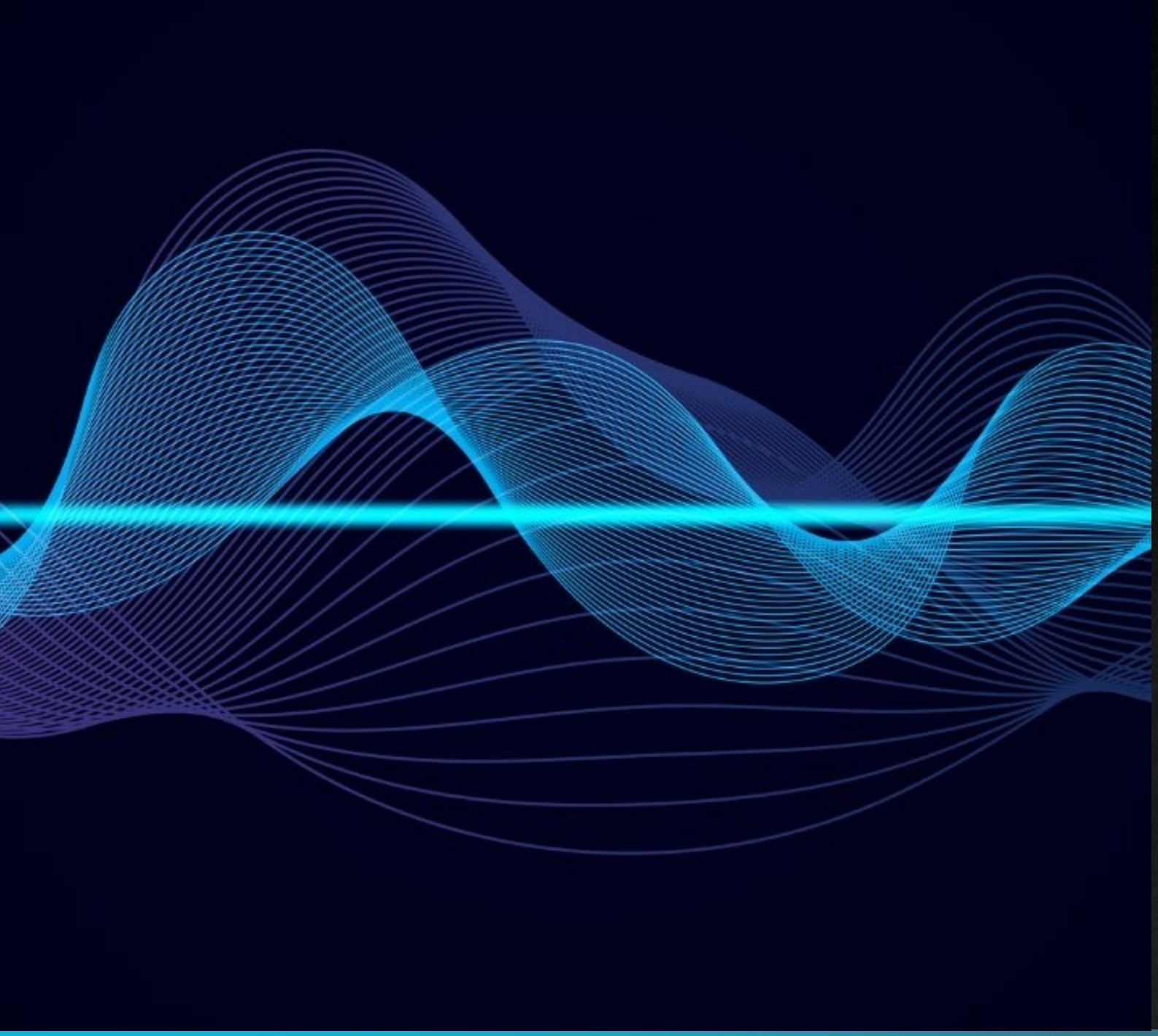


Computational Physics

PHYS 6260

Deep Learning: Time-dependent NNs

Announcements:

- Progress report: Due Friday 3/28
- HW7: Due Friday 4/4

Lecture Outline

We will cover these topics

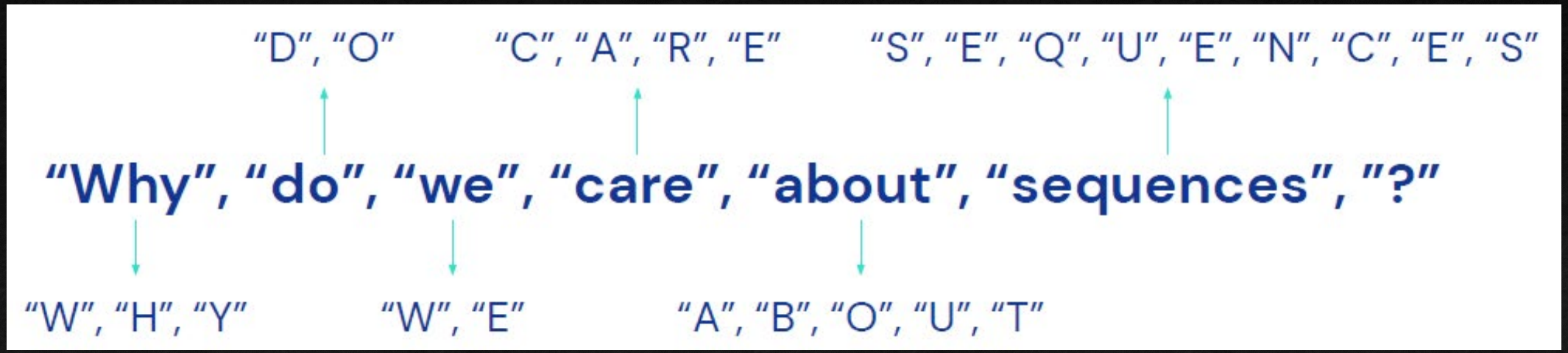
- Sequences
- Recurrent NNs
- Long-term dependencies
- Transformers
- Physics-informed NNs
- Latest research with Physics ML

Sequences

- Classification with NNs and CNNs are analogous to time-independent problems
- How can we capture the time-dependence of a system?
- Consider a collection of elements
  - Elements can repeated
  - Order matters
  - Sequence is a variable (potentially infinite) length
- NN / CNNs don’t do well with sequential data

Sequences

"Why do we care about sequences?"



Examples: language, audio, videos, images, programs, decision making, physics → Train a model to emulate the "real" data

Training ML models

	Supervised learning	Sequence modelling
Data	$\{x, y\}_i$	$\{x\}_i$
Model	$y \approx f_\theta(x)$	$p(x) \approx f_\theta(x)$
Loss	$\mathcal{L}(\theta) = \sum_{i=1}^{N} l(f_\theta(x_i), y_i)$	$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log p(f_\theta(x_i))$
Optimisation	$\theta^* = \arg\min_\theta \mathcal{L}(\theta)$	$\theta^* = \arg\max_\theta \mathcal{L}(\theta)$

Modeling the probability p(x)

- “Modeling word probabilities is really difficult”
- Simplest model: assume independence of words

$$p(x) = \Pi_{t=1}^{T} p(x_t)$$

- p(“modeling”) x p(“word”) x … x p(“difficult”)
- In English, p(“the”) = 0.049 and p(“be”) = 0.028 are the two most common words
- Results in the most likely 6-word sentence: “the the the the the the”
- Independence assumption doesn’t match sequential structure of language

Modeling the probability p(x)

- More realistic model: Assume conditional dependence of words

$$p(x_T) = p(x_T|x_1, \dots, x_{T-1})$$

Modeling word probabilities is really ?

Context | Target

Target	p(x|context)
difficult	0.01
hard	0.009
fun	0.005
...	...
easy	0.00001

Modeling the probability p(x)

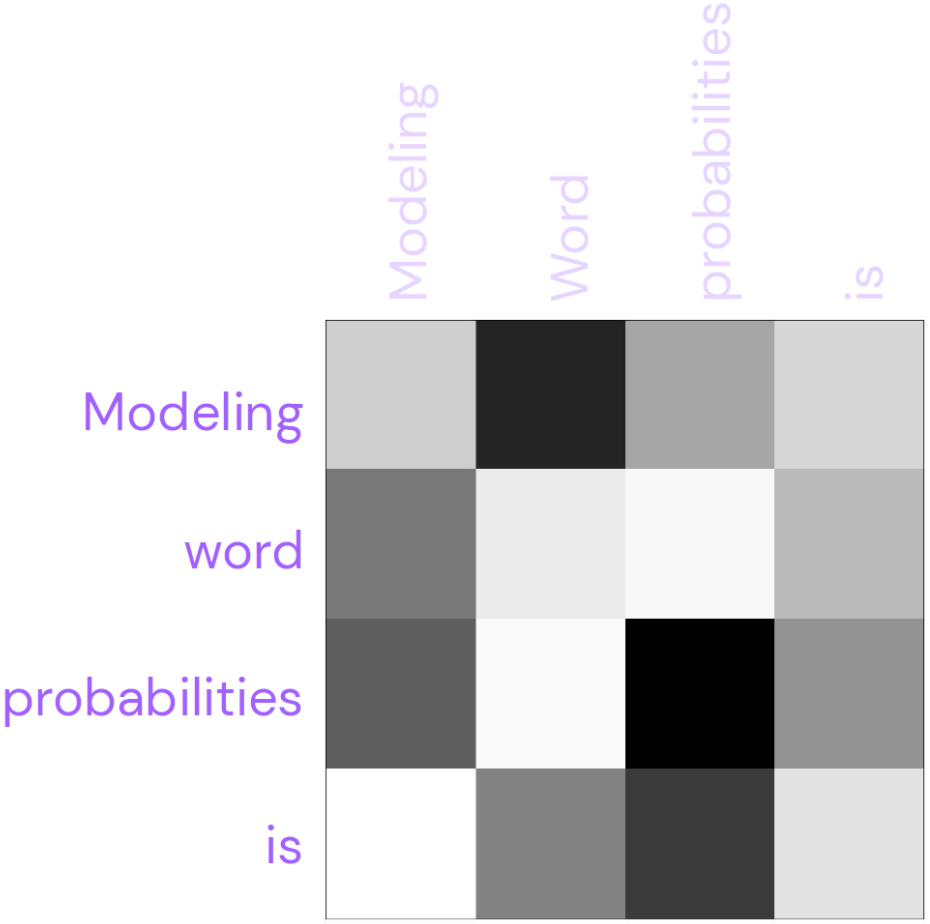
- Chain rule: Computing the joint p(x) from conditionals

$$p(x) = \Pi_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1})$$

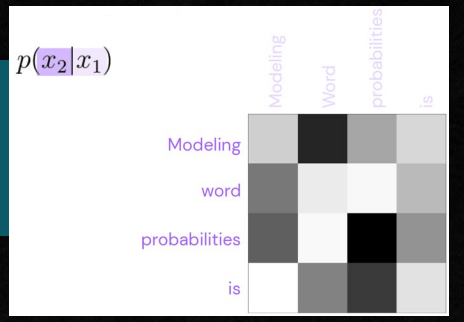
Modeling	$p(x_1)$
Modeling **word**	$p(x_2 \mid x_1)$
Modeling word **probabilities**	$p(x_3 \mid x_2, x_1)$
Modeling word probabilities **is**	$p(x_4 \mid x_3, x_2, x_1)$
Modeling word probabilities is **really**	$p(x_5 \mid x_4, x_3, x_2, x_1)$
Modeling word probabilities is really **difficult**	$p(x_6 \mid x_5, x_4, x_3, x_2, x_1)$

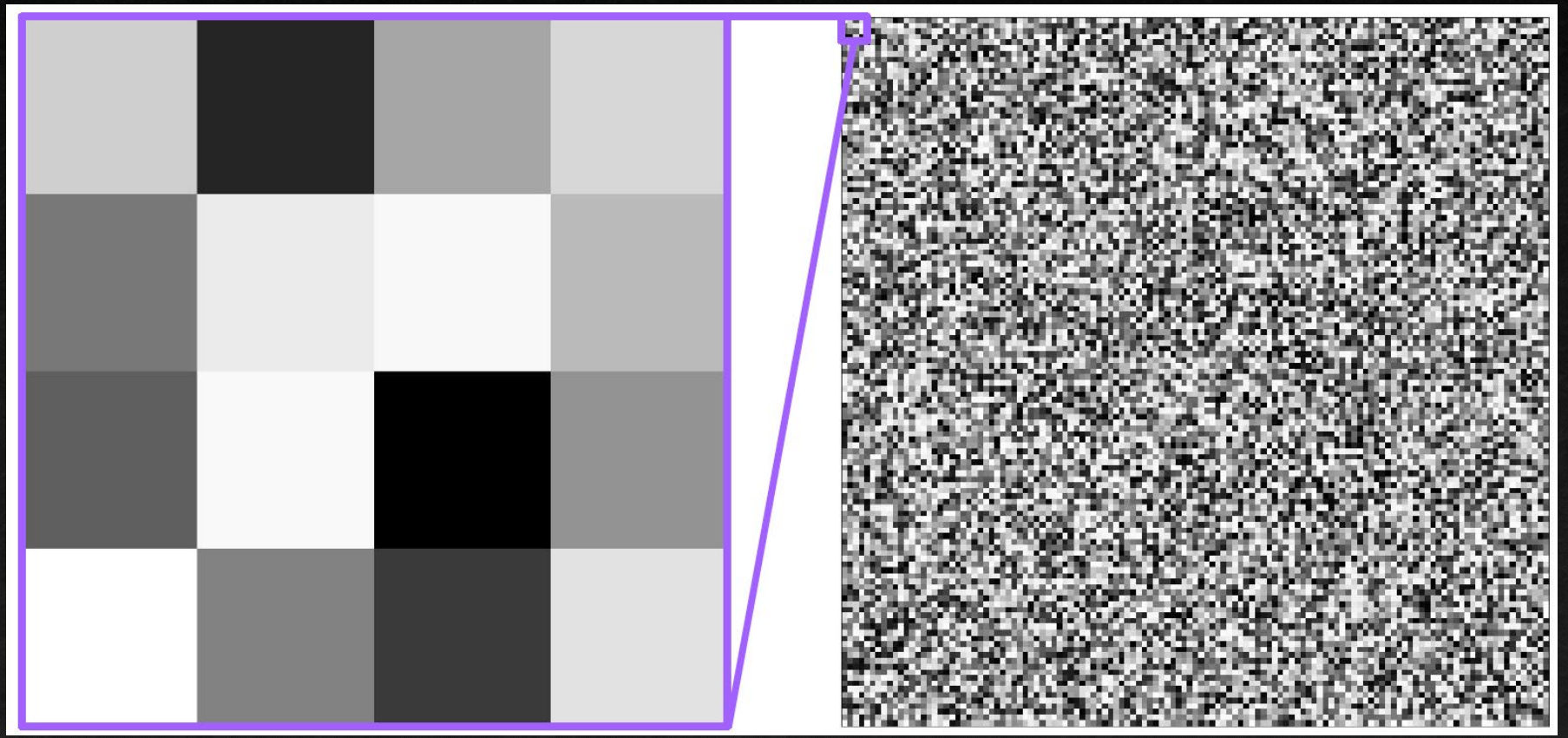
Scalability issues

$p(x_2|x_1)$

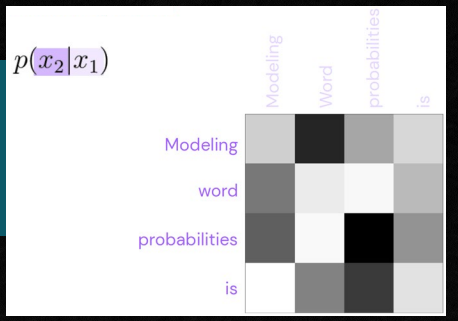


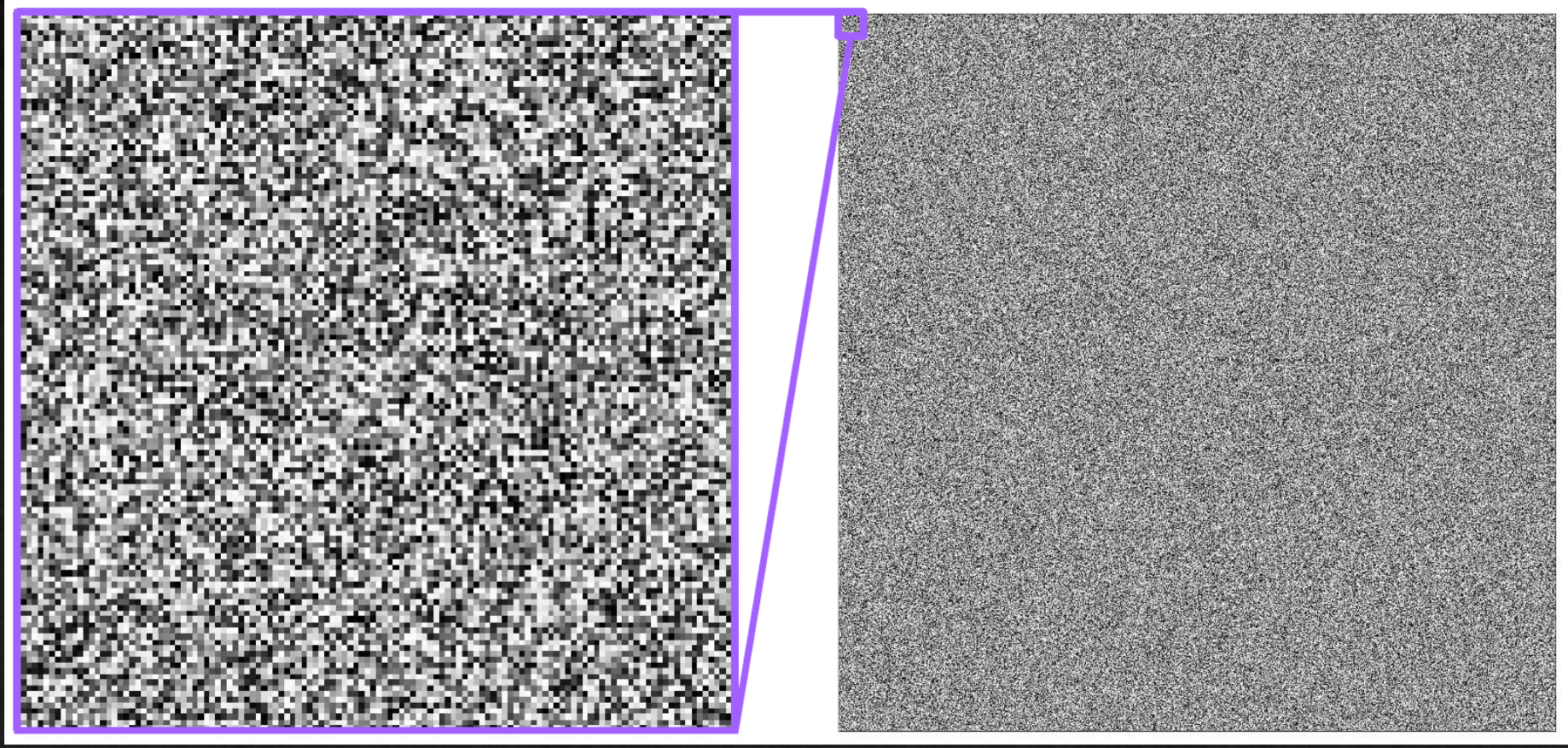
Scalability issues





Scalability issues





Scalability issue fix: N-grams

- Only use condition on N previous words

$$p(x) \approx \Pi_{t=1}^{T} p(x_t | x_{t-N-1}, \dots, x_{t-1})$$

Modeling

Modeling **word**

Modeling word **probabilities**

word probabilities **is**

probabilities is **really**

is really **difficult**

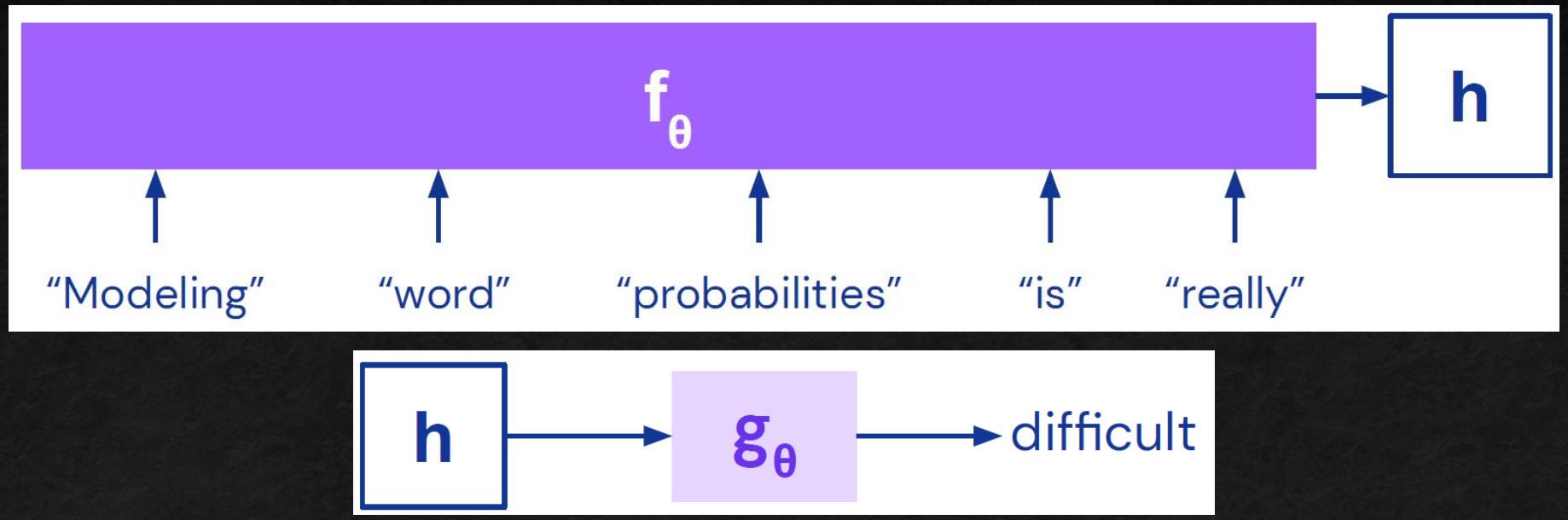
$$p(x_1)$$
$$p(x_2|x_1)$$
$$p(x_3|x_2, x_1)$$
$$p(x_4|x_3, x_2)$$
$$p(x_5|x_4, x_3)$$
$$p(x_6|x_5, x_4)$$

Properties of N-grams as f_θ

	N-gram				
Order matters	✓				
Variable length	X				
Differentiable	X				
Pairwise encoding	✓				
Preserves long-term	X				

Learning model: Addition (probability aggregation)

- Modeling conditional probabilities with a non-linear function g_θ



- Desired properties for g_θ: individual changes can have large effects (non-linear / deep), returns a probability distribution

Properties of Addition as f_θ

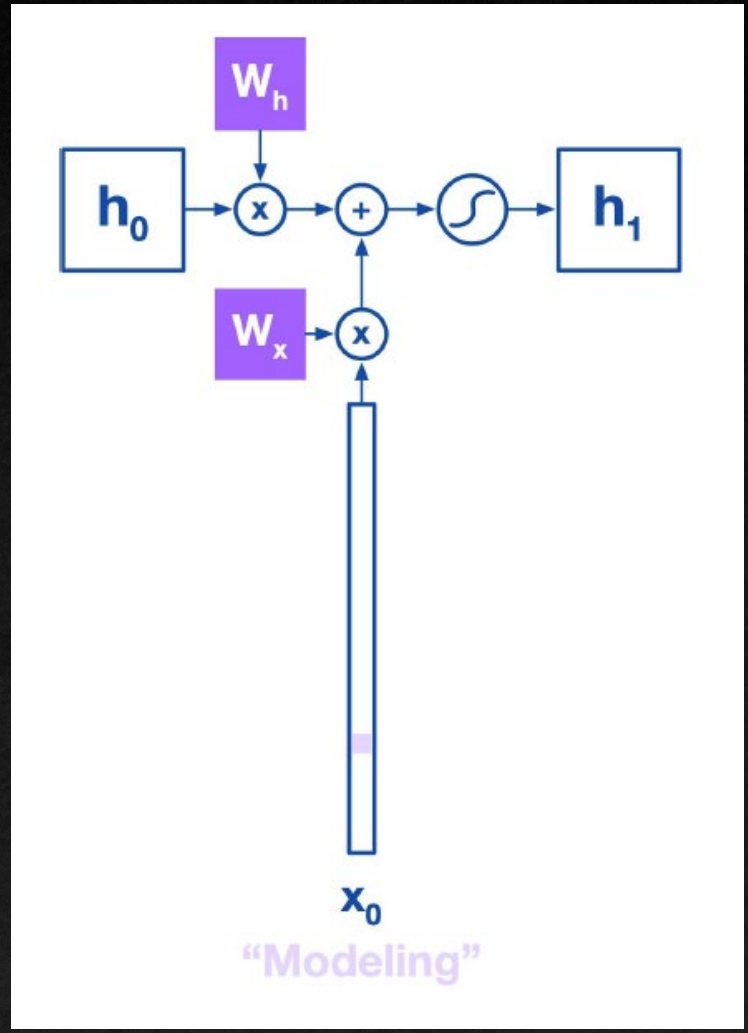
	N-gram	Addition			
Order matters	✓	X			
Variable length	X	✓			
Differentiable	X	✓			
Pairwise encoding	✓	X			
Preserves long-term	X	✓			

Recurrent NNs

- Goal: build a deep network that meets the previous requirements
- Create a persistent state variable h that stores information from the context observed so far

$$h_t = \tanh(W_h h_{t-1} + W_x x_t)$$

- Here W are weights



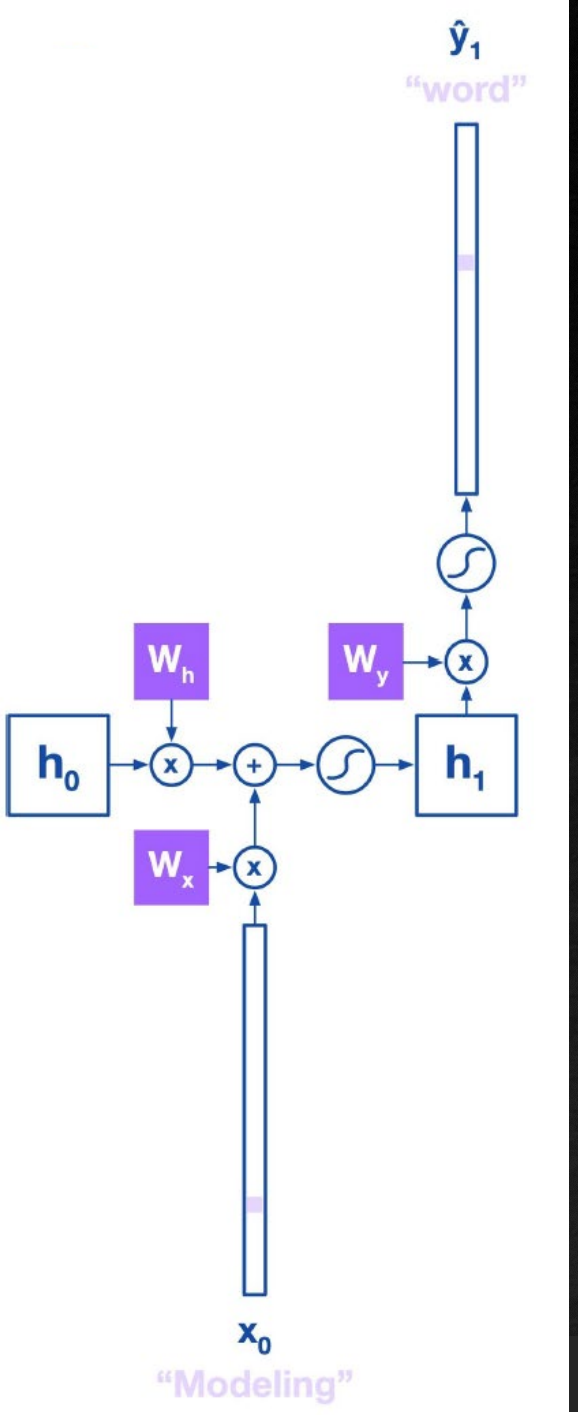
Recurrent NNs

- RNNs predict the target y (the next element in the sequence) from the state h

$$p(y_{t+1}) = softmax(W_y h_t)$$

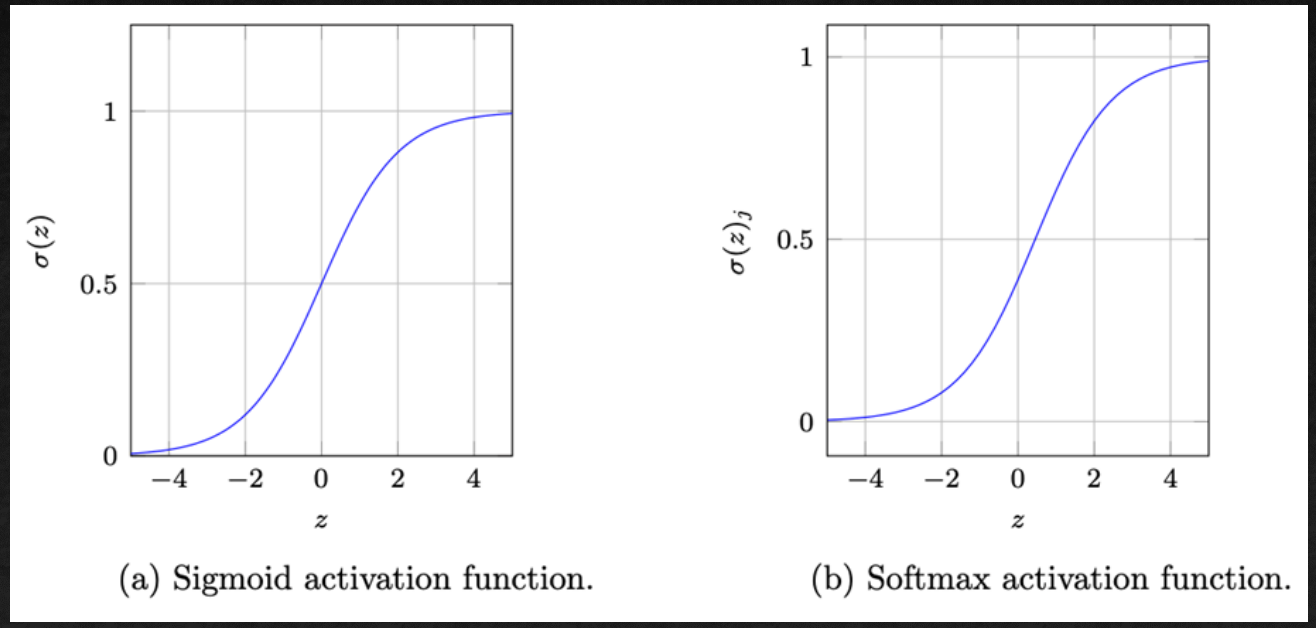
- Softmax is the Boltzmann distribution (non-linear)

$$softmax(z)_i = \frac{\exp(z_i)}{\Sigma_{j=1}^{K} \exp(z_j)}$$



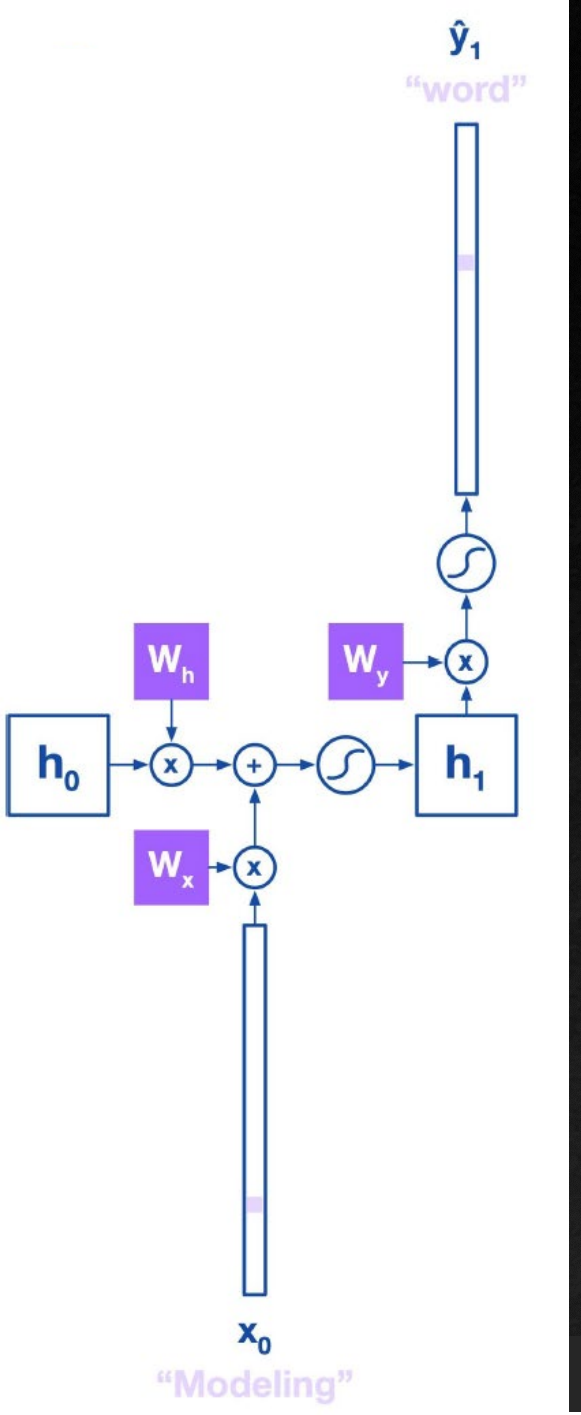
Recurrent NNs

- Sigmoids are usually used for classification
  - Sum of probabilities need not equal 1
- Softmax used for multi-classification
  - Sum of probabilities equals 1



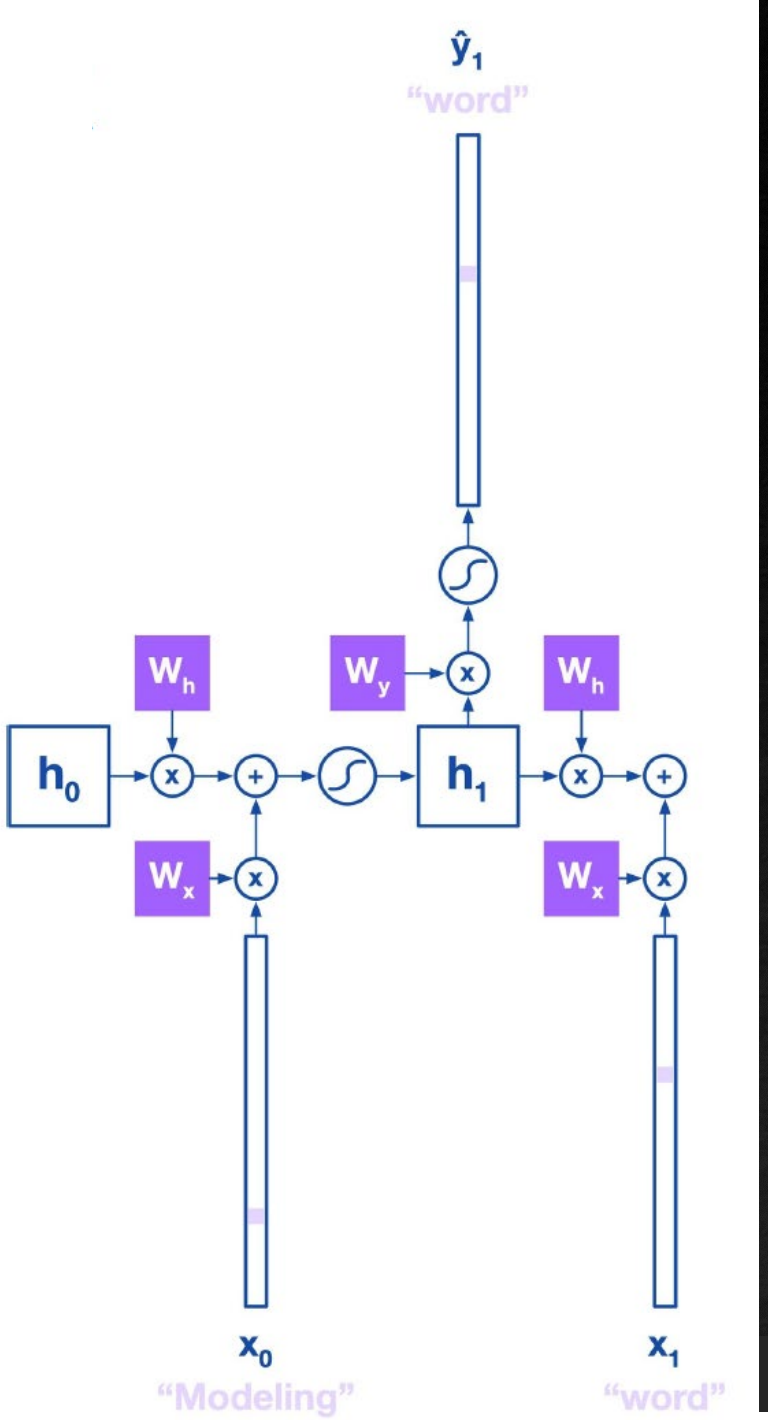
(a) Sigmoid activation function.

(b) Softmax activation function.



Recurrent NNs

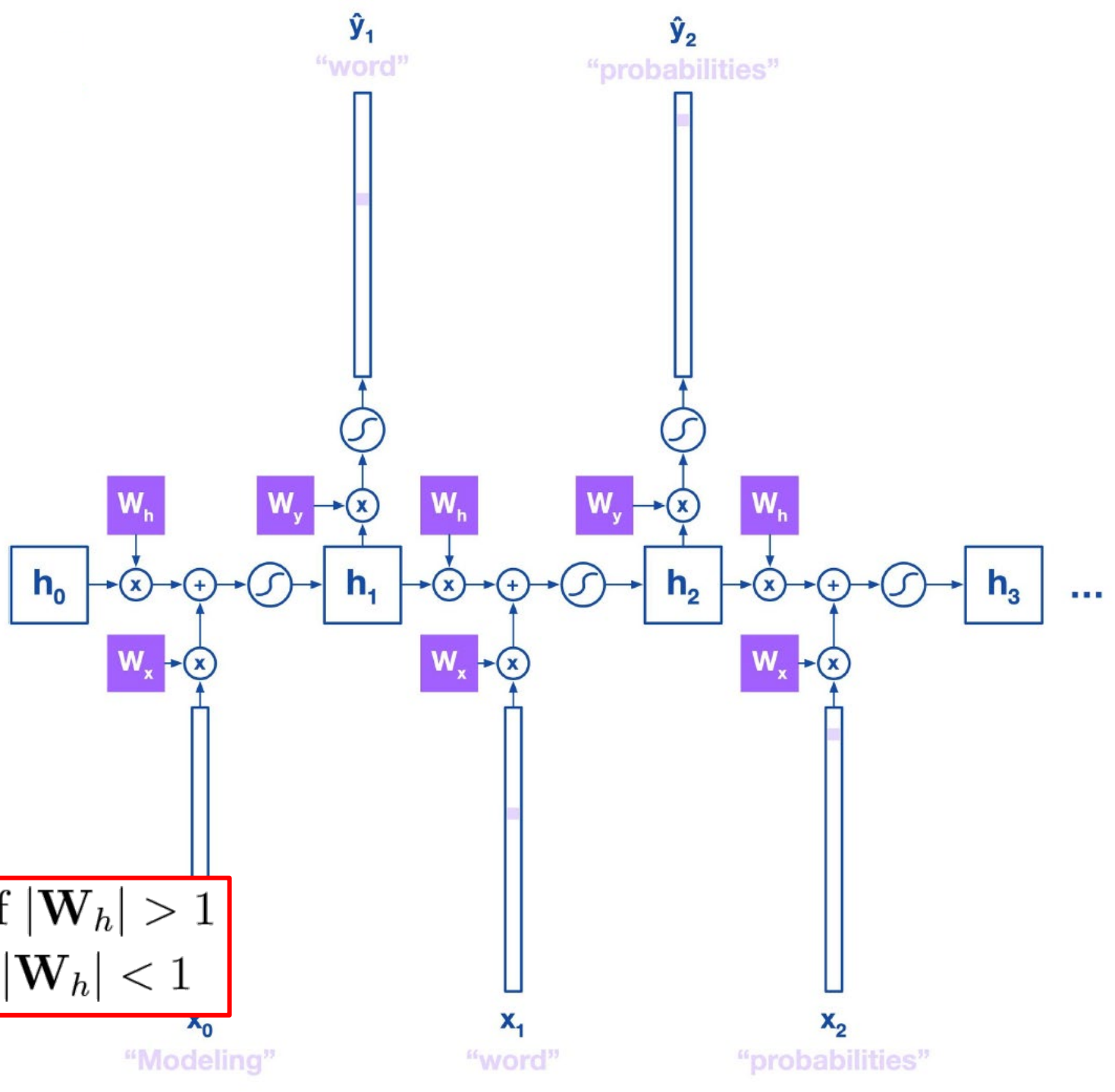
- Input next element in sequence x_1



Recurrent NNs

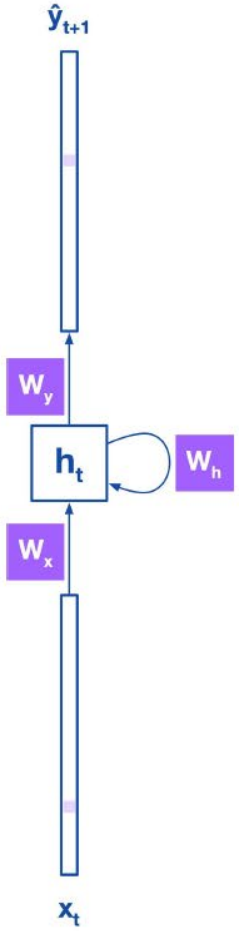
- And so on
- "Memory" is smeared out over time
- The gradient vanishes

$$\mathbf{h}_t = \mathbf{W}_h \mathbf{h}_{t-1} \qquad \mathbf{h}_t \to \infty \text{ if } |\mathbf{W}_h| > 1$$
$$\mathbf{h}_t = (\mathbf{W}_h)^t \mathbf{h}_0 \qquad \mathbf{h}_t \to 0 \text{ if } |\mathbf{W}_h| < 1$$

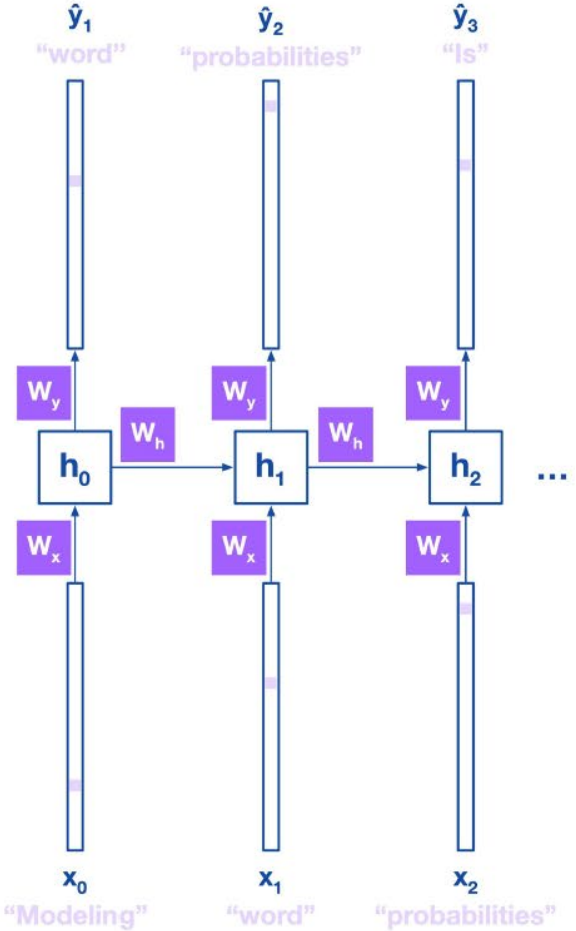


Recurrent NNs

Weights are shared over time steps



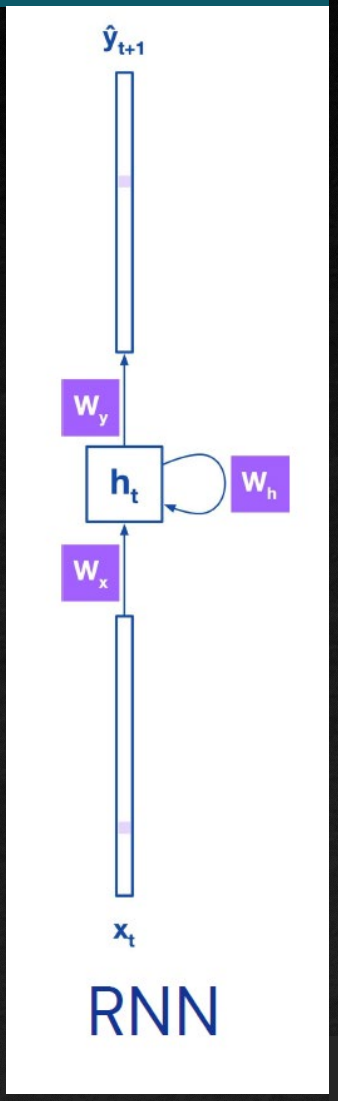
RNN



RNN rolled out over time

Differentiating with respect to W

- Next element prediction is a classification task where the number of classes is the size of the possible choices
- So we use the cross-entropy loss
- For one element: $\mathcal{L}_\theta(y, \hat{y})_t = -y_t \log \hat{y}_t$
- For a sequence: $\mathcal{L}_\theta(y, \hat{y}) = -\sum_{t=1}^{T} y_t \log \hat{y}_t$
- With parameters: $\theta = \{W_y, W_x, W_h\}$

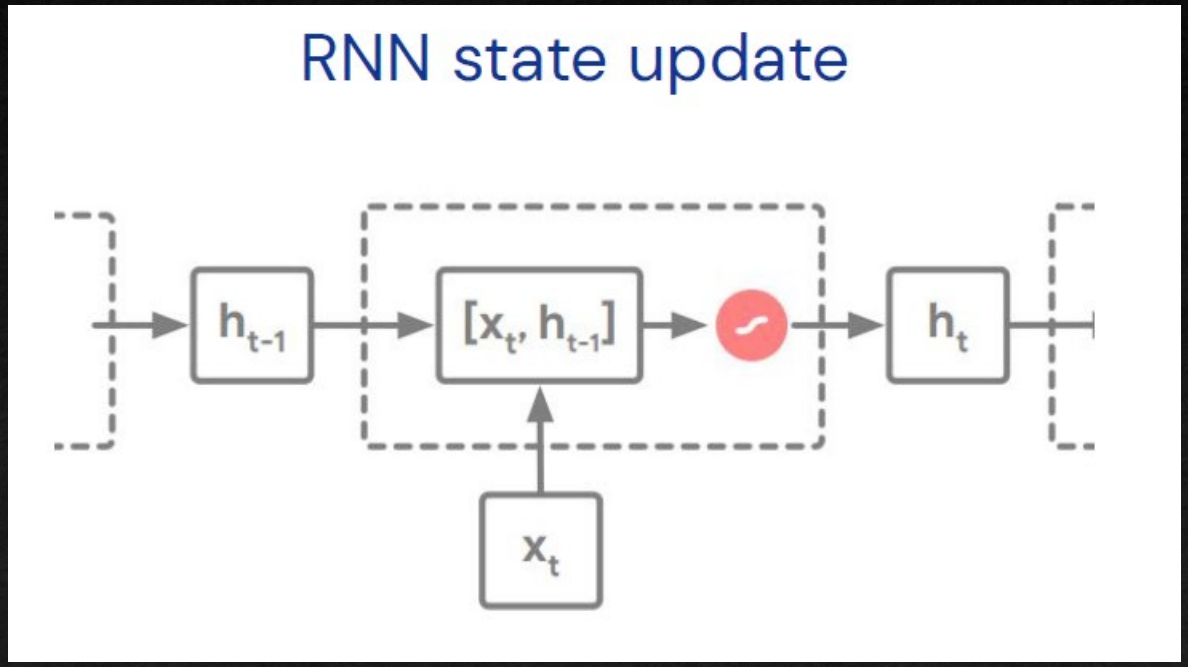


Properties of RNNs as f_θ

	N-gram	Addition	RNN		
Order matters	✓	X	✓		
Variable length	X	✓	✓		
Differentiable	X	✓	✓		
Pairwise encoding	✓	X	X		
Preserves long-term	X	✓	X		

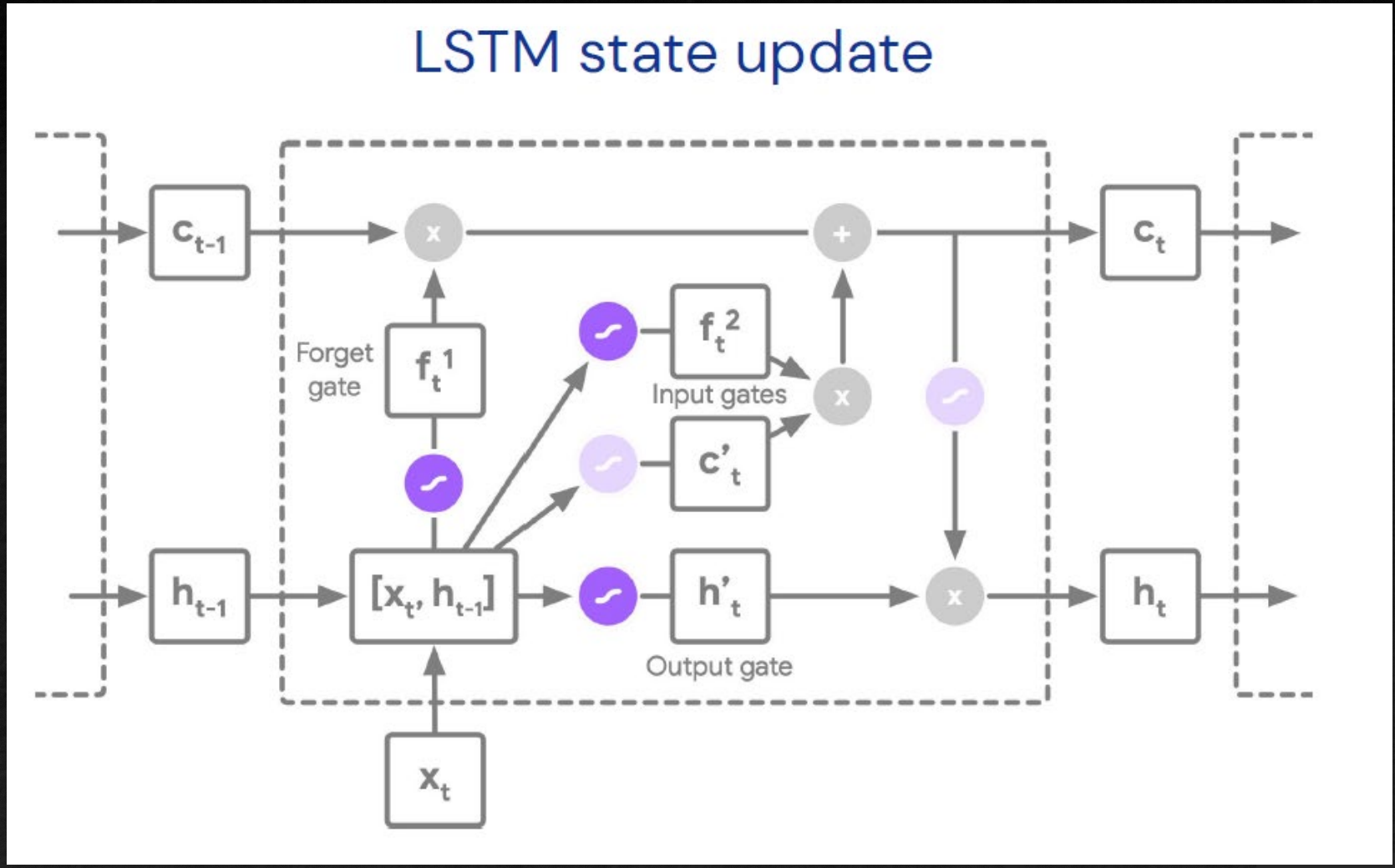
Capturing long-term behavior

- Enter the Long Short-Term Memory (LSTM) network
- Introduces a memory cell into the RNN learning algorithm
- Control the flow of information into the cell with input gates
- Allow for the memory cell to forget information



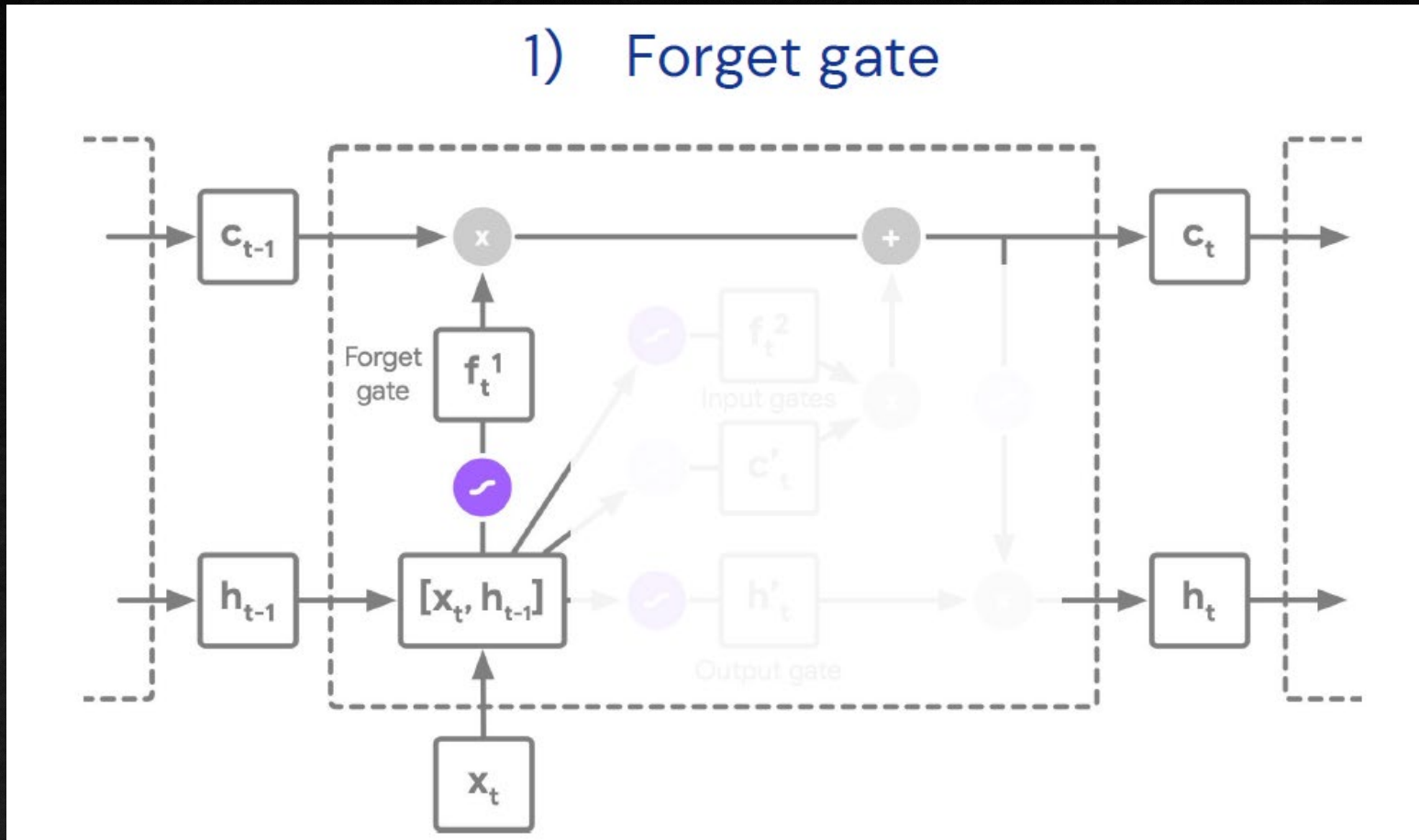
Capturing long-term behavior

- Enter the Long Short-Term Memory (LSTM) network
- Introduces a memory cell into the RNN learning algorithm
- Control the flow of information into the cell with input gates
- Allow for the memory cell to forget information



Capturing long-term behavior

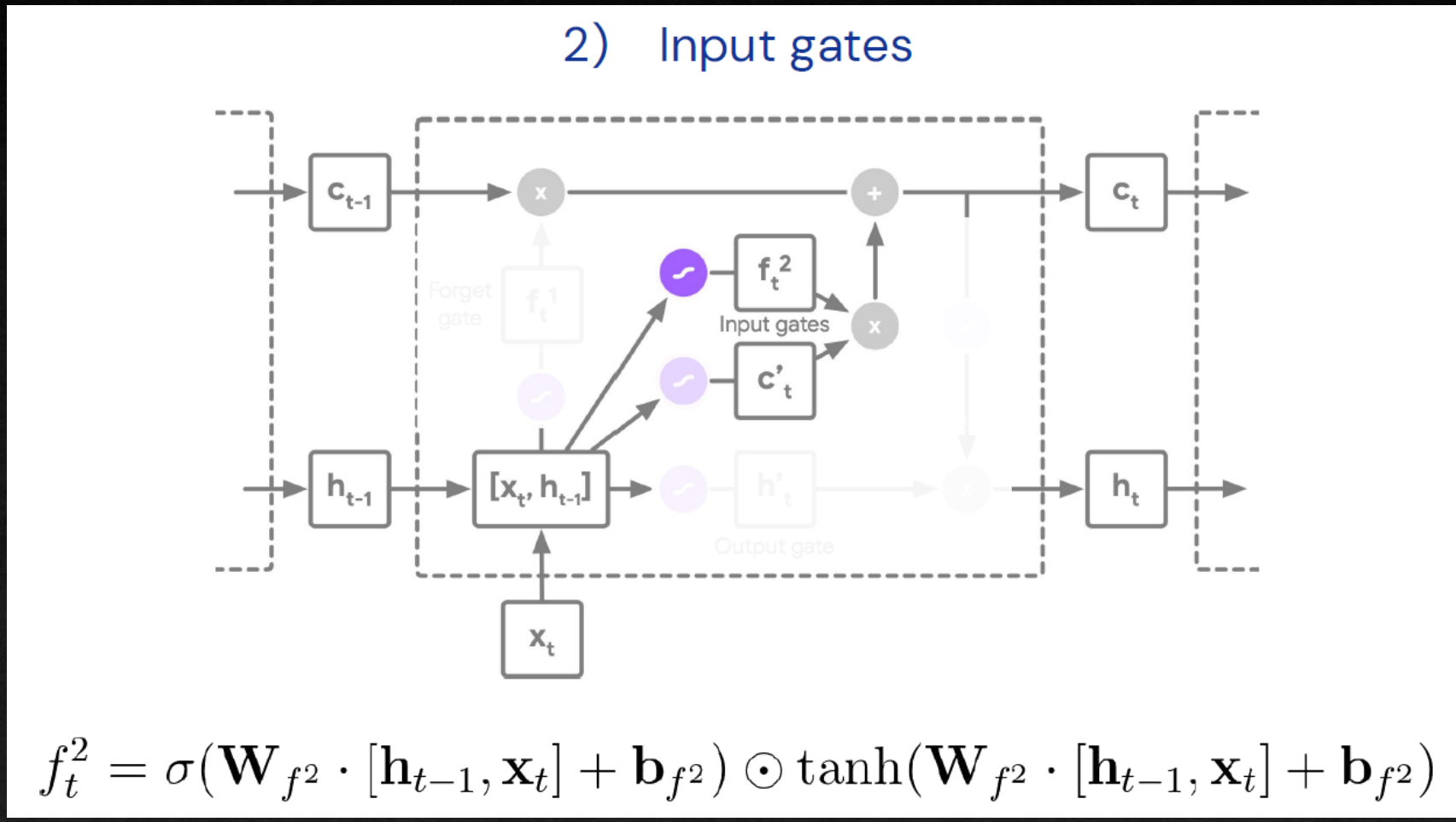
- Enter the Long Short-Term Memory (LSTM) network
- Introduces a memory cell into the RNN learning algorithm
- Control the flow of information into the cell with input gates
- Allow for the memory cell to forget information



$$f_t^1 = \sigma(\mathbf{W}_{f^1} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{f^1})$$

Capturing long-term behavior

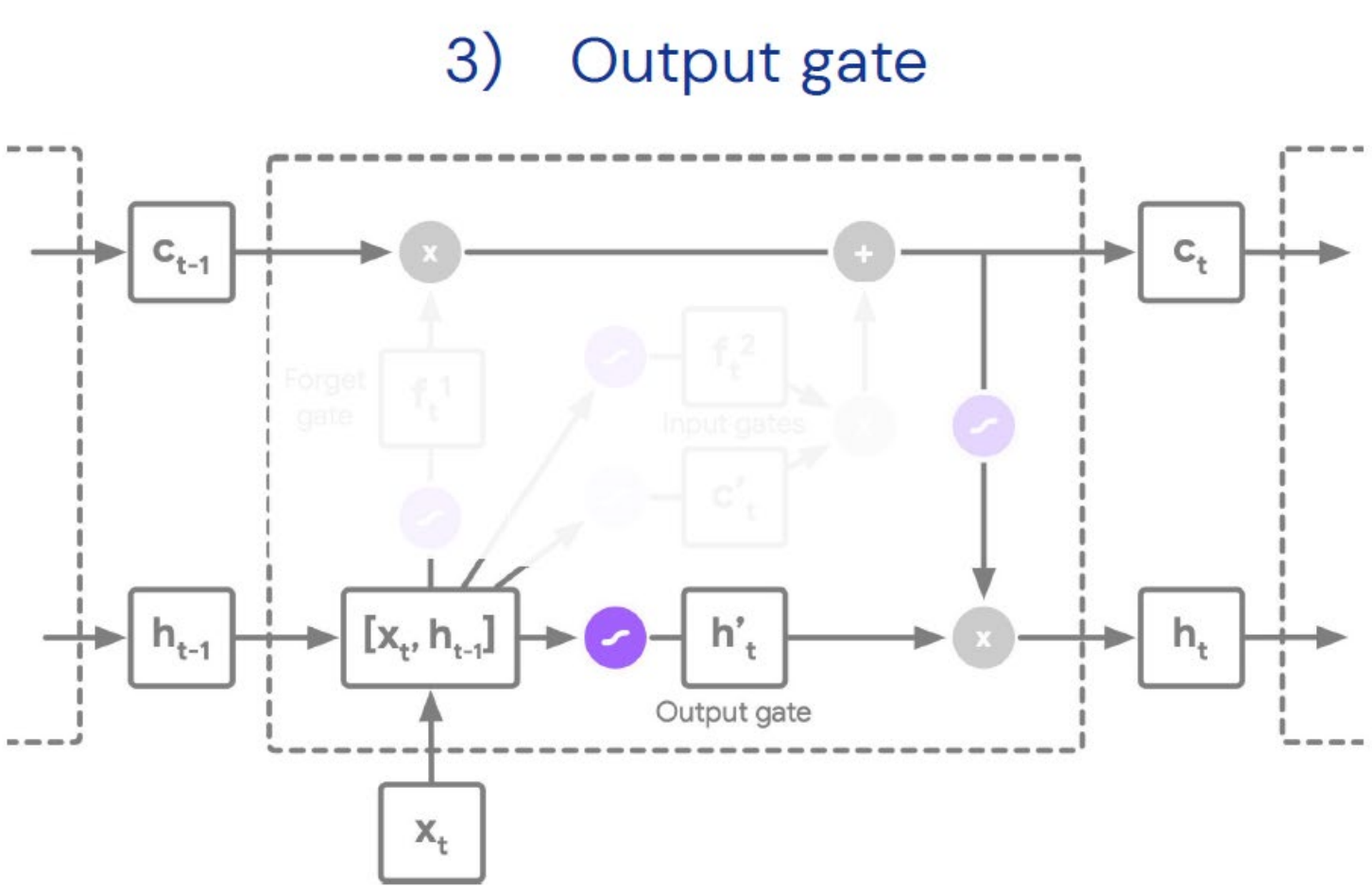
- Enter the Long Short-Term Memory (LSTM) network
- Introduces a memory cell into the RNN learning algorithm
- Control the flow of information into the cell with input gates
- Allow for the memory cell to forget information



$$f_t^2 = \sigma(\mathbf{W}_{f^2} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{f^2}) \odot \tanh(\mathbf{W}_{f^2} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{f^2})$$

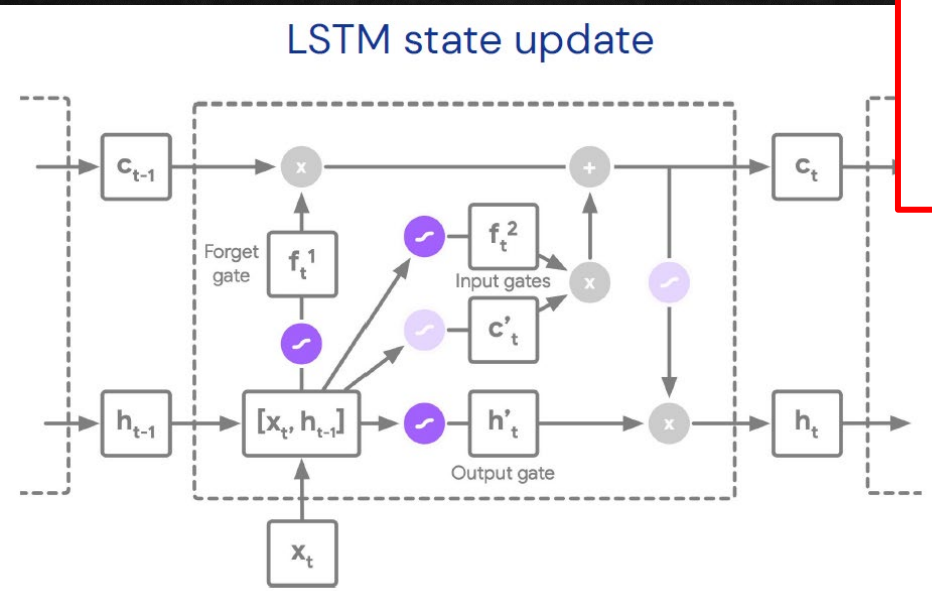
Capturing long-term behavior

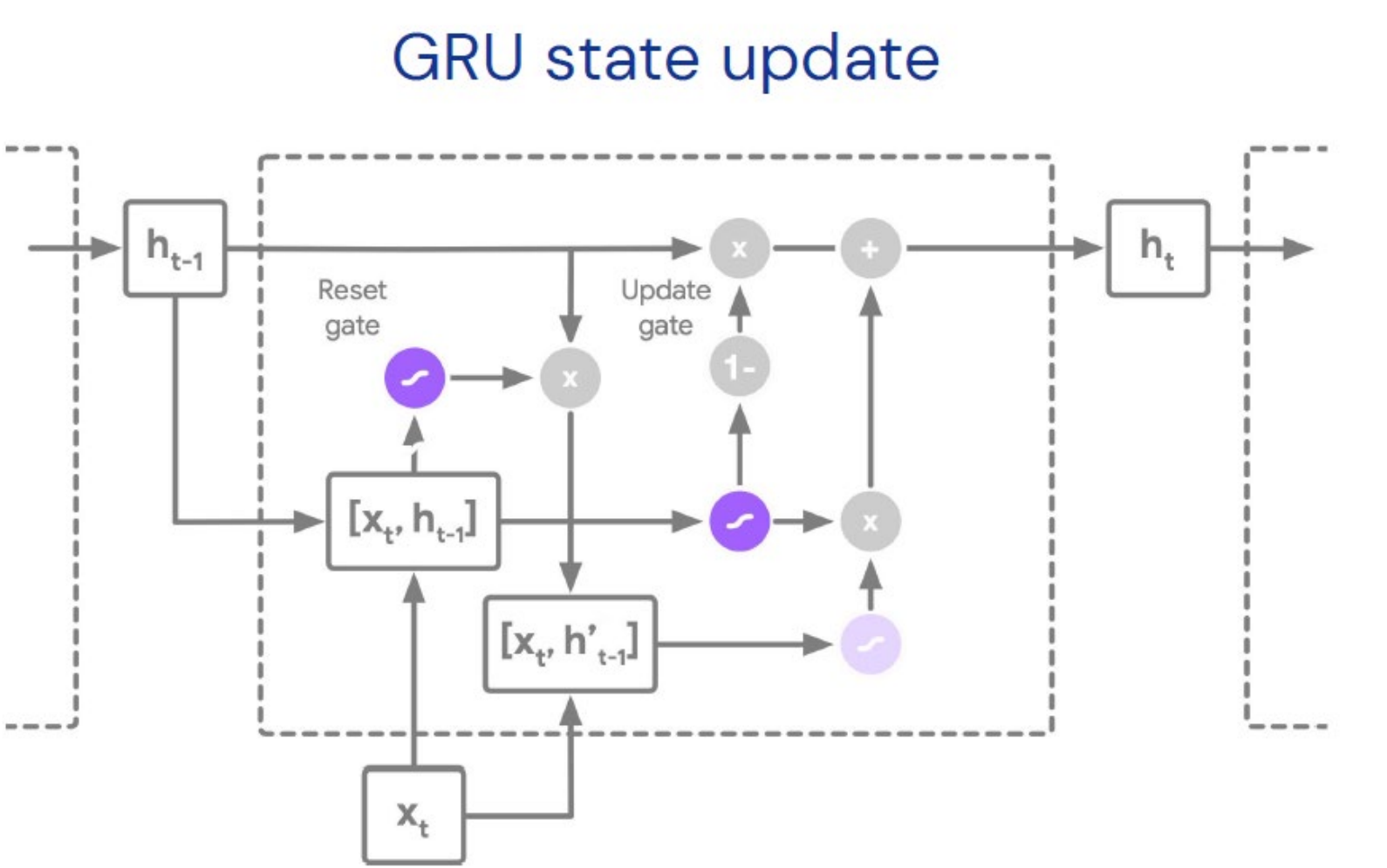
- Enter the Long Short-Term Memory (LSTM) network
- Introduces a memory cell into the RNN learning algorithm
- Control the flow of information into the cell with input gates
- Allow for the memory cell to forget information



$$h'_t = \sigma(\mathbf{W}_{h'_t} \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_{h'_t}) \odot \tanh(\mathbf{c}_t)$$

- Gated Recurrent Unit NNs are a simplified LSTM
- No explicit memory cell
- No output gate

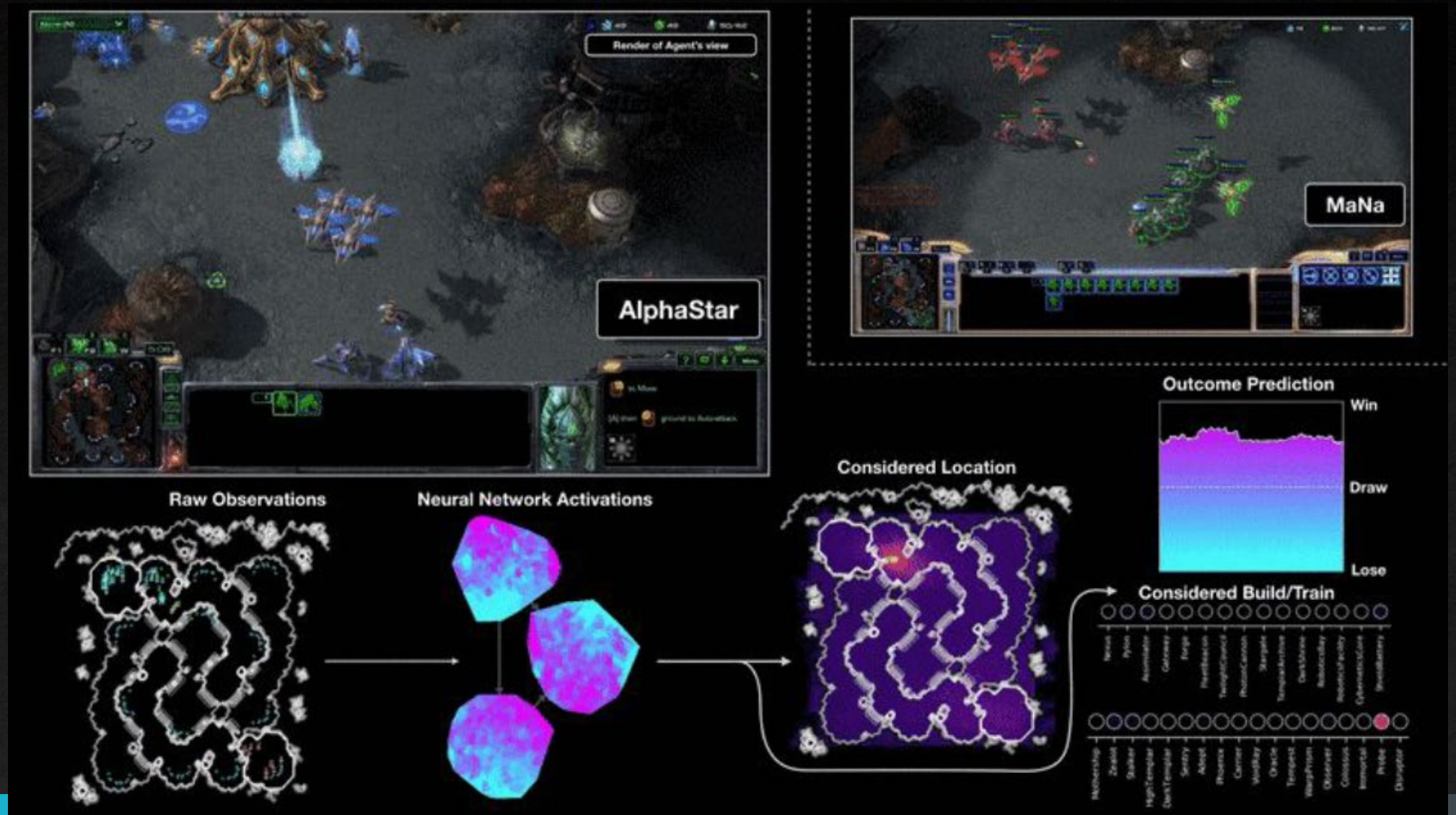




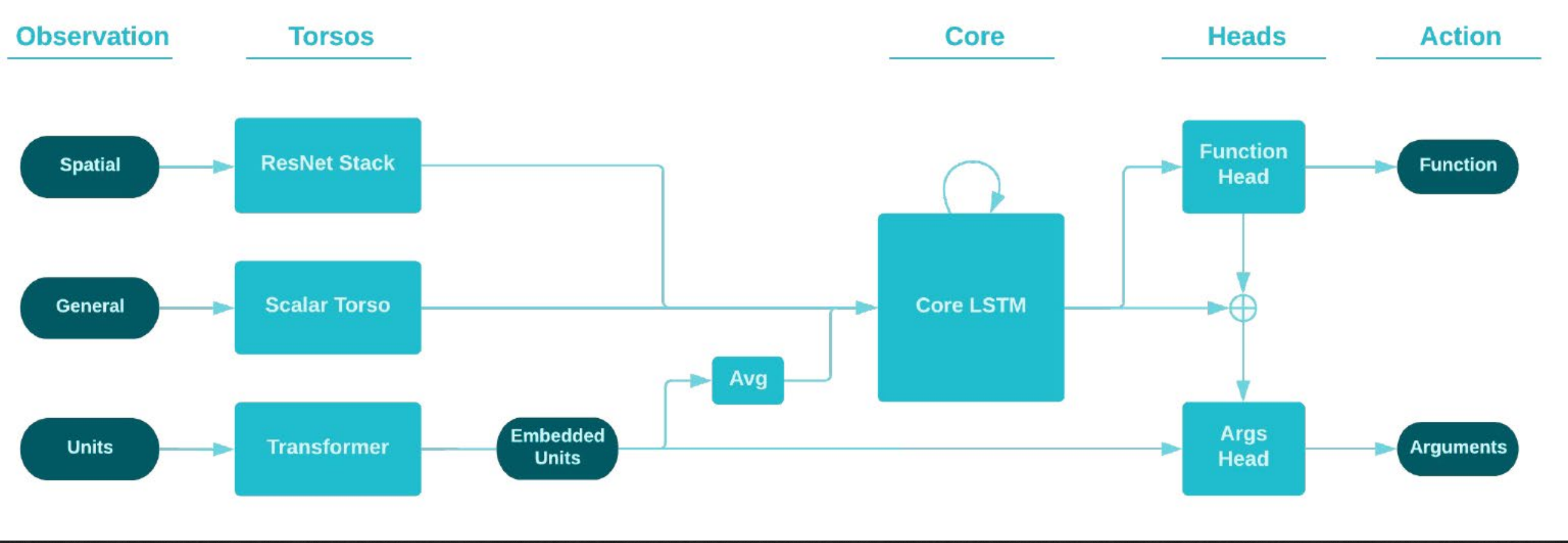
Properties of LSTMs as f_θ

	N-gram	Addition	RNN	LSTM	
Order matters	✓	X	✓	✓	
Variable length	X	✓	✓	✓	
Differentiable	X	✓	✓	✓	
Pairwise encoding	✓	X	X	X	
Preserves long-term	X	✓	X	✓	

Alphastar (2019): LSTMs in StarCraft 2



Alphastar (2019): LSTMs in StarCraft 2



Attention is all you need

Transformers

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com

mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to

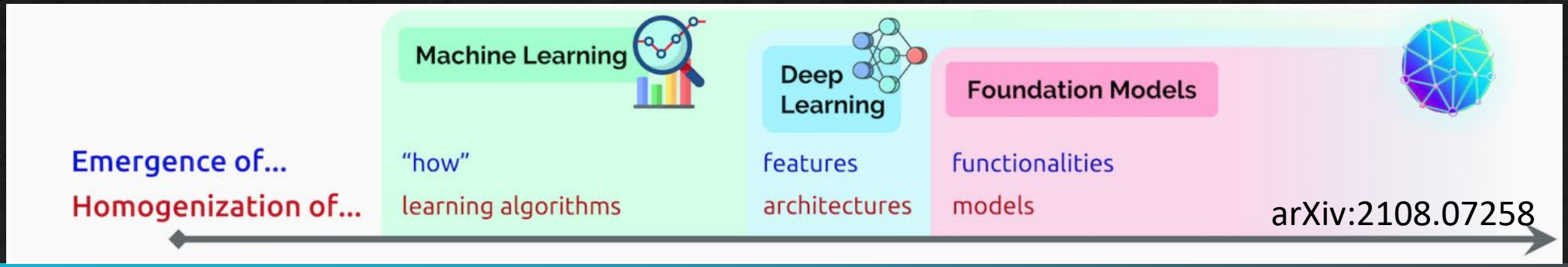
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Transformers

- NN that learns context and thus meaning
- Tracks relationships in sequential data, like words in a sentence
- Applies an evolving set of operations, called attention
  - Detects subtle ways that distant data elements can influence and depend on each other

- Now known as a self-supervised method or a “foundational” method



Transformers

- No labels needed
- Learns by itself by discovering patterns
- Allows for the input database to dramatically increase

- The network architecture and self-supervised nature allows for easy parallel processing

- Now the most popular model, replacing RNNs and CNNs from 5 years ago

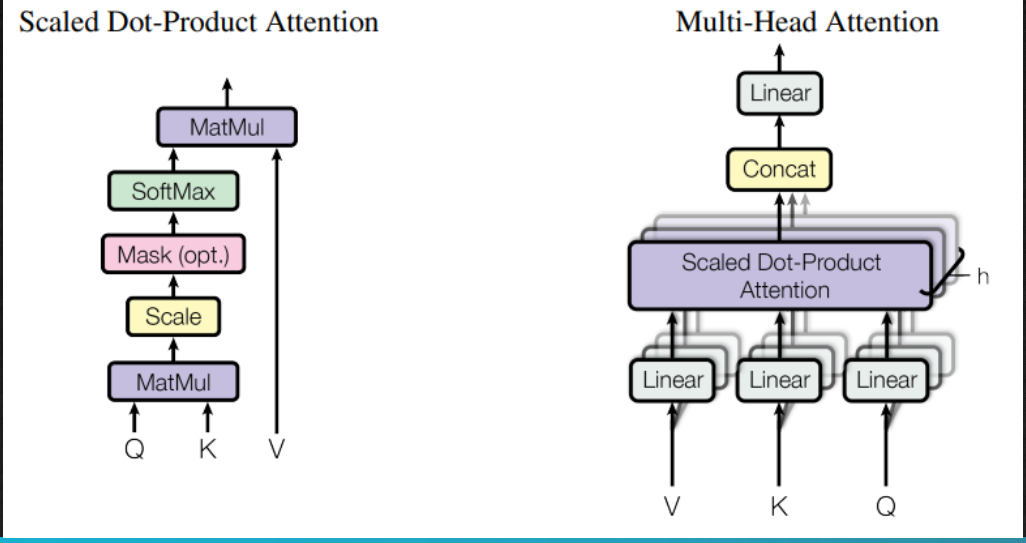
Transformers

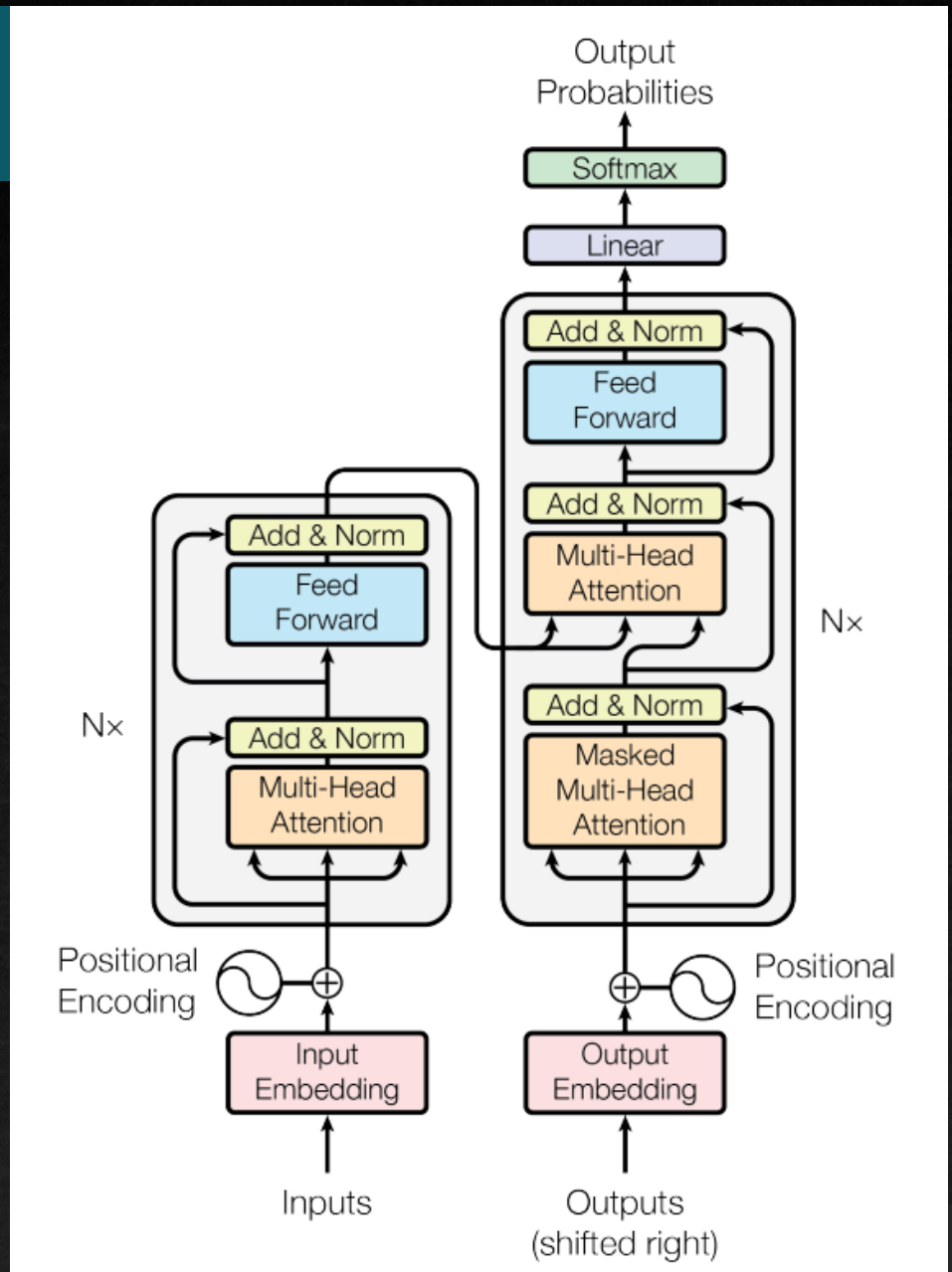
- Use positional encoders to tag data elements passing through the network

$$PE_{pos,2i} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$

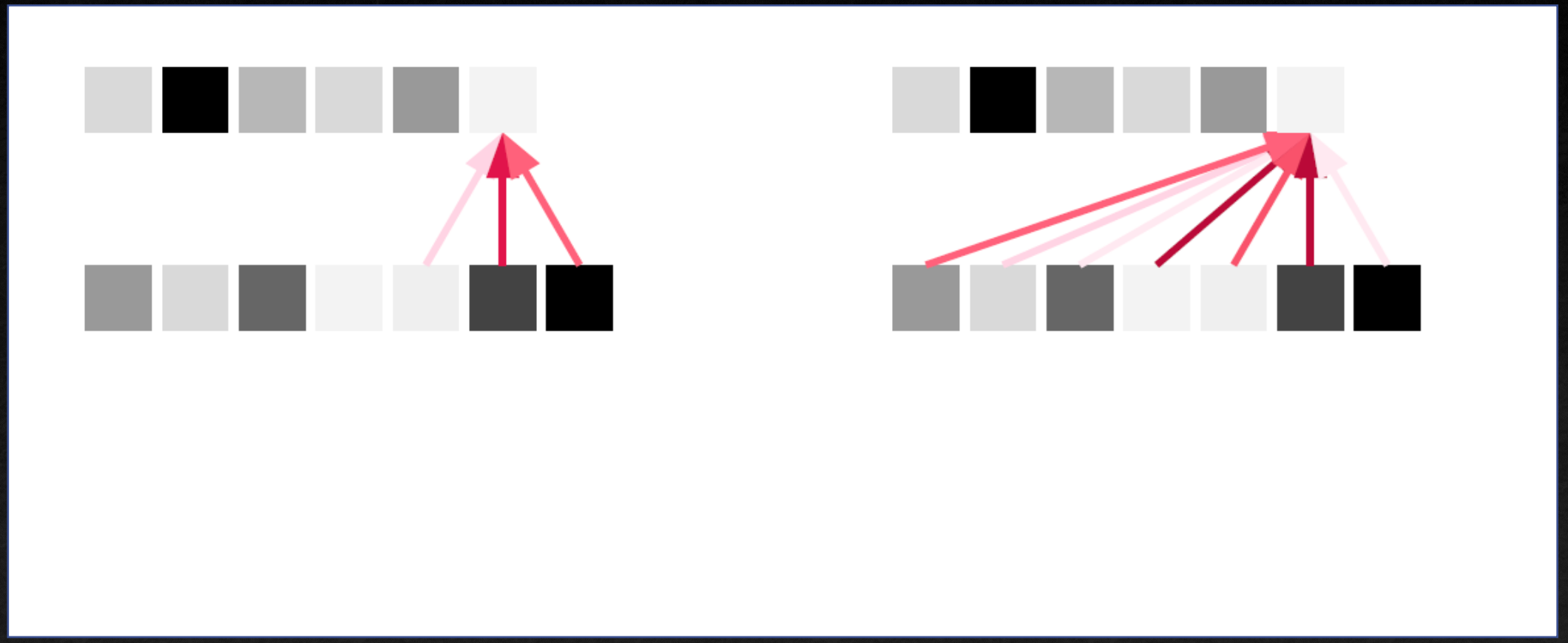
$$PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

- Attention units follow these tags, calculating a correlation map of how each element relates to each other





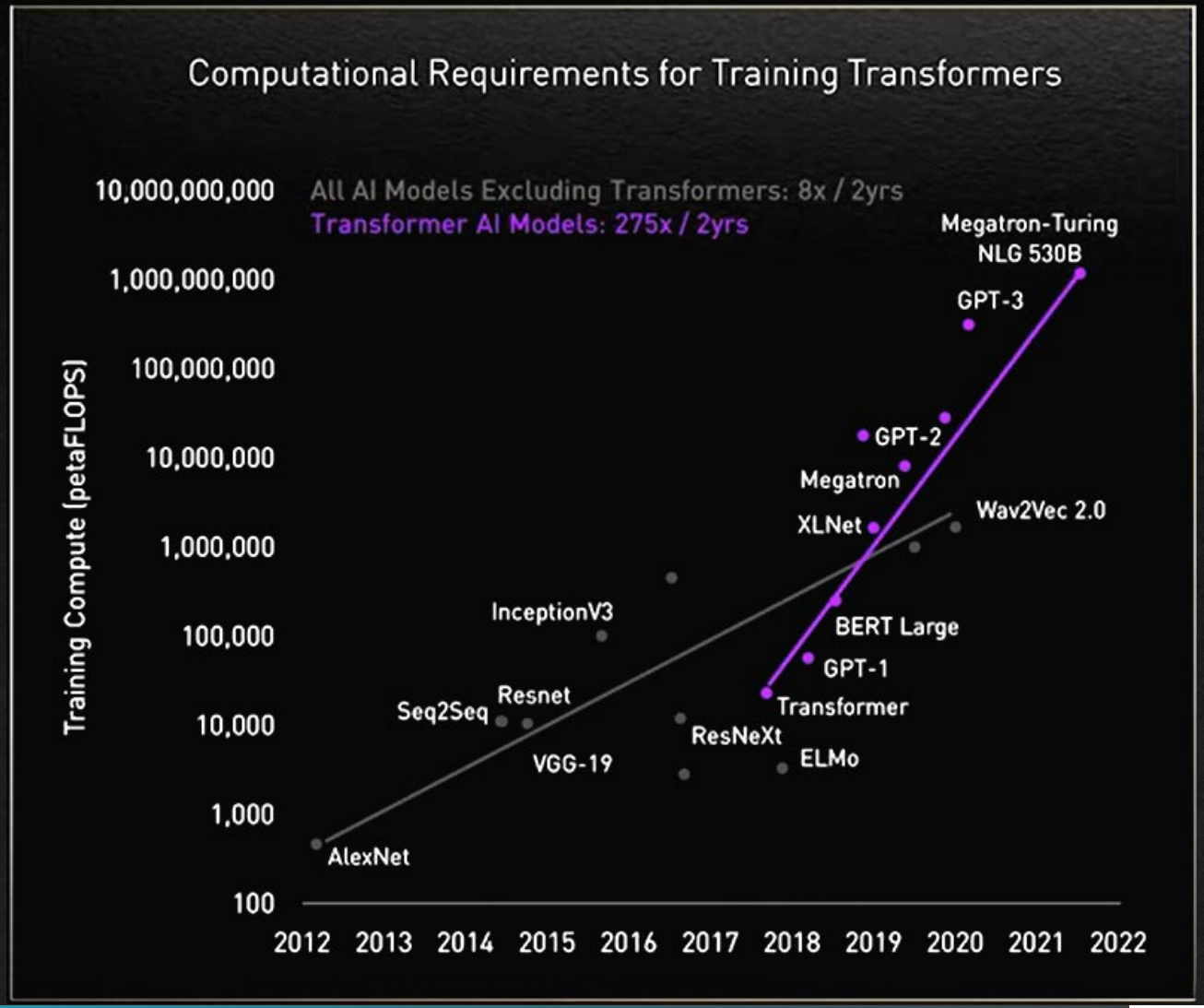
Transformers



Transformers

GPT-4?

- Regular NNs improves with more input dimensions
- Deep learning (RNNs / CNNs) improves with depth
- Transformers improves with input size, i.e. parameters
- In an era of large-scale ML



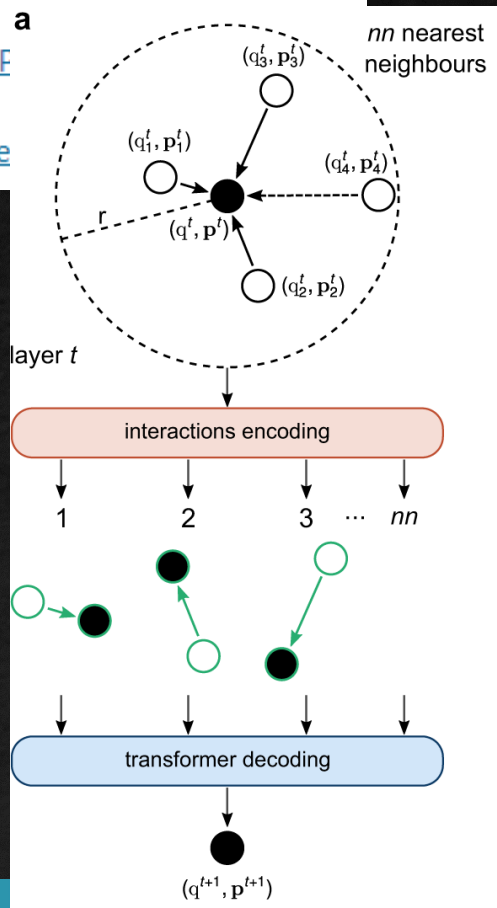
Transformers: Protein sequences

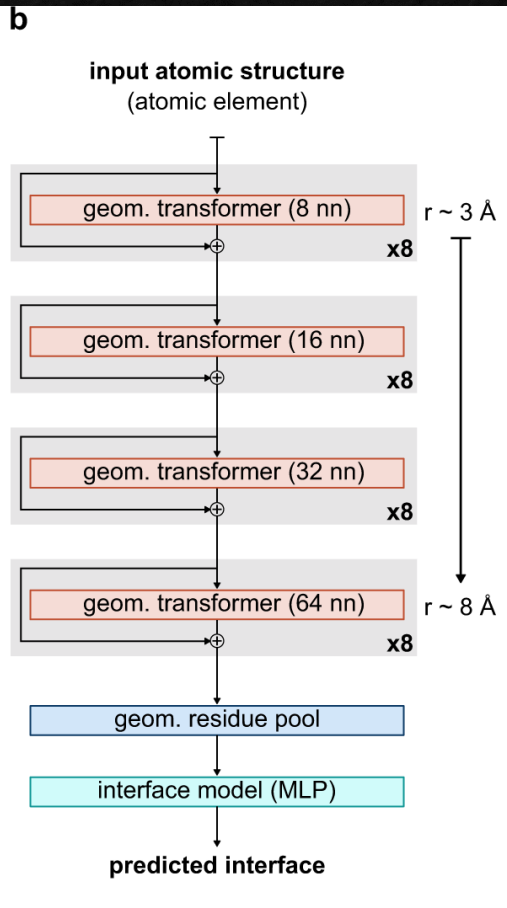
Article | Open Access | Published: 18 April 2023

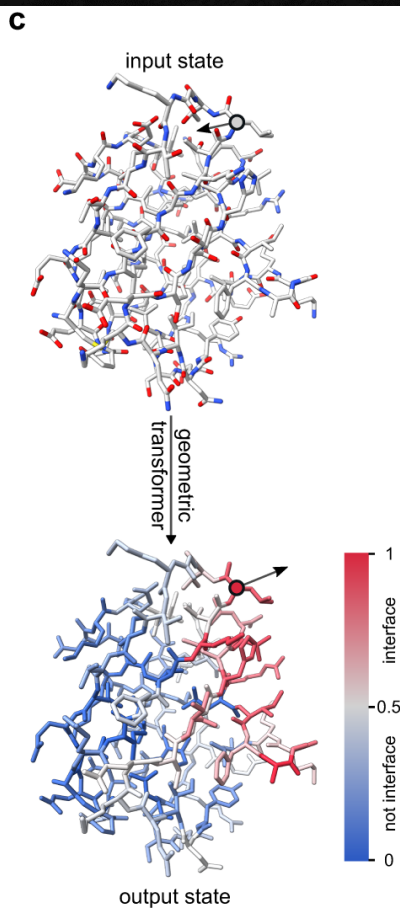
PeSTo: parameter-free geometric deep learning for accurate prediction of protein binding interfaces

Lucien F. Krapp, Luciano A. Abriata, Fabio Cortés Rodriguez & Matteo Dal P

Nature Communications **14**, Article number: 2175 (2023) | Cite this article

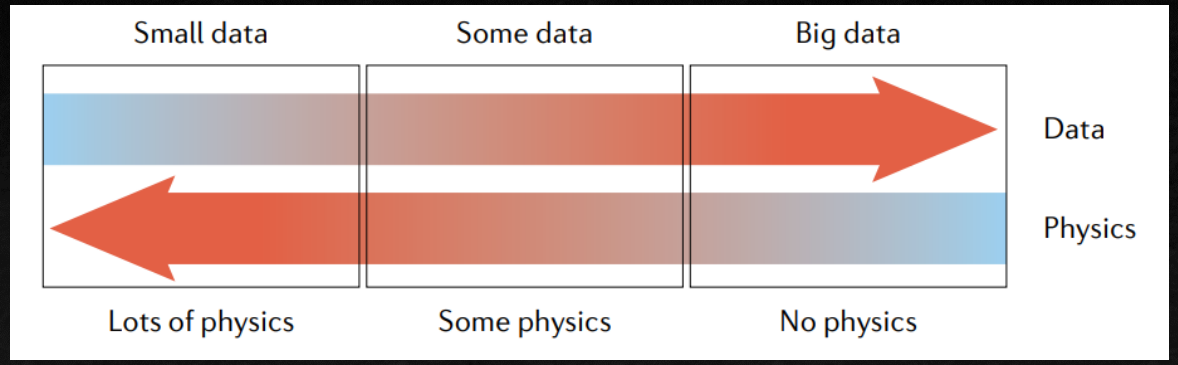






Physics-informed ML

- Solving PDEs can be difficult
  - Noisy data
  - Mesh generation can be complex
  - High dimensions are intractable

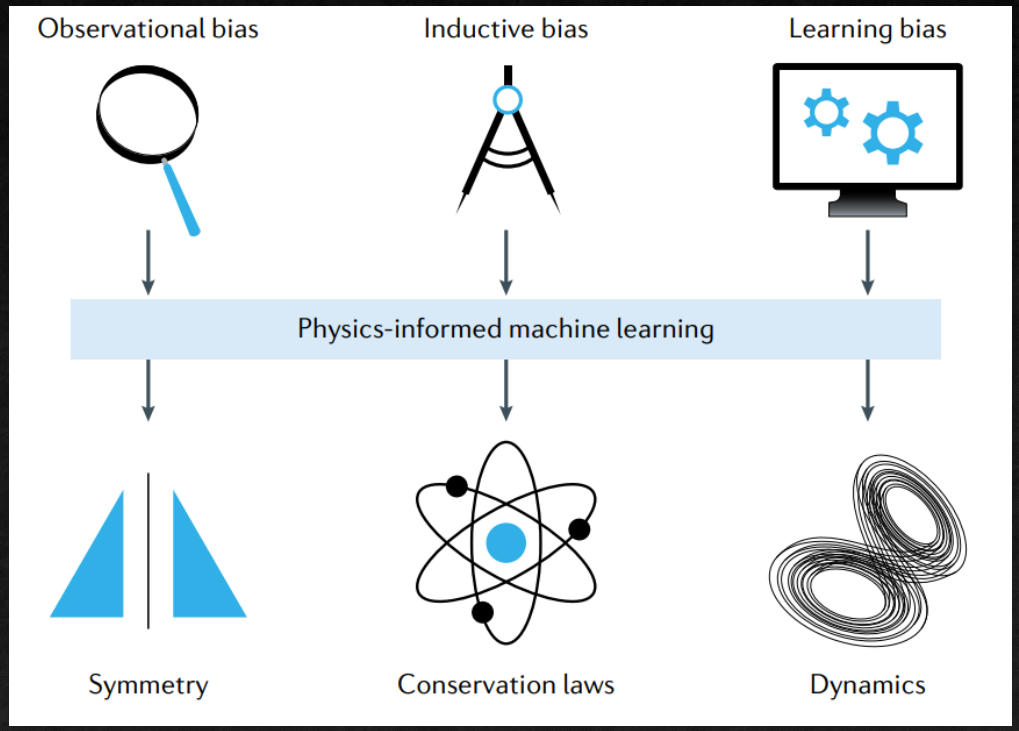


- Design NN architectures that incorporate physics constraints and physical equations

- Can tackle both forward modeling and inverse problems
- Possibly discovering hidden physics and tackling high-dimensional problems

Physics-informed ML: biases

- Observations: data and simulations
- Inductive: predictions are constrained by physical laws
  - Limited by symmetries that are known beforehand
  - May lead to complex implementations that are difficult to scale
- Learning: Choose loss function, constraints, and inference to converge to solutions that adhere to the underlying physics

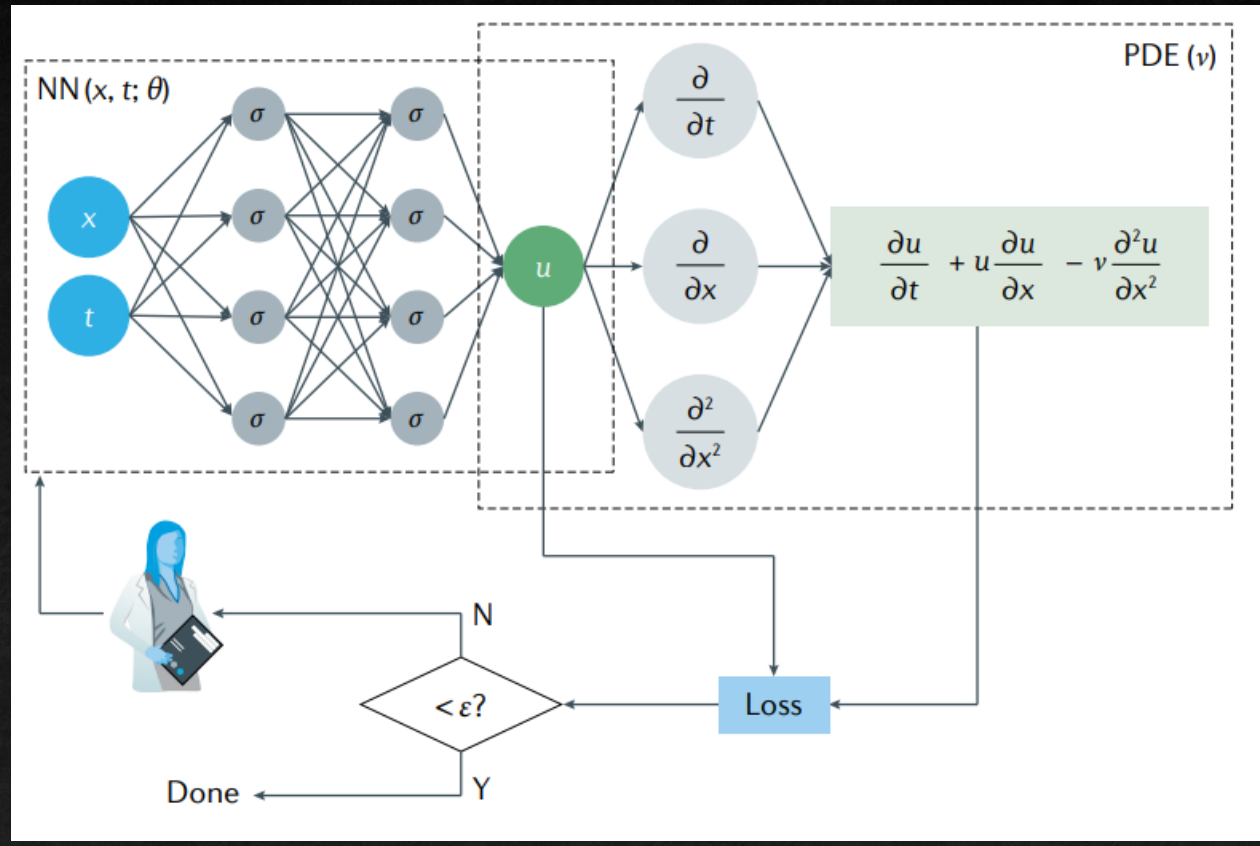


Physics-informed NN Example: Burger's equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}$$

- Compute loss from both the data (Ics and BCs) and PDE

$$\mathcal{L} = w_{data}\mathcal{L}_{data} + w_{PDE}\mathcal{L}_{PDE}$$

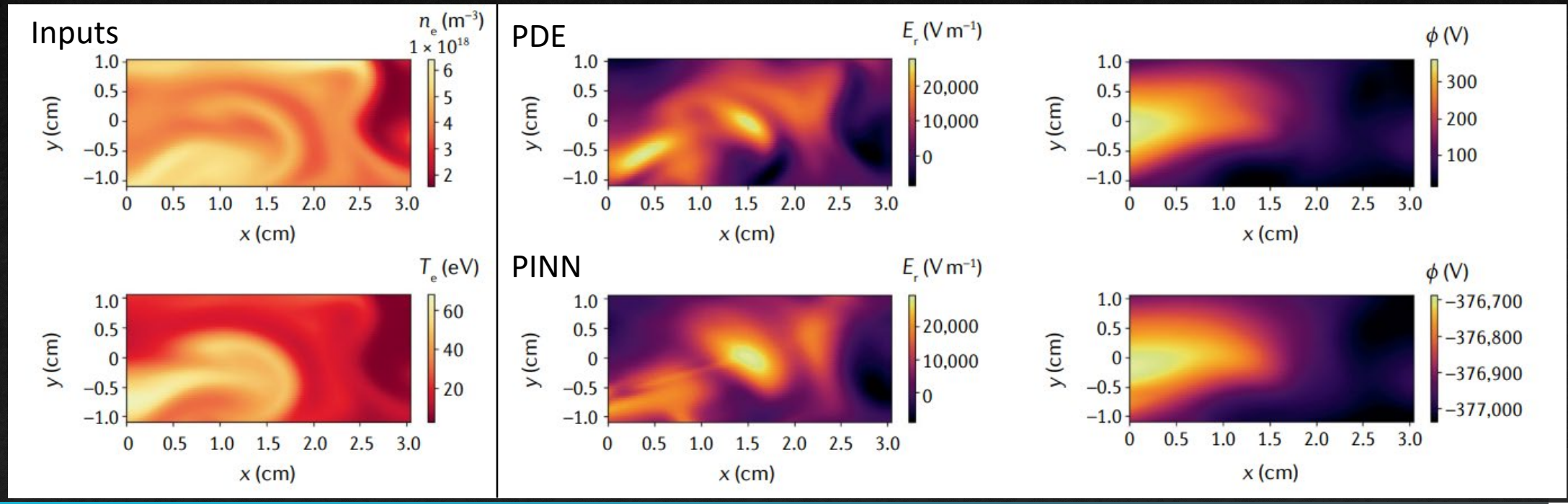


Physics-informed NNs

Table 1 | **Major software libraries specifically designed for physics-informed machine learning**

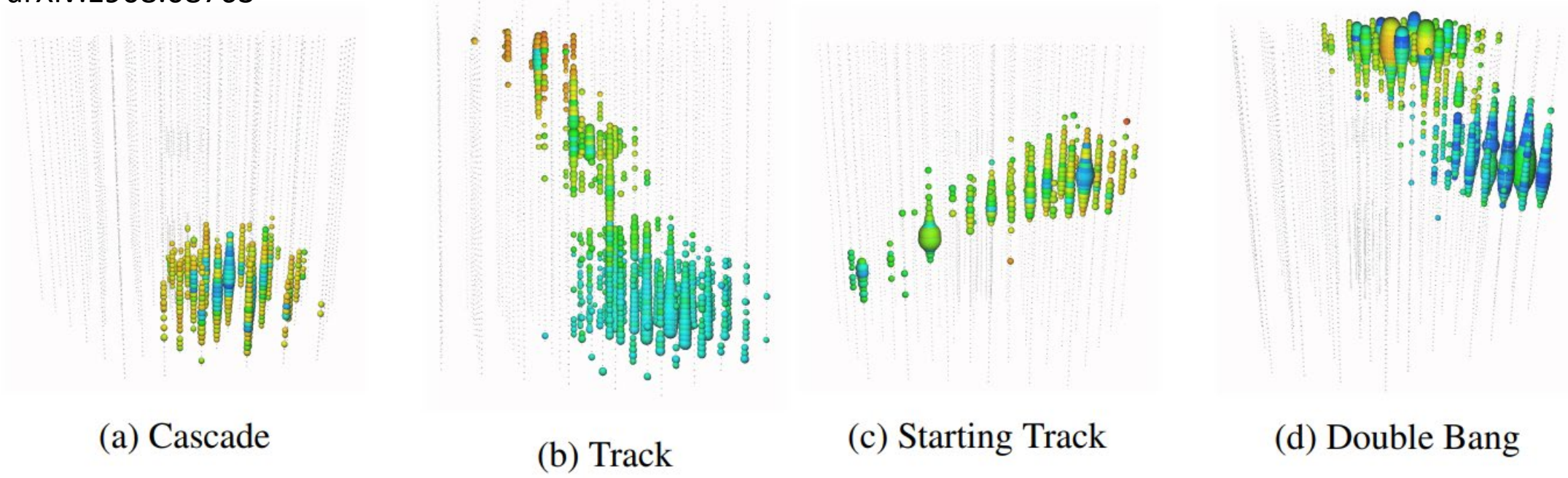
Software name	Usage	Language	Backend	Ref.
DeepXDE	Solver	Python	TensorFlow	154
SimNet	Solver	Python	TensorFlow	155
PyDEns	Solver	Python	TensorFlow	156
NeuroDiffEq	Solver	Python	PyTorch	157
NeuralPDE	Solver	Julia	Julia	158
SciANN	Wrapper	Python	TensorFlow	159
ADCME	Wrapper	Julia	TensorFlow	160
GPyTorch	Wrapper	Python	PyTorch	161
Neural Tangents	Wrapper	Python	JAX	162

3D MHD plasma simulation



Example: IceCube event reconstruction with ML

arXiv:1908.08763

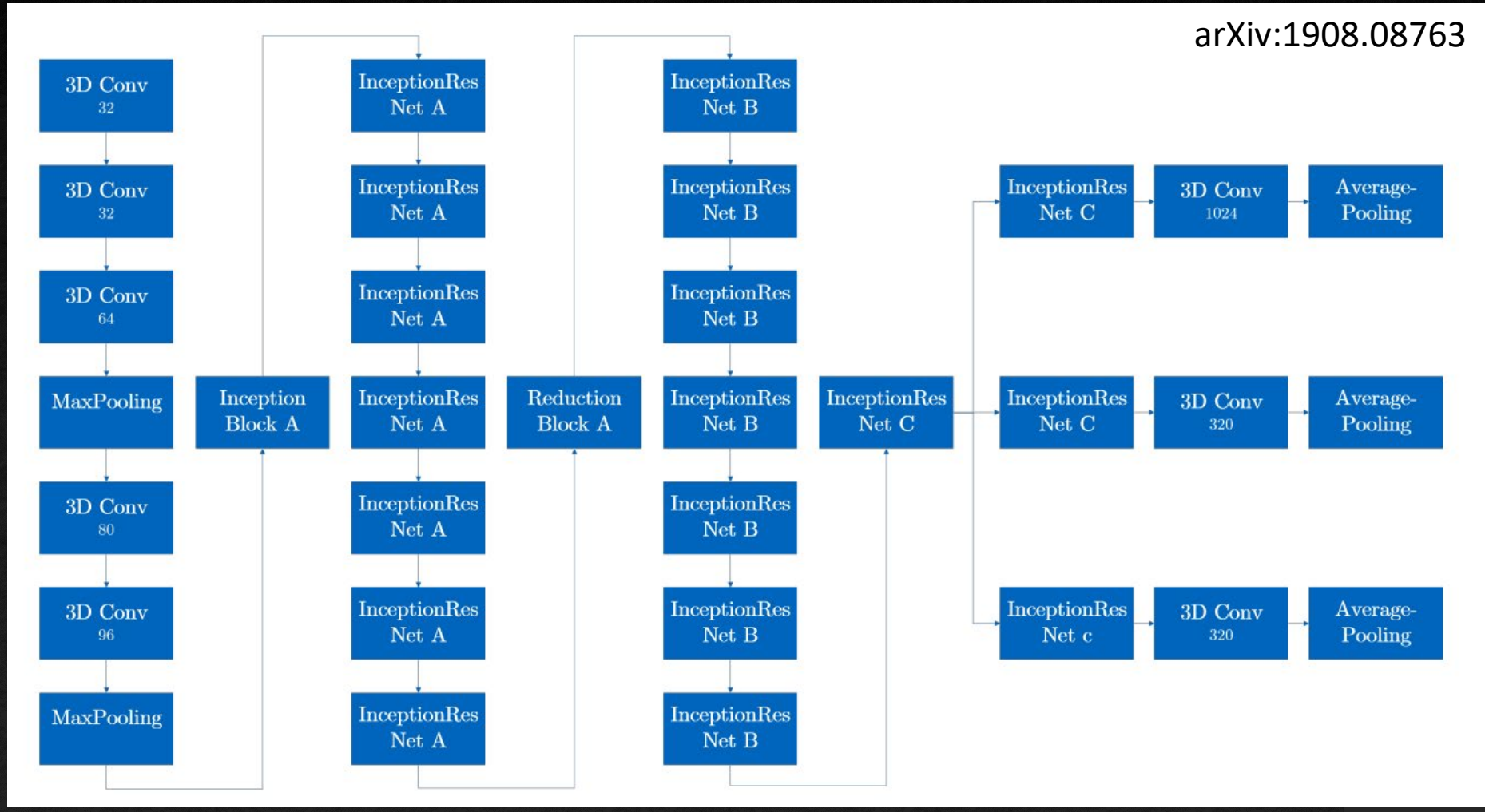


(a) Cascade (b) Track (c) Starting Track (d) Double Bang

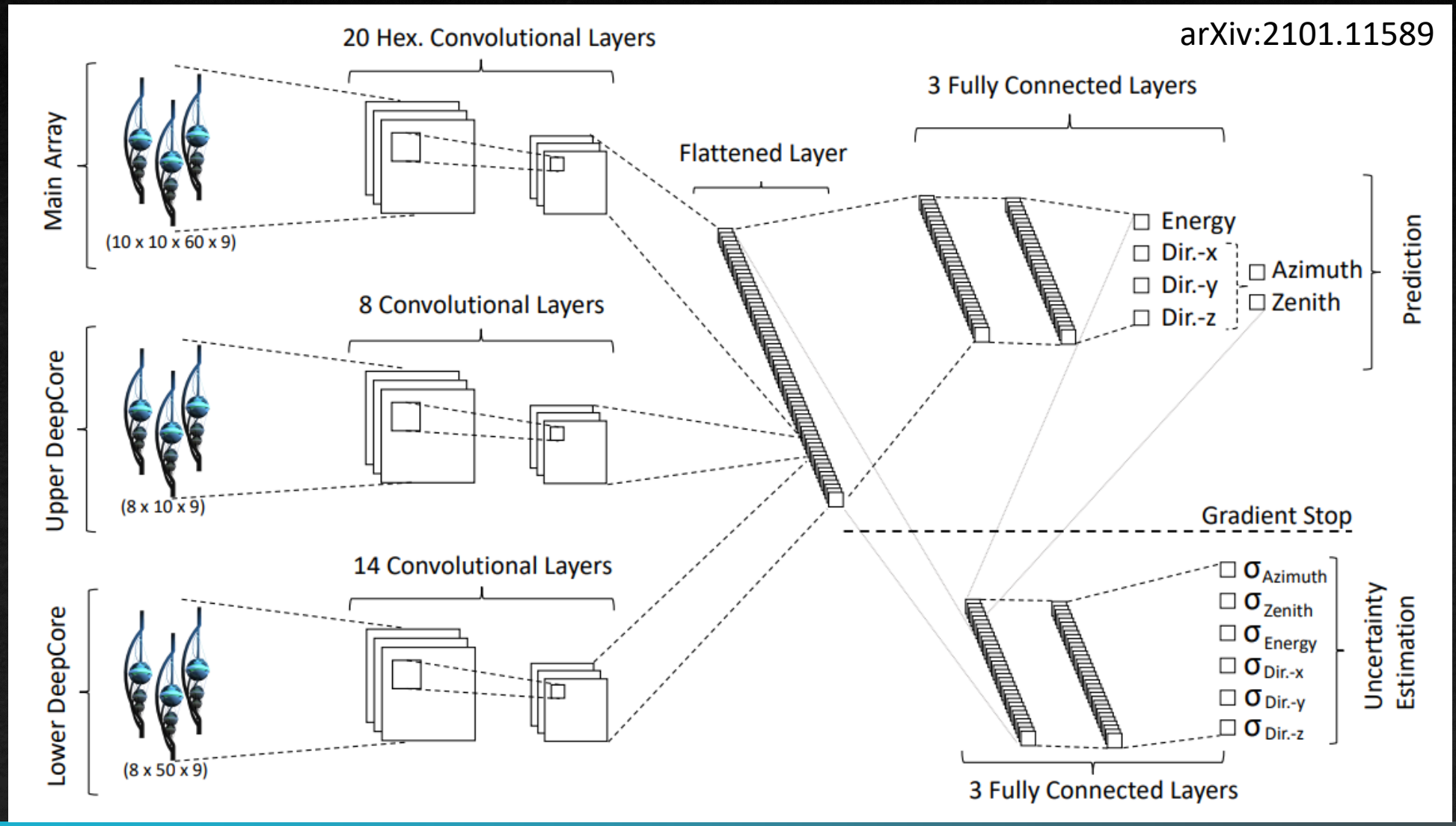
Example: IceCube event reconstruction with ML

- Inception layers learn different filters at once
- Residual connections stability the gradient in backpropagation

arXiv:1908.08763

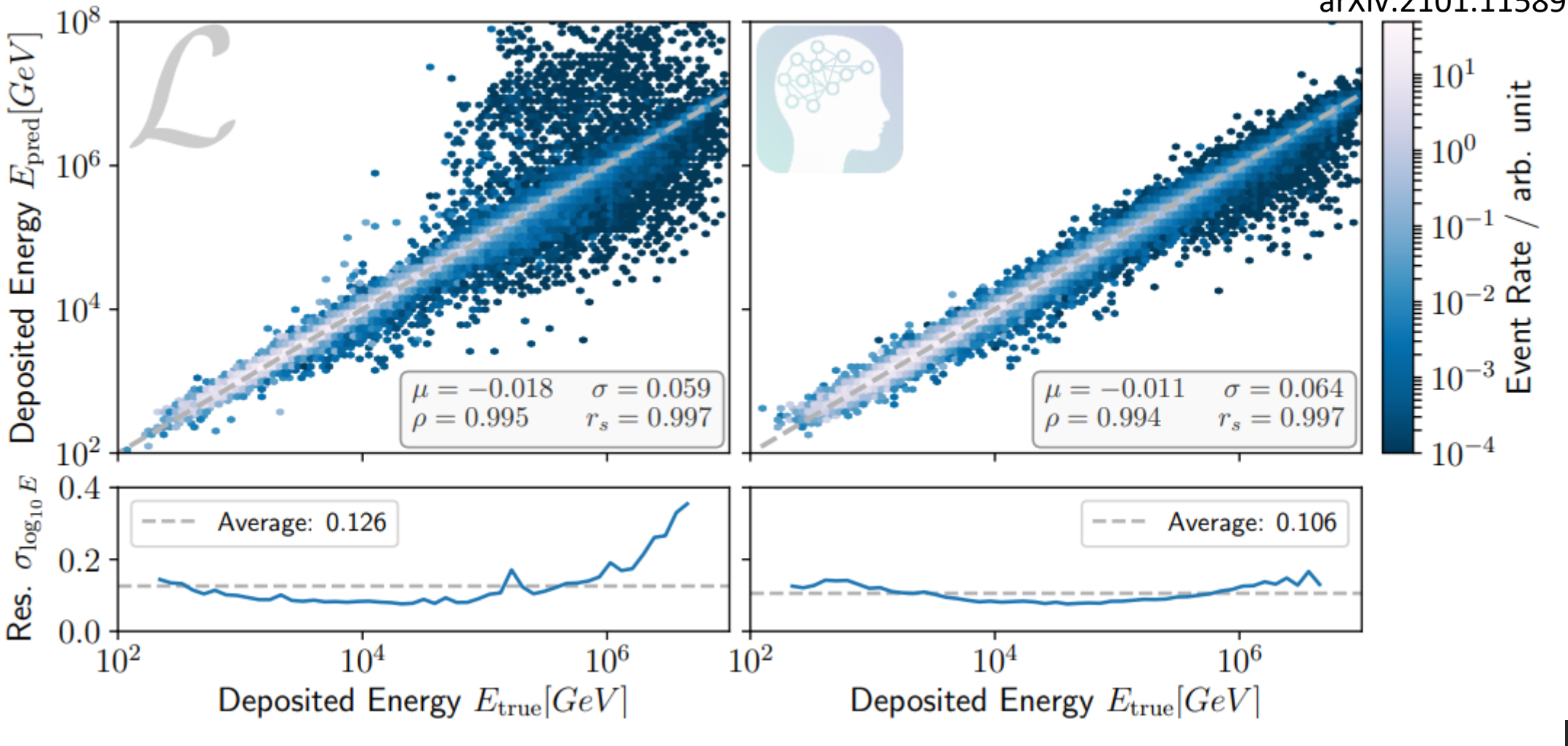


Example: IceCube event reconstruction with ML



Example: IceCube event reconstruction with ML

arXiv:2101.11589



Latest Developments in Physics and Deep Learning

- IME it takes 2-3 years for some in the physical sciences to adopt the latest ML methods
  - Risky and requires interdisciplinary knowledge
  - Overhelmed by the fast progress? Don't worry, some AI researchers are too

- NeurIPS is one of the top ML conference series
- Before looking through some posters / talks from the 2022 NeurIPS Physical Sciences workshop, its website outlined 3 types of ML
  - ML for physics: applications of ML in physics
  - Physics for ML: developments in ML motivated by physical insights
  - Physics with ML: Convergence of ML and physical sciences