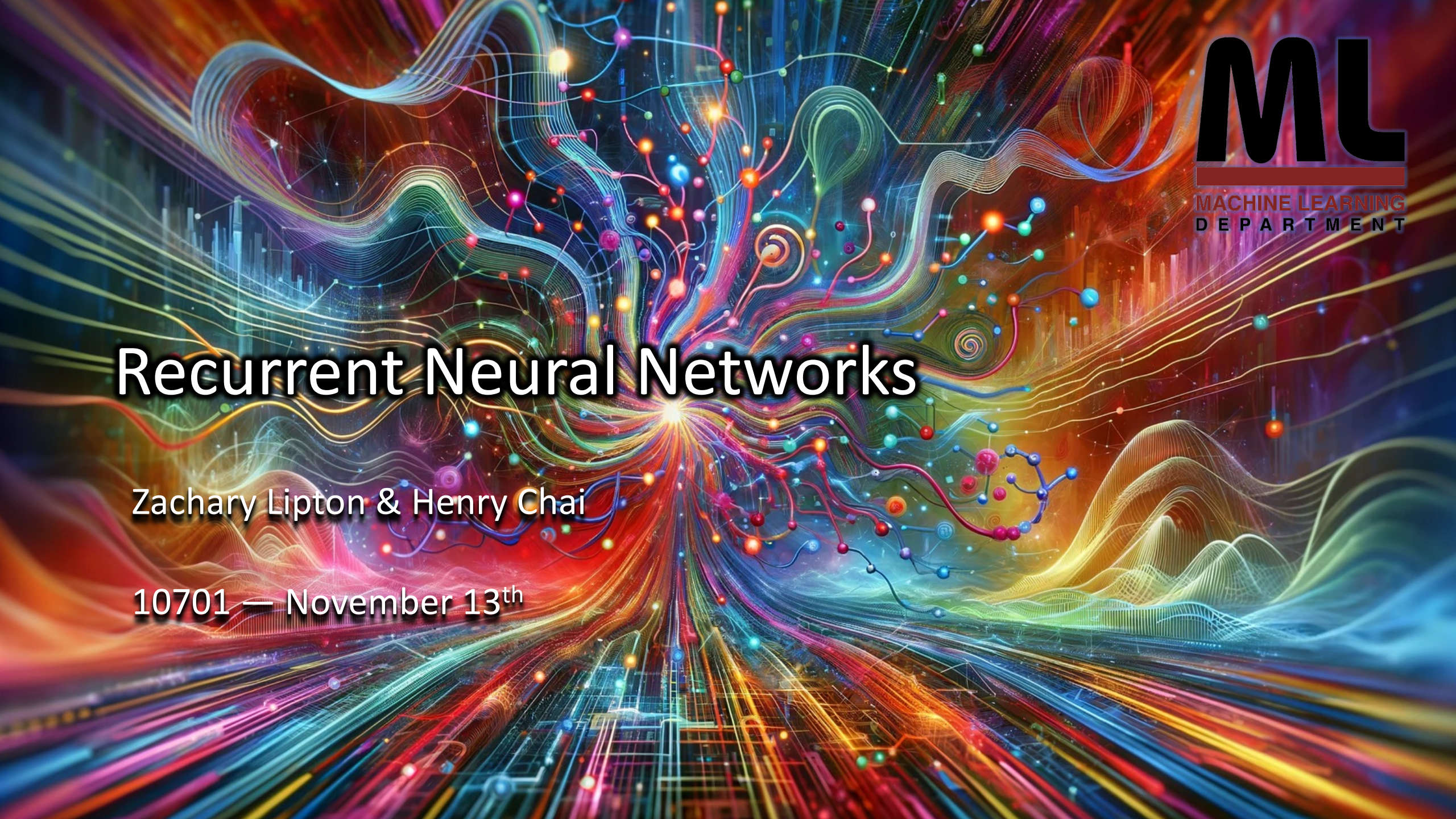
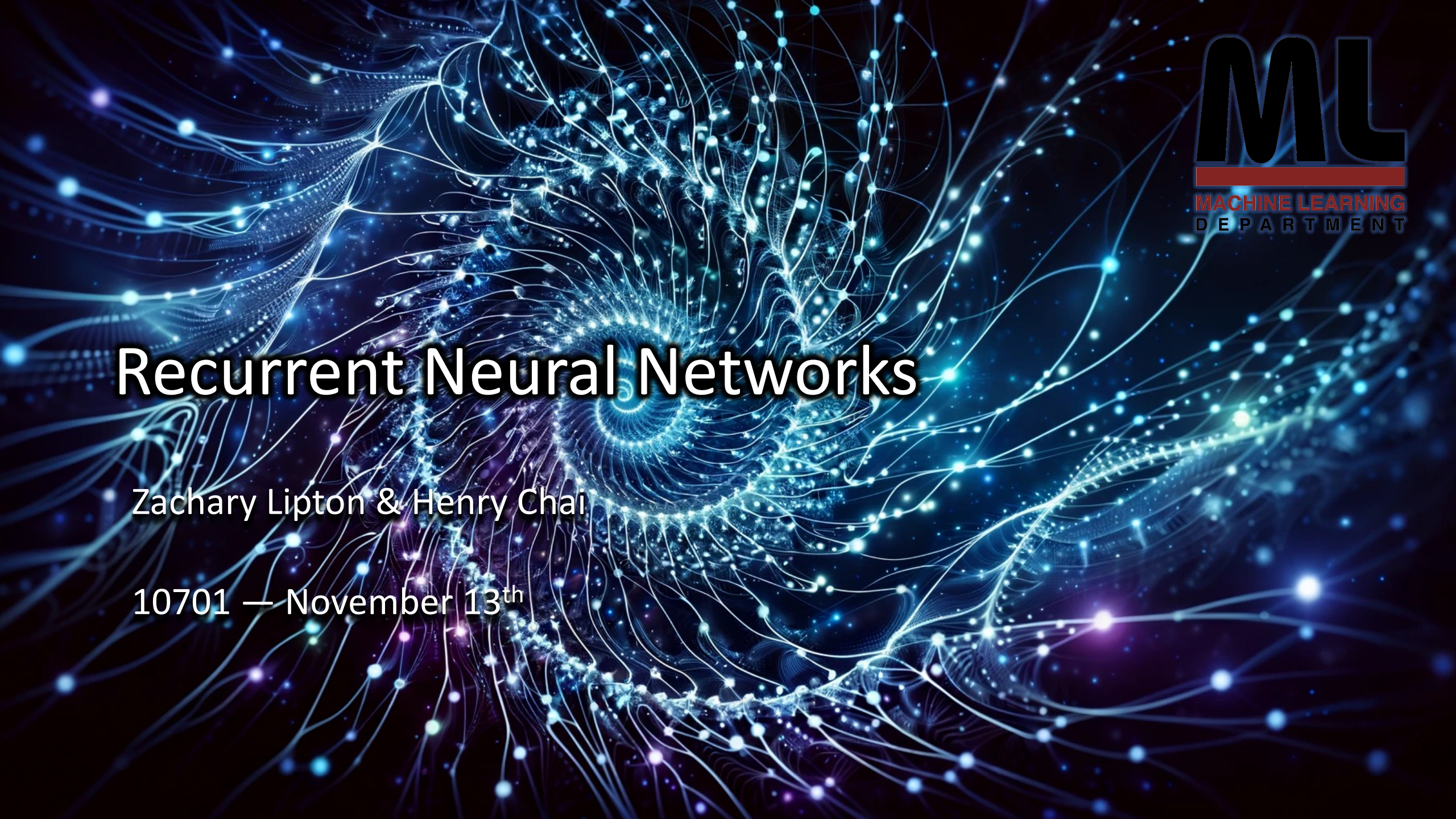


Recurrent Neural Networks

Zachary Lipton & Henry Chai

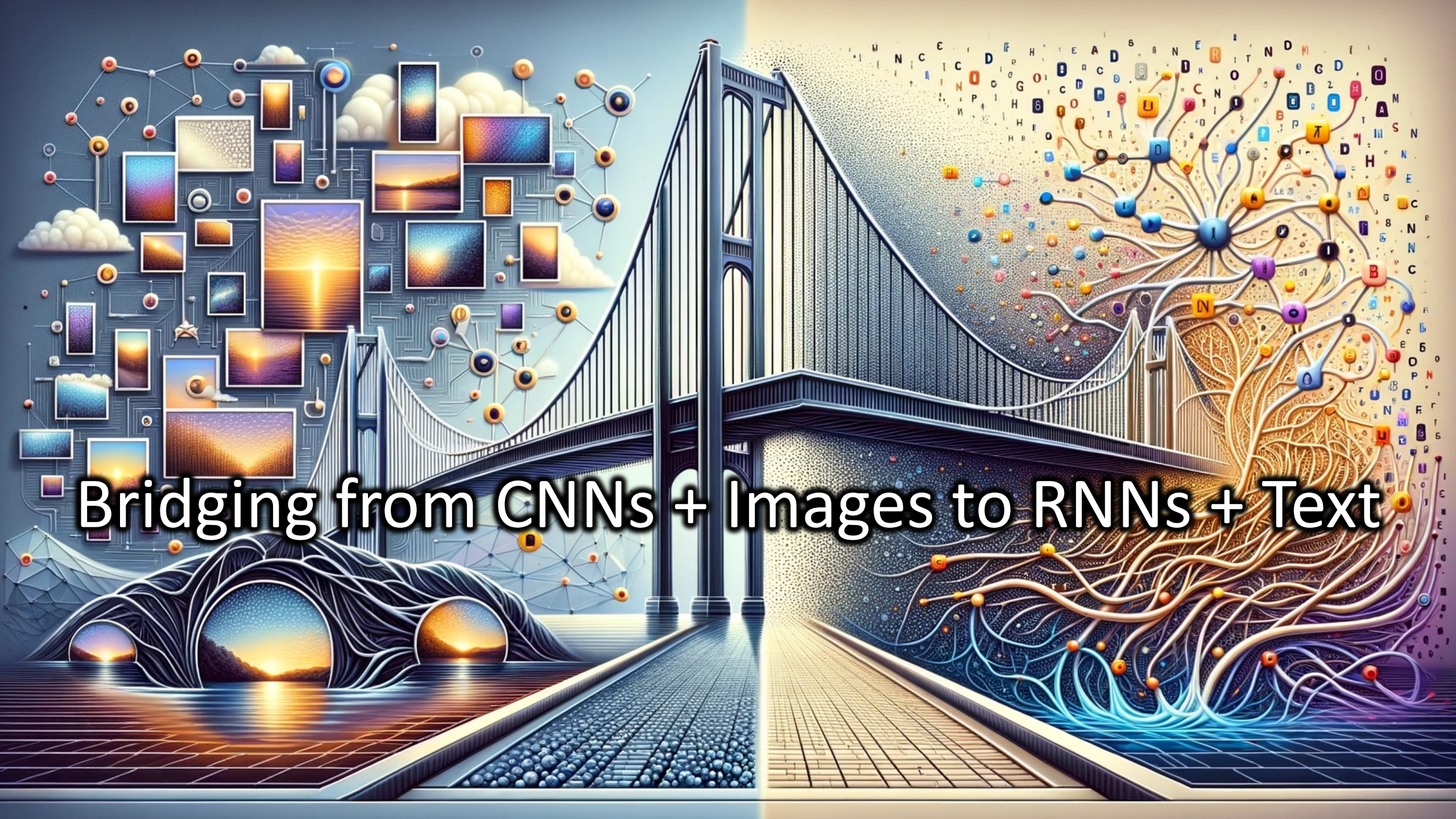
10701 — November 13th



Recurrent Neural Networks

Zachary Lipton & Henry Chai

10701 — November 13th


Bridging from CNNs + Images to RNNs + Text

Recap: Multi-Modal Pretraining

What exactly goes on inside this “text encoder”?

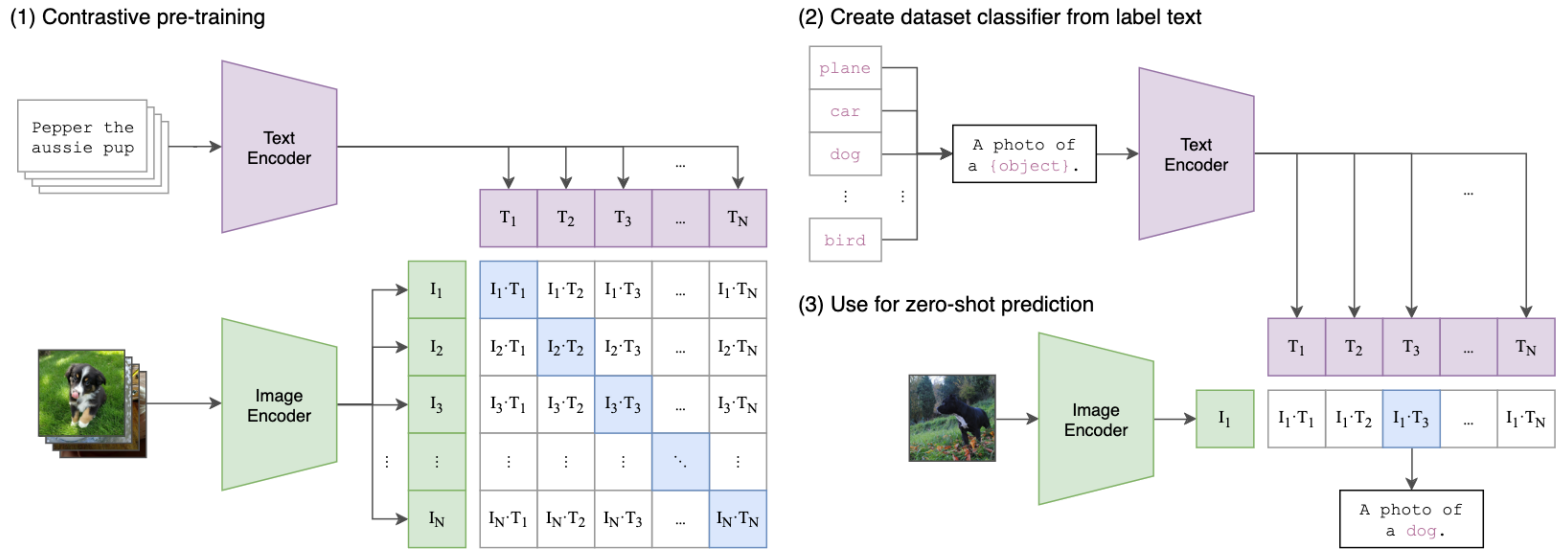
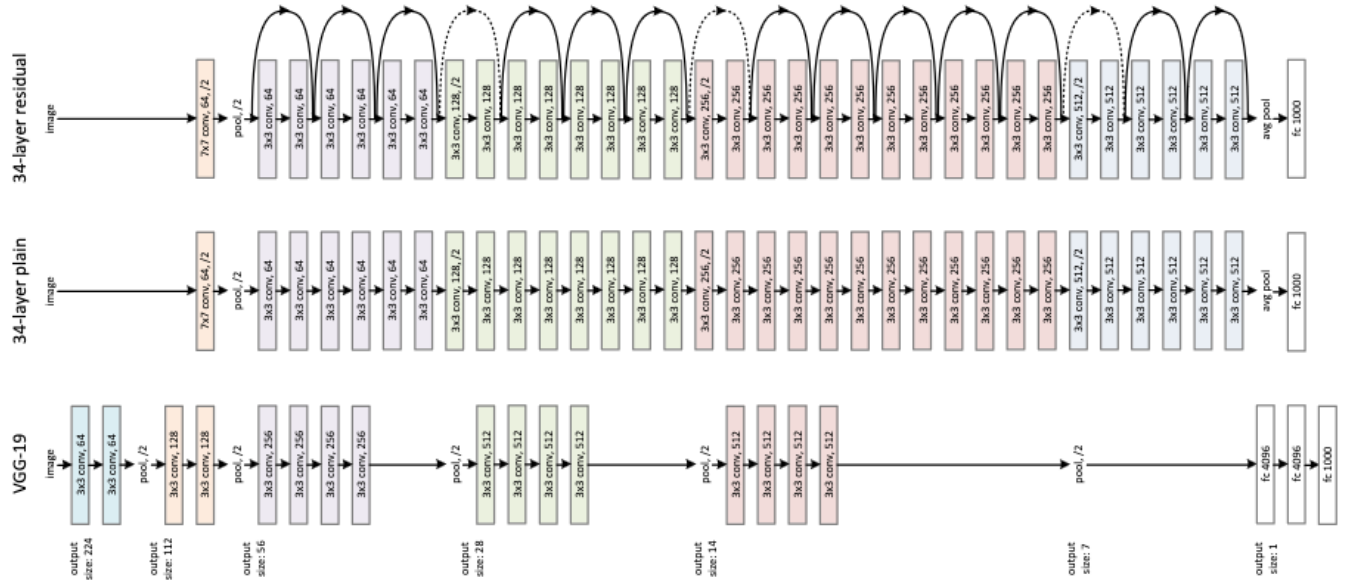
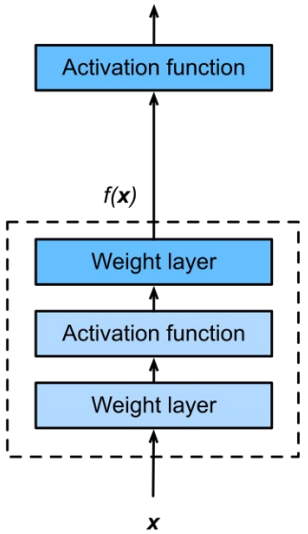
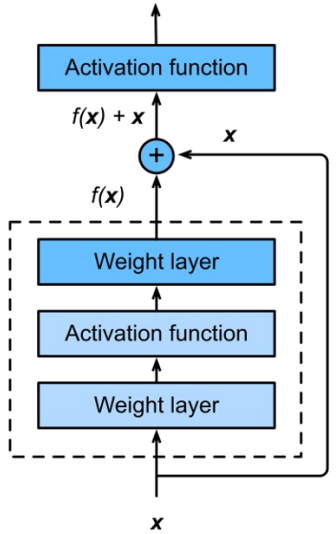


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset’s classes.

ResNet as Gradient “Superhighway”

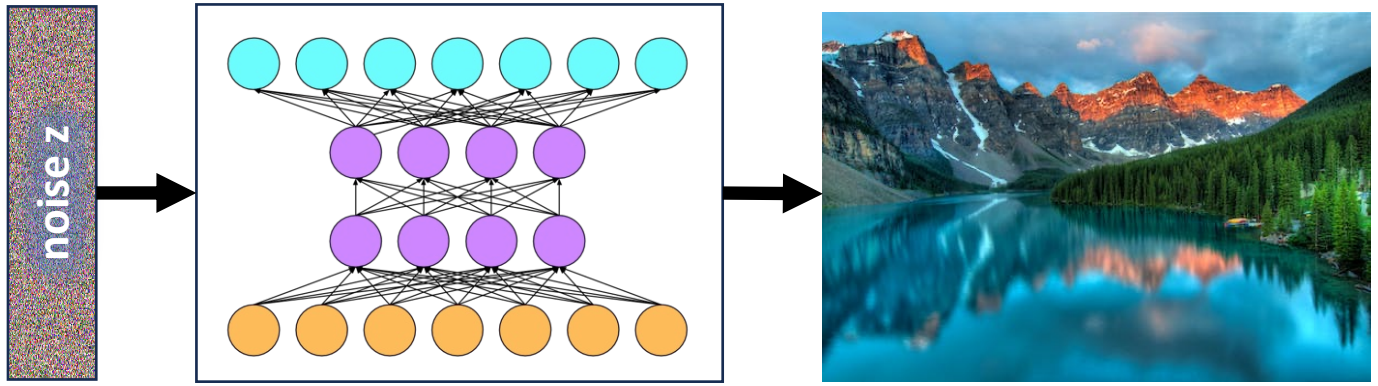






"Deep Residual Learning for Image Recognition" He et al. 2015

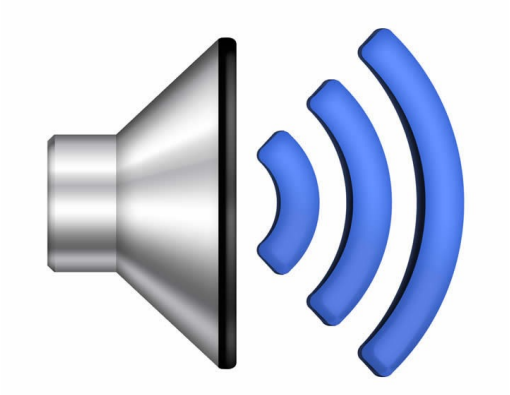
Question: How to Learn Distributions on Discrete Domains?

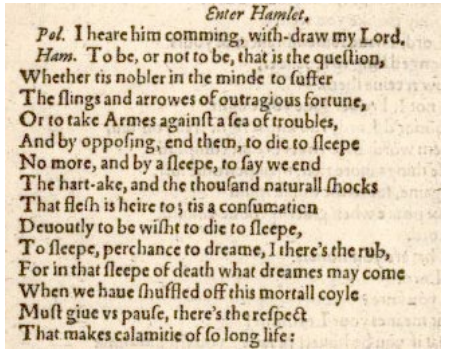


$p($ "Born and raised in North Carolina, Coltrane moved to Philadelphia after graduating from high school, where he studied music. Working in the bebop and hard bop idioms early in his career, Coltrane helped pioneer the use of modes and was one of the players at the forefront of free jazz. He led at least fifty recording sessions and appeared on many albums by other musicians, including trumpeter Miles Davis and pianist Thelonious Monk. Over the course of his career, Coltrane's music took on an increasingly spiritual dimension, as exemplified on his most acclaimed album A Love Supreme (1965) and others." $)$

Sequentially Structured Data

- Have been focusing on data with fixed-length inputs
- But how should we deal with video, text, audio, time series data?

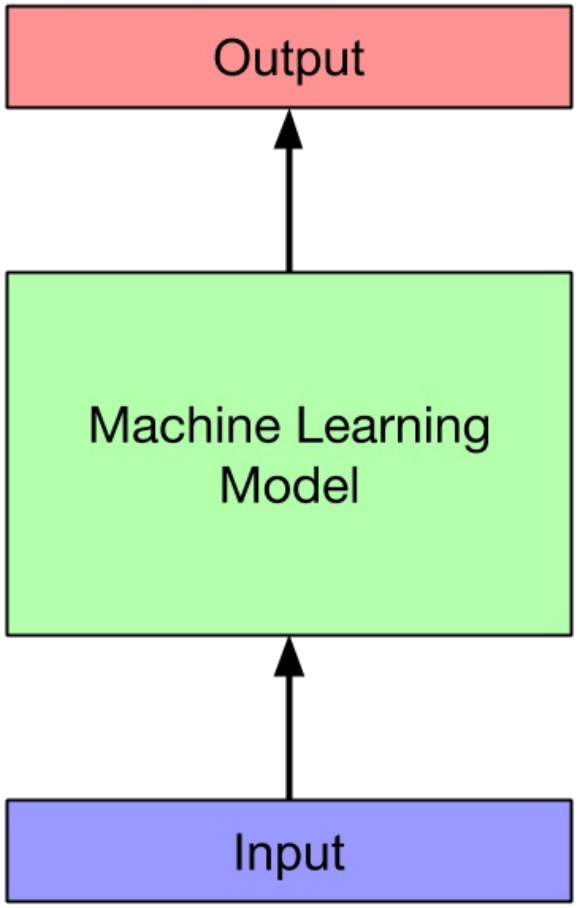


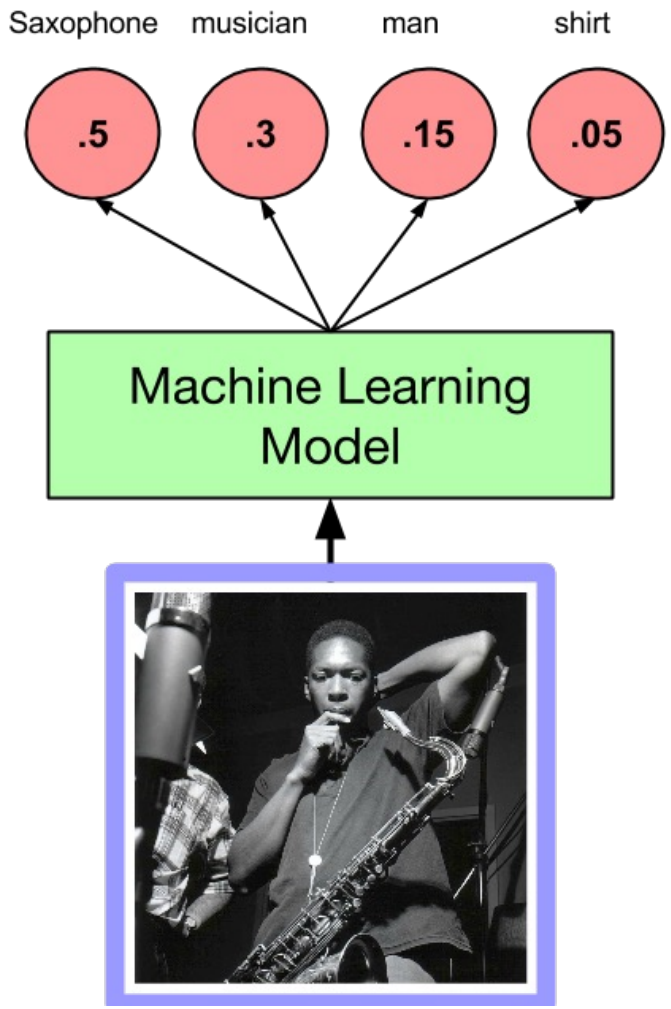


Enter Hamlet,

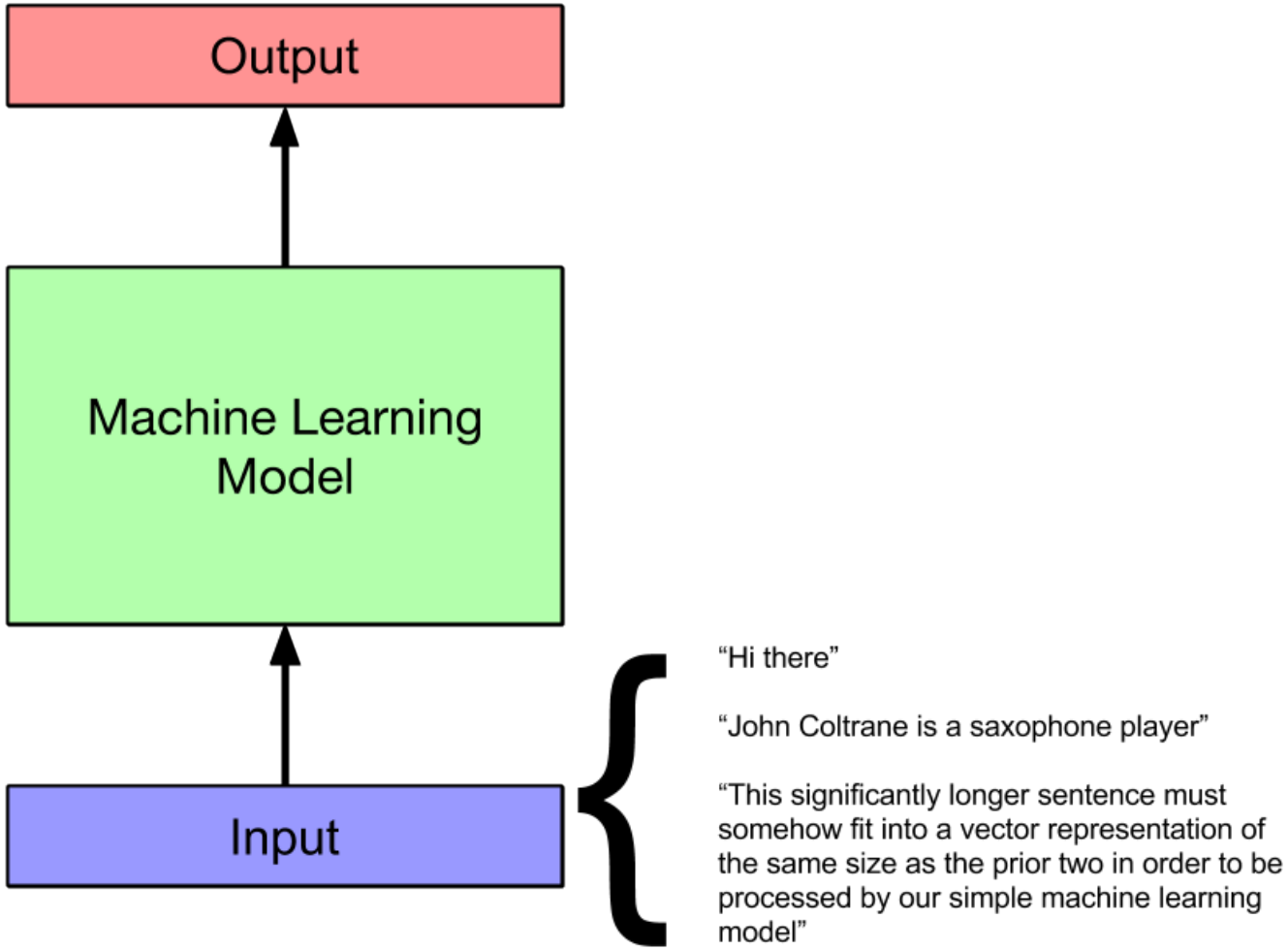
Pol. I heare him comming, with-draw my Lord.
Ham. To be, or not to be, that is the queſtion,
Whether tis nobler in the minde to ſuffer
The ſlings and arrowes of outragious fortune,
Or to take Armes againſt a ſea of troubles,
And by oppoſing, end them, to die to ſleepe
No more, and by a ſleepe, to ſay we end
The hart-ake, and the thouſand naturall ſhocks
That fleſh is heire to; tis a conſumation
Deuoutly to be wiſht to die to ſleepe,
To ſleepe, perchance to dreame, I there's the rub,
For in that ſleepe of death what dreames may come
When we haue ſhuffled off this mortall coyle
Muſt giue vs pauſe, there's the reſpect
That makes calamitie of ſo long life:

Feedforward Nets work for Fixed-Size Data





How to Handle Raw Text as Input (or output)?



Traditional Approach for Text Classification

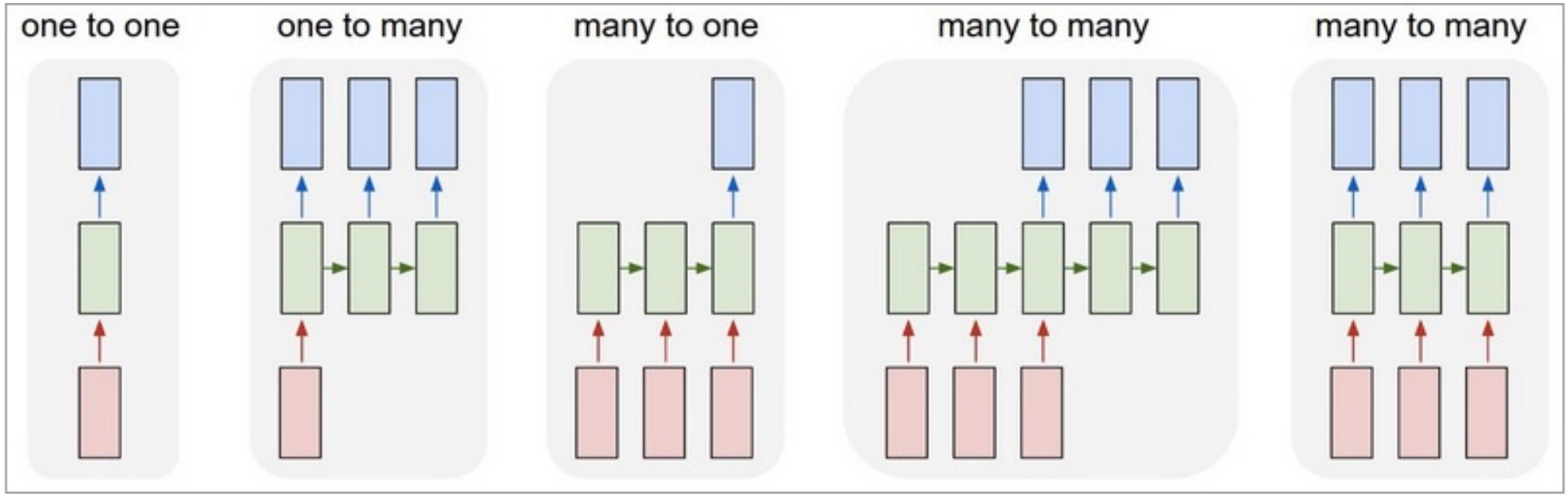
Bag-of-words representations

- Represent each document as vector of word counts $x \in \mathbb{R}^{|V|}$
- Dimension $|V|$ = vocab size
- Loses all information about word order.
- Reasonably effective for spam detection but cannot differentiate:
  i. "Scientist exterminated by raging virus!"
  ii. "Virus exterminated by raging scientist!"
- Limited ability to tackle complex language tasks (e.g., translation)
- No clear output mechanism

Complex Object → Sequence of Simple Ones

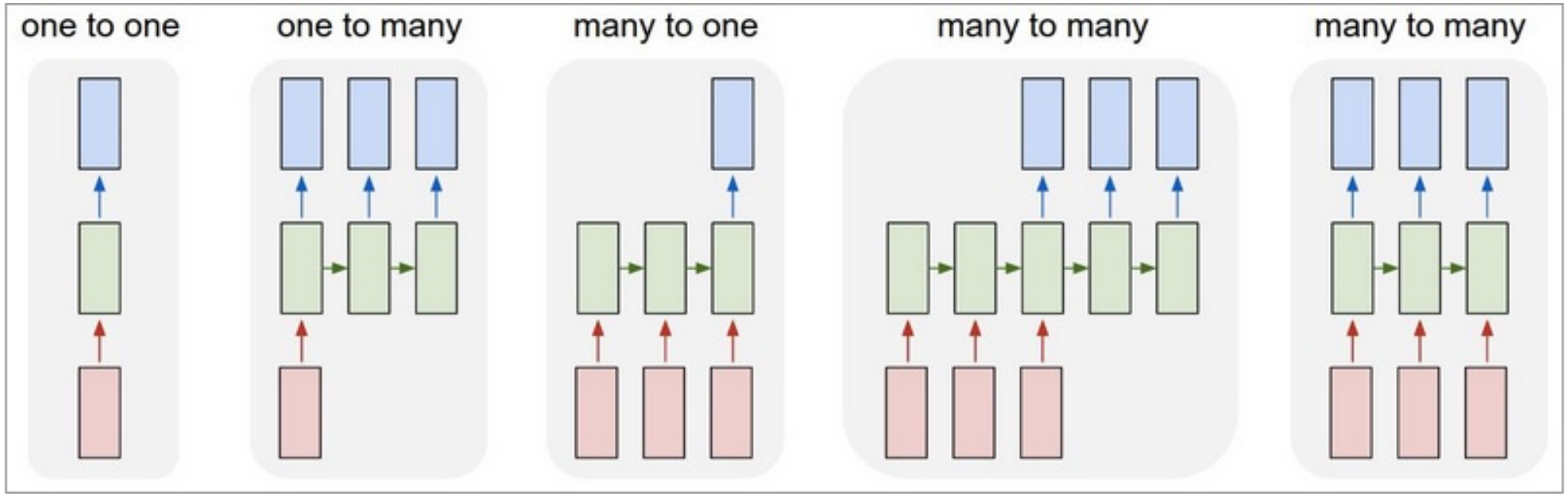
- Hard to represent sentence as fixed-length vector
- Easier to represent characters (or words) as fixed-length vectors
- Breaking up the inputs: ingest a word at each step
- Break up the outputs: predict next token at each step

Archetypal Sequence Learning Tasks



slide credit: Andrej Karpathy

Archetypal Sequence Learning Tasks



Simple prediction tasks: fixed-length input & output

Image captioning

Sentiment analysis

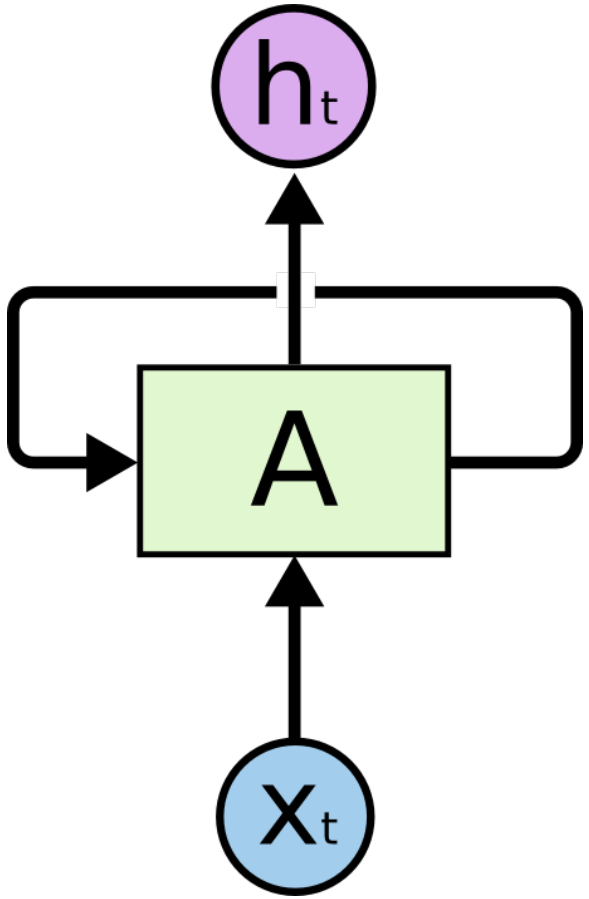
Video captioning, Natural language translation

Text tagging tasks:
Part-of-speech detection,
Named entity recognition,
Semantic Role Labeling

Language Modeling

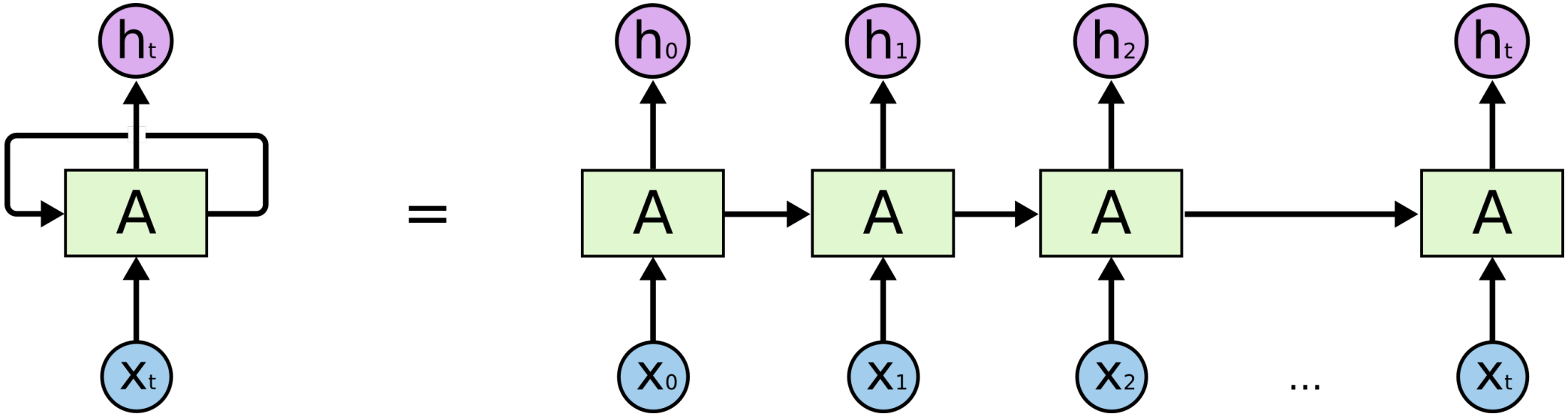
slide credit: Andrej Karpathy

Architecture Strategy: Recurrent Layers



Img credit: Chris Olah blog post (Understanding LSTMs)

Two Views (Self-connection vs Unrolled)

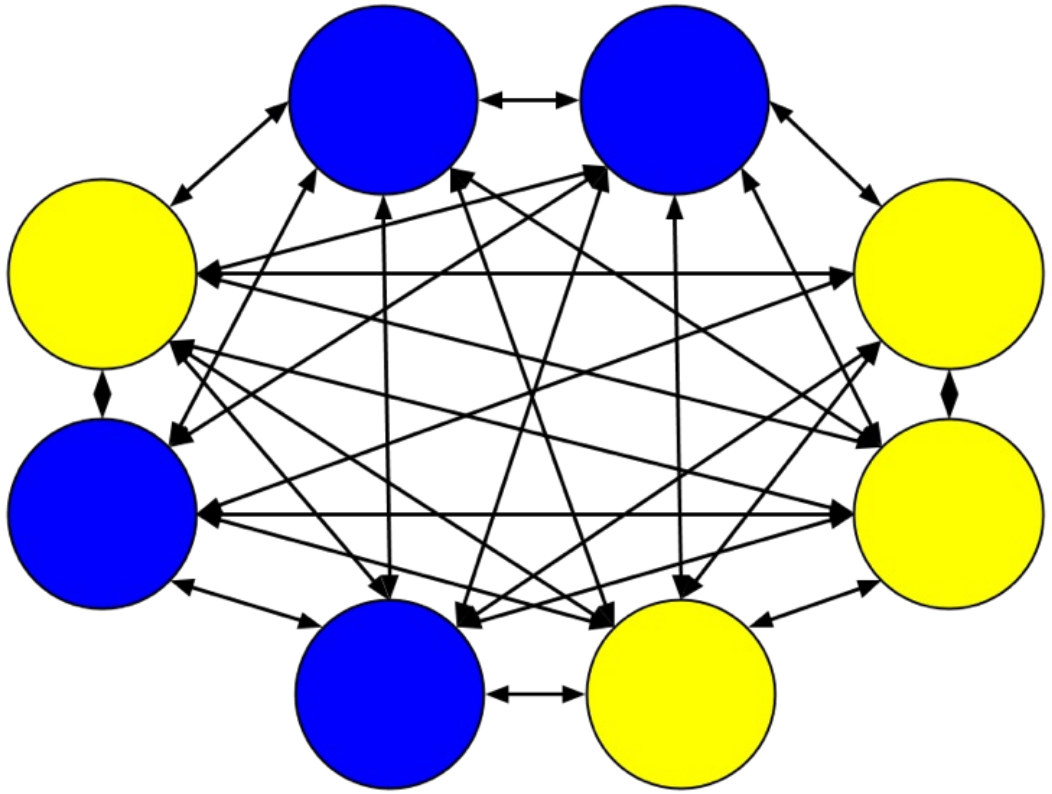


Img credit: Chris Olah blog post (Understanding LSTMs)

A Brief Tour of Classical RNNs

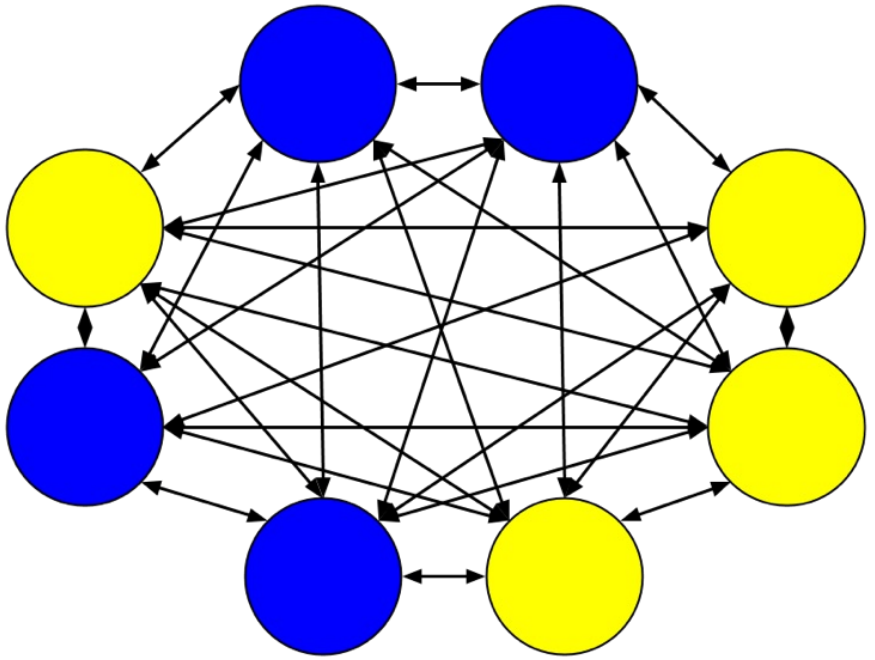
Hopfield Nets (Hopfield, 1982)

- Parametrization
  - Each node pair *i,j* connected by edge with weight w_{ij}
  - No self connections ($w_{ii} = 0$)
- Updates via Hebbian rule
  - Fire together → wire together

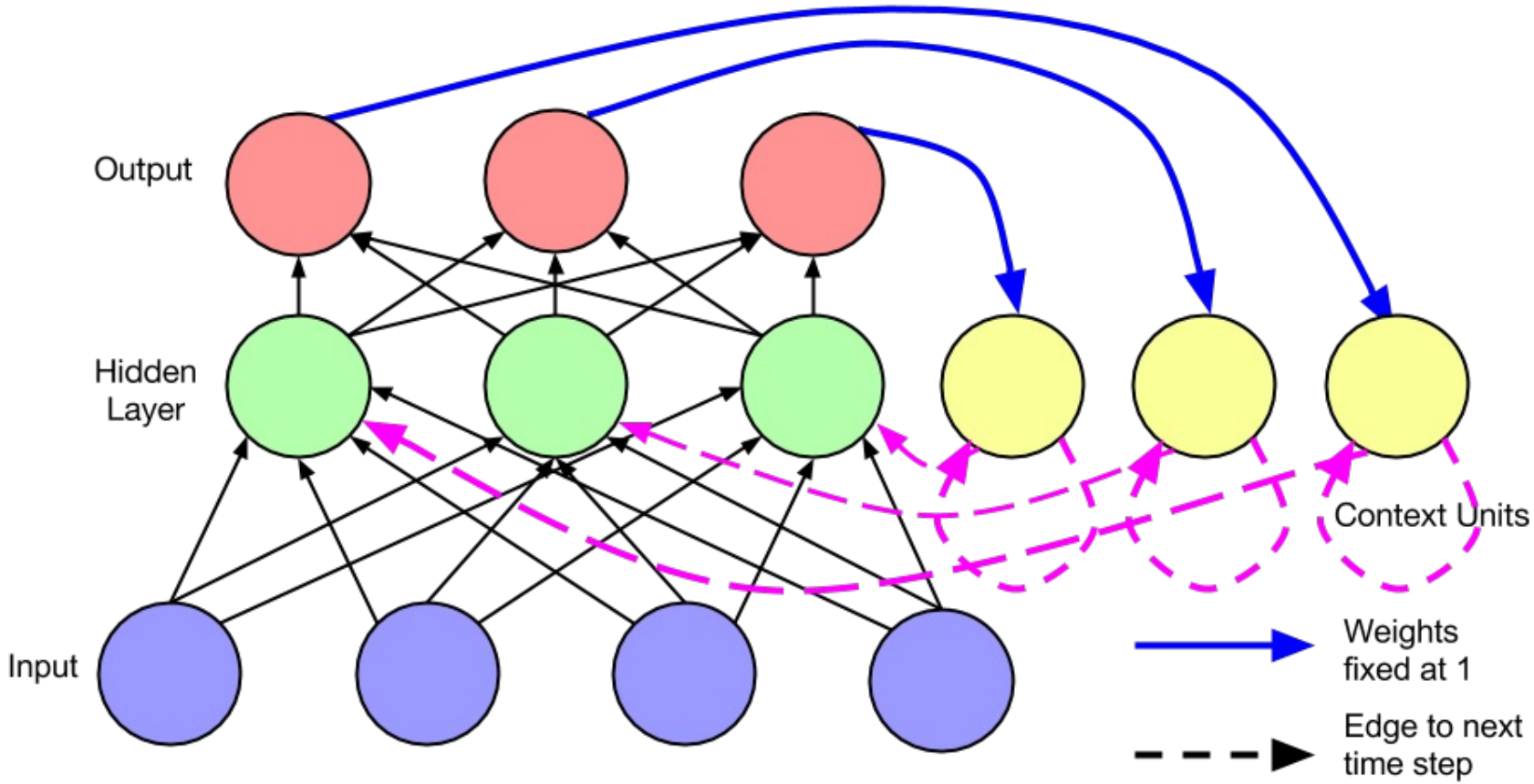


Hopfield nets (cont'd)

- Updates: $$s_i \leftarrow \begin{cases} +1, \text{for } \sum_j w_{ik} > \theta_i \\ -1, \text{otherwise} \end{cases}$$
- Two ways to "run" the net:
  - Asynchronously — update one at a time
  - Synchronously — simultaneously update all
- Learning: $$w_{ij} = 1/n \sum_{u=1}^{n} \epsilon_i^u \epsilon_j^u$$
- Idea: Minimize the "energy"



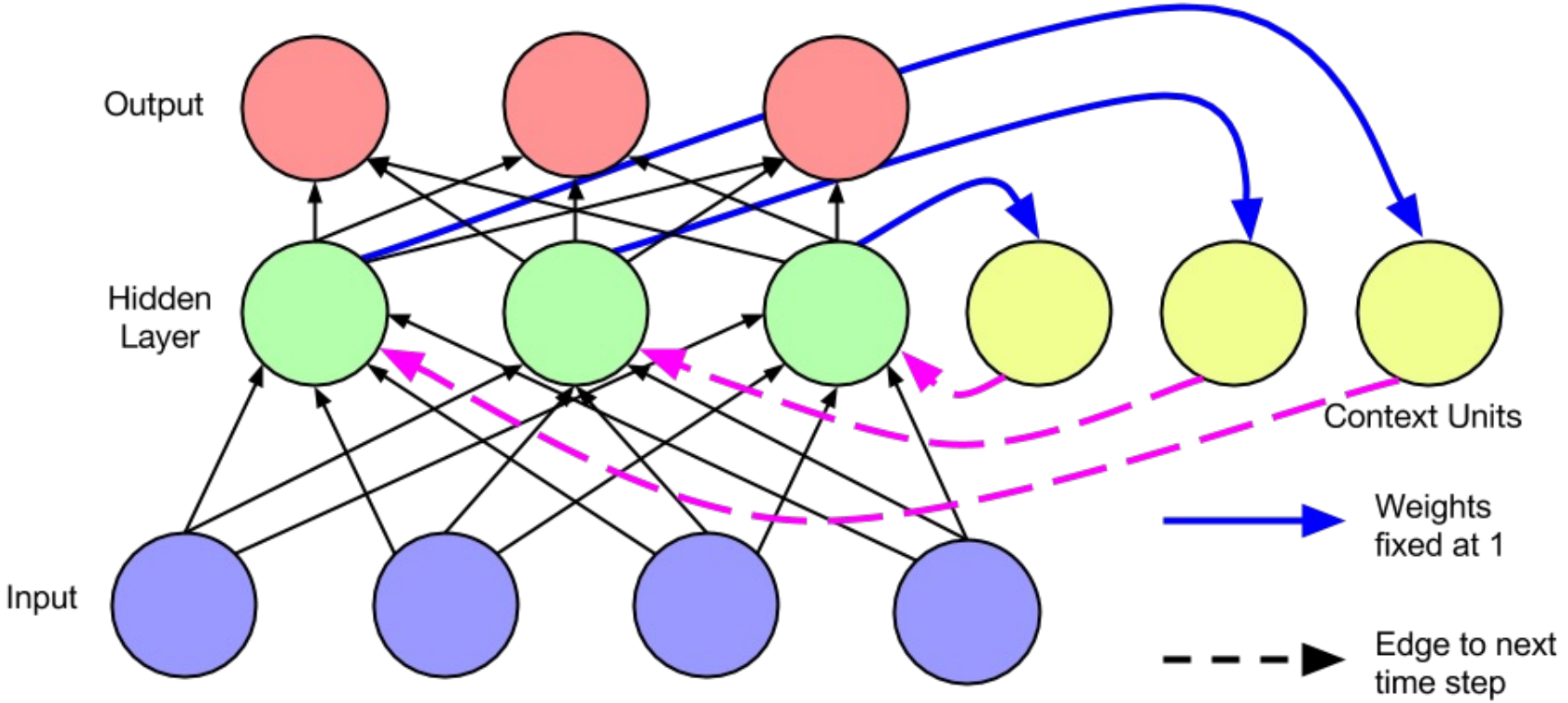
Jordan Nets (1986)



Jordan Nets (Michael I. Jordan, 1986)

- Pass output from previous step into hidden nodes at current step
- Used for planning
- Didn’t use backpropagation owing to biological implausibility

Elman Nets (1990)



Elman's Experiments (*Finding Structure in Time*, 1990)

- Trained net to perform temporal version of XOR
  - Input: 0, 0, 0, 1, 0, 1, 0, 1, 1
  - Every third character is the XOR of the previous two
  - Goal: predict next bit (first two are random)
  - System correctly guessed each third character with 85% accuracy

- Trained net to predict next character in text
  - Accurately predicted next char
  - 21 years before recent resurgence in auto-regressive text modeling

Sequence Model

- Dependent random variables

$$(x_1, \dots x_T) \sim p(x)$$

- Conditional probability expansion

$$p(x) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1, x_2) \cdot \dots p(x_T|x_1, \dots x_{T-1})$$

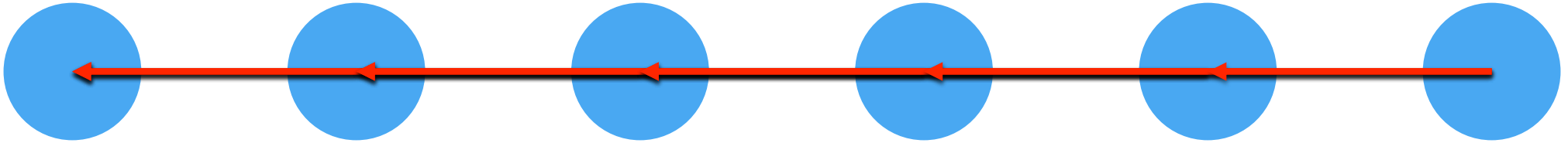
- **So why bother?**

$$p(x) = p(x_T) \cdot p(x_{T-1}|x_T) \cdot p(x_{T-2}|x_{T-1}, x_T) \cdot \dots p(x_1|x_2, \dots x_T)$$

Sequence Model

$$p(x) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1, x_2) \cdot \dots p(x_T|x_1, \dots x_{T-1})$$

$$p(x) = p(x_T) \cdot p(x_{T-1}|x_T) \cdot p(x_{T-2}|x_{T-1}, x_T) \cdot \dots p(x_1|x_2, \dots x_T)$$



- Causality (physics) prevents the reverse direction
- ‘wrong’ direction often much more complex to model
- Typically more useful to predict text in left-to-right fashion

Sequence Model

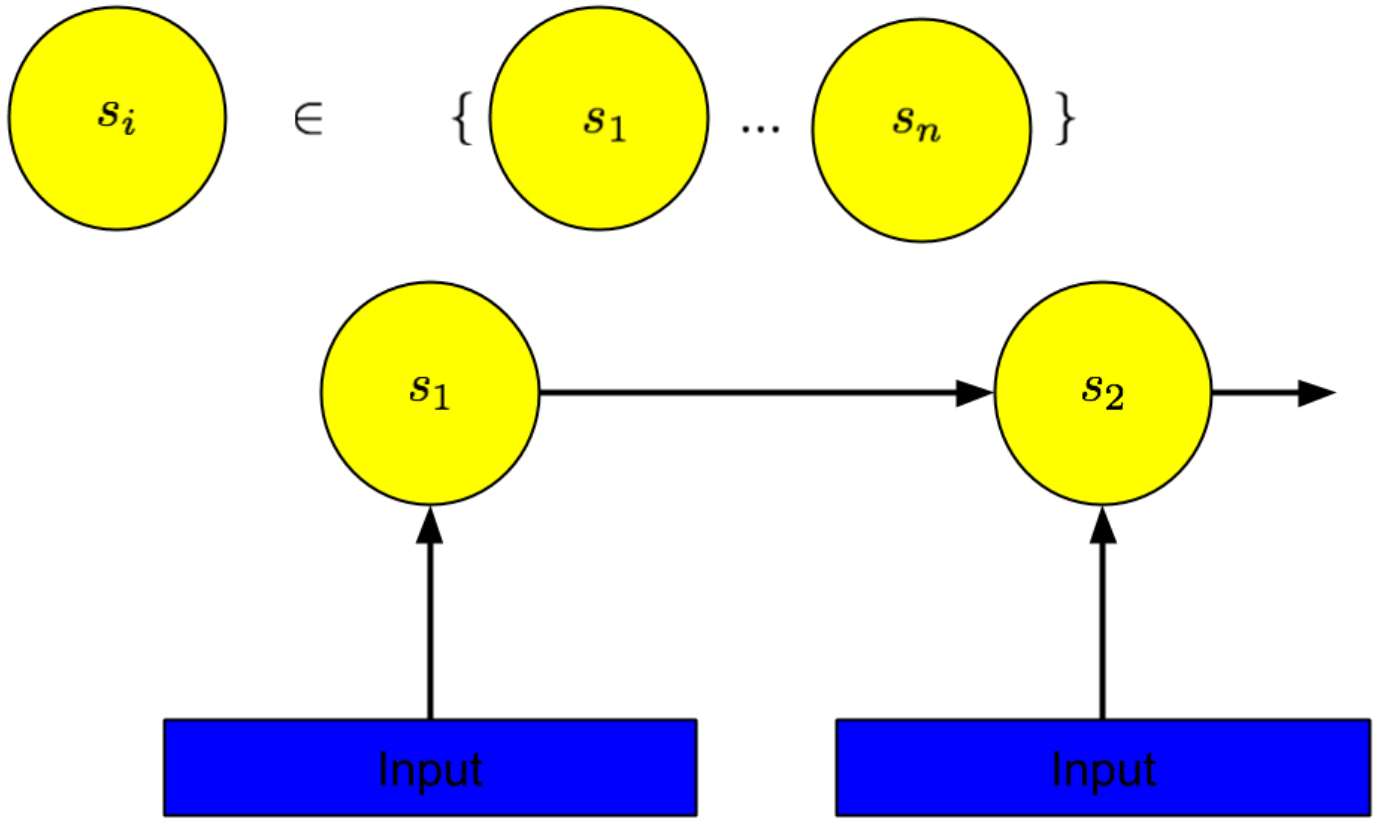
$$p(x) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1, x_2) \cdot \dots p(x_T|x_1, \dots x_{T-1})$$

- Autoregressive model

$$p(x_t|x_1, \dots x_{t-1}) = p(x_t|f(x_1, \dots x_{t-1}))$$

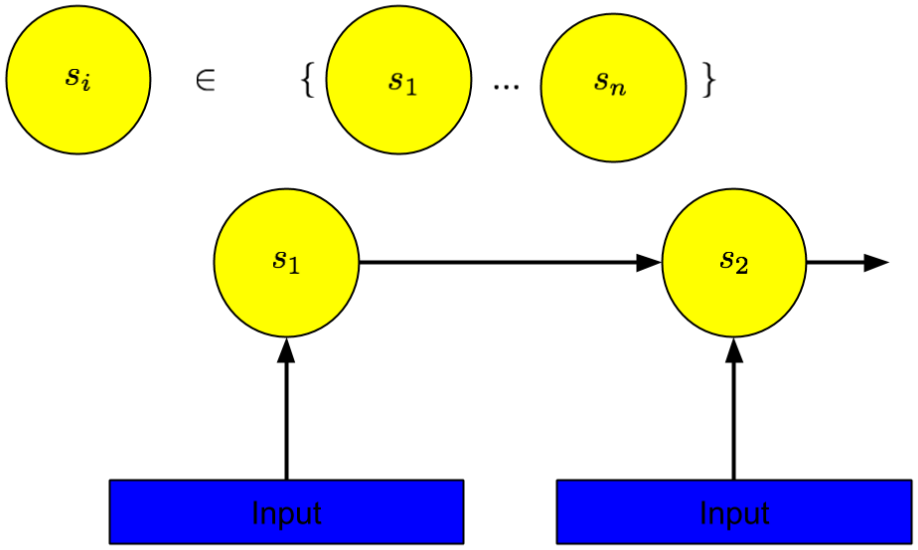
Some function of previously seen data

Why Not (Hidden) Markov Models?



Why Not (Hidden) Markov Models?

- For Markov models, inference is quadratic in number of states
- RNNs representational expressivity grows exponentially in latent dimension, computation scales linearly



Classic n-gram Language Models

- Make (incorrect) Markov assumption

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

- Estimate probabilities by counting exact occurrences of n-grams

$$p(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)} \qquad p(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$$

- Smoothing: give some mass to unseen n-grams so loss doesn't → ∞

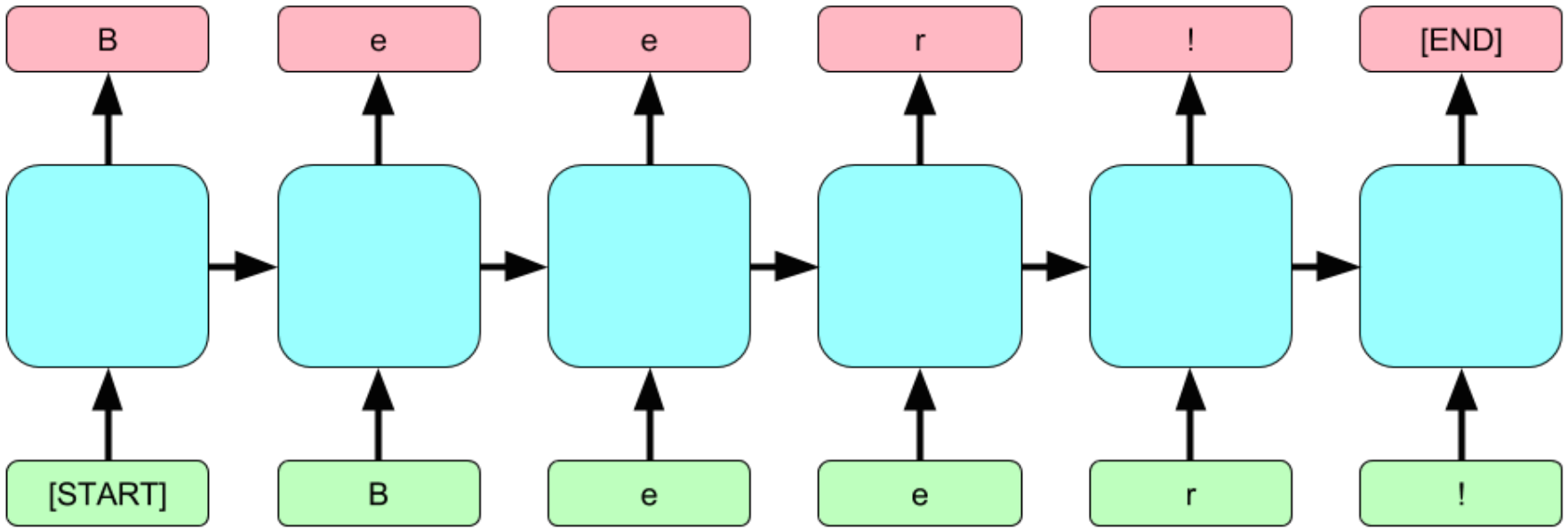
The equation for bigram probabilities is as follows:

$$p_{KN}(w_i|w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - \delta, 0)}{\sum_{w'} c(w_{i-1}, w')} + \lambda_{w_{i-1}} p_{KN}(w_i)$$ [4]

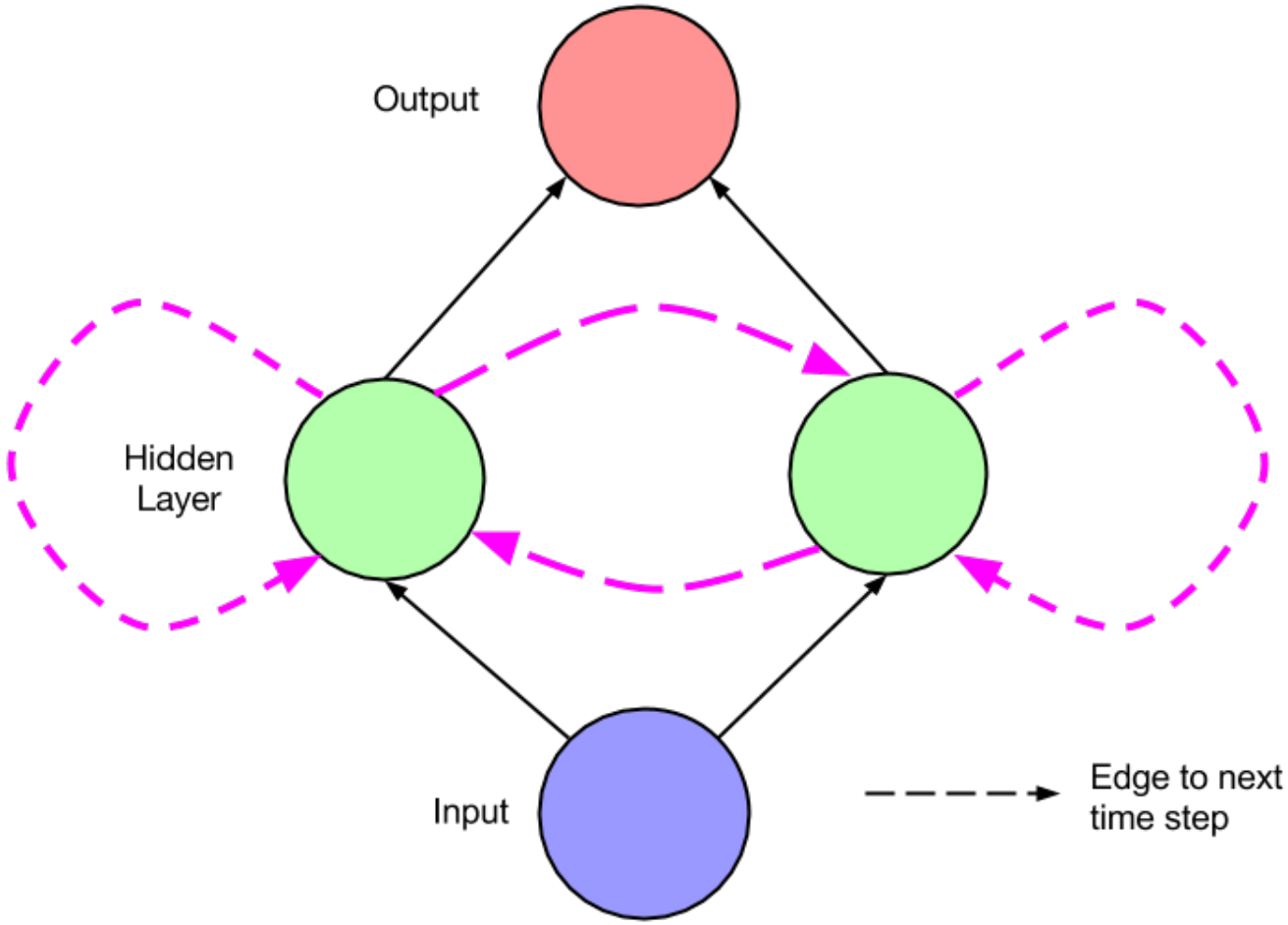
Where the unigram probability $p_{KN}(w_i)$ depends on how likely it is to see the word w_i in an unfamiliar context, which is estimated as the number of times it appears after any other word divided by the number of distinct pairs of consecutive words in the corpus:

Kneser-Ney wiki page

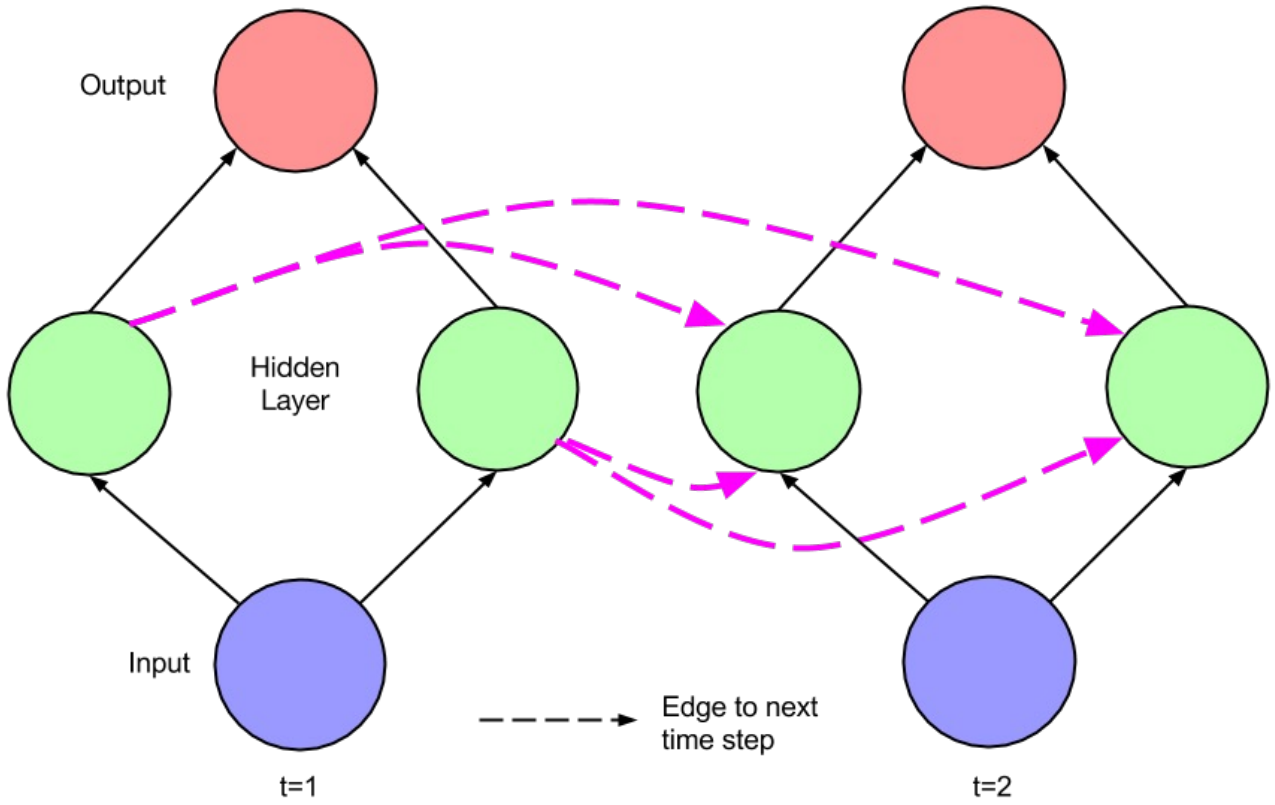
RNN-based Language Modeling



Recurrent Neural Networks



Recurrent Neural Network (Unfolded)



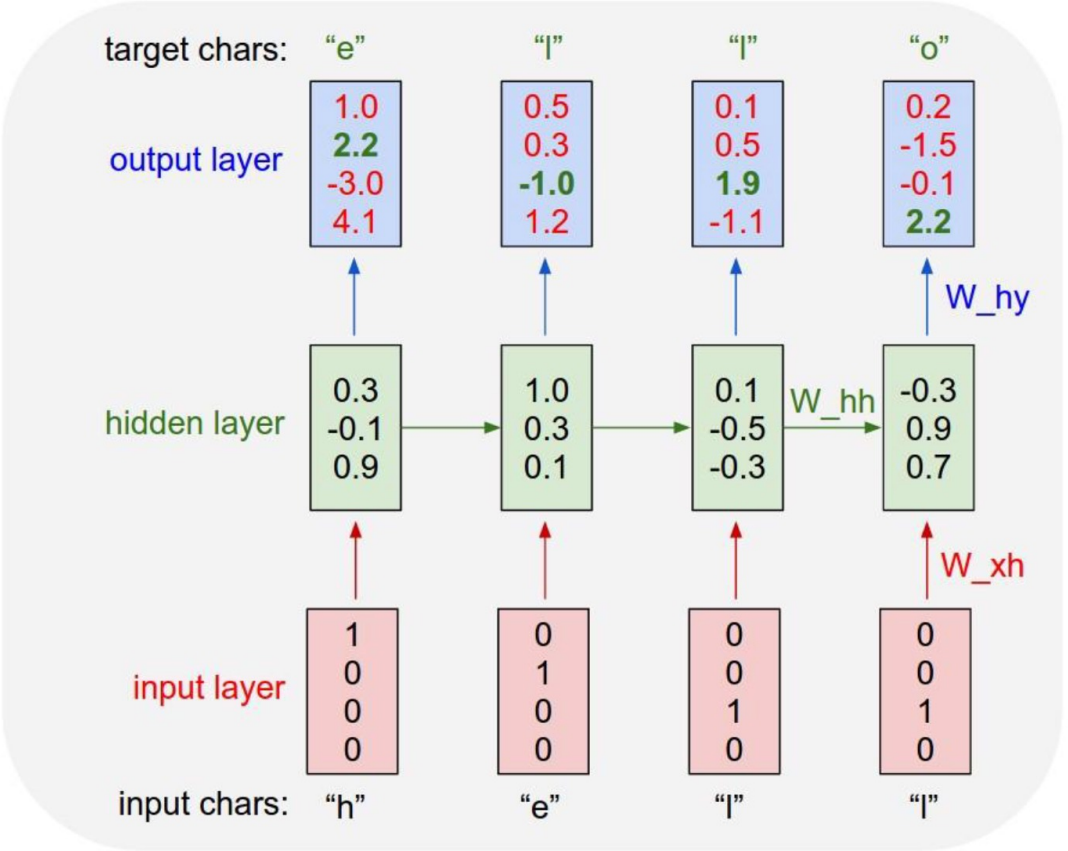
$$h^{(t)} = \sigma(W_{hx}\boldsymbol{x}^{(t)} + W_{hh}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h)$$

$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}(W_{yh}\boldsymbol{h}^{(t)} + \boldsymbol{b}_y)$$

RNN-based LMs at Train Time

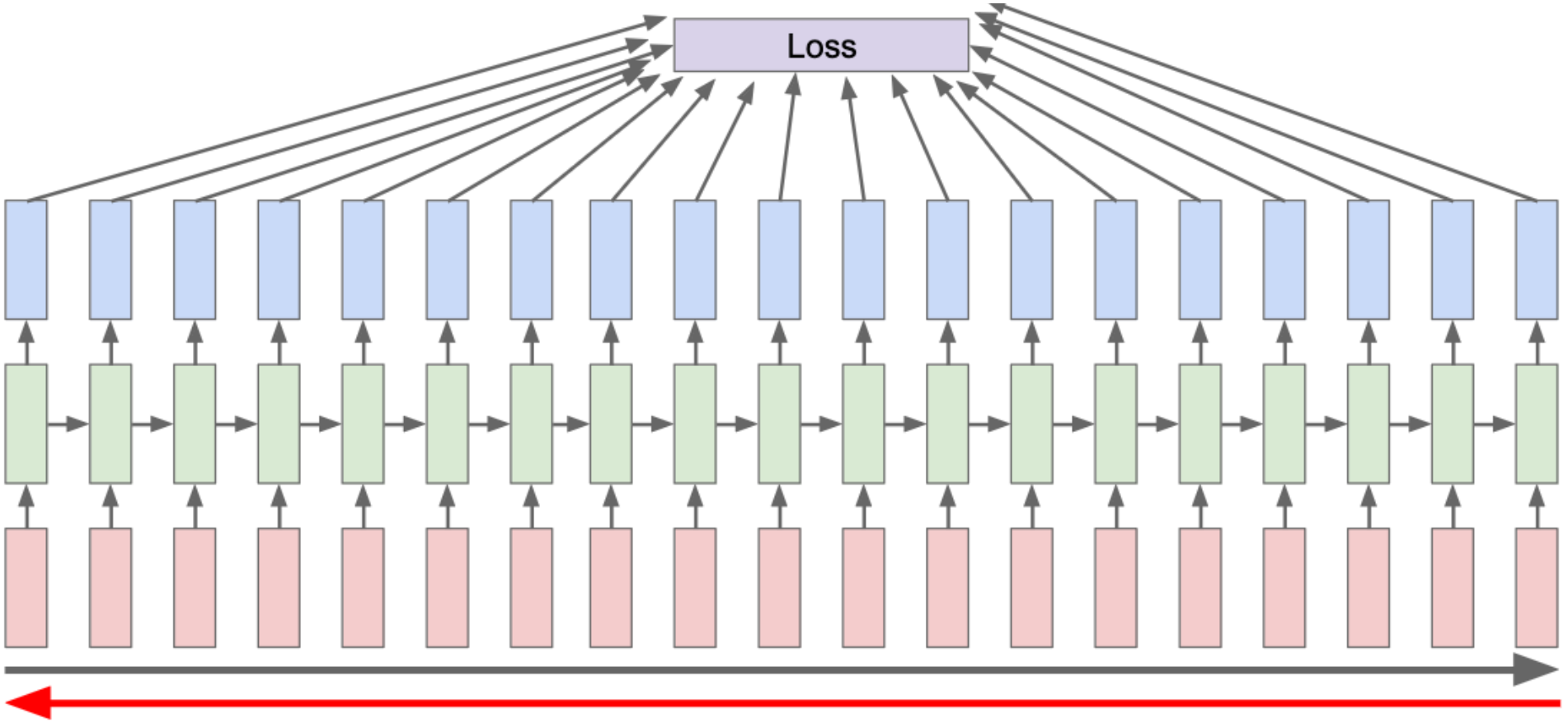
At Each sequence step t:

- Input is current token x_t
- Target is subsequent token x_{t+1}
- Sample sequences up to some maximum length
- Backprop across compute graph



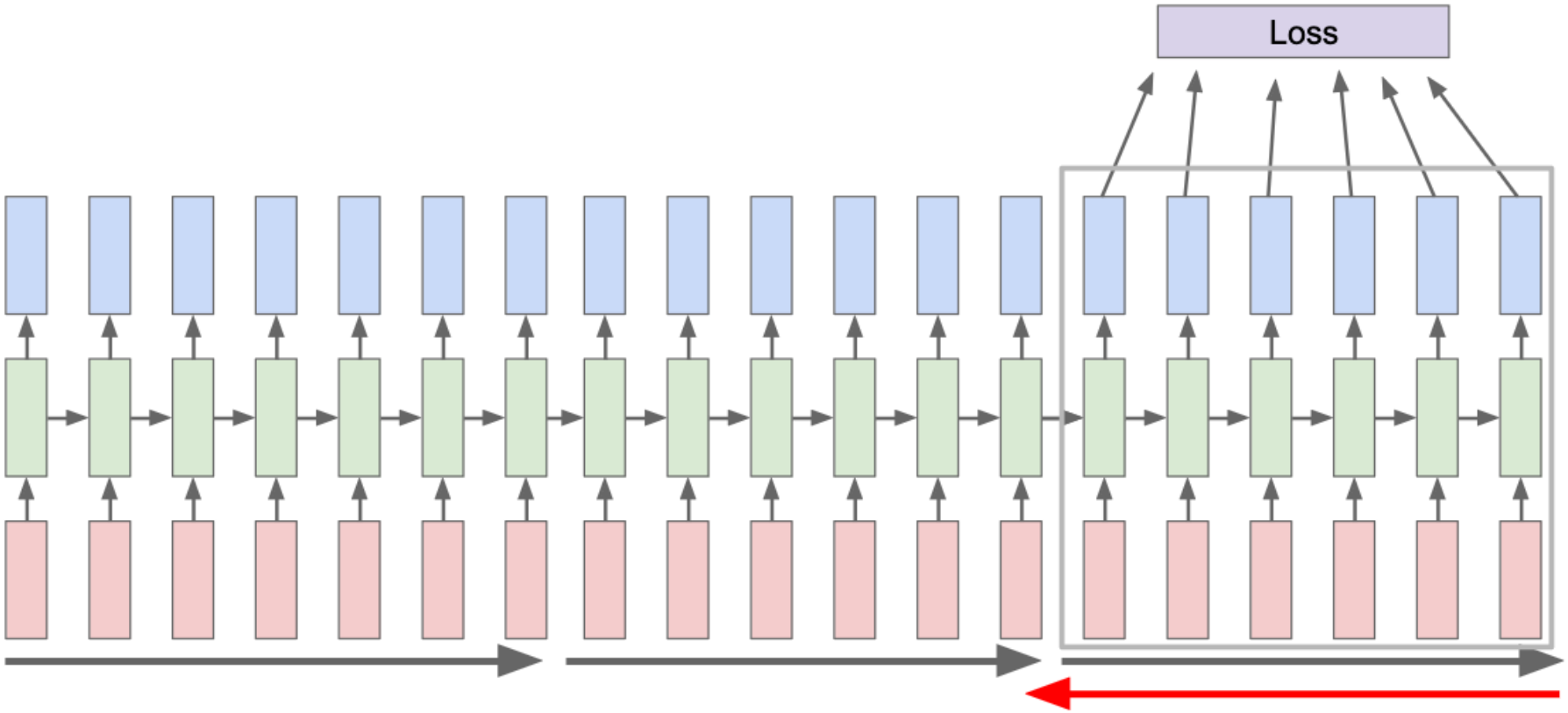
slide credit: Andrej Karpathy

Backpropagation Through Time



slide credit: Andrej Karpathy

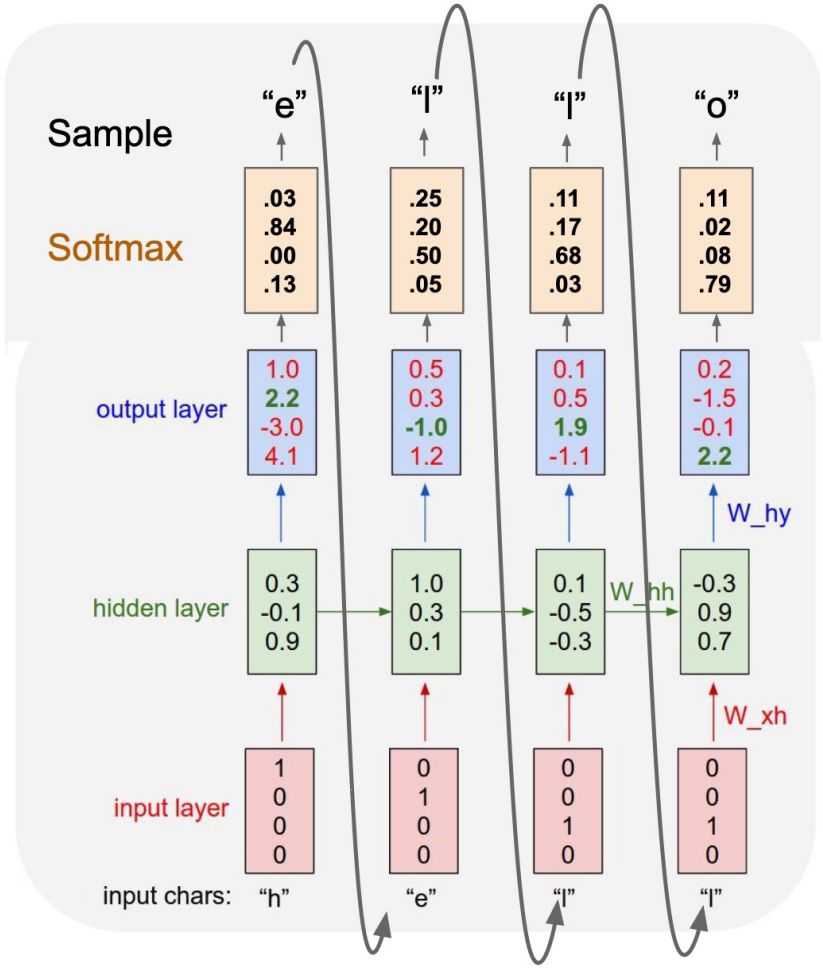
Truncated Backpropagation Through Time



slide credit: Andrej Karpathy

RNN-based LMs at Prediction Time

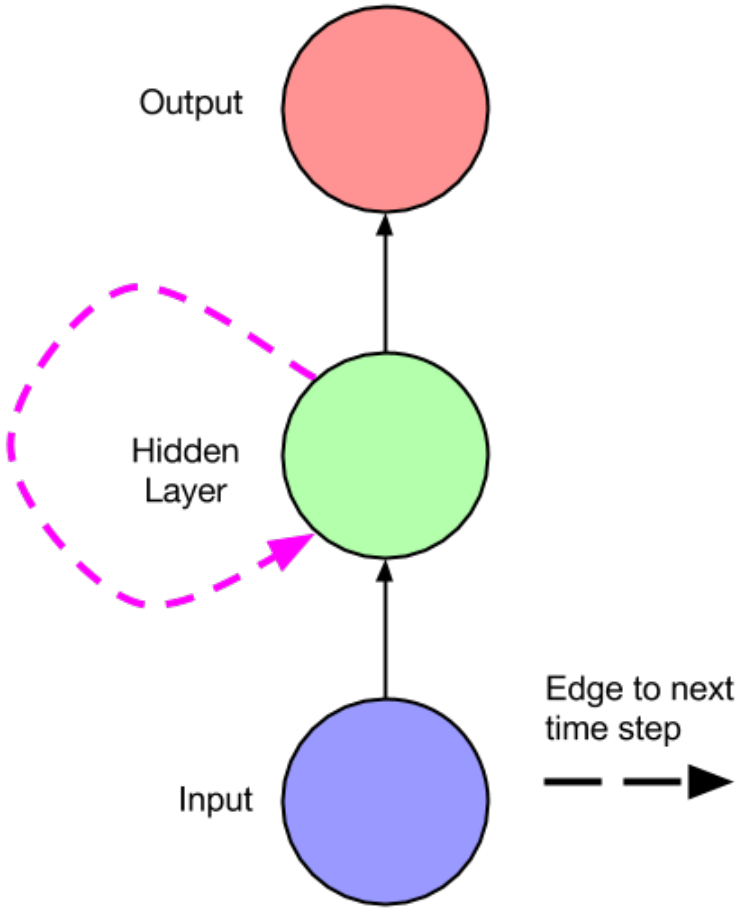
- Feed in some input prefix $x_{1:t}$
- At each step t, predict next token
- Sample from soft-max distribution
- Feed that token back to model as the subsequent input



slide credit: Andrej Karpathy

Vanishing / Exploding Gradients

- Recurrent neural networks are very deep along the sequence axis
- Depending on properties of recurrent weights, can lead to gradients exploding or vanishing
- Manny key innovations and training tricks in RNNs motivated by this problem

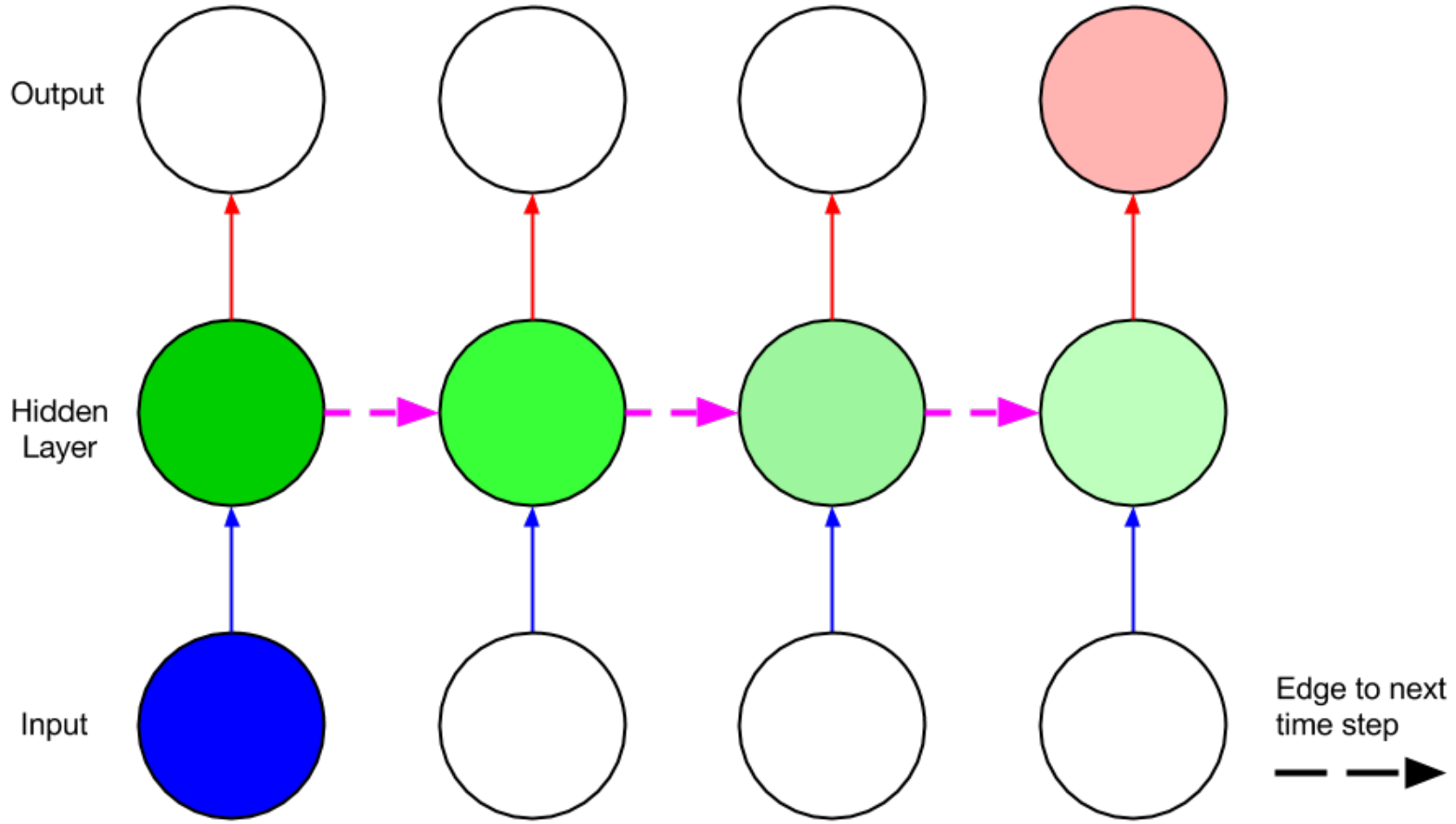


Gradient Clipping

- One strategy to mitigate the exploding gradient problem:
- Set an upper limit on the norm of the gradient
- Whenever gradient exceeds this norm, scale it down
- Apply SGD update on the rescaled gradient

$$\boldsymbol{\nabla_w^L} \leftarrow \begin{cases} \nabla_w^L & \text{if } ||\nabla L_w||_1 \leq \max \\ \max \cdot \nabla_w^L / ||\nabla_w^L||_1 & \text{if } ||\nabla_w^L||_1 > \max \end{cases}$$

Vanishing Gradients



RNN Training Details

- Initial value for hidden state h_0 typically set to 0
- Traditional RNNs used tanh activation, random init
- Some modern RNNs initialize weights as Identity matrix & use ReLU

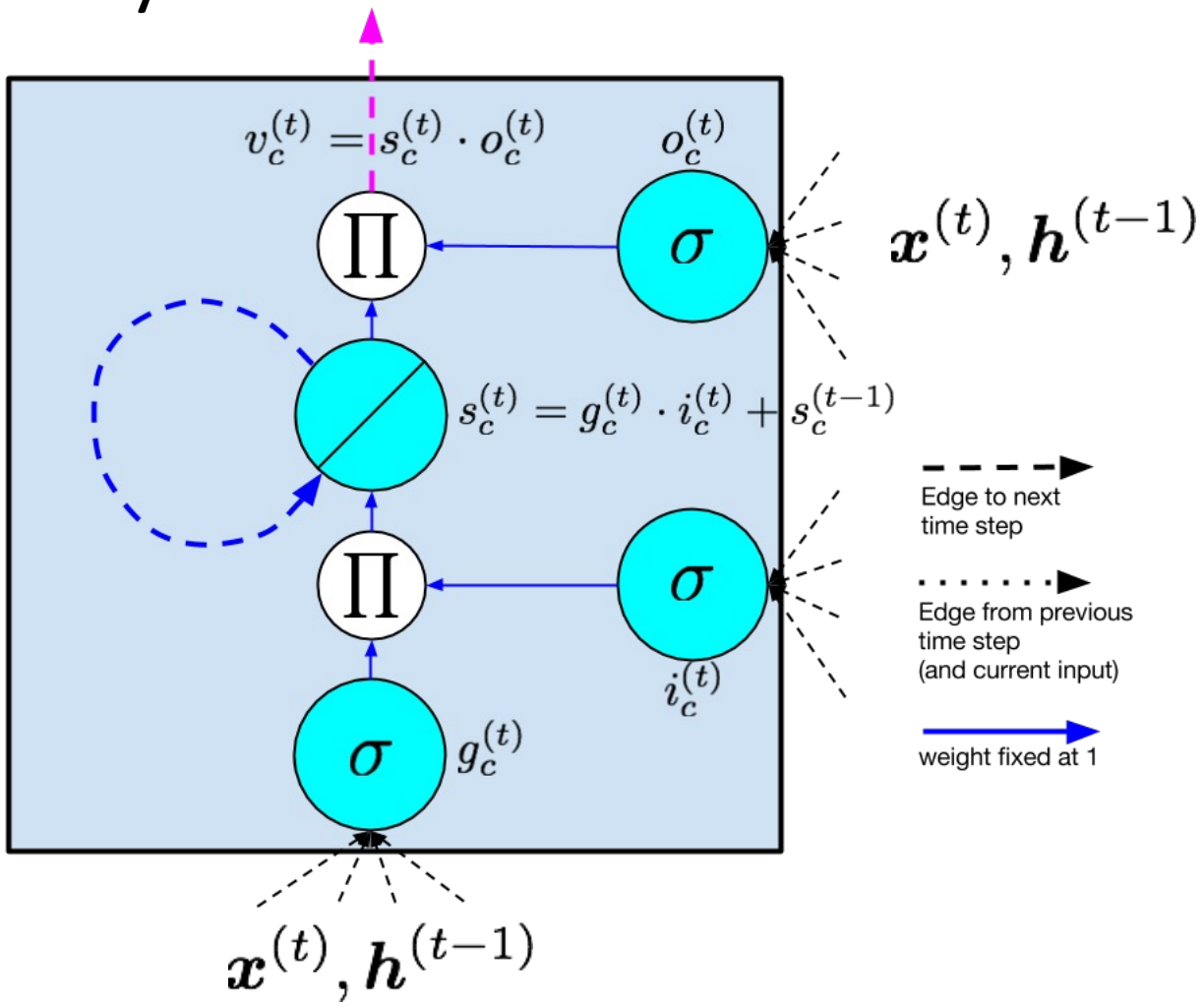
Le, Jaitly, Hinton (2015)

Long Short-Term Memory Cell

- Each **hidden node** replaced by a **memory cell**
- Designed to deal with learn long range time dependencies
- Internal state *s* gets residual self-connection (fixed weight of 1) *(similar idea to residual connections in ResNets)*
- Input gate *i* determines when to let activation into the cell
- Output Gate *o* determines when to let activation out of the cell

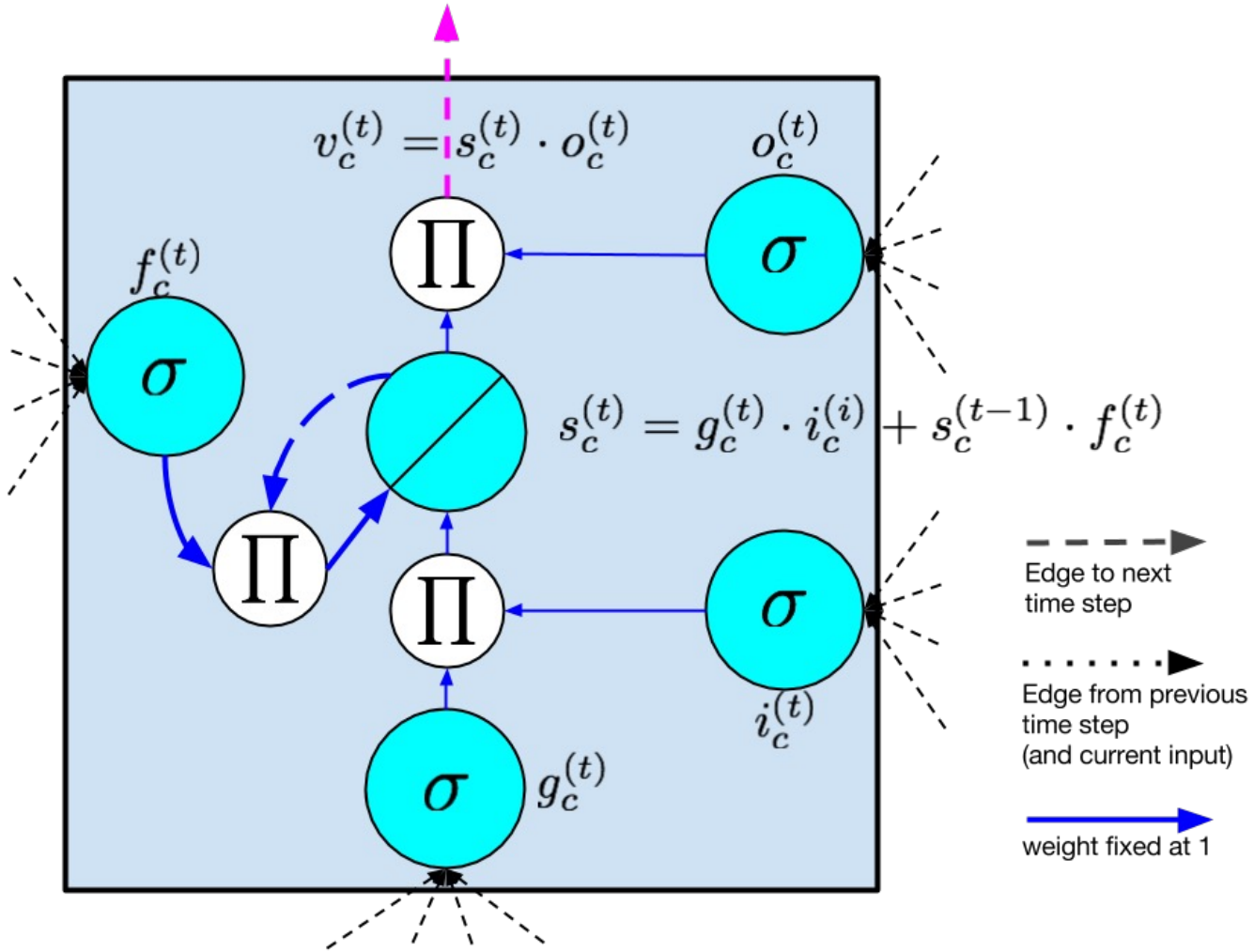
Hochreiter & Schmidhuber 1997

LSTM Memory Cell



Hochreiter & Schmidhuber 1997

Memory Cell with a Forget Gate



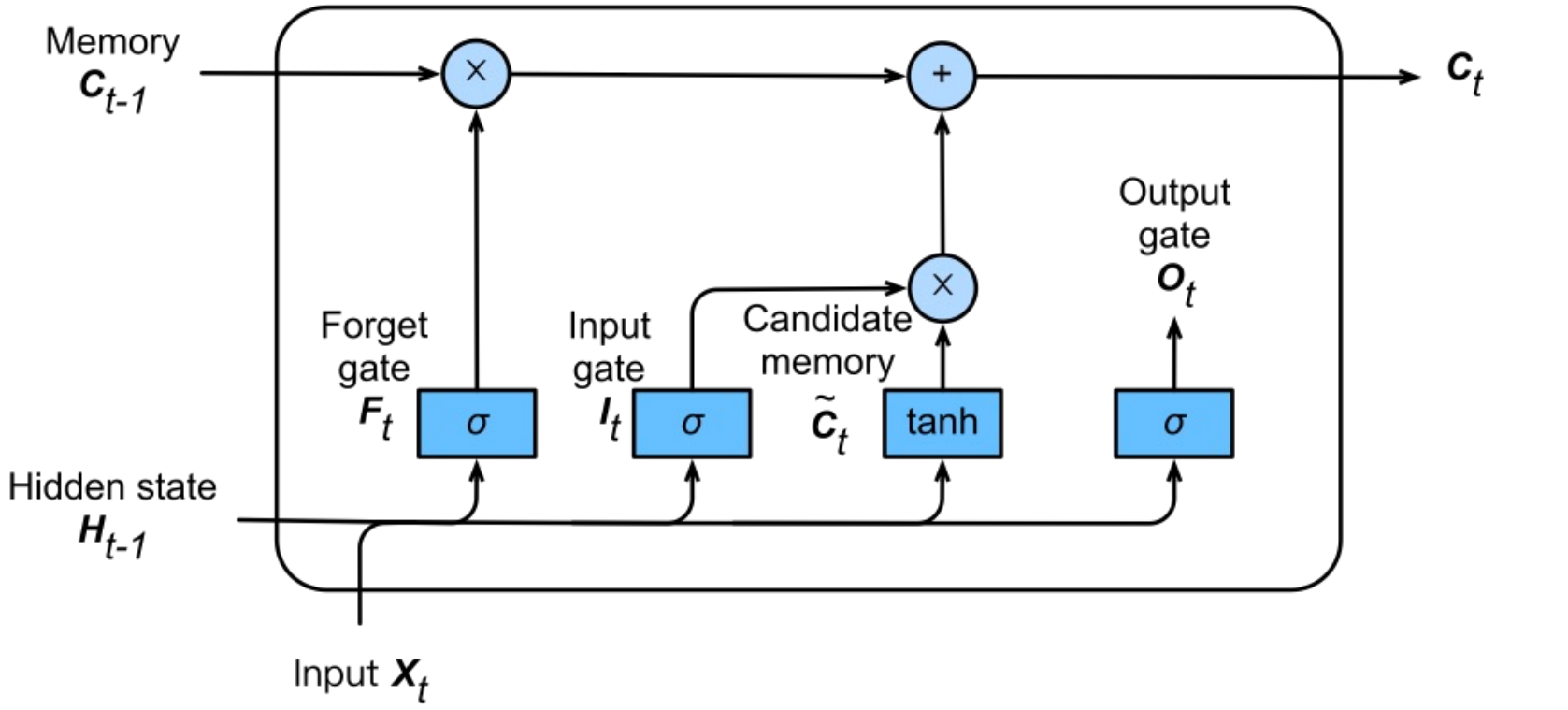
Gers et al., Neural Computation 2000

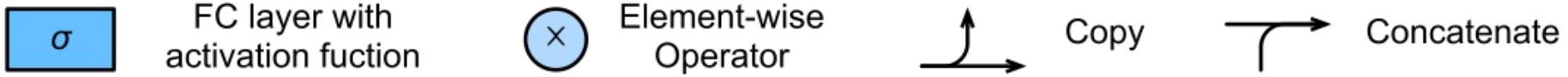
LSTM Forward Pass

$$
\begin{aligned}
\boldsymbol{g}^{(t)} &= \phi(W_{gx}\boldsymbol{x}^{(t)} + W_{ih}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_g) \\
\boldsymbol{i}^{(t)} &= \sigma(W_{ix}\boldsymbol{x}^{(t)} + W_{ih}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i) \\
\boldsymbol{f}^{(t)} &= \sigma(W_{fx}\boldsymbol{x}^{(t)} + W_{fh}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_f) \\
\boldsymbol{o}^{(t)} &= \sigma(W_{ox}\boldsymbol{x}^{(t)} + W_{oh}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o) \\
\boldsymbol{s}^{(t)} &= \boldsymbol{g}^{(t)} \odot \boldsymbol{i}^{(i)} + \boldsymbol{s}^{(t-1)} \odot \boldsymbol{f}^{(t)} \\
\boldsymbol{h}^{(t)} &= \boldsymbol{s}^{(t)} \odot \boldsymbol{o}^{(t)}
\end{aligned}
$$

Memory Cell Update

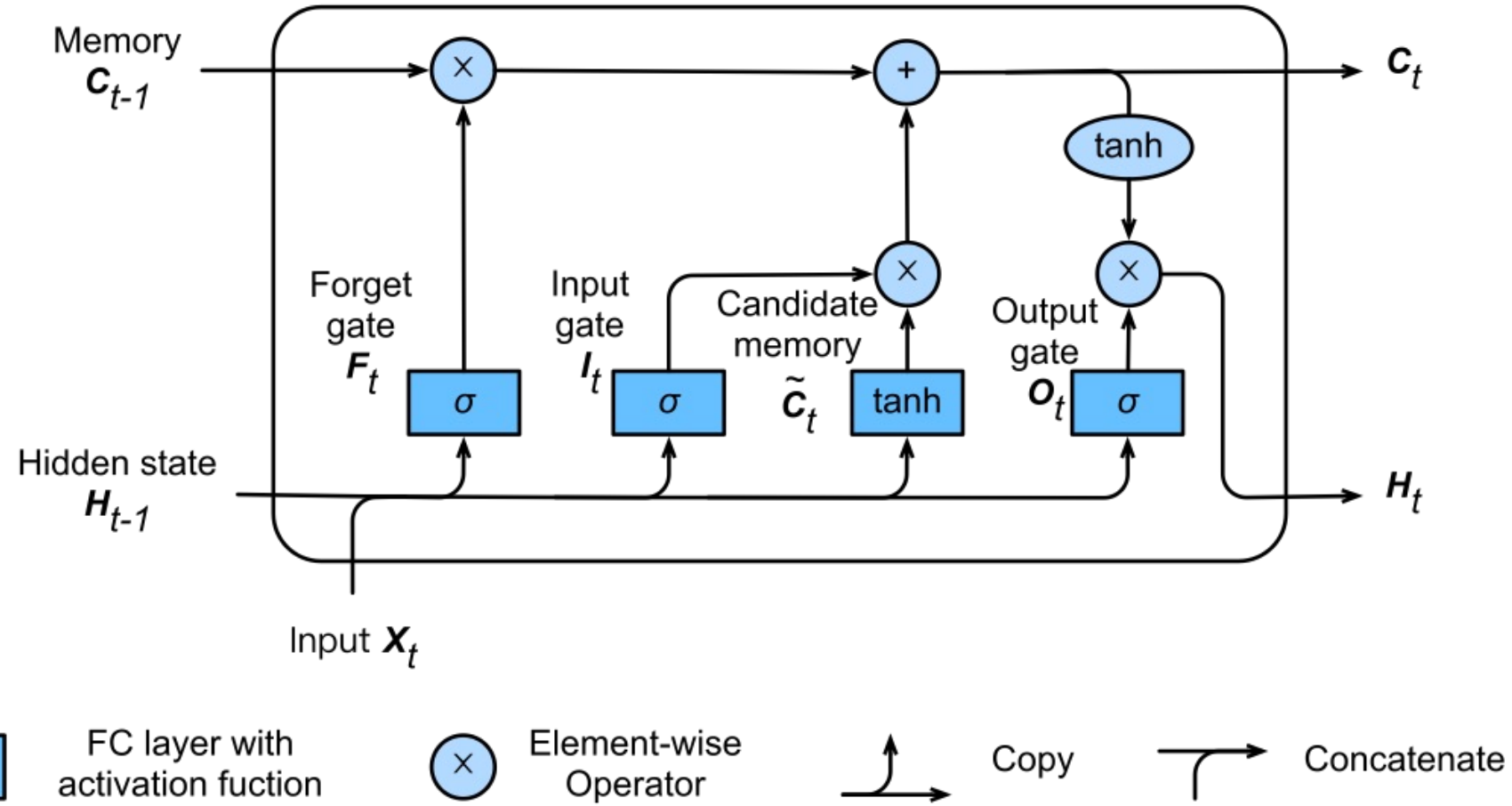
$$\boldsymbol{C}_t = \boldsymbol{F}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{I}_t \odot \tilde{\boldsymbol{C}}_t$$



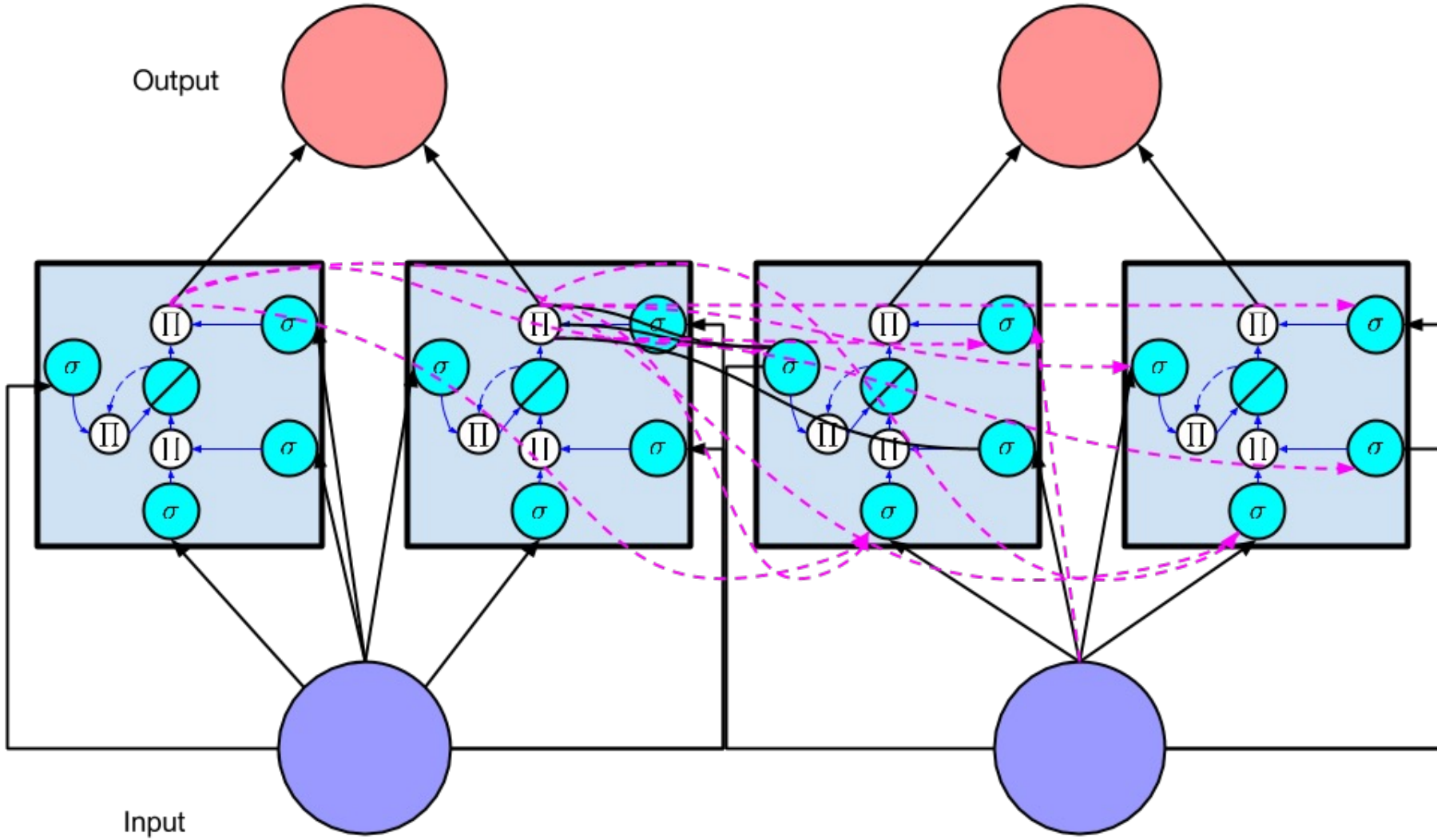


Generating Memory Cell Output

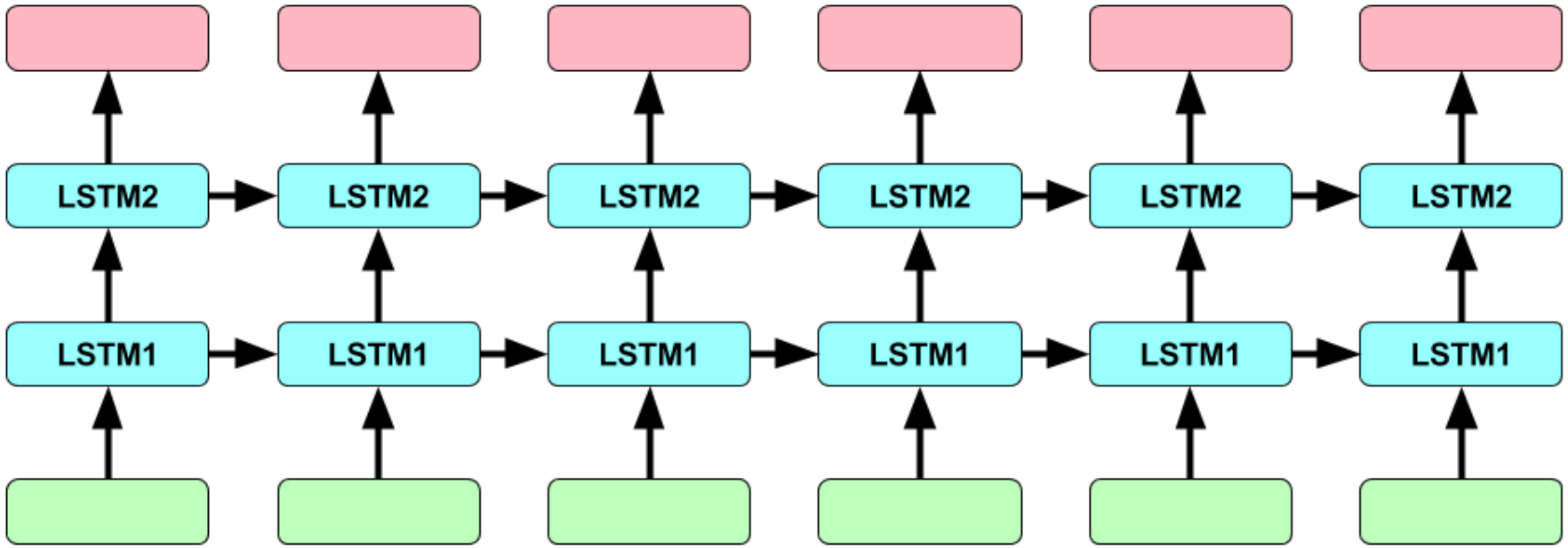
$$\boldsymbol{H}_t = \boldsymbol{O}_t \odot tanh(\boldsymbol{C}_t)$$



Wiring together the LSTM cells



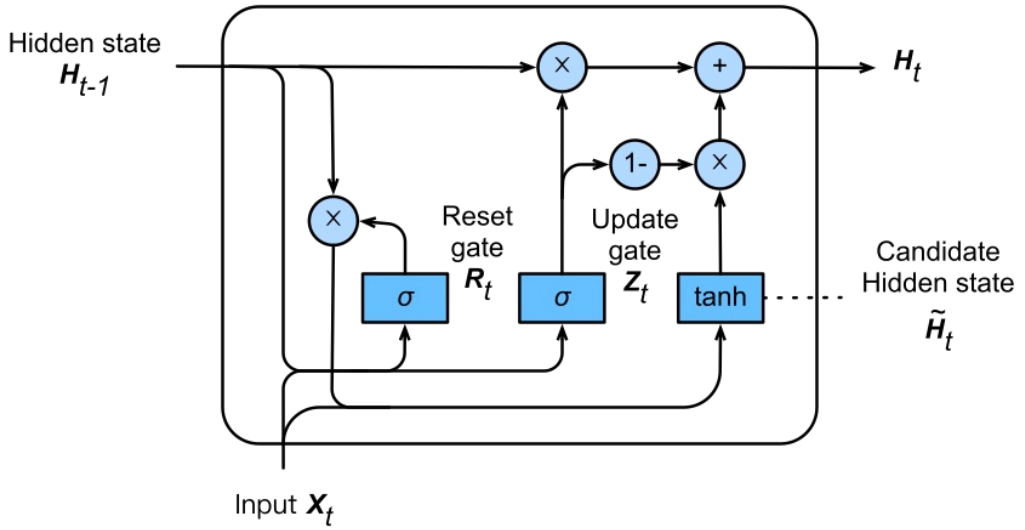
Deep LSTMs (stack layers vertically)



Gated Recurrent Unit (GRUs)

- Simplified version of LSTM cell
- Only two gates (not 3)
- Less parameters than LSTM
- Similar performance
- Keeps core ideas

$$\boldsymbol{R}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xr} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hr} + \boldsymbol{b}_r),$$
$$\boldsymbol{Z}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xz} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hz} + \boldsymbol{b}_z)$$
$$\tilde{\boldsymbol{H}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xh} + \left(\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}\right) \boldsymbol{W}_{hh} + \boldsymbol{b}_h)$$
$$\boldsymbol{H}_t = \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + (1 - \boldsymbol{Z}_t) \odot \tilde{\boldsymbol{H}}_t$$



Training & Generating Shakespeare

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.

Training Data:
All of Shakespeare

Size: 4.4MB

Unreasonable Effectiveness of RNNs (Karpathy 2015)

Further RNN Architecture Exploration

- Stacked LSTM (2013)
- Grid LSTM (2015)
- Recurrent Highway Nets (Zilly et al 2016)
- NAS-based memory cell (Zoph et al. 2016)
- Pointer Sentinal LSTMs (Merity et al. 2016)

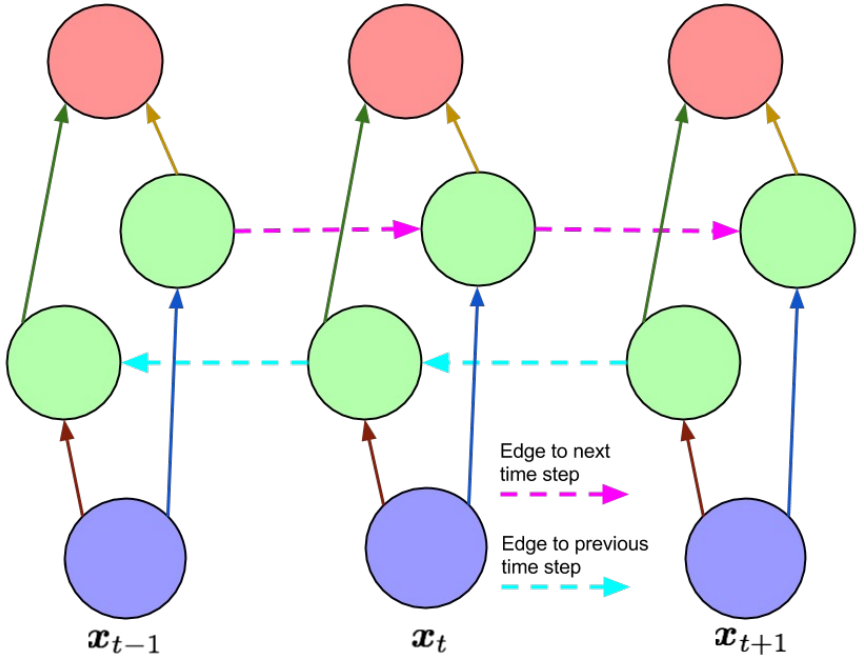
On the State of the Art of Evaluation in Neural Language Models

Gábor Melis[†], **Chris Dyer**[†], **Phil Blunsom**[†‡]
{melisgl,cdyer,pblunsom}@google.com
[†]DeepMind
[‡]University of Oxford

Abstract

Ongoing innovations in recurrent neural network architectures have provided a steady influx of apparently state-of-the-art results on language modelling benchmarks. However, these have been evaluated using differing codebases and limited computational resources, which represent uncontrolled sources of experimental variation. We reevaluate several popular architectures and regularisation methods with large-scale automatic black-box hyperparameter tuning and arrive at the somewhat surprising conclusion that standard LSTM architectures, when properly regularised, outperform more recent models. We establish a new state of the art on the Penn Treebank and Wikitext-2 corpora, as well as strong baselines on the Hutter Prize dataset.

BRNN
(Schuster & Paliwal, 1997)



$$
\begin{aligned}
\boldsymbol{h}_f^{(t)} &= \sigma(W_{h_f x}\boldsymbol{x}^{(t)} + W_{h_f h_f}\boldsymbol{h}_f^{(t-1)} + \boldsymbol{b}_{h_f}) \\
\boldsymbol{h}_b^{(t)} &= \sigma(W_{h_b x}\boldsymbol{x}^{(t)} + W_{h_b h_b}\boldsymbol{h_b}^{(t+1)} + \boldsymbol{b}_{h_b}) \\
\hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(W_{y h_f}\boldsymbol{h_f}^{(t)} + W_{y h_b}\boldsymbol{h_b}^{(t)} + \boldsymbol{b}_y)
\end{aligned}
\tag{1}
$$

Applying Dropout to RNNs

- Recurrent Dropout (Gal & Gharamani 2015):
  - Apply same dropout mask at each time step.
- Another variant Semeniuta et al. (2016):
  - Apply dropout to LSTM “update vector” g, not to hidden state.

$$\mathbf{c}_t = \mathbf{f}_t * \mathbf{c}_{t-1} + \mathbf{i}_t * d(\mathbf{g}_t)$$

- Zoneout (Krueger et al. 2017):
  - Rather than zero–out units, replace with previous step’s activations.
  - Gradient still glows through zone-out units.