CS 589 Fall 2020

Frontier topic 1

Instructor: Susan Liu
TA: Huihui Liu

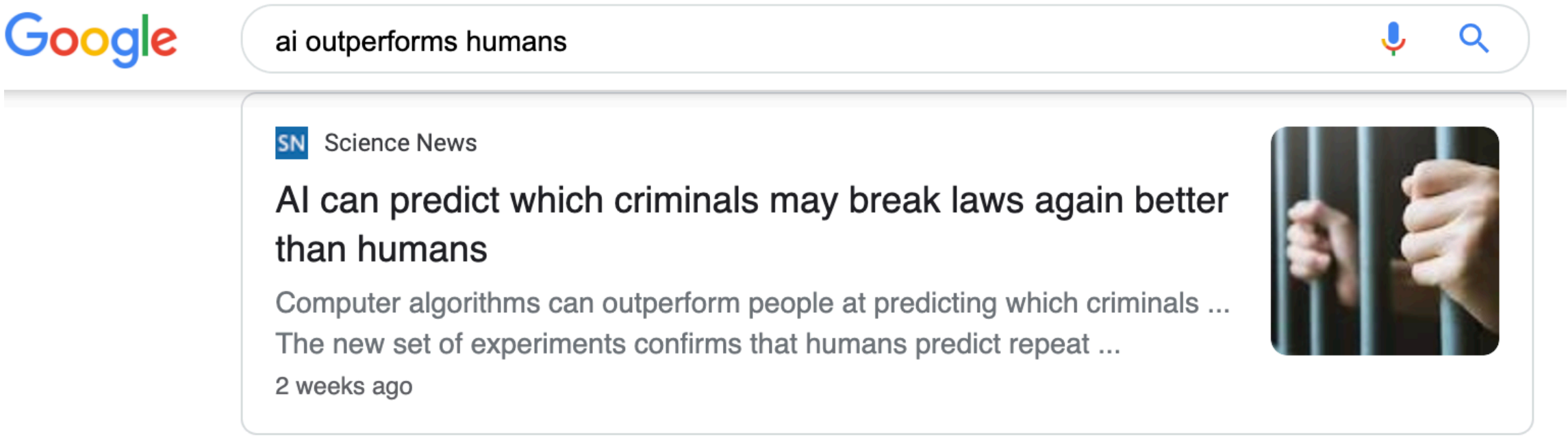
Stevens Institute of Technology

Most slides come from Stanford CS276

Frontier topics

- Neural program synthesis
- Pretrained LMs
- AutoML; hyperparameter optimization
- Fairness/bias in AI
- Machine learning explainability for text mining/NLP

Artificial Intelligence Revolution

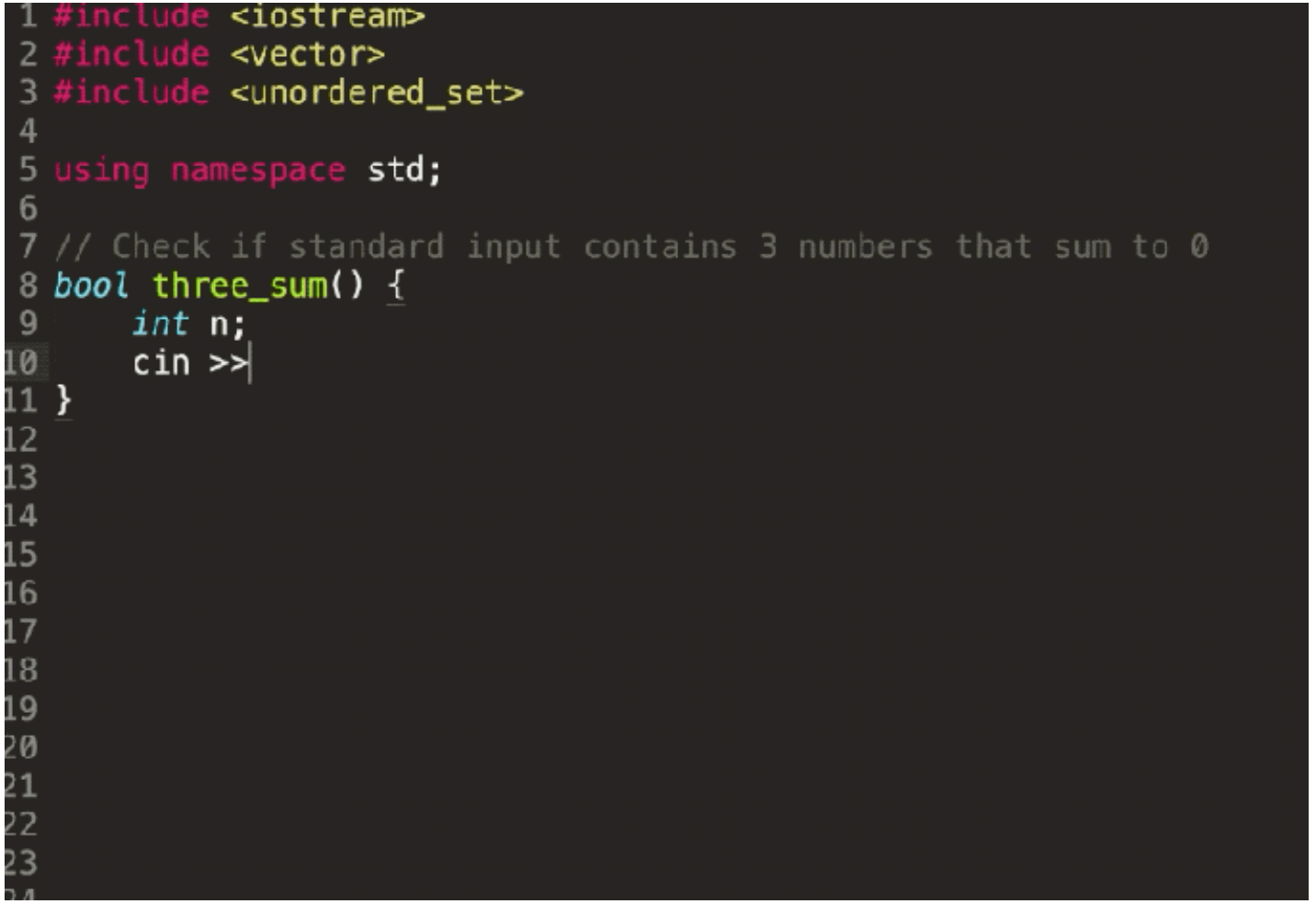


AI Outperforms Human

ebeat.com/
-areas-where-artificial-
rks-outperform-humans/

Image Style Transfer

Can AI Beat Human in Writing Code? (Deep TabNine)



```
#include <iostream>
#include <vector>
#include <unordered_set>

using namespace std;

// Check if standard input contains 3 numbers that sum to 0
bool three_sum() {
    int n;
    cin >>
}
```

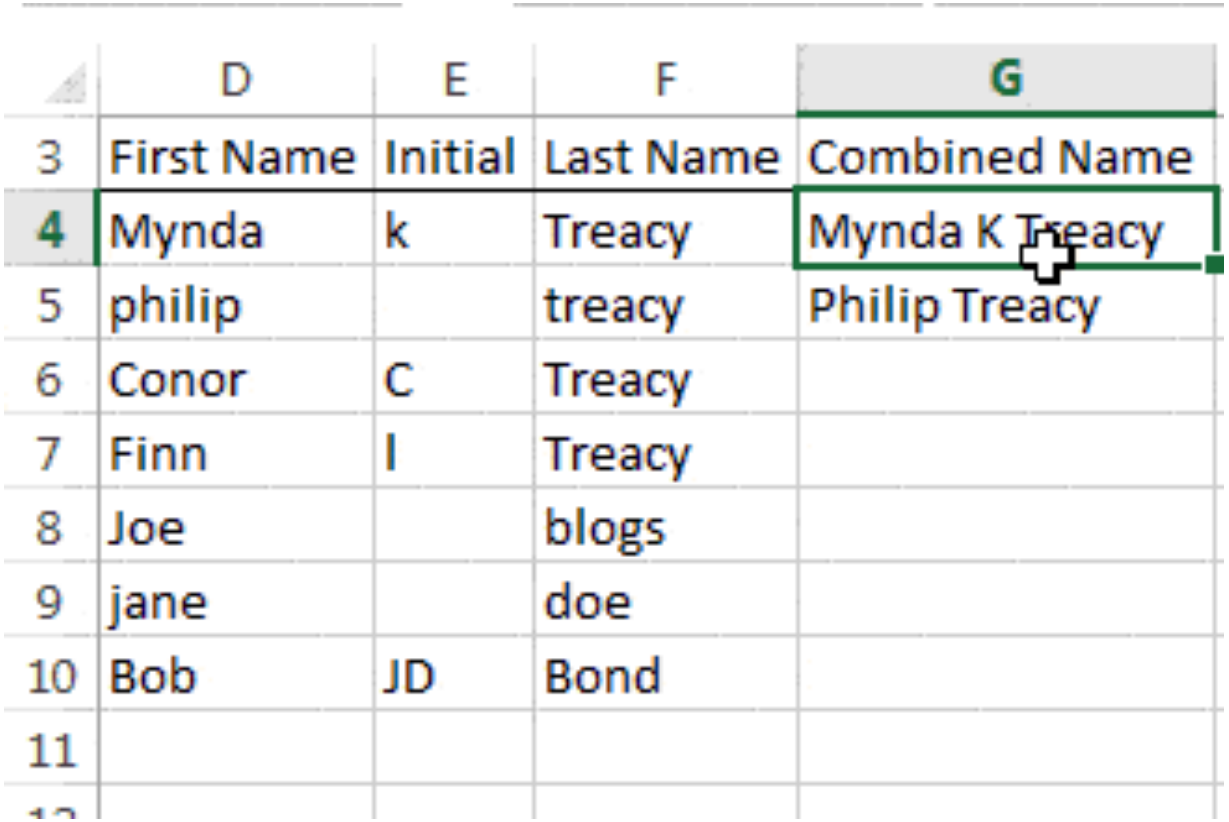

source: https://www.fanaticalfuturist.com/2019/11/the-worlds-most-dangerous-ai-is-now-helping-automate-coding

How AI Learns to Write Code?

- Learning from examples
  - Using neural network to automatically fill in Excel table cells

- Natural language to code generation
  - Natural language interface for database (NLIDB)
  - Generating different programming language
  - Challenge 1: generating longer code
  - Challenge 2: handling natural language ambiguity and cross-domain data

Task Execution by Learning from Examples



	D	E	F	G
3	First Name	Initial	Last Name	Combined Name
4	Mynda	k	Treacy	Mynda K Treacy
5	philip		treacy	Philip Treacy
6	Conor	C	Treacy	
7	Finn	I	Treacy	
8	Joe		blogs	
9	jane		doe	
10	Bob	JD	Bond	
11				

Microsoft FlashFill (Excel 2013)

Gulwani. *Automating String Processing in Spreadsheets Using Input-Output Examples.* 2012

How FlashFill Learns to Unify the Format

- 3-Step Rule-based system
  - Step 1: Generating the trace set from input -> output
  - Step 2: Partition the input examples using if conditions

Input v_1	*Output*
323-708-7700	*323-708-7700*
(425)-706-7709	*425-706-7709*
510.220.5586	*510-220-5586*
235 7654	*425-235-7654*
745-8139	*425-745-8139*

  - Step 3: Create rules for each set in the partition

How FlashFill Learns to Unify the Format

- Manually implemented rules to handle different cases of name, phone number, dates, etc.
- e.g., extracting the first name:

Input v_1	*Output*
Dr. Eran Yahav	*Yahav, E.*
Prof. Kathleen S. Fisher	*Fisher, K.*
Bill Gates, Sr.	*Gates, B.*
George Ciprian Necula	*Necula, G.*
Ken McMillan, II	*McMillan, K.*

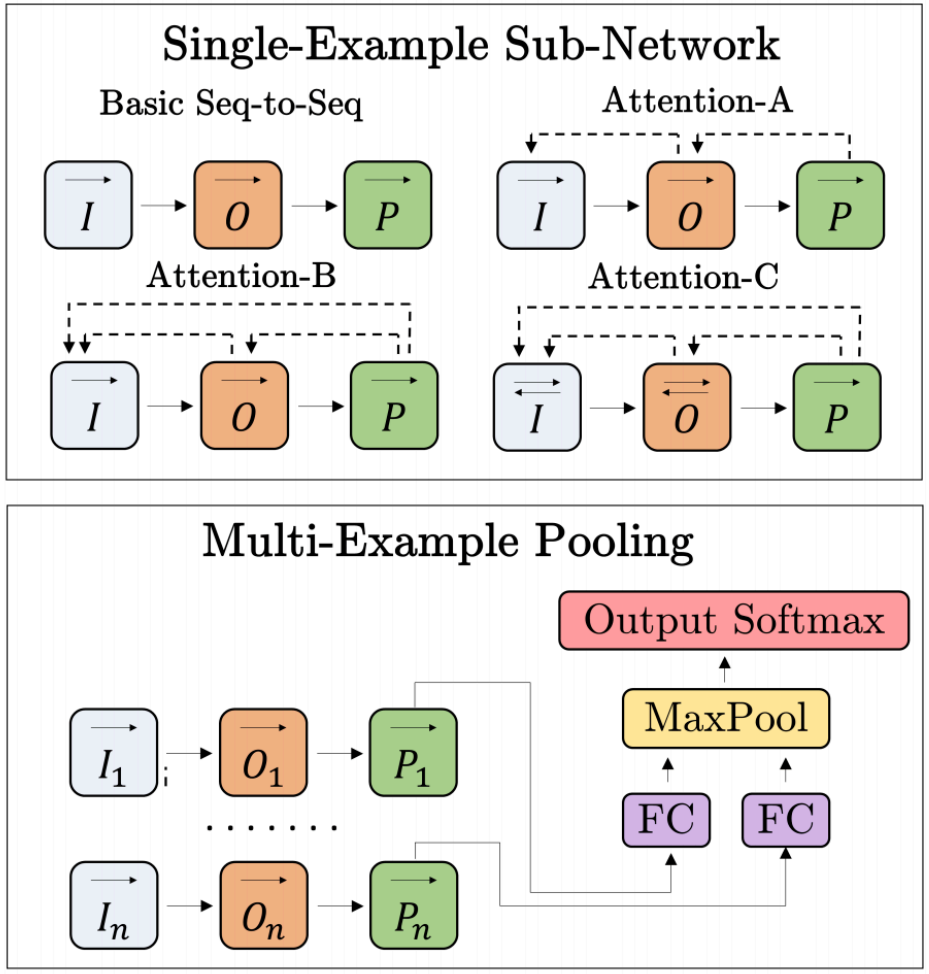
String Program for extracting initial of the first name:
The logic used is that of extracting the initial of the first word not followed by a dot: $\texttt{SubStr}(v_1, p_1, p_2)$, *where*
$p_1 \equiv \texttt{Pos}(\epsilon, \texttt{TokenSeq}(\texttt{AlphTok}, \textit{NonDotTok}), 1)$, *and*
$p_2 \equiv \texttt{Pos}(\epsilon, \texttt{TokenSeq}(\texttt{LowerTok}, \textit{NonDotTok}), 1)$.

FlashFill is not robust for handling typos

Input String	Output String
john Smith	Smith, Jhn
DOUG Q. Macklin	Macklin, Doug
Frank Lee (123)	LEe, Frank
Laura Jane Jones	Jones, Laura
Steve P. Green (9)	?
Program	
`GetToken(Alpha, -1) | ',' | ' ' | ToCase(Proper, GetToken(Alpha, 1))`	

Devlin et al. *RobustFill: Neural Program Learning under Noisy I/O.* 2017

RobustFill: Synthesis using Neural Network



Input String	Output String
john Smith	Smith, Jhn
DOUG Q. Macklin	Macklin, Doug
Frank Lee (123)	LEe, Frank
Laura Jane Jones	Jones, Laura
Steve P. Green (9)	?
Program	
`GetToken(Alpha, -1) | ',' | ' ' | ToCase(Proper, GetToken(Alpha, 1))`	

$$y_i = \text{Softmax}(Vm_i),$$

$$m_i = \text{MaxPool}_{j \in n}(\tanh(Wh_{ji})),$$

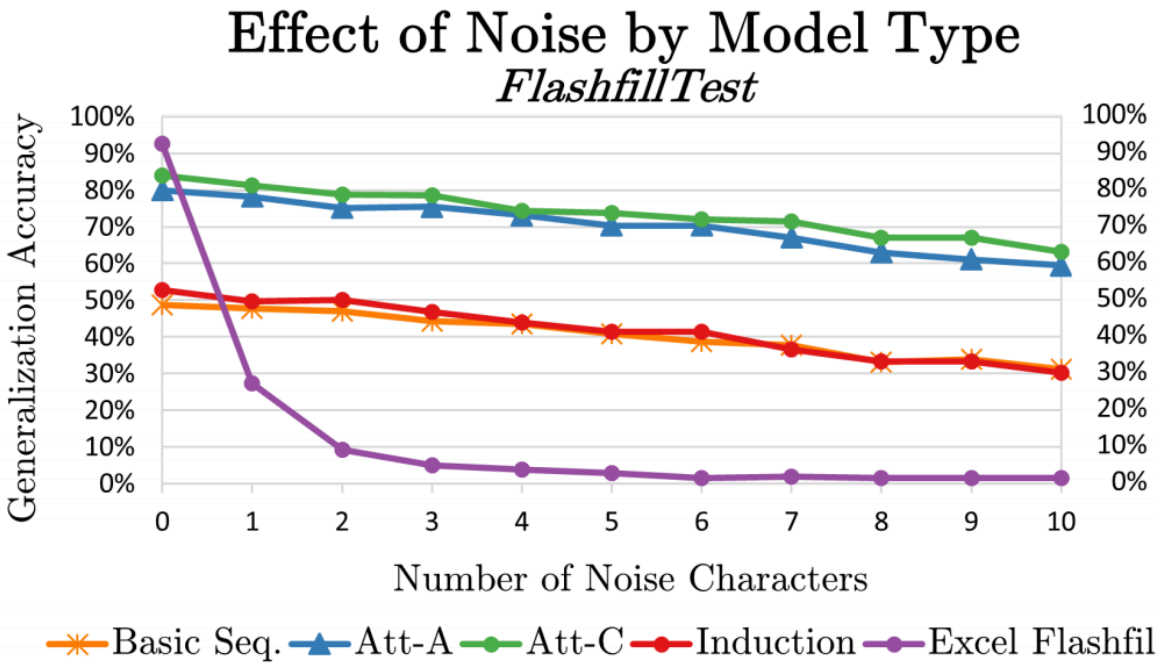
i: current time step

n: observed example

Devlin et al. *RobustFill: Neural Program Learning under Noisy I/O.* 2017

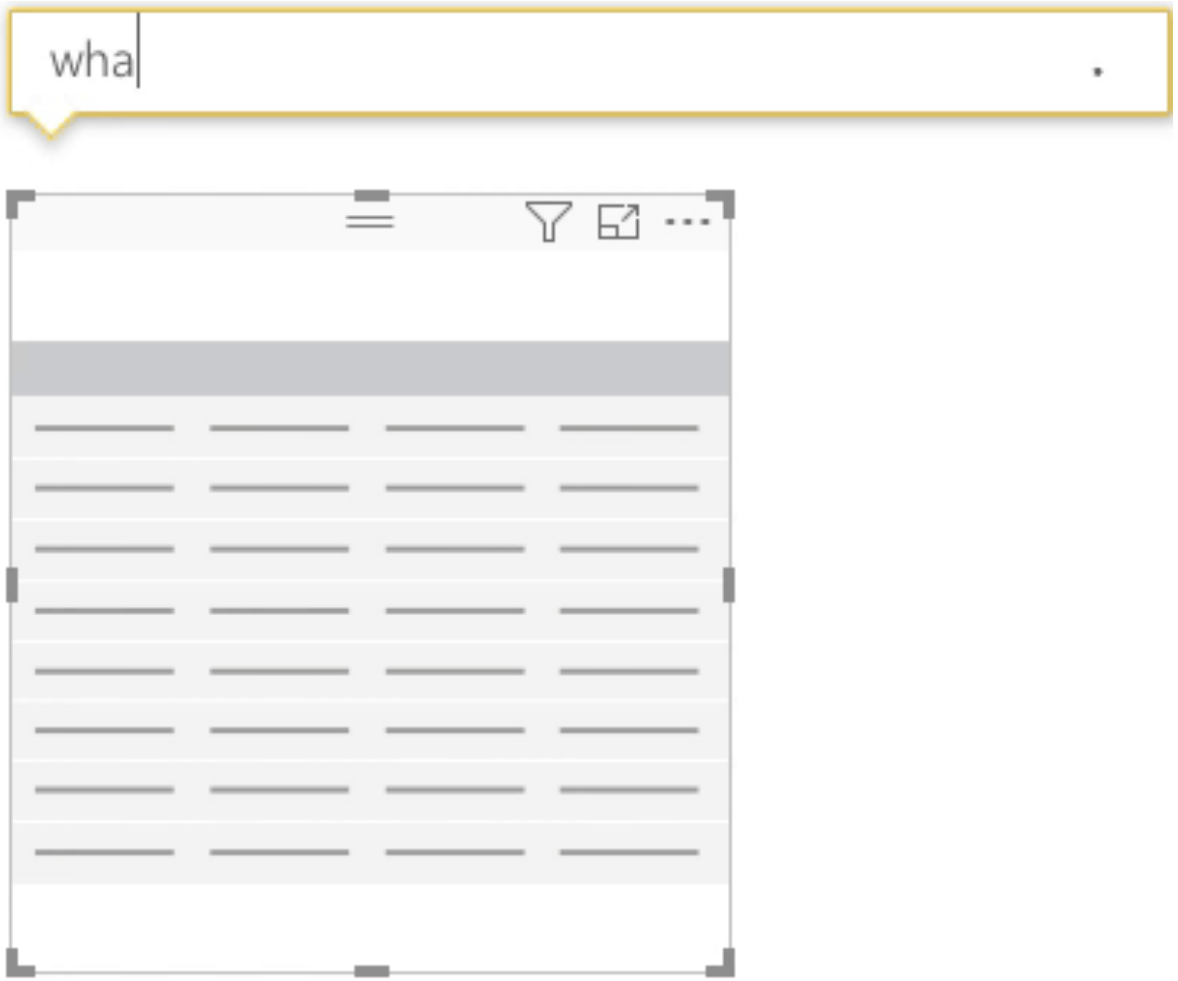
Problem with FlashFill

- Experiment: automatically injecting typos



Devlin et al. *RobustFill: Neural Program Learning under Noisy I/O.* 2017

Natural Language to Database Interface (NLIDB)



Natural Language to Database Interface (NLIDB)

"What is the breed of the dog named 'Betty' ?

SELECT breed FROM dog WHERE dog_**name** = 'Betty'

table 1: dog

breed	…	**dog_name**	dog_id
Husky		'Betty'	001
Yorkie		'Larry'	002
Poodle		'Teddy'	003
…		…	…

table 2: owner

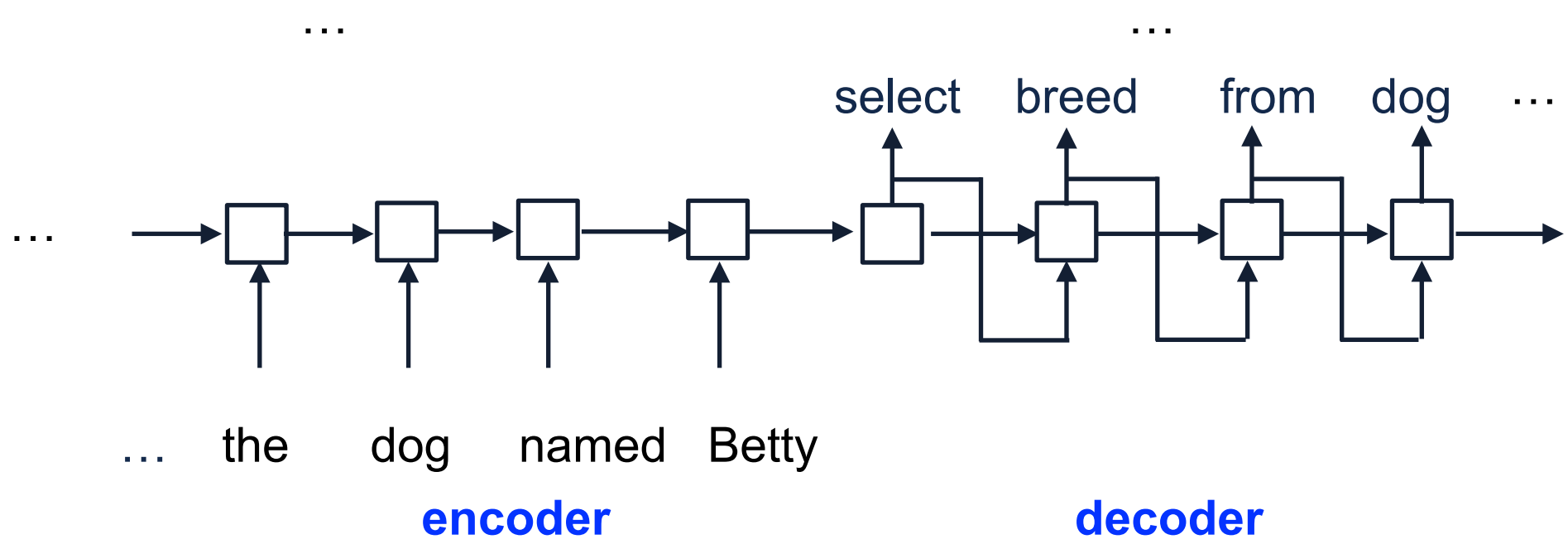
id	…	**name**
101		Elizabeth
102		David
…		…

Using Neural Network to Generate SQL Statements

- Sequence to sequence (neural language modeling)

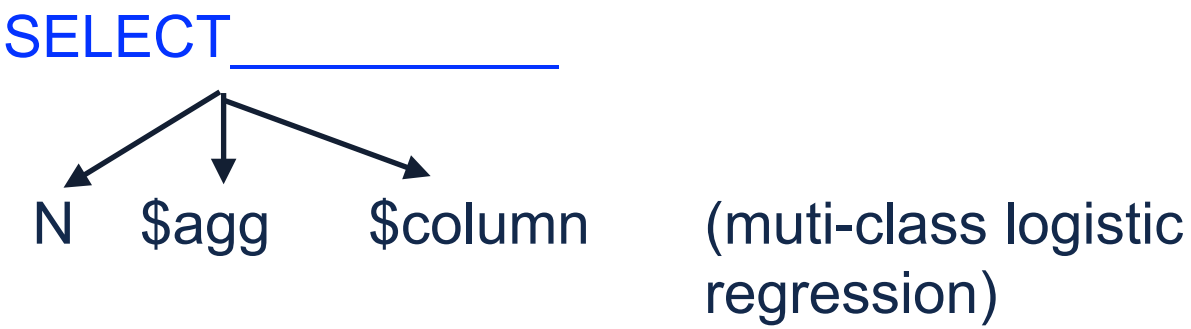
"What is the breed of the dog named 'Betty' ?

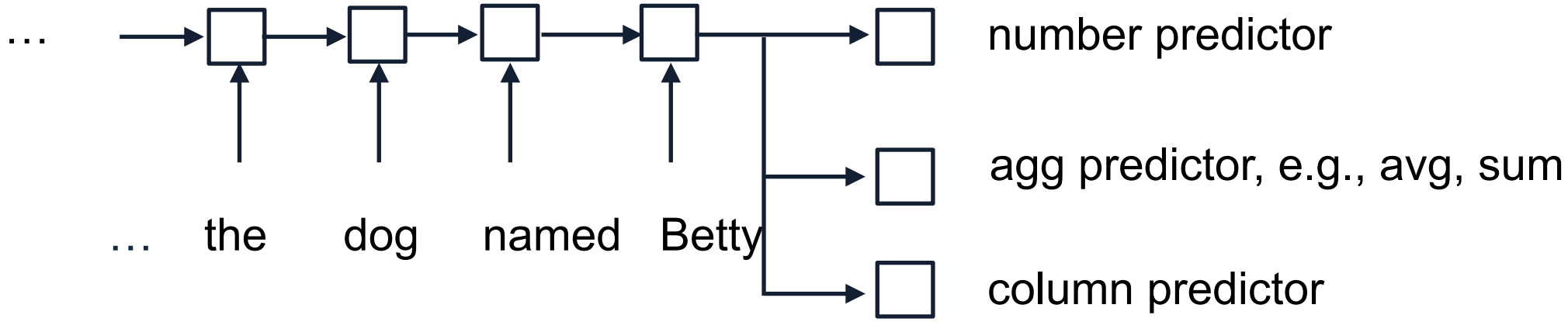
SELECT breed FROM dog
WHERE name = 'Betty'



Incorporating Grammar in Neural Network

- SQLNet [Xu et al. 2017]
  - "What is the breed of the dog named 'Betty' ?

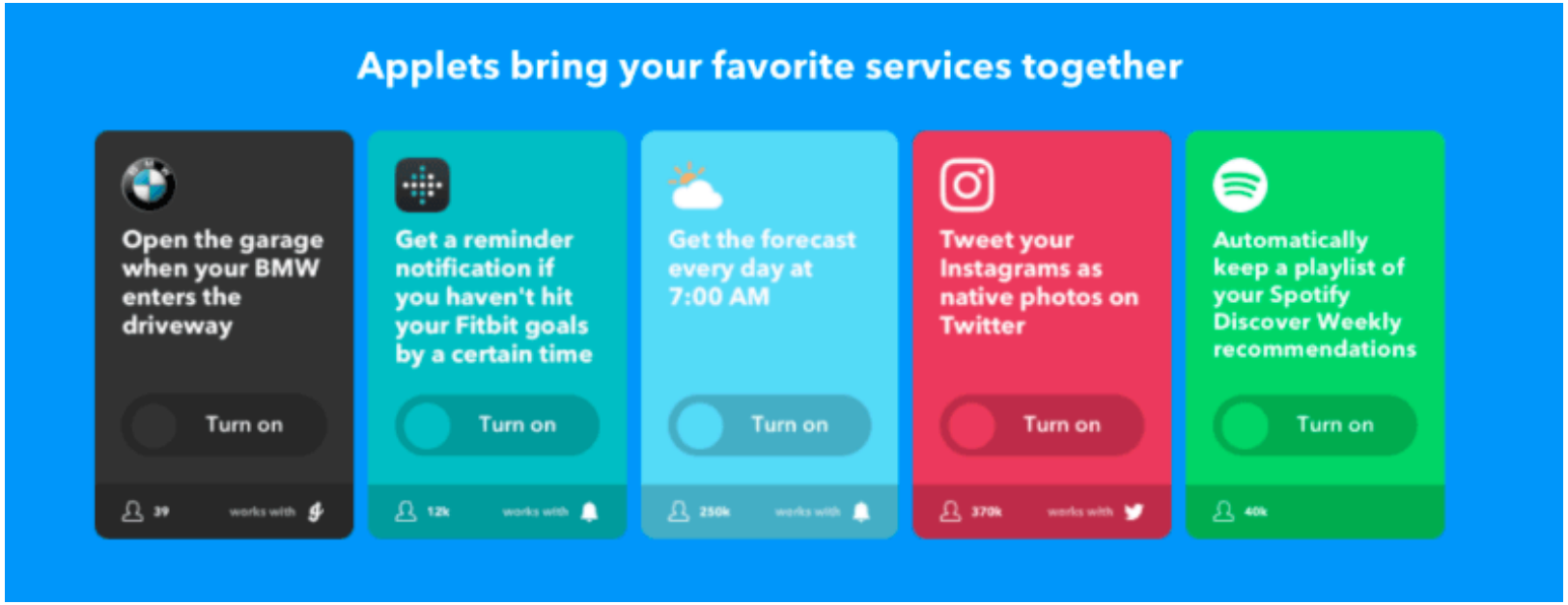




Natural Language to Any Programming Language…

SQL	Guo et al. 2019	60%	Complex SELECT statement, nested query
RegEx	Zhong et al. 2019	62%	Short RegEx expressions
Bash	Lin et al. 2018	58%	display the 5 largest files in the current directory and its sub-directories; `find . -type f | sort -nk 5,5 | tail -5`
Python	Yao et al. 2018	16%	A Python class from a short description
Java	Ling et al. 2019	4.8%	A Java class from a short description
C#	Iyer et al. 2016	-	

Generating Domain Specific Language (DSL)



- **Autosave Instagram photos** to your **Dropbox folder**

The Challenge in Generating Different Programming Languages

- It is easier to generate high-level script language than low-level programming languages

Source code of C ++ 'Bubble sort'

```
// This procedure realizes Bubble sort
void BubbleSort(int a[], int n)
{
  int right = n-1;
  int temp = 0;
  while(right > 0)
  {
    int k = 0;
    for(int i = 0; i < right; i ++)
      if(a[i+1] < a[i])
      {
        temp = a[i+1];
        a[i+1] = a[i];
        a[i] = temp;
        k = i;
      }
    right = k;
  }
}
```

Sorting in C++

```
array = sorted(array)
```

Sorting in Python

How to Generate Low-level Programming Languages?

- Leveraging an intermediate layer

```
SELECT name FROM stadium EXCEPT SELECT
T2.name FROM concert AS T1 JOIN stadium AS
T2 ON T1.stadium_id  =  T2.stadium_id WHERE
T1.year  =  2014
```

Root1(2) Root(5) Sel(0) N(0) A(0) C(3) T(0) Root(3) Sel(0) N(0) A(0) C(3) T(0) Filter(2) A(0) C(17) T(2)

Handling Natural Language Ambiguity and Cross-Domain

- Natural language is ambiguous!

  Table 1: Facebook
  PostId, timestamp, likes, etc.

  Table 2: Highschool
  StudentId, likes

- Cross-domain Challenges
  - Training and testing schemas are completely non-overlapping

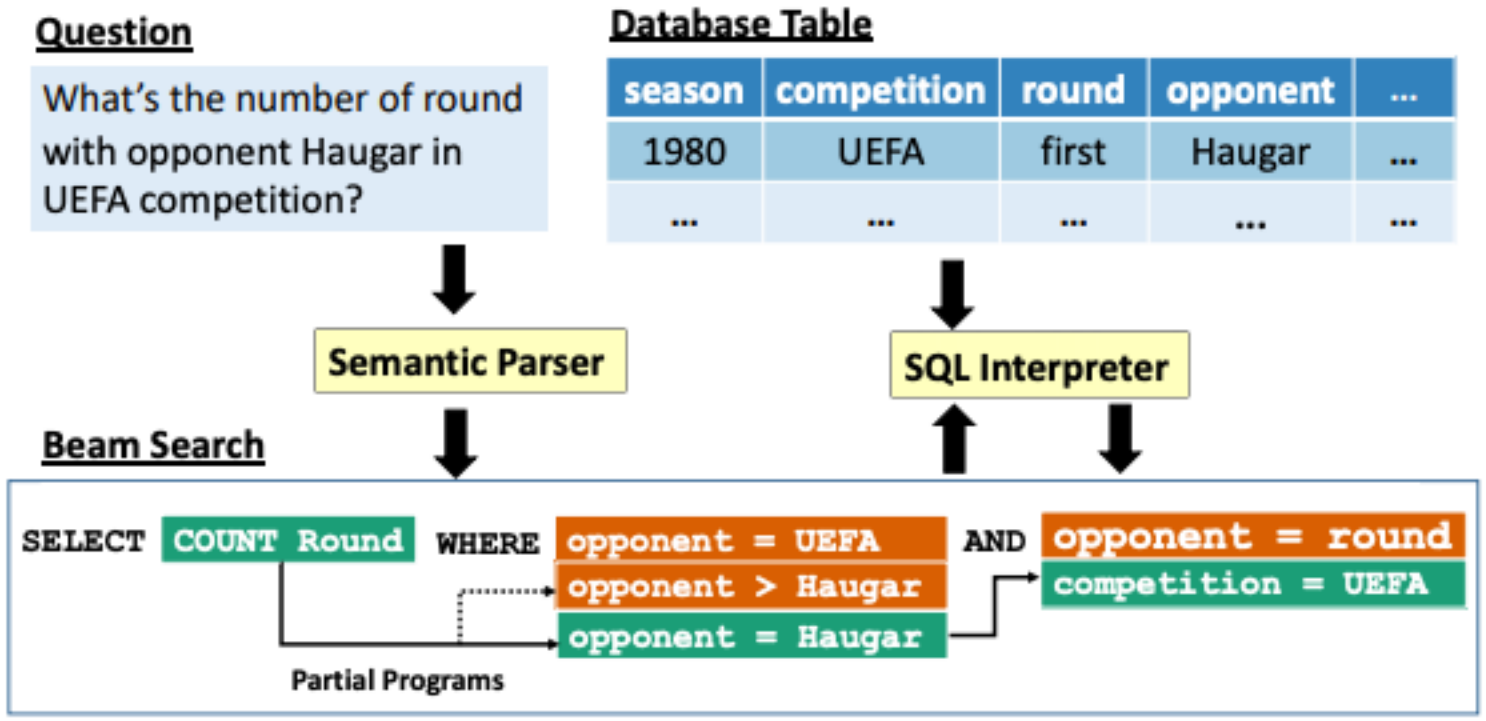
  What is the average salaries of all employees?

  Table 1: salary
  salary, average

  SELECT avg(salary) FROM salary

  SELECT average FROM salary

Leveraging Execution Results as Guidance [Wang et al. 2018]



Industry Practice and Future Work

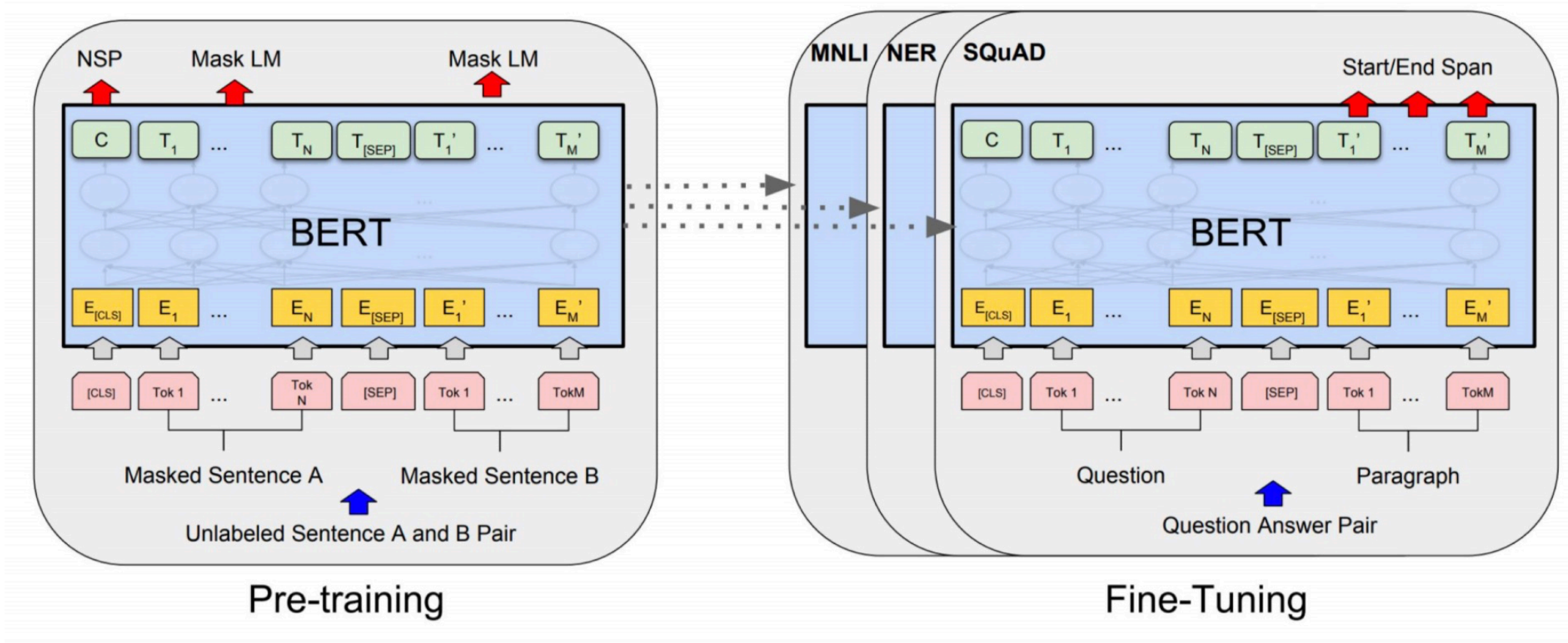
- Generating IoT commands
  - IFTTT, Alexa, etc.
- Kite
  - AI autocompletion
  - raised $17M
- Future directions
  - Longer, more complex datasets
  - More “natural” code than synthesized datasets

Frontier topics

- Neural program synthesis
- **Post-BERT pre-training advancements**
- AutoML; hyperparameter optimization
- Fairness/bias in AI
- Machine learning explainability for text mining/NLP

BERT: Bidirectional Encoder Representations

Fine-tuning BERT



RoBERTa [Liu et al. 2019]

- RoBERTa: A Robustly Optimized BERT Pre-training Approach (Liu et al, University of Washington and Facebook, 2019)

- An ablation study on BERT for design choices:
  - **Static vs. dynamic masking**
  - **Sentence completeness**:
    - segment pair + NSP, sentence pair + NSP, full sentences, doc sentences
  - **Larger batch sizes**
    - batch size = 256, 2K, 8K
  - **Larger BPE vocab**
    - roberta -> “ro” “bert” “a##”
    - 50K subword units

Fine-tuning on GLUE datasets [Wang et al. 2019]

- Multi-task sentence classification &regression (Stanford sentiment treebank)
- Natural language inference (MultiNLI)
  - Entailment, contradiction, neutral
- Corpus of linguistic acceptability (CoLA)
  - Whether the sentence is grammarly correct

MultiNLI

Premise: Hills and mountains are especially sanctified in Jainism.
Hypothesis: Jainism hates nature.
Label: Contradiction

CoLa

Sentence: The wagon rumbled down the road.
Label: Acceptable

Sentence: The car honked down the road.
Label: Unacceptable

RoBERTa fine tuning on GLUE

- Two settings for fine-tuning GLUE

- Single task on dev
  - batch size {16, 32}
  - lr = {1e-5, 2e-5, 3e-5}
  - linear warmup for the first 6% of steps

- Ensemble on test
  - For RTE, STS-B & MRPC, helpful to fine tune starting from model fine-tuned on MNLI
  - Run 15 different seeds for the selected hyperparam, and ensemble top 7 model based on dev set metrics

RoBERTa results

Model	SQuAD 1.1		SQuAD 2.0	
	EM	F1	EM	F1
Single models on dev, w/o data augmentation				
$\text{BERT}_{\text{LARGE}}$	84.1	90.9	79.0	81.8
$\text{XLNet}_{\text{LARGE}}$	**89.0**	94.5	86.1	88.8
RoBERTa	88.9	**94.6**	**86.5**	**89.4**
Single models on test (as of July 25, 2019)				
$\text{XLNet}_{\text{LARGE}}$			86.3†	89.1†
RoBERTa			86.8	89.8
XLNet + SG-Net Verifier			**87.0**†	**89.9**†

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-

XLNet [Yang et al. 2019, CMU and Google]

- Innovation #1: Relative position embeddings
  - Sentence: John ate a hot dog
  - Relative attention: “How much should dog attend to hot (in any position) and how much should dog attend to the previous word?”

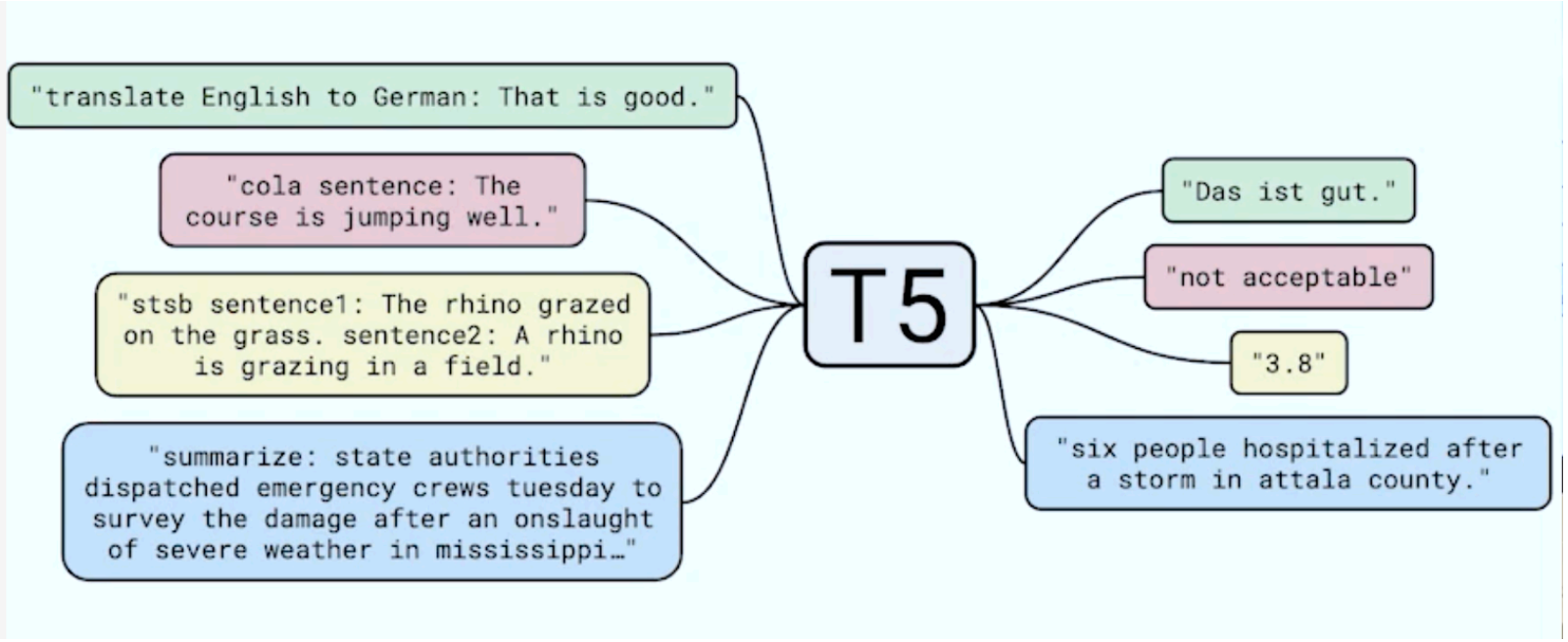
- Innovation #2: Randomly permute the training sequences

- Training with more data and larger models..

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B
Single-task single models on dev								
BERT [2]	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0
RoBERTa [21]	90.2/90.2	94.7	92.2	**86.6**	96.4	**90.9**	68.0	92.4
XLNet	**90.8/90.8**	**94.9**	**92.3**	85.9	**97.0**	90.8	**69.0**	**92.5**

T5: a seq-to-seq framework [Raffel et al. 2019]

- T5: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (Raffel et al, Google, 2019)



T5: a seq-to-seq framework [Raffel et al. 2019]

- T5 conclusions:
  - Scaling up model size and amount of training data helps a lot
  - Best model is 11B parameters (BERT-Large is 330M), trained on 120B words of cleaned common crawl text
  - Exact masking/corruptions strategy doesn't matter that much

	Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm
	1	HFL iFLYTEK	MacALBERT + DKM		90.7	74.8	97.0	94.5/92.6	92.8/92.6	74.7/90.6	91.3	91.1
+	2	Alibaba DAMO NLP	StructBERT + TAPT		90.6	75.3	97.3	93.9/91.9	93.2/92.7	74.8/91.0	90.9	90.7
+	3	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0	91.6	91.3
	4	ERNIE Team - Baidu	ERNIE		90.4	74.4	97.5	93.5/91.4	93.0/92.6	75.2/90.9	91.4	91.0
	5	T5 Team - Google	T5		90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	91.9

LongFormer [Beltagy et al. 2020]

- Transformer-based models suffer from quadratic complexity due to **self attention**

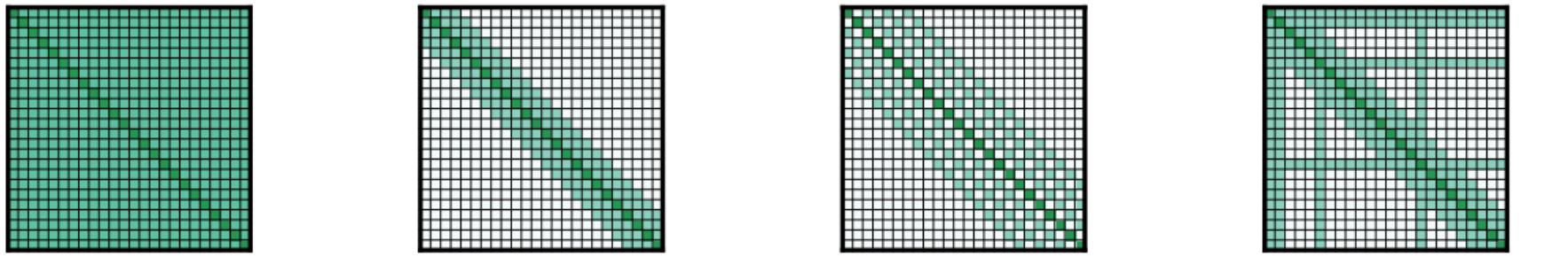
- Proposes an attention mechanism that scales **linearly** with sequence length
  - Making it easy to process documents of thousands of tokens or longer

Model	base	large
RoBERTa (seqlen: 512)	1.846	1.496
Longformer (seqlen: 4,096)	10.299	8.738
+ copy position embeddings	1.957	1.597
+ 2K gradient updates	1.753	1.414
+ 65K gradient updates	1.705	1.358

	QA			Coref.	Classification	
Model	WikiHop	TriviaQA	HotpotQA	OntoNotes	IMDB	Hyperpartisan
RoBERTa-base	72.4	74.3	63.5	78.4	95.3	87.4
Longformer-base	**75.0**	**75.2**	**64.4**	**78.6**	**95.7**	**94.8**

LongFormer [Beltagy et al. 2020]

- **Full n^2 attention**: O(n^2)
- **Sliding window**: O(n x w)
- **Dilated sliding window**: O(l x d x w)
- **Global attention**: O(n)



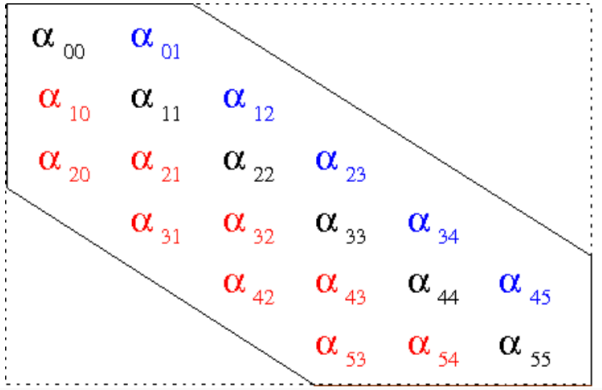
(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

LongFormer [Beltagy et al. 2020]

- In regular attention, attention scores are computed using

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

- The dilated sliding window attention not straightforward using modern DL libs
  - Banded matrix multiplication, not supported in PyTorch/TF
  - Naive implementation extremely slow
  - Provide a highly efficient CUDA kernel using TVM [Chen et al. 2018]: high level description of a function -> optimized device specified code



Domain-specific BERT

- BERT has different performance in specific
  - e.g., programming language domain, biomedicine
  - “json”: “##j”, “son”

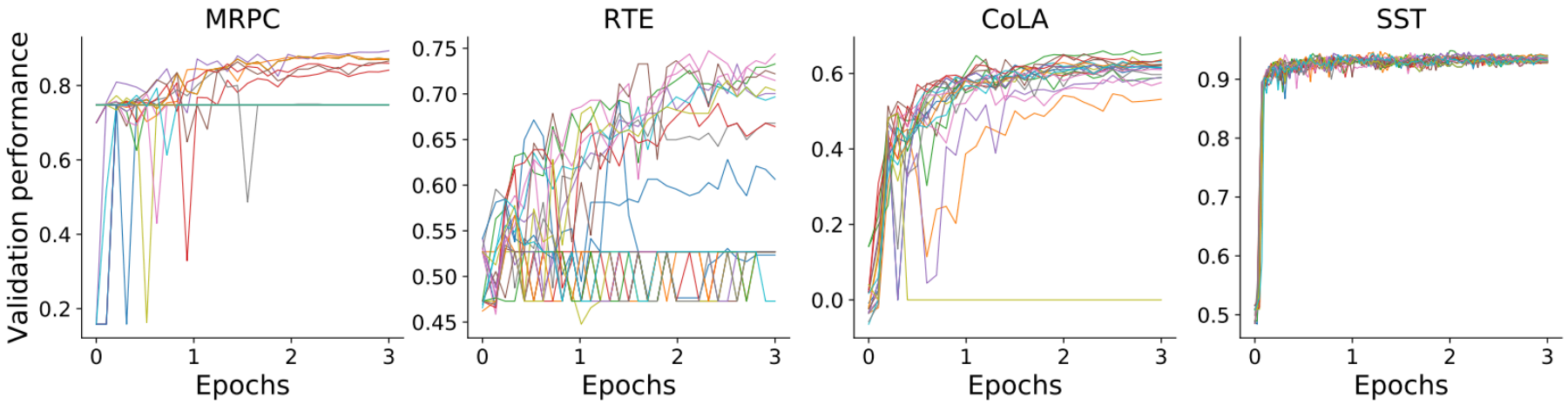
- **CodeBERT [Feng et al. 2020]**
  - **GitHub code**: 2.1M bimodal datapoints and 6.4M unimodal codes across six programming languages (Python, Java, JavaScript, PHP, Ruby, and Go)
  - **Downstream task**: NL2code search, code2NL generation

- **SciBERT** [Beltagy et al. 2019], **BioBERT** [Lee et al. 2019]
  - **Corpus**: scientific papers
  - **Fine tune task**: NER, relation extraction, classification

Effect of random seed [Dodge et al. 2020]

- Fine-tuning language models is brittle
  - Substantially different performance across different training episodes [Phang et al. 2018]

- Rerun the same experiments, fixing everything, varying random seed:
  - Weight initialization for fine-tuning
  - Data order



Frontier topics

- Neural program synthesis
- Pretrained LMs
- **AutoML; hyperparameter optimization**
- Fairness/bias in AI
- Machine learning explainability for text mining/NLP

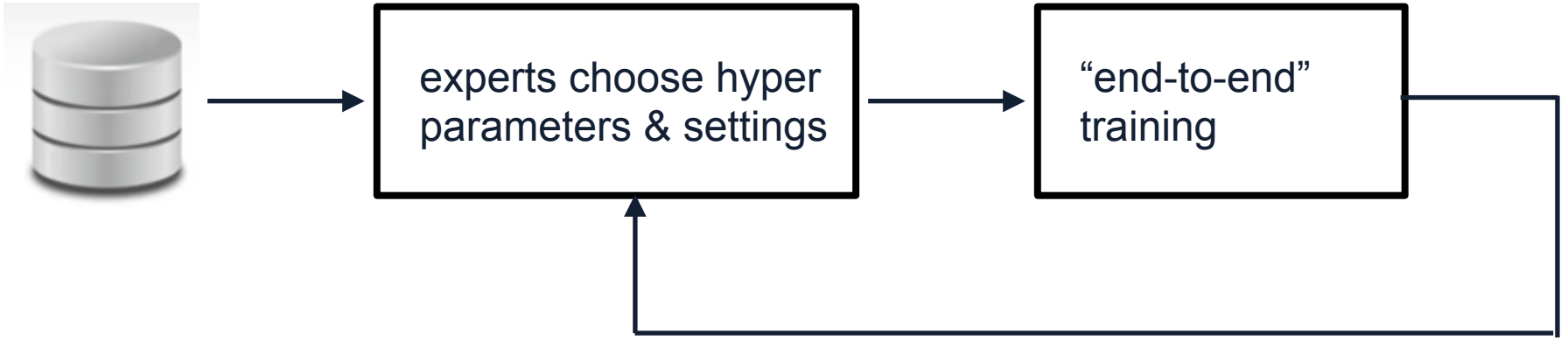
Hyperparameter optimization

- Deep neural network is sensitive to hyperparameters
- Tuning hyperparameter is a black magic
- Training requires significant time and efforts!
  - P100 ~$500 per month
  - V100 ~$1,000- $1,500 per month

NVIDIA® Tesla® V100	1 GPU	16 GB HBM2	$2.48 USD per GPU
	2 GPUs	32 GB HBM2	
	4 GPUs	64 GB HBM2	
	8 GPUs	128 GB HBM2	

Automated machine learning

Current deep learning practice



AutoML: true end-to-end training

source: Automatic Machine Learning (NeurIPS 2018 Tutorial). Frank Hutter and Joaquin Vanschoren

Auto-sklearn [Feurer et al. 2018]

- Tuning happens behind the scene

sklearn:
NB

```
>>> X = [[0], [1], [2], [3]]
>>> y = [0, 0, 1, 1]
>>> from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
>>> neigh.fit(X, y)
KNeighborsClassifier(...)
>>> print(neigh.predict([[1.1]]))
[0]
>>> print(neigh.predict_proba([[0.9]]))
[[0.66666667 0.33333333]]
```

auto-sklearn

```
>>> import autosklearn.classification
>>> cls = autosklearn.classification.AutoSklearnClassifier()
>>> cls.fit(X_train, y_train)
>>> predictions = cls.predict(X_test)
```

Hyperparameter tuning

- Deep neural network is sensitive to hyper parameters

Config 1:
lr = 2e-5
batch size = 32
epoch = 3.0
max_seq_length = 128

Config 2:
lr = 5e-5
batch size = 128
epoch = 10.0
max_seq_length = 128

- Which config to choose?

BERT: epoch = 3.0, RoBERTa: epoch = 10.0

Hyperparameter optimization

- Definition of hyperparameter tuning:

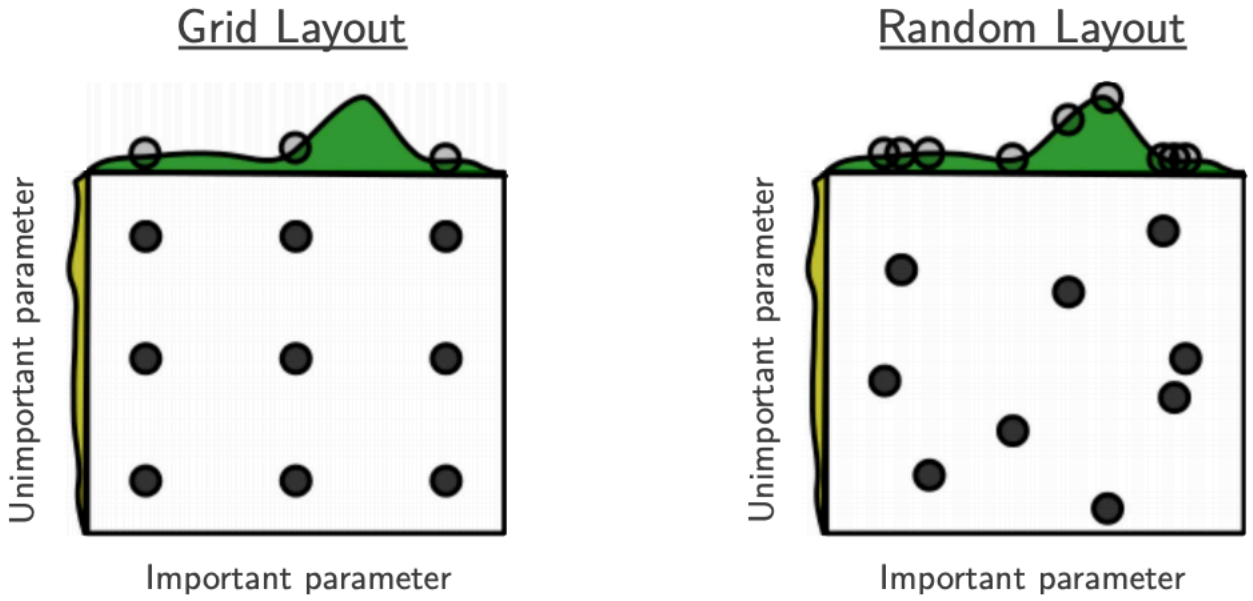
$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}}{\operatorname{argmin}} \, \mathbb{E}_{(D_{train}, D_{valid}) \sim \mathcal{D}} \mathbf{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, D_{train}, D_{valid}),$$

- $\boldsymbol{\lambda}$: hyperparameter
- $\boldsymbol{\Lambda}$: search space for hyperparameter

- D_{train}: training dataset
- D_{valid}: validation dataset
- L: loss function (objective)

Hyperparameter optimization [Bergstra et al. 2012]

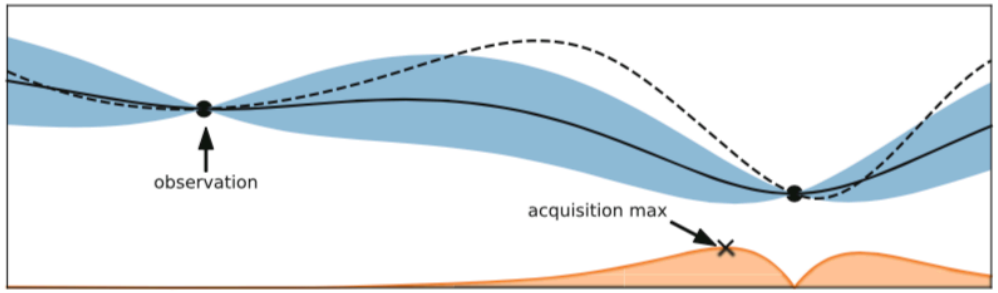
- Grid search vs. random search

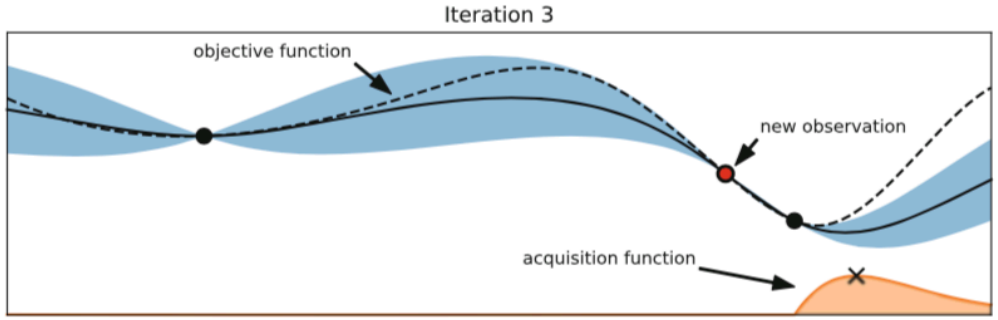


- Grid search: explored 3 important hyperparameters in 9 **trials**
- Random search: explored 9 hyperparameters in 9 trials

Bayesian optimization [Bergstra et al. 2012]

- Approach: fix a probabilistic model to the function evaluation
- Use that model for exploration vs. exploitation
- During the development of AlphaGO, many hyper parameters were tuned with BO many times, e.g., before the match with Sedol Lee, HPO improved the win-rate from 50% -> 65%





SMBO(f, M_0, T, S)

1 $\mathcal{H} \leftarrow \emptyset$,
2 For $t \leftarrow 1$ **to** T,
3 $x^* \leftarrow \text{argmin}_x S(x, M_{t-1})$,
4 Evaluate $f(x^*)$, ▷ *Expensive step*
5 $\mathcal{H} \leftarrow \mathcal{H} \cup (x^*, f(x^*))$,
6 Fit a new model M_t to $\mathcal{H}$.
7 **return** $\mathcal{H}$

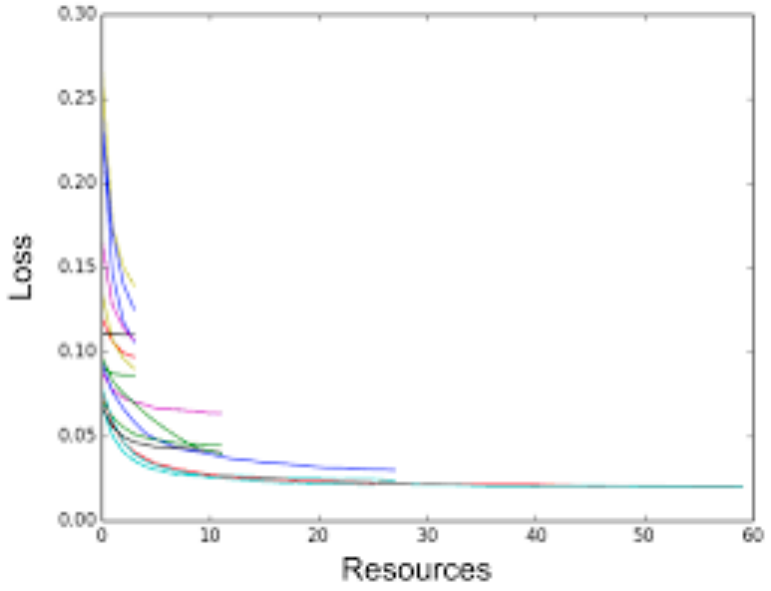
acquisition function

source: Automatic Machine Learning (NeurIPS 2018 Tutorial). Frank Hutter and Joaquin Vanschoren

Early stopping and pruning

- If a trial shows bad performance, no need to keep running it

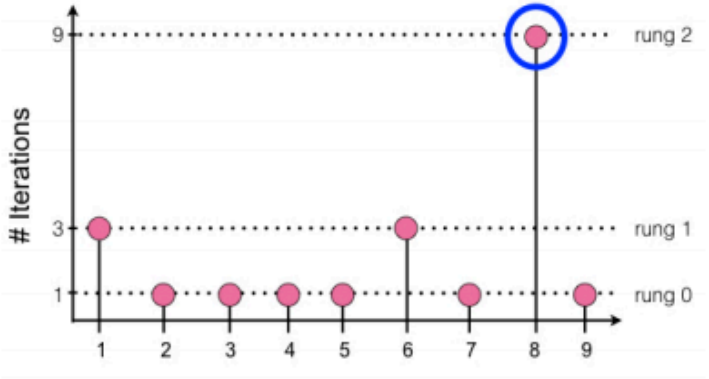
- Successive halving [Jamieson et al. 2015]
  - A fixed amount of resources
  - Allocate the resources by first aggressively exploring, then exploiting



	$s = 4$	
i	n_i	r_i
0	81	1
1	27	3
2	9	9
3	3	27
4	1	81

Asynchronous Successive Halving [Li et al. 2020]

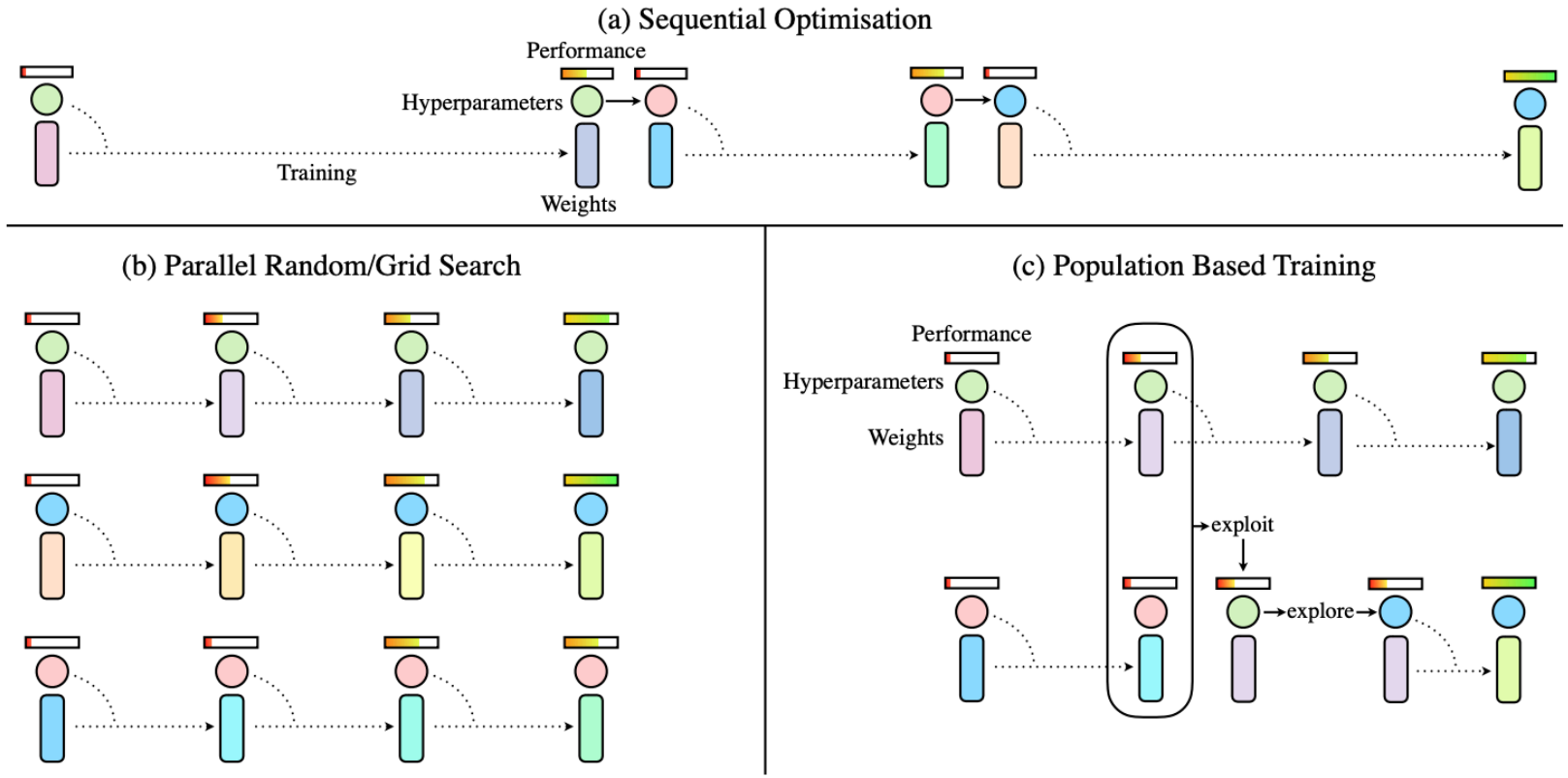
- Synchronous SHA are not efficient because the higher rungs are waiting for the lower rungs to finish
- Instead of waiting, we can promote lower rungs to higher rungs
- Time cost = 13/9 time (R) vs. 3 time (R)



bracket s	rung i	n_i	r_i	total budget
0	0	9	1	9
	1	3	3	9
	2	1	9	9
1	0	9	3	27
	1	3	9	27
2	0	9	9	81

Population-based training [Jaderberg et al. 2017]

- Evolutionary algorithm for hyperparameter optimization



Frontier topics

- Neural program synthesis
- **Compression for pretrained LMs**
- AutoML; Neural architecture search
- Fairness/bias in AI
- Machine learning explainability for text mining/NLP

Language models are growing rapidly in size

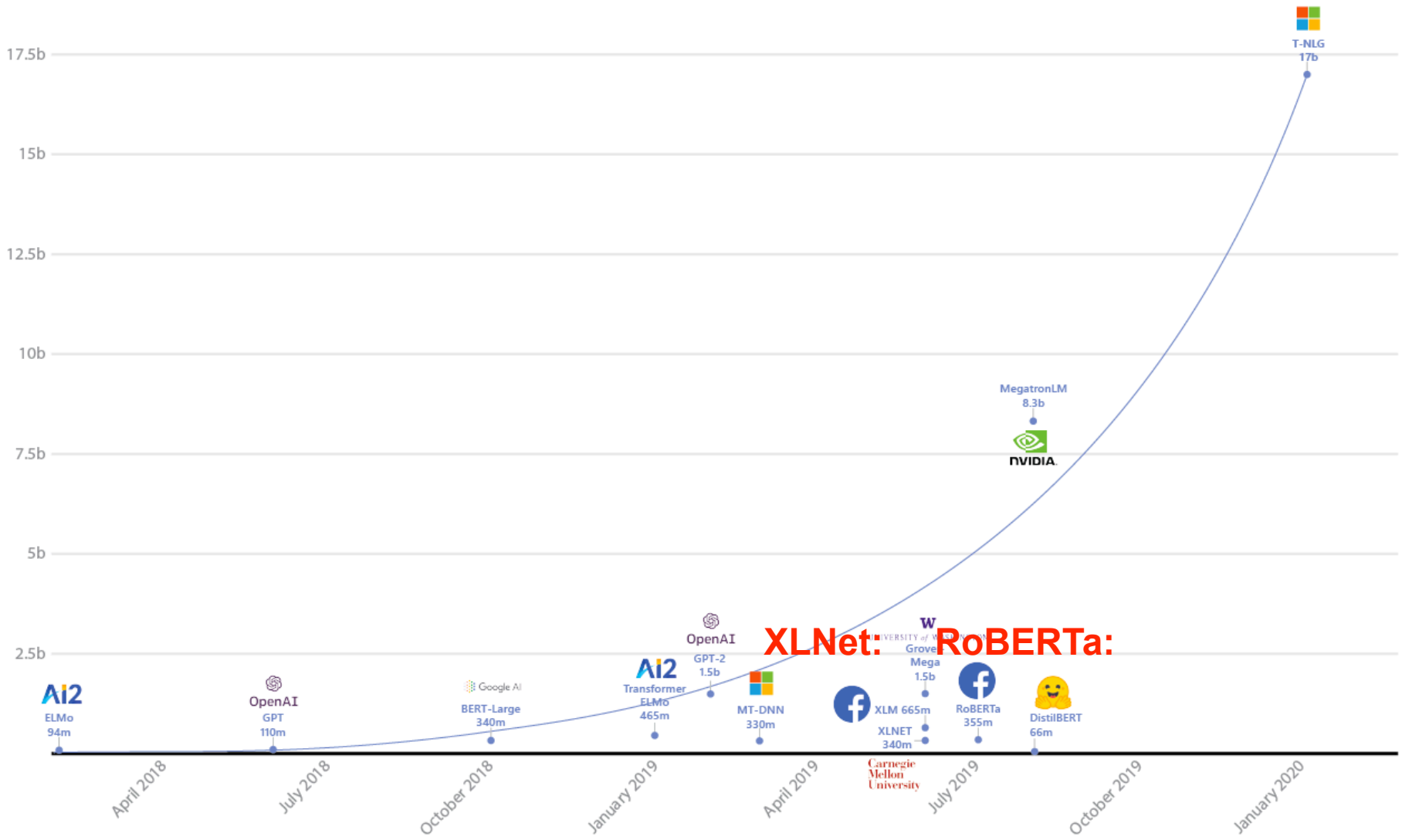
	1	HFL iFLYTEK	MacALBERT + DKM		90.7	74.8	97.0	94.5/92.6	92.8/92.6	74.7/90.6	91.3
+	2	Alibaba DAMO NLP	StructBERT + TAPT		90.6	75.3	97.3	93.9/91.9	93.2/92.7	74.8/91.0	90.9
+	3	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0	91.6
	4	ERNIE Team - Baidu	ERNIE		90.4	74.4	97.5	93.5/91.4	93.0/92.6	75.2/90.9	91.4
	5	T5 Team - Google	T5		90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2
	6	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART					93.7/91.6	92.9/92.5	73.9/90.2	91.0
+	7	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024)		89.7	70.5	97.5	93.4/91.2	92.6/92.3	75.4/90.7	91.4
+	8	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4	71.7	97.1	93.1/90.7	92.9/92.5	75.6/90.8	91.3
+	9	Huawei Noah's Ark Lab	NEZHA-Large		89.1	69.9	97.3	93.3/91.0	92.4/91.9	74.2/90.6	91.0
+	10	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4	68.0	96.8	93.1/90.8	92.3/92.1	74.8/90.3	91.1
	11	Junjie Yang	HIRE-RoBERTa		88.3	68.6	97.1	93.0/90.7	92.4/92.0	74.3/90.2	90.7
	12	Facebook AI	RoBERTa		88.1	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8

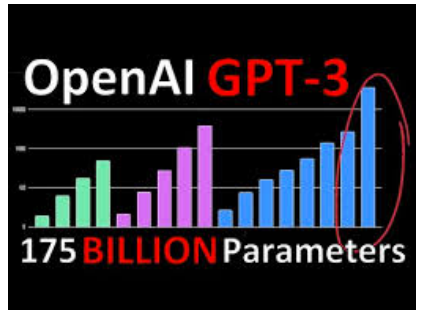
T5-large: 11B, 45G

ELECTRA-large: 356M, 1.3G

RoBERTa-large: 356M, 1.4G

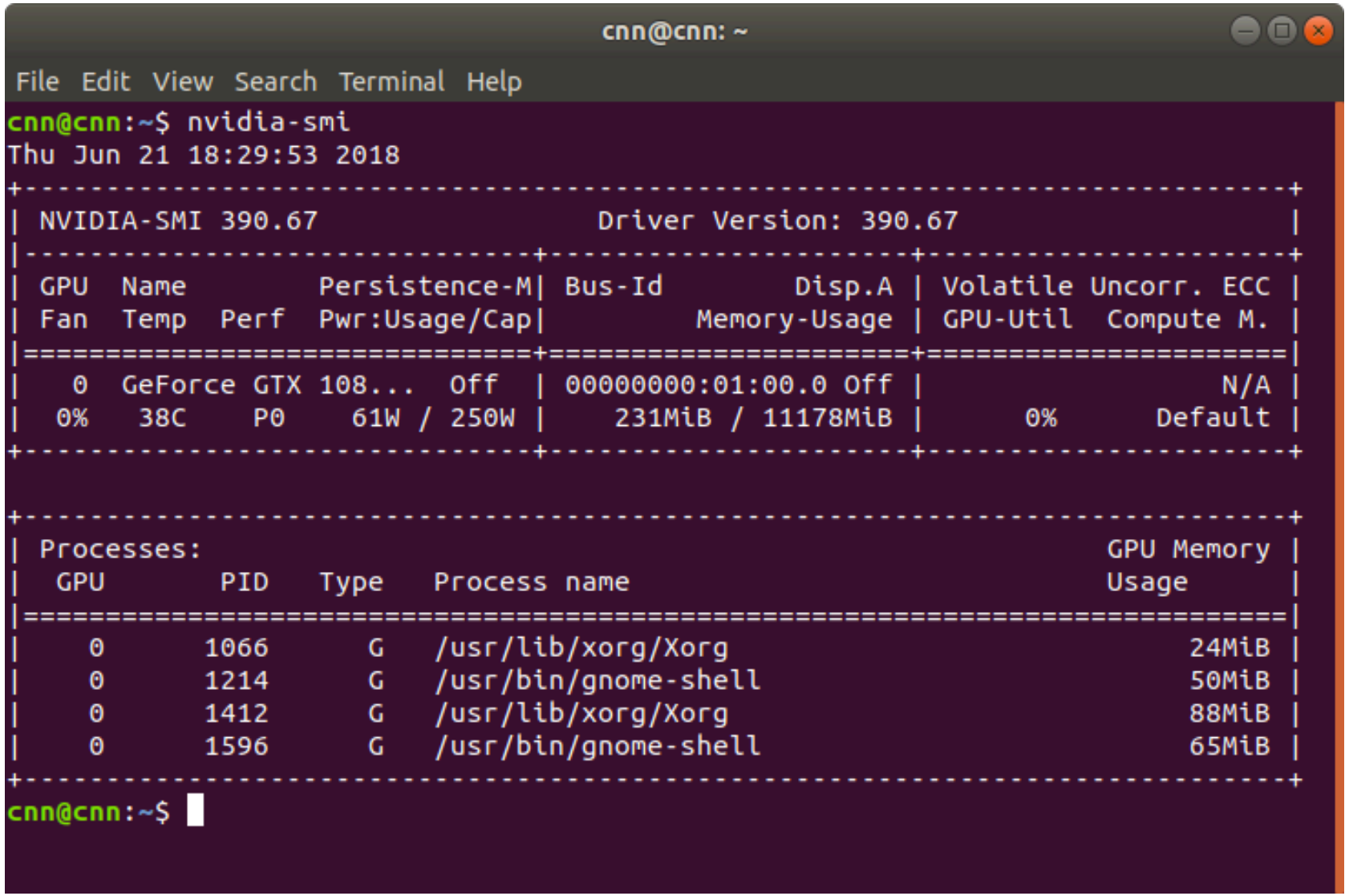
Language models are growing rapidly in size



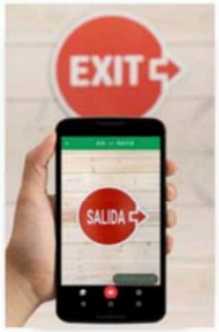


T5

Computation resource 💰 does not grow as fast



Deep learning on mobile/IoT devices



Phones

Drones

Robots

Glasses

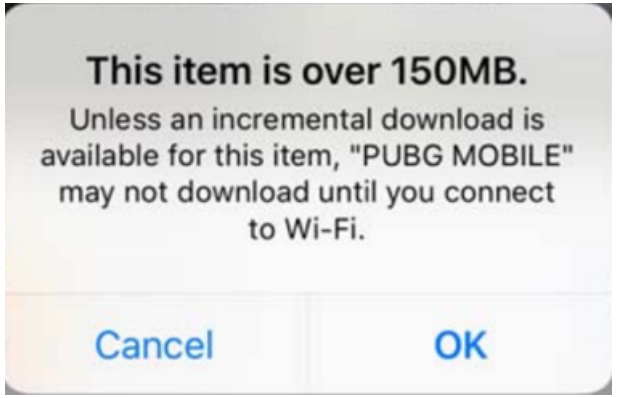
Self Driving Cars

Battery Constrained!

source: "Compression of Deep Learning Models for NLP", Gupta et al.

Resource constraints for pretrained LMs

- GPU rams
  - Can only run models up to the RAM of GPU
- Latency requirements
  - Applications deploying to large number of users, e.g., Google
- Mobile-first computational requirements
  - Apps > 150MB will receive much more scrutiny
- Cloud computing
  - Cost for renting GPUs

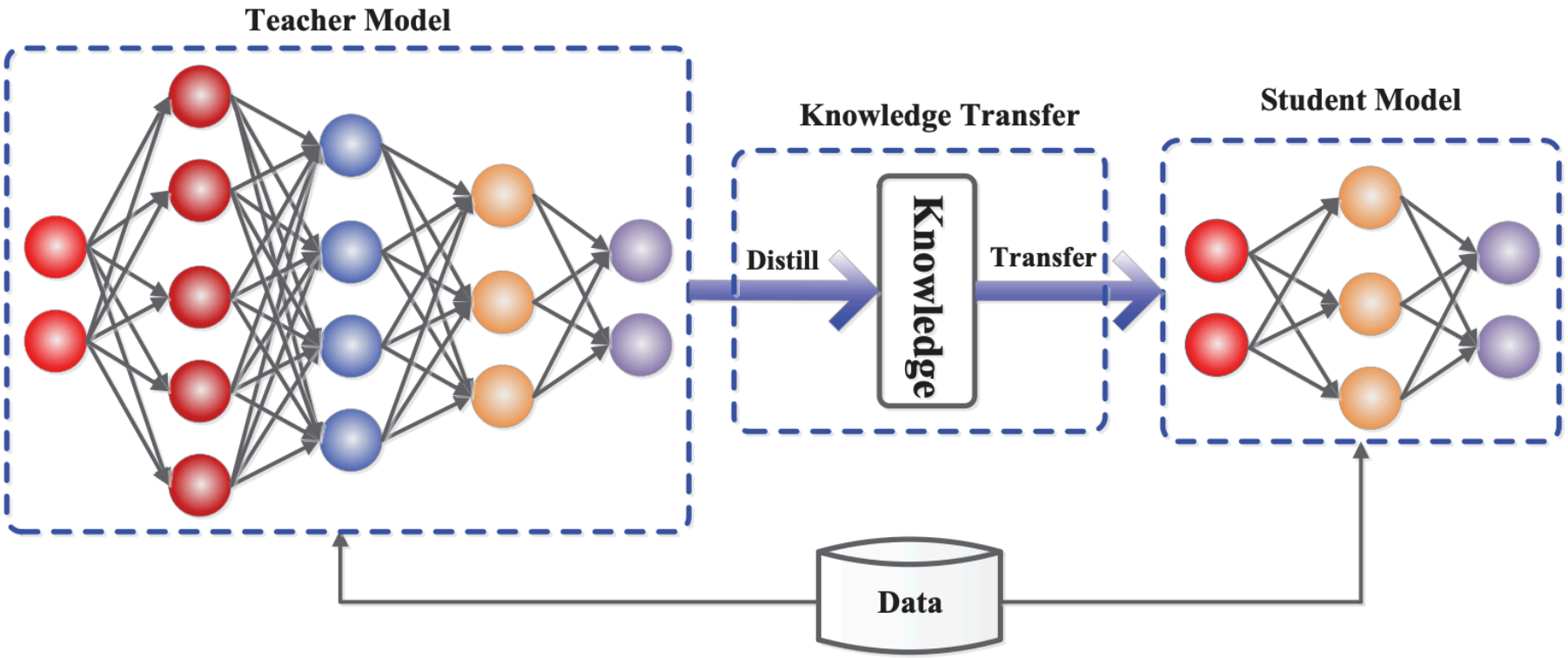


source: "Compression of Deep Learning Models for NLP", Gupta et al. 2020

Knowledge distillation: Student-teacher network

a large cumbersome network

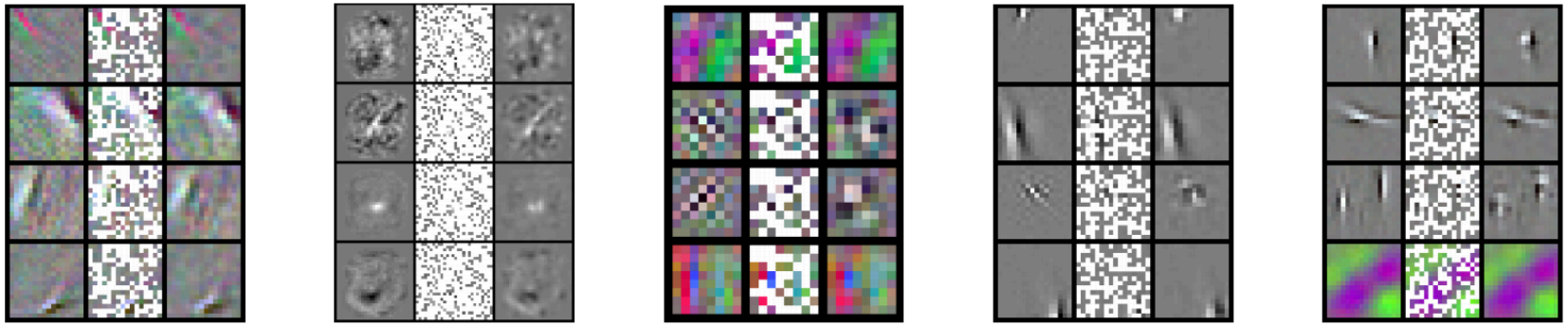
a small network that learns from the teacher



source: "Knowledge Distillation: A Survey", Gou et al. 2019

Parameters are predictable in neural networks

- Significant redundancy in neural network models
- Representing the value of each pixel in the feature separately is redundant
  - Since it is highly likely that the value of a pixel will be equal to a weighted average of its neighbors
  - Predicting 95% parameters w/o loss in accuracy



source: "Predicting parameters in deep learning.", Denil et al. 2013

Knowledge distillation in neural networks

- Previous work [Ba et al. 2014] proposes to learn the student network using L2-loss

$$\mathcal{L}(W, \beta) = \frac{1}{2T} \sum_t ||g(x^{(t)}; W, \beta) - z^{(t)}||_2^2,$$

student network

logits from teacher network

- [Hinton et al. 2015] proposes to 'distill' the knowledge from network instead
  - Step 1: training a cumbersome teacher network
  - Step 2: distillation with a small model
  - Training cumbersome model + distillation is proved better than training a small model directly

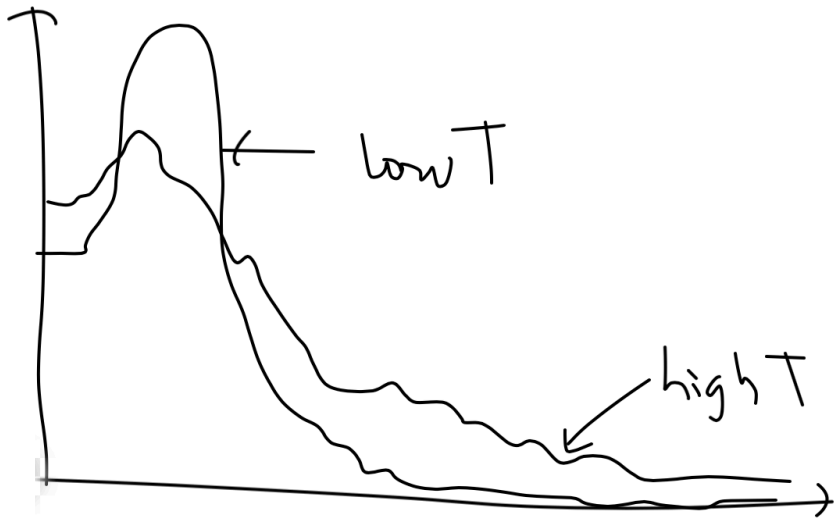
source: "Ditilling the knowledge in a neural network", Hinton et al. 2015

Knowledge distillation in neural networks

- Multi-class classification problem
  - e.g., digit recognition MNIST: 2 is more similar to 3 than 1
- Student learns from soft target in a **transfer set**:
  - qi: student network probability, pi: teacher network probability

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

zi: logits
T: temperature.
High temperature -> a softer probability distribution



source: "Ditilling the knowledge in a neural network", Hinton et al. 2015

Knowledge distillation in neural networks

- Deriving an approximation of cross-entropy gradient:

$$\frac{\partial C}{\partial z_i} = \frac{1}{T}(q_i - p_i) = \frac{1}{T}\left(\frac{e^{z_i/T}}{\sum_j e^{z_j/T}} - \frac{e^{v_i/T}}{\sum_j e^{v_j/T}}\right)$$

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{T}\left(\frac{1 + z_i/T}{N + \sum_j z_j/T} - \frac{1 + v_i/T}{N + \sum_j v_j/T}\right)$$

- If student and teacher logits sums to $\sum_j z_j = \sum_j v_j = 0$:

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{NT^2}(z_i - v_i) \qquad \Leftrightarrow \qquad min\ 1/2(z_i - v_i)^2$$

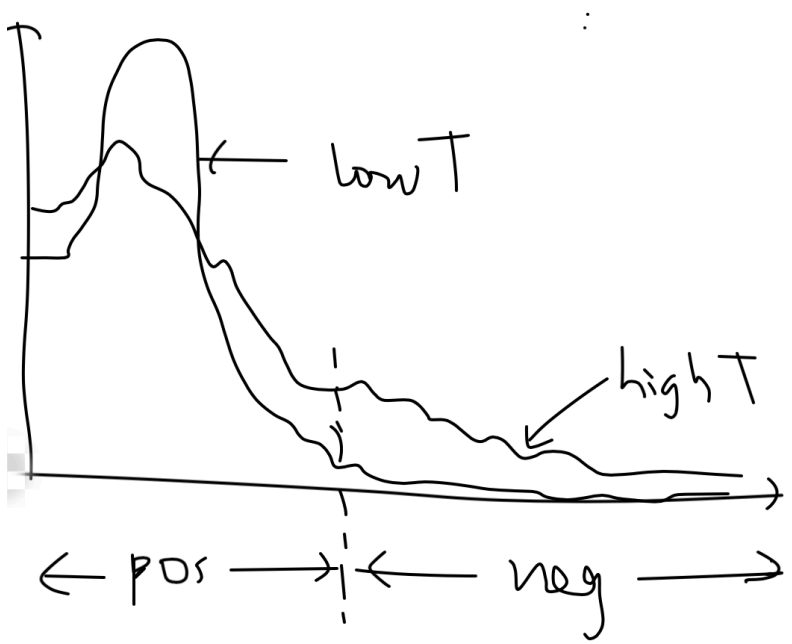
source: "Ditilling the knowledge in a neural network", Hinton et al. 2015

How to set the temperature T?

- High temperature produces a softer distribution
  - Retains knowledge about similarity between classes

- However, high temperature suffer from the high negative value problem
  - Noisy fluctuation
  - Clear distinction between two classes
  - A moderate temperature limits their impact

- Start from low temperature, increase until no gain



source: https://medium.com/@ahmdtaha/distilling-the-knowledge-in-a-neural-network-77b7232bb631

How to set the temperature T?

- **Cumbersome model**
  - 2 hidden layers
  - 1200 ReLU units
  - regularization
  - 67 errors
- **Small model**
  - 2 hidden layers
  - 800 ReLU units
  - no regularization
  - 146 errors

matching logit T=20 → 74 errors

- When the distilled net has > 300 units, all temperatures above 8 gave fairly similar results
- When radically reduced to 30 units, T from 2.5 to 4 works better
- Even if a digit is omitted in the transfer set, the distilled model still learns to recognize it even if the learned bias on the cumbersome model is high

source: "Ditilling the knowledge in a neural network", Hinton et al. 2015

Sobolev Training for neural networks

- In many cases we only have input-output examples, however it is becoming common to have access to derivatives of the target output
  - e.g., knowledge distillation

- Optimizing the network to not only approximate the function's output but also the derivatives
  - Universal approximation theorem [Honik et al. 1991] in the Sobolev space, i.e., metric difference not only defined by value but also derivatives

$$\sum_{i=1}^{N} \ell(m(x_i|\theta), f(x_i))$$

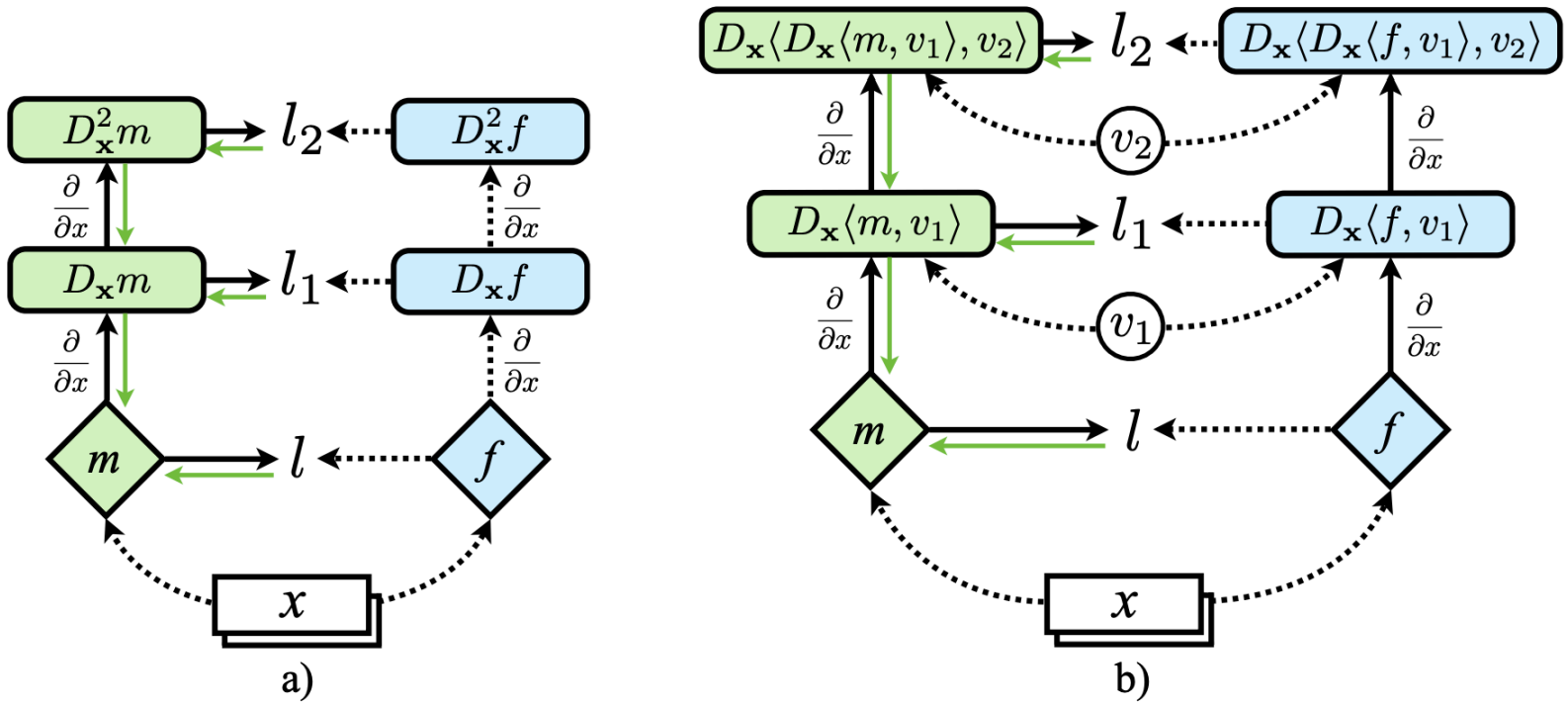
traditional training

$$\sum_{i=1}^{N} \left[\ell(m(x_i|\theta), f(x_i)) + \sum_{j=1}^{K} \ell_j \left(D^j_{\mathbf{x}} m(x_i|\theta), D^j_{\mathbf{x}} f(x_i) \right) \right]$$

Sobolev training

source: "Sobolev Training for Neural Networks", Czarnecki et al. 2015

Sobolev Training for neural networks



Left: Sobolev Training of order 2. Diamond nodes m and f indicate parameterised functions, where m is trained to approximate f. Green nodes receive supervision.

Rght: stochastic Sobolev Training. If f and m are multivariate functions, the gradients are Jacobian matrices. To avoid computing high dimensional objects, compute and fit their projections on a random vector vj

source: "Sobolev Training for Neural Networks", Czarnecki et al. 2015

Mimicing a pretrained LM

- Distillation can help reduce the size of pretrained language models

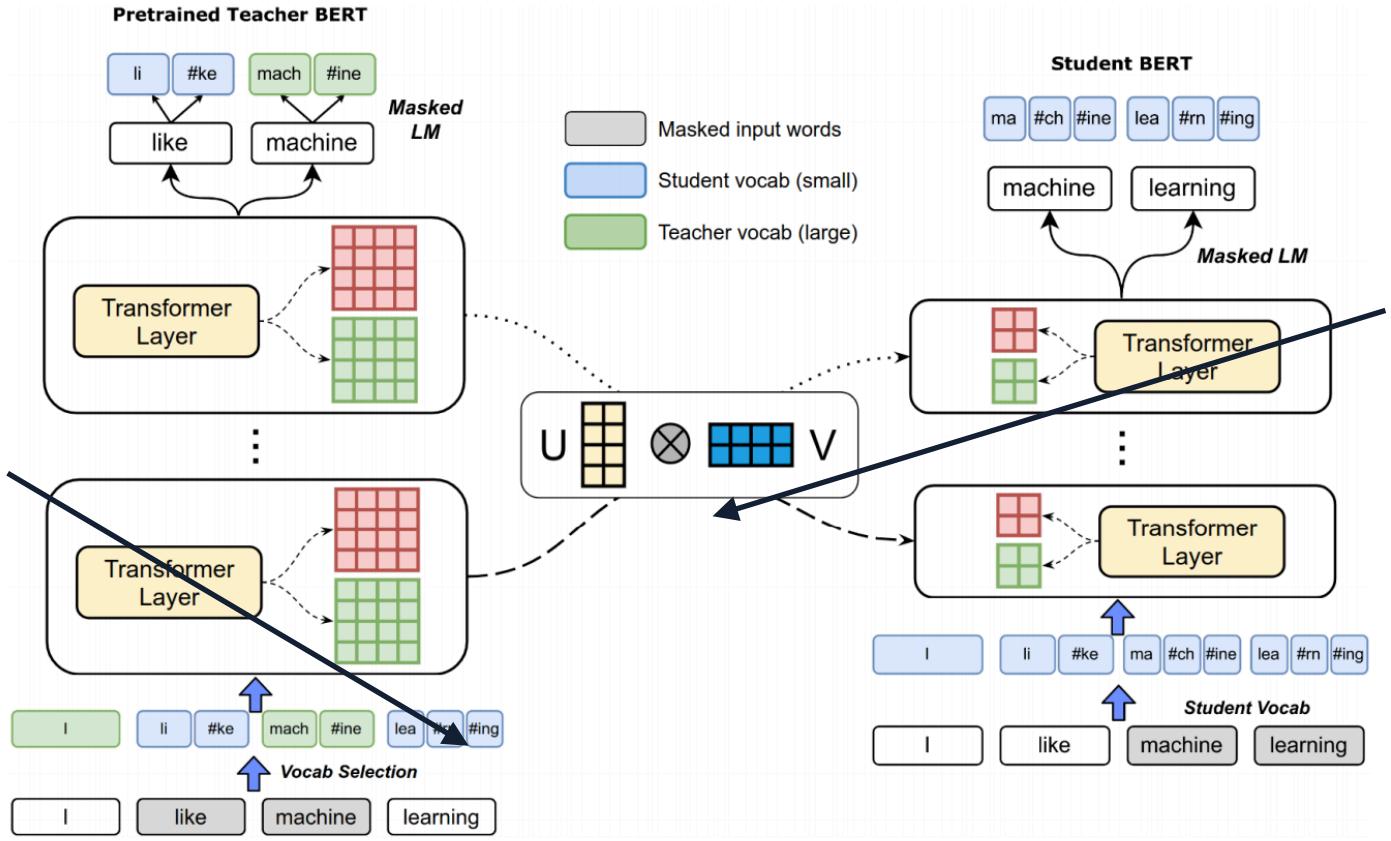
- However, previous work has not considered reducing vocabulary size
  - The word embedding table of bert-base accounts for 21% of the model size, teacher: 30K, student: ~3k

- Can we reduce the size of vocabulary?
  - Poses a challenge, since existing trainings all assume the teacher and student share the same vocabulary
  - Use two strategies: **dual training** and **shared variable projection** to reduce the model to x61 in size

source: "Extreme language model compression with optimal subwords and shared project", Zhao et al. 2020

Mimicing a pretrained LM

- Teacher has a mixture of vocabulary from student and teacher
- Forces the model to learn to predict words from student vocab using teacher vocab, and vice versa



- Projecting model parameters into the same space to encourage alignment

source: "Extreme language model compression with optimal subwords and shared project", Zhao et al. 2020

ALBERT: a lite bert for self-supervised learning of language repre

- Factorized embedding parameterization (horizontal)
  - Decomposing the large vocabulary embedding matrix into two small matrices, makes it easier to grow the hidden size from vocab size

- Cross-layer parameter sharing (vertical)
  - e.g., only sharing FFN parameters, or only sharing attention

- An ALBERT config similar to BERT-large has 18x fewer parameters and can train 1.7x times faster

- Achieving SOTA on GLUE, SQUAD 2.0, and RACE

source: "ALBERT: a lite bert for self-supervised learning of language representations", Lan et al. 2020

ALBERT: a lite bert for self-supervised learning of language repre

- The parameter reduction techniques also act as a form of regularization that stabilizes the training and helps with generalization.

Model		Parameters
BERT	base	108M
	large	334M
ALBERT	base	12M
	large	18M
	xlarge	60M
	xxlarge	235M

400M
1.3G

200M
800M

Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa-large	90.2	94.7	**92.2**	86.6	96.4	**90.9**	68.0	92.4	-	-
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7	-	-
ALBERT (1.5M)	**90.8**	**95.3**	**92.2**	**89.2**	**96.9**	**90.9**	**71.4**	**93.0**	-	-
Ensembles on test (from leaderboard as of Sept. 16, 2019)										
ALICE	88.2	95.7	**90.7**	83.5	95.2	92.6	**69.2**	91.1	80.8	87.0
MT-DNN	87.9	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2	98.6	90.3	86.3	96.8	93.0	67.8	91.6	90.4	88.4
RoBERTa	90.8	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0	88.5
Adv-RoBERTa	91.1	98.8	90.3	88.7	96.8	93.1	68.0	92.4	89.0	88.8
ALBERT	**91.3**	**99.2**	90.5	**89.2**	**97.1**	**93.4**	69.1	**92.5**	**91.8**	**89.4**

source: "ALBERT: a lite bert for self-supervised learning of language representations", Lan et al. 2020

ALBERT: a lite bert for self-supervised learning of language repre

- The effect of different cross-layer parameter sharing strategies:
  - Shared attention has better performance than the other two

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base E=768	all-shared	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8
	shared-attention	83M	89.9/82.7	80.0/77.2	84.0	91.4	67.7	81.6
	shared-FFN	57M	89.2/82.1	78.2/75.4	81.5	90.8	62.6	79.5
	not-shared	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base E=128	all-shared	12M	89.3/82.3	80.0/77.1	82.0	90.3	64.0	80.1
	shared-attention	64M	89.9/82.8	80.7/77.9	83.4	91.9	67.6	81.7
	shared-FFN	38M	88.9/81.6	78.6/75.6	82.3	91.7	64.4	80.2
	not-shared	89M	89.9/82.8	80.3/77.3	83.2	91.5	67.9	81.6

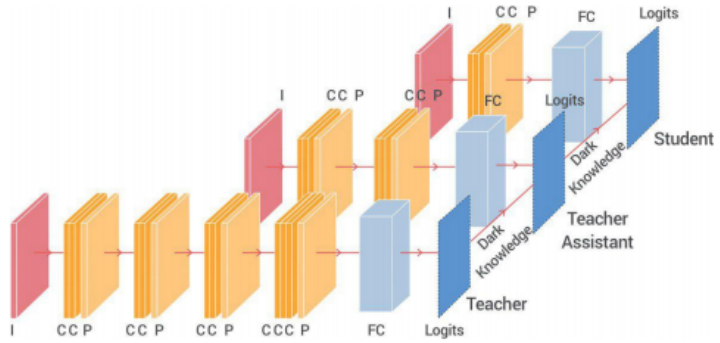
source: "ALBERT: a lite bert for self-supervised learning of language representations", Lan et al. 2020

Knowledge distillation with teacher assistant

- Student network performance degrades when the gap between student and teacher is large

- Given a fixed student network, one cannot employ an arbitrarily large teacher
  - A teacher can effectively transfer its knowledge to students up to a certain size, not smaller

- TA models are distilled from the teacher, and the student is then only distilled from the TAs, i.e., chain distillation



source: "Improved Knowledge Distillation via Teacher Assistant", Mirzadeh et al. 2020

TinyBERT

- TinyBERT achieves >96% the performance of teacher BERT_BASE on GLUE benchmark, while being 7.5x smaller and 9.4x faster on inference.

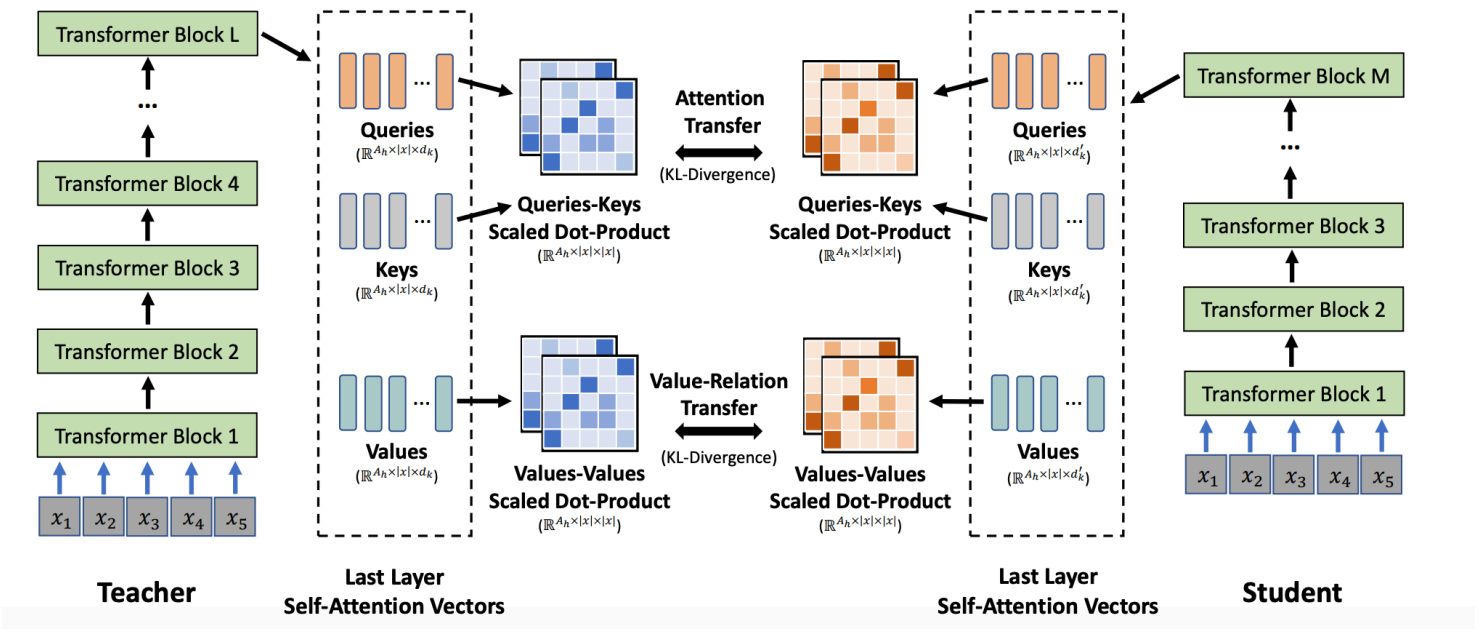
System	#Params	#FLOPs	Speedup	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg
$\text{BERT}_{\text{BASE}}$ (Teacher)	109M	22.5B	1.0x	83.9/83.4	71.1	90.9	93.4	52.8	85.2	87.5	67.0	79.5
$\text{BERT}_{\text{TINY}}$	14.5M	1.2B	9.4x	75.4/74.9	66.5	84.8	87.6	19.5	77.1	83.2	62.6	70.2
$\text{BERT}_{\text{SMALL}}$	29.2M	3.4B	5.7x	77.6/77.0	68.1	86.4	89.7	27.8	77.0	83.4	61.8	72.1
BERT_4-PKD	52.2M	7.6B	3.0x	79.9/79.3	70.2	85.1	89.4	24.8	79.8	82.6	62.3	72.6
DistilBERT_4	52.2M	7.6B	3.0x	78.9/78.0	68.5	85.2	91.4	32.8	76.1	82.4	54.1	71.9
$\text{MobileBERT}_{\text{TINY}}$†	15.1M	3.1B	-	81.5/81.6	68.9	**89.5**	91.7	**46.7**	80.1	**87.9**	65.1	**77.0**
TinyBERT_4 (ours)	14.5M	1.2B	9.4x	**82.5/81.8**	**71.3**	87.7	**92.6**	44.1	**80.4**	86.4	**66.6**	**77.0**
BERT_6-PKD	67.0M	11.3B	2.0x	81.5/81.0	70.7	89.0	92.0	-	-	85.0	65.5	-
PD	67.0M	11.3B	2.0x	82.8/82.2	70.4	88.9	91.8	-	-	86.8	65.3	-
DistilBERT_6	67.0M	11.3B	2.0x	82.6/81.3	70.1	88.9	92.5	49.0	81.3	86.9	58.4	76.8
TinyBERT_6 (ours)	67.0M	11.3B	2.0x	**84.6/83.2**	**71.6**	**90.4**	**93.1**	**51.1**	**83.7**	**87.3**	**70.0**	**79.4**

source: "TinyBERT: Distilling BERT for Natural Language Understanding", Jiao et al. 2020

MiniLM

- Distilling self-attention using the last layer

$$\mathcal{L}_{\text{AT}} = \frac{1}{A_h|x|}\sum_{a=1}^{A_h}\sum_{t=1}^{|x|} D_{KL}(\mathbf{A}_{L,a,t}^{T} \parallel \mathbf{A}_{M,a,t}^{S}) \qquad \mathcal{L}_{\text{VR}} = \frac{1}{A_h|x|}\sum_{a=1}^{A_h}\sum_{t=1}^{|x|} D_{KL}(\mathbf{VR}_{L,a,t}^{T} \parallel \mathbf{VR}_{M,a,t}^{S}) \qquad \mathcal{L} = \mathcal{L}_{\text{AT}} + \mathcal{L}_{\text{VR}}$$



source: "MINILM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers", Wang

MiniLM

- Distilling self-attention using the last layer

$$\mathcal{L}_{\text{AT}} = \frac{1}{A_h|x|}\sum_{a=1}^{A_h}\sum_{t=1}^{|x|} D_{KL}(\mathbf{A}_{L,a,t}^{T} \parallel \mathbf{A}_{M,a,t}^{S}) \qquad \mathcal{L}_{\text{VR}} = \frac{1}{A_h|x|}\sum_{a=1}^{A_h}\sum_{t=1}^{|x|} D_{KL}(\mathbf{VR}_{L,a,t}^{T} \parallel \mathbf{VR}_{M,a,t}^{S}) \qquad \mathcal{L} = \mathcal{L}_{\text{AT}} + \mathcal{L}_{\text{VR}}$$

Model	#Param	SQuAD2	MNLI-m	SST-2	QNLI	CoLA	RTE	MRPC	QQP	Average
$\text{BERT}_{\text{BASE}}$	109M	76.8	84.5	93.2	91.7	58.9	68.6	87.3	91.3	81.5
DistillBERT	66M	70.7	79.0	90.7	85.3	43.6	59.9	87.5	84.9	75.2
TinyBERT	66M	73.1	83.5	91.6	90.5	42.8	**72.2**	**88.4**	90.6	79.1
MINILM	66M	**76.4**	**84.0**	**92.0**	**91.0**	**49.2**	71.5	**88.4**	**91.0**	**80.4**

source: "MINILM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers", Wang

Reformer

- Replace dot−product attention by LSH, O(L^2) -> O(LlogL), where L is the length of the sequence.

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

- **Hashing attention**. We mainly need to compute the QK^T, but note we mainly need to compute the softmax, which is dominated by the largest elements
  - For each query qi only need to focus on keys in K that are closest to qi

- Use reversible residual layers, which allows storing activations only once in the training process instead of N times (N is the number of layers)

source: "REFORMER: THE EFFICIENT TRANSFORMER", Kitaev et al. 2020

Reformer

- Performance of reformer on efficiency and accuracy

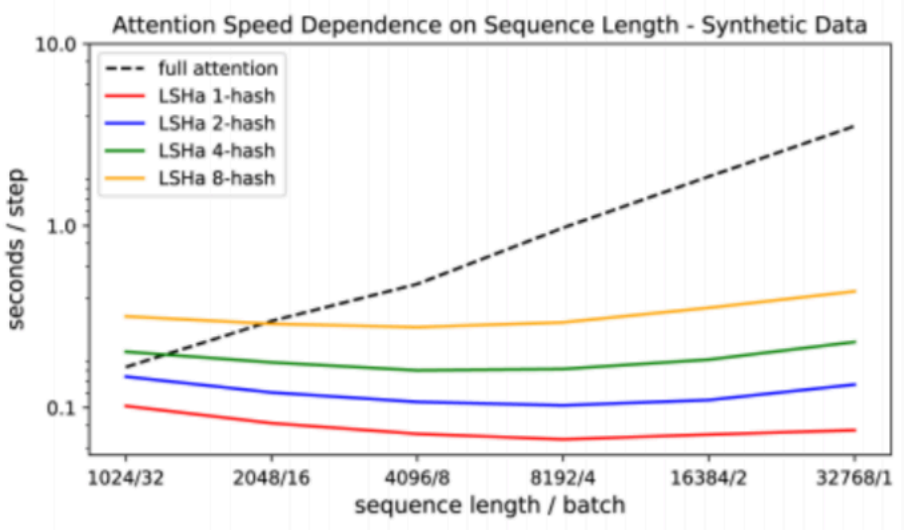


Table 4: BLEU scores on newstest2014 for WMT English-German (EnDe). We additiona
detokenized BLEU scores as computed by sacreBLEU (Post, 2018).

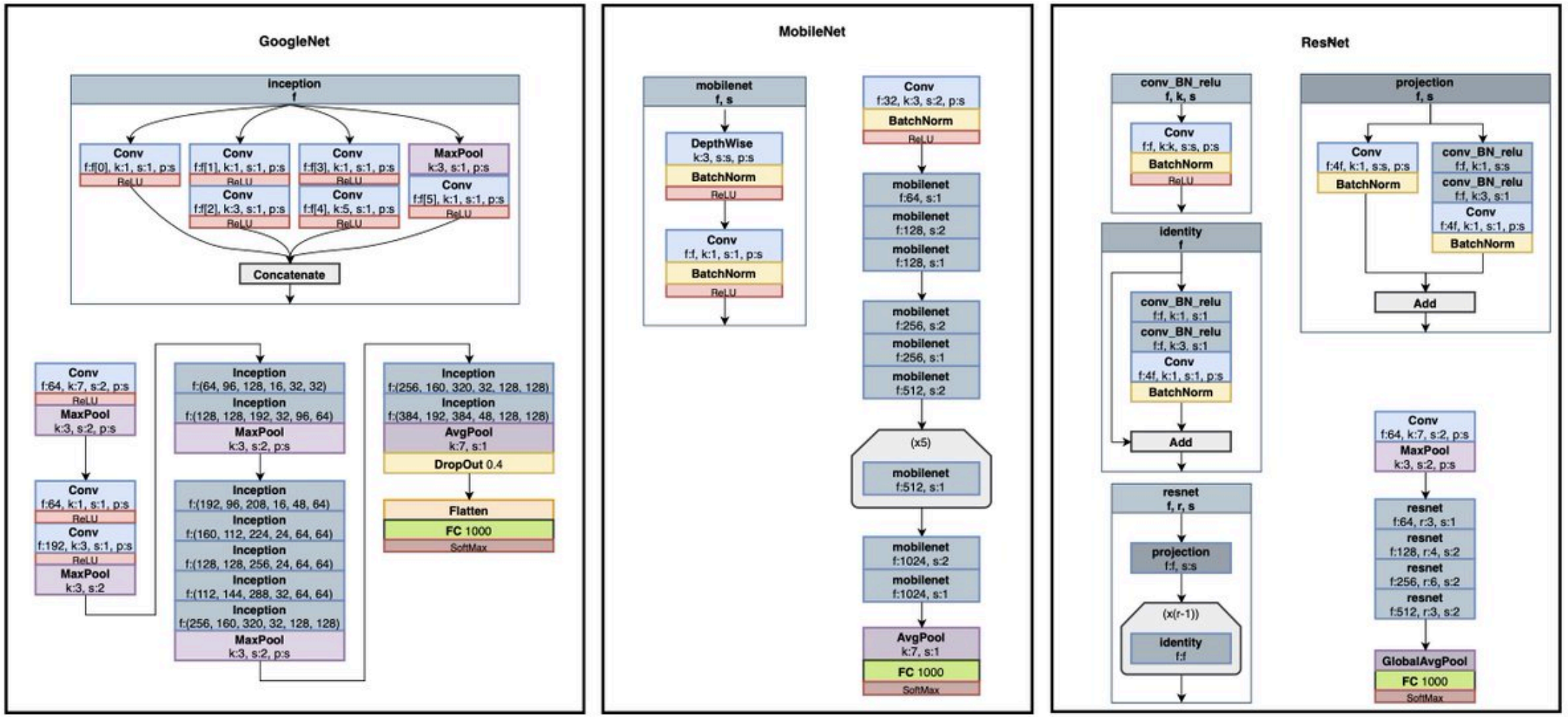
Model	BLEU	*sacreBL Uncased*[3]
Vaswani et al. (2017), base model	27.3	
Vaswani et al. (2017), big	28.4	
Ott et al. (2018), big	29.3	
Reversible Transformer (base, 100K steps)	27.6	*27.4*
Reversible Transformer (base, 500K steps, no weight sharing)	28.0	*27.9*
Reversible Transformer (big, 300K steps, no weight sharing)	29.1	*28.9*

source: "REFORMER: THE EFFICIENT TRANSFORMER", Kitaev et al. 2020

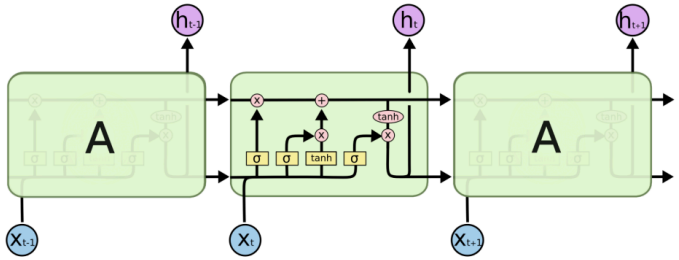
Frontier topics

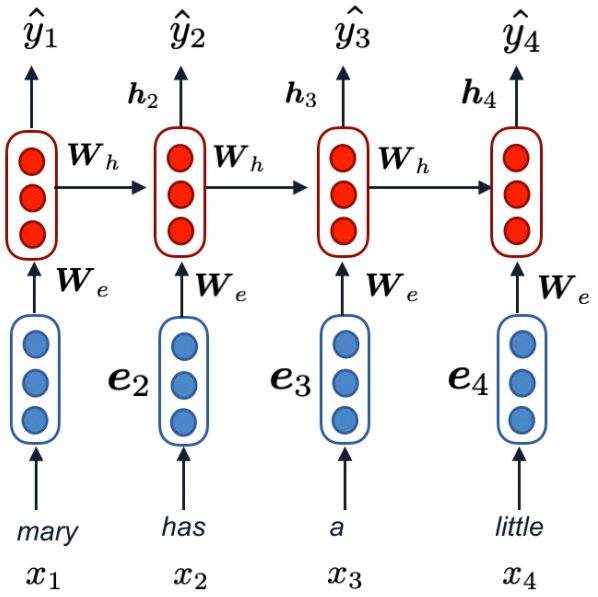
- Neural program synthesis
- Compression for pretrained LMs
- **AutoML; Neural architecture search**
- Fairness/bias in AI
- Machine learning explainability for text mining/NLP

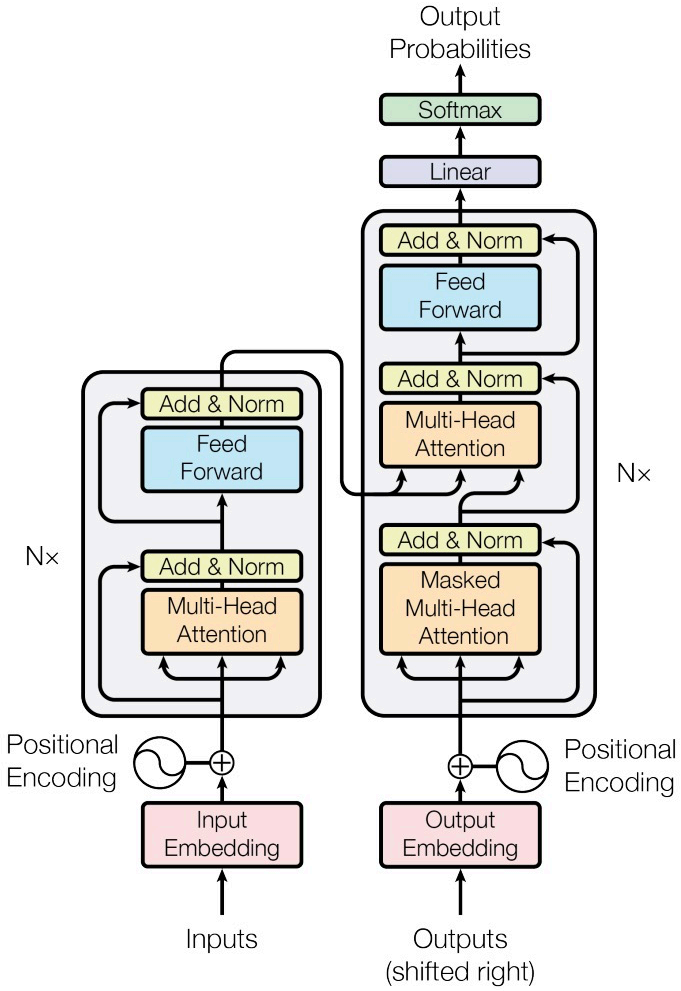
Neural network architectures



Neural network architectures

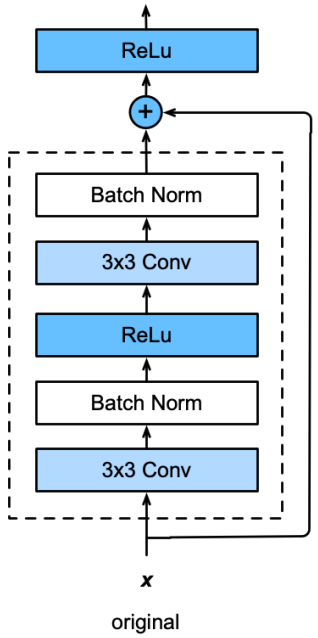






Neural architecture search

- Neural architectures are important in determining model performance
  - Depth and width [Lu et al. 2017]
  - Pooling, convolution, dilated convolution, skip connections [He et al. 2015]



source: "The Expressive Power of Neural Networks: A View from the Width", Lu et al. 2017

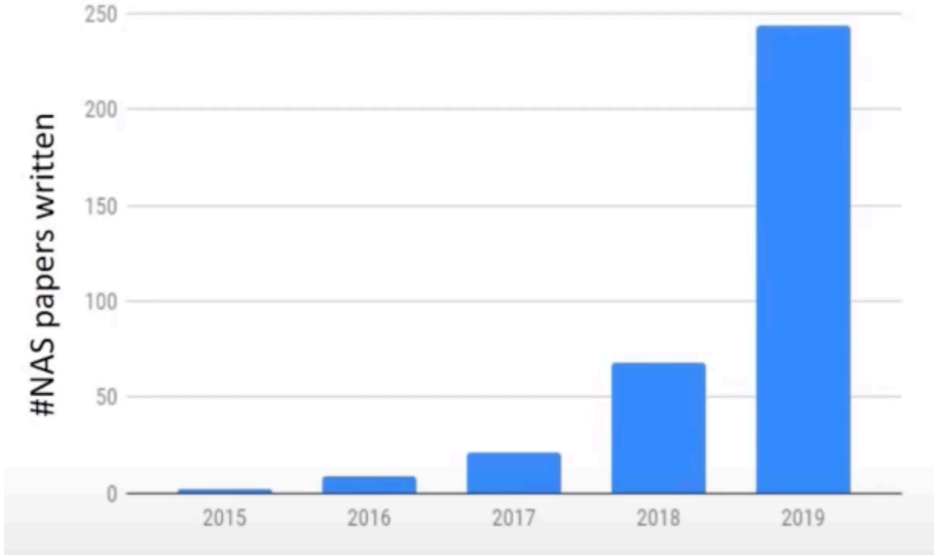
NAS: an exploding area

Neural Architecture Search: A Survey

Thomas Elsken THOMAS.ELSKEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany
and University of Freiburg

Jan Hendrik Metzen JANHENDRIK.METZEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany

Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg
79110 Freiburg, Germany

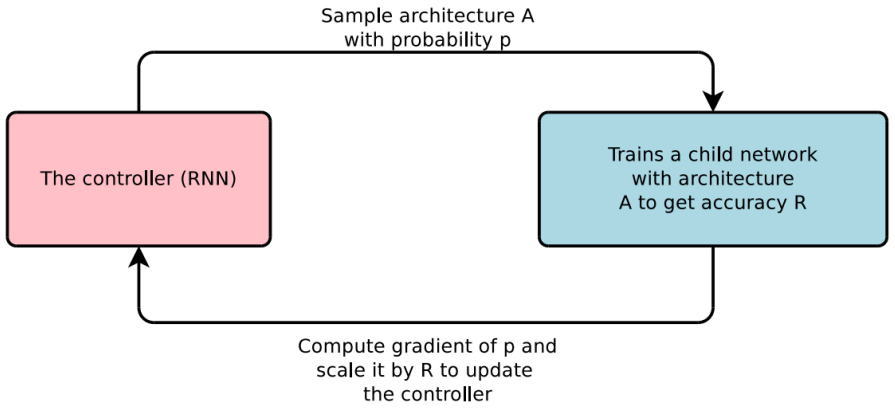


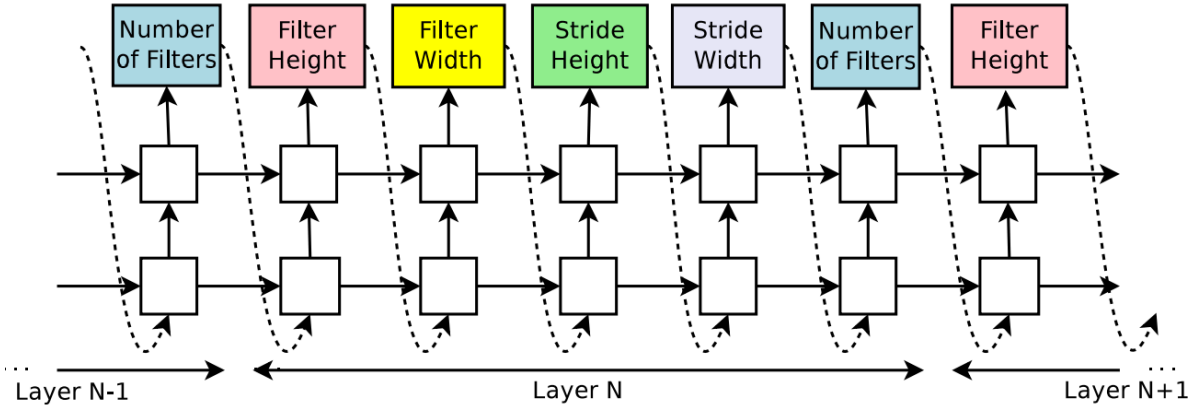
- automl.org
- “Neural architecture search with reinforcement learning” by Zoph & Le

source: Frank Hutter’s talk “Neural architecture search: coming of an age”

Neural architecture search with reinforcement learning

- Use an RNN to generate the neural architecture
- Use test accuracy as reward for controller actions with policy gradient
- On CIFAR-10, starting from scratch, can design a novel network architecture that rivals the best human-invented architecture

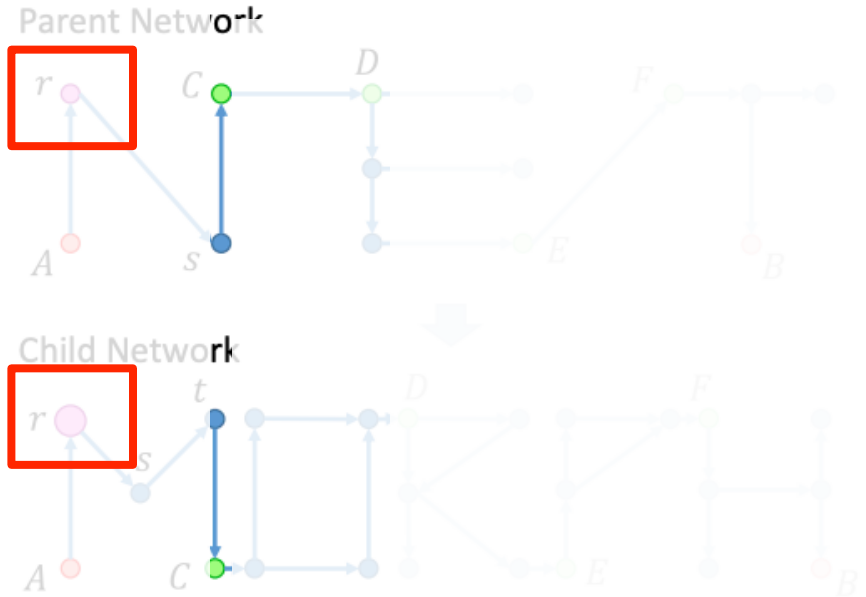




source: "Neural architecture search with reinforcement learning", So et al. 2017

Network morphism

- Morphism:
  - A structure-preserving map from one math structure to another



Type 3: Depth morphing

subset embedded in segment CD

Type 2: kernel size morphing

source: "Network morphism", Wei et al. 2016

Network morphism as matrix factorization

- In a fully connected network:

$$B_{l+1} = G \cdot B_{l-1},$$

- Adding a new layer = matrix factorization

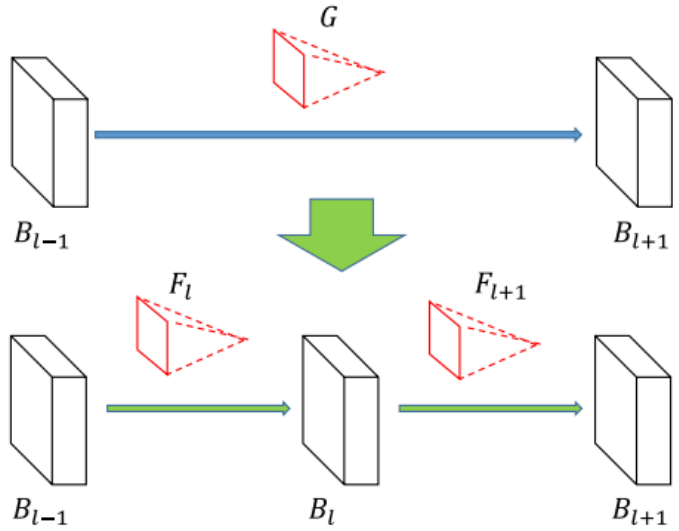
$$B_{l+1} = F_{l+1} \cdot B_l = F_{l+1} \cdot (F_l \cdot B_{l-1}) = G \cdot B_{l-1},$$

$$G = F_{l+1} \cdot F_l.$$

- In a convolutional network, similarly:

$$\tilde{G} = F_{l+1} \circledast F_l,$$

parent network



child network

source: "Network morphism", Wei et al. 2016

Morphism algorithms: Solving matrix factorization for linear case

- Desiderata for morphism algorithms
  - Parameters contain as many 0 as possible
  - Parameters need to be in a consistent scale

- Introduces two algorithms for morphism:
  - Initialize matrices with random noise
  - Iteratively perform the decomposition/ deconvolution for one layer with another layer fixed

Algorithm 1 General Network Morphism

Input: G of shape (C_{l+1}, C_{l-1}, K, K); C_l, K_1, K_2
Output: F_l of shape (C_l, C_{l-1}, K_1, K_1), F_{l+1} of shape (C_{l+1}, C_l, K_2, K_2)
Initialize F_l and F_{l+1} with random noise.
Expand G to $\tilde{G}$ with kernel size $\tilde{K} = K_1 + K_2 - 1$ by padding zeros.
repeat
  Fix F_l, and calculate $F_{l+1} = deconv(\tilde{G}, F_l)$
  Fix F_{l+1}, and calculate $F_l = deconv(\tilde{G}, F_{l+1})$
  Calculate loss $l = \|\tilde{G} - conv(F_l, F_{l+1})\|^2$
until $l = 0$ or $maxIter$ is reached
Normalize F_l and F_{l+1} with equal standard variance.

source: "Network morphism", Wei et al. 2016

Morphism algorithm: Handling non-linear case

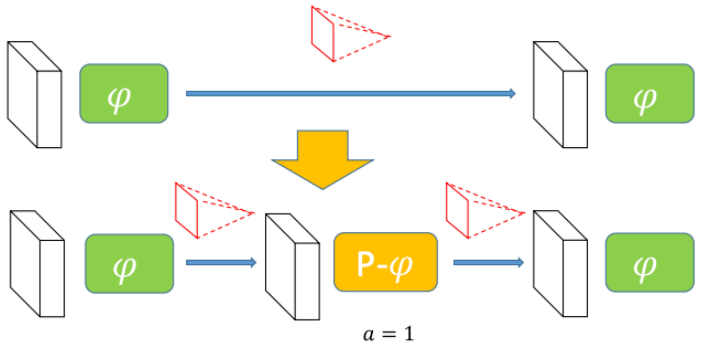
- With a non-linear activation function ϕ:

$$\varphi(I \circledast \varphi(G \circledast B_{l-1}) = \varphi \circ \varphi(G \circledast B_{l+1}) = \varphi(G \circledast B_{l+1}).$$

- Define a family of P-activation function to handle arbitrary continuous non-linear activation ϕ_{id} is the identity mapping:

$$P\text{-}\varphi \triangleq \{\varphi^a\}|_{a\in[0,1]} = \{(1-a)\cdot\varphi + a\cdot\varphi_{id}\}|_{a\in[0,1]}$$

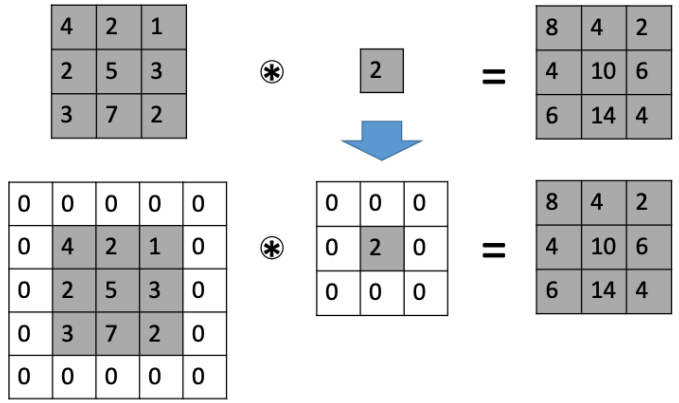
- Activations in yellow needs to be set as linear at the beginning, then grow into a non-linear

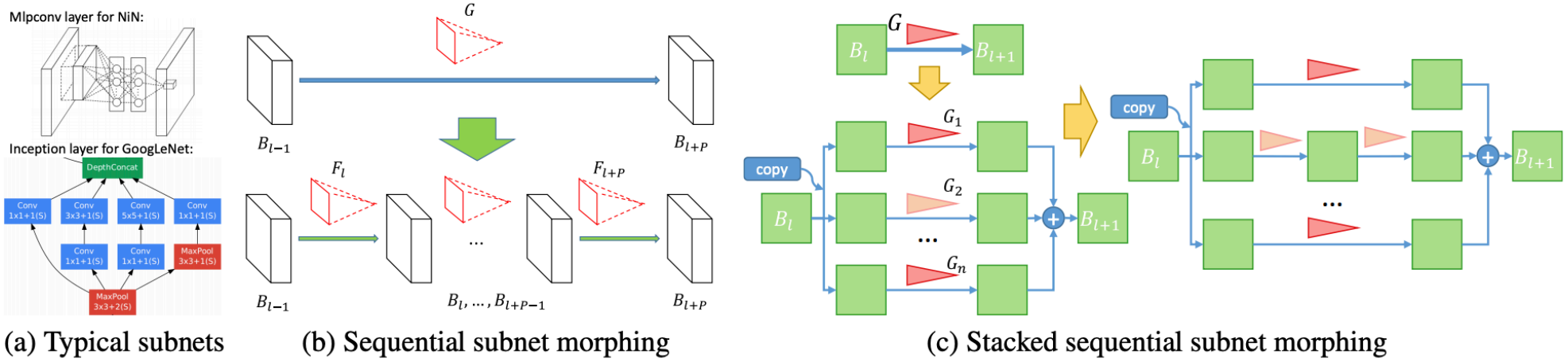


source: "Network morphism", Wei et al. 2016

Morphism algorithm: Other morphing

- Kernel size morphing
  - Both filters and blobs are padded with the same size of zeros to keep the final results unchanged
- Subnet morphing

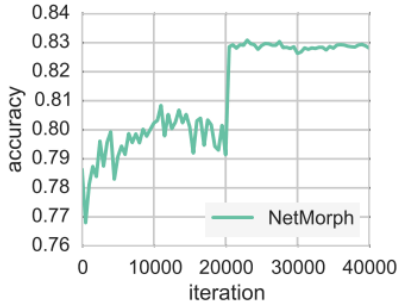




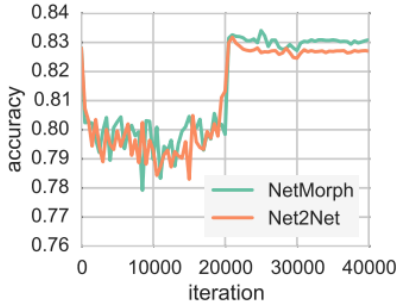
(a) Typical subnets (b) Sequential subnet morphing (c) Stacked sequential subnet morphing

source: "Network morphism", Wei et al. 2016

Morphism algorithm: Performance

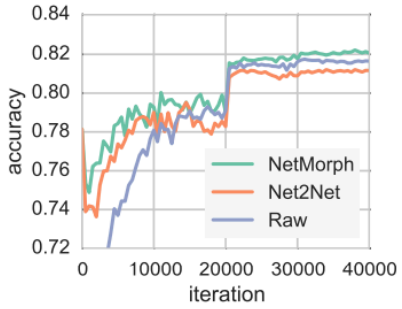


(a) Kernel size morphing

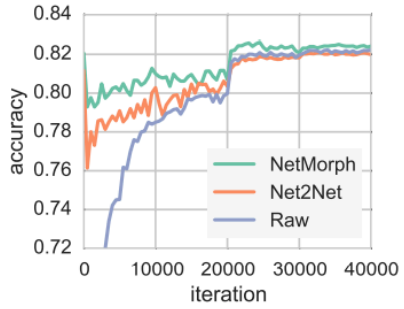


(b) Width morphing

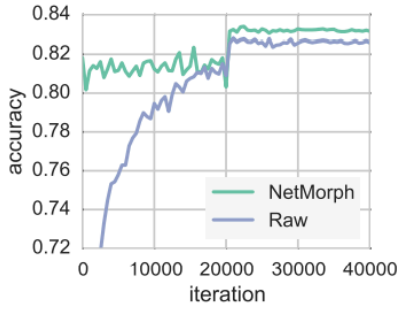
	Top-1 1-view	Top-5 1-view	Top-1 10-view	Top-5 10-view
VGG16 (multi-scale)	68.35%	88.45%	69.59%	89.02%
VGG19 (multi-scale)	68.48%	88.44%	69.44%	89.21%
VGG16 (baseline)	67.30%	88.31%	68.64%	89.10%
VGG16 (NetMorph)	**69.14%**	**89.00%**	**70.32%**	**89.86%**



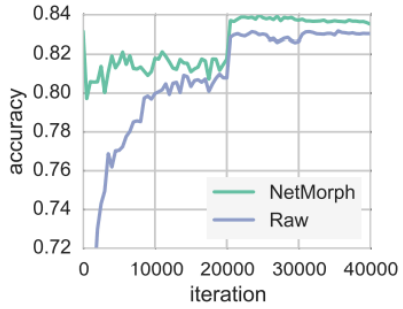
(a) cifar_111 → 211



(b) cifar_211 → 222



(c) cifar_222 → 2222



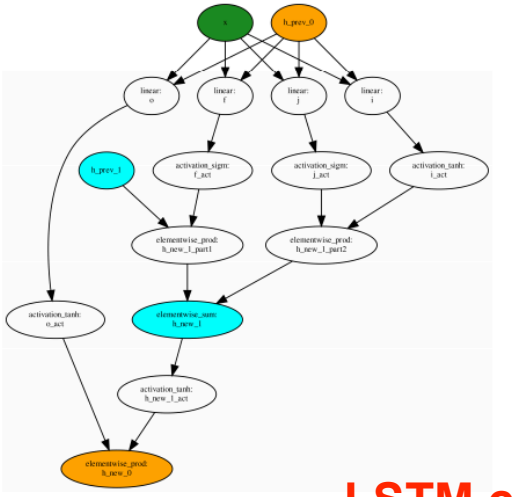
(d) cifar_2222 → 3333

Figure 8: Depth morphing and subnet morphing on CIFAR10.

source: "Network morphism", Wei et al. 2016

DARTS: differentiable architecture search

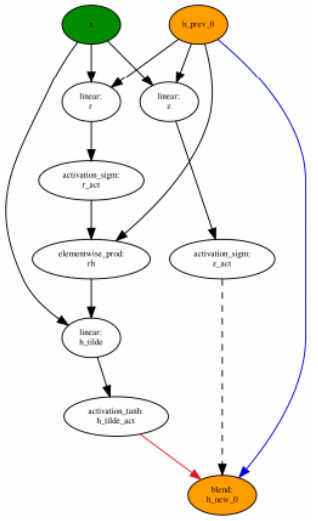
- Search space
  - The inner structure of a cell
  - Cell can be stacked into a CNN or recursively connected to an RNN



LSTM cell

- Each intermediate node is computed based on all its predecessors

$$x^{(j)} = \sum_{i<j} o^{(i,j)}(x^{(i)})$$



GRU cell

source: "DARTS: differentiable architecture search", Liu et al. 2019

DARTS: differentiable architecture search

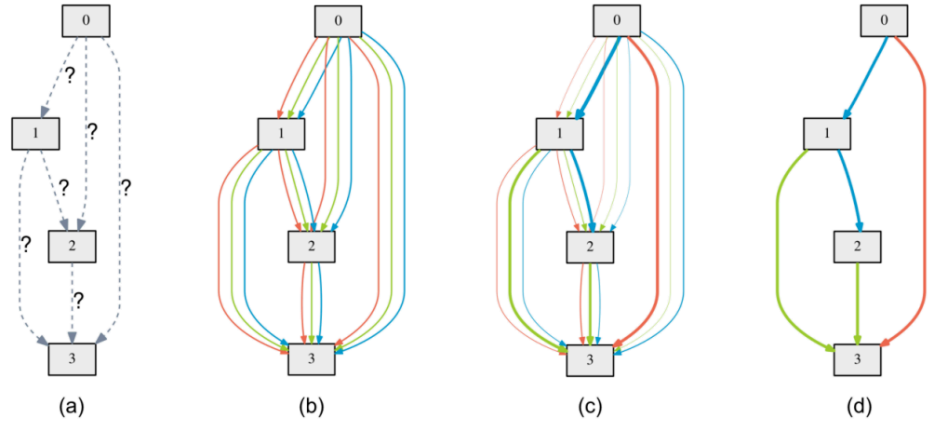
- Relax the **discrete problem** -> continuous
  - One-shot model with continuous architecture weight α (mixing weight) for each operator:

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

  - Solving a bi-level optimization problem

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{val}(w^*(\alpha), \alpha) \\ \text{s.t.} \quad & w^*(\alpha) = \text{argmin}_w \ \mathcal{L}_{train}(w, \alpha) \end{aligned}$$

5x5 col 3x3 col maxpooling



Iterative updating alpha and w using gradient based methods

source: "DARTS: differentiable architecture search", Liu et al. 2019

Auto-Pytorch: Joint hyperparameter optimization and NAS

- Auto-pytorch-tabular: making deep learning easily accessible to tabular dataset for Pytorch

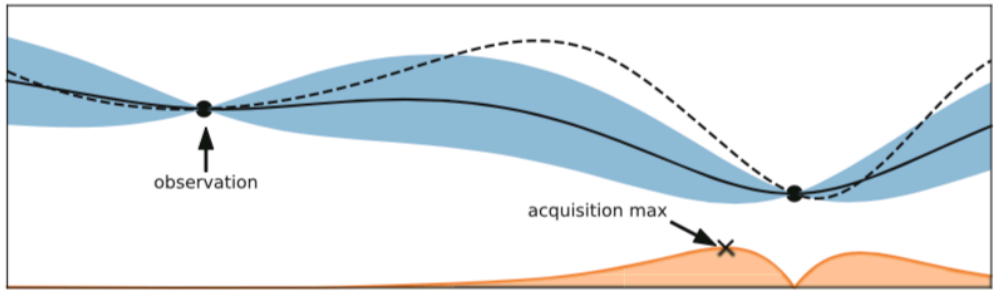
- Search space
  - 7 hyper parameters
  - shaped MLP + shaped ResNet

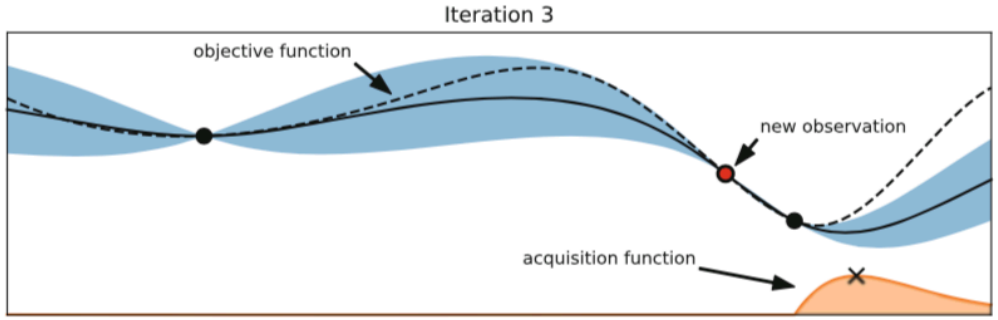
	Name	Range	log	type	cond.
Architecture	network type	[ResNet, MLPNet]	-	cat	-
	num layers (MLP)	[1, 6]	-	int	✓
	max units (MLP)	[64, 1024]	✓	int	✓
	max dropout (MLP)	[0, 1]	-	float	✓
	num groups (Res)	[1, 5]	-	int	✓
	blocks per group (Res)	[1, 3]	-	int	✓
	max units (Res)	[32, 512]	✓	int	✓
	use dropout (Res)	[F, T]	-	bool	✓
	use shake drop	[F, T]	-	bool	✓
	use shake shake	[F, T]	-	bool	✓
	max dropout (Res)	[0, 1]	-	float	✓
	max shake drop (Res)	[0, 1]	-	float	✓
Hyperparameters	batch size	[16, 512]	✓	int	-
	optimizer	[SGD, Adam]	-	cat	-
	learning rate (SGD)	[1e-4, 1e-1]	✓	float	✓
	L2 reg. (SGD)	[1e-5, 1e-1]	-	float	✓
	momentum	[0.1, 0.999]	-	float	✓
	learning rate (Adam)	[1e-4, 1e-1]	✓	float	✓
	L2 reg. (Adam)	[1e-5, 1e-1]	-	float	✓
	training technique	[standard, mixup]	-	cat	-
	mixup alpha	[0, 1]	-	float	✓
	preprocessor	[none, trunc. SVD]	-	cat	-
	SVD target dim	[10, 256]	-	int	✓

source: "Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL", Zimmer et al. 2020

Bayesian optimization [Bergstra et al. 2012]

- Approach: fix a probabilistic model to the function evaluation
- Use that model for exploration vs. exploitation
- During the development of AlphaGO, many hyper parameters were tuned with BO many times, e.g., before the match with Sedol Lee, HPO improved the win-rate from 50% -> 65%





SMBO(f, M_0, T, S)

1 $\mathcal{H} \leftarrow \emptyset$,
2 For $t \leftarrow 1$ **to** T,
3 $x^* \leftarrow \text{argmin}_x S(x, M_{t-1})$,
4 Evaluate $f(x^*)$, ▷ *Expensive step*
5 $\mathcal{H} \leftarrow \mathcal{H} \cup (x^*, f(x^*))$,
6 Fit a new model M_t to $\mathcal{H}$.
7 **return** $\mathcal{H}$

acquisition function

source: Automatic Machine Learning (NeurIPS 2018 Tutorial). Frank Hutter and Joaquin Vanschoren

Auto-Keras

- Proposes to use Bayesian optimization for neural architecture search
- Challenge applying BO to NAS as BO is designed only for the Euclidean space
  - If BO samples an architecture, it may cause input shape inconsistency
- Proposes an edit-distance neural network kernel

$$\kappa(f_a, f_b) = e^{-\rho^2(d(f_a, f_b))},$$

- Use dynamic programming-based morphing:

$$A_{i,j} = max[A_{i-1,j} + 1, A_{i,j-1} + 1, A_{i-1,j-1} + d_l(l_a^{(i)}, l_b^{(j)})],$$

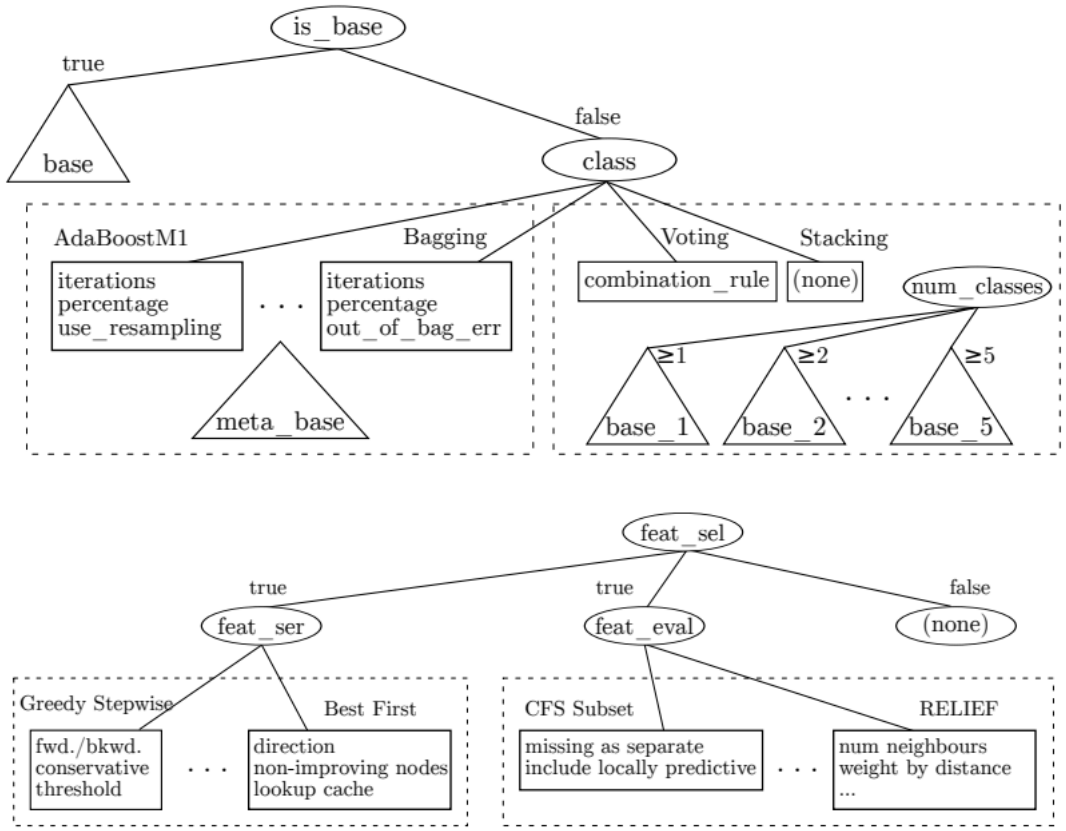
source: "Auto-Keras: An Efficient Neural Architecture Search System", Jin et al. 2019

Auto-Weka

- Combined algorithm selection and hyperparameter optimization problem (CASH) as computing

$$A^*_{\boldsymbol{\lambda}^*} \in \underset{A^{(j)} \in \mathcal{A}, \boldsymbol{\lambda} \in \boldsymbol{\Lambda}^{(j)}}{\operatorname{argmin}} \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(A^{(j)}_{\boldsymbol{\lambda}}, \mathcal{D}^{(i)}_{\text{train}}, \mathcal{D}^{(i)}_{\text{valid}})$$

- Uses two hyperparameter optimization algorithm that can handle hyperparameter search in a hierarchical space
  - Sequential model-based configuration
  - Tree-structured Parzen estimator



source: "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms", Thornton et al. 2013