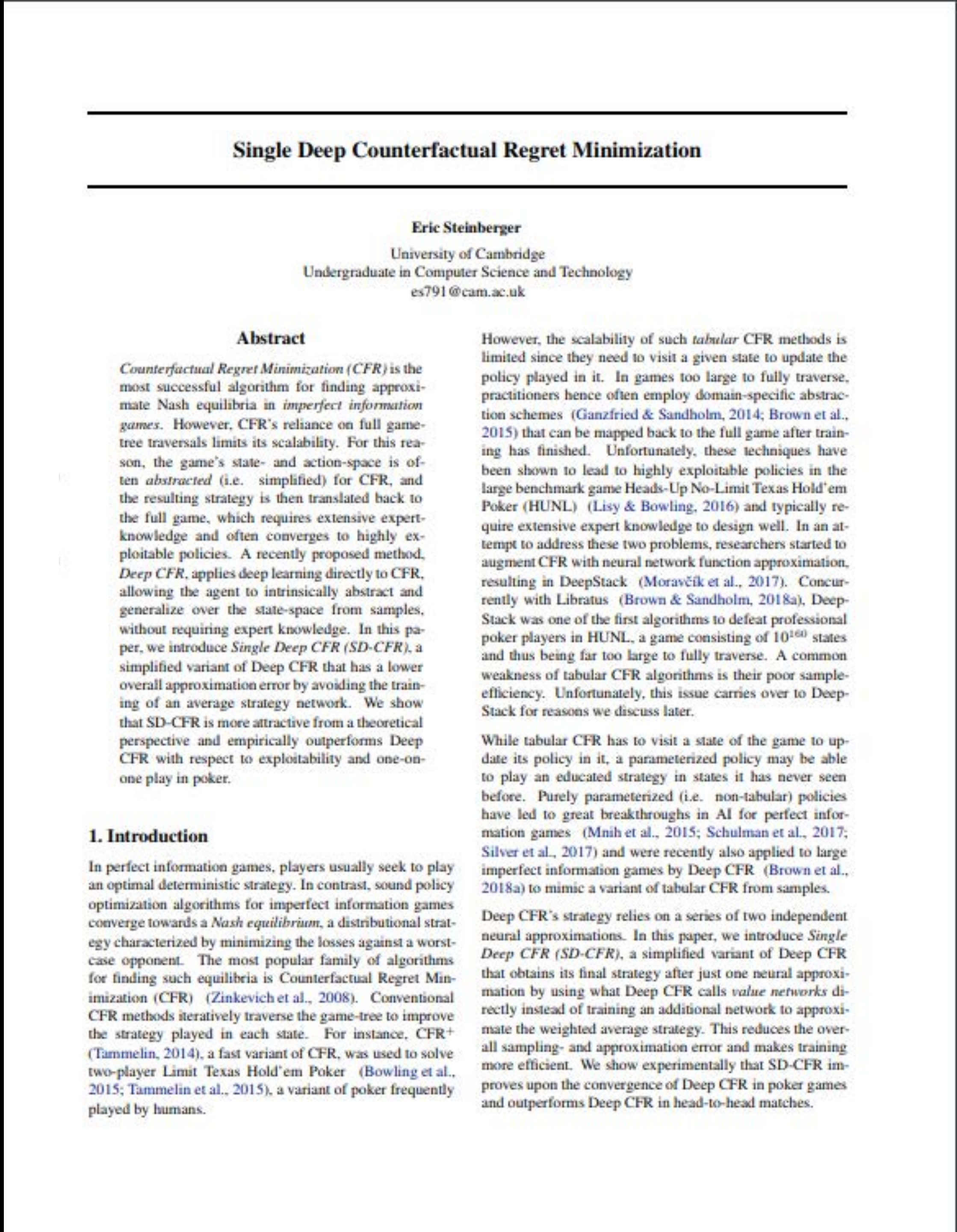
Improving Supervised Deep Learning with Unsupervised Learning

Eric Steinberger

MIT Europe Conference 2019

Single Deep Counterfactual Regret Minimization (SD-CFR)

- Find Nash Equilibria in humongous imperfect information games
- Use Deep Neural Networks to approximate Counterfactual Regret Minimization (CFR) to sample only a part of the game tree.
- Performs better and trains faster with less memory than Deep CFR (Brown et al.)



Single Deep Counterfactual Regret Minimization

Eric Steinberger
University of Cambridge
Undergraduate in Computer Science and Technology
es791@cam.ac.uk

Abstract

Counterfactual Regret Minimization (CFR) is the most successful algorithm for finding approximate Nash equilibria in *imperfect information games*. However, CFR's reliance on full game-tree traversals limits its scalability. For this reason, the game's state- and action-space is often *abstracted* (i.e. simplified) for CFR, and the resulting strategy is then translated back to the full game, which requires extensive expert-knowledge and often converges to highly exploitable policies. A recently proposed method, *Deep CFR*, applies deep learning directly to CFR, allowing the agent to intrinsically abstract and generalize over the state-space from samples, without requiring expert knowledge. In this paper, we introduce *Single Deep CFR (SD-CFR)*, a simplified variant of Deep CFR that has a lower overall approximation error by avoiding the training of an average strategy network. We show that SD-CFR is more attractive from a theoretical perspective and empirically outperforms Deep CFR with respect to exploitability and one-on-one play in poker.

1. Introduction

In perfect information games, players usually seek to play an optimal deterministic strategy. In contrast, sound policy optimization algorithms for imperfect information games converge towards a *Nash equilibrium*, a distributional strategy characterized by minimizing the losses against a worst-case opponent. The most popular family of algorithms for finding such equilibria is Counterfactual Regret Minimization (CFR) (Zinkevich et al., 2008). Conventional CFR methods iteratively traverse the game-tree to improve the strategy played in each state. For instance, CFR^+ (Tammelin, 2014), a fast variant of CFR, was used to solve two-player Limit Texas Hold'em Poker (Bowling et al., 2015; Tammelin et al., 2015), a variant of poker frequently played by humans.

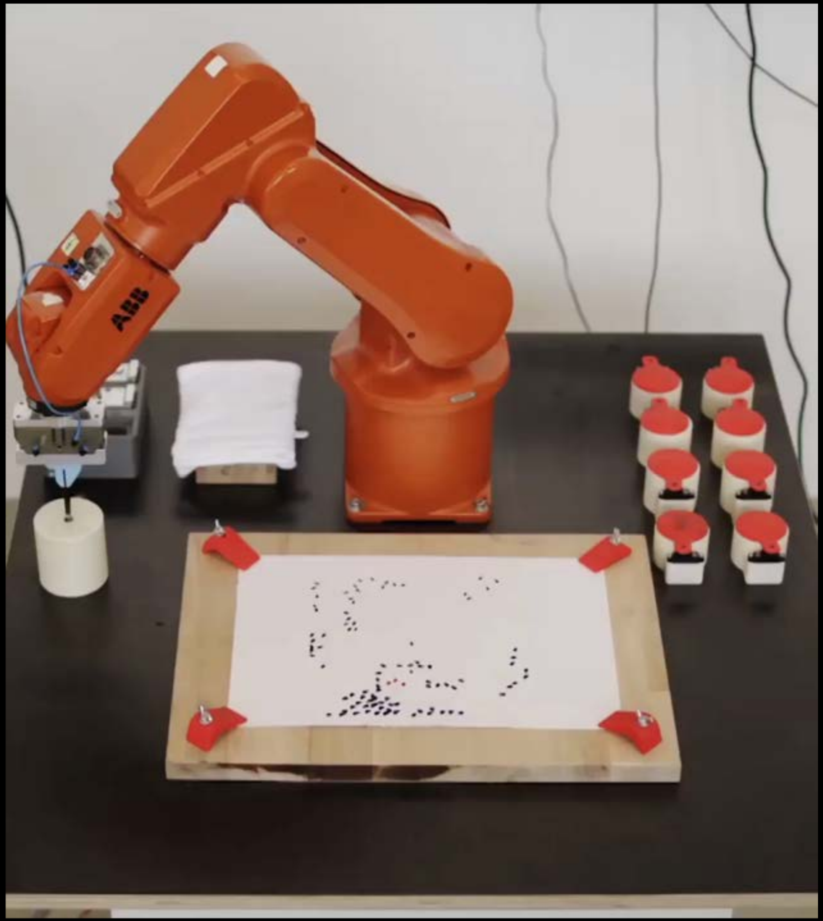
However, the scalability of such *tabular* CFR methods is limited since they need to visit a given state to update the policy played in it. In games too large to fully traverse, practitioners hence often employ domain-specific abstraction schemes (Ganzfried & Sandholm, 2014; Brown et al., 2015) that can be mapped back to the full game after training has finished. Unfortunately, these techniques have been shown to lead to highly exploitable policies in the large benchmark game Heads-Up No-Limit Texas Hold'em Poker (HUNL) (Lisy & Bowling, 2016) and typically require extensive expert knowledge to design well. In an attempt to address these two problems, researchers started to augment CFR with neural network function approximation, resulting in DeepStack (Moravčík et al., 2017). Concurrently with Libratus (Brown & Sandholm, 2018a), DeepStack was one of the first algorithms to defeat professional poker players in HUNL, a game consisting of 10^{160} states and thus being far too large to fully traverse. A common weakness of tabular CFR algorithms is their poor sample-efficiency. Unfortunately, this issue carries over to DeepStack for reasons we discuss later.

While tabular CFR has to visit a state of the game to update its policy in it, a parameterized policy may be able to play an educated strategy in states it has never seen before. Purely parameterized (i.e. non-tabular) policies have led to great breakthroughs in AI for perfect information games (Mnih et al., 2015; Schulman et al., 2017; Silver et al., 2017) and were recently also applied to large imperfect information games by Deep CFR (Brown et al., 2018a) to mimic a variant of tabular CFR from samples.

Deep CFR's strategy relies on a series of two independent neural approximations. In this paper, we introduce *Single Deep CFR (SD-CFR)*, a simplified variant of Deep CFR that obtains its final strategy after just one neural approximation by using what Deep CFR calls *value networks* directly instead of training an additional network to approximate the weighted average strategy. This reduces the overall sampling- and approximation error and makes training more efficient. We show experimentally that SD-CFR improves upon the convergence of Deep CFR in poker games and outperforms Deep CFR in head-to-head matches.

smART

- See: github.com/TinkeringCode/smart
- Team:
  - Eric Steinberger - *Implementation of Style Transfer, ISS, stroke GA, and the RAPID translator*
  - Patrick Pelzmann (TU Vienna) - *Colour management & brush cleaning machines, hacking the ABB robot*
  - Benjamin Mörzinger (TU Vienna) - *Physical setup, management, supervision*
  - Manuel Stadler (TU Vienna) - *Helped setting up the ABB robot arm*
  - Alexander Raschendorfer (TU Vienna) - *Brush swapping mechanism*
  - Ralph Oswald (TU Vienna) - *Visualization of robot monitoring logs*



Industry work

- Employee scheduling algorithm
- Predictive systems for various industries
- Domain-specific language prediction
- Concept work on matching algorithms
- Talks/workshops for companies

Motivation

Your data:

- Input: Some text or images
- Model should make some classification
- You have only 1000 examples :C

But wait…

- All books ever written
- Wikipedia
- Huge open-source images datasets
- Do we need more? There is more.

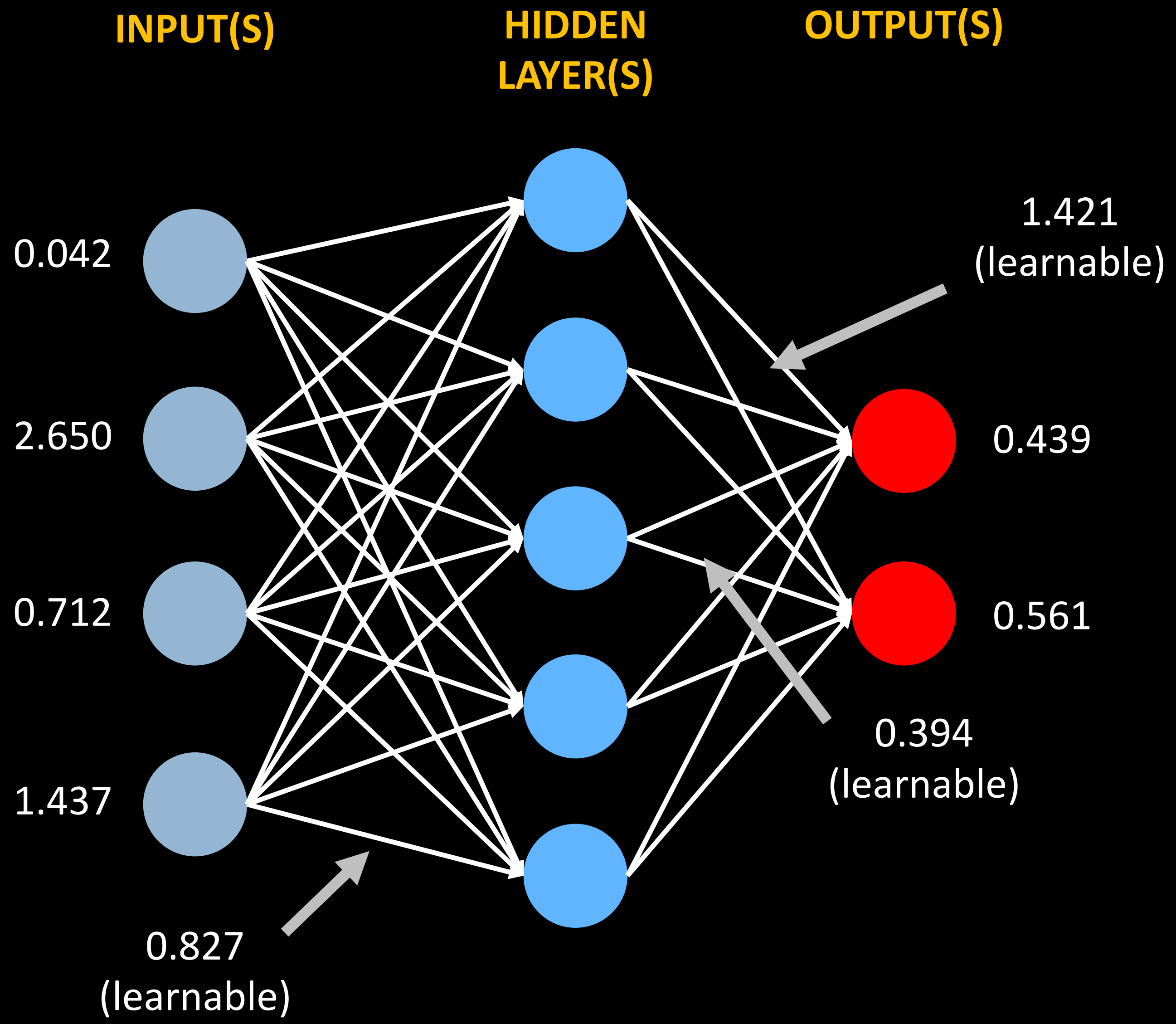
Humans vs. Supervised ML

How humans are trained:

- Years of seeing, reading, speaking
- Supervised Training
  - Parents, schools, books, …
- Millions of years of evolution

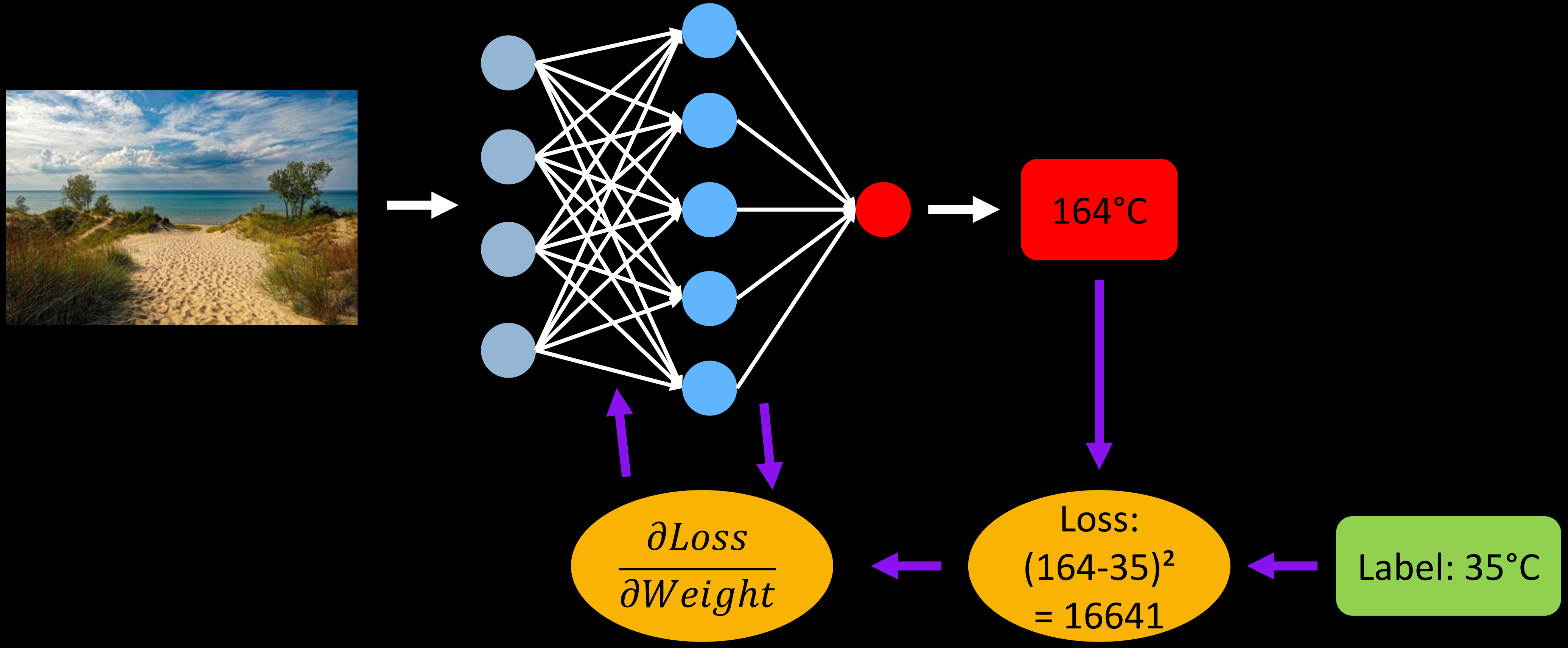
How AI is conventionally trained:

- The target dataset
- Some hyperparameter tuning

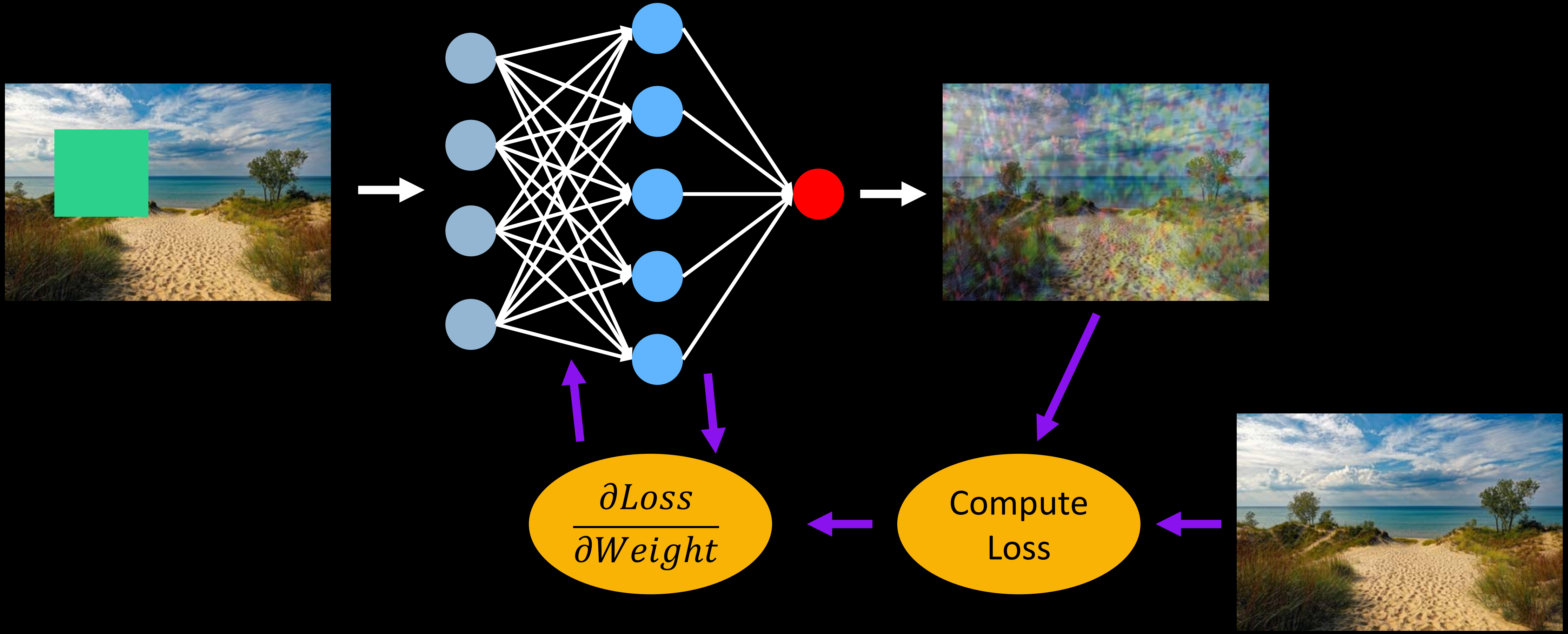

INPUT(S)
HIDDEN LAYER(S)
OUTPUT(S)
0.042
2.650
0.712
1.437
1.421 (learnable)
0.439
0.561
0.394 (learnable)
0.827 (learnable)

Inputs with Labels

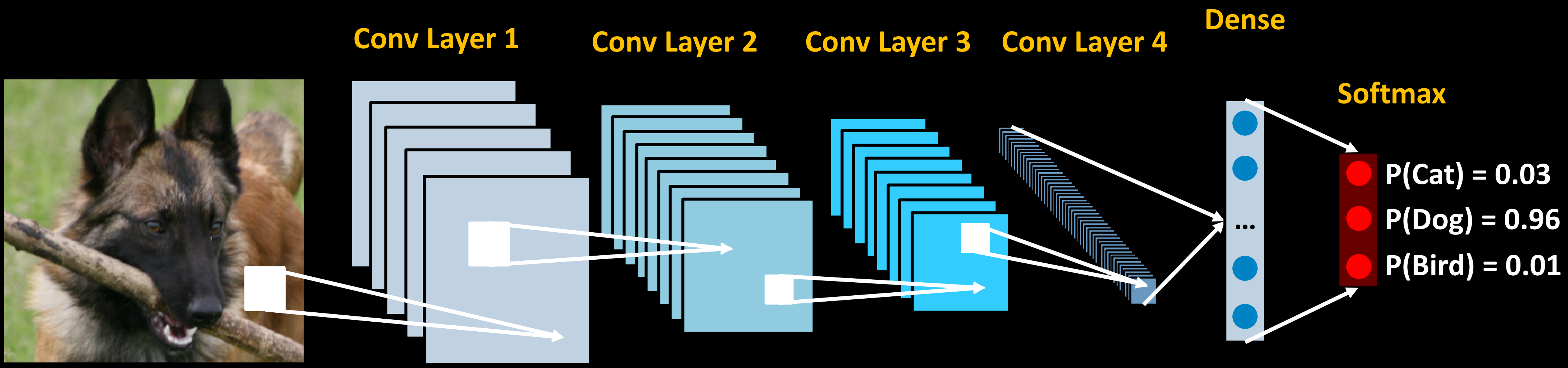
Supervised Learning



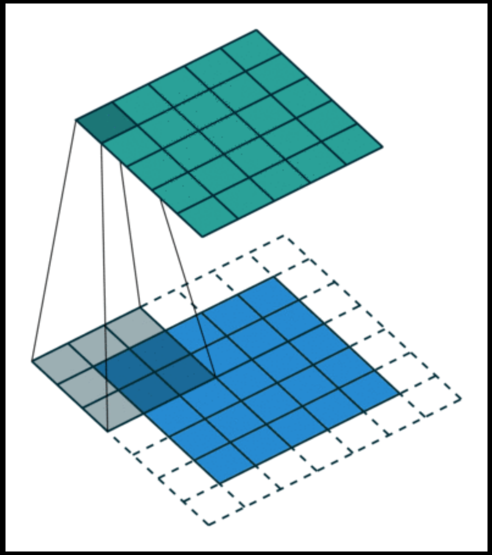
Unsupervised Learning (this is just one kind of it)



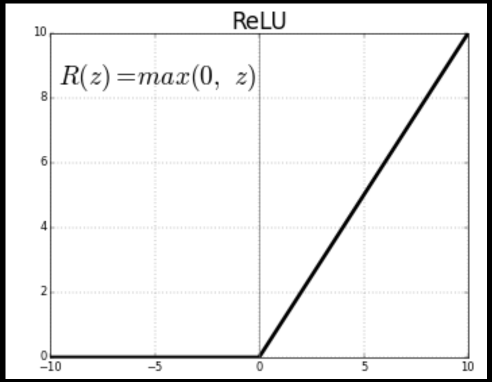
Convolutional Neural Networks



0	1	2
2	2	0
0	1	2

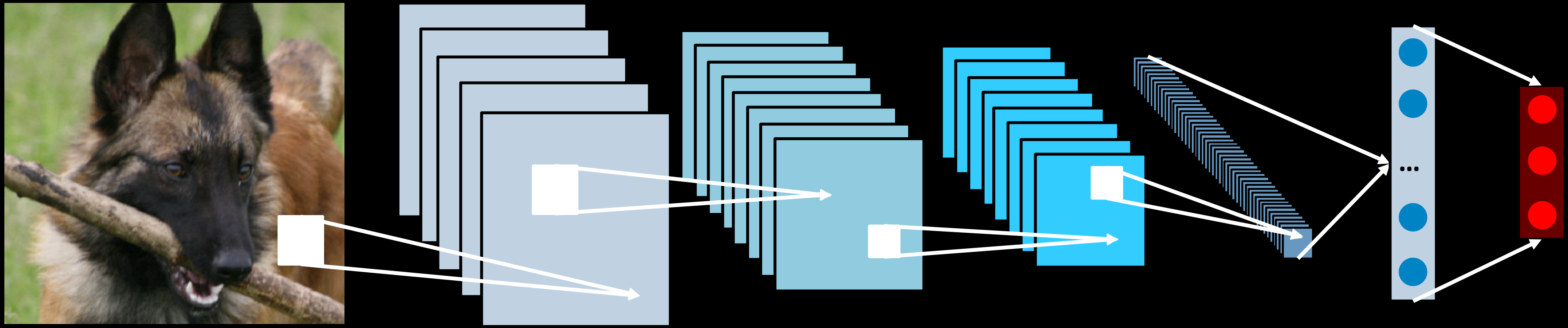


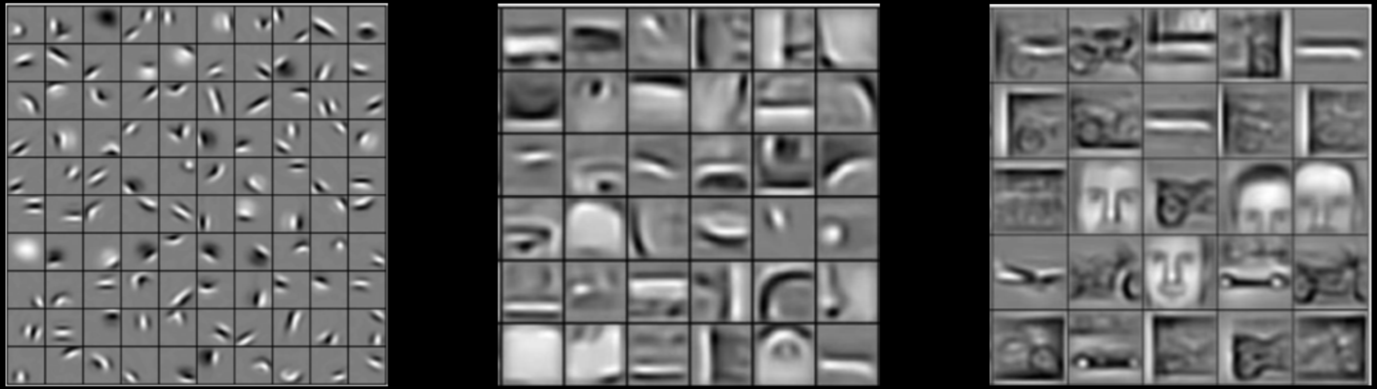
->



=

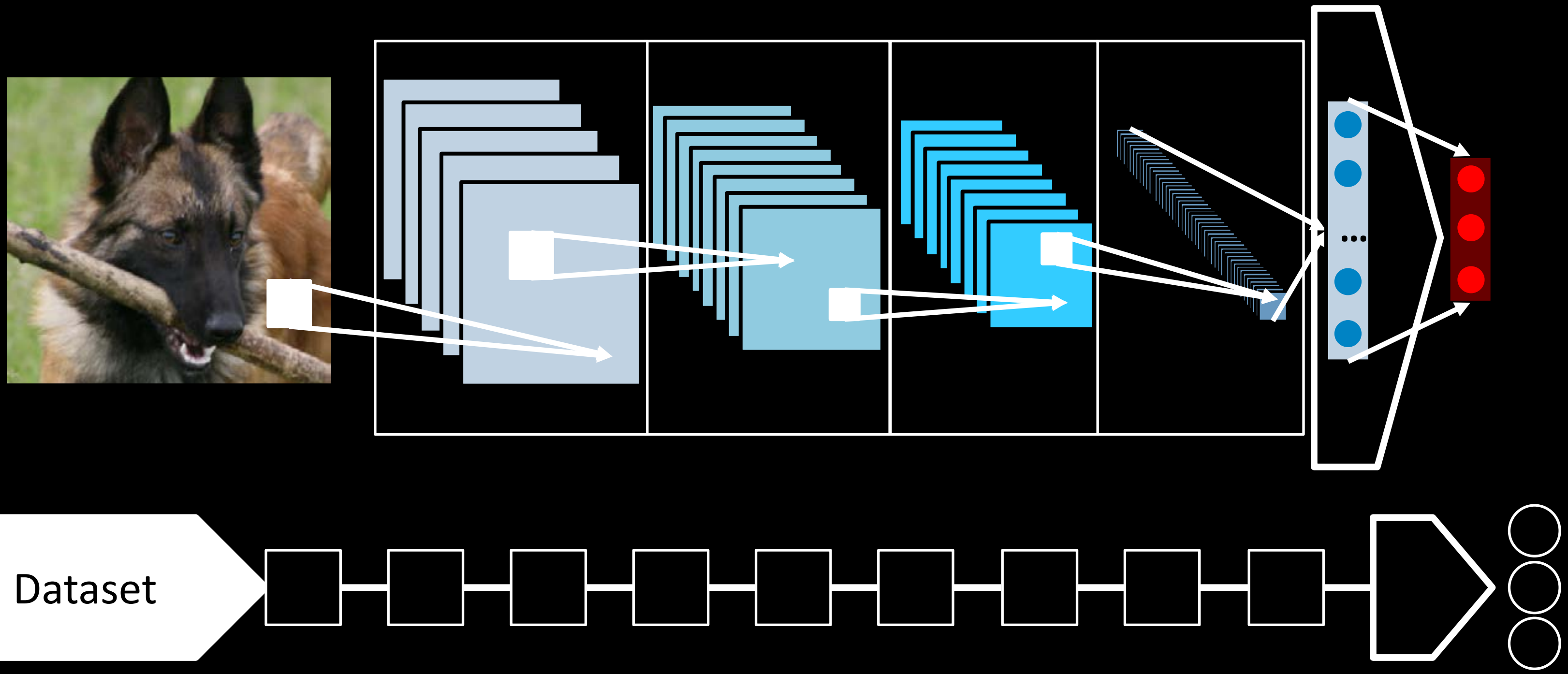
Conv gif: http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html





Visualization: Unsupervised Learning of Hierarchical Representations with Convolutional DBNs [Honglak Lee et al. 2011]

New visualization (more convenient)

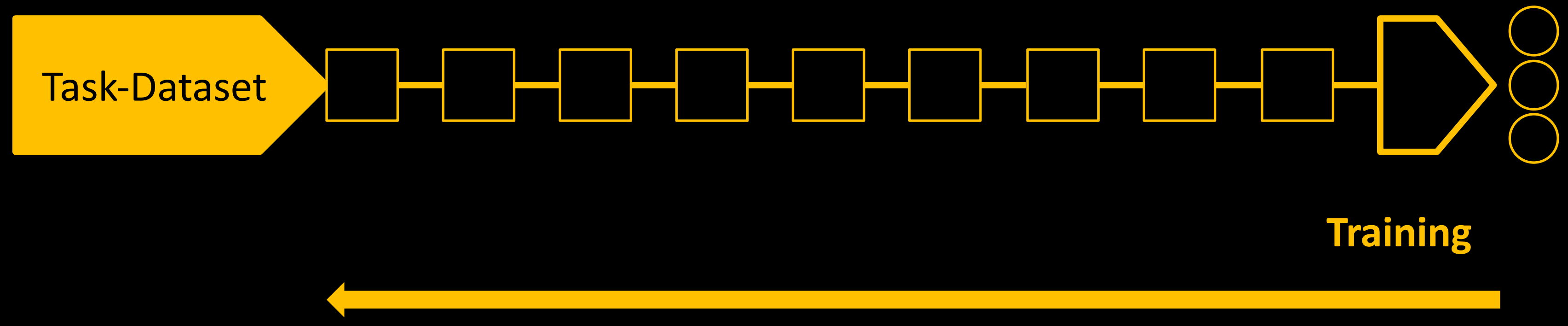


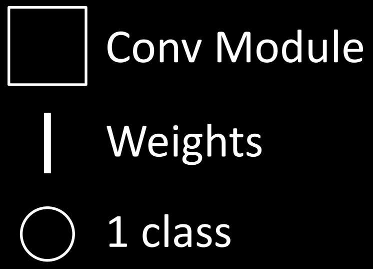
☐ Conv Module

| Weights

○ 1 class

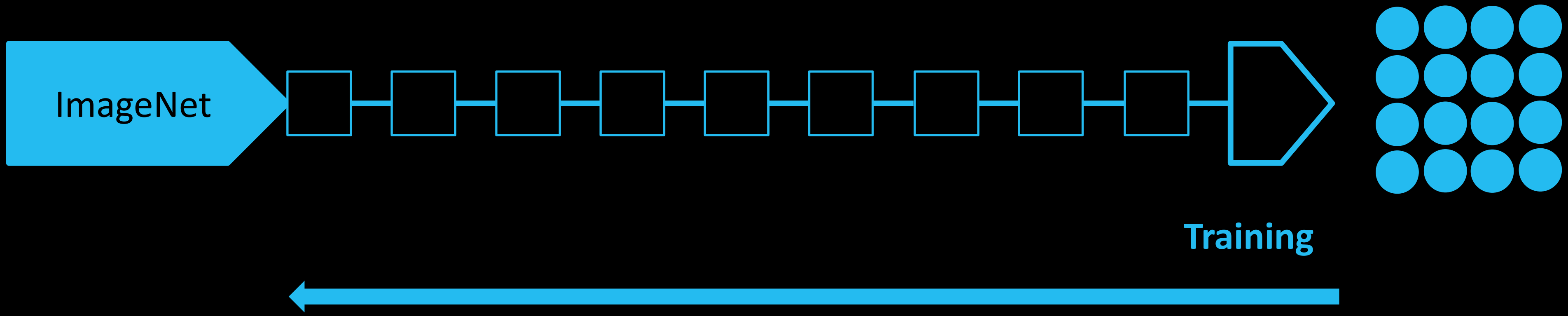
Basic Process: Train on your own Dataset





IMAGENET

Supervised Pre-Training

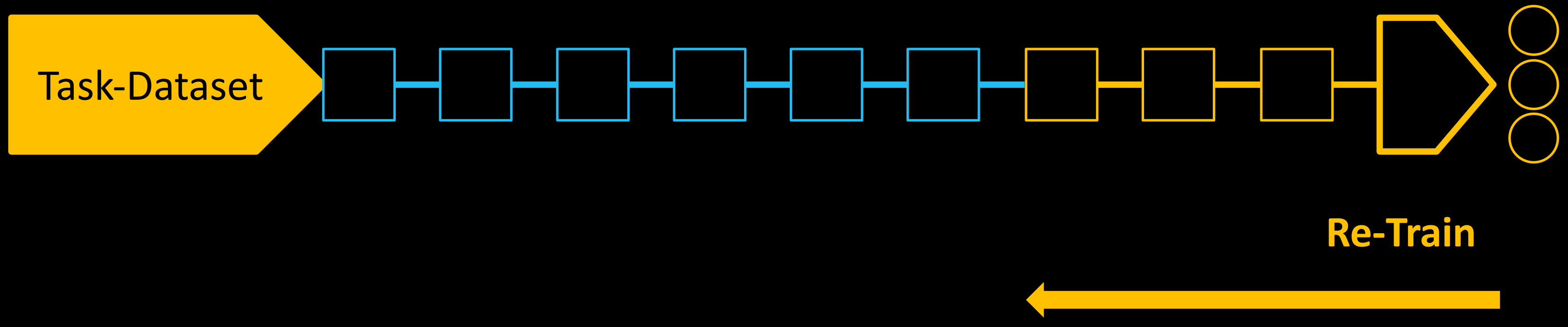


□ Conv Module

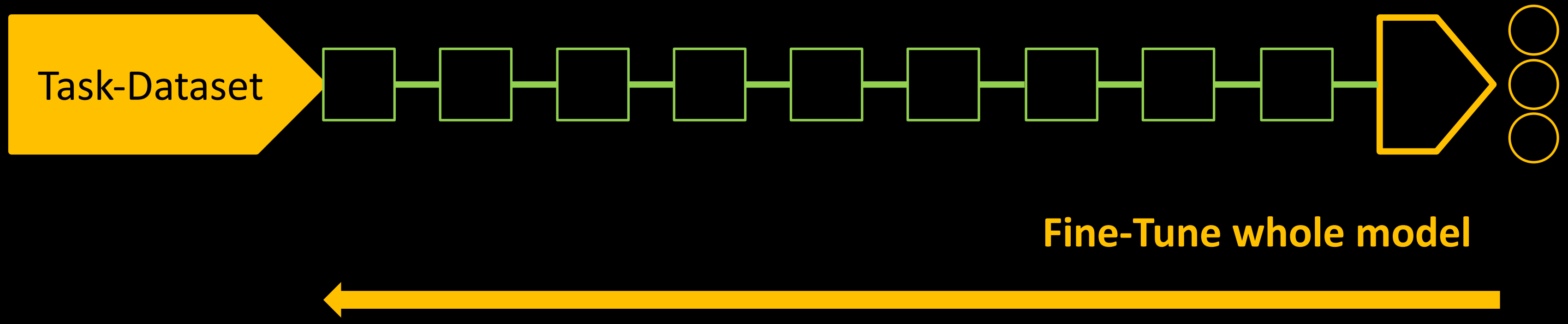
| Weights

○ 1 class

Re-Training last few layers.



Fine Tuning.



Language Models. Let’s play a game!

Machine

Language Models. Let’s play a game!

Machine learning algorithms build a mathematical

Language Models. Let’s play a game!

Machine learning algorithms build a mathematical model of sample

Language Models. Let’s play a game!

Machine learning algorithms build a mathematical model of sample data, known as “training data", in

Language Models. Let’s play a game!

Machine learning algorithms build a mathematical model of sample data, known as “training data", in order to make

Language Models. Let’s play a game!

Machine learning algorithms build a mathematical model of sample data, known as “training data", in order to make predictions or decisions without being explicitly

Language Models. Let’s play a game!

Machine learning algorithms build a mathematical model of sample data, known as “training data", in order to make predictions or decisions without being explicitly programmed

Language Models. Let’s play a game!

Machine learning algorithms build a mathematical model of sample data, known as “training data", in order to make predictions or decisions without being explicitly programmed to perform the

Language Models. Let’s play a game!

Machine learning algorithms build a mathematical model of sample data, known as “training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

Language Models. Let’s play a game!

Machine learning algorithms build a mathematical model of sample data, known as “training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

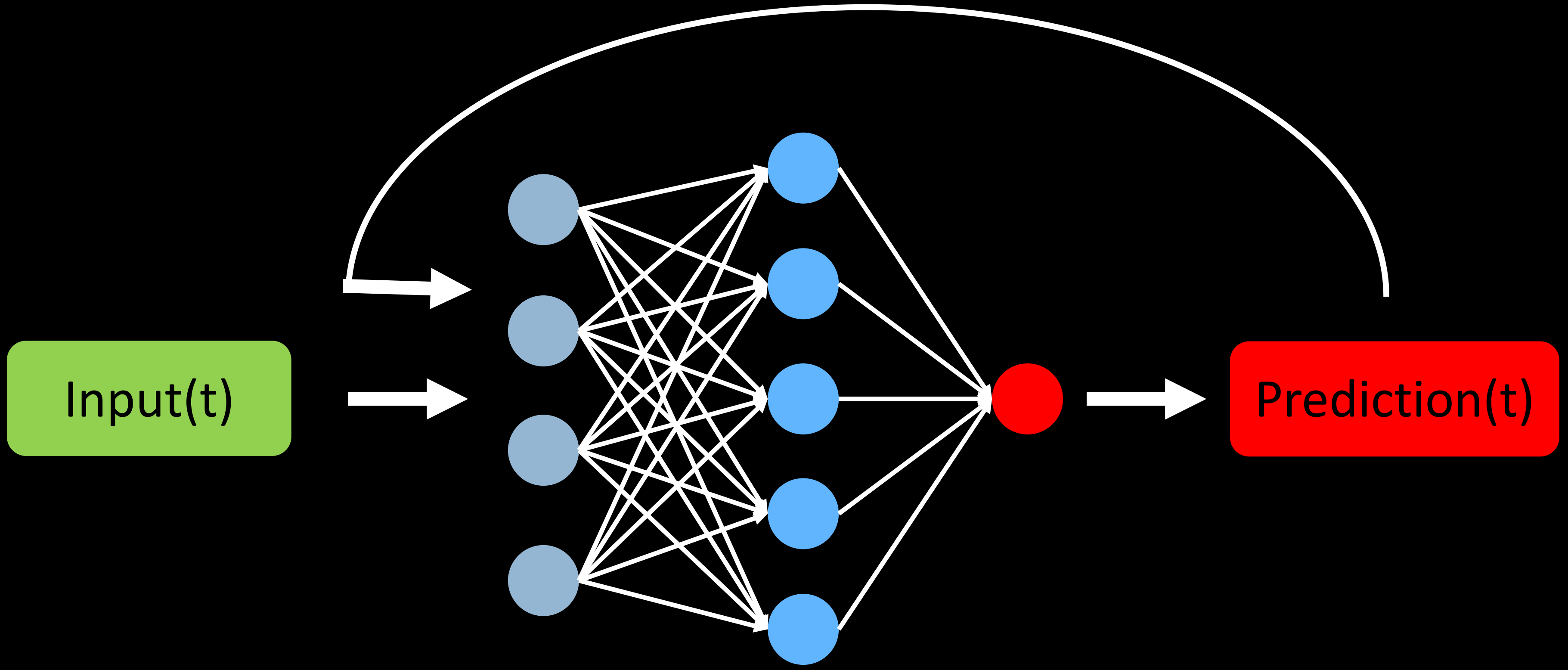
Source: https://en.wikipedia.org/wiki/Machine_learning

Natural Language Processing (NLP) tasks

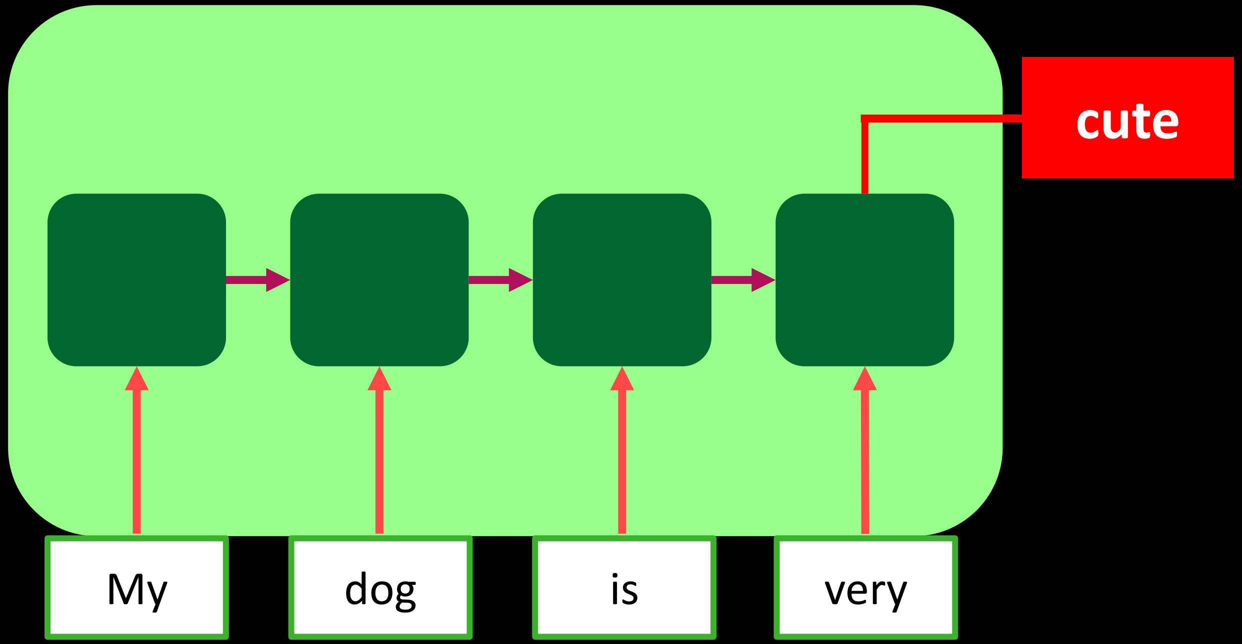
- **Classification (e.g. sentiment)**: “I love cats” -> Positive.
- **Language Models**: “I want to eat” -> “cake”
- **Translation**: “I like cats” -> “Ich mag Katzen”
- **Q&A**: “How gives this talk?” -> “Eric”
- **Similarity**: “I like cats” & “I like dogs” -> Yes.
- Summarization: long text -> summary

…

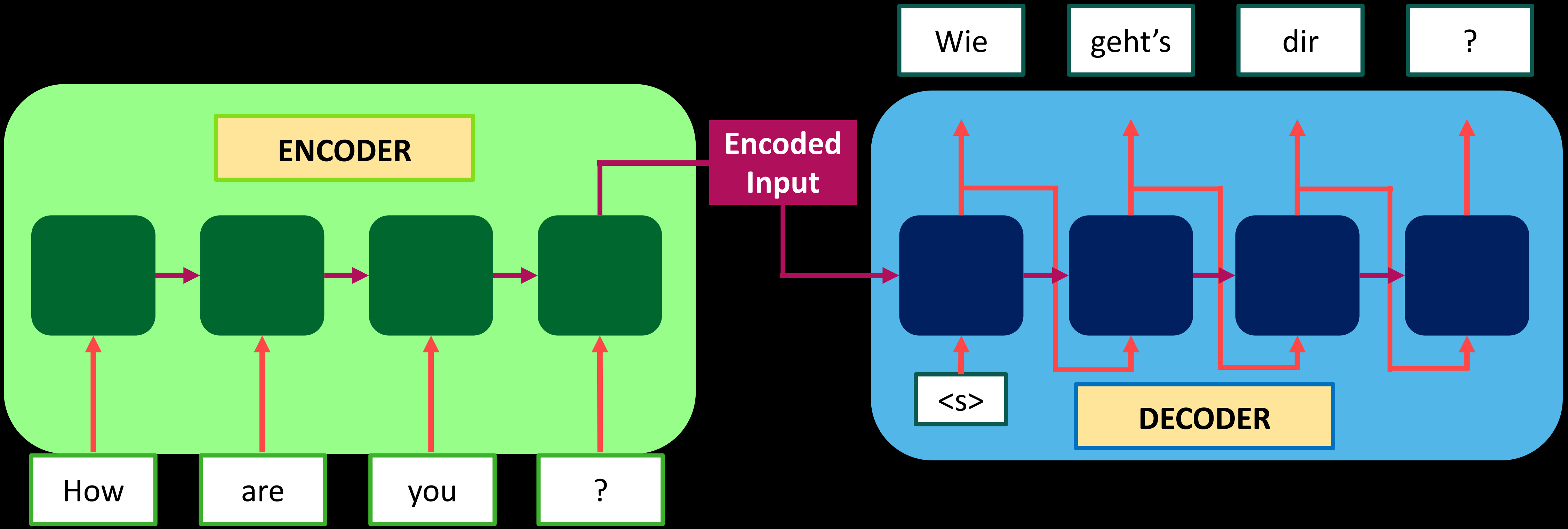
Recurrent Neural Networks (RNNs, LSTMs, GRUs, …)



Recurrent Neural Networks

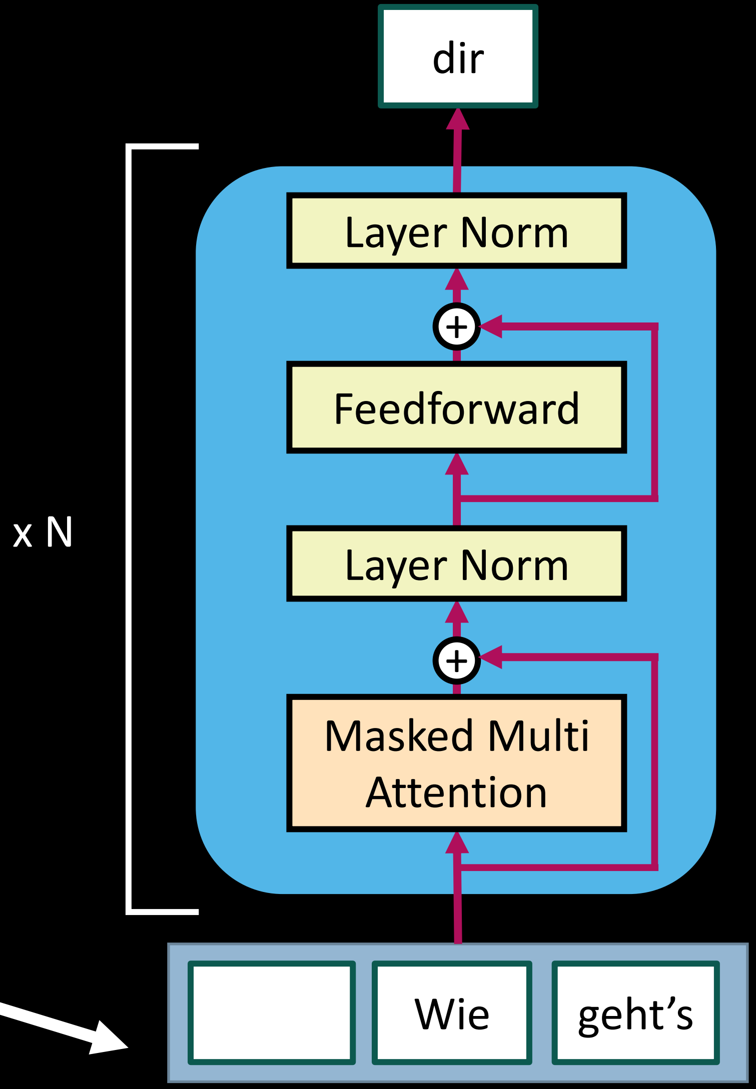


Recurrent Neural Networks: Seq2Seq Translation



Sequence to Sequence Learning with Neural Networks (Sutskever et al. 2014)

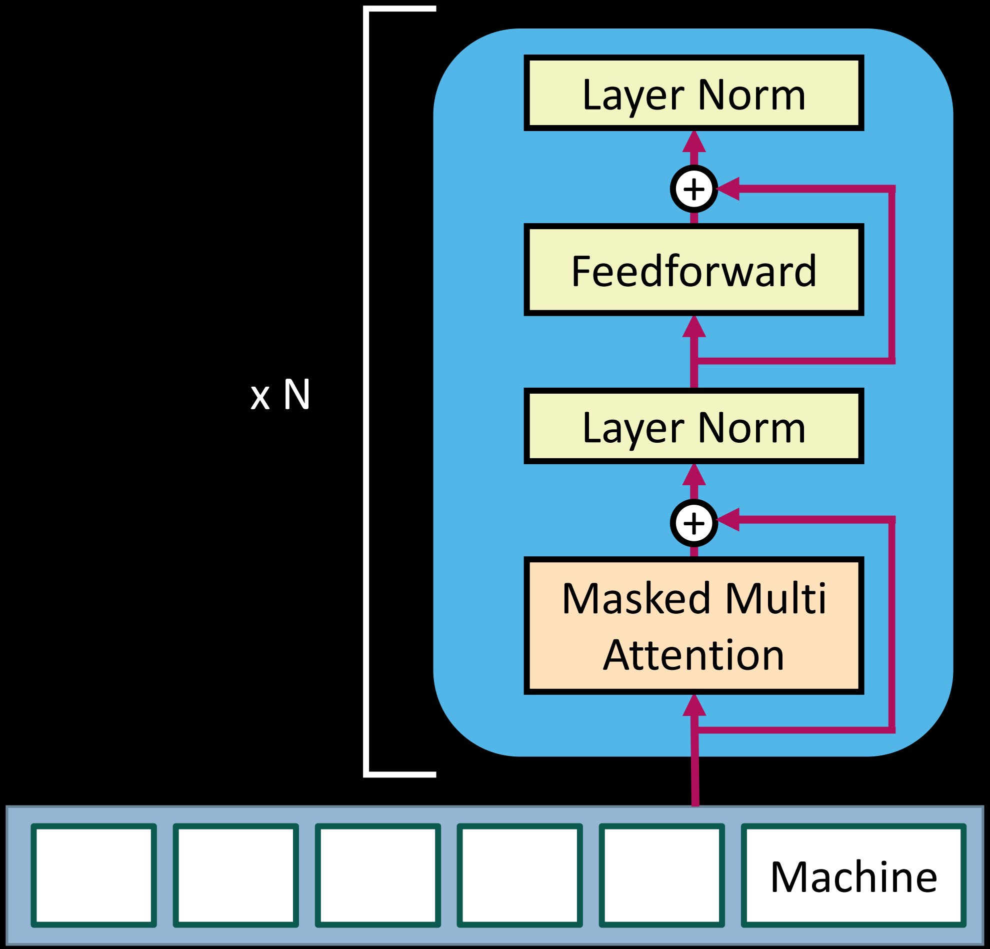
Transformers to replace RNNs



Use Position- and Word/Character Embedding

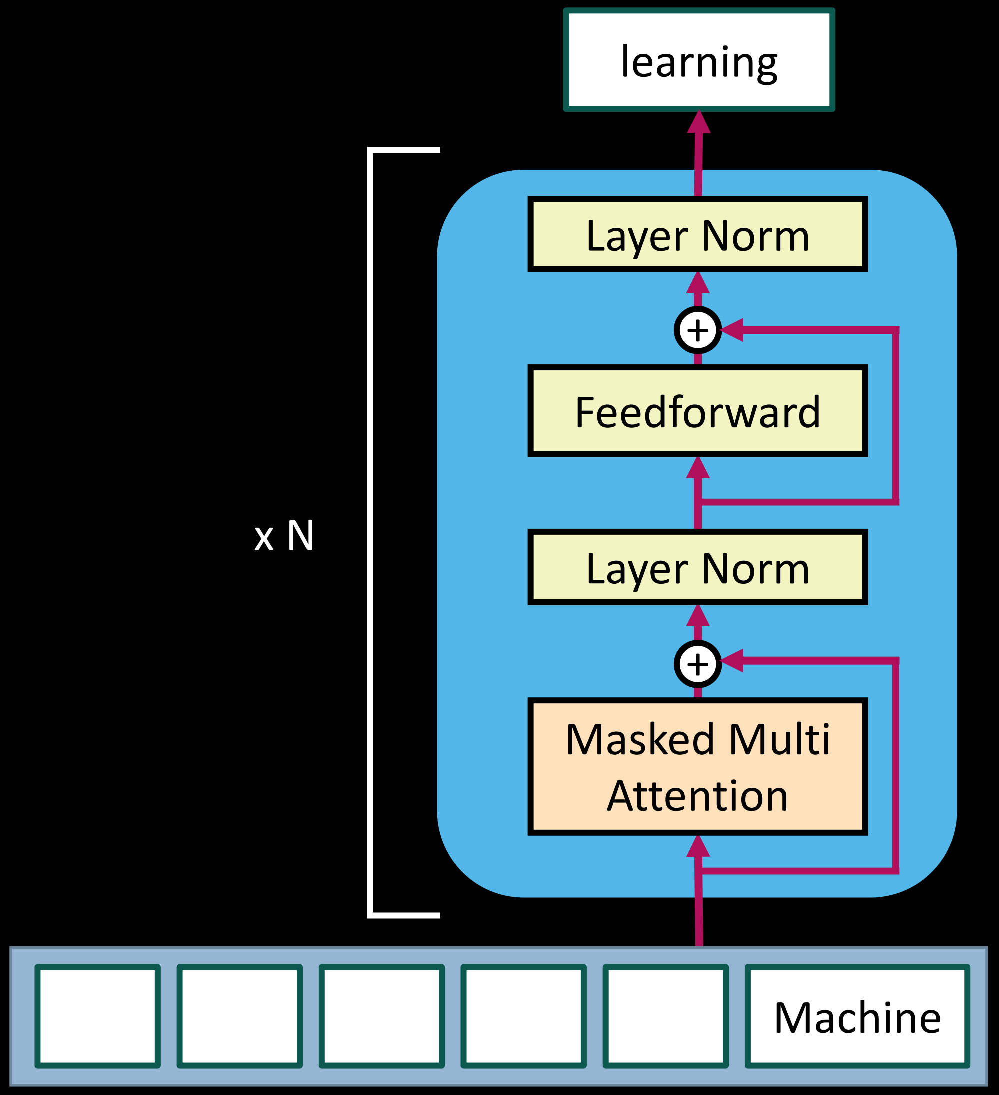
Attention Is All You Need (Vaswani et al.; 2017)

Language Models (LM)



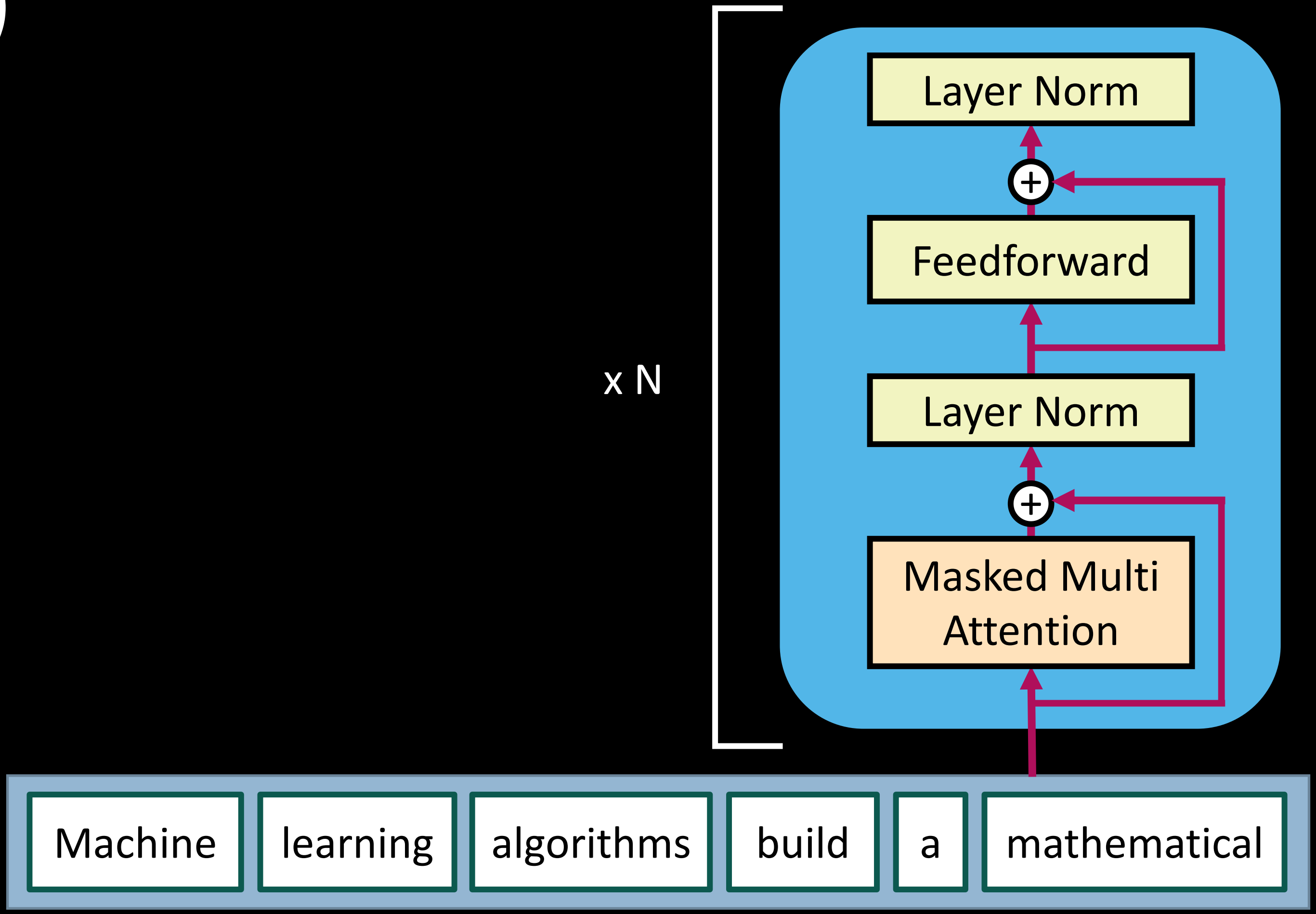
“Machine learning algorithms build a mathematical model of sample data, …”

Language Models (LM)



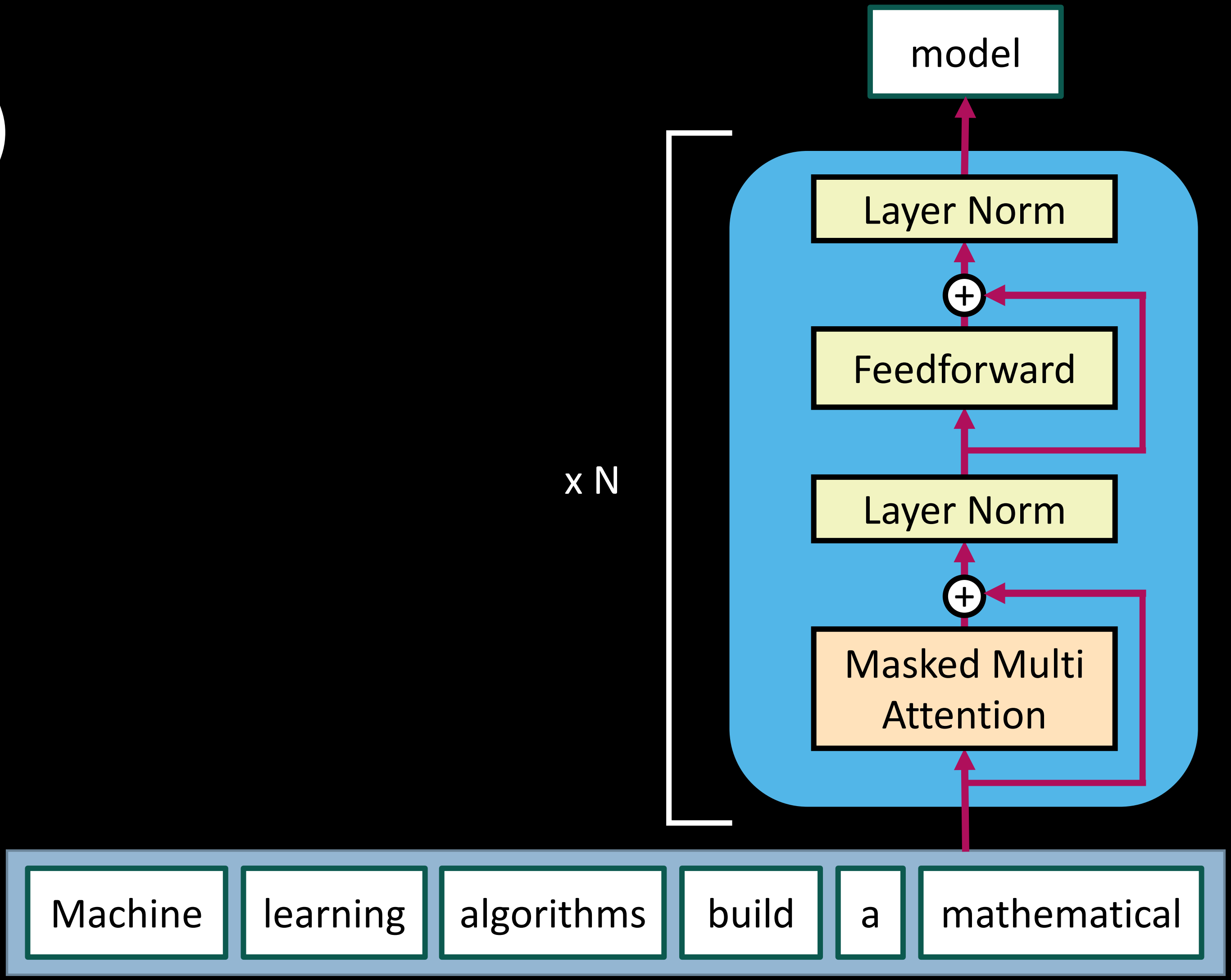
“Machine learning algorithms build a mathematical model of sample data, …”

Language Models (LM)



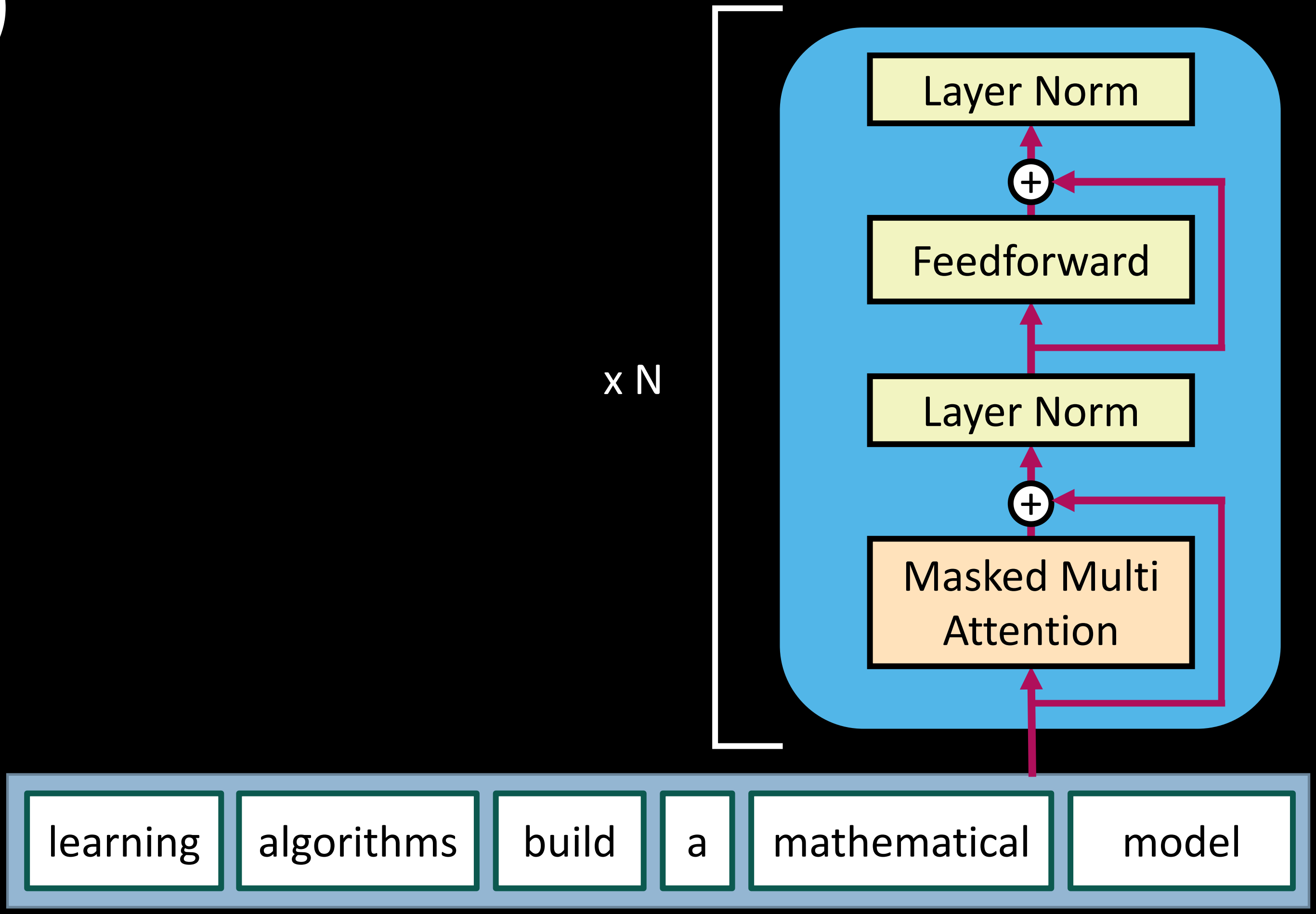
“Machine learning algorithms build a mathematical model of sample data, …”

Language Models (LM)



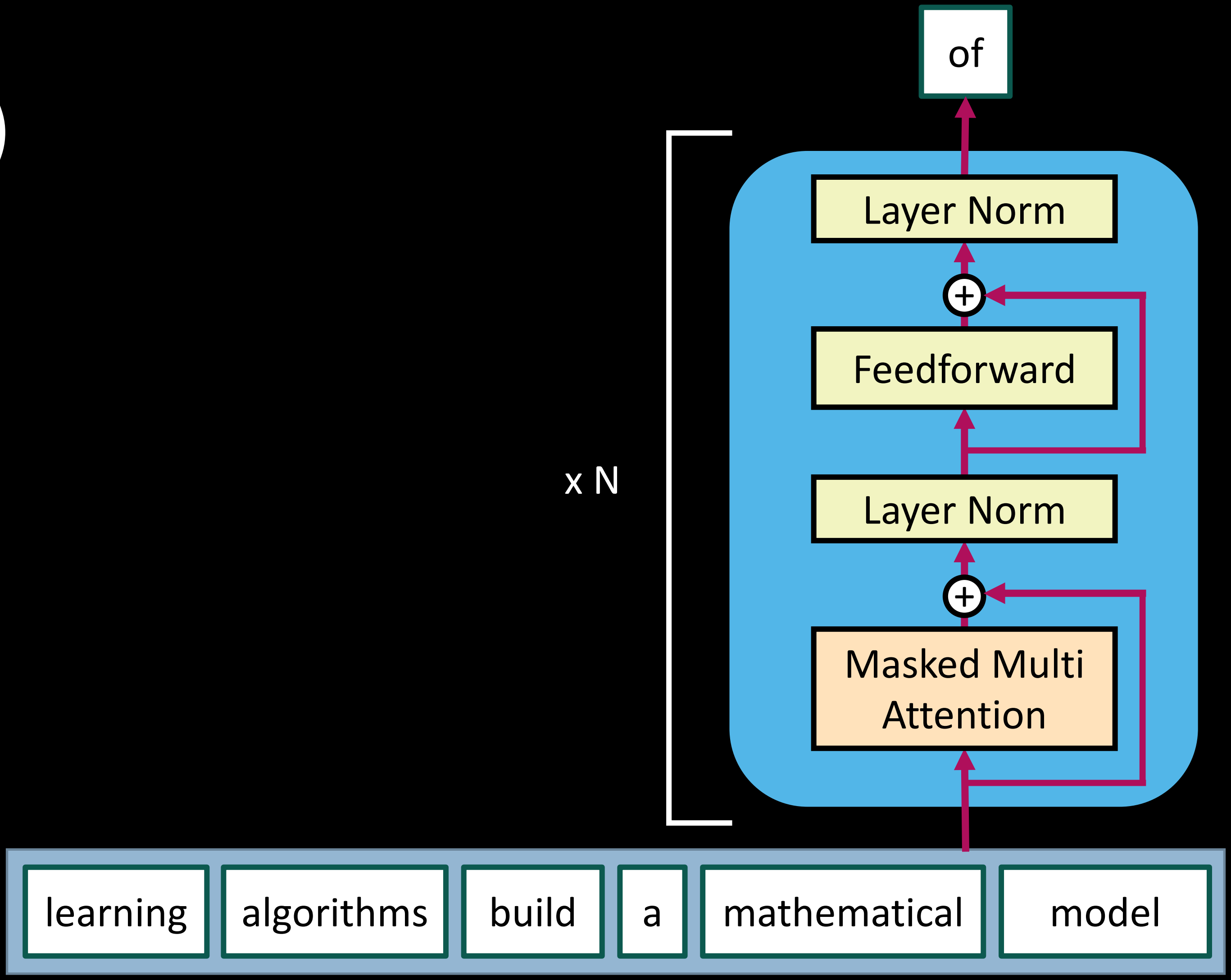
“Machine learning algorithms build a mathematical model of sample data, …”

Language Models (LM)



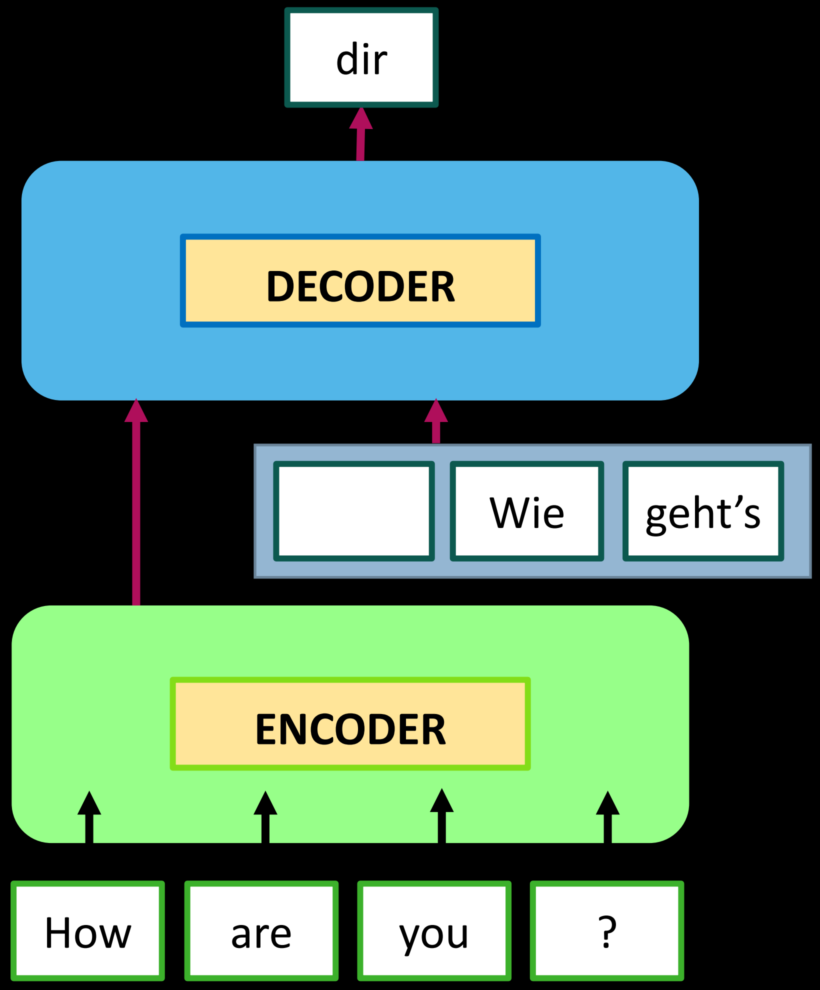
"Machine learning algorithms build a mathematical model of sample data, …"

Language Models (LM)



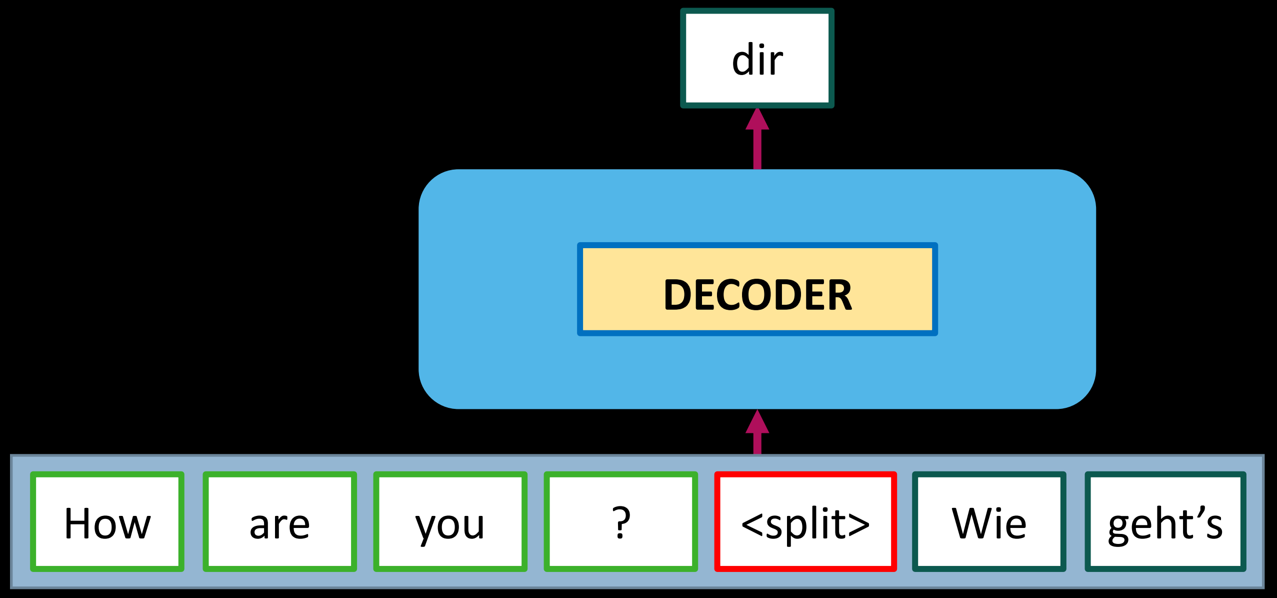
“Machine learning algorithms build a mathematical model of sample data, …”

Transformer



Attention Is All You Need (Vaswani et al.; 2017)

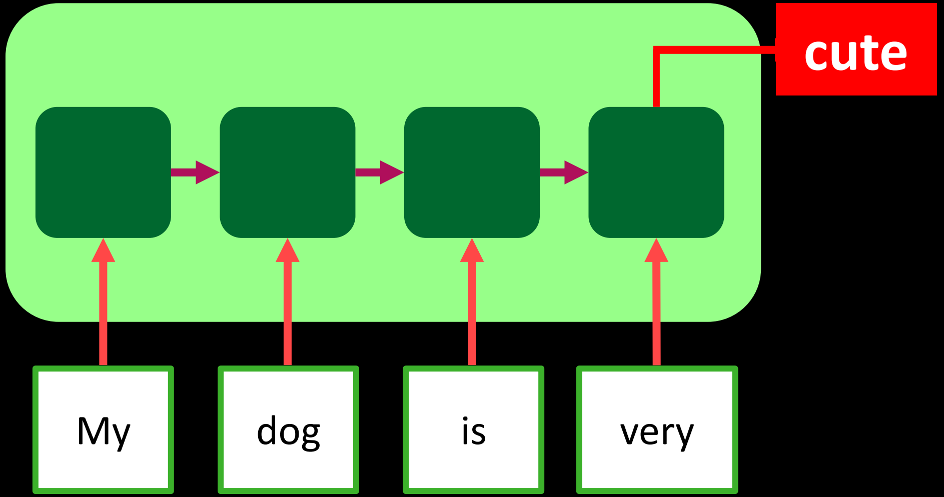
Transformer Decoder



Generating Wikipedia by summarizing long sequences (Liu et al.; 2018)

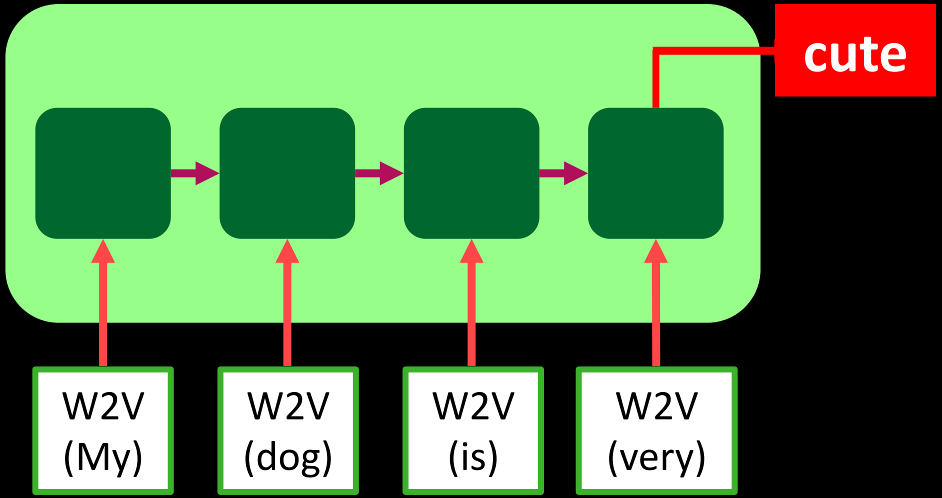
Word Embeddings

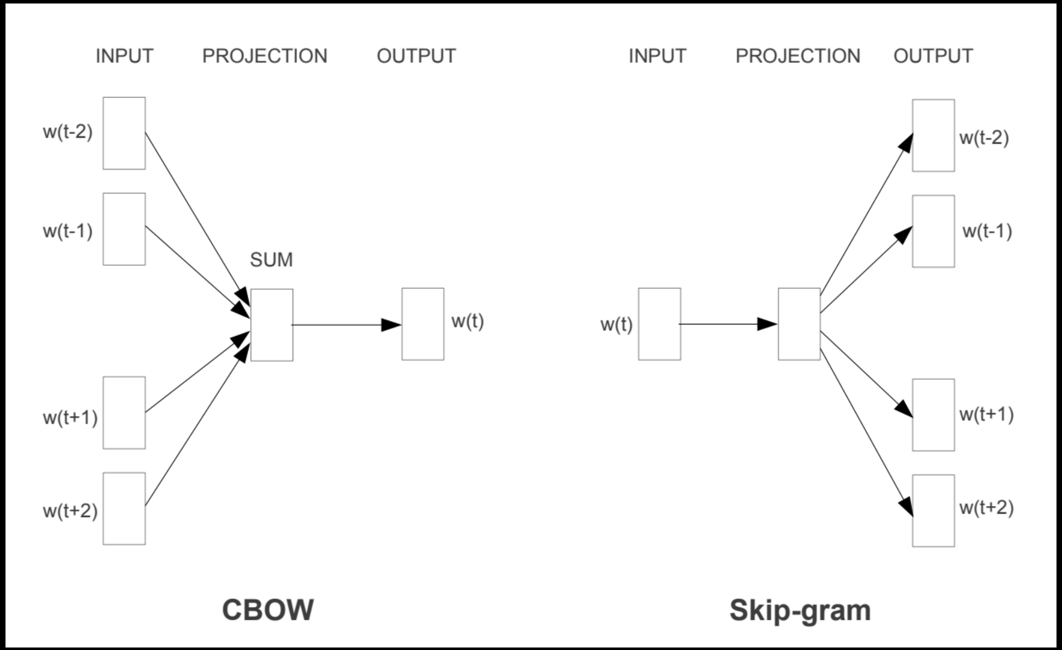
- Say we know 20,000 words. Each word is a vector like [0,0,0, …, 0, 1, 0, …, 0,0,0]
- Can we do better?



Word Embeddings: Word2Vec (Mikolov et al. 2013)

- E.g. „dog“ = [...,0,0,0,0,0,1,0,0,0,0,0,...]
- We want: [0.23, 1.48, 0.62, 0.52, 1.14]
- Use the weights from input as embedding





Word Embeddings: Word2Vec (Mikolov et al. 2013)

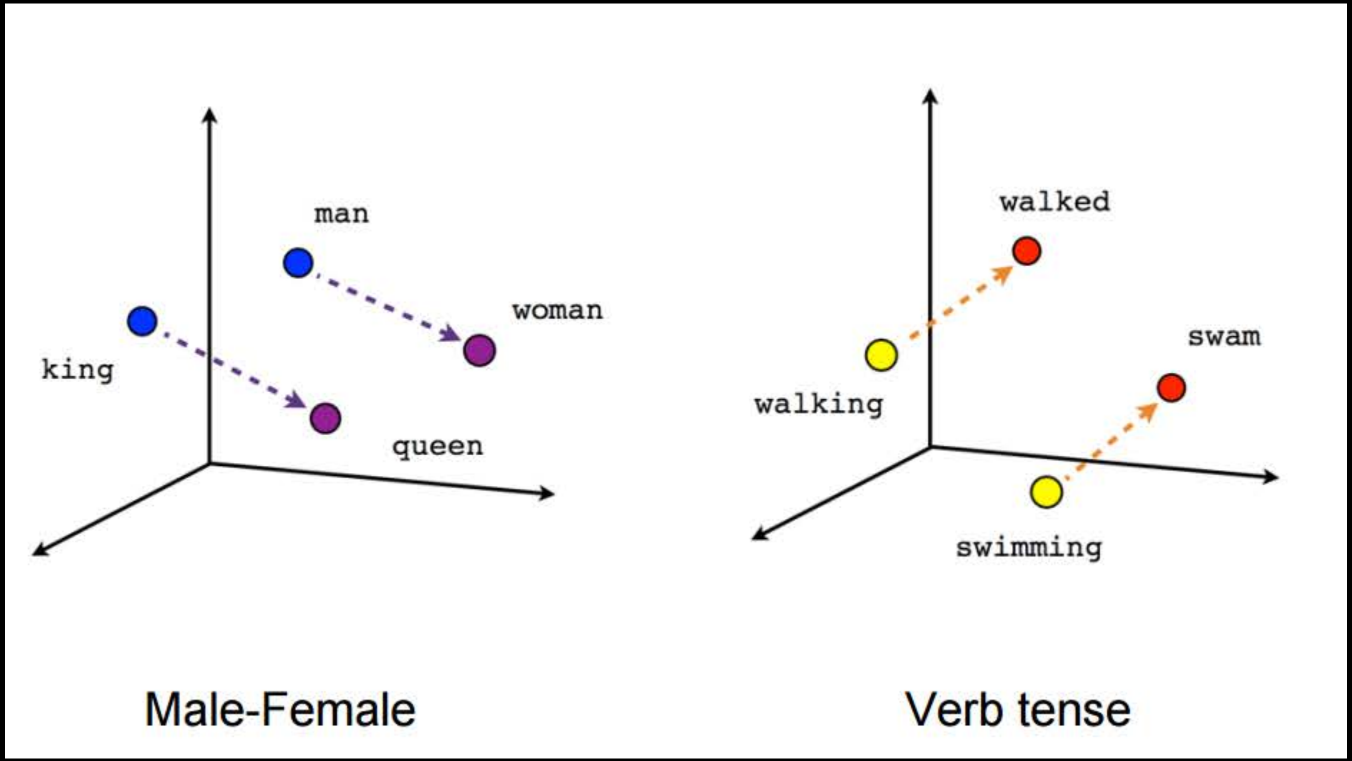
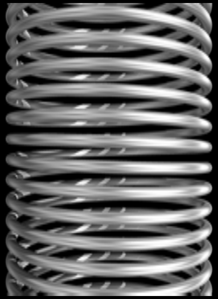


Image: https://www.tensorflow.org/tutorials/representation/word2vec

Better Word Embeddings

- Word2Vec look-up ignores context of word in given sentence, e.g. spring:

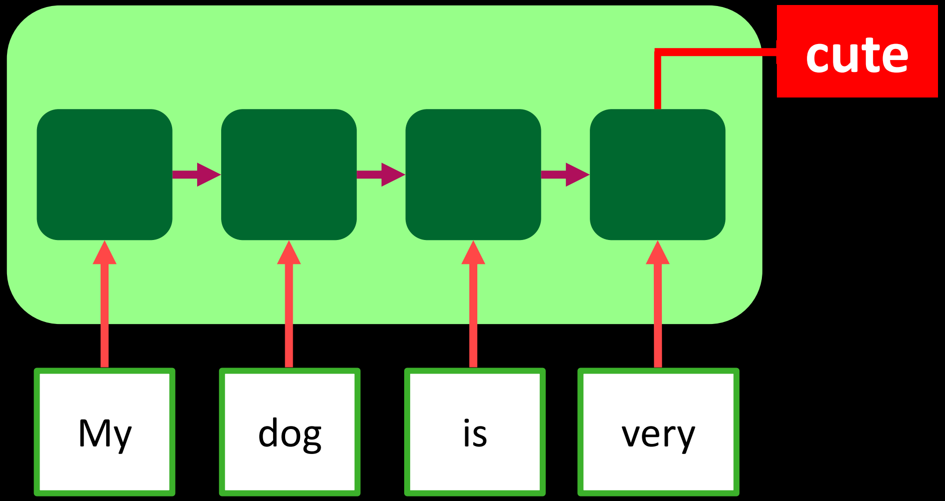


- **ELMo** (Embeddings from Language Models) (Peters et al. Feb 2018)
  - Is contextual
  - Deep recurrent bi-directional language model
  - Learns linear combination of all internal layer representations for each downstream task

ULMFiT (Howard & Ruder; Jan 2018)

- Universal Language Model Fine-tuning (**ULMFiT)**
  - Successful transfer learning of task net in NLP
  - Until ULMFiT, this didn‘t work well (Mou et al. 2016)

1. Pre-train LM on large data corpus
2. Fine-tune LM on target data
   - Fine-tunes later layers with higher learning rates
   - Slanted triangular learning rate schedules
3. Train classifier on target *task*
   - Concat pooling
   - Gradual unfreezing



Transformers?

- **GPT** (Generative Pre-training of Transformers) (Radford et al.; Jun 2018)
  - Why aren‘t we using Transformers?
- **BERT** (Bi-directional Encoder Representations from Transformers) (Devlin et al.; Oct 2018)
  - Uses bi-directional Transformer Encoder
  - Masked bi-directional LM (e.g. Books can <PREDICT ME> you a lot about the world)
  - Next Sentence Prediction
- **GPT v2** (Radford et al.; Feb 2019)
  - Ditch all the fancy BERT stuff
  - bigger = better
  - Zero-shot LM evaluation, new SoTA.
  - Can perform tasks without ever being shown (although not SoTA)

Fine-tuning the GPT pre-trained Transformer

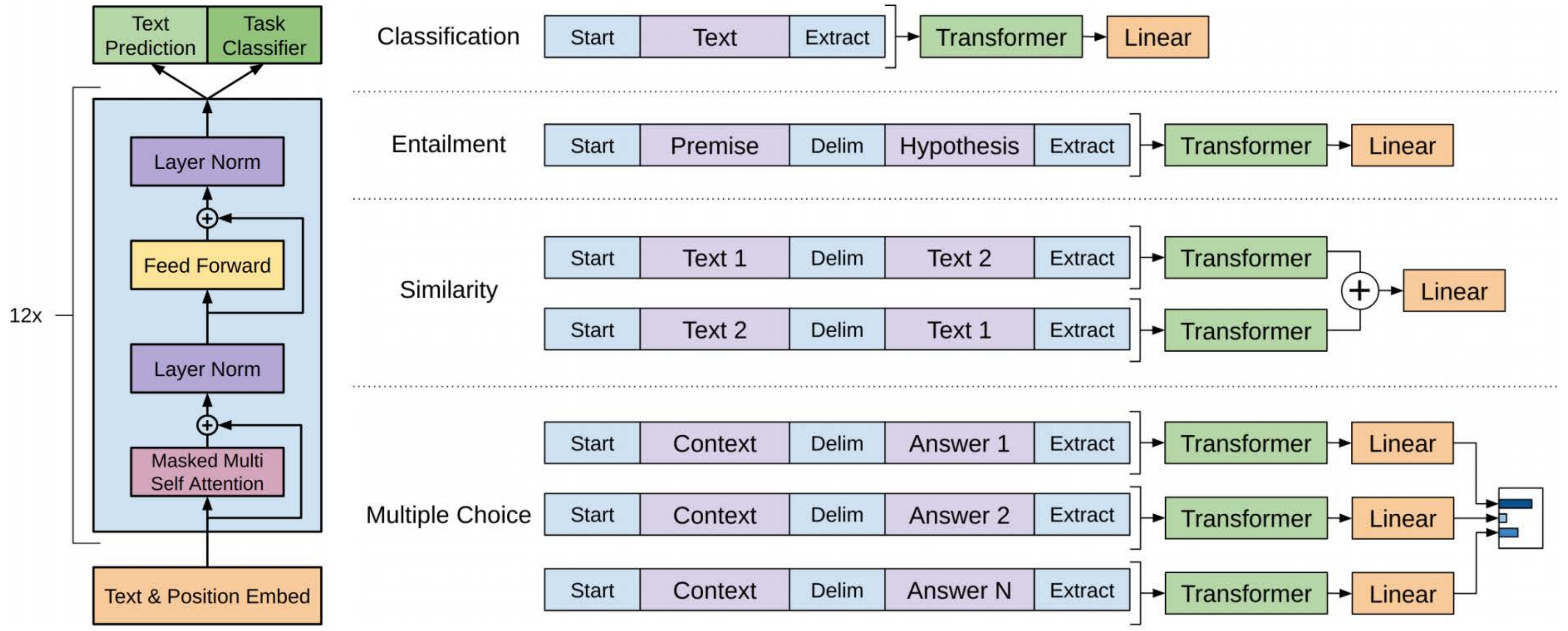


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Improving Language Understanding by Generative Pre-Training [Radford et al. 2018]

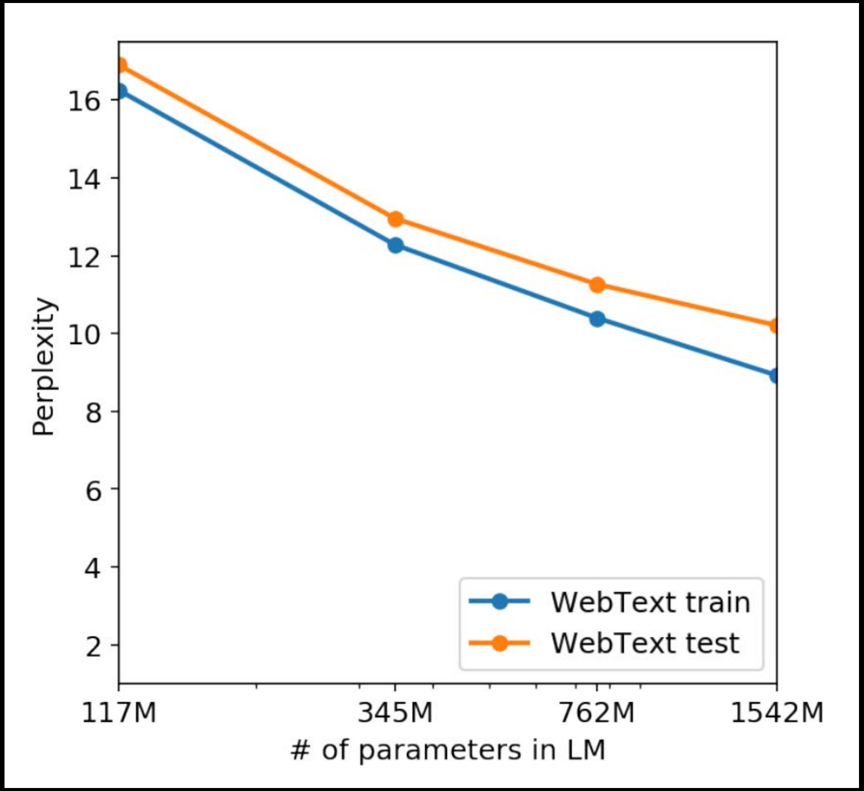
Task embedding for GPT v2: e.g. Translation

[English sentence 1 = French sentence 1 <X> English sentence 2 = French sentence 2 … English sentence =]

--> The network then continues “language modelling” this text with a French translation.

GPT v2

- #Parameters: 117M -> 1.5B
- Data: 40gb of diverse internet text
- Minor neural architecture modifications

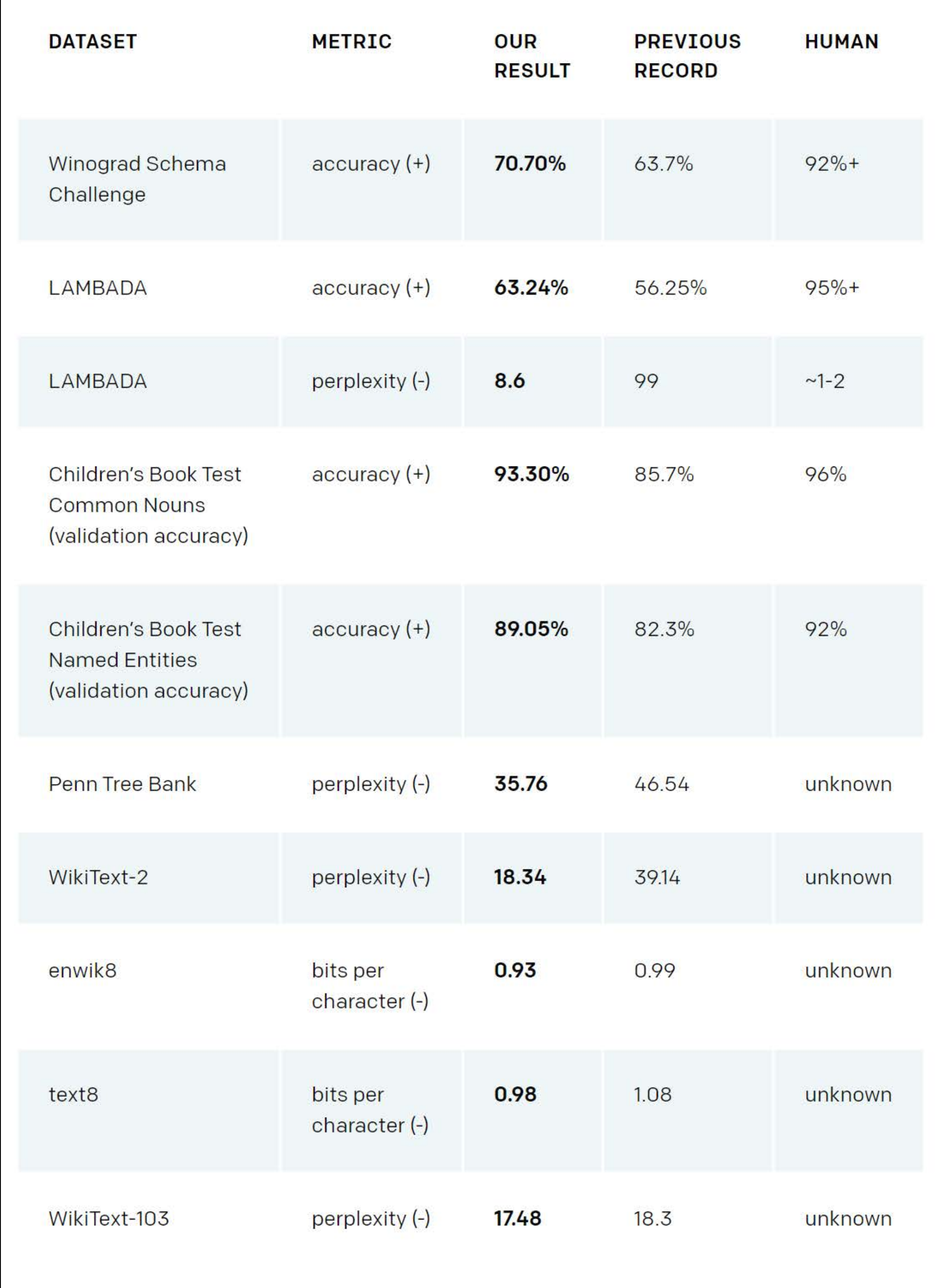


Language Models are Unsupervised Multitask Learners [Radford et al. 2019]

GPT v2 results

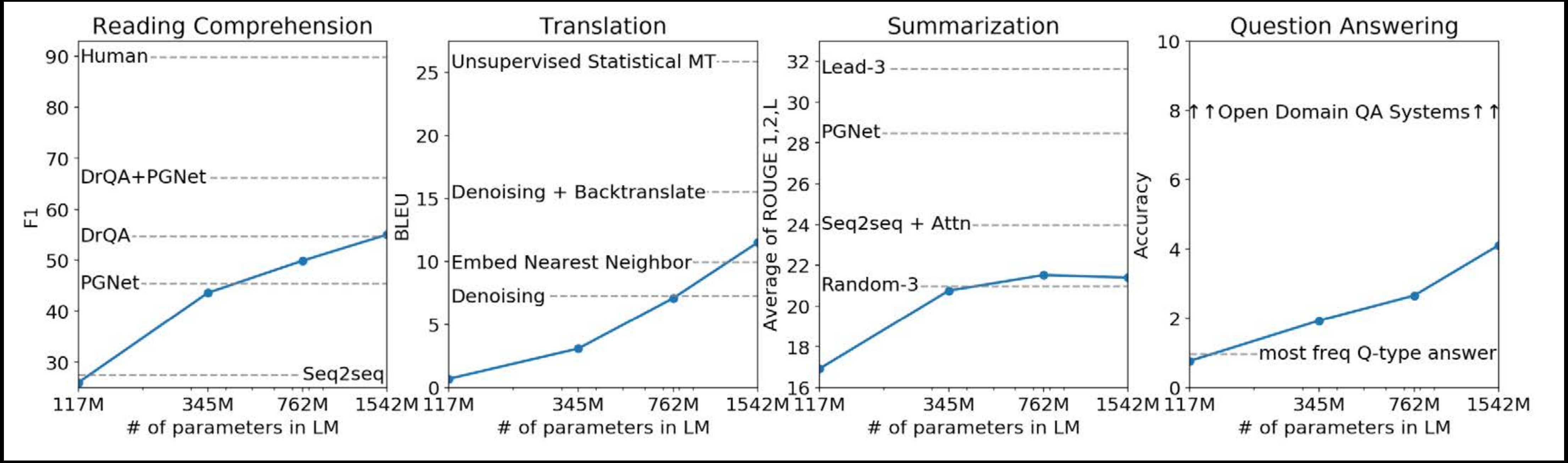
Without any
***dataset-specific* data!**

Language Models are Unsupervised Multitask Learners (Radford et al. @OpenAI)



DATASET	METRIC	OUR RESULT	PREVIOUS RECORD	HUMAN
Winograd Schema Challenge	accuracy (+)	**70.70%**	63.7%	92%+
LAMBADA	accuracy (+)	**63.24%**	56.25%	95%+
LAMBADA	perplexity (-)	**8.6**	99	~1-2
Children's Book Test Common Nouns (validation accuracy)	accuracy (+)	**93.30%**	85.7%	96%
Children's Book Test Named Entities (validation accuracy)	accuracy (+)	**89.05%**	82.3%	92%
Penn Tree Bank	perplexity (-)	**35.76**	46.54	unknown
WikiText-2	perplexity (-)	**18.34**	39.14	unknown
enwik8	bits per character (-)	**0.93**	0.99	unknown
text8	bits per character (-)	**0.98**	1.08	unknown
WikiText-103	perplexity (-)	**17.48**	18.3	unknown

GPT v2 zero-shot task transfer performance



Language Models are Unsupervised Multitask Learners (Radford et al. @OpenAI)

Takeaways

- Large networks can learn generally applicable knowledge from huge, diverse but labeled or unlabeled datasets
- These learnings can transfer to other domains and tasks
- In Deep Learning, bigger = better

Improving Supervised Deep Learning with Unsupervised Deep Learning

Eric Steinberger

E-mail: eric@steinberger-ai.com
GitHub: TinkeringCode
Twitter: @EricSteinb
LinkedIn: Eric Steinberger

Learn more at

Transformers & BERT: jalammar.github.io
GPT & GPT-2: blog.OpenAI.com
Deep Unsupervised Learning: Berkeley CS294-158 (YouTube)
Papers referenced on my slides are all on Arxiv.org

Q&A