Digest

Neural Network Design, Scaling Laws and Transformers

Material sourced from 4F12 (Computer Vision) 2021 lecture series

Digest by Samuel Albanie, April 2022

Outline

- Strategies for Neural Network Design
- Scaling Phenomena
- Transformers

Strategies for Neural Network Design

Background

Modern deep learning stems from the **connectionist** approach, in which the wiring of computational networks plays an important role in building intelligent machines.

Conceptually, it can be helpful to categorise the structures that define the wiring between neural network units into two categories:

- Network architecture - connections between units that are (typically) fixed throughout training (e.g. operation types)
- Network parameters - connections between units that are updated during training (e.g. kernel weights learned via backpropagation)

Neural Network Design focuses principally on the finding good network architectures (although the distinction between the architecture and the parameters can be somewhat blurry).

Goals

We exist (probably) in a resource-limited environment. We have *limited supplies of*:

Energy | Computation | Memory | Time

For any given task, neural network design aims to produce architectures with:

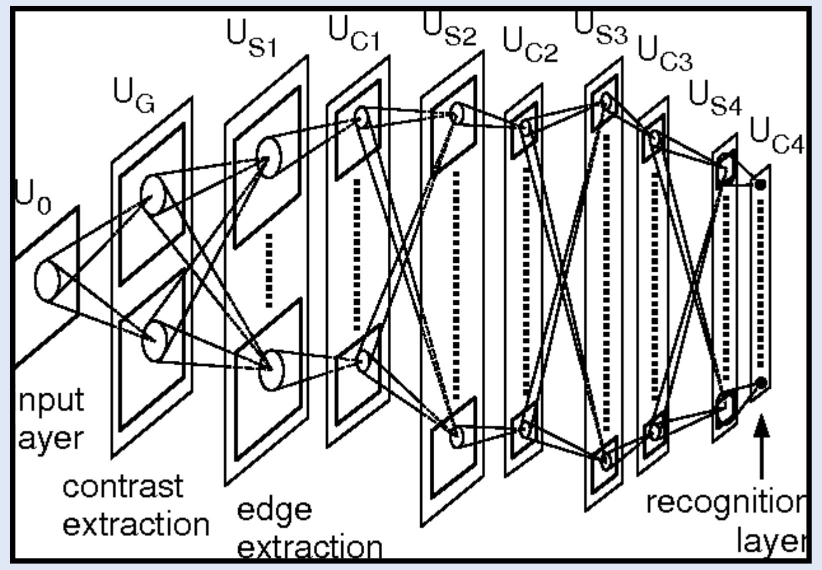
Greater task-specific performance (e.g. accuracy) | Lower resource burden

References/Image credits

J. A. Fodor and Z. W. Pylyshyn, "Connectionism and cognitive architecture: A critical analysis", *Cognition* (1988)

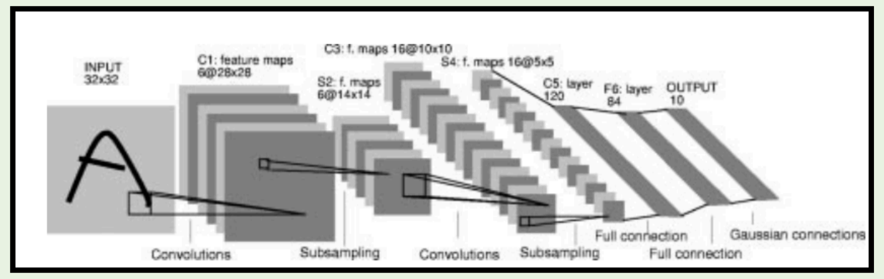
D.E. Rumelhart, G. E. Hinton and J. L. McClelland, "A general framework for parallel distributed processing", *PDP: Explorations in the microstructure of cognition* (1986)

Strategy 1: Neural Network Design by Hand



Fukushima and Miyake,
Neocognitron, 1982

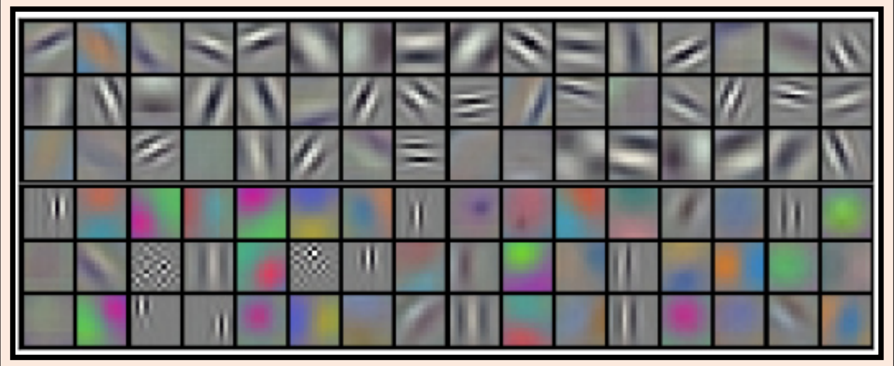
Convolutions



LeCun et al.,
LeNet, 1998

ILSVRC12

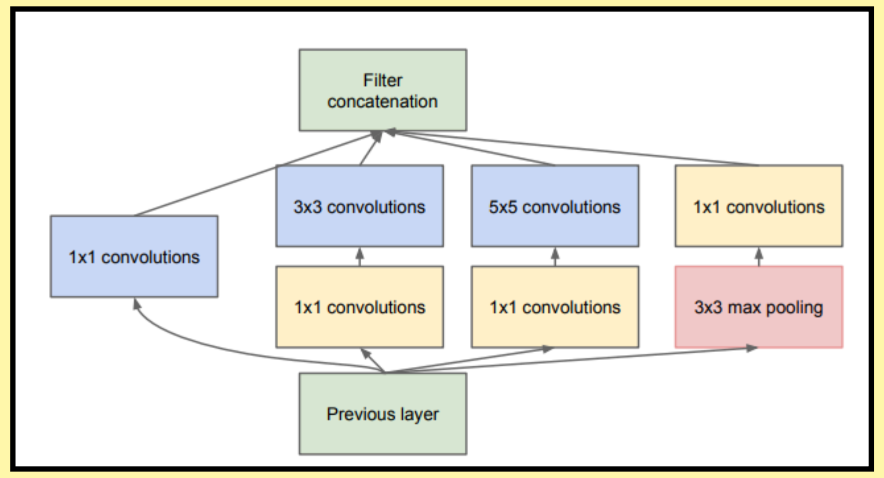
Scale up (with GPUs)



Krizhevsky et al.,
AlexNet, 2012

ILSVRC14

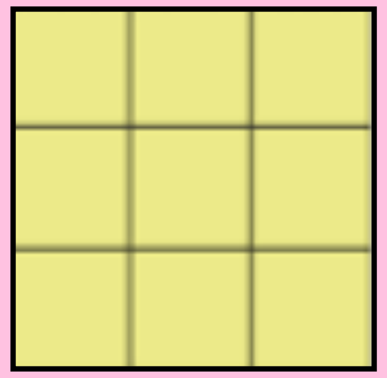
Multiscale



Szegedy et al.,
Inception, 2015

ILSVRC14

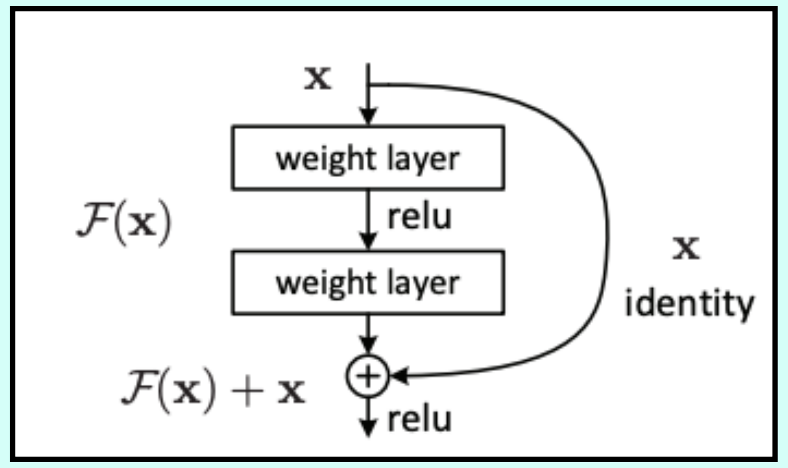
3x3 convs



Simonyan and Zisserman,
VGGNets, 2015

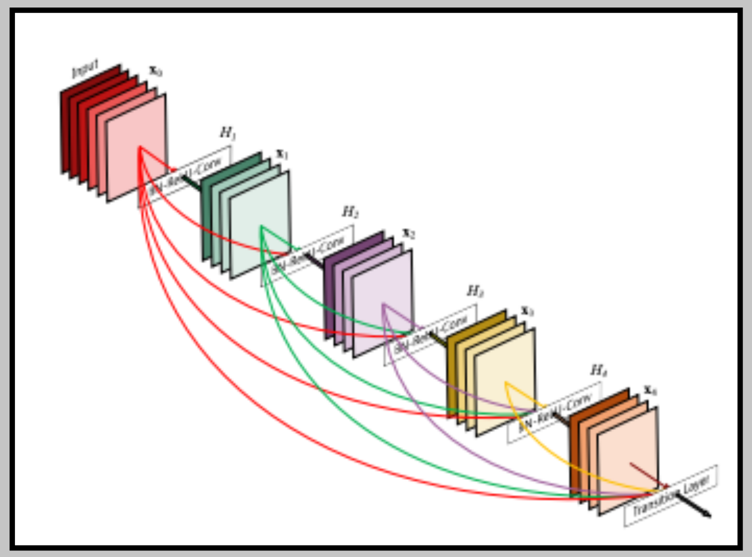
ILSVRC15

Residuals



He et al.,
ResNet, 2016

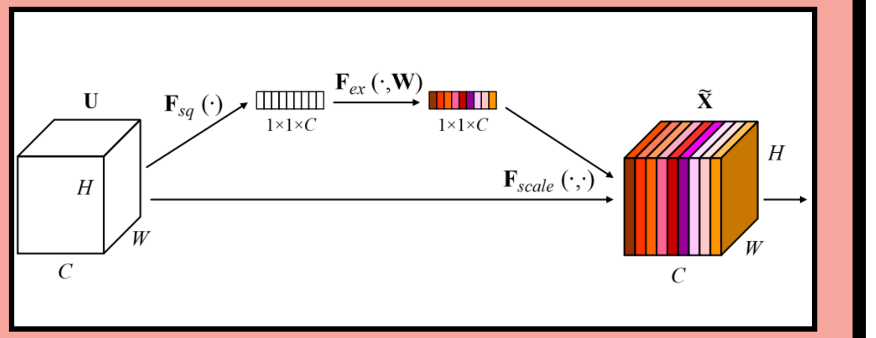
Dense Connections



Huang et al.,
DenseNets, 2017

ILSVRC17

SE Blocks



Hu et al.,
SENets, 2018

Aside: several of these architectures rose to prominence through strong performance on the ImageNet ILSVRC competition.

References/Image credits:

K. Fukushima and S. Miayke. "Neocognitron", CCNN (1982)

Y. LeCun et al., "Gradient-based learning applied to document recognition", IEEE (1998)

A. Krizhevsky et al., "Imagenet classification with deep CNNs", NeurIPS (2012)

C. Szegedy et al., "Going deeper with convolutions", CVPR (2015)

K. Simonyan et al. "Very Deep Convolutional Networks for Large-Scale Image Recognition", ICLR (2015)

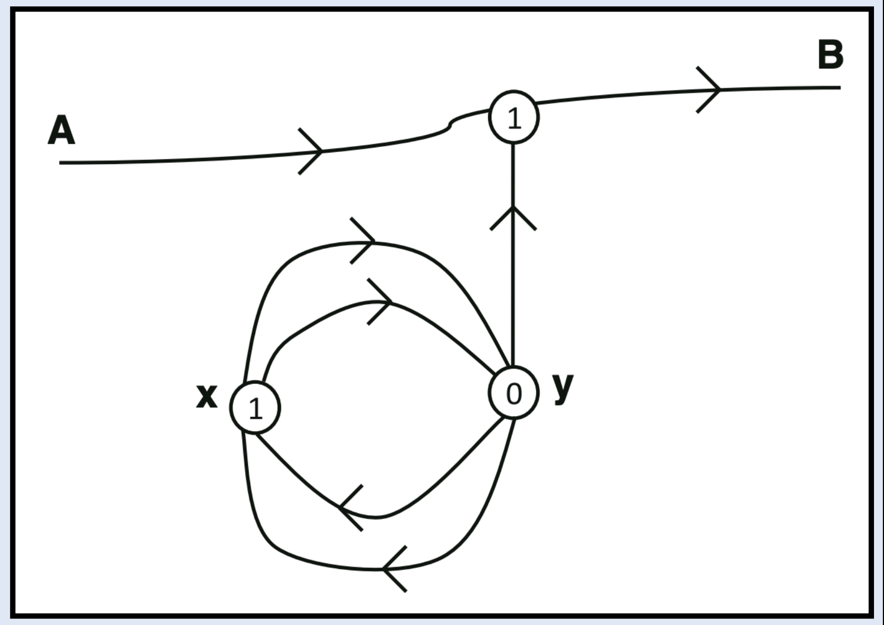
K. He et al., "Deep residual learning for image recognition", CVPR (2016)

G. Huang et al., "Densely connected convolutional networks", CVPR (2017)

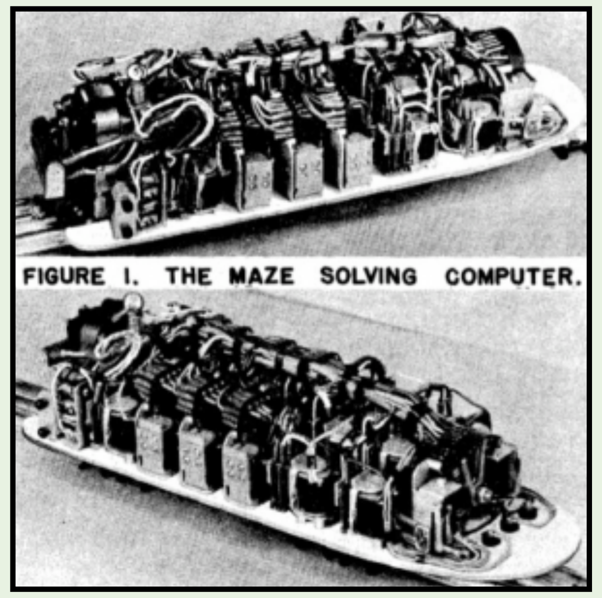
J. Hu et al., "Squeeze-and-Excitation Networks", CVPR (2018)

O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge", IJCV (2015)

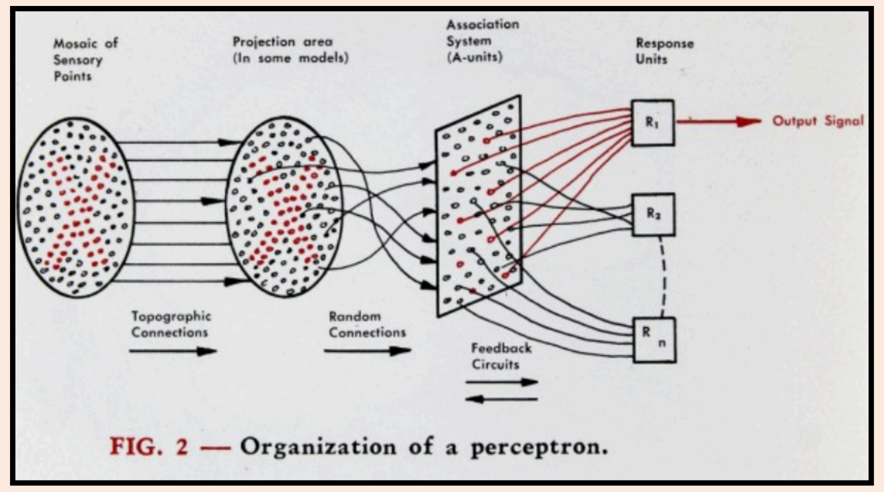
Strategy 2: Random Wiring



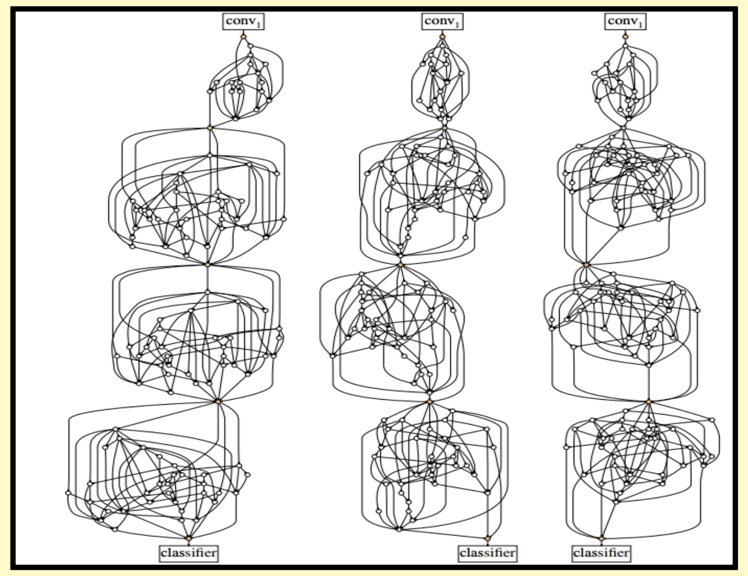
Turing, A-type/B-type
Unorganised machines, 1948



Minsky, SNARC, 1951

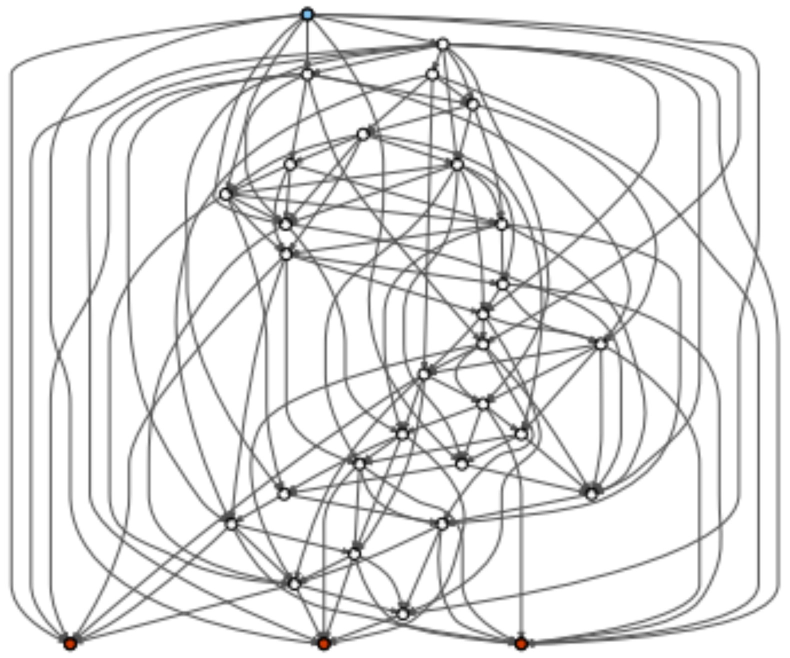


Rosenblatt, The Perceptron,
MARK I, 1957



Xie et al., Exploring Randomly
Wired Neural Networks for Image
Recognition, 2019

Randomly Wired Architectures

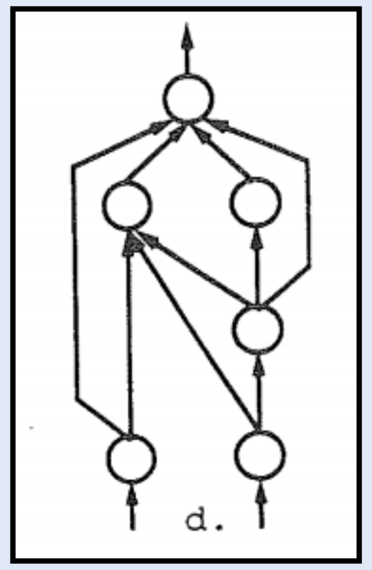


Method: Randomly sample connections between nodes
Different random graphs (e.g. Watts-Strogatz) produce
different architecture characteristics

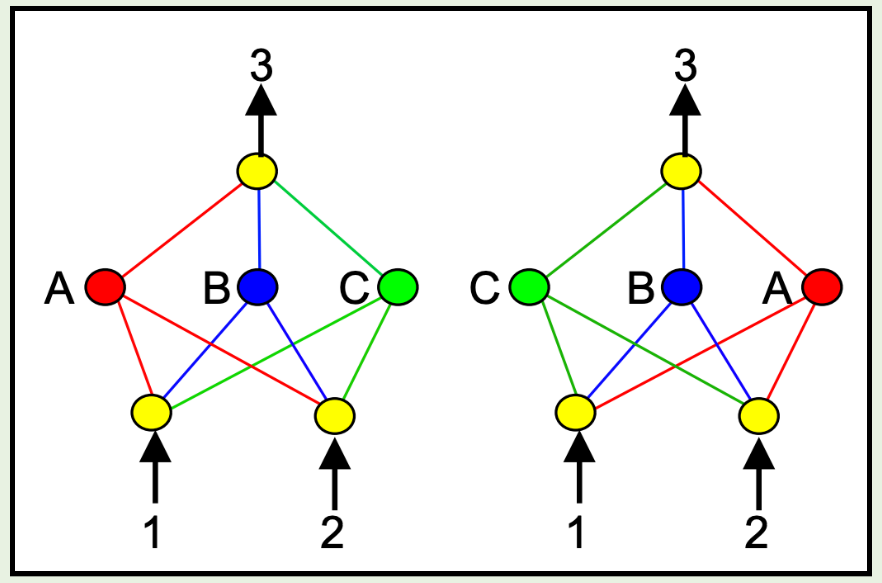
References/Image credits:

D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks", Nature (1998)
A. M. Turing, "Intelligent machinery" (1948)
(Figure) S. Russell and P. Norvig, "Artificial intelligence: a modern approach" (2002)
F. Rosenblatt, "The perceptron, a perceiving and recognizing automaton" Project Para. Cornell Aeronautical Laboratory (1957)
S. Xie et al., "Exploring randomly wired neural networks for image recognition", CVPR (2019)
R. Girshick, https://neuralarchitects.org/slides/girshick-slides.pdf (2019)

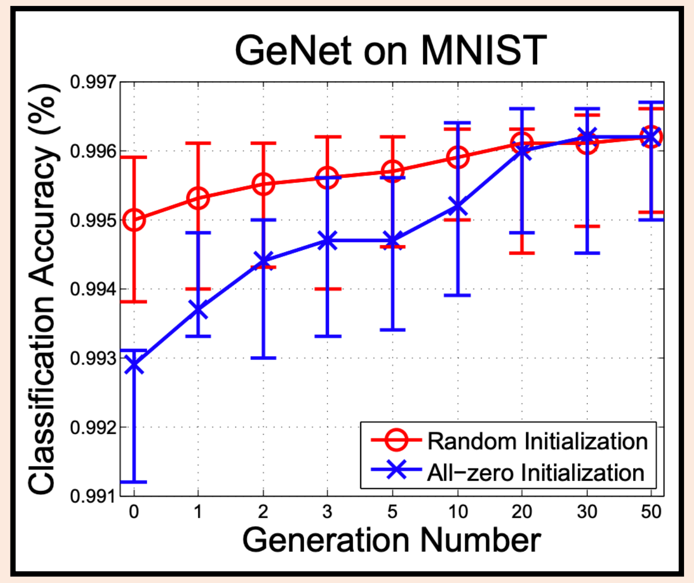
Strategy 3: Evolutionary Algorithms



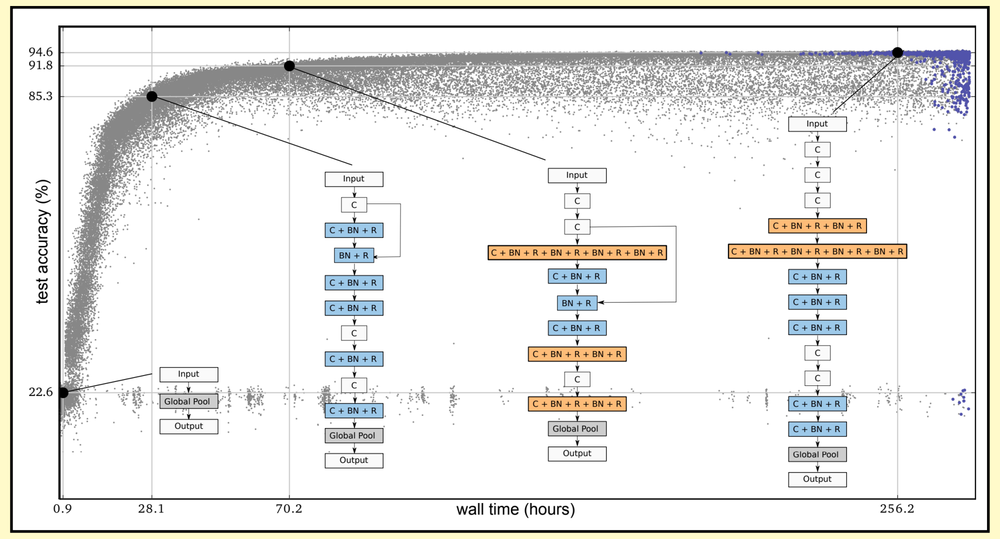
Miller et al.,
“INNERVATOR”, 1989



Stanley and Miikkulainen,
NEAT, 2002

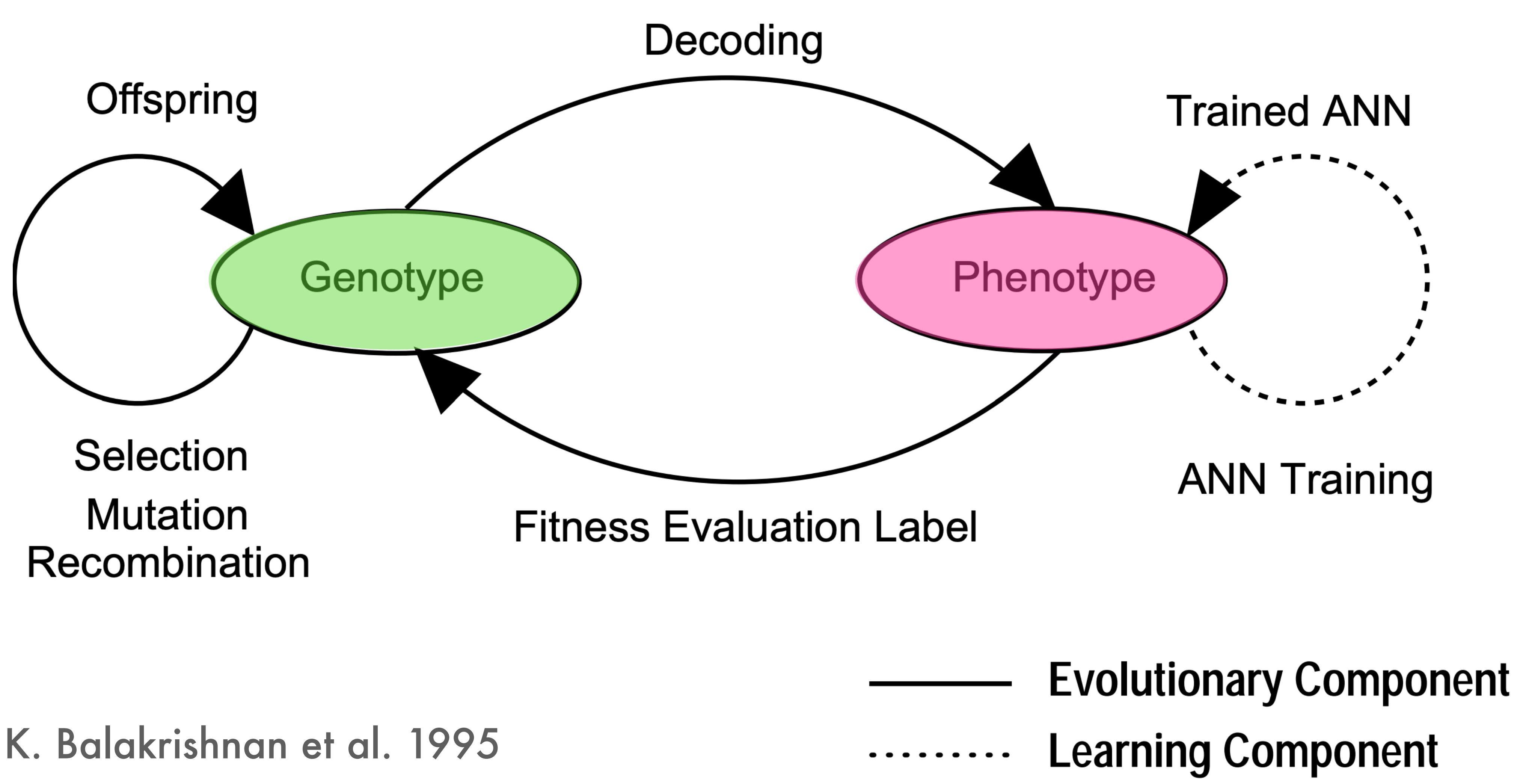


Xie and Yuille,
Genetic CNN, 2017



Real et al. Large-Scale Evolution of
Image Classifiers, 2017

Evolutionary Algorithms for Network Architectures



K. Balakrishnan et al. 1995

References/Image credits:

Figure sourced from K. Balakrishnan et al., "Evolutionary Design of Neural Architectures – A Preliminary Taxonomy and Guide to Literature" (1995)

I. Rechenberg, "Cybernetic solution path of an experimental problem", Royal Aircraft Establishment (1965)

J. Holland, "Adaptation in natural and artificial systems" (1975)

P. M Todd, "Evolutionary methods for connectionist architectures", Unpublished manuscript, (1988)

G. F. Miller et al., "Designing neural networks using genetic algorithms", ICGA (1989)

K. O. Stanley et al., "Evolving neural networks through augmenting topologies", Evolutionary computation (2002)

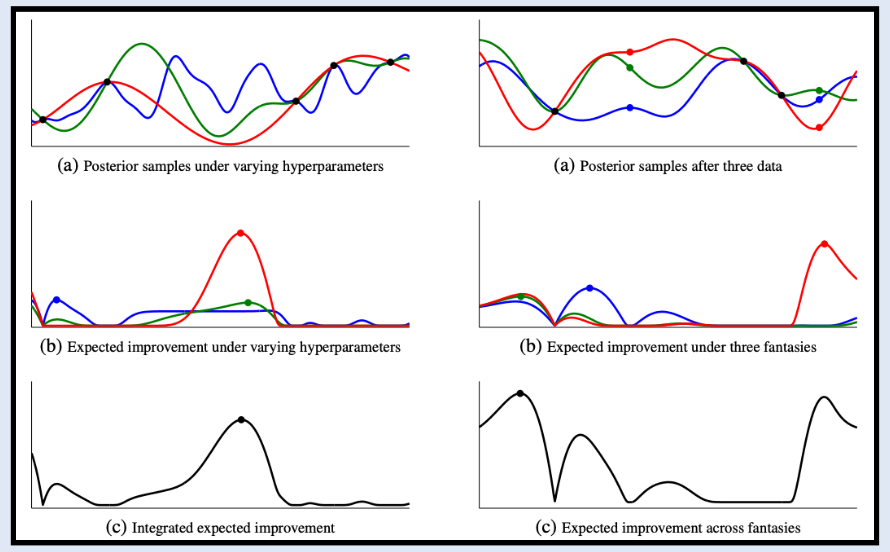
J. Bayer et al., "Evolving memory cell structures for sequence learning", ICANN (2009)

R. Jozefowicz et al., "An empirical exploration of recurrent network architectures", ICML (2015)

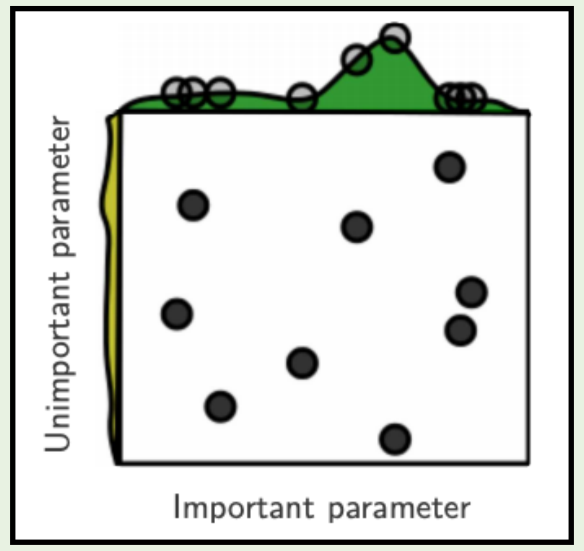
L. Xie and A. L. Yuille, “Genetic CNN”, ICCV (2017)

E. Real et al., "Large-scale evolution of image classifiers", ICML (2017)

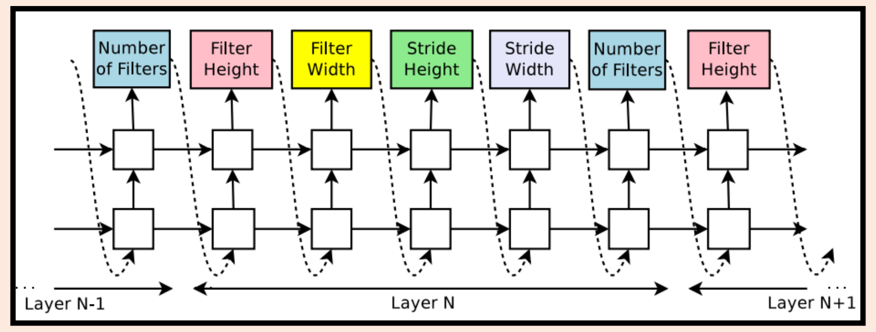
Strategy 4: Neural Architecture Search



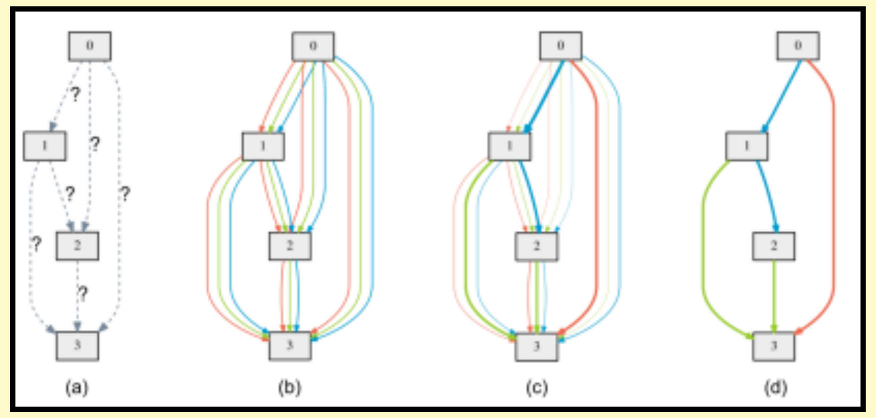
Snoek et al., Practical Bayesian Optimization of Machine Learning Algorithms, 2012



Bergstra and Bengio, Random search for hyper-parameter optimization, 2012



Zoph and Le, *Neural architecture search with reinforcement learning, 2017*



Liu et al. DARTS: Differentiable Architecture Search, 2019

References:

J. Snoek et al., "Practical bayesian optimization of machine learning algorithms", NeurIPS (2012)

J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization", JMLR (2012)

B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning", ICLR (2017)

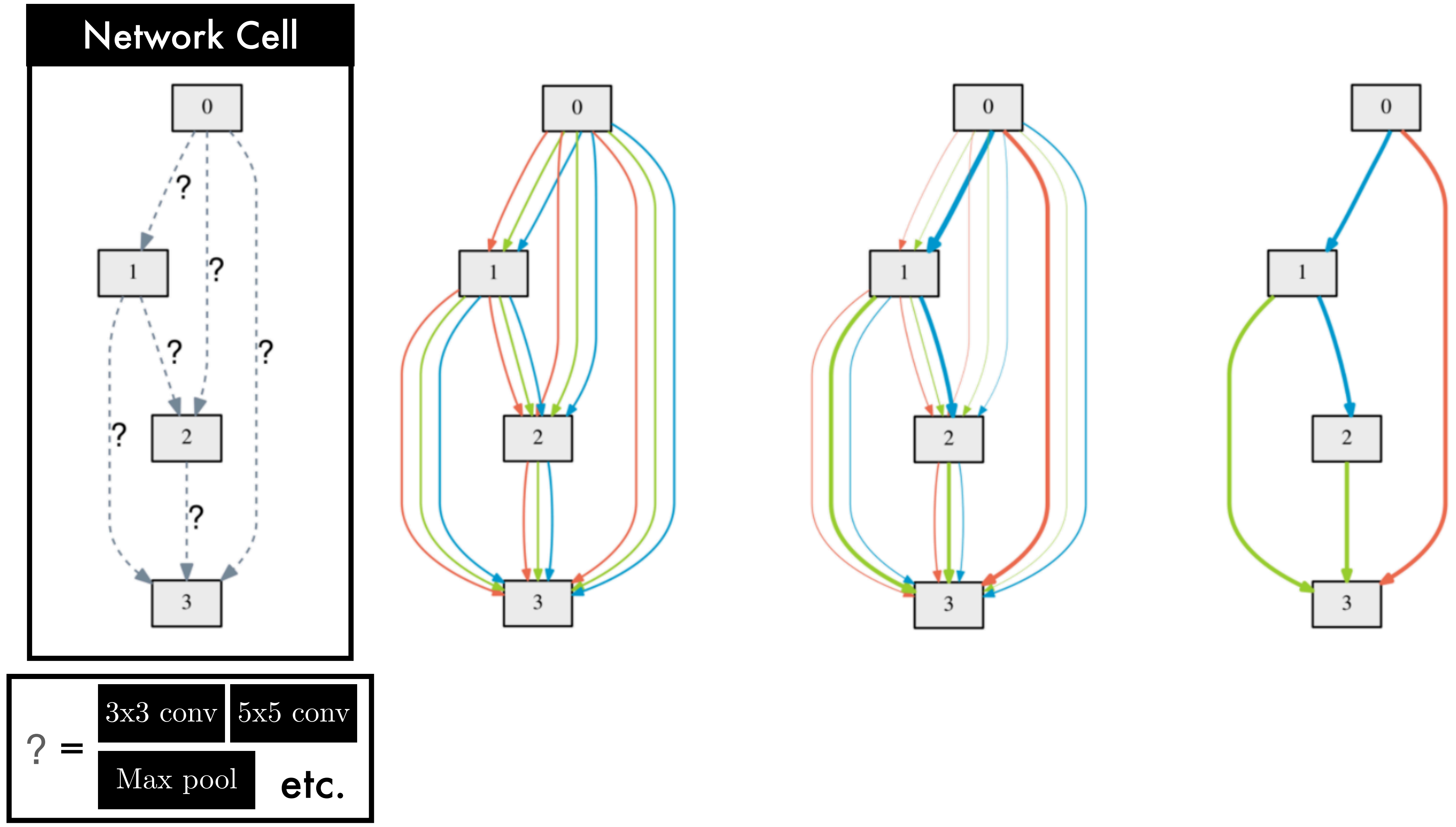
B. Baker et al., "Accelerating neural architecture search using performance prediction", arXiv (2017)

H. Liu et al., "Darts: Differentiable architecture search", ICLR (2019)

DARTS: Differentiable Architecture Search

Challenge: architecture search is non-differentiable

Problem: Network performance (e.g. accuracy) does not change smoothly w.r.t architecture changes
- we cannot use gradient-based optimisation :(



DARTS solution: solve a continuous relaxation of the problem. To learn a cell:

- Place a mixture (weighted sum) of operations on each edge
- Jointly optimise network parameters and mixture probabilities
- Induce final architecture from mixing probabilities

Bilevel Optimisation

Each node can be computed from predecessors:

$$x^{(j)} = \sum_{i<j} o^{(i,j)}(x^{(i)})$$ operation from node i to node j

Relaxation: Consider mixtures of candidate operations, $\mathcal{O}$, via:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$$ operation weights

The goal is then to learn $\alpha = \{\alpha^{(i,j)}\}$.

Let $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ denote training/validation loss.

Let w the denote network parameters (e.g. convolution weights).

We'd like to solve a bilevel optimisation problem:

$$\min_{\alpha} \mathcal{L}_{val}(w^*(\alpha), \alpha)$$ α is the upper-level variable

$$\text{s..t. } w^*(\alpha) = \text{argmin}_w \mathcal{L}_{train}(w, \alpha)$$ w is the lower-level variable

Evaluating architecture gradients is prohibitively slow (the inner loop requires training a network) so we use an approximation:

1 step of gradient descent

$$\nabla_\alpha \mathcal{L}_{val}(w^*(\alpha), \alpha) \approx \nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$$

No formal convergence guarantees, but works in practice...

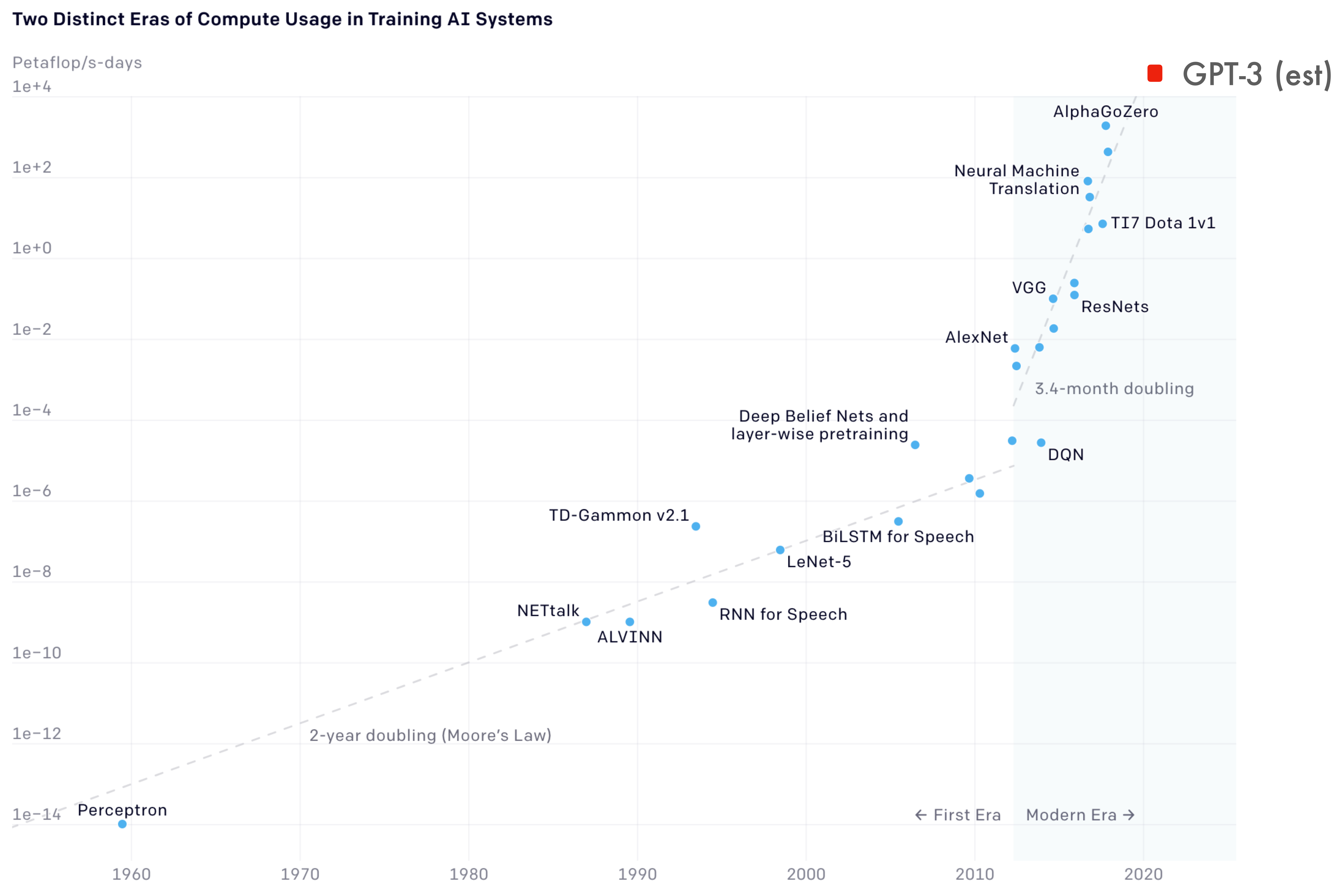
References/Image credits

H. Liu et al., "Darts: Differentiable architecture search", ICLR (2019)

Outline

- Strategies for Neural Network Design
- Scaling Phenomena
- Transformers

Scaling phenomena and the role of hardware



1 petaflop-day is approx.
8 V100 GPUs running for 1 day

GPT-3 (175B parameters) reportedly trained on a server with several thousand GPUs

Megatron-Turing NLG 530B (Nov, 2021) trained on 4K A100 GPUs

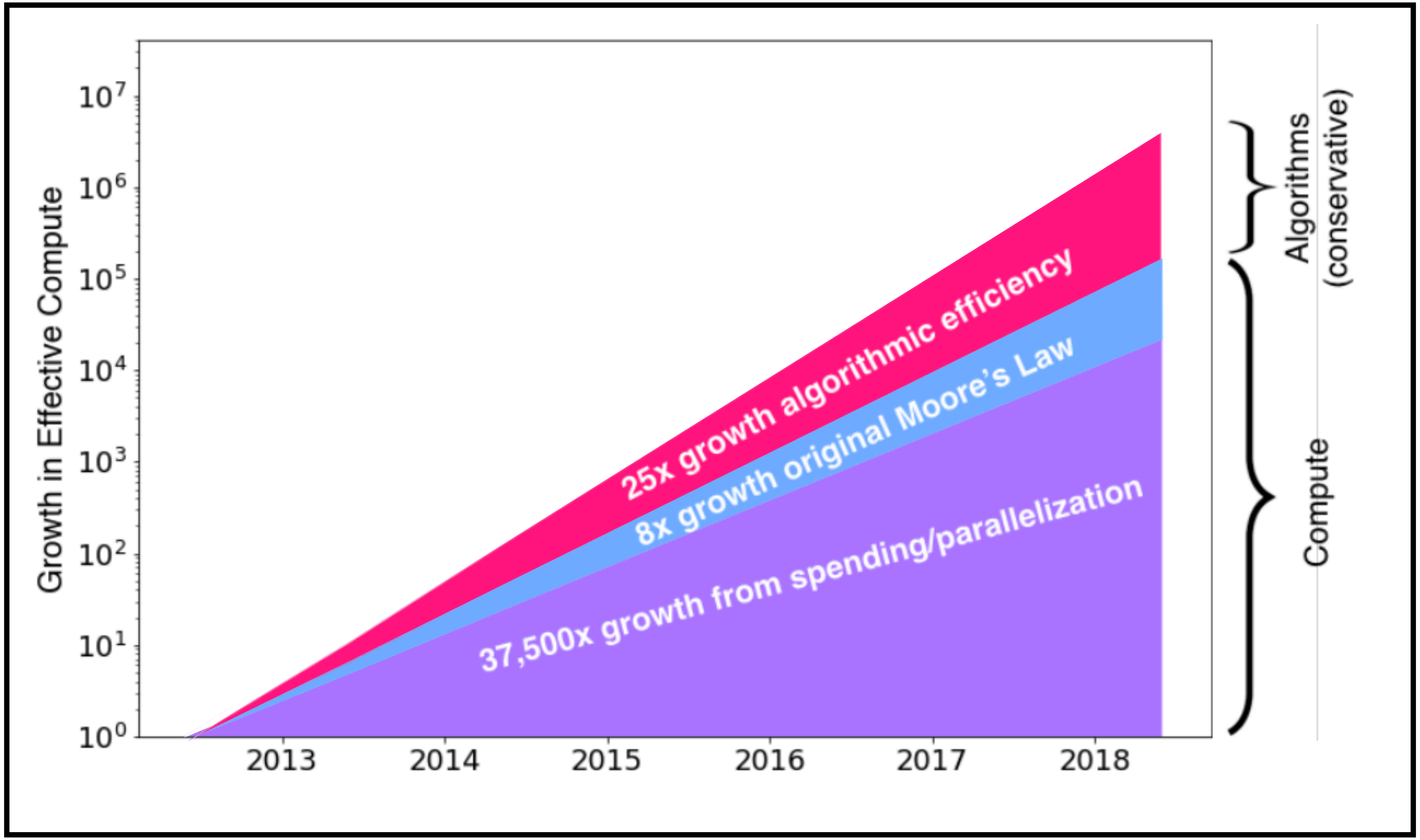
References/Image credits

D. Amodei and D. Hernandez, "AI and Compute", 2018

What factors are enabling effective compute scaling?

Effective compute ≈ FLOPs required to reach AlexNet-level ImageNet performance

Estimated cost of cloud compute for models like GPT-3: O(10 Million) USD



References/Image credits

D. Hernandez and T. Brown, "Measuring the Algorithmic Efficiency of Neural Networks", arXiv (2020)
https://twitter.com/eturner303/status/1266264358771757057

Scaling phenomena and the role of hardware

How important is scale for Deep Neural Networks?

Is it "just engineering", or something more fundamental?

Note: It is often challenging to analyse shifts from quantitative to qualitative differentiation.

Hierarchy of sciences

Is cell biology "just" applied molecular biology?
Is molecular biology "just" applied chemistry?
Is chemistry "just" applied many-body physics?
....
One science obeys the laws of the other.
But at each stage, new laws and concepts are necessary.

Qualitative vs Quantitative

FITZGERALD: The rich are different from us.
HEMINGWAY: Yes, they have more money.

"In almost all fields, a factor of ten means fundamentally new effects. If you increase magnification by a factor of 10 in Biology, you will see new things."

Hamming, Art of doing science and engineering, 1997

References/Footnotes:

P. Anderson, "More is different", Science (1972)

The "wisecrack" of Hemingway appears as a comment made by a character in one of his novels (http://www.quotecounterquote.com/2009/11/rich-are-different-famous-quote.html)

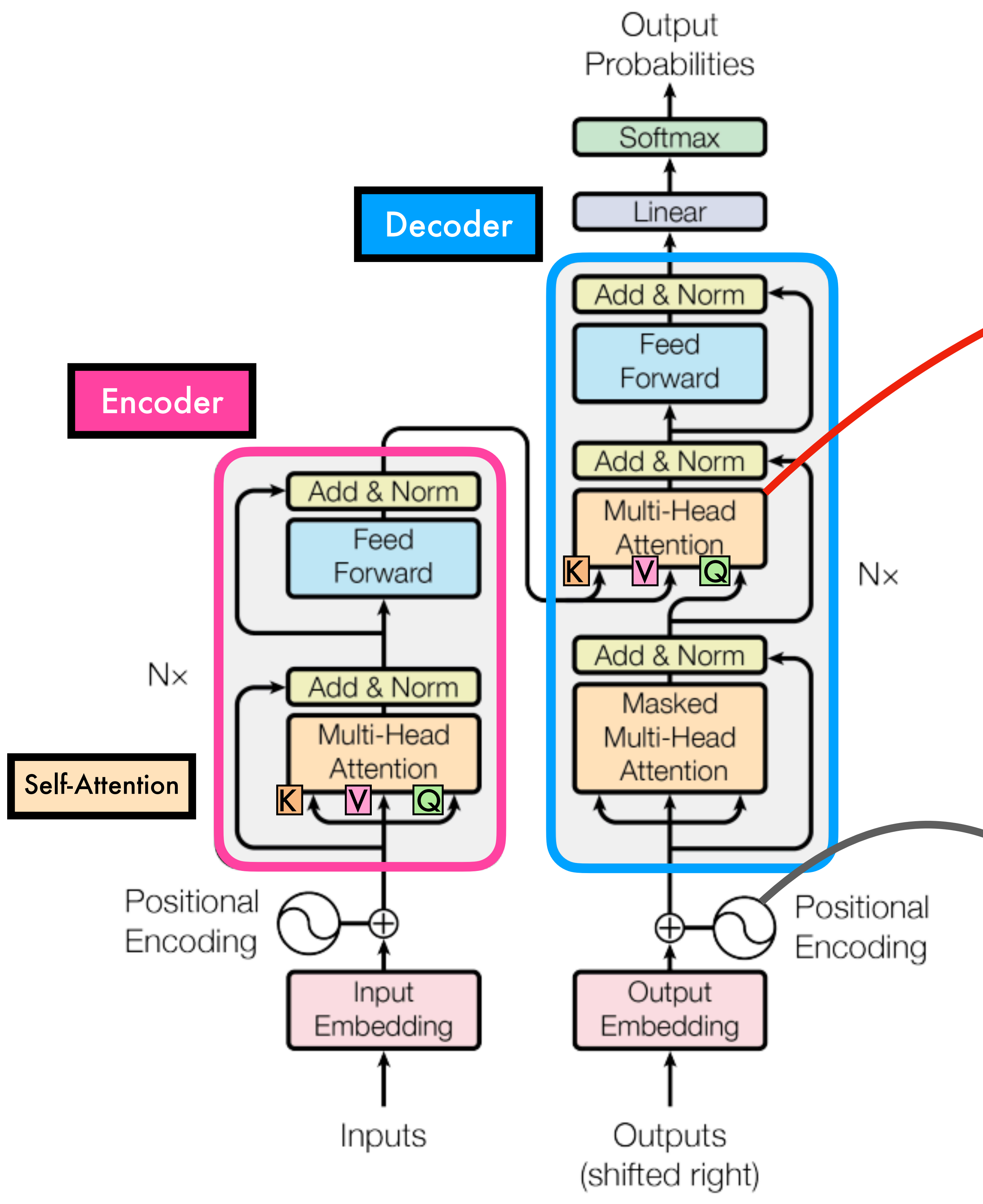
R. Hamming, "The Art of Doing Science and Engineering: Learning to Learn" (1997)

Outline

- Strategies for Neural Network Design
- Scaling Phenomena
- Transformers

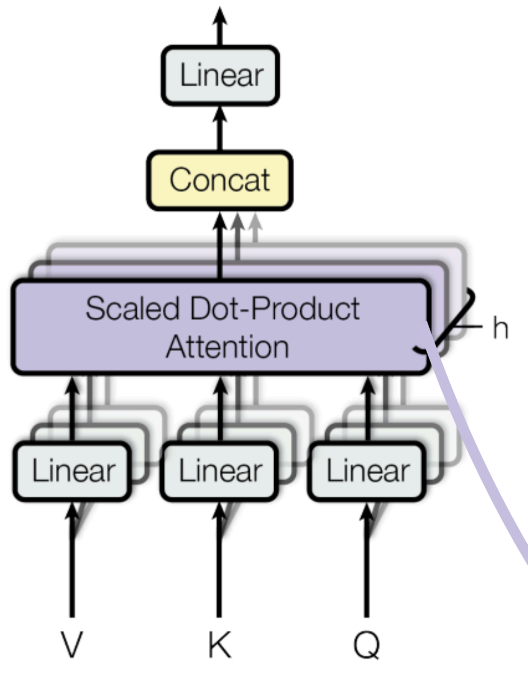
The Transformer: a model that scales particularly well...

Transformer Architecture



Note: Unlike recurrent networks, transformers are amenable to parallelisation.

Multi-Head Attention

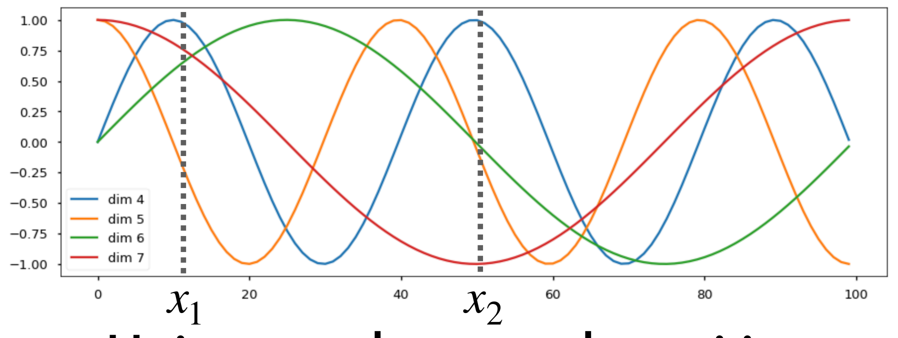


Concatenate attention outputs

Positional Encoding

By default queries and keys have no information about sequence position. Solution - add a code for position:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_m}}}\right) \qquad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_m}}}\right)$$



Unique code at each position

Scaled Dot-Product Attention

Intuition: The humble Python dictionary

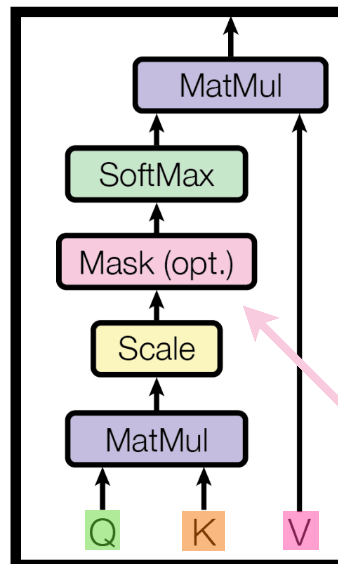
Keys (K) Values (V) Query (Q)

```
subject_quality = {
    "computer_vision": 10,
    "everything_else": 3,
}
print(subject_quality["computer_vision"]) # 10
```

```
print(subject_quality["machine_learning"]) # ?
# raises KeyValueError :(
```

But suppose we had a similarity function, `sim()`, that told us how similar the query is to each key.

```
sim("machine_learning", "computer_vision") = 0.8
sim("machine_learning", "everything_else") = 0.2
machine_learning_quality = 0.8 * 10 + 0.2 * 3 = 8.6
```



Ensures similarities sum to 1

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaling to improve softmax gradients (d_k is key dimensionality)

Masking is optionally applied to control which keys the query is compared to

References/Image credits

A. Vaswani et al., “Attention is All you Need”, NeurIPS (2017)

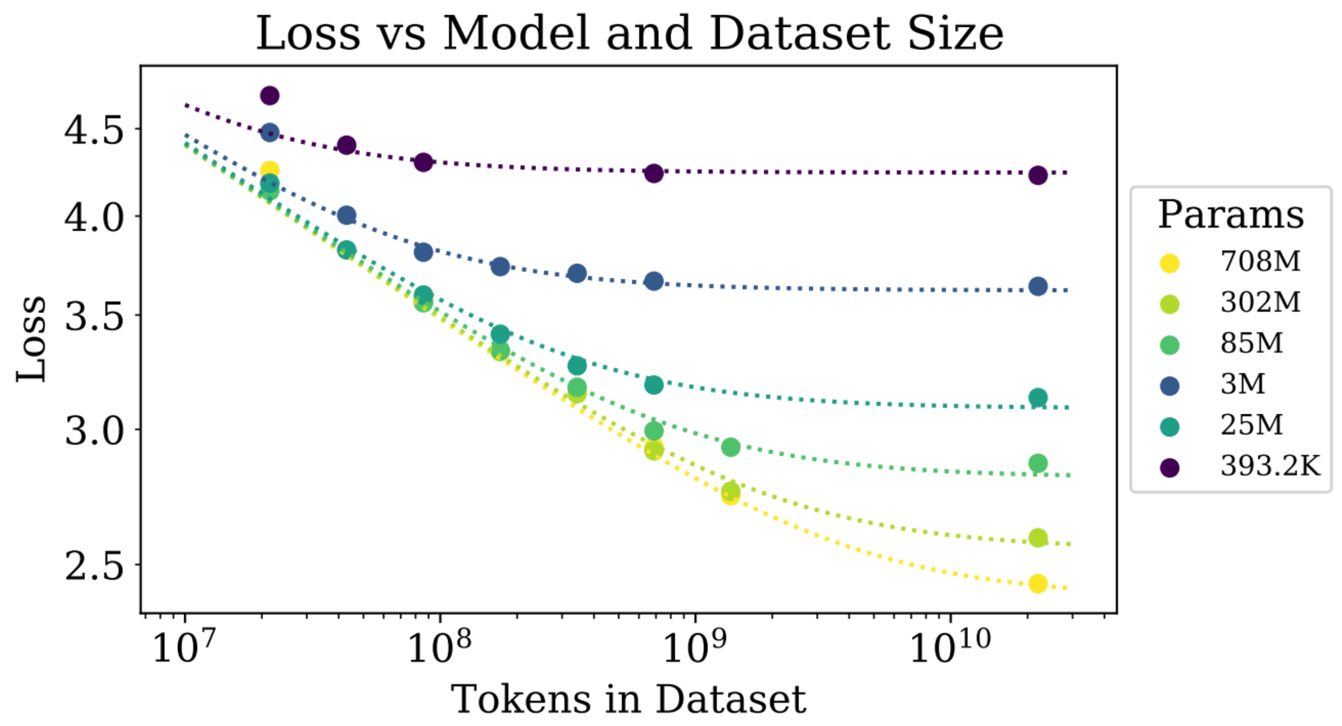
A. Rush, "The Annotated Transformer", https://nlp.seas.harvard.edu/2018/04/03/attention.html (2018)

Transformer scaling laws for natural language

Predictable scaling

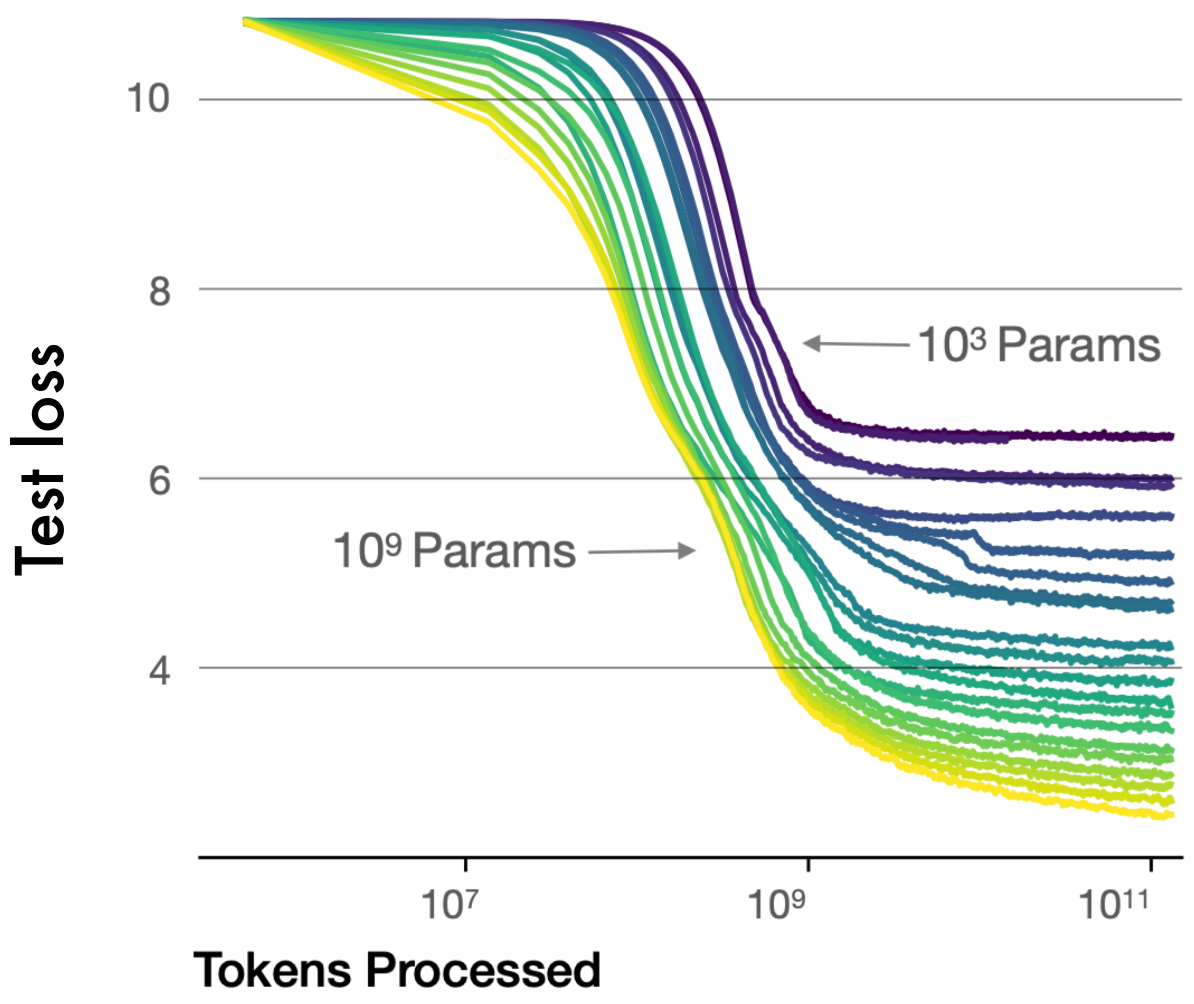
Transformer performance on language modelling tasks scales predictably as a *power law* with:

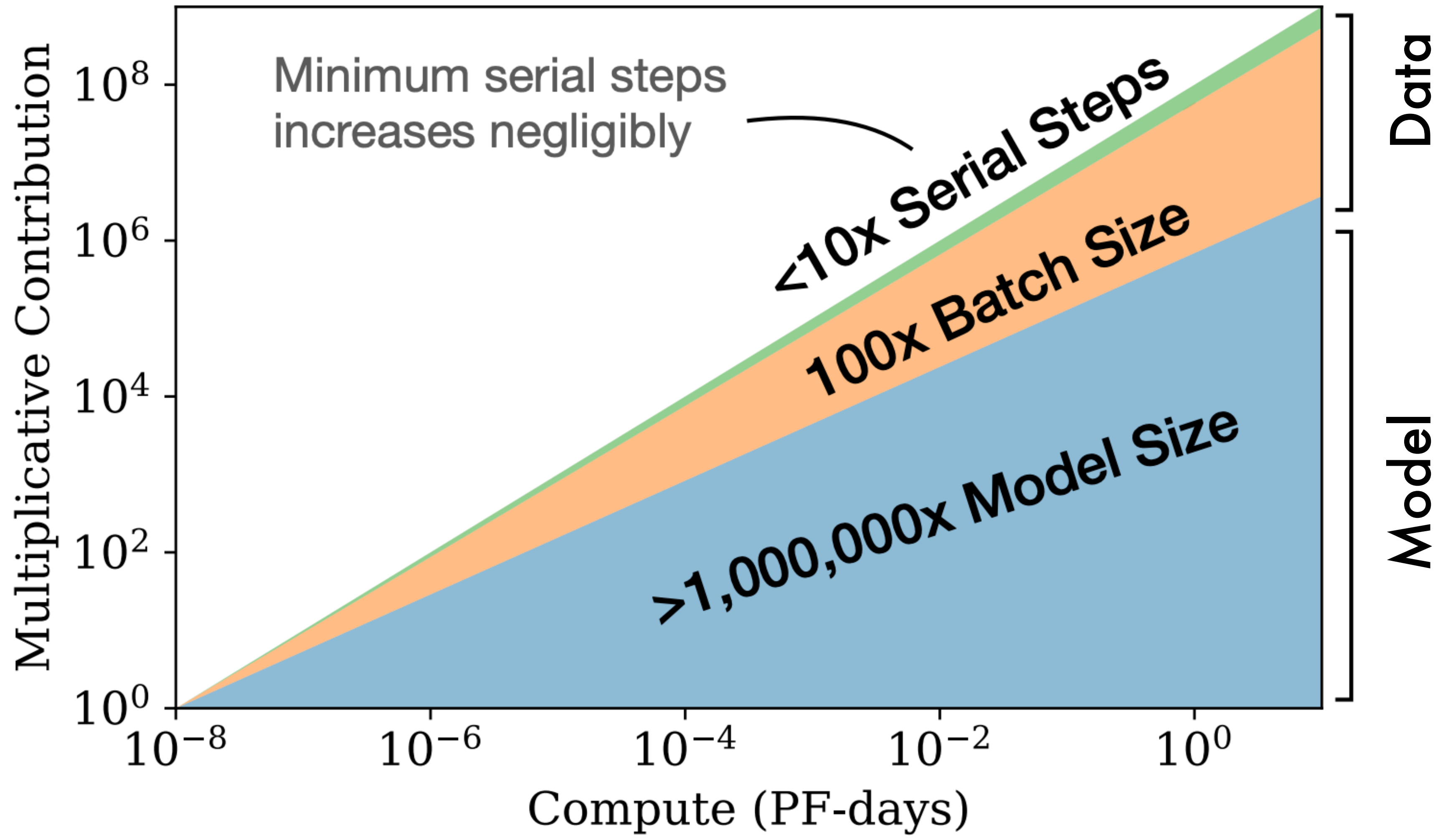
- Compute
- Training data size
- Model size



Some power laws were found that span more than seven orders of magnitude.

Intriguing characteristics





Larger models require **fewer samples** to reach the same performance.

If extra compute is available, allocate most towards increasing the **model size**!

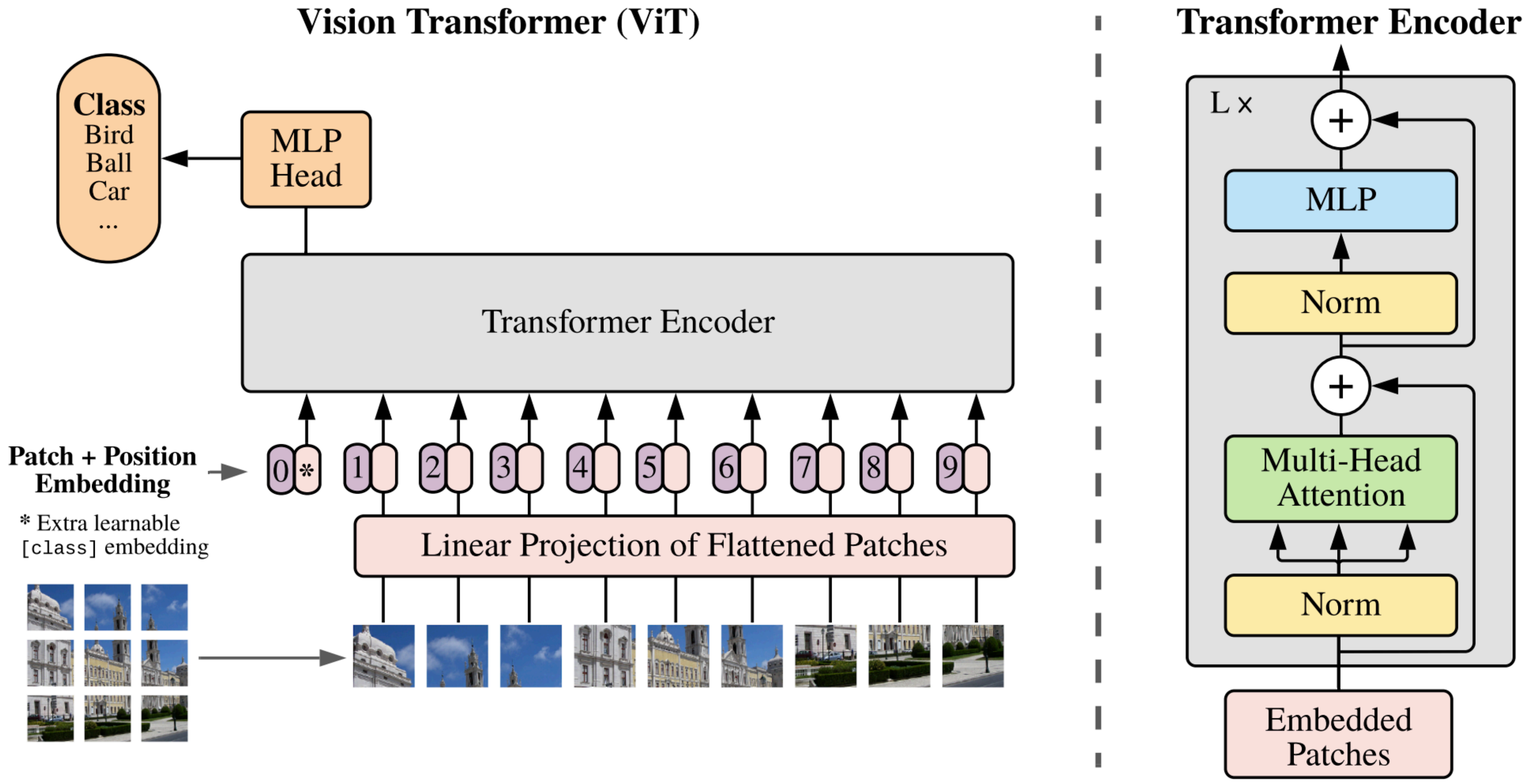
References/Image credits

Kaplan et al., "Scaling Laws for Neural Language Models", arxiv (2020)

J. Hoffmann et al., "Training Compute-Optimal Large Language Models", arXiv (2022)

Vision Transformer

Vision Transformer (ViT) Architecture

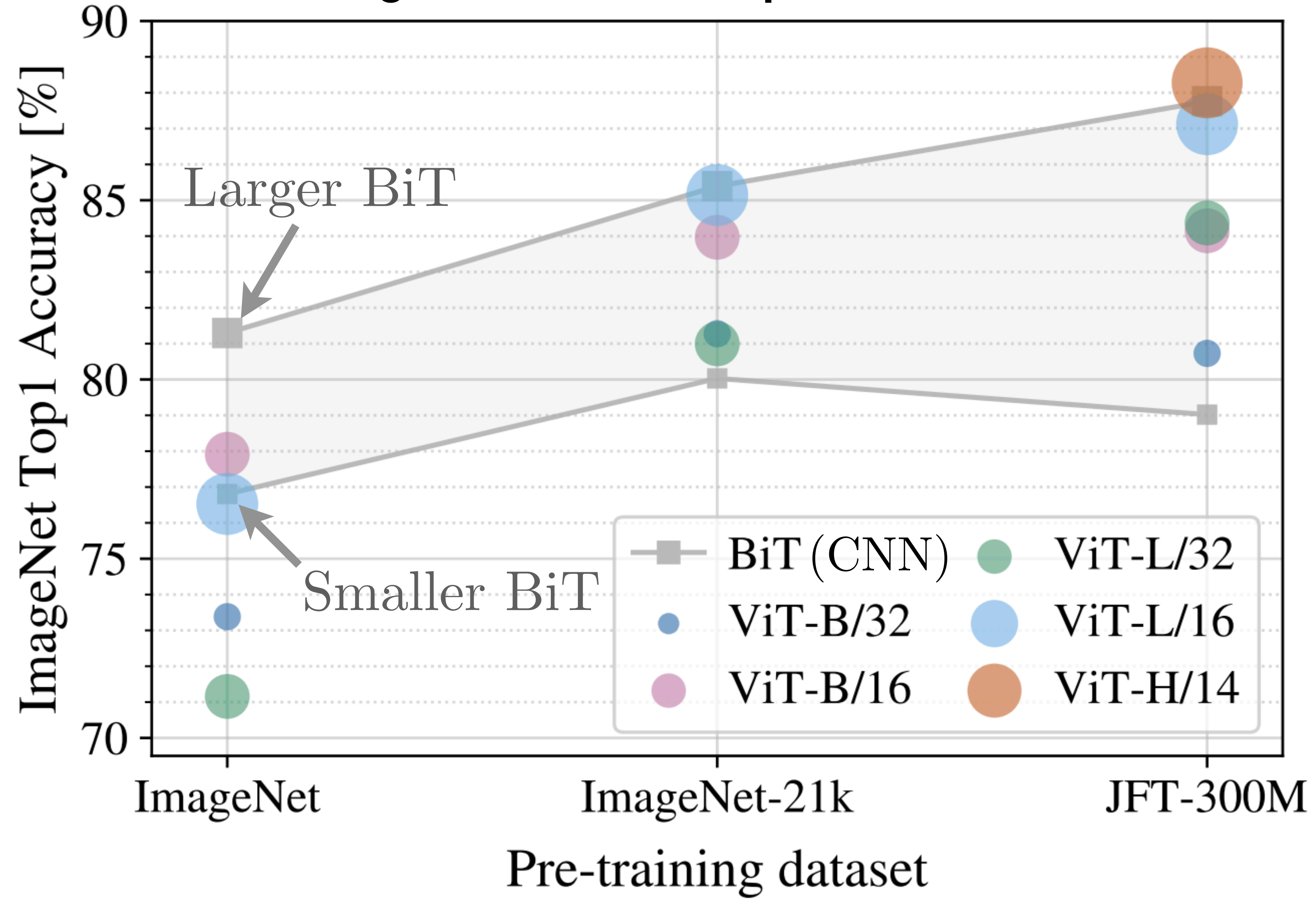


Re-purposes the Transformer (encoder) for vision by:

- Splitting images into patches, projecting to embeddings
- Inserting an extra [CLASS] token
- Adding on position embeddings

The importance of pre-training scale

ImageNet transfer performance



1.3M images | 14M images | 303M images

In lower-data regime, the *stronger inductive biases* (locality, translation invariance) of the CNN works better. But in the higher-data regime (e.g. JFT-300M), ViT shines.

References/Image credits

Dosovitskiy et al., “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR (2021)

Transformer Explosion

Historical context: non-local means

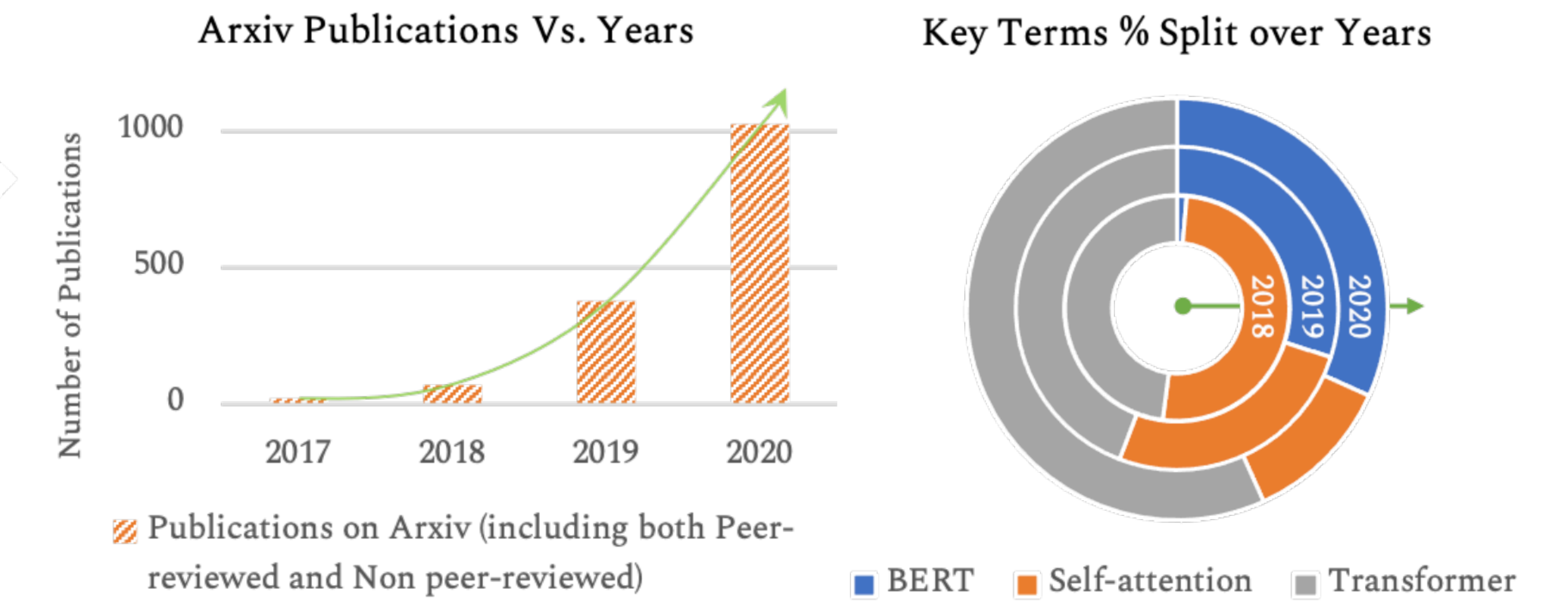
The "self-attention" operation has long been used in the image processing community for de-noising, under the name "non-local means":

$$NL[v](i) = \sum_{j \in I} w(i,j)v(j)$$

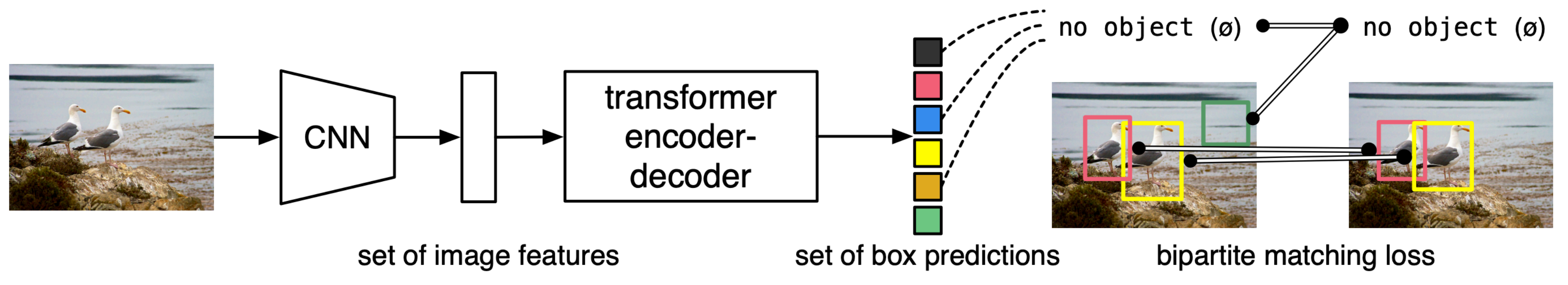
Here $v = \{v(i) \mid i \in I\}$ is a noisy image, and the weights $\{w(i,j)\}_j$ depend on the similarity between pixels i and j.

However, the broad applicability and value of this (highly flexible) operation has become clearer in recent years.

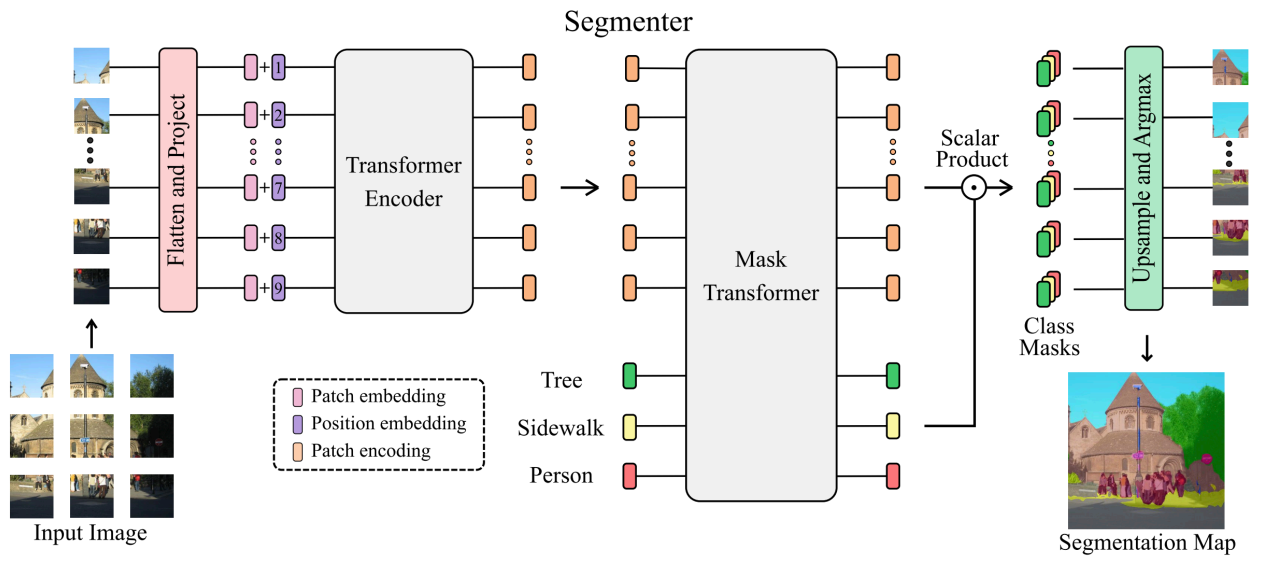
Research Interest in the Vision Community



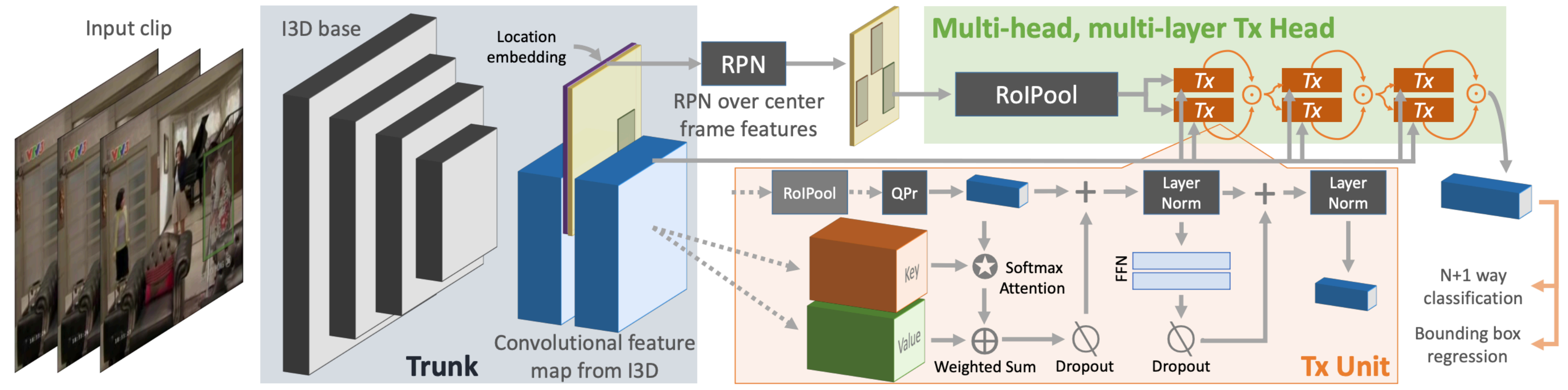
Object Detection: DETR



Semantic Segmentation: Segmenter

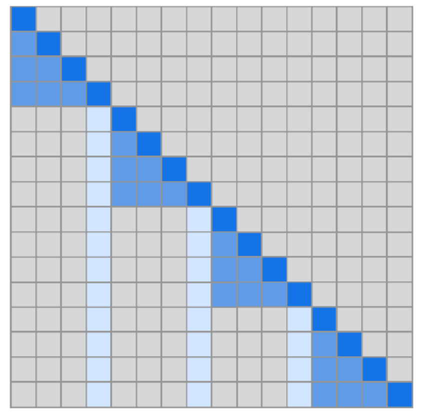


Action Recognition: Video Action Transformer Network

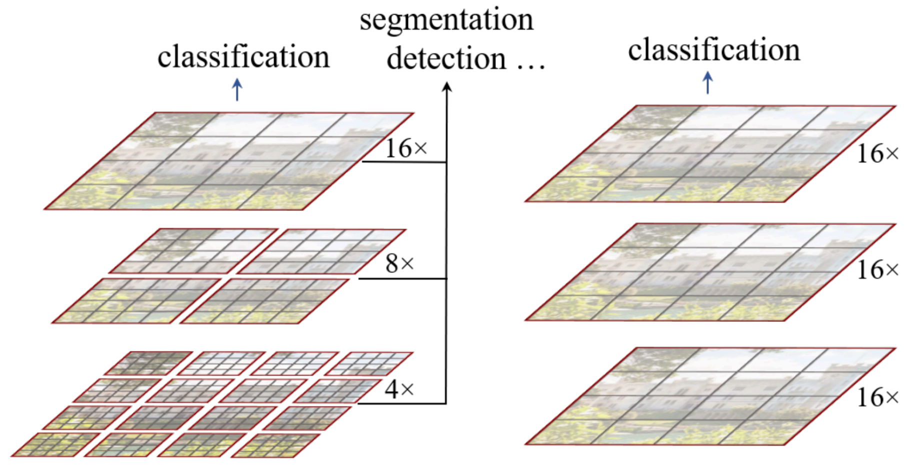


Computational tricks

Problem: self-attention has quadratic complexity in the input size (every element attends to every other element).
Many solutions have been proposed, including:



The **Sparse Transformer** factors attention to reduce complexity to $\mathcal{O}(n\sqrt{n})$



The **Swin Transformer** achieves linear complexity by restricting self-attention to fixed regions (like a CNN....).

References/Image credits

A. Buades et al., "A non-local algorithm for image denoising", CVPR (2005)
X. Wang et al., "Non-local Neural Networks", CVPR (2018)
S. Khan et al., "Transformers in Vision: A Survey", arxiv (2021)

Carion et al., "End-to-End Object Detection with Transformers", ECCV (2020)
Strudel et al., "Segmenter: Transformer for Semantic Segmentation", ICCV (2021)
Girdhar et al., "Video Action Transformer Network", CVPR (2019)

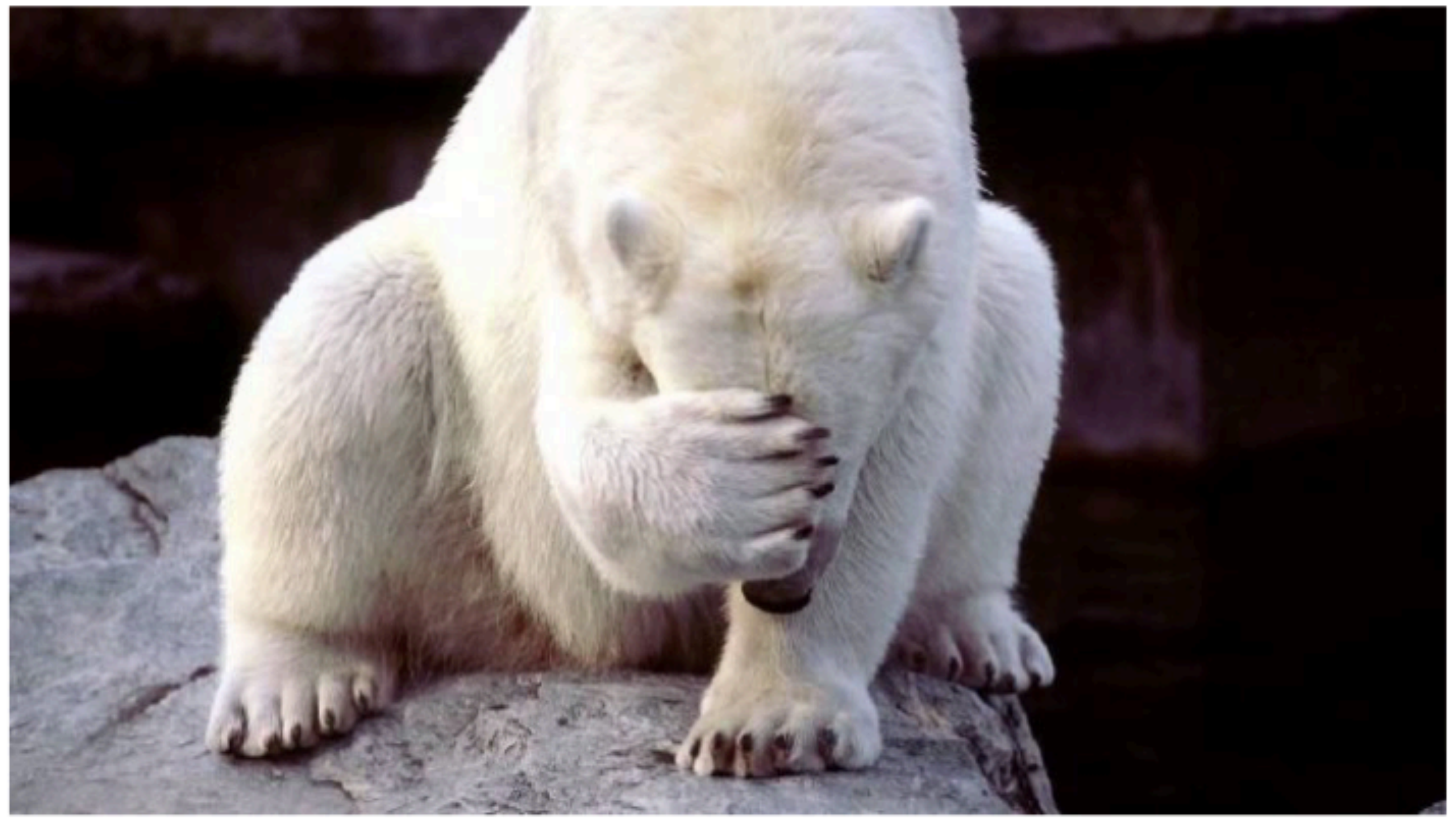
R. Child et al., "Generating Long Sequences with Sparse Transformers", arxiv (2019)
Liu et al., "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", ICCV (2021)

Neural Network Design and Energy Consumption

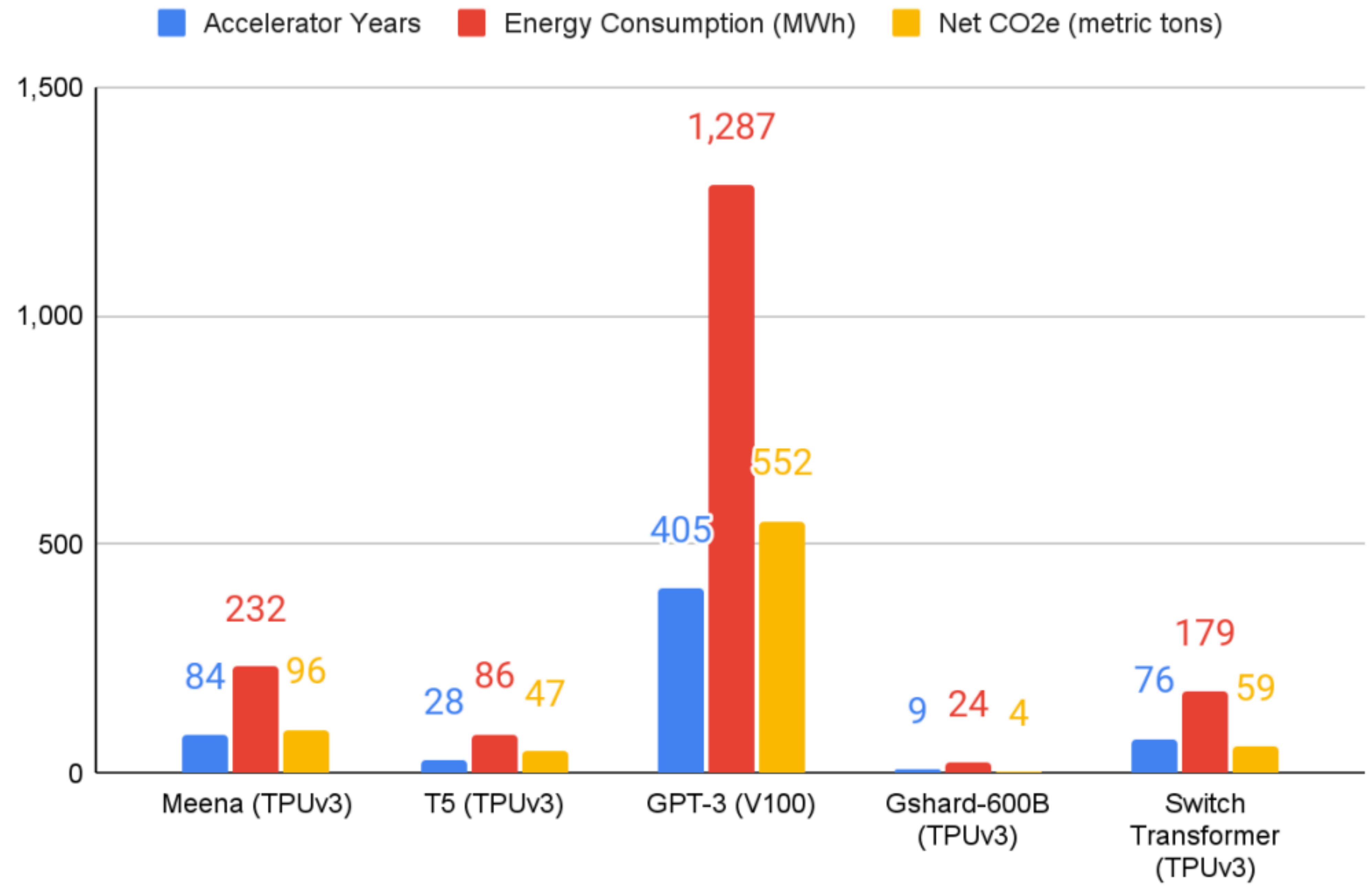
Deep Neural Networks are Energy Intensive

Consumption	CO_2e (lbs)
Air travel, 1 passenger, NY↔SF	1984
Human life, avg, 1 year	11,023
American life, avg, 1 year	36,156
Car, avg incl. fuel, 1 lifetime	126,000

Training one model (GPU)	
NLP pipeline (parsing, SRL)	39
w/ tuning & experimentation	78,468
Transformer (big)	192
w/ neural architecture search	626,155



Transformers represent many of the biggest models



Reasons for optimism:

- There are significant opportunities for grid efficiency: training is not time-sensitive (can be scheduled to maximise peak renewable energy times)
- Fusion is only 30 years away....

References/Image credits
E. Strubell et al., "Energy and Policy Considerations for Deep Learning in NLP", arxiv (2019)
Image credit: https://www.desktopbackground.org/wallpaper/white-bear-put-hand-on-head-wild-animal-wallpaper-jpg-492933
D. Patterson et al., "Carbon Emissions and Large Neural Network Training", *arxiv* (2021)