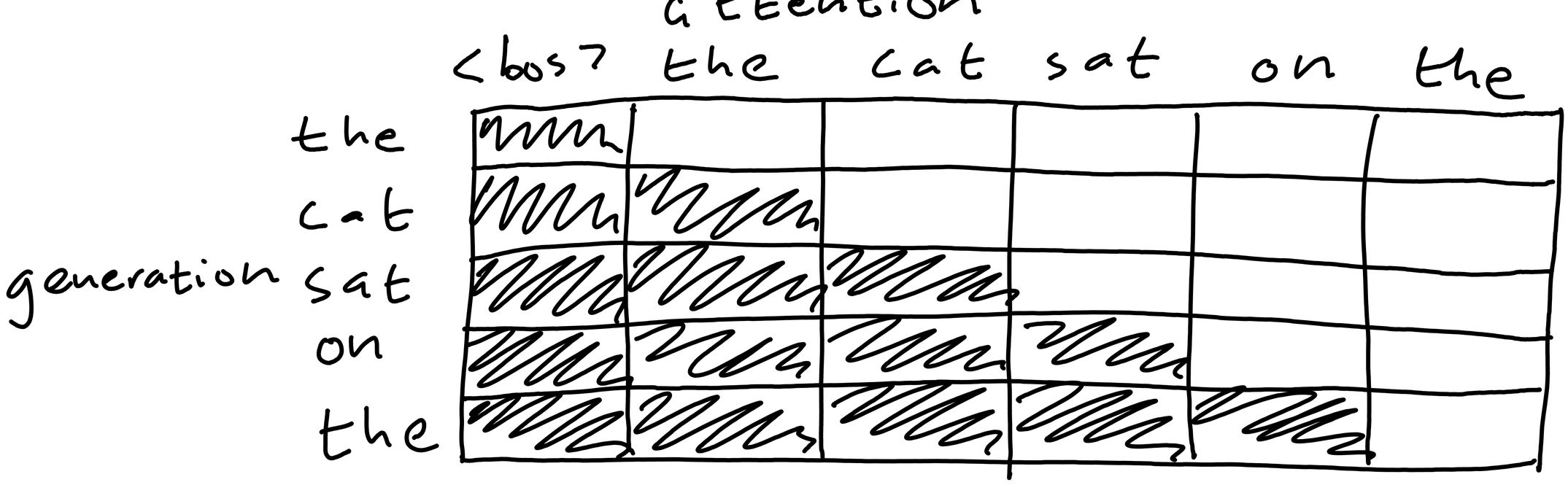
Decoder-only Transformers

How can we use a Transformer for seq. generation? (language modeling)

For each generated word, we can only attend to past words.

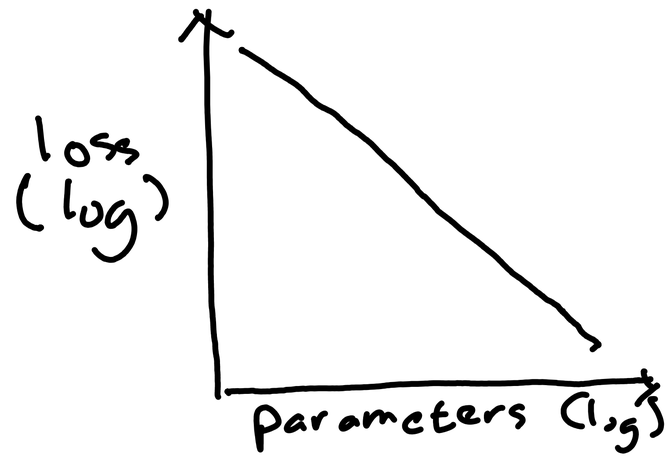


Can use the Transformer stack for generation!

How to train a Transformer for generation?
Feed in full ground-truth sequence but
zero-out invalid (current or future)
positions in self-attention.
Called "causal masking."
Note: This is completely parallelizable
across time! (unlike RNN training)

Note: LM training in done on "unlabeled" text.
There's tons of text data. (from the internet)
So, we can scale up models!
Scaling up RNNs was hard because of
vanishing/exploding gradients.
Transformers tend to be more stable.
why? 1) "path length" is always 1.
2) Residual connection and layernorm.

Finding: Scaling leads to predictable improvements!



Lower loss on held-out data → better able to predict the next token.

1. When was Wayne Gretzky born? ...
2. Do these sentences mean the same thing?
3. [long article] tl;dr ...
4. "I like cheese" in French is ...

We can formulate any text-based task as a "text-to-text" task.

Lower loss → better at possibly everything.

Recall these models are often trained on web text. Often there are many continuations:

1. I don't know.
2. I don't care.
3. means "too long; didn't read"
4. a common saying.

Importance of prompting.

1. Try different prompts:
   [sentence] in French is ...
   The French translation of [sentence] is ...
2. For classification tasks, we can inspect the probabilities of the strings corresponding to the classes.

3. "few-shot in-context learning"

Past examples were "zero-shot" prompting
(no traing data)

what if we have a small task-specific training dataset? Provide it as context.

ex/ Who played Batman in the Dark Knight?
2-shot { Christian Bale. Who is the lead singer of the Foo Fighters? David Grohl.
When was Wayne Gretzky born? ...

Pros: 1. Often improves performance.
2. We don't update parameters.
3. We can use the same model for all tasks.

Cons: 1. Cost of prediction grows.
2. Only works with large-ish models
3. Often works worse than fine-tuning.

Encoder-Decoder Transformer

Encoder layer stak processes some input with full visibility; decoder generates the out put conditioned on encoder outputs and past generations.
Encoder is unchanged. How do we feed encoder outputs to the decoder?

Encoder-decoder attention (Cross-attention)

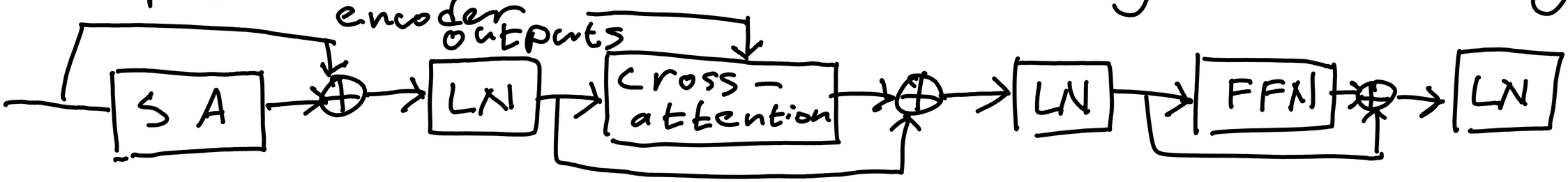
Queries: Decoder states

Keys: encoder outputs

Values: encoder outputs

Comparison f'n: Scaled dot-product

Reduction: Softmax-weighted average



How to train an encoder-decoder Transformer...
(can always train on supervised seq2seq)
on unlabeled data?

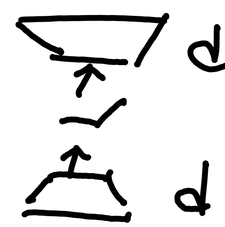
Similar to MLM: Given some masked-out text, predict missing parts.

original text: Thank you for inviting me to your party.
corrupted input: Thank you <mask> me to your <mask>.
target: for inviting <sep> party

Conventions for Transformer architecture

dimensions / hyperparameters:

1. L - # of layer blocks
2. d_{model} - dimensionality along residual path (including embeddings)
3. d_{ff} - dimension "inside" FFN
4. d_{kv} - dimensionality of queries, keys, values
5. H - # of attention heads



Conventions:

$d_{ff} = 4\, d_{model}$

$H d_{kv} = d_{model}$

L and d_{model} grow together