

Towards Automatic Architecture Design for Emerging Machine Learning Tasks



WISCONSIN
UNIVERSITY OF WISCONSIN-MADISON

Misha Khodak
Carnegie Mellon University



Hewlett Packard
Enterprise



DDPS WEBINAR LLNL
4 NOVEMBER 2021

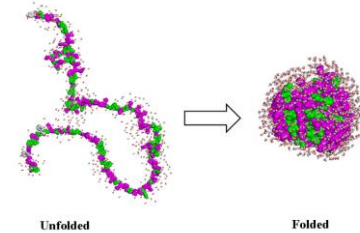
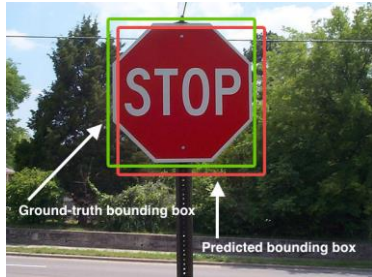


The past decade in machine learning

U Toronto

DeepMind (Google)

DeepMind (Google)

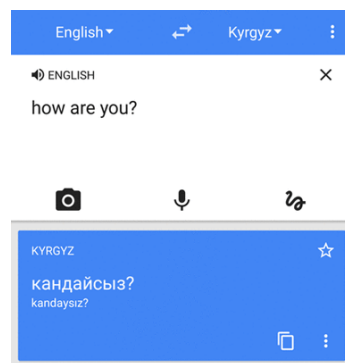


2014

2016

2020

2012



Google



OpenAI / Microsoft

Democratizing deep learning

Deep learning achieves impressive results in some domains:

- Vision (image classification, segmentation)
- Text (language prediction, translation)

Applying it in other domains is trickier because the types of neural networks being used are not well-tuned for them:

- Experimental data from the natural sciences
- Simulation data from science and engineering
- Graph-based
- Biological se



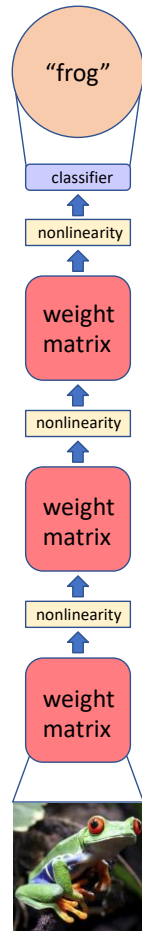
Transportation: Autonomous vehicles
Internet: Image Search, Voice-powered interfaces
Telecom: Voice assistants

Science: Data-driven PDE solvers
Healthcare: Drug discovery, Cancer Detection
Manufacturing: Anomaly detection
Finance: Fraud detection, quantitative trading
Retail: No-checkout shopping
AdTech: Improved bidding, recommendation
Education: Personalized lesson plans
Software: Automated customer service
Agriculture: Aerial imagery analysis
Construction: Site monitoring and optimization
IoT: Smart buildings, auto-cooled data centers
....
Waste Management: Detection and logistics

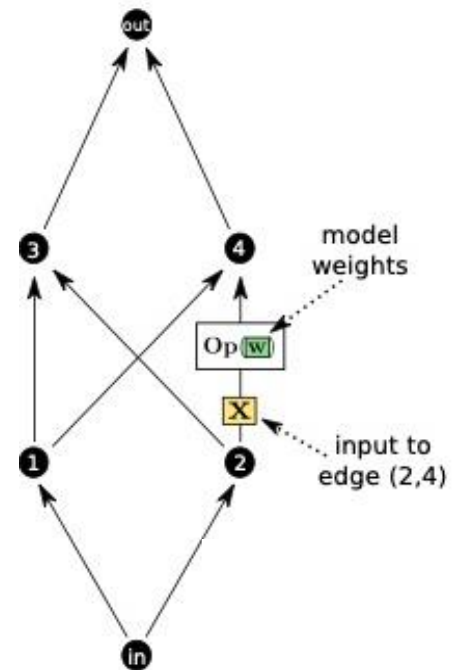
Can we automate this tuning, enabling the straightforward application of deep learning to many tasks?

What is a neural architecture?

Standard
neural
network



General
computational
graph



Developments in human architecture design

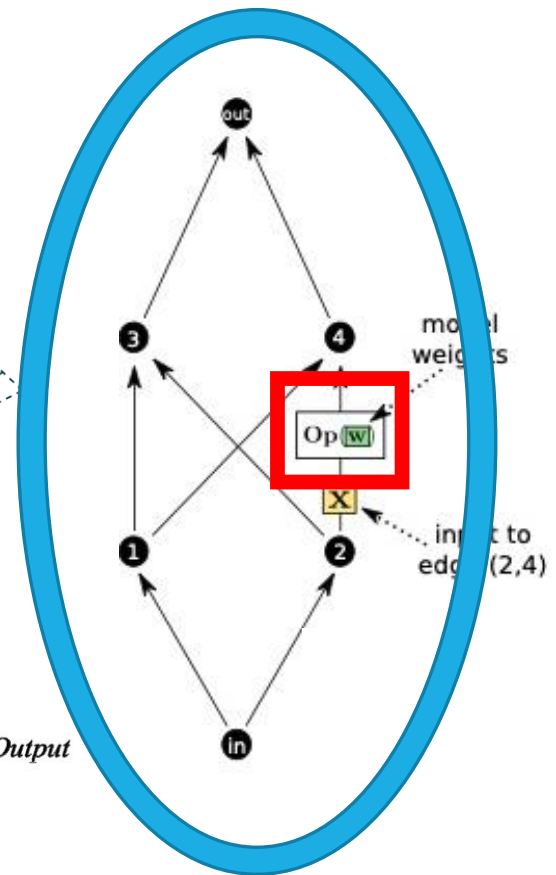
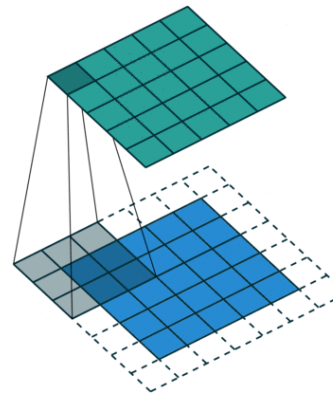
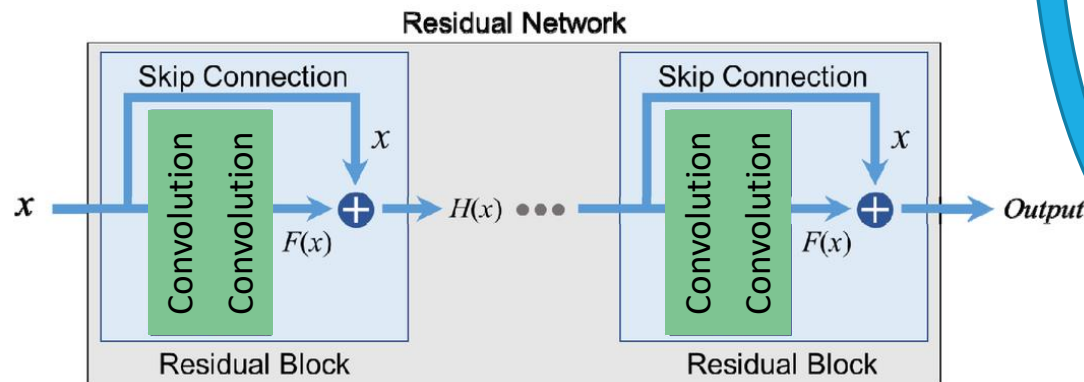
Human-driven architecture search has proceeded in two directions:

Operations:

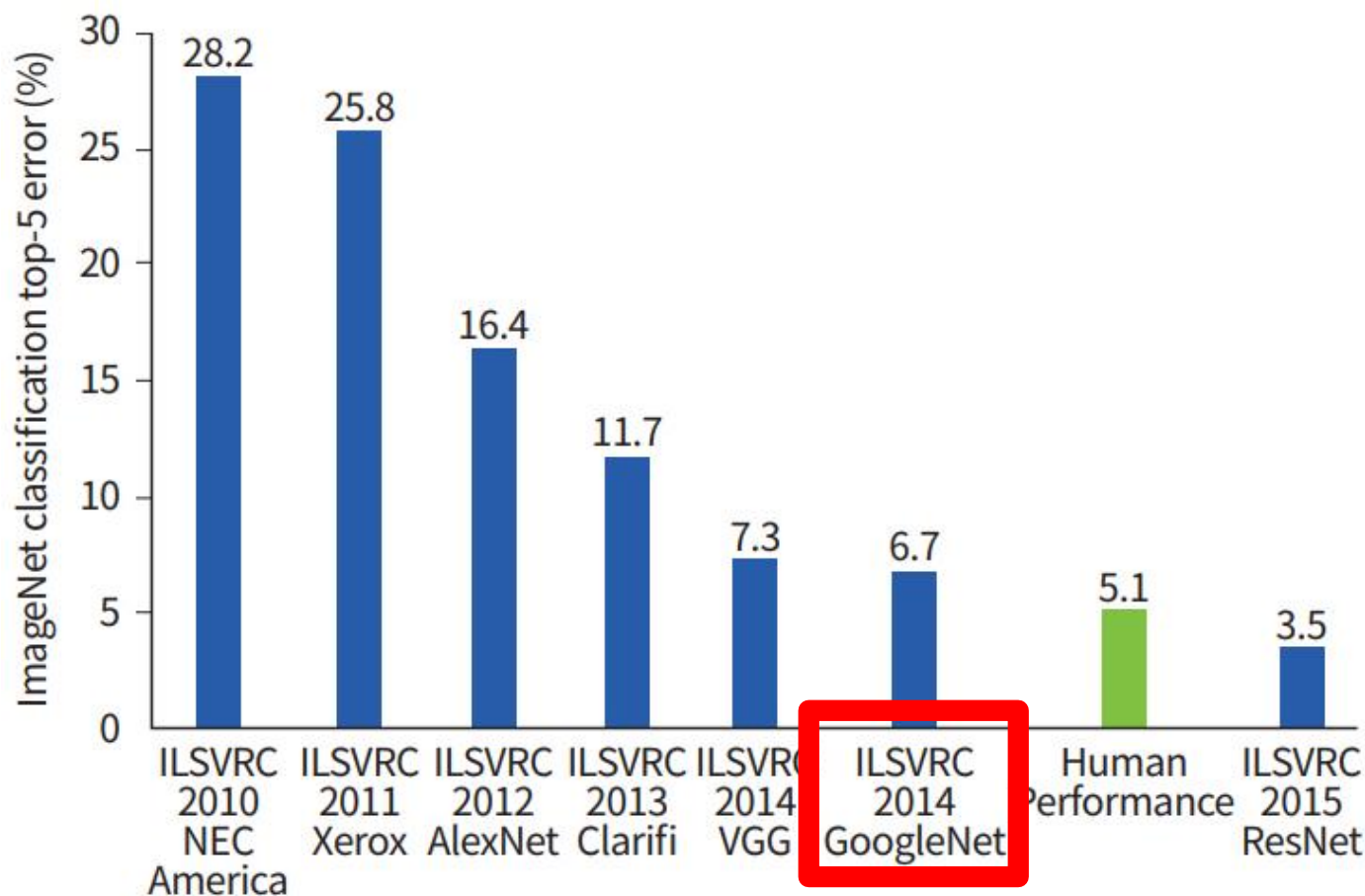
- Linear
- Convolutions
- Transformers

Network topology:

- Skip connections (ResNet)



New architectures drive progress in machine learning

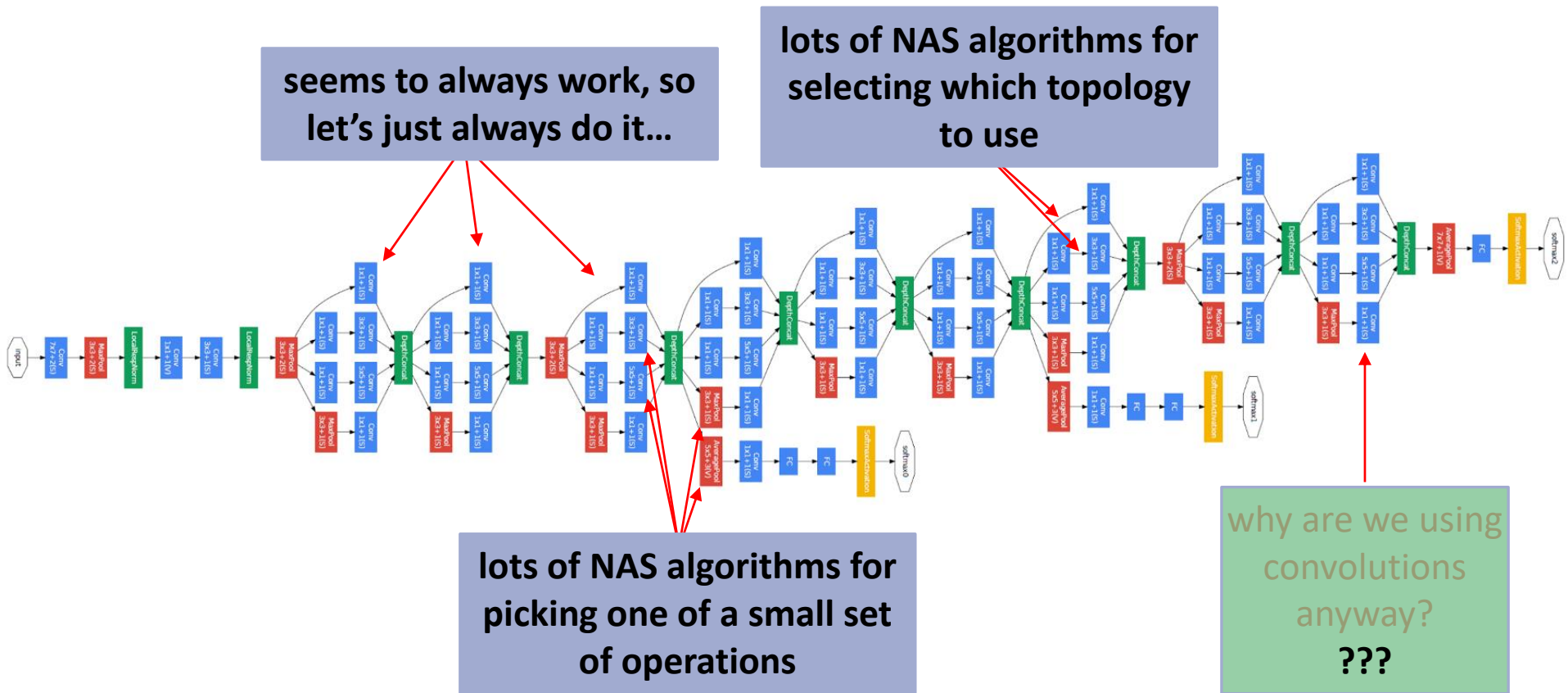


Neural architecture search (NAS)

Goal: automate away the design of neural architectures

- help practitioners in advanced areas of ML (vision, language, ...)
- accelerate progress in budding application areas (natural sciences, social sciences, ...)

Some might say we've made a lot of progress in NAS



...but much of this progress has been limited in scope

Goal: automate away the design of neural architectures

Focus of existing NAS research:

- help practitioners in advanced areas of ML (vision, language, ...)
- **NAS methods glue together existing primitives (e.g. convolutions) that we know work well on heavily studied tasks (vision).**
- **Why do this if Google is already doing human architecture search for this?**

Goal of our work:

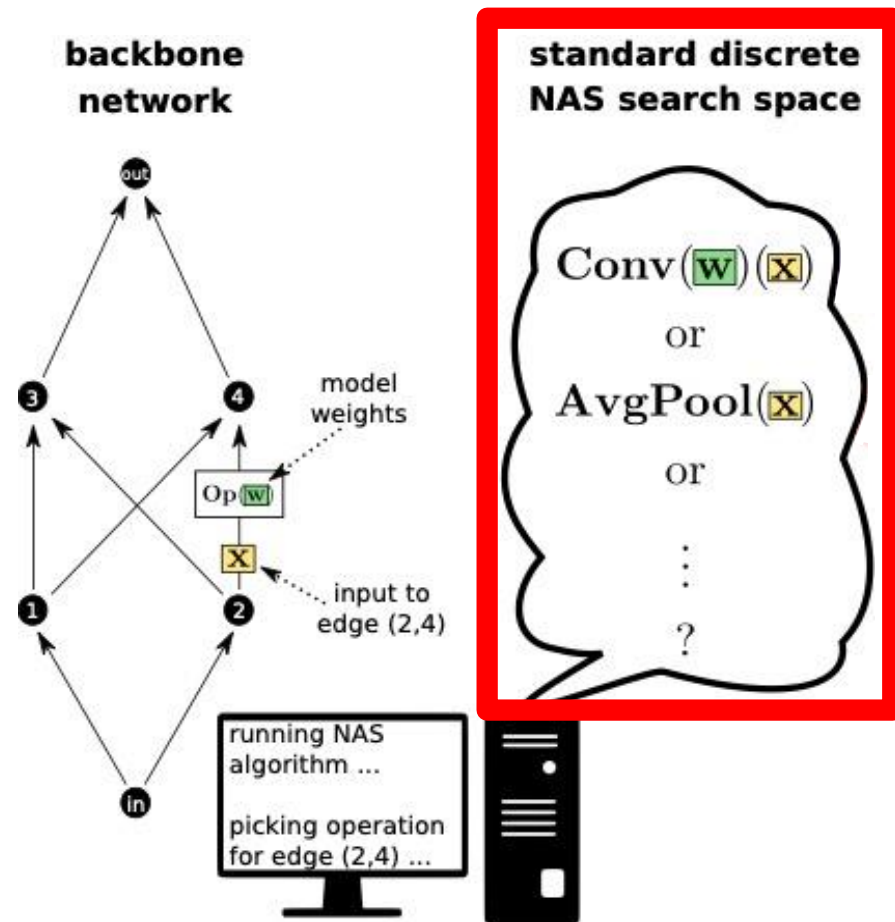
- accelerate progress in budding application areas (natural sciences, social sciences, computational sciences ...)

Specifically:

a new search space that can serve as an initial, automated solution to any ML problem on diverse domains

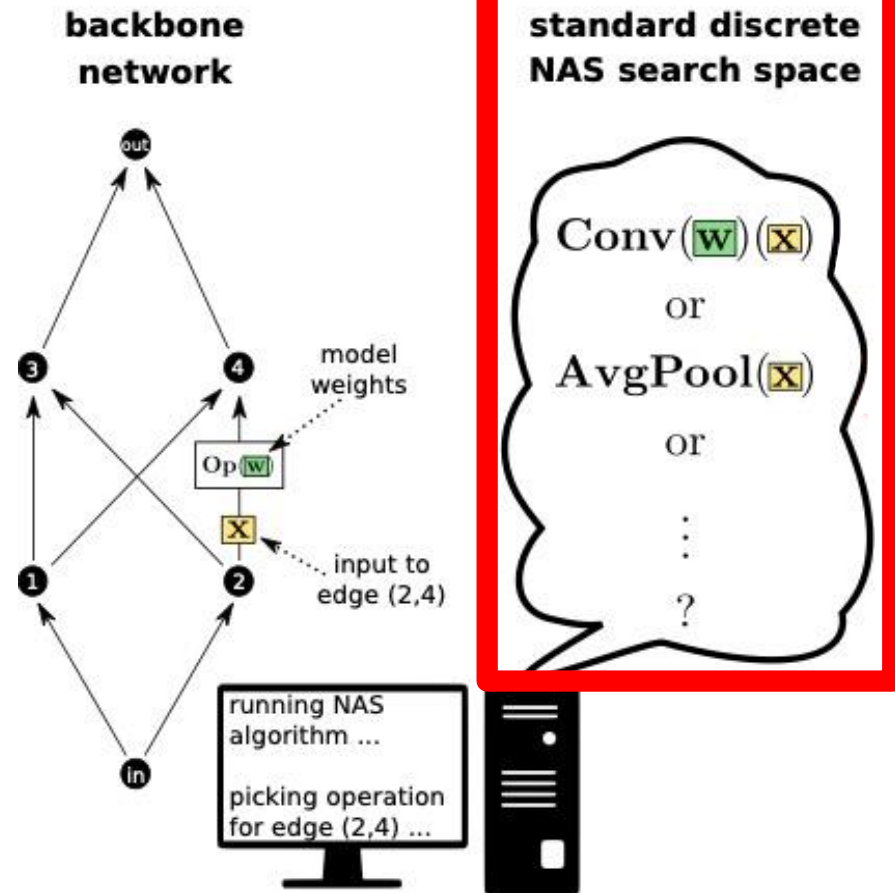
What is the NAS problem?

- Pick an operation for each edge in a computational graph to maximize some objective (accuracy, latency, ...).
- To do so we define a set of operations, i.e. a *search space*.
- Popular DARTS search space [Liu et al., '18] has 8 operations:
 - Identity
 - Zero-operation
 - Conv 3x3 and 5x5
 - Dilated Conv 3x3 and 5x5
 - Avg Pool
 - Max Pool



What is the NAS problem?

- DARTS operations:
 - Identity
 - Zero-operation
 - Conv 3x3 and 5x5
 - Dilated Conv 3x3 and 5x5
 - Avg Pool
 - Max Pool
- We know convolutions work well for visions tasks.
- Would this search space work for other data domains?



What other domains?

Many problems involve data consisting of multi-dimensional arrays:

- Sequence modeling
 - Convolutions work okay but need large dilations [Bai et al., '18], and Transformers are still better [Vaswani et al., '17].

used ResNet

- Solving PDEs
 - Convolutions work poorly, but are related to the state-of-the-art operation [Li et al., '21].

topology

basic multi-

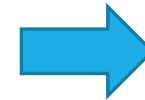
layer topology

- Prediction on graphs
 - Regular convolutions don't make sense; graph convolutions [Kipf & Welling, '17] work well.

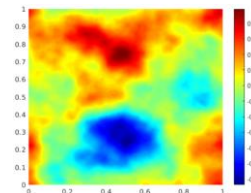
<start token>

<Neural>

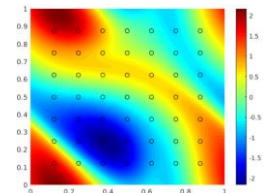
<Architecture>



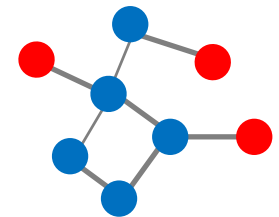
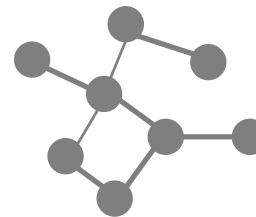
<Search>



time=0



time=50



How can we handle these new domains?

Possible solutions:

- Just use this search space anyway.
 - Large gap in performance.
 - Cannot discover truly novel architectures.
- Add new operations one-by-one:
 - Discrete NAS methods scale poorly with number of operations.
 - What if we don't know the right operation to use?

We need a more general search space focused on operations, NOT topology.

Constructing a more general search space

DARTS operations:

- Identity
- Zero-operation
- Conv 3x3 and 5x5
- Dilated Conv 3x3 and 5x5
- Avg Pool
- Max Pool

$$\begin{aligned} \text{Conv}(\mathbf{w})(\mathbf{x}) &= \mathbf{A}_w \mathbf{x} \\ &= \mathbf{F}^{-1} \text{diag}(\mathbf{F}\mathbf{w}) \mathbf{F}\mathbf{x} \end{aligned}$$

Diagram illustrating the convolution operation and its representation using the Discrete Fourier Transform (DFT). The top equation shows the convolution of filter weights \mathbf{w} (labeled "filter weights") with input \mathbf{x} (labeled "input") resulting in $\mathbf{A}_w \mathbf{x}$. The bottom equation shows the equivalent operation using the DFT: $\mathbf{F}^{-1} \text{diag}(\mathbf{F}\mathbf{w}) \mathbf{F}\mathbf{x}$. The DFT operation is indicated by red arrows pointing from the word "DFT" to the \mathbf{F} and \mathbf{F}^{-1} terms in the second equation.

7/8 of the DARTS operations are

- linear
- diagonalized by the discrete Fourier transform (DFT)

Key idea: replace the DFTs by a more general family of efficient matrices

Expressive diagonalization (XD) operations

Key idea: replace the DFTs by a more general family of efficient matrices

$$\text{Conv}(\underline{\mathbf{w}})(\underline{\mathbf{x}}) = \mathbf{A}_{\underline{\mathbf{w}}} \underline{\mathbf{x}}$$

filter weights input

$$= \mathbf{F}^{-1} \text{diag}(\mathbf{F}\underline{\mathbf{w}}) \mathbf{F}\underline{\mathbf{x}}$$

DFT

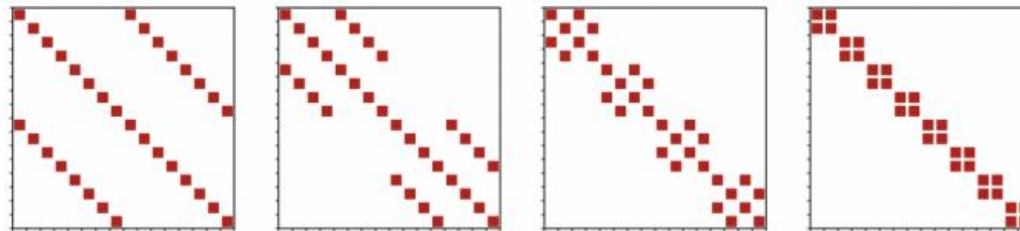
$$\text{XD}_{\alpha}^1(\underline{\mathbf{w}})(\underline{\mathbf{x}}) = \text{Real}(\mathbf{K} \text{diag}(\mathbf{L}\underline{\mathbf{w}}) \mathbf{M}\underline{\mathbf{x}})$$

architecture parameters $\alpha = (\mathbf{K}, \mathbf{L}, \mathbf{M})$

Which family of efficient matrices should we use?

Kaleidoscope (K-) matrices [Dao et al., '20]

- FFT-like product of small factor matrices



- Provably expresses any efficient matrix-vector operation:
 - sparse
 - low-rank
 - permutation
 - DFT
 - DCT
 - wavelet transform

Substituting K-matrices into the diagonalization

Allowing each DFT to instead be any K-matrix

- preserves the efficiency/short description length of convolutions
- allows us to express many different operations of interest

$$\begin{aligned} \text{Conv}(\underline{\mathbf{w}})(\underline{\mathbf{x}}) &= \mathbf{A}_{\underline{\mathbf{w}}} \underline{\mathbf{x}} \\ &= \mathbf{F}^{-1} \text{diag}(\mathbf{F}\underline{\mathbf{w}}) \mathbf{F}\underline{\mathbf{x}} \\ \mathbf{XD}_{\alpha}^1(\underline{\mathbf{w}})(\underline{\mathbf{x}}) &= \text{Real}(\mathbf{K} \text{diag}(\mathbf{L}\underline{\mathbf{w}}) \mathbf{M}\underline{\mathbf{x}}) \end{aligned}$$

architecture parameters $\alpha = (\mathbf{K}, \mathbf{L}, \mathbf{M})$

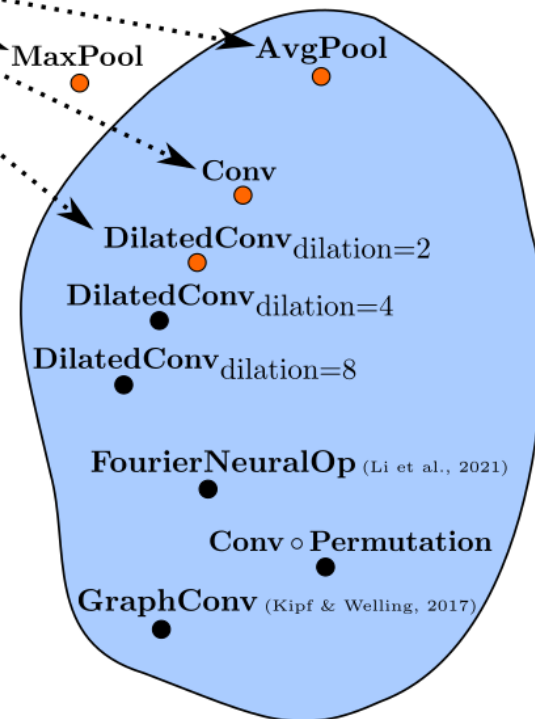
What does the set of XD-operations contain?

standard discrete
NAS search space

search space of
XD-operations

XD-operations include

- 7/8 DARTS operations
- all convolutions in PyTorch
- graph convolutions (for fixed graphs)
- the Fourier neural operator (SOTA for PDE solvers)



How to use XD-operations

Standard pipeline:

- Obtain data
- Choose a conv net topology (LeNet, ResNet, VGG, ...)
- Train model weights using SGD or Adam
- Evaluate and tune as needed

To use XD-operations:

- replace all convolutions in the network by XD-operations initialized (“warm-started”) as convolutions
- simultaneously train architecture parameters using SGD or Adam
- only extra tuning is of the architecture optimizer settings

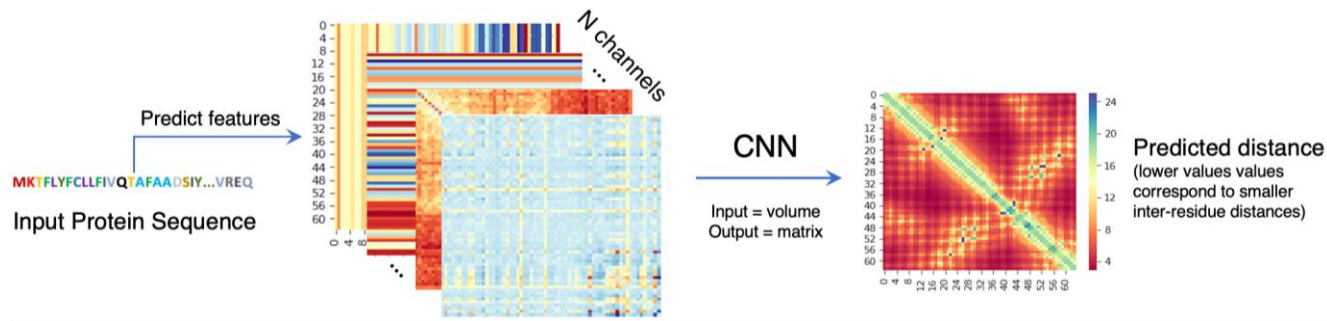
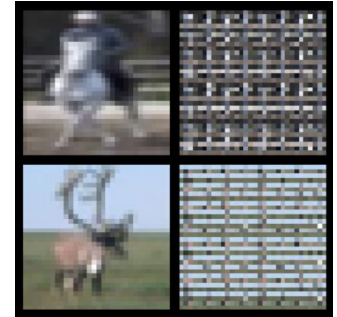
Applications

Classifying permuted images:

- XD outperforms baseline CNN and expensive NAS methods when images in a standard vision benchmark are permuted.

Predicting protein folding / next note prediction in sheet music:

- XD beats CNNs with custom-designed dilation pattern



Neural PDE solving

Neural PDE Solvers

Setup:

- Sample problems from some distribution over initial conditions
- For each problem, generate a “ground-truth” solution at some later time using a standard solver, e.g. Crank-Nicholson
- Train a neural network to learn a function from initial conditions to the later state

Target applications:

- Direct use
- Inverse problems
- Transfer to new problem domains



Operations for training neural PDE solvers

- Convolutions
 - Baseline
 - Performance decreases as resolution increases
- Fourier Neural Operator (FNO) [Li et al., '21]
 - Much stronger performance
 - Three orders of magnitude faster than traditional solvers
 - Consistent across resolutions
- XD-operations:
 - Contain both convolutions and FNOs!
 - Slower, but still fast enough to be useful

Fourier neural operator

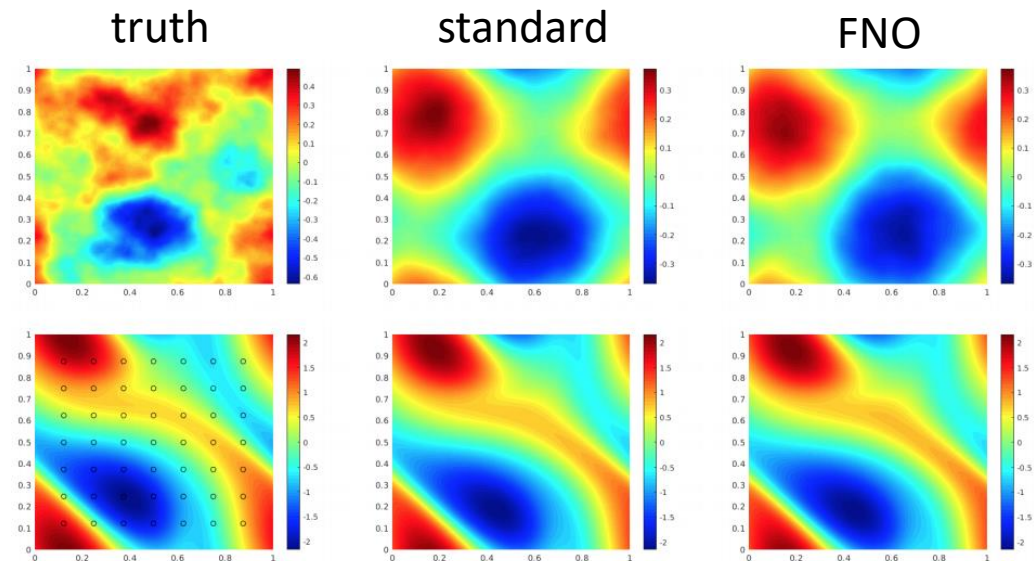
$$\text{FNO} \\ \text{Conv}(\mathbf{w})(\mathbf{x}) = \mathbf{F}^{-1} \text{diag}(\mathbf{X}\mathbf{w})\mathbf{F}\mathbf{x}$$

↓ ↓ ↓

$$\mathbf{X}\mathbf{D}_{\alpha}^1(\mathbf{w})(\mathbf{x}) = \text{Real}(\mathbf{K} \text{diag}(\mathbf{L}\mathbf{w})\mathbf{M}\mathbf{x})$$

On an inverse problem requiring 30K solves to determine the initial vorticity of a Navier-Stokes system:

- FNO network takes 2.5 minutes (+ 12 hours data-generation & training)
- Standard solver takes 18 hours



Experimental setup

Settings

- Burgers' equation on $[0,1]$ with random initial conditions.
- Darcy flow on $[0,1]^2$ with random initial diffusion.
- 2d Navier-Stokes on $[0,1]^2 \times [0, T]$ with random initial vorticity

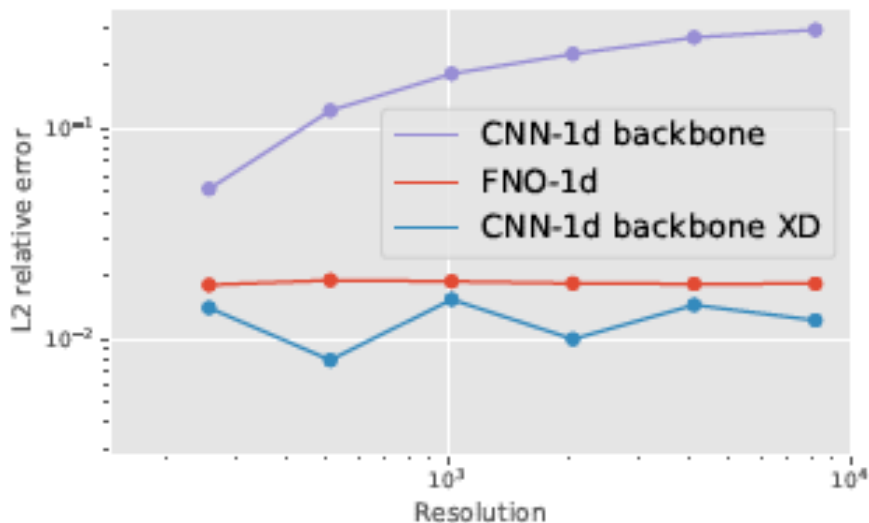
Discretization

- Square grid, including in time
- Performance measured as average of squared losses over grid vertices

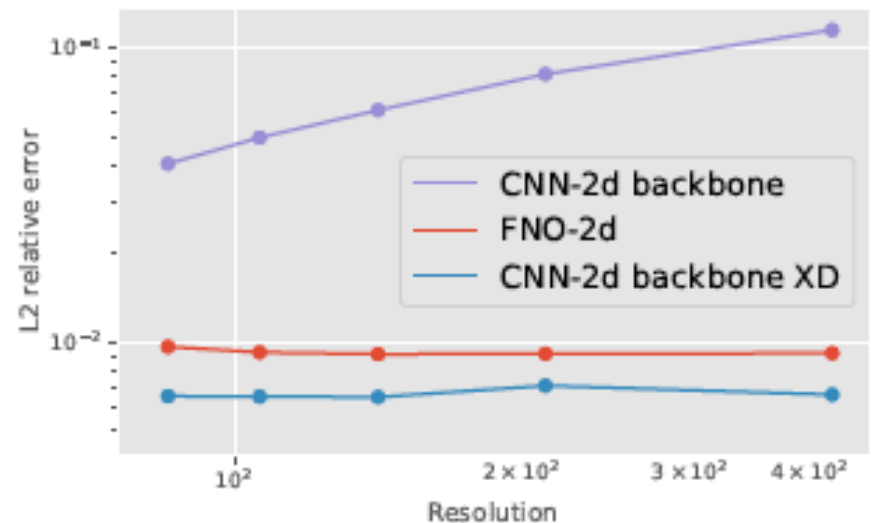
All backbone networks and settings borrowed from FNO paper where possible.

XD-operations across resolutions (Burgers' and Darcy flow)

Burgers' equation (1d)

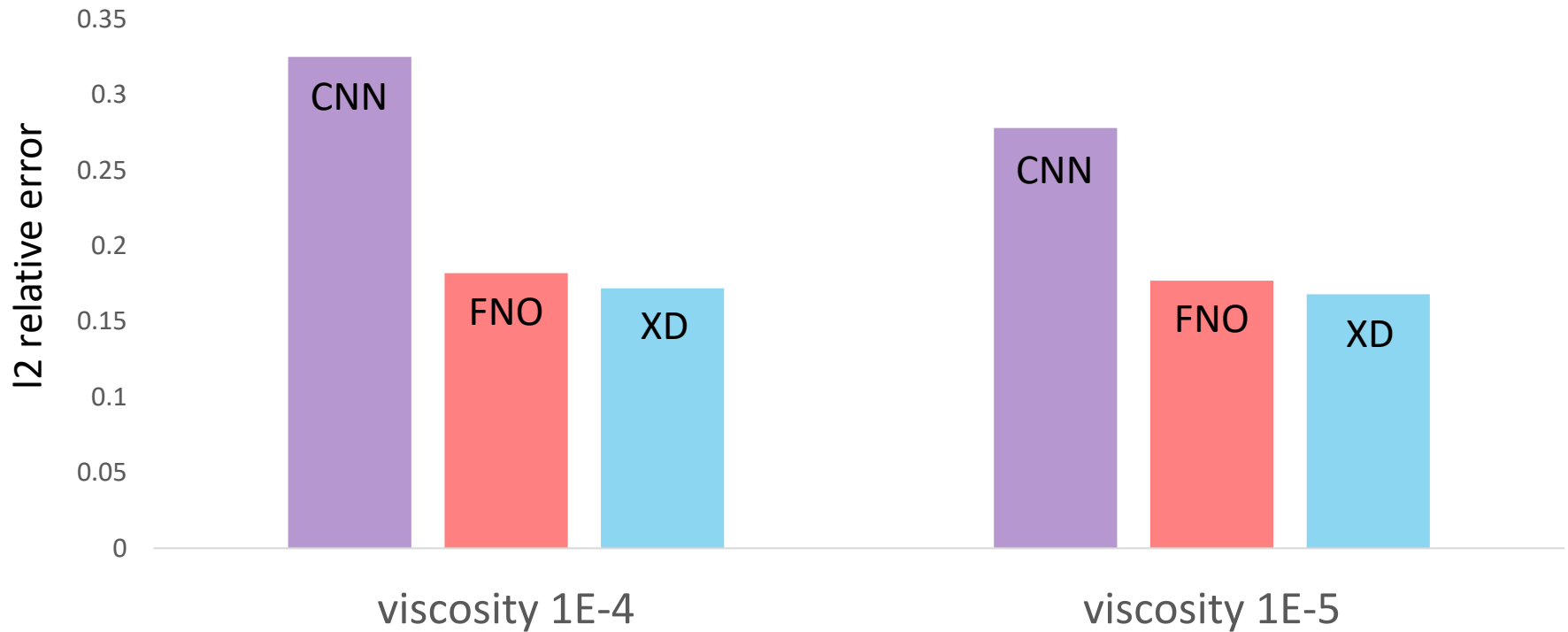


Darcy flow (2d)



- Vanilla CNN performance decreases with increasing grid resolution
- Starting from CNNs, XD gets much smaller error and consistency across resolutions
- It even slightly outperforms (the custom-designed) FNO

XD-operations across viscosities (Navier-Stokes)



Recap: why use XD?

Transform any CNN backbone (ResNet, VGG, etc.) into a search space over operations and find something better than convolution for your problem.

Successful applications:

- Permuted/flattened image classification
- Neural PDE solving
- Distance prediction for protein folding
- Music modeling
- Language modeling
- Financial time series prediction

Unsuccessful (little better than baseline):

- Image classification
- Audio classification

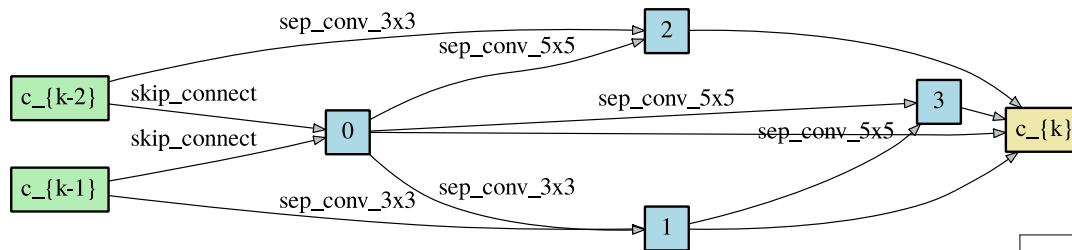


What XD is not

A replacement for classical topology search NAS (DARTS, DeepHyper, etc.)

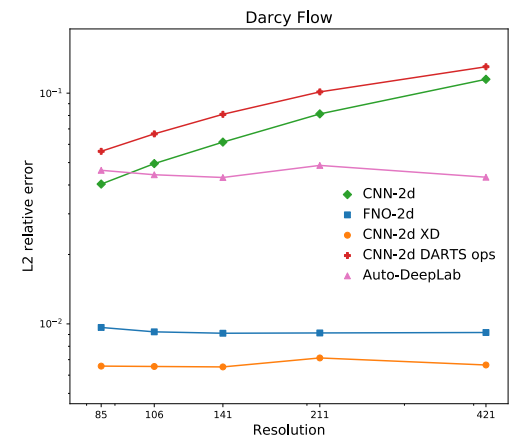


- XD is solving a different problem: discovering *new* operations, **not** connecting existing ones
- Can be used in-conjunction (first find a good CNN, then apply XD)



A direct way to improve efficiency

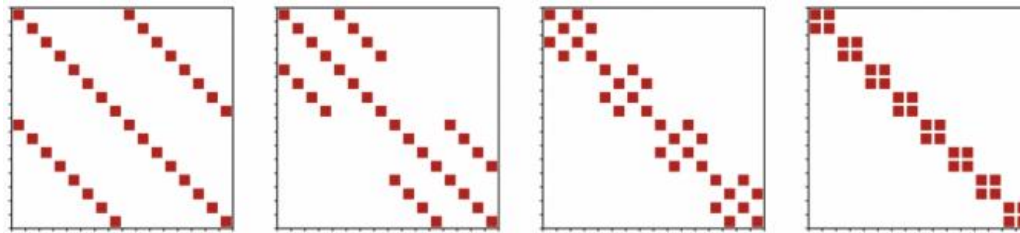
- Applying XD to a CNN makes it:
 - 5x slower to train on the PDE task
 - 2x slower on the protein task
 - Similar slow-downs for inference
- Large memory costs



Ongoing and future work: Efficiency

Causes of inefficiency in XD-operations:

- Dependence on new software for K-matrices instead of the FFT
- Cannot take advantage of Hermitian symmetry of DFT
- Kernel filters must be padded to the input size



Ongoing efforts:

- Better software for K-matrices
- Developing alternative structured matrices that are more GPU-friendly
- Approximations/truncations











Ongoing and future work: Benchmarking

Evaluating NAS on diverse tasks is difficult:

- What is a diverse task? How many do you need?
- How to assess different methods?
- How to consider computational tradeoffs?

Ongoing efforts: **NAS-Bench-360** (nb360.ml.cmu.edu)

- 10 (and counting) understudied tasks that are feasible to evaluate on an academic budget
- Includes tasks from PDE solving, protein folding, genomics, audio, ...
- Evaluations of NAS methods at different budget constraints

NAS-Bench-360		
A NAS Benchmark for Diverse Tasks		
CIFAR-100	Computer Vision	
Spherical	Omnidirectional Vision	
NinaPro DB5	Prosthetics Control	
FSD50K	Audio Classification	
Darcy Flow	PDE Solver	
PSICOV	Protein Folding	
Cosmic	Astronomy Imaging	
ECG	Medical Diagnostics	
Satellite	Earth Monitoring	
DeepSEA	Genetic Prediction	

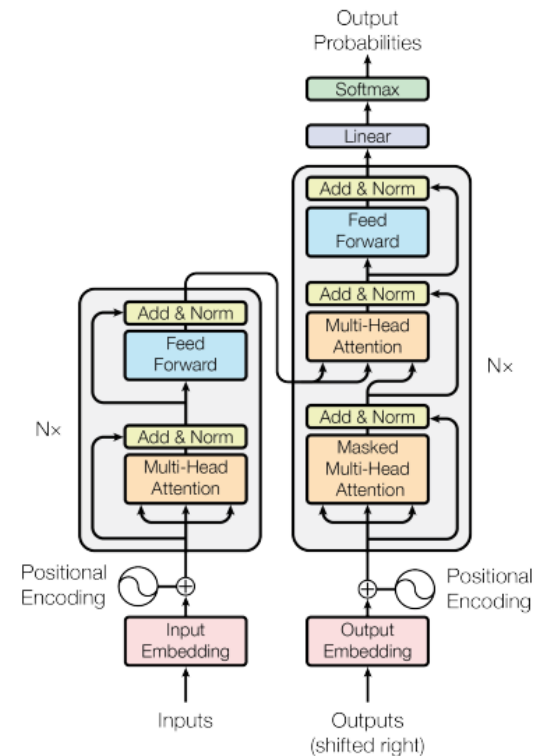
Ongoing and future work: New search spaces

Can we design search spaces that efficiently contain other important operations?

- Self-attention
- Max-pooling

Can similar ideas be applied to search over efficient operation spaces?

- Different types of convolutions
- Approximations to self-attention



Thank you!

Collaborators:

Nick Roberts, Tri Dao, Liam Li, Chris Ré, Ameet Talwalkar

Paper (to appear at NeurIPS 2021):

<https://arxiv.org/abs/2103.15798>

Software:

<https://github.com/mkhodak/relax>

Contact:

khodak@cmu.edu