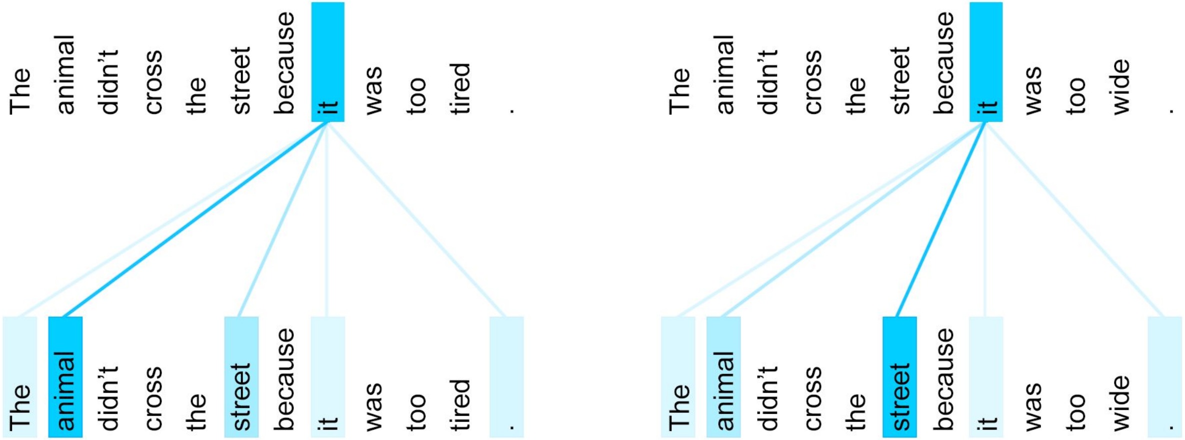
| | $N$ | $d_{\text{model}}$ | $d_{\text{ff}}$ | $h$ | $d_k$ | $d_v$ | $P_{drop}$ | $\epsilon_{ls}$ | train steps | PPL (dev) | BLEU (dev) | params $\times 10^6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| base | 6 | 512 | 2048 | 8 | 64 | 64 | 0.1 | 0.1 | 100K | 4.92 | 25.8 | 65 |
| (A) | | | | 1 | 512 | 512 | | | | 5.29 | 24.9 | |
| | | | | 4 | 128 | 128 | | | | 5.00 | 25.5 | |
| | | | | 16 | 32 | 32 | | | | 4.91 | 25.8 | |
| | | | | 32 | 16 | 16 | | | | 5.01 | 25.4 | |
| (B) | | | | | 16 | | | | | 5.16 | 25.1 | 58 |
| | | | | | 32 | | | | | 5.01 | 25.4 | 60 |
| (C) | 2 | | | | | | | | | 6.11 | 23.7 | 36 |
| | 4 | | | | | | | | | 5.19 | 25.3 | 50 |
| | 8 | | | | | | | | | 4.88 | 25.5 | 80 |
| | | 256 | | | 32 | 32 | | | | 5.75 | 24.5 | 28 |
| | | 1024 | | | 128 | 128 | | | | 4.66 | 26.0 | 168 |
| | | | 1024 | | | | | | | 5.12 | 25.4 | 53 |
| | | | 4096 | | | | | | | 4.75 | 26.2 | 90 |
| (D) | | | | | | | 0.0 | | | 5.77 | 24.6 | |
| | | | | | | | 0.2 | | | 4.95 | 25.5 | |
| | | | | | | | | 0.0 | | 4.67 | 25.3 | |
| | | | | | | | | 0.2 | | 5.47 | 25.7 | |
| (E) | positional embedding instead of sinusoids | | | | | | | | | 4.92 | 25.7 | |
| big | 6 | 1024 | 4096 | 16 | | | 0.3 | | 300K | **4.33** | **26.4** | 213 |

# Constituency Parser

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

| Parser | Training | WSJ 23 F1 |
|---|---|---|
| Vinyals & Kaiser el al. (2014) [37] | WSJ only, discriminative | 88.3 |
| Petrov et al. (2006) [29] | WSJ only, discriminative | 90.4 |
| Zhu et al. (2013) [40] | WSJ only, discriminative | 90.4 |
| Dyer et al. (2016) [8] | WSJ only, discriminative | 91.7 |
| Transformer (4 layers) | WSJ only, discriminative | 91.3 |
| Zhu et al. (2013) [40] | semi-supervised | 91.3 |
| Huang & Harper (2009) [14] | semi-supervised | 91.3 |
| McClosky et al. (2006) [26] | semi-supervised | 92.1 |
| Vinyals & Kaiser el al. (2014) [37] | semi-supervised | 92.1 |
| Transformer (4 layers) | semi-supervised | 92.7 |
| Luong et al. (2015) [23] | multi-task | 93.0 |
| Dyer et al. (2016) [8] | generative | 93.3 |

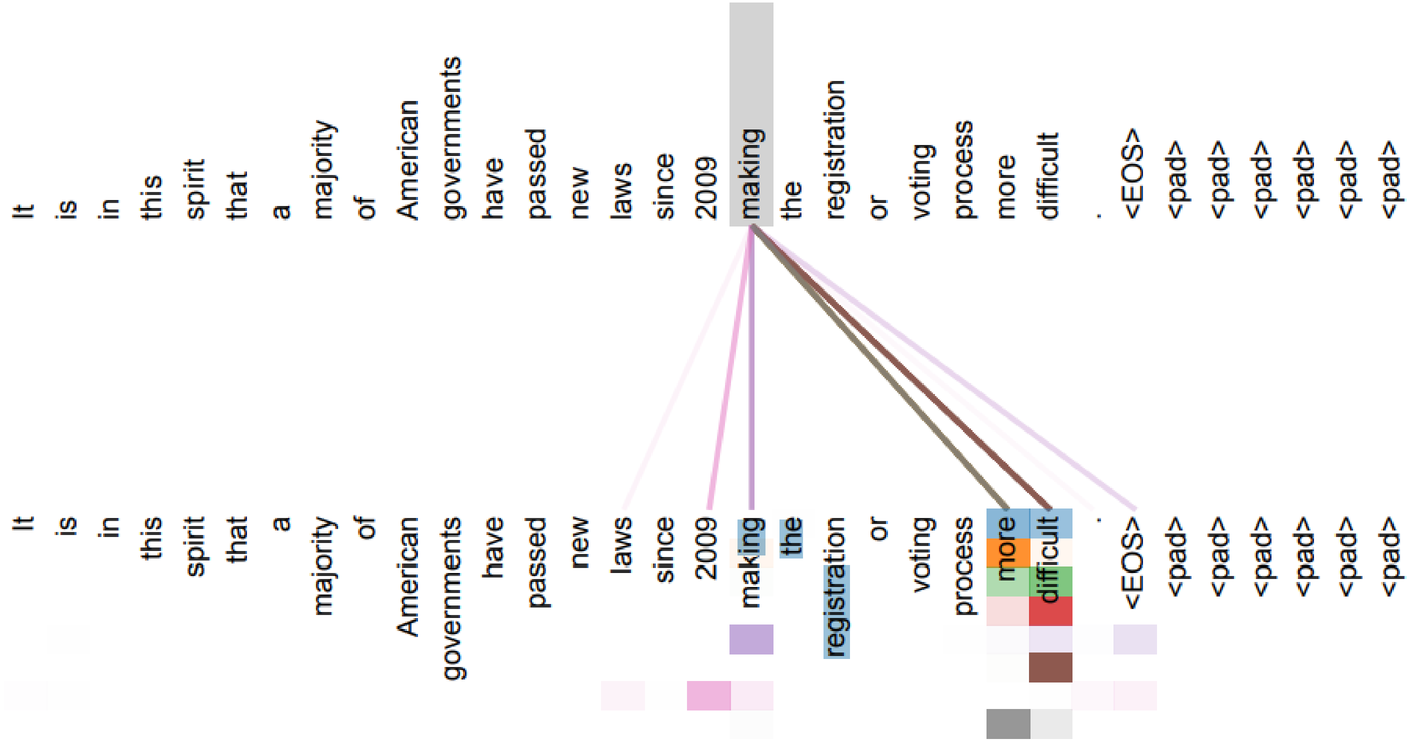
# Coreference resolution (Winograd schemas)



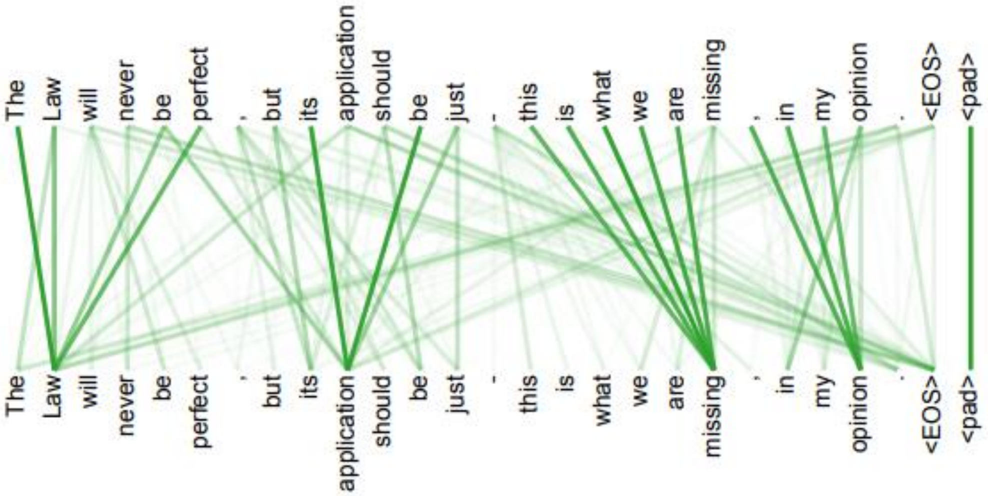
# Coreference resolution (Winograd schemas)

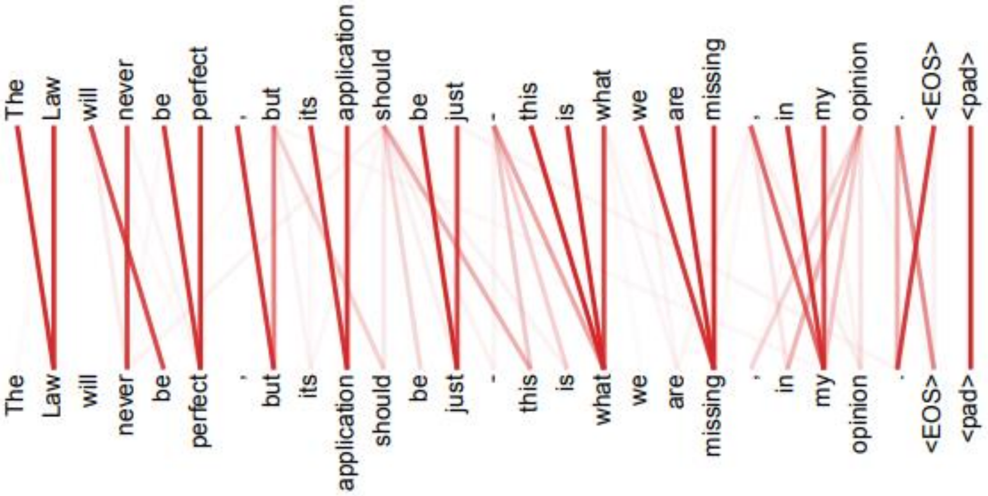
| Sentence | Google Translate | Transformer |
|---|---|---|
| The cow ate the hay because it was **delicious**. | La vache mangeait le foin parce **qu'elle** était délicieuse. | La vache a mangé le foin parce **qu'il** était délicieux. |
| The cow ate the hay because it was **hungry**. | La vache mangeait le foin parce **qu'elle** avait faim. | La vache mangeait le foin parce **qu'elle** avait faim. |
| The women stopped drinking the wines because they were **carcinogenic**. | Les femmes ont cessé de boire les vins parce **qu'ils** étaient cancérogènes. | Les femmes ont cessé de boire les vins parce **qu'ils** étaient cancérigènes. |
| The women stopped drinking the wines because they were **pregnant**. | Les femmes ont cessé de boire les vins parce **qu'ils** étaient enceintes. | Les femmes ont cessé de boire les vins parce **qu'elles** étaient enceintes. |
| The city councilmen refused the female demonstrators a permit because they **advocated** violence. | Les conseillers municipaux ont refusé aux femmes manifestantes un permis parce **qu'ils** préconisaient la violence. | Le conseil municipal a refusé aux manifestantes un permis parce **qu'elles** prônaient la violence. |
| The city councilmen refused the female demonstrators a permit because they **feared** violence. | Les conseillers municipaux ont refusé aux femmes manifestantes un permis parce **qu'ils** craignaient la violence | Le conseil municipal a refusé aux manifestantes un permis parce **qu'elles** craignaient la violence.* |

# Attention Visualization



# Attention Vis



# Tensor2Tensor Library

https://github.com/tensorflow/tensor2tensor

Transformer (Attention is All You Need)
MultiModel (One Model to Learn Them All)
SliceNet
NeuralGPU
ByteNet, Xception, LSTM, ...

# Tensor2Tensor Baselines

Finally a good single-gpu few-days translation model!

```
pip install tensor2tensor && t2t-trainer \
  --generate_data \
  --data_dir=~/t2t_data \
  --problems=wmt_ende_tokens_32k \
  --model=transformer \
  --hparams_set=transformer_base_single_gpu \
  --output_dir=~/t2t_train/base \
  --decode_interactive
```

# Join Tensor2Tensor
# add datasets and models

https://github.com/tensorflow/tensor2tensor

```
pip install tensor2tensor && t2t-trainer \
  --generate_data \
  --data_dir=~/t2t_data \
  --problems=wmt_ende_tokens_32k \
  --model=transformer \
  --hparams_set=transformer_base_single_gpu \
  --output_dir=~/t2t_train/base \
  --decode_interactive
```

Thank you for your attention