New frontiers - NLP research in 2022

Deep Natural Language Processing, 2022

Sebastian Jaszczur, Spyros Mouselinos

Final group projects

- **No** final presentation!
- Due date - 28th of June

- Check my comments from the project proposals

- Mateusz’s slides are coming today, no recordings :(

Remember about right format

Instructions for *ACL Proceedings

Anonymous ACL submission

Abstract

This document is a supplement to the general instructions for *ACL authors. It contains instructions for using the LaTeX style files for ACL conferences. The document itself conforms to its own specifications, and is therefore an example of what your manuscript should look like. These instructions should be used both for papers submitted for review and for final versions of accepted papers.

1 Introduction

These instructions are for authors submitting papers to *ACL conferences using LaTeX. They are not self-contained. All authors must follow the general instructions for *ACL proceedings [1] and this

For the final version, omit the `review` option:

```
\usepackage{acl}
```

To use Times Roman, put the following in the preamble:

```
\usepackage{times}
```

(Alternatives like txfonts or newtx are also acceptable.)

Please see the LaTeX source of this document for comments on other packages that may be useful.

Set the title and author using `\title` and `\author`. Within the author list, format multiple authors using `\and` and `\And` and `\AND`; please see the LaTeX source for examples.

By default, the box containing the title and au-

We have not scratched the surface…

- dependency parsing
- natural language generation
- summarization
- ethics of LLMs/NLP
- coreference resolution
- named-entity recognition
- sentiment analysis
- argument mining
- speech, multimodality

Efficient Transformer Architectures

Sebastian Jaszczur

NLP lecture, 13th of June 2022

Bottlenecks in Transformer

There are multiple places to improve Transformer efficiency:

- Attention mechanism
  - also: linear projections in before/after attention mechanism
- Feed Forward layer
  - primarily conditional computation
- Memory usage
  - this often bottlenecks training process!

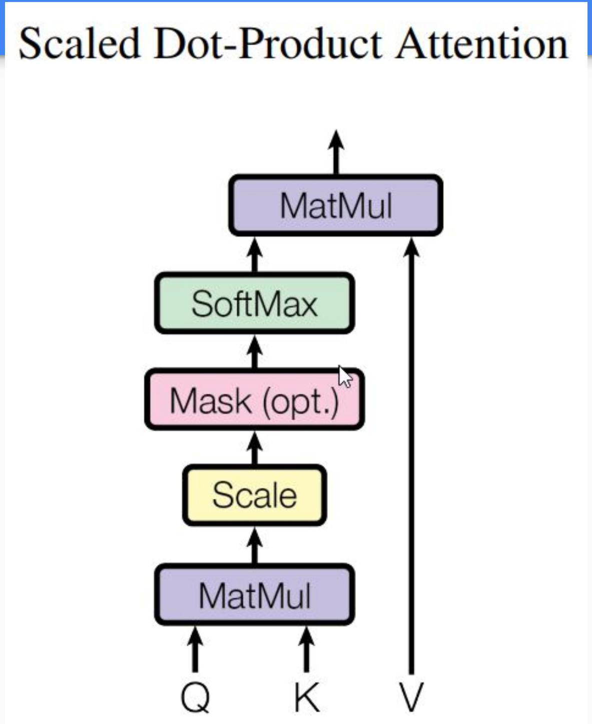
Speeding up Attention Mechanism

Standard Attention Mechanism

Attention mechanism computes "attention matrix", which has the size of Q x K.

This means O(#tokens ^{2}) operations!

Processing very long sequences is infeasible.



Picture: Vaswani et al., 2017

Reformer:
LSH Attention

Reformer: LSH Attention

First idea: we won't separate Queries and Keys vectors.
This makes attention more "symmetrical", but it still works.

Second idea: we will compute a Locality Sensitive Hashing for each query/key.

REFORMER: THE EFFICIENT TRANSFORMER

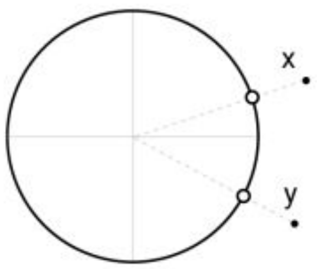
Nikita Kitaev*
U.C. Berkeley & Google Research
kitaev@cs.berkeley.edu

Łukasz Kaiser*
Google Research
{lukaszkaiser,levskaya}@google.com

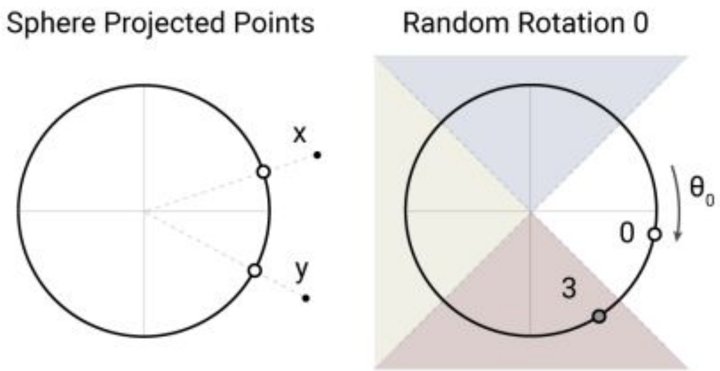
Anselm Levskaya
Google Research

Reformer: LSH Attention

Sphere Projected Points



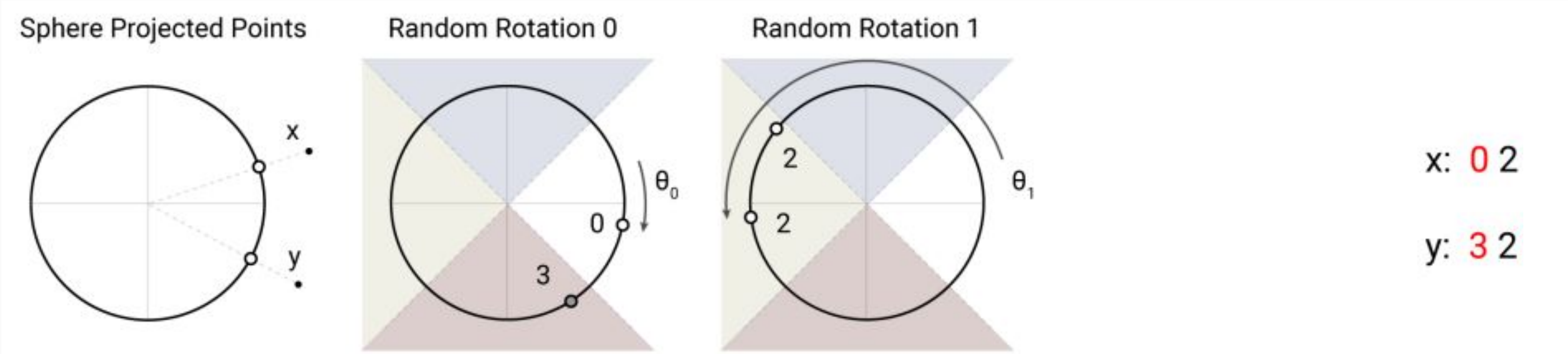
Reformer: LSH Attention



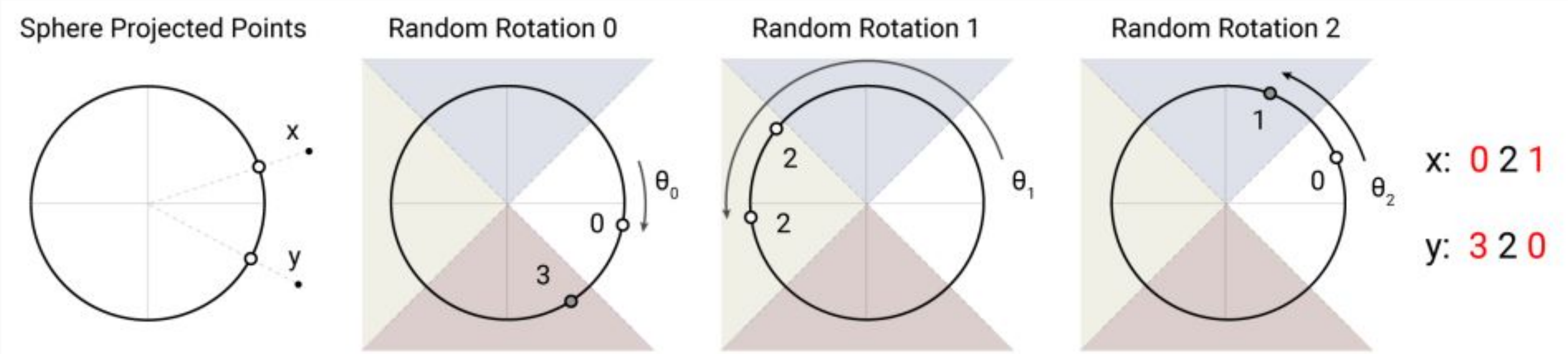
x: 0

y: 3

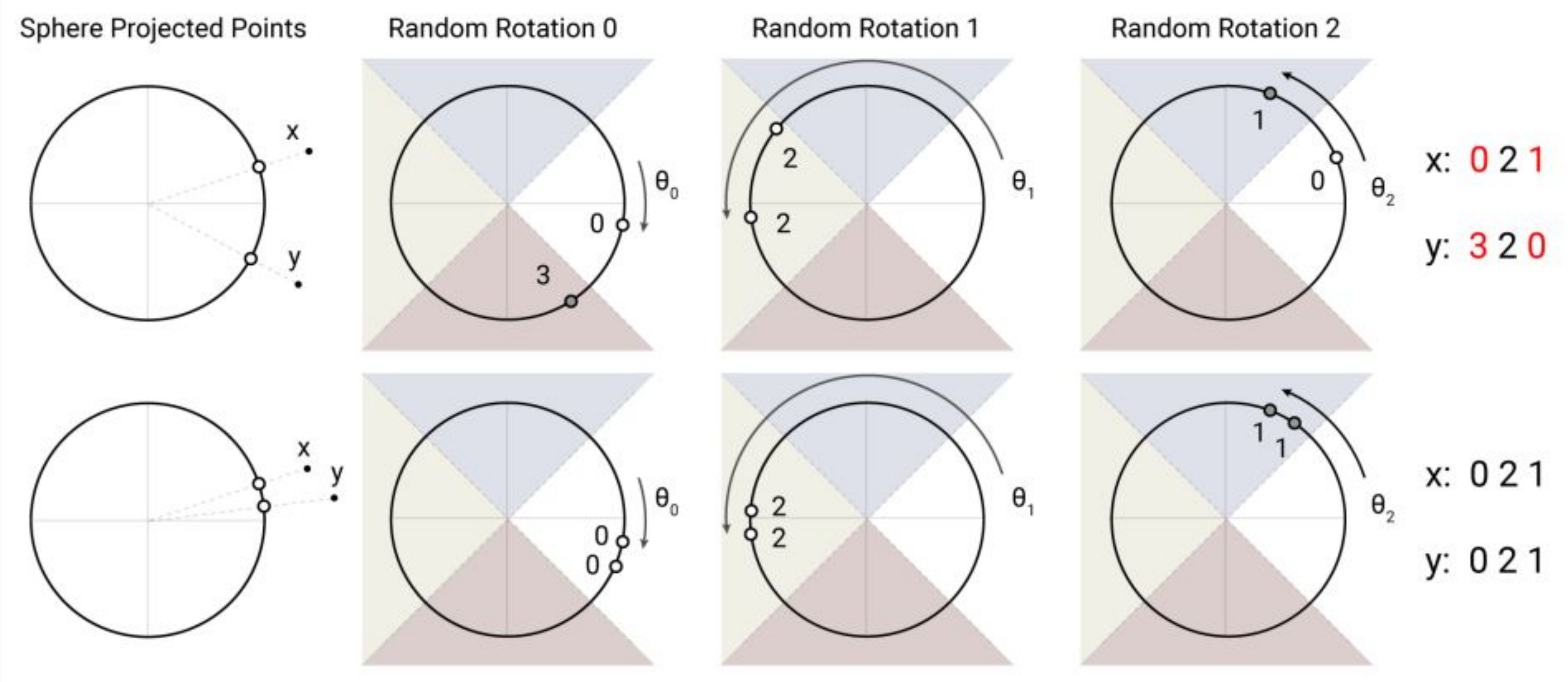
Reformer: LSH Attention



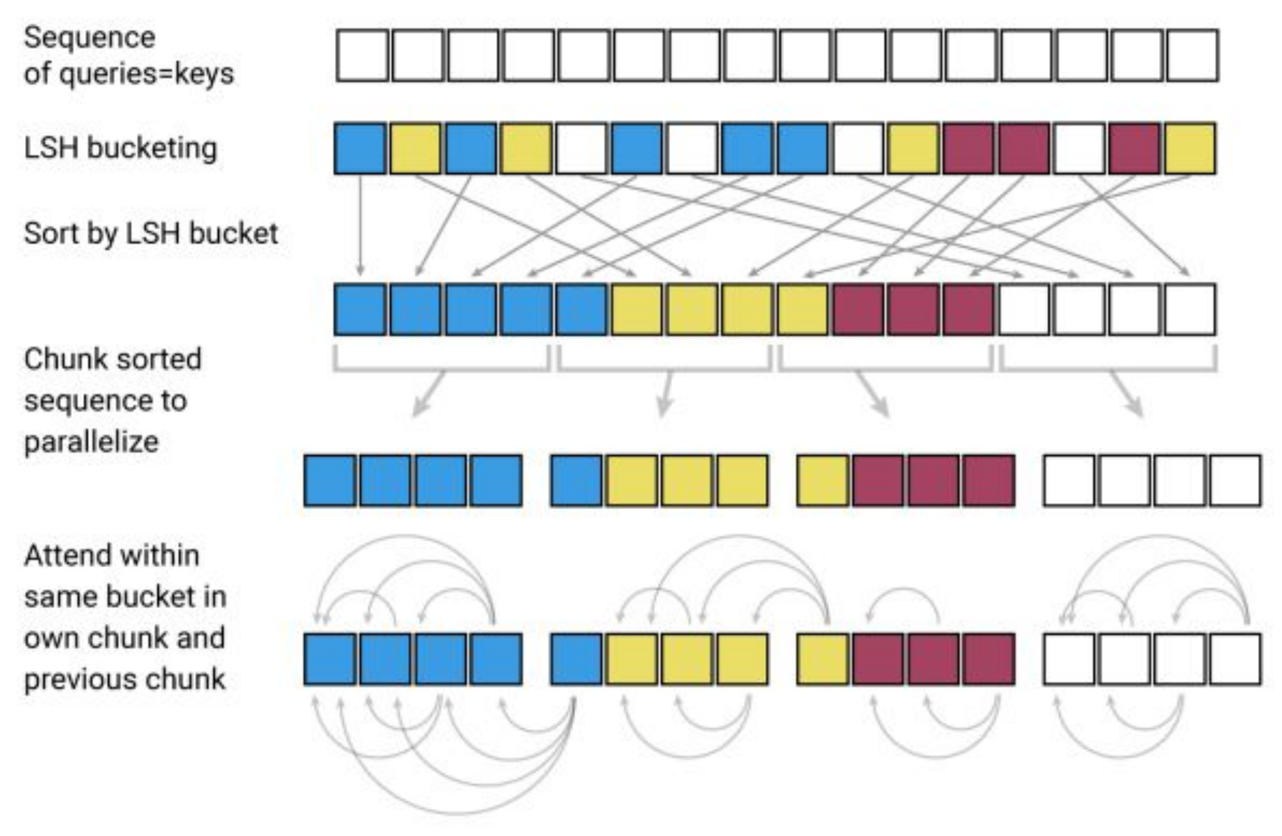
Reformer: LSH Attention



Reformer: LSH Attention



Reformer: LSH Attention continued



Reformer: LSH Attention

The computational complexity depends on number of hash-buckets, but it can reach complexity of $O(N \log N)$, with N being #tokens.

The method also keeps good accuracy of the attention mechanism, especially with multiple rounds of hashing.

Linformer:
FAVOR+

Linformer: FAVOR+

This is more mathematically complicated modification.

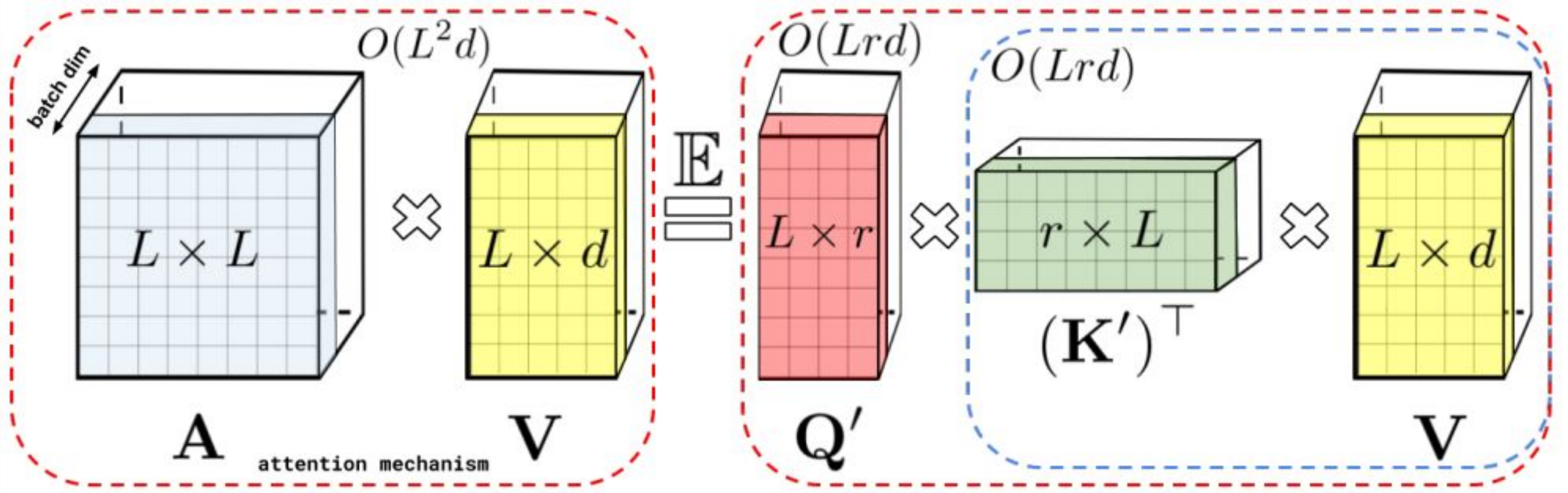
However, it achieves linear complexity wrt. #tokens!

The basic idea is to convert Queries and Keys to random feature maps, and approximate the attention mechanism.

Linformer: Self-Attention with Linear Complexity

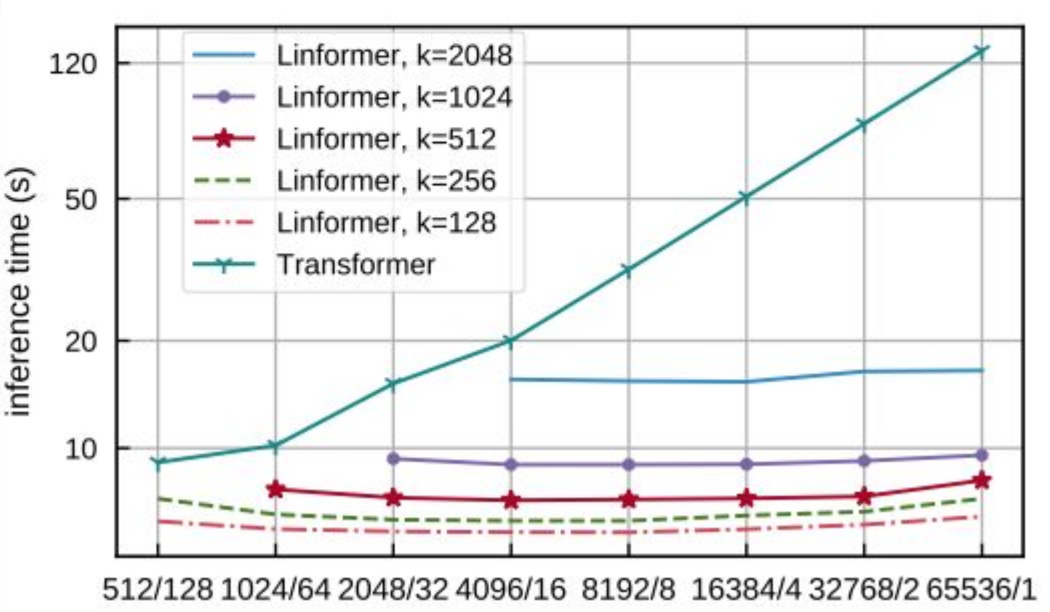
Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, Hao Ma
Facebook AI, Seattle, WA
{sinongwang, belindali, hanfang, mkhabsa, haom}@fb.com

Linformer: FAVOR+. Attention approximation



Linformer: FAVOR+

This approach is both mathematically sound (see proofs in the paper), and brings good results in practice.



Other approaches: there's a few…

Other approaches: there is a few…

Other approaches: there is a few…

In general, this is a reasonably well explored area.

Table source:
Efficient Transformers: A Survey
(Tay et al., 2020)

Model / Paper	Complexity
Memory Compressed (Liu et al., 2018)	$\mathcal{O}(N_c^2)$
Image Transformer (Parmar et al., 2018)	$\mathcal{O}(N.m)$
Set Transformer (Lee et al., 2019)	$\mathcal{O}(kN)$
Transformer-XL (Dai et al., 2019)	$\mathcal{O}(N^2)$
Sparse Transformer (Child et al., 2019)	$\mathcal{O}(N\sqrt{N})$
Reformer (Kitaev et al., 2020)	$\mathcal{O}(N \log N)$
Routing Transformer (Roy et al., 2020)	$\mathcal{O}(N\sqrt{N})$
Axial Transformer (Ho et al., 2019)	$\mathcal{O}(N\sqrt{N})$
Compressive Transformer (Rae et al., 2020)	$\mathcal{O}(N^2)$
Sinkhorn Transformer (Tay et al., 2020b)	$\mathcal{O}(B^2)$
Longformer (Beltagy et al., 2020)	$\mathcal{O}(n(k+m))$
ETC (Ainslie et al., 2020)	$\mathcal{O}(N_g^2 + NN_g)$
Synthesizer (Tay et al., 2020a)	$\mathcal{O}(N^2)$
Performer (Choromanski et al., 2020a)	$\mathcal{O}(N)$
Funnel Transformer (Dai et al., 2020)	$\mathcal{O}(N^2)$
Linformer (Wang et al., 2020c)	$\mathcal{O}(N)$
Linear Transformers (Katharopoulos et al., 2020)	$\mathcal{O}(N)$
Big Bird (Zaheer et al., 2020)	$\mathcal{O}(N)$
Random Feature Attention (Peng et al., 2021)	$\mathcal{O}(N)$
Long Short Transformers (Zhu et al., 2021)	$\mathcal{O}(kN)$
Poolingformer (Zhang et al., 2021)	$\mathcal{O}(N)$
Nyströmformer (Xiong et al., 2021b)	$\mathcal{O}(kN)$
Perceiver (Jaegle et al., 2021)	$\mathcal{O}(kN)$
Clusterformer (Wang et al., 2020b)	$\mathcal{O}(N \log N)$
Luna (Ma et al., 2021)	$\mathcal{O}(kN)$
TokenLearner (Ryoo et al., 2021)	$\mathcal{O}(k^2)$

Speeding up Feed Forward layer: Conditional Computation

Intuitive explanation of conditional computing

Neurons in neural network encode information.

All information (knowledge of the model) is encoded in neurons.

A given neuron, probably, contains information regarding specific task.

In Transformer, every token is processed by all the neurons!

Even if information encoded in those neurons is irrelevant!

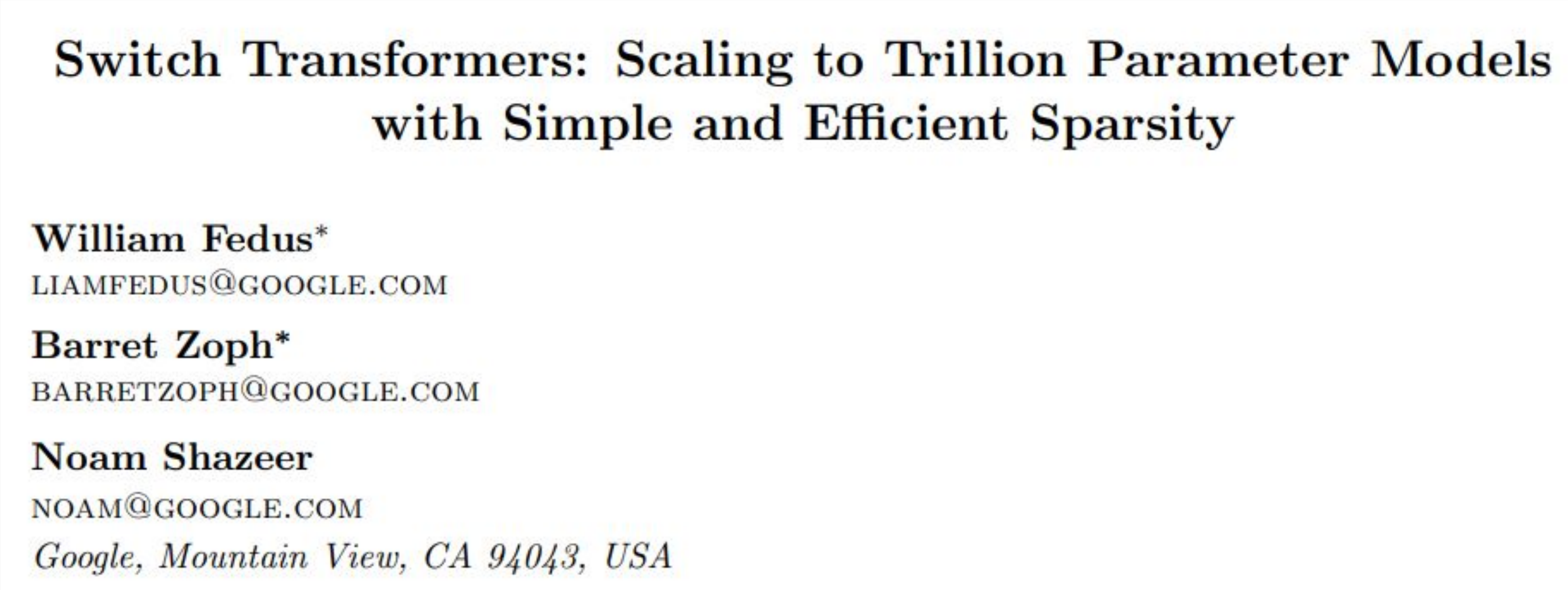
Conditional computation approaches

We can see the problem from two perspectives:

- Increase model size while keeping computation budget.
- Decrease computation budget while keeping model size.

Switch Transformers:
Scaling to Trillion Parameter Models

Switch Transformer


Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

William Fedus*
LIAMFEDUS@GOOGLE.COM

Barret Zoph*
BARRETZOPH@GOOGLE.COM

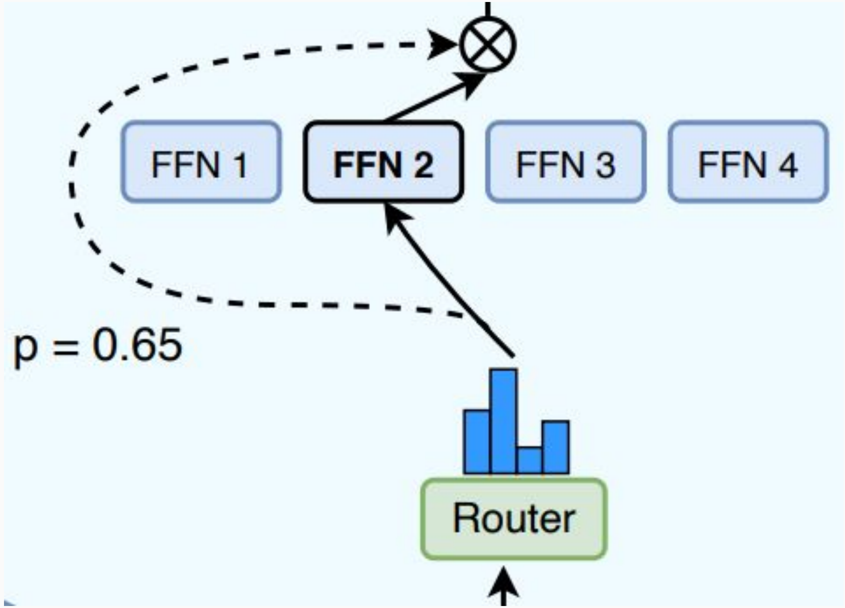
Noam Shazeer
NOAM@GOOGLE.COM

Google, Mountain View, CA 94043, USA

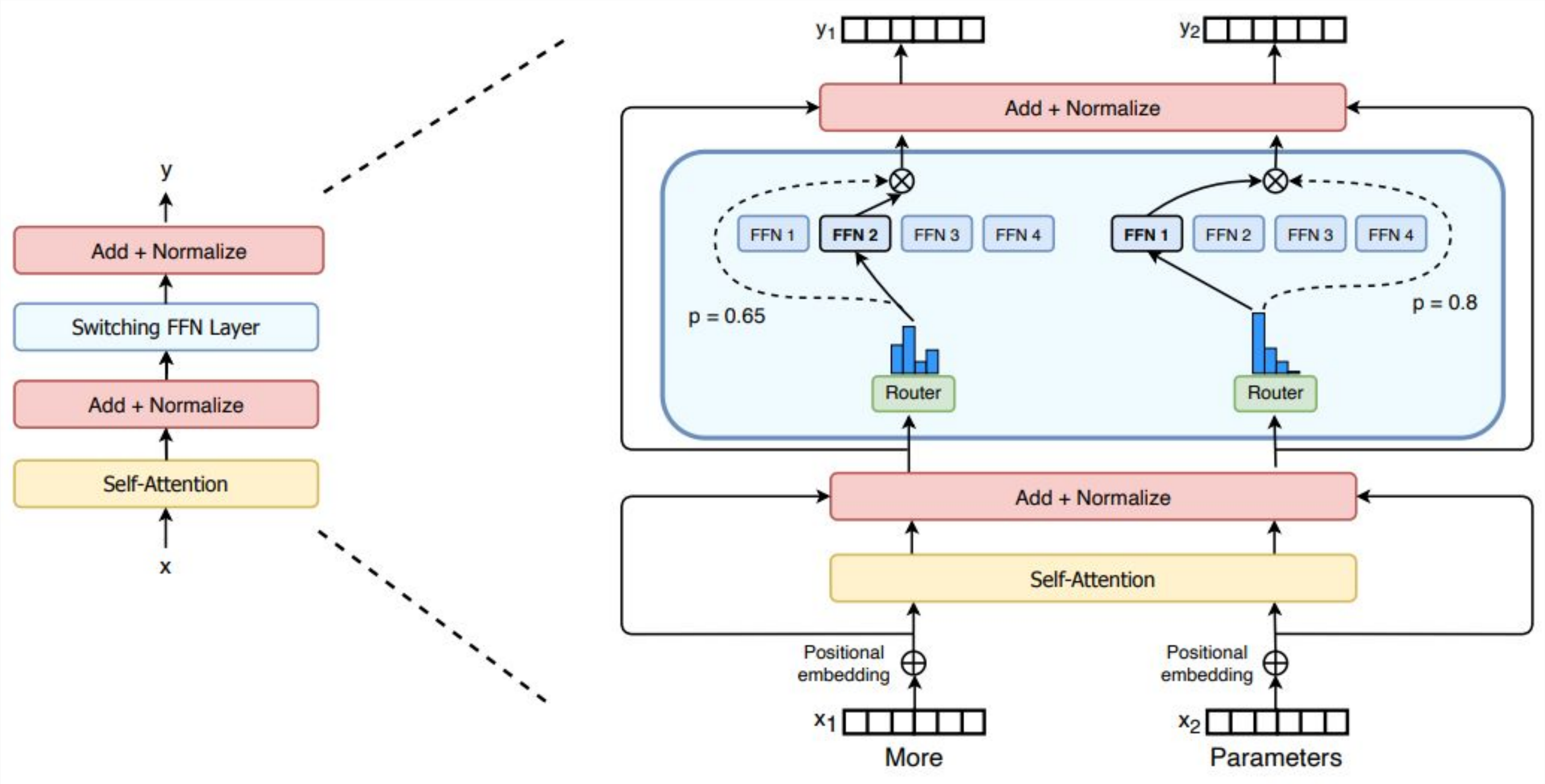
Switch Transformer

We duplicate Feed Forward layer N times, getting N experts.

We insert a router, which decides which version of the layer (which expert) to use.



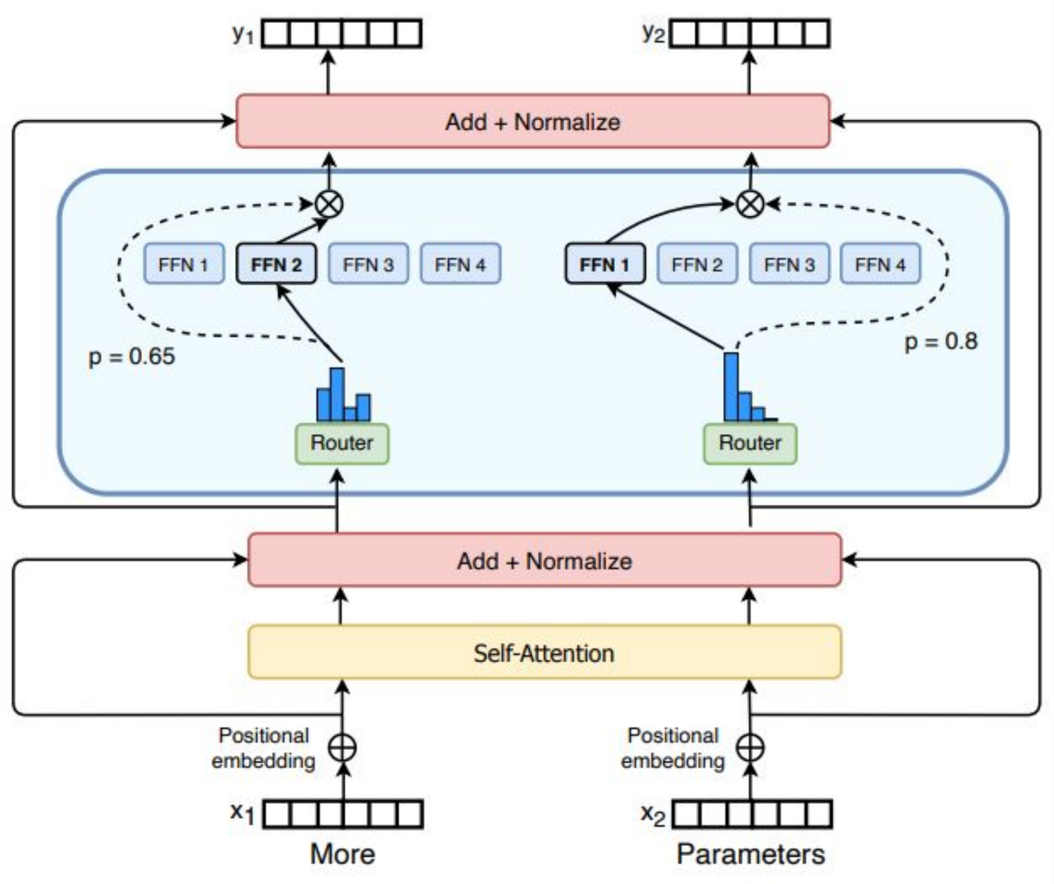
Switch Transformer



Switch Transformer

We can see on the picture that router chooses the expert for each token independently…

almost independently at least.

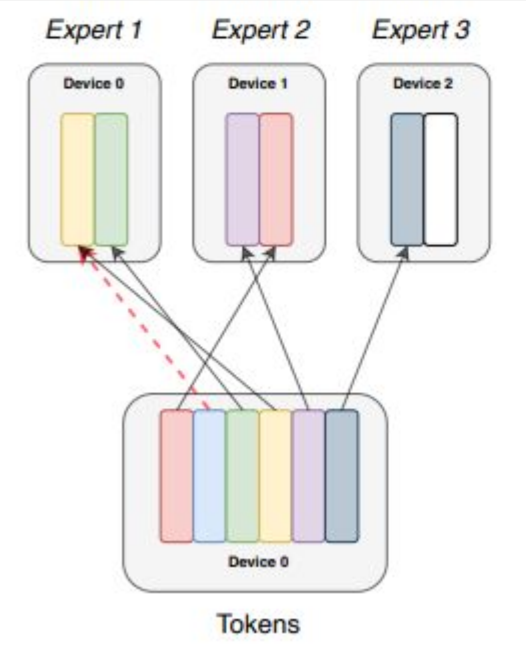


Switch Transformer

For technical reasons, to speed-up training as well, each expert receives some limited amount of tokens.

Often those different experts can be located on different devices!

If an expert receives too many tokens, some are not processed.



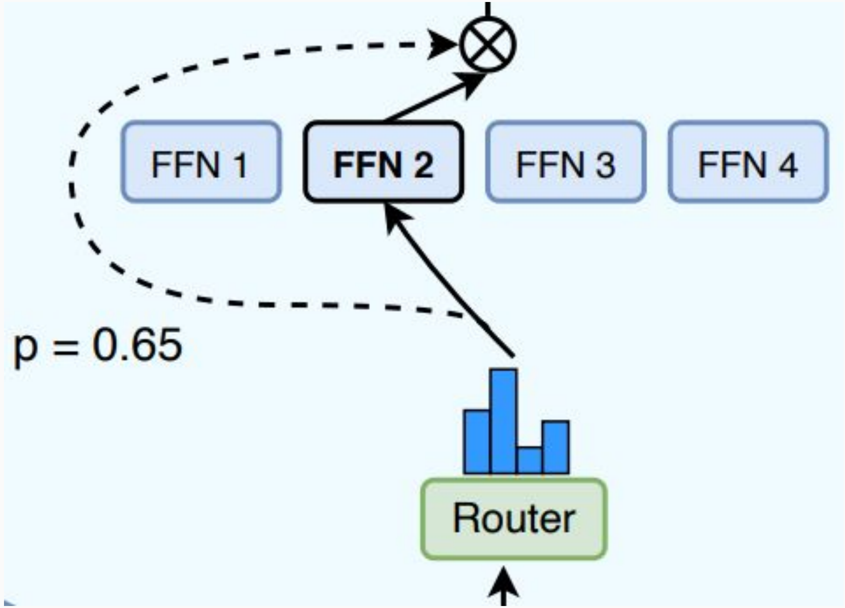
Switch Transformer

How training works?

We just get the probability of choosing a particular expert (0.65 on the picture), and multiply it by expert's output.

This allows for gradients to pass to the router!

(We also need auxiliary exploration loss to balance the load across experts)



Switch Transformer - results

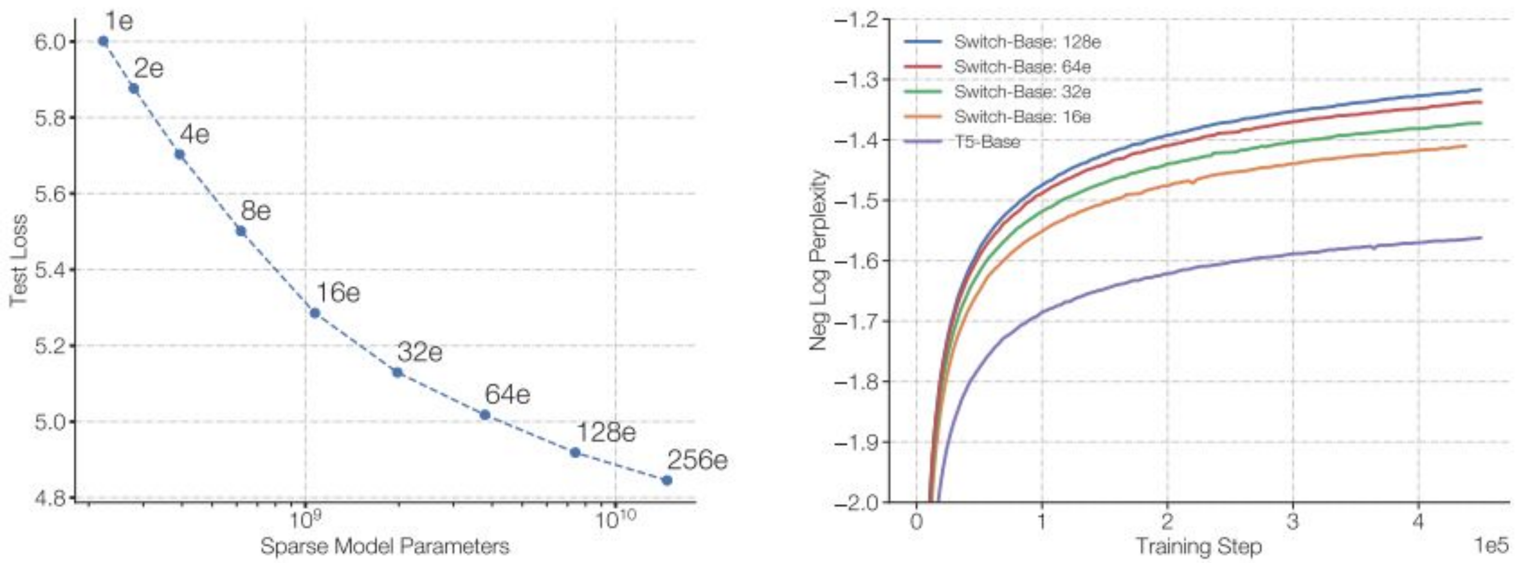


Figure 1: Scaling and sample efficiency of Switch Transformers. Left Plot: Scaling properties for increasingly sparse (more experts) Switch Transformers. Right Plot: Negative log perplexity comparing Switch Transformers to T5 (Raffel et al., 2019) models using the same compute budget.

Scaling Transformers:
Fine-grained Mixture of Experts

Sparse is Enough in Scaling Transformers

The goal was to make conditional computation more fine-grained than MoE.

Instead of scaling up the model, we tried speeding up a model of given size.

Sparse is Enough in Scaling Transformers

Sebastian Jaszczur* University of Warsaw

Aakanksha Chowdhery Google Research

Afroz Mohiuddin Google Research

Łukasz Kaiser* OpenAI

Wojciech Gajewski Google Research

Henryk Michalewski Google Research

Jonni Kanerva Google Research

Sparse is Enough in Scaling Transformers

	Params	Dec. time	Dec. time per block
baseline Transf.	800M	0.160s	5.9ms
+ Sparse FF	-	0.093s	3.1ms
+ Sparse QKV	-	0.152s	6.2ms
+ Sparse FF+QKV	-	0.061s	1.9ms
Speedup		2.62x	3.05x
baseline Transf.	17B	3.690s	0.581s
+Sparse FF	-	1.595s	0.259s
+Sparse QKV	-	3.154s	0.554s
+Sparse FF+QKV	-	0.183s	0.014s
Speedup		20.0x	42.5x

Table 1: Decoding speed (in seconds) of a single token. For Transformer model (equivalent to T5 large with approximately 800M parameters), Scaling Transformers with proposed sparsity mechanisms (FF+QKV) achieve up to 2x speedup in decoding compared to baseline dense model and 20x speedup for 17B param model. [2]

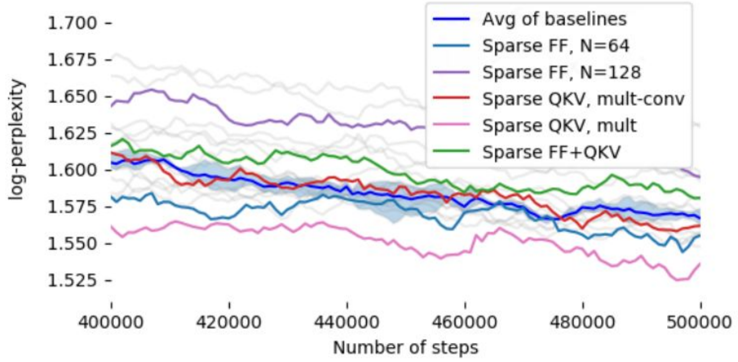


Figure 1: Log-perplexity of Scaling Transformers (equivalent to T5 large with approximately 800M parameters) on C4 dataset with proposed sparsity mechanisms (FF, QKV, FF+QKV) is similar to baseline dense model. Other models used in this paper are shown in grey lines; raw data is available in the appendix.

Sparse is Enough in Scaling Transformers

Basic idea: single expert is no longer the whole layer, it is a single neuron!

Of course, we will activate multiple experts in each layer.

For brevity, we will skip over technical details.

Decreasing RAM usage

Decreasing RAM usage

During training, we normally have to remember activations of all the layers!

This is often a bottleneck when training Transformers.

Could we decrease this memory usage? The answer is yes!

Gradient checkpointing

Gradient checkpointing

(It really should be called *activation* checkpointing.)

Let's say we have N layers in the model. Instead of remembering activations after each layer, we remember activations of every sqrt(N)-th layer.

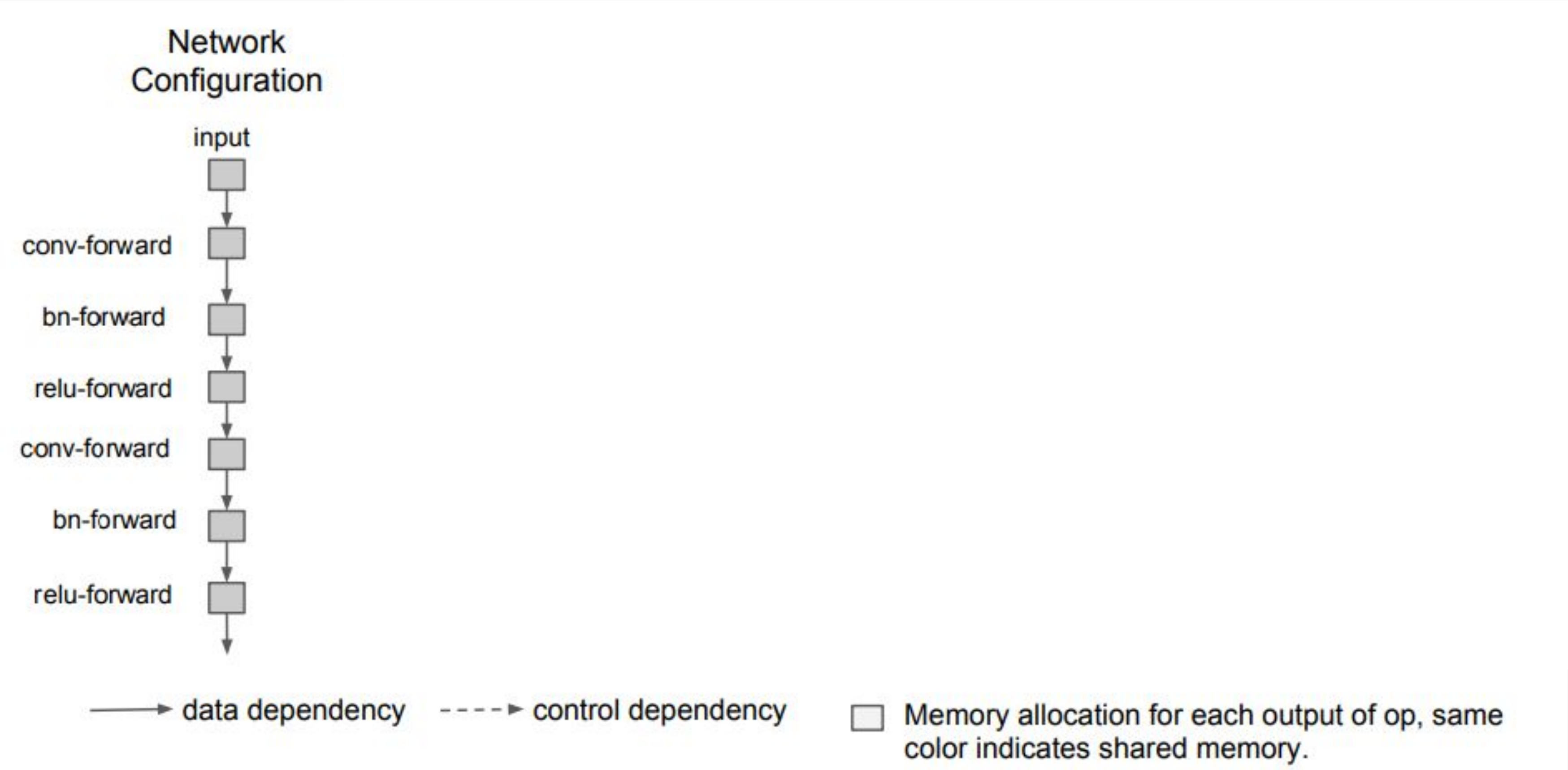
During backward pass, we simply recompute last sqrt(N) layers again.

Training Deep Nets with Sublinear Memory Cost

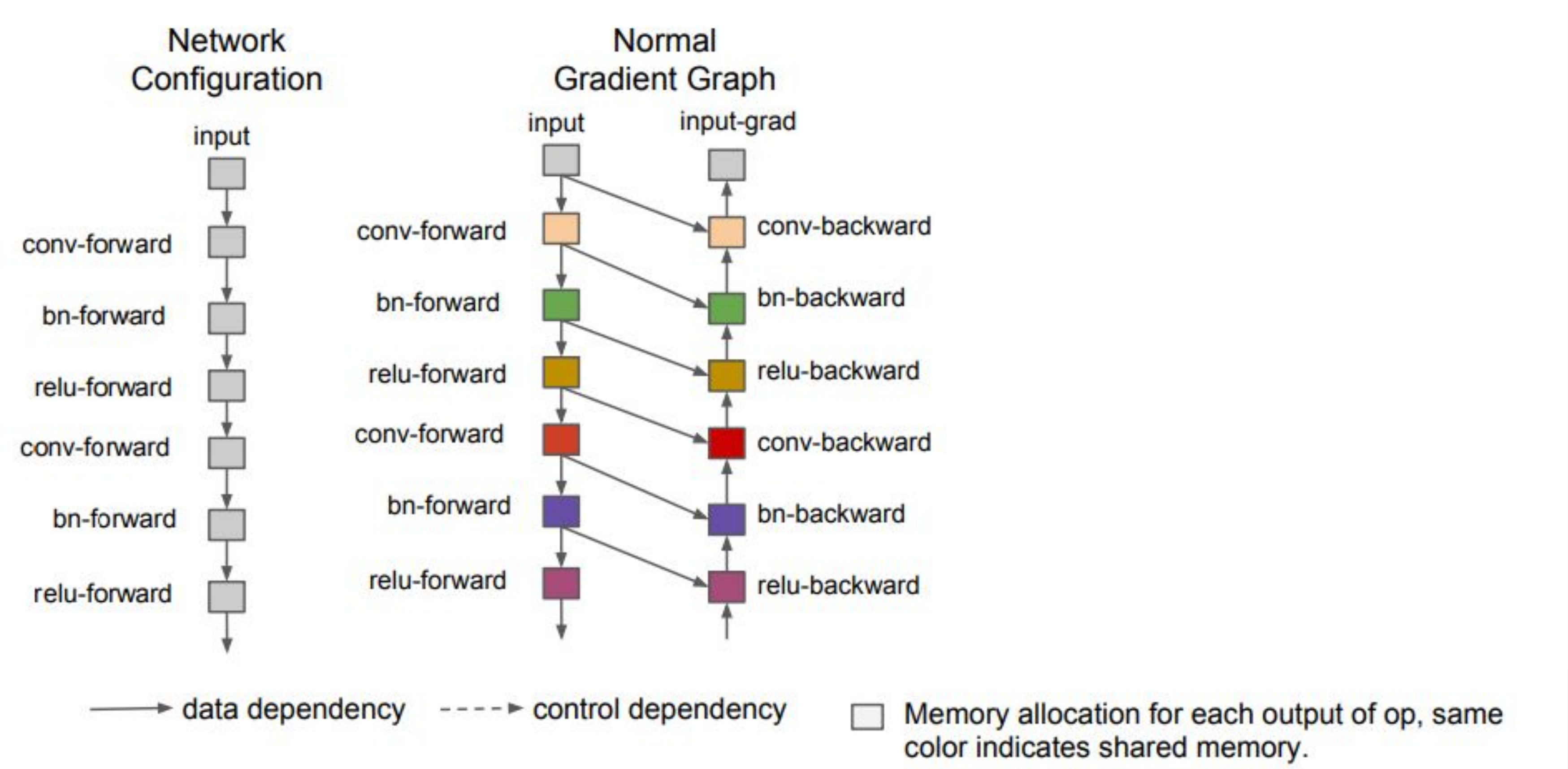
Tianqi Chen [1], **Bing Xu** [2], **Chiyuan Zhang** [3], and **Carlos Guestrin** [1]

[1] Unveristy of Washington [2] Dato. Inc [3] Massachusetts Institute of Technology

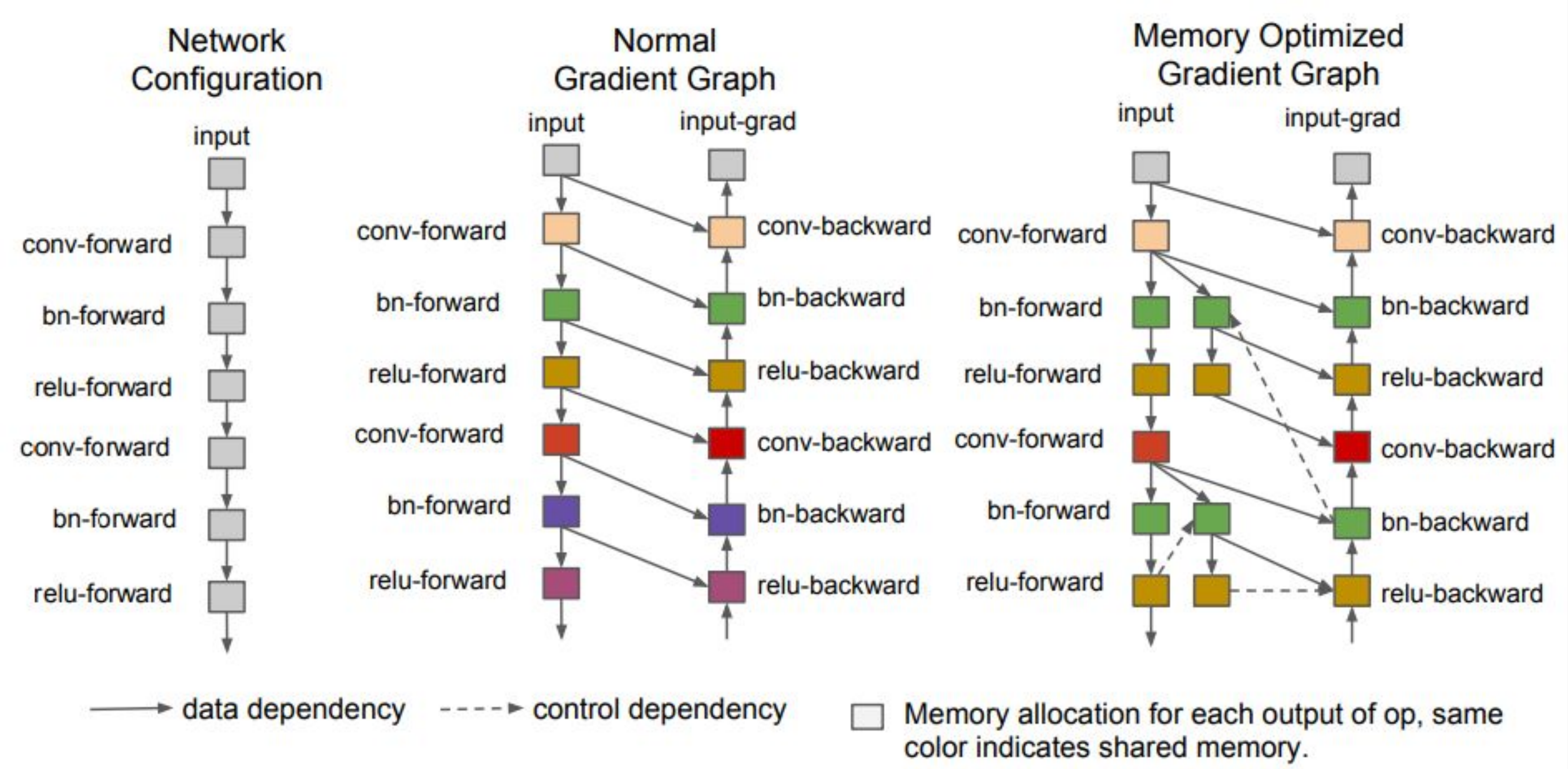
Gradient checkpointing



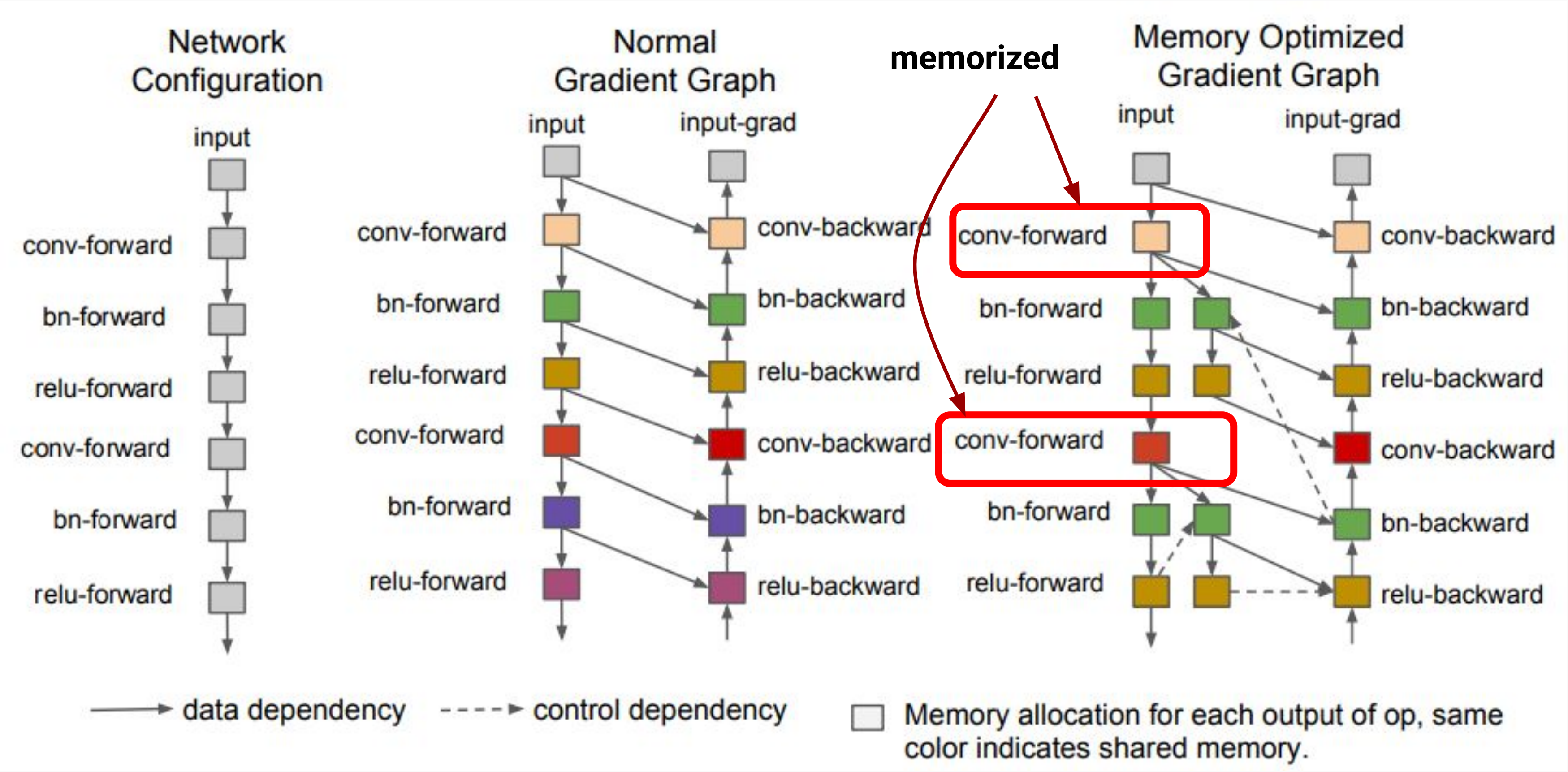
Gradient checkpointing



Gradient checkpointing



Gradient checkpointing



Gradient checkpointing

In optimal scenario, this approach can result in RAM usage around O(sqrt(#layers)) instead of O(#layers)!

This is at cost of doing most of the computation twice.

Can we do better?

Reformer:
Reversible Transformer

Reversible Transformer

If all our layers were fully reversible functions, we wouldn't need to remember any activations!

We would just recompute them all during backward pass, from the result.

Let's do that...

Reversible Transformer

Instead of one activation vector for each token, let's have two: X_1 and X_2.

Let's take our FeedForward or Attention functions, let's call them F and G.

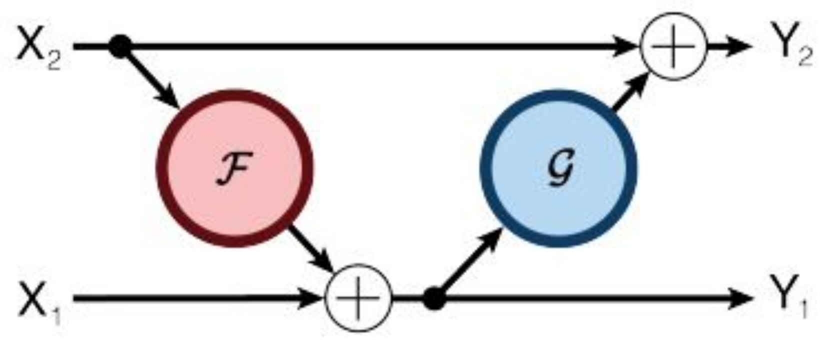
Reversible Transformer

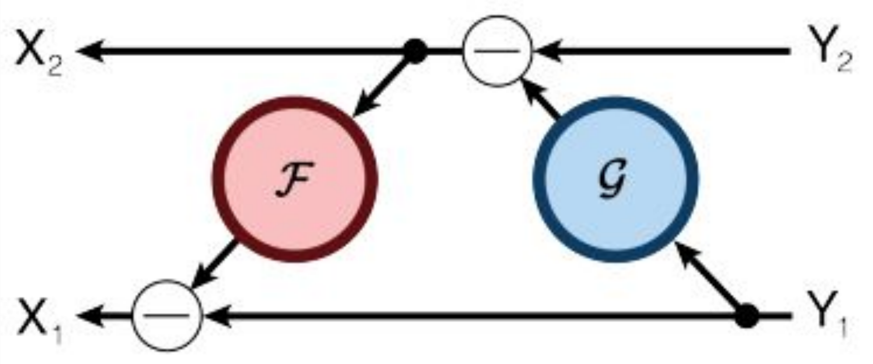
Forward pass:

$$Y_1 = X_1 + F(X_2)$$
$$Y_2 = X_2 + G(Y_1)$$

Backward pass:

$$X_2 = Y_2 - G(Y_1)$$
$$X_1 = Y_1 - F(X_2)$$





Pictures: Gomez et al., 2017

Reversible Transformer

With this approach, we can have RAM usage independent of number of layers!

Sidenote: **sometimes** you can have numerical stability problems. Why?

Because adding/subtracting is not **exactly** reversible on floats!

Thanks for listening

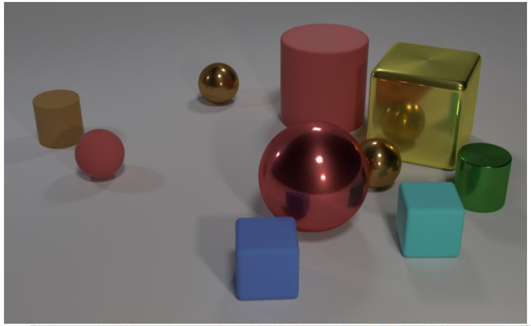
Visual Models and NLP

an interesting research cross section.

Spyridon Mouselinos
13/06/2022

Areas of application

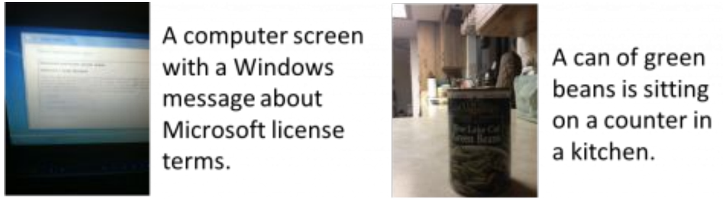
- Visual Question Answering
- Visual Reasoning



Q: Are there an **equal number** of **large things** and **metal spheres**?

Areas of application

- Captioning
- Retrieval



Areas of application

- Text-guided image generation

Prompt: A Shiba-inu in front of University of Warsaw.

Prompt: A tomato on the Parthenon.

The great question: How to mix modalities?

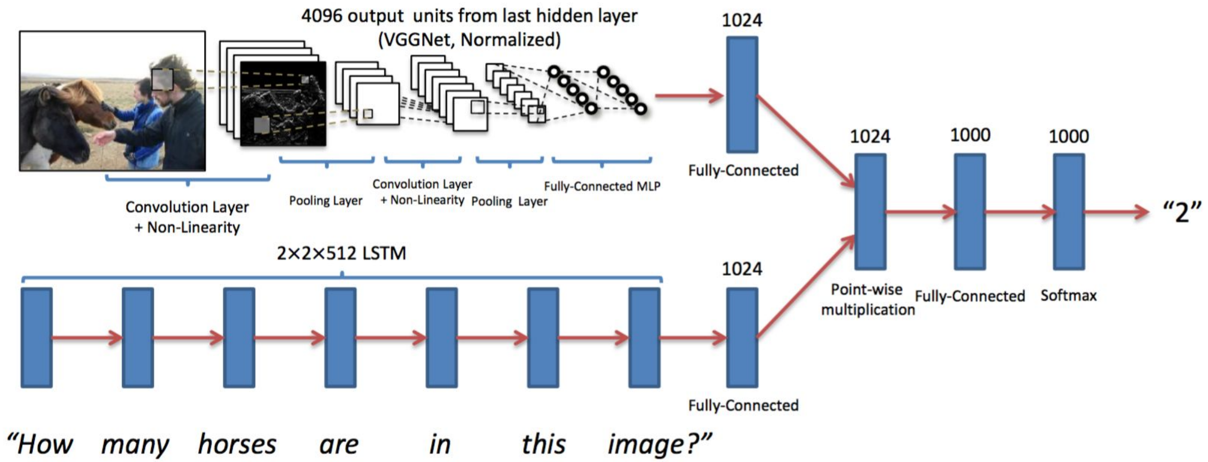
Let's focus on the generic VQA problem for a bit.

How would you do it?

"How many horses are in this image?"

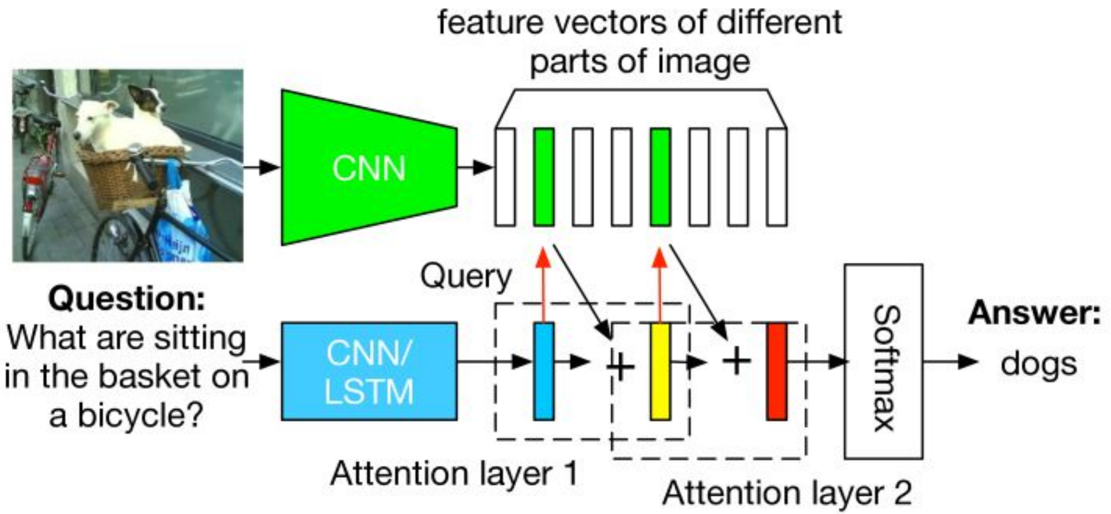
The great question: How to mix modalities?

Looks simple enough, will probably work under a specific dataset.



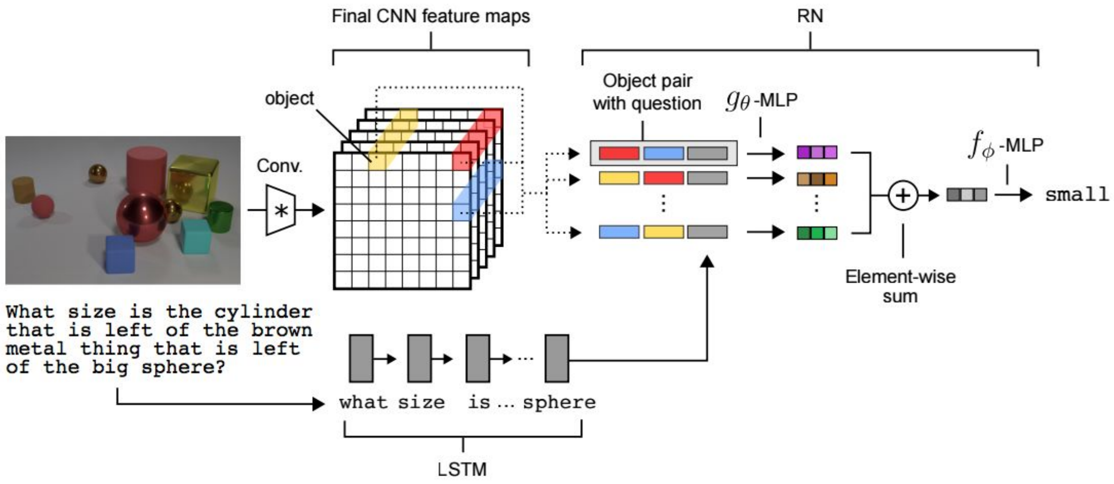
Evolution of mixing approaches

Stack Attention Networks 2016 - Yang et al.



Evolution of mixing approaches

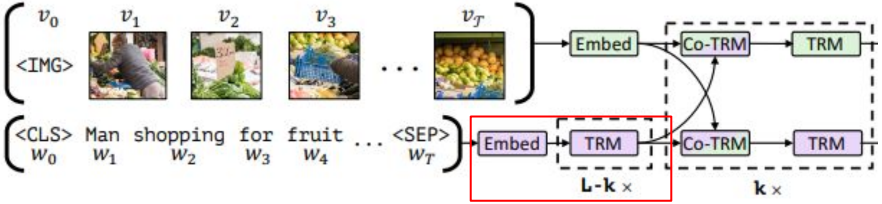
Relation Networks 2017 - Malinowski et al.

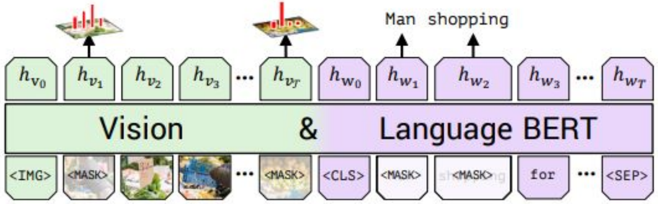


Evolution of mixing approaches

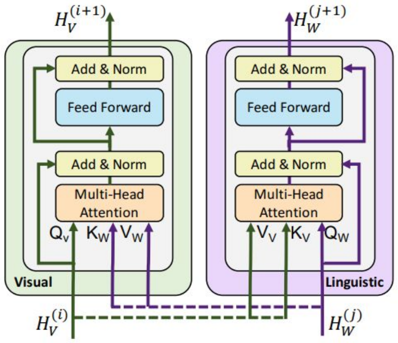
ViLBERT - 2019 Lu et al.

Object- Aware Image Patches with RCNN





(a) Masked multi-modal learning



(b) Our co-attention transformer layer

Train it like BERT!

Evolution of mixing approaches

MDETR - 2021 Kamath et al.

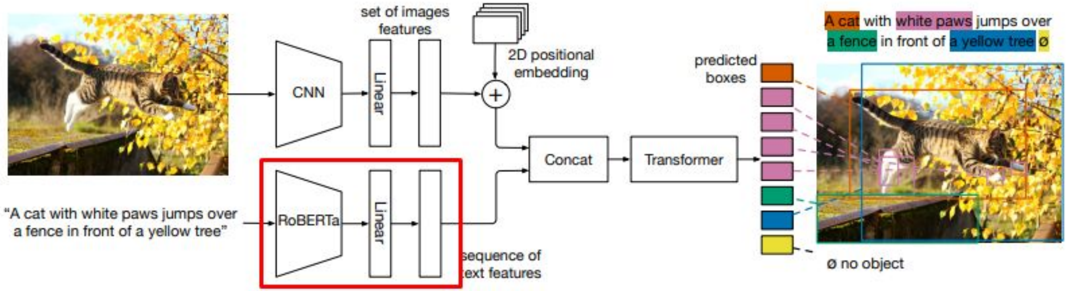
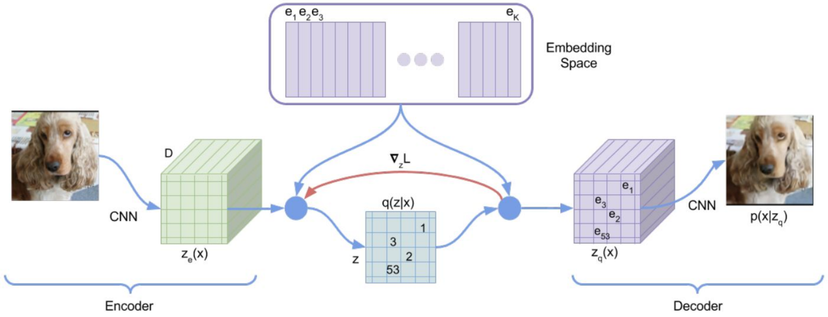


Figure 2: MDETR uses a convolutional backbone to extract visual features, and a language model such as RoBERTa to extract text features. The features of both modalities are projected to a shared embedding space, concatenated and fed to a transformer encoder-decoder that predicts the bounding boxes of the objects and their grounding in text.

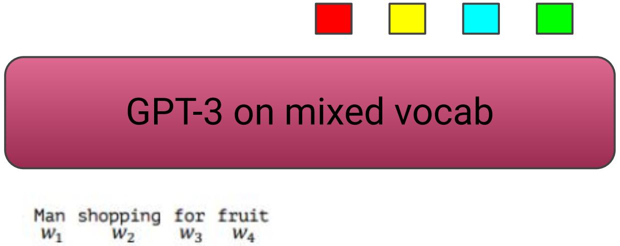
Generative modality mixing: DALL-E (OpenAI)

Can we use language to inform / instruct a generative model?

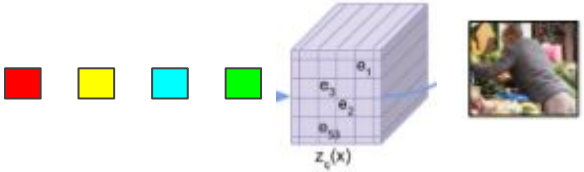
Step 1: Discretize Visual Concepts



Step 2: Use your prompt to generate visual concepts with a decoder-only model!



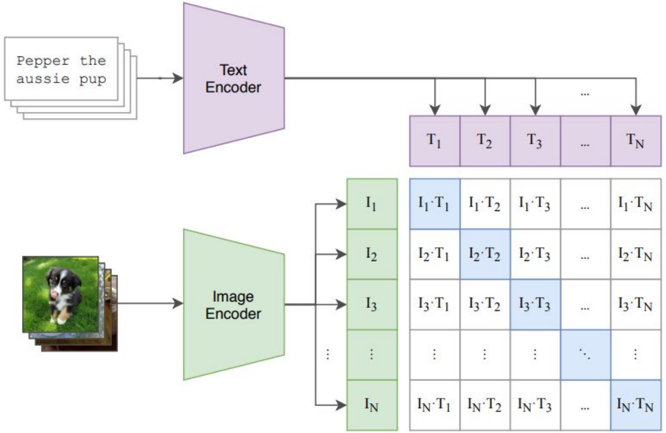
Step 3: Translate concepts back to an image



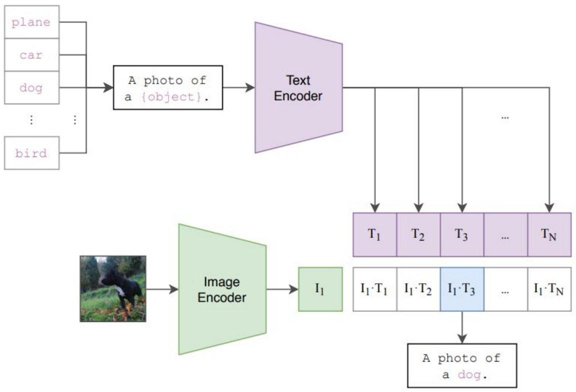
Retrieval and Alignment: CLIP (OpenAI)

Or how we learned to stop worrying about mixing modalities with contrastive learning.

Step 1) Train jointly text/image encoders to perform similarity matching.



Step 2) Go to a never seen before dataset and just translate labels into words. Then classify by finding the best similarity



Creating Visual Programs with language

FiLM 2017 - Perez et al.

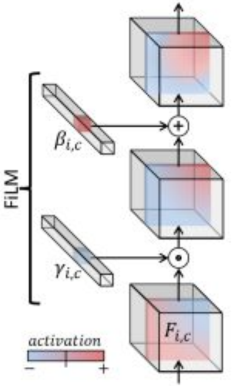
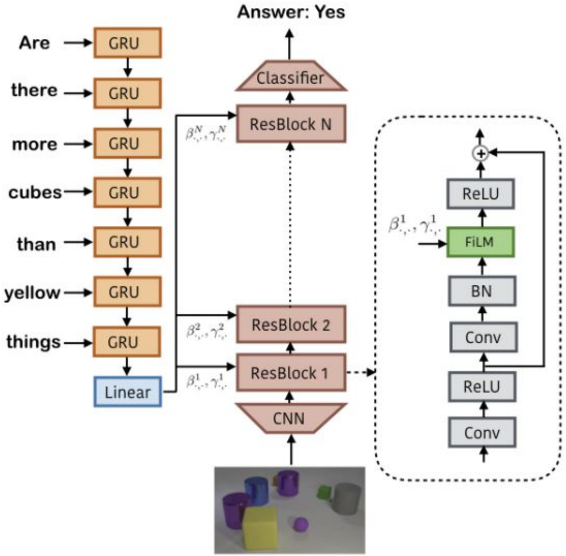


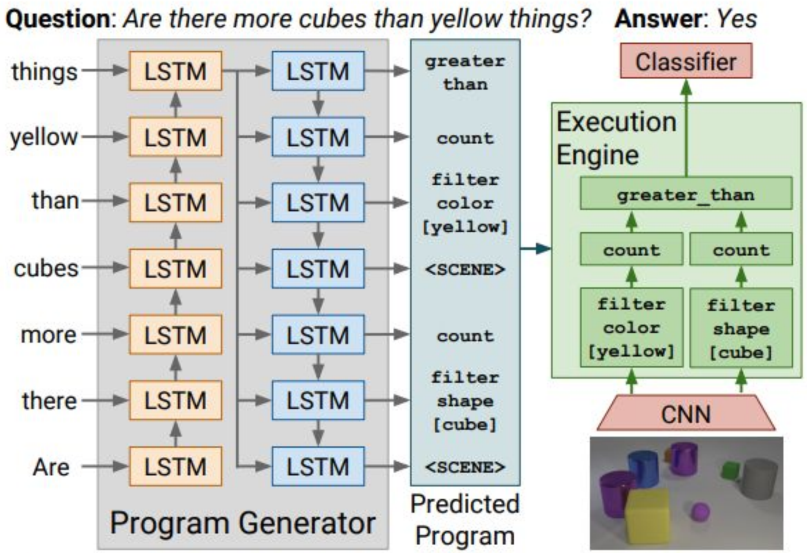
Figure 2: A single FiLM layer for a CNN. The dot signifies a Hadamard product. Various combinations of γ and β can modulate individual feature maps in a variety of ways.



“Soft Visual Program Generation”

Creating Visual Programs with language

IEP 2018: Johnson et al.



Many more things to explore

- Quality / Variety of image generation (have a look at Imagen by Google).
- Efficiency and effectiveness of Transformer Architectures in open-world VQA scenarios.
- Mixture of Vision Language and Robotics for autonomous systems.
- Scaling to Videos and Point Clouds as visual input.

Thank you for your time!