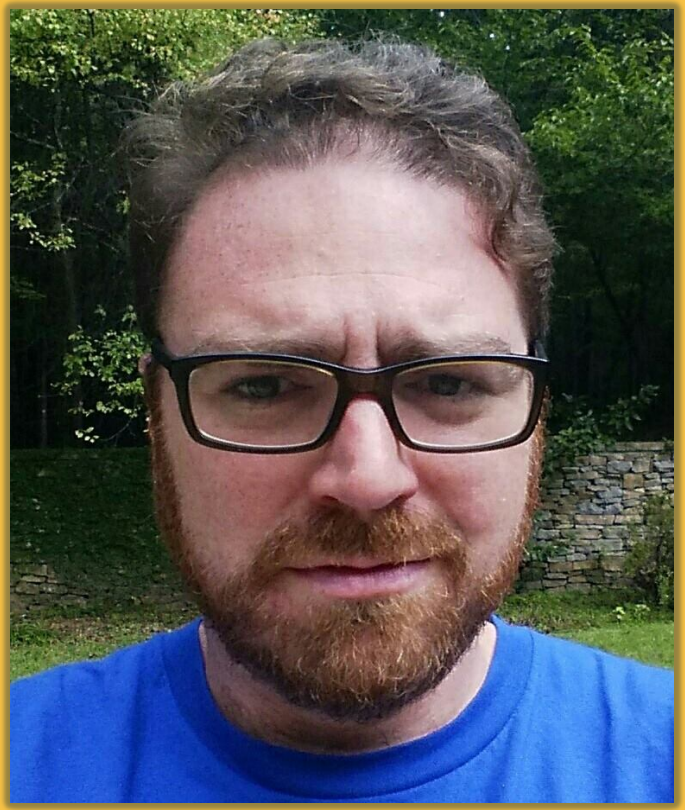
Lecturer Introduction

FACEBOOK AI

Georgia Tech



James Cross

James Cross is an applied researcher at Facebook working on machine translation on the Language and Translation Technologies (LATTE) team since 2017. He received his Ph.D. in computer science from Oregon State University in 2016 for work primarily on syntactic parsing of natural language with neural networks and previously worked as a lawyer.

Neural Machine Translation

FACEBOOK AI

Georgia Tech

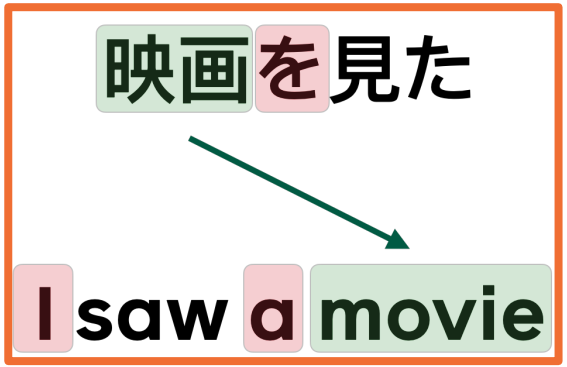
There are **many correct translations** for a typical input sentence

Language is ambiguous

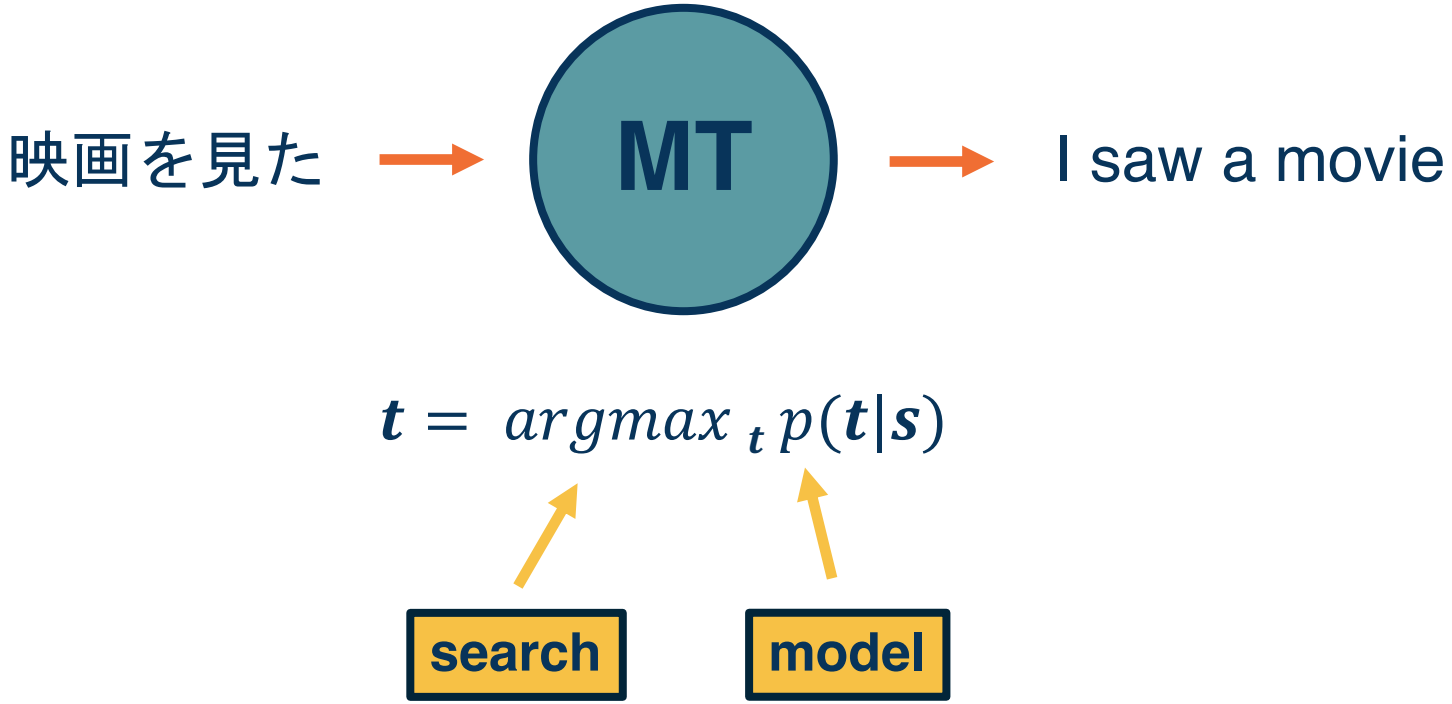
- *The professor said on Monday we would have an exam*

Language depends on context

- I saw her duck 👸
- I saw 👁️ her duck 🦆
- I saw 🪚 her duck 🦆



Languages are **very different** (e.g., structure, and what is implicit versus explicit)


映画を見た
MT
I saw a movie
$t = argmax_t\, p(t|s)$
search
model

Translation is often modeled as a **conditional language model**:

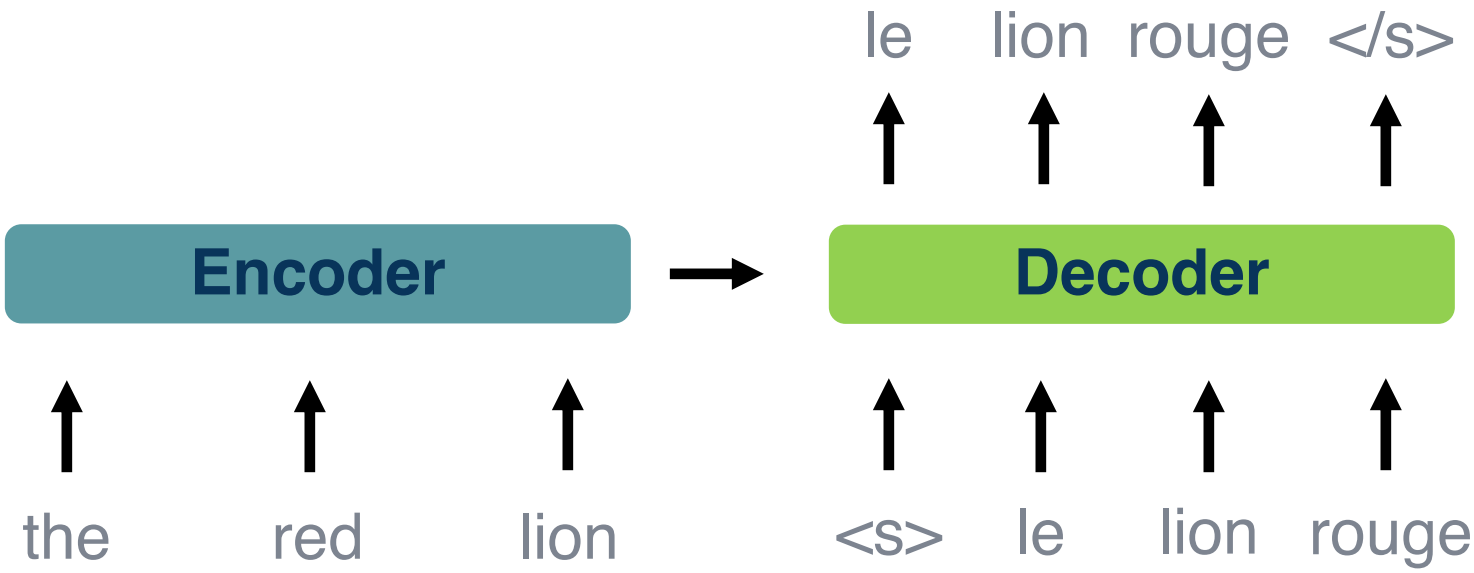
$$p(\boldsymbol{t}|\boldsymbol{s}) = p(t_1|\boldsymbol{s}) * p(t_2|t_1, \boldsymbol{s}) * \ldots * p(tn|t_1, \ldots, tn_{-1}, \boldsymbol{s})$$

Probability of each output token **estimated separately** (left-to-right) based on:

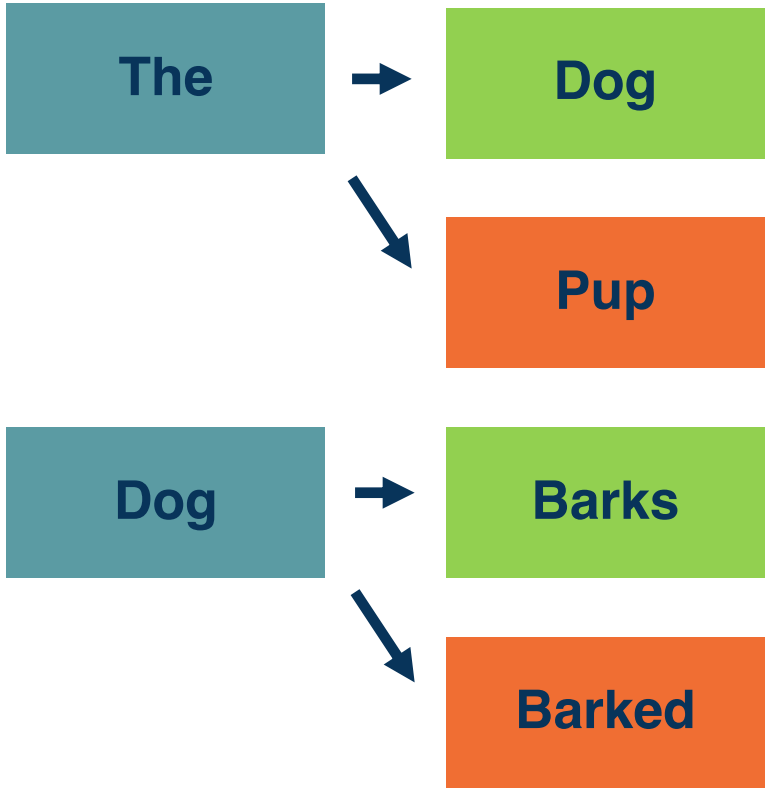
- Entire input sentence (encoder outputs)
- All previously predicted tokens (decoder "state")

$argmax\ p(\boldsymbol{t} \mid \boldsymbol{s})$ is **intractable**

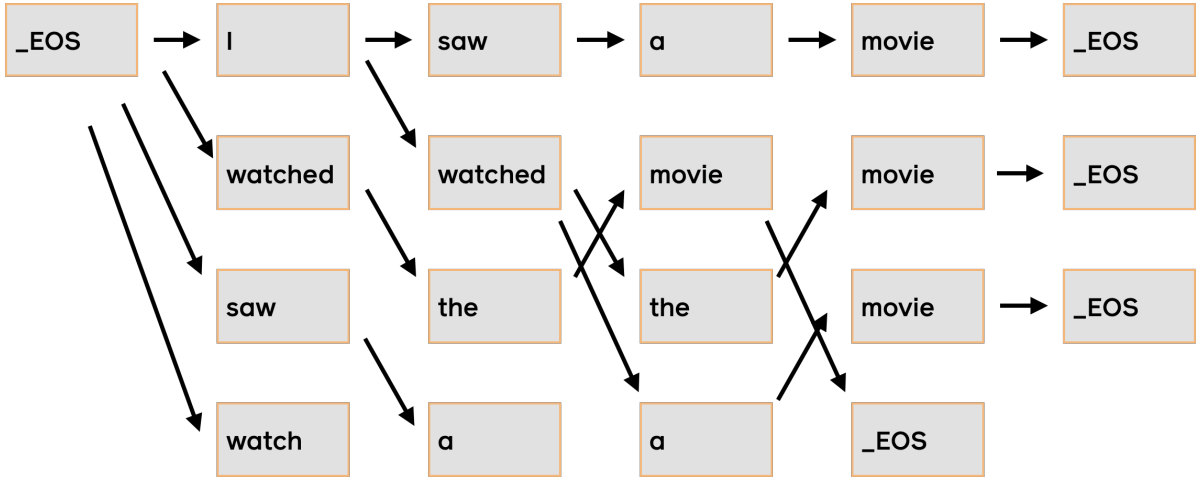
- Exponential search space of possible sequences
- Estimated by beam search (covered shortly)
- Typical beam sizes: 4 to 6


le
lion
rouge
</s>
Encoder
Decoder
the
red
lion
<s>
le
lion
rouge

- Search exponential space in linear time
- Beam size ***k*** determines “width” of search
- At each step, extend each of ***k*** elements by one token
- Top ***k*** overall then become the hypotheses for next step



映画を見た



- Each beam element has different state (for transformer decoder, this is self-attention input for previous steps)
- Total computation scales linearly with beam width
- However computation is highly parallelizable over the beam (efficient on GPU)

RNN (typically LSTM)

- Bi-directional encoder
- Major breakthrough: encoder-decoder attention (Bahdanau et al. 2014)

Convolutional

- Encoder and decoder based on fixed-width convolutions
- Simple activation function (gated linear units)

Transformers

- Architecture used in most recent work
- Originally proposed for machine translation
- (Though now especially well known due to BERT and friends)

Inference Efficiency

FACEBOOK AI Georgia Tech

Models are typically trained once (order 100k updates) and used billions of times

What's expensive?

- Step-by-step computation (Auto-regressive inference)
- Output projection: Θ(|vocabl * output length * beam size)
- Deeper models

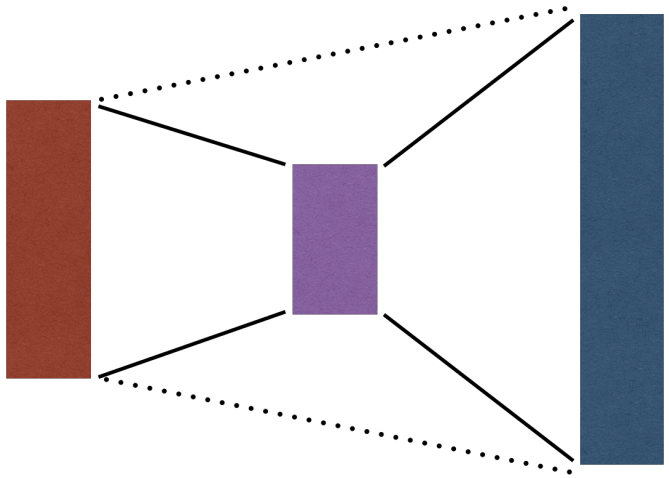
Strategies:

- Smaller vocabularies
- More efficient computation
- Reduce depth / increase parallelism

- Large vocabulary makes output projection expensive
- However not all tokens likely for every input sequence
- IBM alignment models use statistical techniques to model the probability of one word being translated into another
- Lexical probabilities can be used to predict most likely output tokens for a given input
- Achieved up to 60% speedup

$$y = Wx + b$$

$$W \in \mathbf{R}^{|V| x\ h}$$



- Languages have very large (or open-ended) vocabularies
- **Problem:** how to model rare or unseen words
- Character models are general but can be too slow in practice
- Subword models solve this by breaking up words based on frequency
- One widely used algorithm is byte-pair encoding (BPE), illustrated at right

token							frequency
l	o	w	e	r	</w>		2
l	o	w	</w>				5
n	e	w	e	s	t	</w>	6
w	i	d	e	s	t	</w>	3

learned merge operations

r • → r•
l o → lo
lo w → low
e r• → er•

example from Sennrich et al. 2016

- Byte-pair encoding comes from **compression**, where the **most frequent adjacent pair is iteratively replaced**

- **Example:** consider this string

```
abcdeababce
```

- **Step 1:** replace most frequent pair “ab” by “X” (and add replacement rule)

```
XcdeXXce
X=ab
```

- **Step 2:** replace next most frequent pair (here including replacement byte)

```
YdeXYe
X=ab
Y=Xc
```

Example adapted from https://en.wikipedia.org/wiki/Byte_pair_encoding

Layer Drop

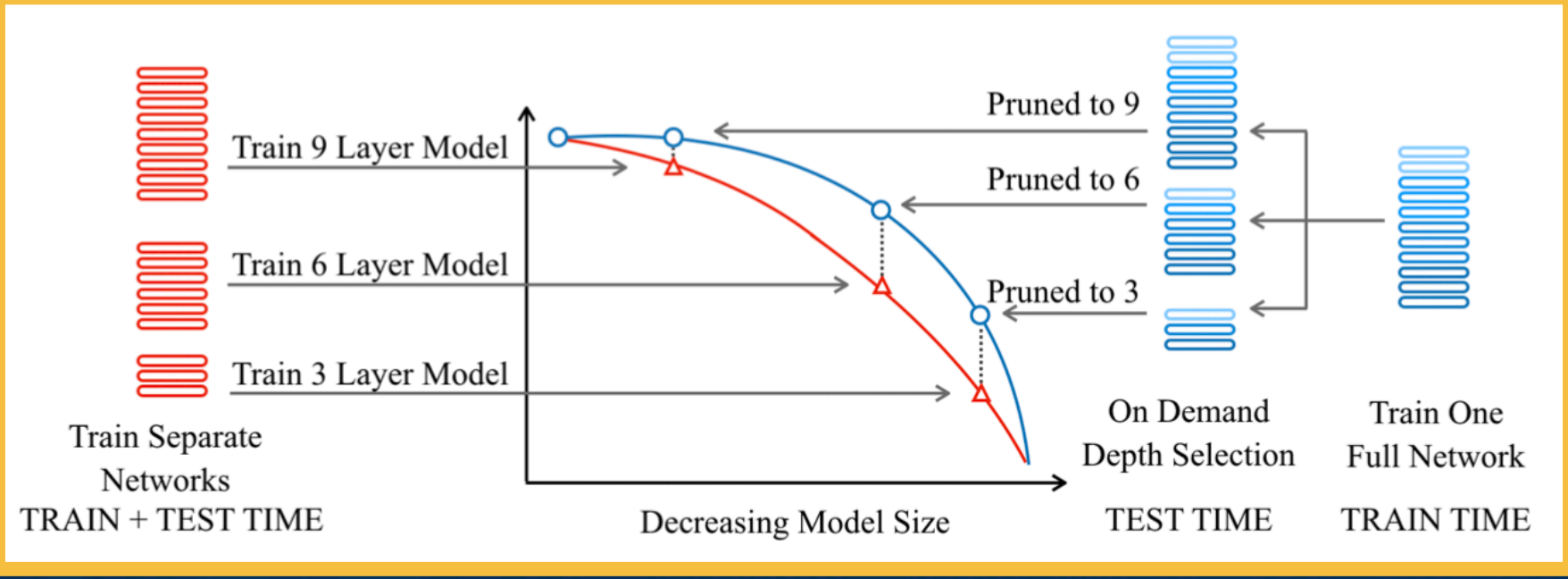


Figure from Fan et al. 2019

Inference is **faster on specialized hardware** (GPUs, TPUs, etc.)

Batched vs. on-demand: average efficiency can be increased by translating multiple inputs at once

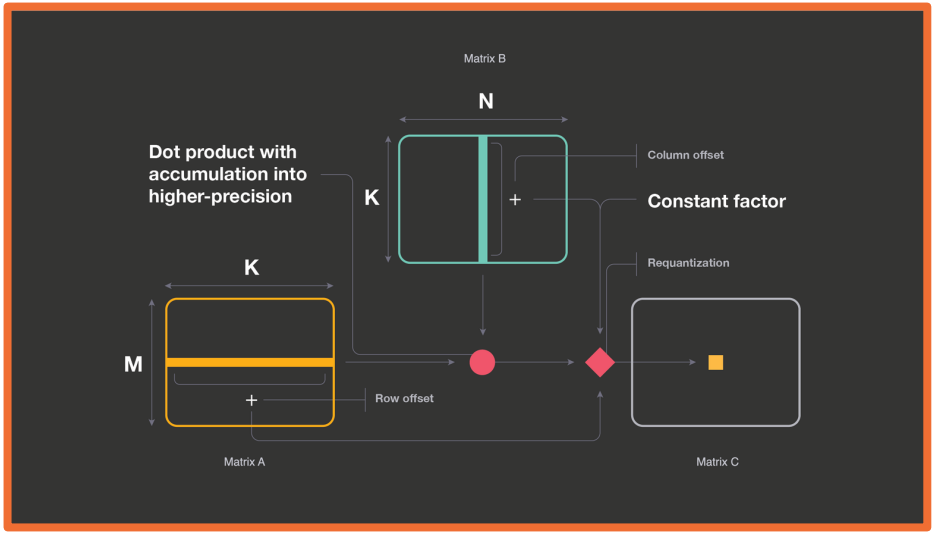
- Especially with accelerators (GPUs, TPUs, etc.)
- However may not be practical for real-time systems

Big speedup from **parallel computation**

- Especially with accelerators
- Autoregressive inference time dominated by decoder

- Computation time is dominated by matrix multiplication
- Inference can be sped up by performing operation in lower precision domain
- Training quantization also possible (e.g, FP16 on current Nvidia GPUs)

$$A[i] = a_{scale}(A_q[i] - a_{zero\_point})$$



https://engineering.fb.com/ml-applications/fbgemm/

Non-Autoregressive Machine Translation

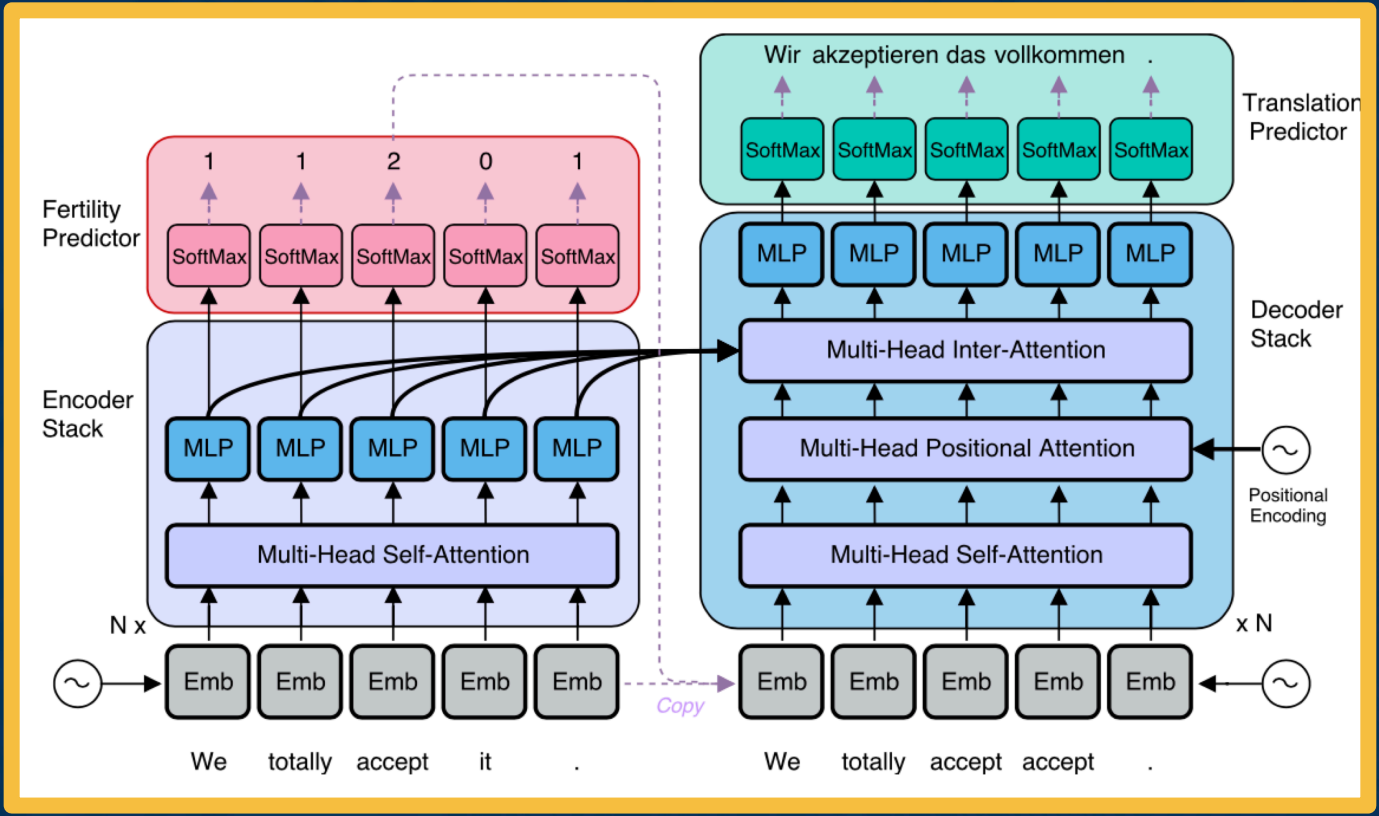


Figure from Gu et al. 2018

FACEBOOK AI Georgia Tech

Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate (2014)

Fan et al. Reducing Transformer Depth on Demand with Structured Dropout (2019)

Gehring et al. A Convolutional Encoder Model for Neural Machine Translation (2017)

Ghazvininejad et al. Mask-Predict: Parallel Decoding of Conditional Masked Language Models (2019)

Gu et al. Non-Autoregressive Neural Machine Translation (2018)

Gu et al. Levenshtein Transformer (2019)

Kasai et al. Deep Encoder, Shallow Decoder: Reevaluating the Speed-Quality Tradeoff in Machine Translation (2020)

Ott et al. Scaling Neural Machine Translation (2018)

Sennrich et al. Neural Machine Translation of Rare Words with Subword Units (2016)

Sutskever et al. Sequence to Sequence Learning with Neural Networks (2014)

Vaswani et al. Attention Is All You Need (2017)