Self-Attention For Generative Models

Ashish Vaswani and Anna Huang

Learning Representations of Variable Length Data

Basic building block of sequence-to-sequence learning

Neural machine translation, summarization, QA, …
Recurrent Neural Networks

Model of choice for learning variable-length representations.

Natural fit for sentences and sequences of pixels.

LSTMs, GRUs and variants dominate recurrent models.
Recurrent Neural Networks

Let’s represent this sentence,
But...

Sequential computation inhibits parallelization.

No explicit modeling of long and short range dependencies.

We want to model hierarchy.

**RNNs (w/ sequence-aligned states) seem wasteful!**
Convolutional Neural Networks?

Let's represent this sentence,
Convolutional Neural Networks?

Trivial to parallelize (per layer).

Exploits local dependencies

‘Interaction distance’ between positions linear or logarithmic.

**Long-distance dependencies require many layers.**
Attention

Attention between encoder and decoder is crucial in NMT.

Why not use attention for representations?
Self-Attention

Let's represent this sentence,
Text generation
Self-Attention

Constant ‘path length’ between any two positions.

Gating/multiplicative interactions.

Trivial to parallelize (per layer).

Can replace sequential computation entirely?
Previous work

**Classification & regression with self-attention:**
Parikh et al. (2016), Lin et al. (2016)

**Self-attention with RNNs:**
Long et al. (2016), Shao, Gows et al. (2017)

**Recurrent attention:**
Sukhbaatar et al. (2015)
The Transformer

Let's represent this sentence,

Representieren wir diesen Satz,
Encoder Self-Attention

\[ A(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \]
Decoder Self-Attention

Value

Key

Query

matmul_K

matmul_Q

d_1
d_2

dot-prod

softmax

matmul_V

d_1

d_2

d_3

d_4

d'_2

...
Attention is Cheap!

<table>
<thead>
<tr>
<th>Method</th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Attention</td>
<td>$O(\text{length}^2 \cdot \text{dim})$</td>
</tr>
<tr>
<td>RNN (LSTM)</td>
<td>$O(\text{length} \cdot \text{dim}^2)$</td>
</tr>
<tr>
<td>Convolution</td>
<td>$O(\text{length} \cdot \text{dim}^2 \cdot \text{kernel_width})$</td>
</tr>
</tbody>
</table>
## FLOPs

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Attention</td>
<td>$O(\text{length}^2 \cdot \text{dim})$</td>
<td>$4 \cdot 10^9$</td>
</tr>
<tr>
<td>RNN (LSTM)</td>
<td>$O(\text{length} \cdot \text{dim}^2)$</td>
<td>$16 \cdot 10^9$</td>
</tr>
<tr>
<td>Convolution</td>
<td>$O(\text{length} \cdot \text{dim}^2 \cdot \text{kernel}_\text{width})$</td>
<td>$6 \cdot 10^9$</td>
</tr>
</tbody>
</table>

- length = 1000
- dim = 1000
- kernel_width = 3

**Attention is Cheap!**
Attention: a weighted average

The cat stuck out its tongue and licked its owner.
I kicked the ball.
I kicked the ball.
Parallel attention heads

I kicked the ball

Who did what?

To whom?
Attention head: Who

I kicked the ball

Who did what?

To whom?
Parallel attention heads

I kicked the ball

Who did what?

I kicked the ball
Parallel attention heads

I kicked the ball

Who Did what?

To whom?

I kicked the ball
Parallel attention heads

I kicked the ball

Who Did what?

To whom?

I kicked the ball
Self-Attention: Averaging

I kicked the ball. Who did what to whom?
Attention head: Who
Attention head: Did What?

I kicked the ball?
Attention head: To Whom?

Who

Did what?

kicked

To whom?

I

kicked

the

ball
Multihead Attention

Who did what?

To whom?

I kicked the ball.
Convolution:
Different linear transformations by relative position.

The cat stuck out its tongue and licked its owner.
Attention: a weighted average

The cat stuck out its tongue and licked its owner.
Multi-head Attention

Parallel attention layers with different linear transformations on input and output.

The cat stuck out its tongue and licked its owner.
Results
## Machine Translation: WMT-2014 BLEU

<table>
<thead>
<tr>
<th></th>
<th>EN-DE</th>
<th>EN-FR</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNMT (orig)</td>
<td>24.6</td>
<td>39.9</td>
</tr>
<tr>
<td>ConvSeq2Seq</td>
<td>25.2</td>
<td>40.5</td>
</tr>
<tr>
<td>Transformer*</td>
<td><strong>28.4</strong></td>
<td><strong>41.8</strong></td>
</tr>
</tbody>
</table>

*Transformer models trained >3x faster than the others.*

Frameworks:

tensor2tensor
Sockeye
Importance of residuals

Figure 1: The Transformer - model architecture.
Importance of Residuals

Residuals carry positional information to higher layers, among other information.

With residuals

Without residuals

Without residuals, with timing signals
Training Details

- ADAM optimizer with a learning rate warmup (warmup + exponential decay)
- Dropout during training at every layer just before adding residual
- Layer-norm
- Attention dropout (for some experiments)
- Checkpoint-averaging
- Label smoothing
- Auto-regressive decoding with beam search and length biasing
  …
## What Matters?

<table>
<thead>
<tr>
<th></th>
<th>$N$</th>
<th>$d_{\text{model}}$</th>
<th>$d_{\text{ff}}$</th>
<th>$h$</th>
<th>$d_k$</th>
<th>$d_v$</th>
<th>$P_{\text{drop}}$</th>
<th>$\epsilon_{18}$</th>
<th>train steps</th>
<th>PPL (dev)</th>
<th>BLEU (dev)</th>
<th>params $\times 10^6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>6</td>
<td>512</td>
<td>2048</td>
<td>8</td>
<td>64</td>
<td>64</td>
<td>0.1</td>
<td>0.1</td>
<td>100K</td>
<td>4.92</td>
<td>25.8</td>
<td>65</td>
</tr>
<tr>
<td>(A)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.29</td>
<td>24.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.00</td>
<td>25.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.91</td>
<td>25.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.01</td>
<td>25.4</td>
<td></td>
</tr>
<tr>
<td>(B)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.16</td>
<td>25.1</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.01</td>
<td>25.4</td>
<td>60</td>
</tr>
<tr>
<td>(C)</td>
<td>2</td>
<td>256</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>128</td>
<td>128</td>
<td></td>
<td>6.11</td>
<td>23.7</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.19</td>
<td>25.3</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.88</td>
<td>25.5</td>
<td>80</td>
</tr>
<tr>
<td>(D)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.75</td>
<td>24.5</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.66</td>
<td>26.0</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.12</td>
<td>25.4</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.75</td>
<td>26.2</td>
<td>90</td>
</tr>
<tr>
<td>(E)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>positional embedding instead of sinusoids</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>big</td>
<td>6</td>
<td>1024</td>
<td>4096</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.33</td>
<td>26.4</td>
<td>213</td>
</tr>
</tbody>
</table>

---

### Note

- **$N$** represents the model size.
- **$d_{\text{model}}$** is the model dimensionality.
- **$d_{\text{ff}}$** is the dimensionality of the feed-forward network.
- **$h$** is the number of hidden layers.
- **$d_k$** and **$d_v$** are the dimensions of the key and value vectors, respectively.
- **$P_{\text{drop}}$** is the dropout probability.
- **$\epsilon_{18}$** is a parameter.
- **train steps** is the number of training steps.
- **PPL (dev)** is the perplexity on the development set.
- **BLEU (dev)** is the BLEU score on the development set.
- **params $\times 10^6$** is the number of model parameters.
Generating Wikipedia by Summarizing Long Sequences

msaleh@ et al. submission to ICLR’18

<table>
<thead>
<tr>
<th>Model</th>
<th>ROUGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>seq2seq-attention</td>
<td>12.7</td>
</tr>
<tr>
<td>Transformer-ED (L=500)</td>
<td>34.2</td>
</tr>
<tr>
<td>Transformer-DMCA (L=11000)</td>
<td><strong>36.2</strong></td>
</tr>
</tbody>
</table>
Self-Similarity, Image and Music Generation
Self-similarity in images

Self-Similarity in Images

Starry Night (Van Gogh, June 1889)
Self-similarity in music

Motifs repeat, immediately and also at a distance
Probabilistic Image Generation

Model the joint distribution of pixels

Turning it into a sequence modeling problem

Assigning probabilities allows measuring generalization
Probabilistic Image Generation

RNNs and CNNs are state-of-the-art (PixelRNN, PixelCNN)

CNNs incorporating gating now match RNNs in quality

CNNs are much faster due to parallelization

A Oord et al. (2016), Salimans et al. (2017), Kalchbrenner et al. (2016)
Probabilistic Image Generation

Long-range dependencies matter for images (e.g. symmetry)

Likely increasingly important with increasing image size

Modeling long-range dependencies with CNNs requires either

- **Many layers** likely making training harder

- **Large kernels** at large parameter/computational cost
Texture Synthesis with Self-Similarity

Texture Synthesis by Non-parametric Sampling (Efros and Leung, 1999)
Non-local Means

Figure 1. Scheme of NL-means strategy. Similar pixel neighborhoods give a large weight, $w(p,q1)$ and $w(p,q2)$, while much different neighborhoods give a small weight $w(p,q3)$. 
Non-local Means

A Non-local Algorithm for Image Denoising (Buades, Coll, and Morel. CVPR 2005)

Non-local Neural Networks (Wang et al., 2018)
Previous work

**Self-attention:**
Parikh et al. (2016), Lin et al. (2016), Vaswani et al. (2017)

**Autoregressive Image Generation:**
A Oord et al. (2016), Salimans et al. (2017)
Self-Attention

\[ A(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \]
Decoder Self-Attention
Attention is Cheap!

<table>
<thead>
<tr>
<th></th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Attention</td>
<td>$O(\text{length}^2 \cdot \text{dim})$</td>
</tr>
<tr>
<td>RNN (LSTM)</td>
<td>$O(\text{length} \cdot \text{dim}^2)$</td>
</tr>
<tr>
<td>Convolution</td>
<td>$O(\text{length} \cdot \text{dim}^2 \cdot \text{kernel_width})$</td>
</tr>
</tbody>
</table>
Attention is Cheap if length $<<$ dim!

<table>
<thead>
<tr>
<th>Method</th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Attention</td>
<td>$O(length^2 \cdot dim)$ ($length=3072$ for images)</td>
</tr>
<tr>
<td>RNN (LSTM)</td>
<td>$O(length \cdot dim^2)$</td>
</tr>
<tr>
<td>Convolution</td>
<td>$O(length \cdot dim^2 \cdot kernel_width)$</td>
</tr>
</tbody>
</table>
Combining Locality with Self-Attention

Restrict the attention windows to be local neighborhoods

Good assumption for images because of spatial locality
Local 1D Attention

Memory Block

Query Block

q
Image Transformer Layer
Tasks

Super-resolution

Unconditional and Conditional Image generation
Results

Image Transformer
Parmar*, Vaswani*, Uszkoreit, Kaiser, Shazeer, Ku, and Tran. ICML 2018
## Unconditional Image Generation

<table>
<thead>
<tr>
<th>Model</th>
<th>Cifar-10 (Test)</th>
<th>Imagenet (Validation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PixelRNN</td>
<td>3.00</td>
<td>3.86</td>
</tr>
<tr>
<td>Gated PixelCNN</td>
<td>3.03</td>
<td>3.83</td>
</tr>
<tr>
<td>PixelCNN++</td>
<td>2.92 (dmol)</td>
<td>-</td>
</tr>
<tr>
<td>PixelSNAIL</td>
<td><strong>2.85</strong></td>
<td>3.8</td>
</tr>
<tr>
<td>Image Transformer, 1D local</td>
<td>2.9 (xent)</td>
<td><strong>3.77</strong></td>
</tr>
<tr>
<td>Image Transformer, 1D local</td>
<td>2.9 (dmol)</td>
<td>3.78</td>
</tr>
</tbody>
</table>

Cross entropy of various models on CIFAR-10 and Imagenet datasets.
Cifar10 Samples
CelebA Super Resolution

<table>
<thead>
<tr>
<th>Input</th>
<th>Local 1D</th>
<th>Local 2D</th>
<th>Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Γ=0.8</td>
<td>Γ=0.9</td>
<td>Γ=1.0</td>
</tr>
<tr>
<td></td>
<td>Γ=0.8</td>
<td>Γ=0.9</td>
<td>Γ=1.0</td>
</tr>
<tr>
<td></td>
<td>Γ=0.8</td>
<td>Γ=0.9</td>
<td>Γ=1.0</td>
</tr>
</tbody>
</table>
# CelebA Super Resolution

<table>
<thead>
<tr>
<th>Method</th>
<th>% Fooled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Gamma = \text{n/a}$</td>
</tr>
<tr>
<td>ResNet</td>
<td>4.0</td>
</tr>
<tr>
<td>srez GAN (Garcia, 2016)</td>
<td>8.5</td>
</tr>
<tr>
<td>Pixel Recursive (Dahl et al., 2017)</td>
<td>-</td>
</tr>
<tr>
<td>Image Transformer, 1D local</td>
<td>$35.94 \pm 3.0$</td>
</tr>
<tr>
<td>Image Transformer, 2D local</td>
<td>$36.11 \pm 2.5$</td>
</tr>
</tbody>
</table>

Human Eval performance for the Image Transformer on CelebA. The fraction of humans fooled is significantly better than the previous state of art.
Cifar10 SuperResolution
Conditional Image Completion
Music generation using self-attention
Raw representations in music and language

(Image from Simon & Oore, 2016)
Music Language model:
Prior work Performance RNN (Simon & Oore, 2016)
Continuations to given initial motif

Given motif

RNN-LSTM

Transformer

Music Transformer
Continuations to given initial motif

Given motif
Continuations to given initial motif
Continuations to given initial motif

Given motif

RNN-LSTM
Continuations to given initial motif

Given motif

RNN-LSTM
Continuations to given initial motif

Given motif

RNN-LSTM

Transformer
Continuations to given initial motif

Given motif

RNN-LSTM

Transformer
Continuations to given initial motif

Given motif

RNN-LSTM

Transformer

Music Transformer
Continuations to given initial motif

Given motif

RNN-LSTM

Transformer

Music Transformer
Self-Similarity in Music
Sample from Music Transformer
Attention: a weighted average
Attention: a weighted average
Convolution:
Different linear transformations by relative position.
Relative attention (Shaw et al, 2018)
Multihead attention + convolution?
Closer look at attention

\[ \text{softmax}(QK^\top) \]
Closer look at relative attention

$$\text{softmax}(Q K^\top + Q f(E_{rel}))$$

Modulated by relative positions
Machine Translation

<table>
<thead>
<tr>
<th>Model</th>
<th>Position Representation</th>
<th>BLEU En-De</th>
<th>BLEU En-Fr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer Big</td>
<td>Absolute</td>
<td>27.9</td>
<td>41.3</td>
</tr>
<tr>
<td>Transformer Big</td>
<td>Relative</td>
<td>29.2</td>
<td>41.5</td>
</tr>
</tbody>
</table>
Previous work \( O(L^2D) \): 8.5 GB per layer (Shaw et al, 2018)

Per layer, \( L=2048, D=512 \)

Relative embeddings \( E_{rel} \)

Relative distances

\[
softmax(QK^\top + Q f(E_{rel}))
\]

Multiply by Q
Our formulation $O(LD)$: **4.2 MB** per layer

$$\text{softmax}(QK^\top + \text{skew}(QE_{rel}^\top))$$

Per layer, $L=2048$, $D=512$

Absolute by relative

$QE^\top$

Absolute by absolute

$\text{Skew}$

Pad $\rightarrow$ Reshape $\rightarrow$ Slice $\rightarrow i_q$
Goal of skewing procedure

Indexed by

absolute by relative

absolute by absolute
Skewing to reduce relative memory from $O(L^2D)$ to $O(LD)$

Per layer, $L=2048$, $D=512$

Previous work
$O(L^2D)$: 8.5 GB

Our work
$O(LD)$: 4.2 MB
A Jazz sample from Music Transformer
A Jazz sample from Music Transformer
Convolutions and Translational Equivariance
Relative positions Translational Equivariance
Relative Attention And Graphs
Relative Attention And Graphs

Relational inductive biases, deep learning, and graph networks. (Battaglia et al., 2018)

Self-Attention With Relative Position Representations (Shaw et al., 2018)
Message Passing Neural Networks

\[ m^{t+1}_v = \sum_{w \in N(v)} M_t(h^t_v, h^t_w, e_{vw}) \]

\[ h^{t+1}_v = U_t(h^t_v, m^{t+1}_v) \]

\[ \hat{y} = R(\{h^T_v | v \in G\}) \]

Neural Message Passing For Quantum Chemistry. Gilmer et al. ICML 2017

Slide credit: Justin Gilmer
Multiple Towers

- Run k smaller copies of the MPNN in parallel.
- Mix node states after each message pass.
- Offers a factor of k speedup for the same node dimension d (>2x speedup when d=200).
- Also helped improve performance when used with matrix multiply message function.

Slide credit: Justin Gilmer
Graph Library

Code

With Justin Gilmer, Jonathan Frankle, and David Bieber
Self-Attention

Constant ‘path length’ between any two positions.

Unbounded memory.

Trivial to parallelize (per layer).

Models Self-Similarity.

Relative attention provides expressive timing, equivariance, and extends naturally to graphs.
Next Steps

Generate larger images and eventually video

Semi-supervised/transfer learning for images

Multiscale modeling

Patch models/dictionary learning
Less autoregressive generation
Generation Can be Slow

Multimodality prohibits naive parallel generation.

Consider the following plausible translations for Thank You:

Danke Schön
Danke
Vielen Dank
Sequential generation breaks modes.

\[ P(y_1, \ldots, y_n \mid x) = P(y_1 \mid x) P(y_2 \mid x, y_1) \ldots P(y_n \mid x, y_1 \ldots, y_{n-1}) \]
Decision Making in The Latent Space

Multimodality prohibits naive parallel generation.

Consider the following plausible translations for Thank You:

- Danke Schön
- Danke
- Vielen Dank

Autoregressive factorization: \( P(y_1, \ldots, y_n \mid x) = P(y_1 \mid x) P(y_2 \mid x, y_1) \ldots P(y_n \mid x, y_1 \ldots, y_{n-1}) \)

Break mode in latent: \( P(y_1, \ldots, y_n, z \mid x) = P(z \mid x) P(y_1 \mid x, z) \ldots P(y_n \mid x, z) \)
Active Research Area

Non autoregressive transformer (Gu and Bradbury et al., 2018)

Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement (Lee, Manismov, and Cho, 2018)


Towards a Better Understanding of Vector Quantized Autoencoders Roy, Vaswani, Parmar, Neelakantan, 2018

Blockwise Parallel Decoding For Deep Autogressive Models (NeurIPS 2019) Stern, Shazeer, Uszkoreit,
Non Autoregressive transformer:

Transformer allows parallel training

Generation is still sequential

Non autoregressive transformer (Gu and Bradbury et al., 2018)

Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement (Lee, Manismov, and Cho, 2018)
Figure 3. **Greedy decoding** – example of how a trained NMT model produces a translation for a source sentence "Je suis étudiant" using greedy search.
Vector Quantized Autoencoders

Combines a learned code-book with nearest-neighbor search.

Encoder is trained using straight-through estimator, i.e. do not perform nearest neighbor in backward pass.

Code-book trained using Exponential Moving Averages (EMA) of encoder states.

Applied to speech and video generation, where authors presented some evidence of (unsupervised) learning of phonemes.
Main Contributions

Propose the Latent Transformer (ICML' 18) using Semantic Hashing and VQ-VAE as discrete autoencoders.

Study the VQ-VAE autoencoder in a principled manner using the lens of Expectation Maximization (EM) algorithm.

EM together with Distillation from an autoregressive teacher helps us almost close the gap with autoregressive models.
Latent Transformer

Encoder $e(x)$

Latent Predictor $p(z)$

Nearest Neighbors

Decoder

$z_e(y)$

$z_q(y)$

Codebook

<table>
<thead>
<tr>
<th>1</th>
<th>0.1 2.3 ... 0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2.5 -1.2 ... -6.7</td>
</tr>
<tr>
<td>3</td>
<td>-0.8 0.9 ... 7.2</td>
</tr>
<tr>
<td>...</td>
<td>-3.9 2.1 ... 0.7</td>
</tr>
<tr>
<td>K</td>
<td>4.8 -0.3 ... -5.2</td>
</tr>
</tbody>
</table>

$x \rightarrow$ Encoder $e(x) \rightarrow$ Latent Predictor $p(z) \rightarrow$ Nearest Neighbors $z_e(y) \rightarrow$ Decoder $y'$
## Results on English-German Translation

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Latency</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoregressive baseline</td>
<td>27.0</td>
<td>265 ms</td>
<td>1x</td>
</tr>
<tr>
<td>NAT + distillation (Gu et al)</td>
<td>18.7</td>
<td>79 ms</td>
<td>3.3x</td>
</tr>
<tr>
<td>VQ-VAE + EM + distillation</td>
<td>26.7</td>
<td>81 ms</td>
<td>3.3x</td>
</tr>
</tbody>
</table>

- Wordpiece-level autoregressive model achieves 27.0 BLEU.

- Latent-level autoregressive model achieves 26.7 BLEU while being >3x faster to decode.
Results on English-French Translation

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Latency</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autoregressive baseline</td>
<td>27.0</td>
<td>771 ms</td>
<td>1x</td>
</tr>
<tr>
<td>VQ-VAE</td>
<td>29.0</td>
<td>215 ms</td>
<td>3.6x</td>
</tr>
<tr>
<td>VQ-VAE + EM</td>
<td>29.5</td>
<td>215 ms</td>
<td>3.6x</td>
</tr>
<tr>
<td>VQ-VAE + EM + denoising</td>
<td>30.0</td>
<td>215 ms</td>
<td>3.6x</td>
</tr>
</tbody>
</table>

- Latent-level autoregressive model achieves 33.0 BLEU (without distillation) while being >3x faster to decode.
- Additional gain from using denoising autoencoder for learning latents.
- Anticipate distillation to close the gap.
Analysis of Latent Codes

| 7 89 517 3773 760 760 760 760 |
| 607 1901 1901 3051 760 760 760 760 |
| 2388 15 850 2590 760 760 760 760 |
| 670 127 17 3773 760 760 760 760 |
| 2335 26 129 2986 760 760 760 760 |
| 10 45 1755 766 760 760 760 760 |
| 3773 1082 13 91 760 760 760 760 |
| 1790 38 270 554 760 760 760 760 |
| 2951 2015 91 2418 760 760 760 760 |
| 2951 27 760 760 760 760 760 760 |
| 463 201 3410 3051 760 760 760 760 |

Table 5: Example latent codes for sentences from the WMT’14 English-German dataset highlighting the emergence of the EOS/PAD latent (760 in this case).

- Can speed up even further by stopping at EOS latent.
- Additional analysis using PML/tf-idf with n-grams did not reveal much.
Papers

Fast Decoding in Sequence Models Using Discrete Latent Variables (ICML 2018)
Kaiser, Roy, Vaswani, Pamar, Bengio, Uszkoreit, Shazeer

Towards a Better Understanding of Vector Quantized Autoencoders (Submitted to ICLR 2019)
Roy*, Vaswani*, Parmar*, Neelakantan*
Let's represent this sentence,

Representieren wir diesen Satz,
Relative Self-Attention and Graphs
Relative Self-Attention

\[ A(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right)V \]
Relative Self-Attention

Before

Logits:

\[ l_{ij} = \frac{e_i W_Q (e_j W_K)^T}{\sqrt{d}} \]

Attention:

\[ a_{ij} = \frac{e_{ij}}{\sum_k e_{ik}} \]

With Relative

Logits:

\[ l_{ij} = \frac{e_i W_Q (e_j W_K + a_{ij})^T}{\sqrt{d}} \]

Attention:

\[ a_{ij} = \frac{e_{ij}}{\sum_k e_{ik}} \]
Results

Self-Attention With Relative Position Representations
Shaw, Uszkoreit, and Vaswani. NAACL 2018

Music Transformer
Huang, Vaswani, Uszkoreit, Shazeer, Simon, Hawthorne, Dai, Hoffman, Dinculescu, and Eck
Extending self-attention to Graphs

Logits:

\[ l_{ij} = \frac{e_i W_Q(e_j W_K + a_{ij})^T}{\sqrt{d}} \]

Attention:

\[ a_{ij} = \frac{e_{ij}}{\sum_k e^{l_{ik}}} \]
Graph Library

https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/layers/common_message_passing_attention.py

With Justin Gilmer, Jonathan Frankle, and David Bieber

Neural Message Passing For Quantum Chemistry. Gilmer et al. ICML 2017
Transfer learning
Improving Language Understanding by Generative Pre-Training (Radford, Narsimhan, Salimans, and Sutskever)

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin, Chang, Lee, and Toutanova)
Optimization and Large Models


Self-attention in Other Work.


A Time-Restricted Self-Attention Layer for ASR (ICASSP 2018). Povey, Hadian, Gharemani, Li, Khudanpur.
Ongoing and Future Work
Ongoing

Self-supervision and classification for images and video

Understanding Transfer
Future

Multitask learning

Long-range attention
CIFAR reconstructions

Figure 4: Samples of original and reconstructed images from CIFAR-10 using VQ-VAE trained using EM with a code-book of size $2^8$. 
Tensor2Tensor library

https://github.com/tensorflow/tensor2tensor/

Image_transformer.py

image_transformer_2d.py