PennState
College of the
Liberal Arts

C-SoDA
Center for Social Data Analytics

Day 5 (6) - Introduction to Neural Nets / Deep Learning for NLP

Advanced Text as Data: Natural Language Processing
Essex Summer School in Social Science Data Analysis

Burt L. Monroe (Instructor) & Sam Bestvater (TA)
Pennsylvania State University

July 30 (Aug 2), 2021

Today

- Regularization
  - Early stopping
  - Dropout
  - L1/L2 weight regularization
  - Data augmentation
- Optimizers / learning rates / adaptive learners
- Embeddings
  - Using pretrained embeddings
  - Training embedding layers
  - Visualizing embeddings with TensorBoard Projector
- Received wisdom on deep learning.
- Modeling sequence with recurrent neural nets (RNNs/LSTMs)

Today

- Regularization
  - Early stopping
  - Dropout
  - L1/L2 weight regularization
  - Data augmentation
- Optimizers / learning rates / adaptive learners
- Embeddings
  - Using pretrained embeddings
  - Training embedding layers
  - Visualizing embeddings with TensorBoard Projector
- Received wisdom on deep learning.
- Modeling sequence with recurrent neural nets (RNNs/LSTMs)

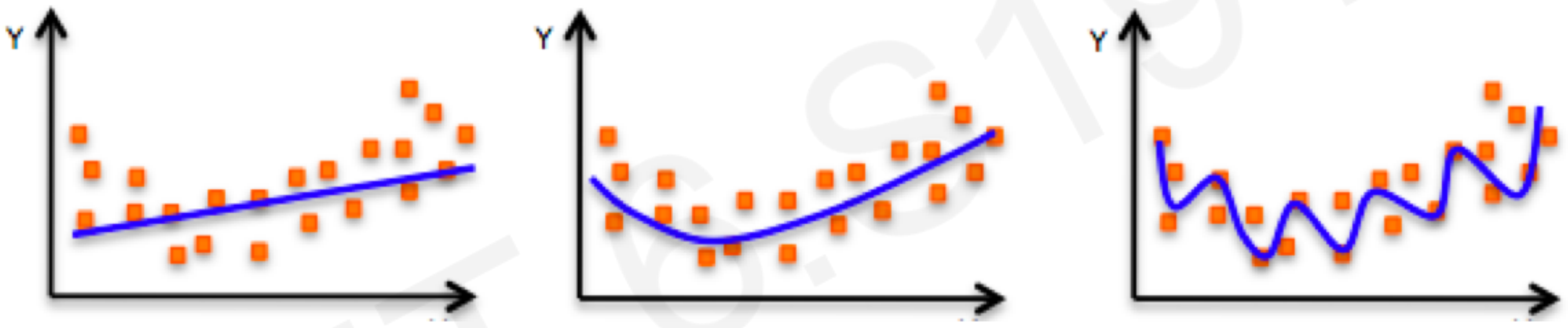
Tomorrow

- Recurrent neural nets (RNNs) / LSTMs / bi-LSTMs / GRUs
- Convolutional neural nets (CNNs)
- Attention
- Self-attention and transformers

Overfitting and Regularization

Neural Networks in Practice: Overfitting

The Problem of Overfitting



Underfitting
Model does not have capacity to fully learn the data

← **Ideal fit** →

Overfitting
Too complex, extra parameters, does not generalize well

Regularization

What is it?

Technique that constrains our optimization problem to discourage complex models

Regularization

What is it?

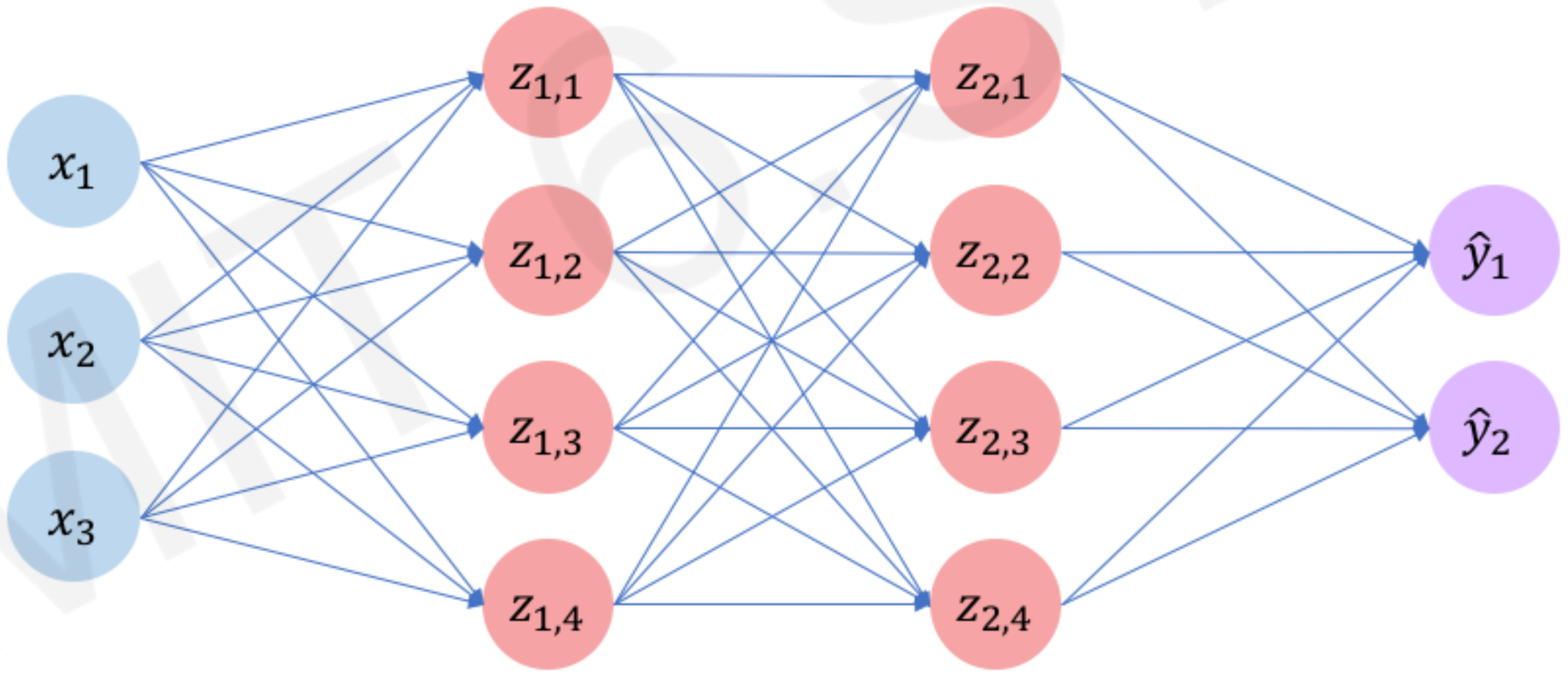
Technique that constrains our optimization problem to discourage complex models

Why do we need it?

Improve generalization of our model on unseen data

Regularization 1: Dropout

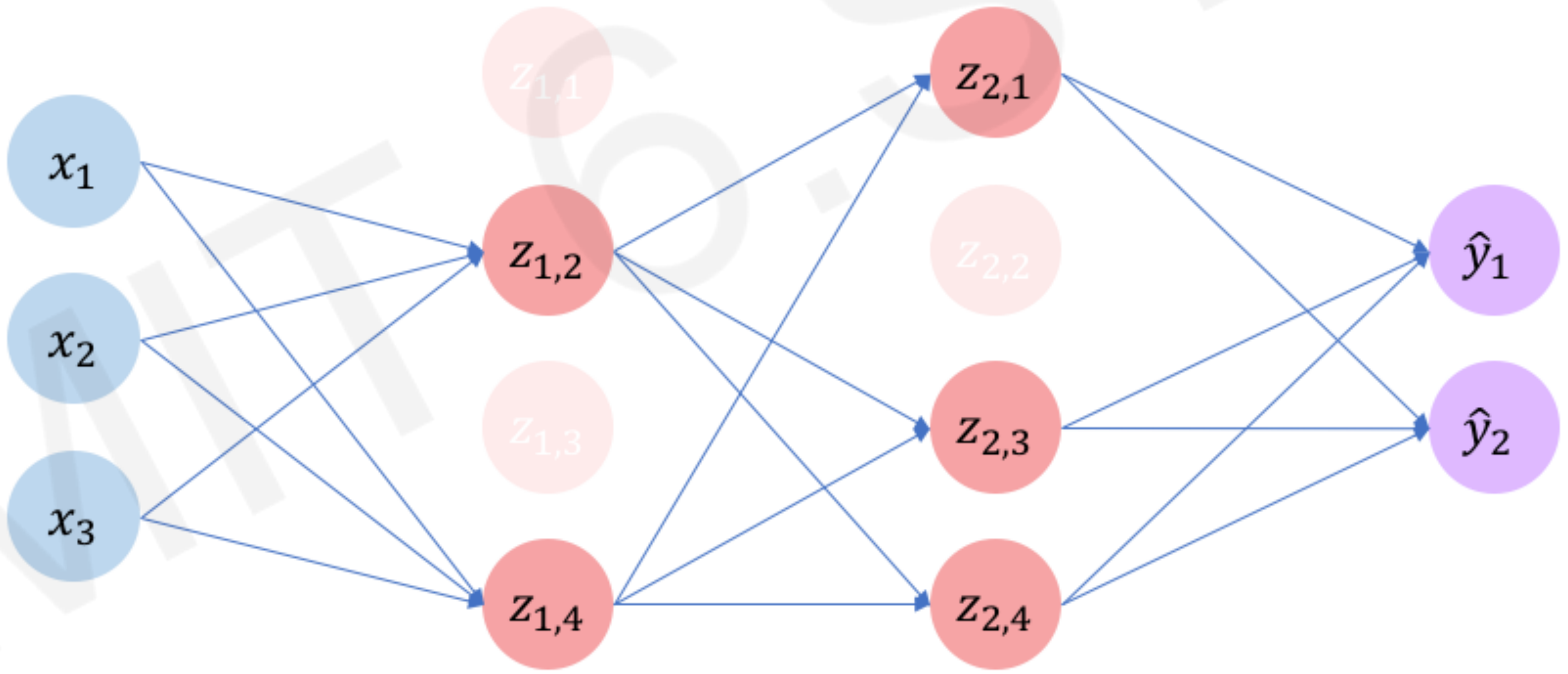
- During training, randomly set some activations to 0



Regularization 1: Dropout

- During training, randomly set some activations to 0
  - Typically 'drop' 50% of activations in layer
  - Forces network to not rely on any 1 node

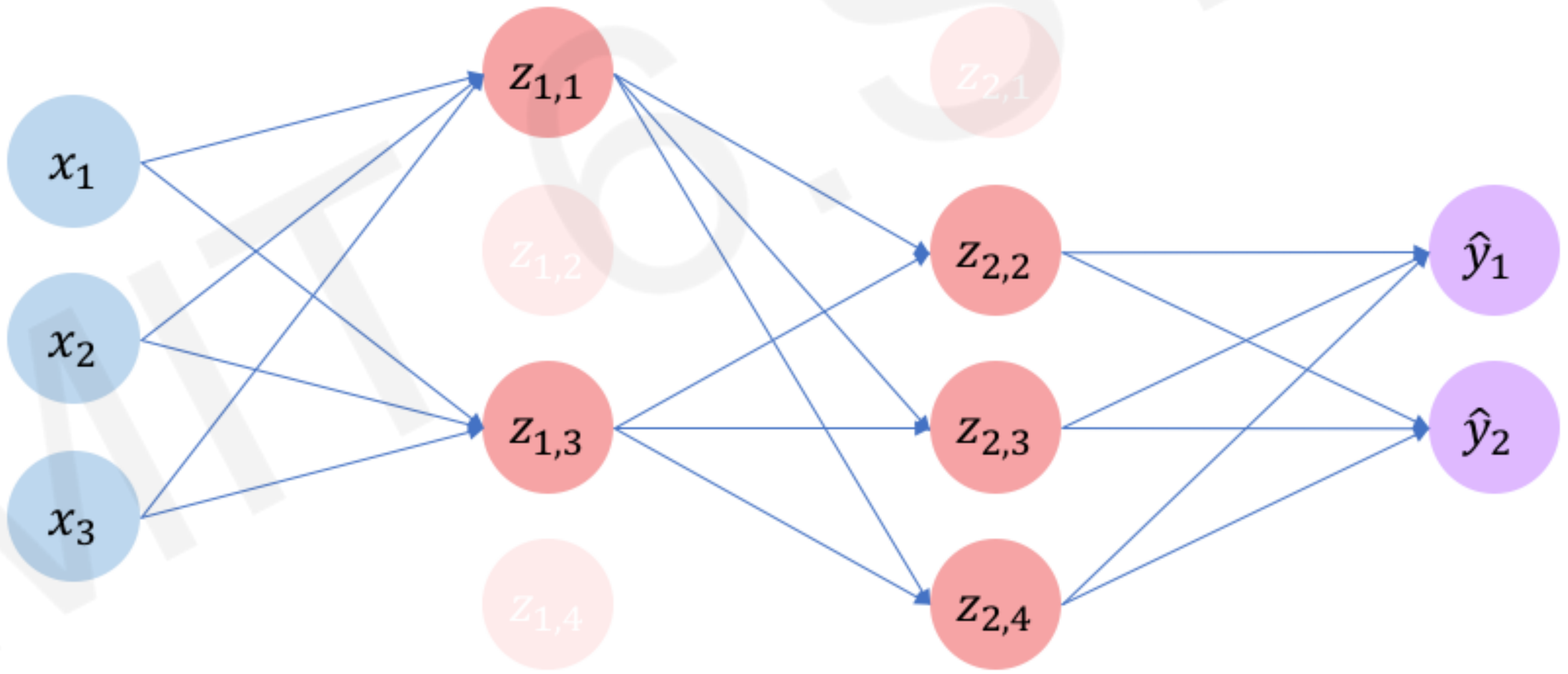
```
tf.keras.layers.Dropout(p=0.5)
```



Regularization 1: Dropout

- During training, randomly set some activations to 0
  - Typically 'drop' 50% of activations in layer
  - Forces network to not rely on any 1 node

```
tf.keras.layers.Dropout(p=0.5)
```



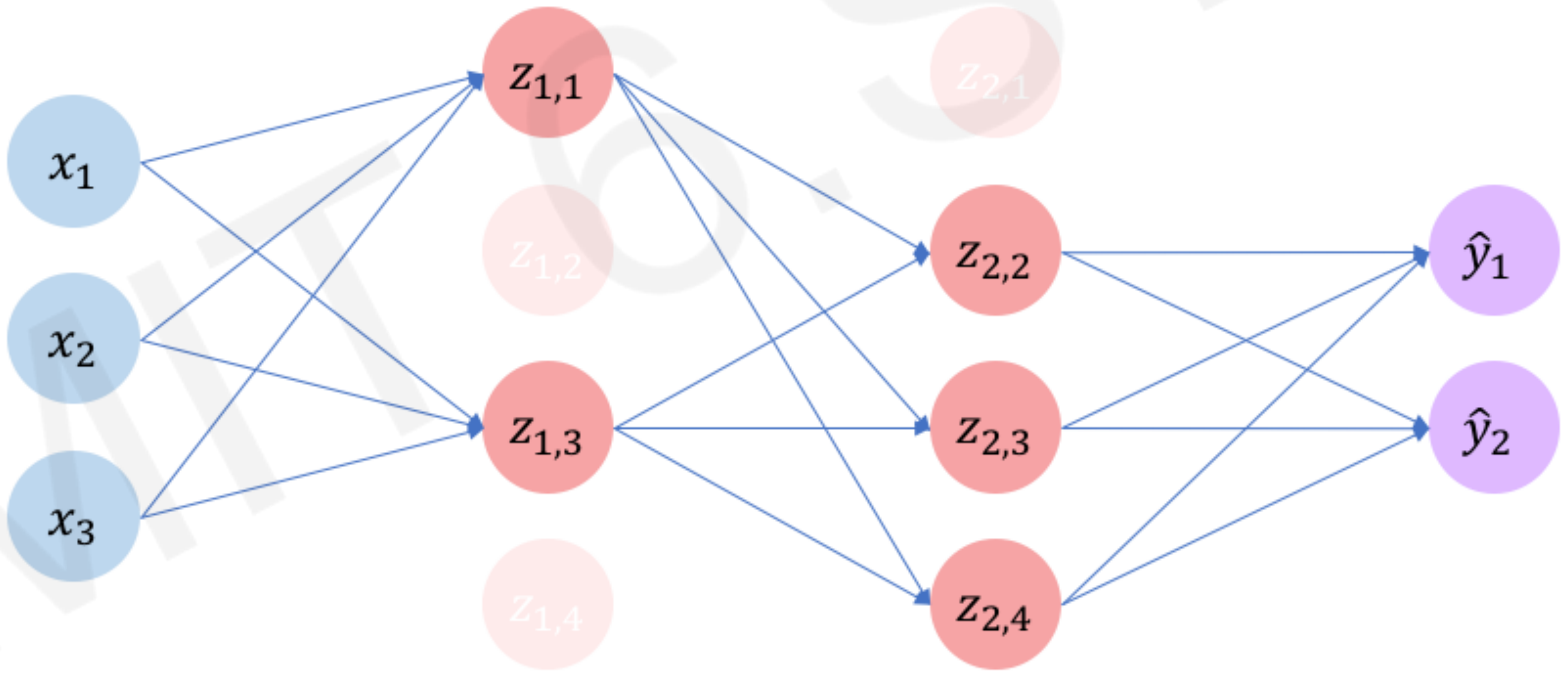
Dropout can be thought of as ensembling or model averaging.

Somewhat like random forests, bagging, boosting

Regularization 1: Dropout

- During training, randomly set some activations to 0
  - Typically 'drop' 50% of activations in layer
  - Forces network to not rely on any 1 node

```
tf.keras.layers.Dropout(p=0.5)
```



Regularization 2: Early Stopping

- Stop training before we have a chance to overfit

Regularization 2: Early Stopping

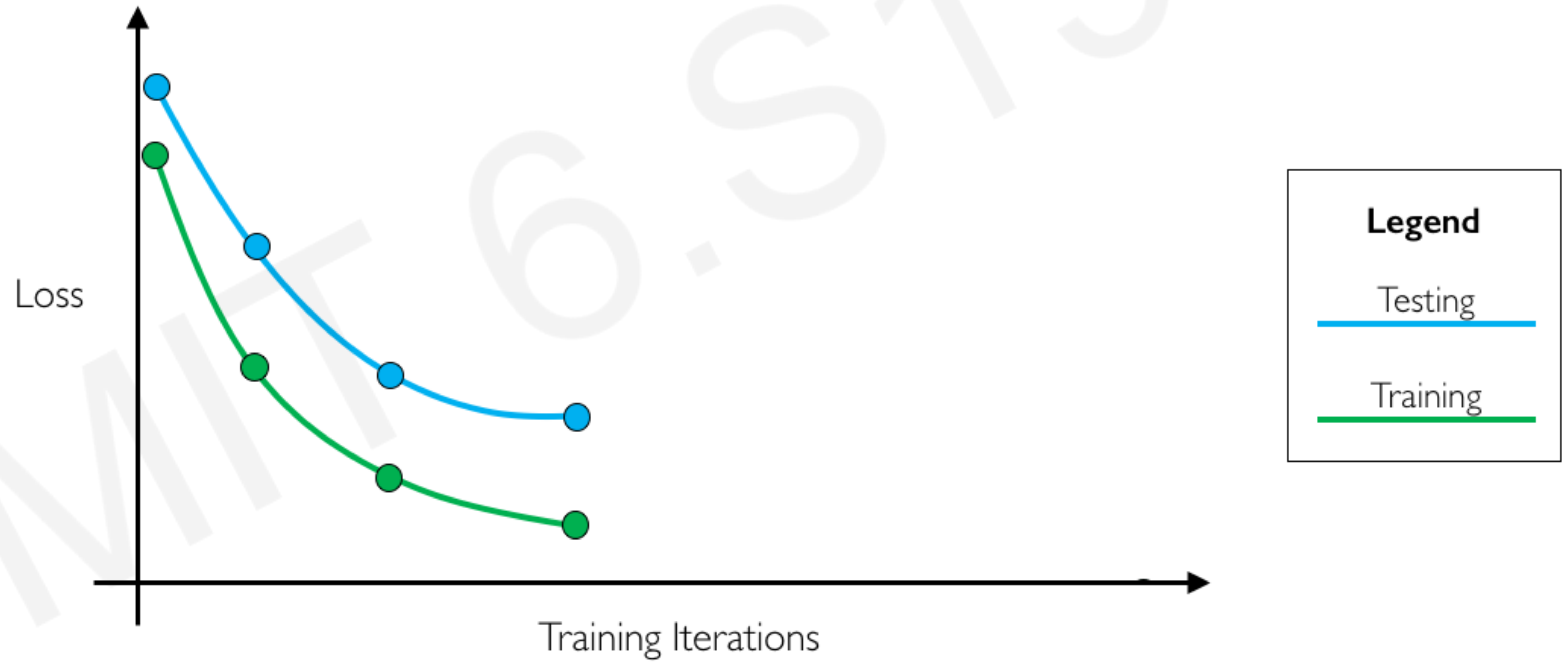
- Stop training before we have a chance to overfit

Regularization 2: Early Stopping

- Stop training before we have a chance to overfit

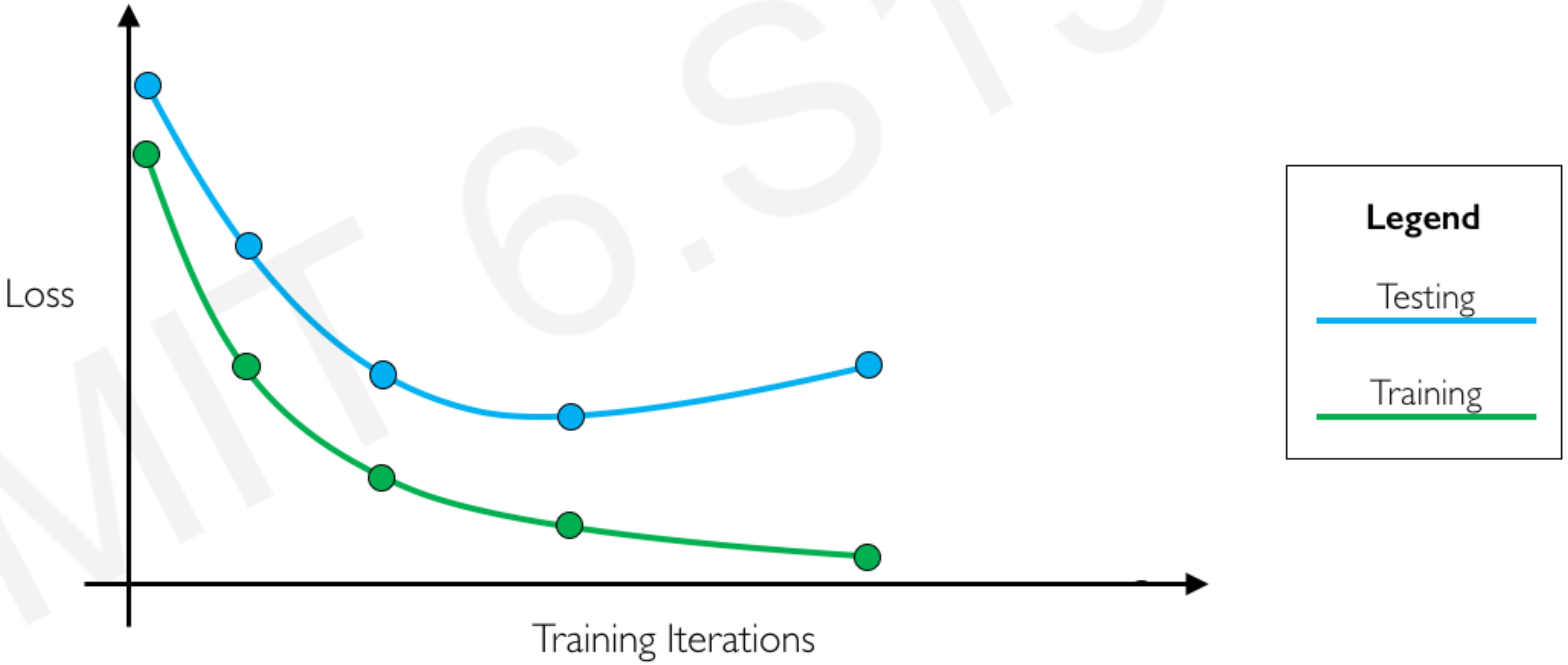
Regularization 2: Early Stopping

- Stop training before we have a chance to overfit



Regularization 2: Early Stopping

- Stop training before we have a chance to overfit



Regularization 2: Early Stopping

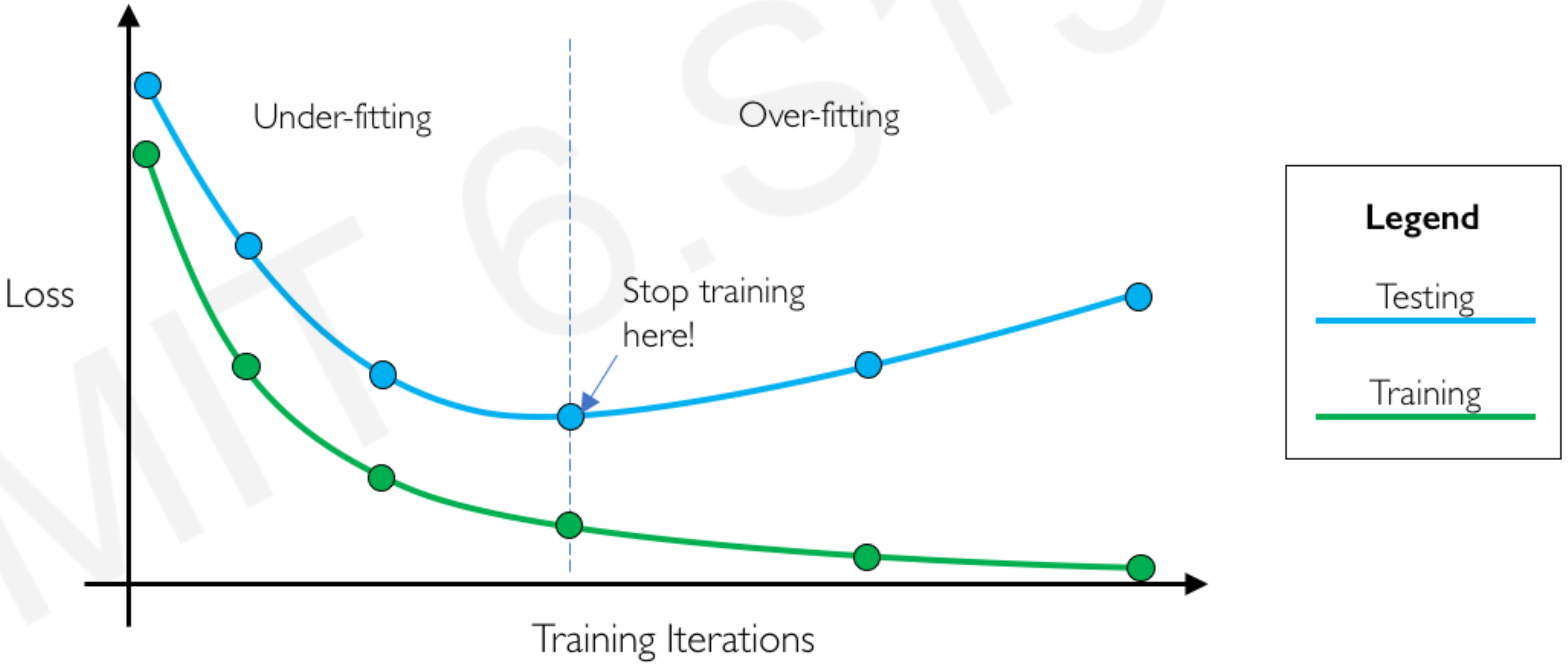
- Stop training before we have a chance to overfit

Regularization 2: Early Stopping

- Stop training before we have a chance to overfit

Regularization 2: Early Stopping

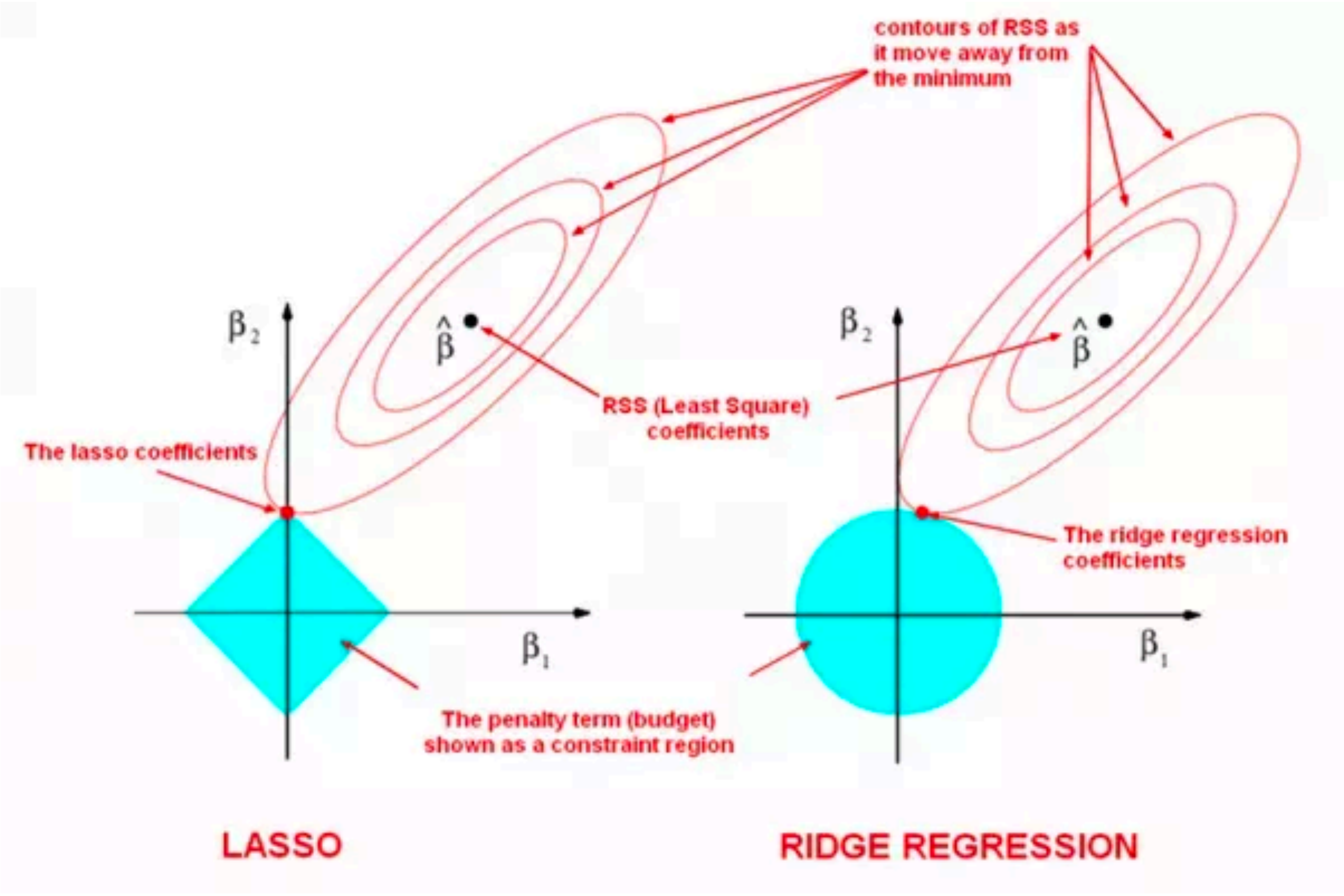
- Stop training before we have a chance to overfit



Regularization applied to

- Model structure / model averaging (e.g., dropout)
- Parameters / weights (e.g., L1 or L2 penalty, weight decay)
- Data -
    - Smoothing, filtering (related to convolution / kernel smoothing)
    - Noise / differential privacy (e.g., Laplacian mechanism)
    - Priors / pseudodata (e.g., Laplace L1 or Gaussian L2)
    - Data augmentation

L1/L2 Regularization, LASSO/Ridge Regression, Laplace/Gaussian Noise/Prior/Pseudodata

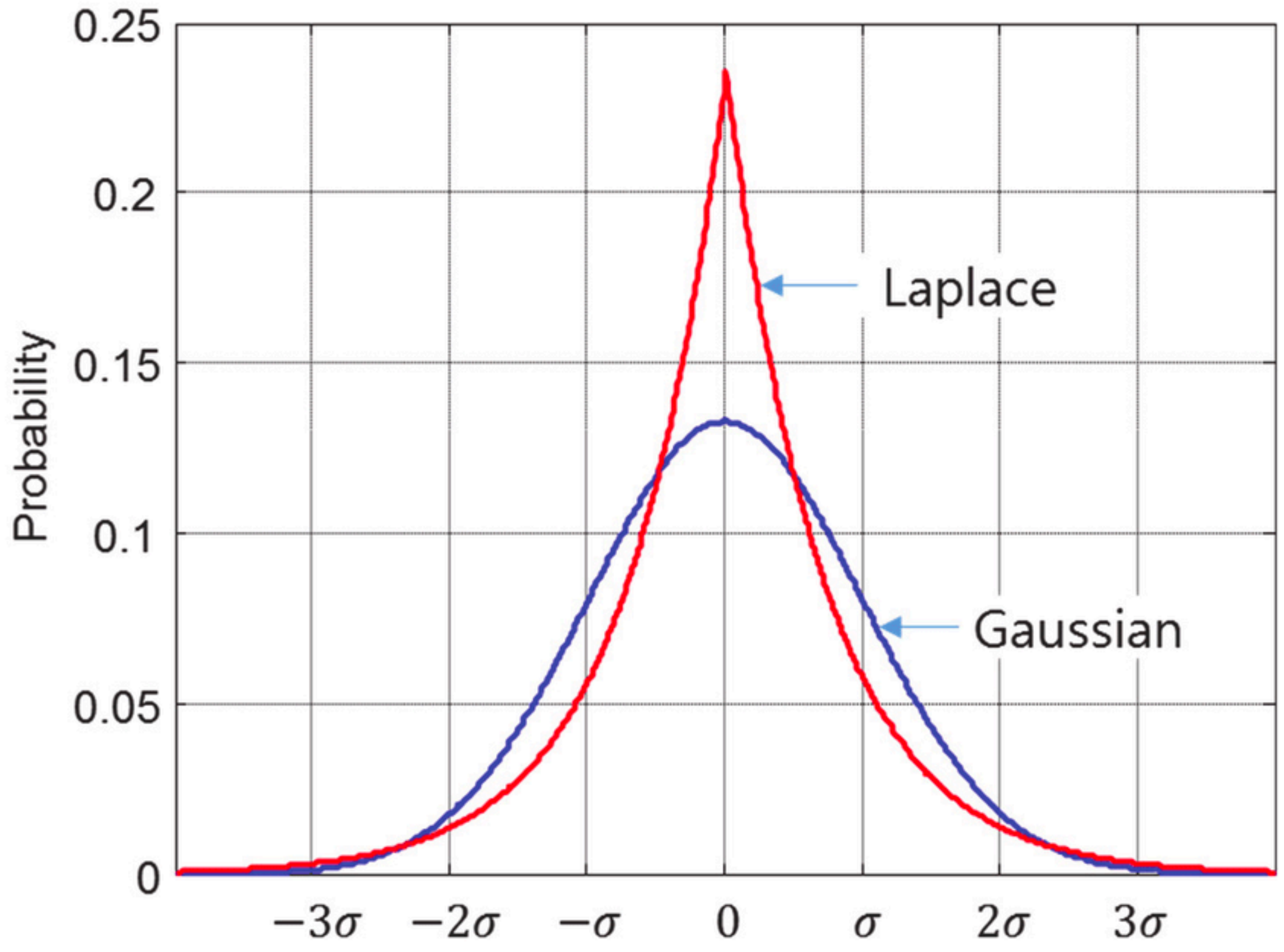


$$\lambda \sum_{j=0}^{M} |W_j|$$

L1 Penalty

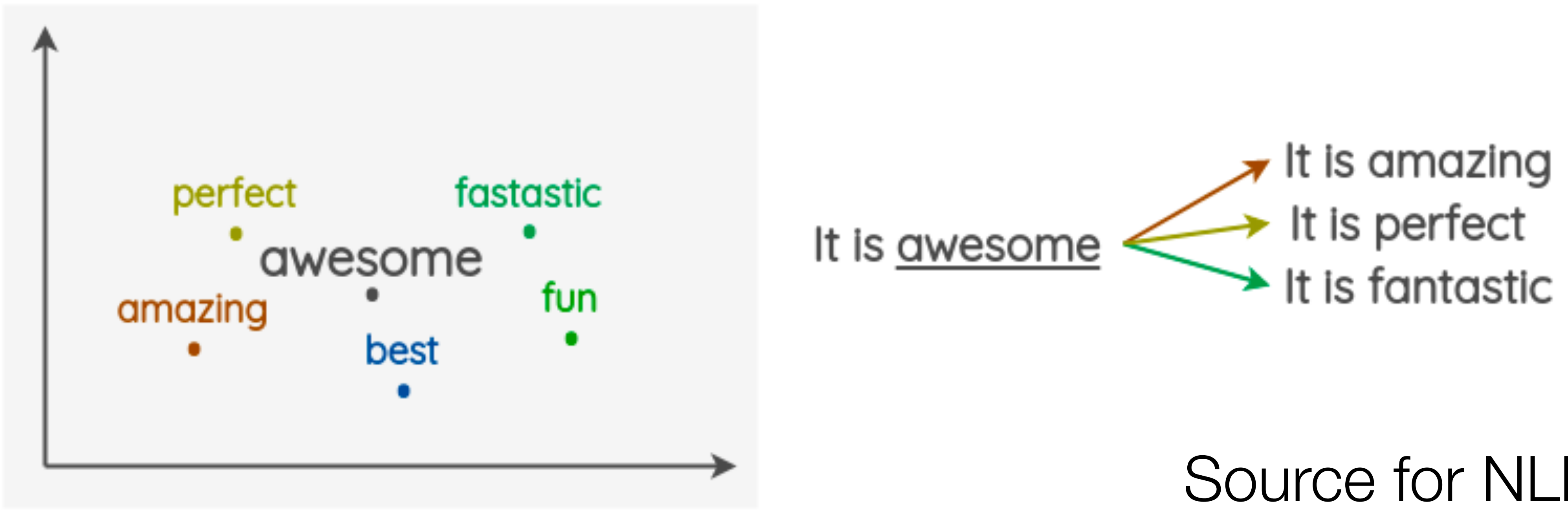
$$\lambda \sum_{j=0}^{M} W_j^2$$

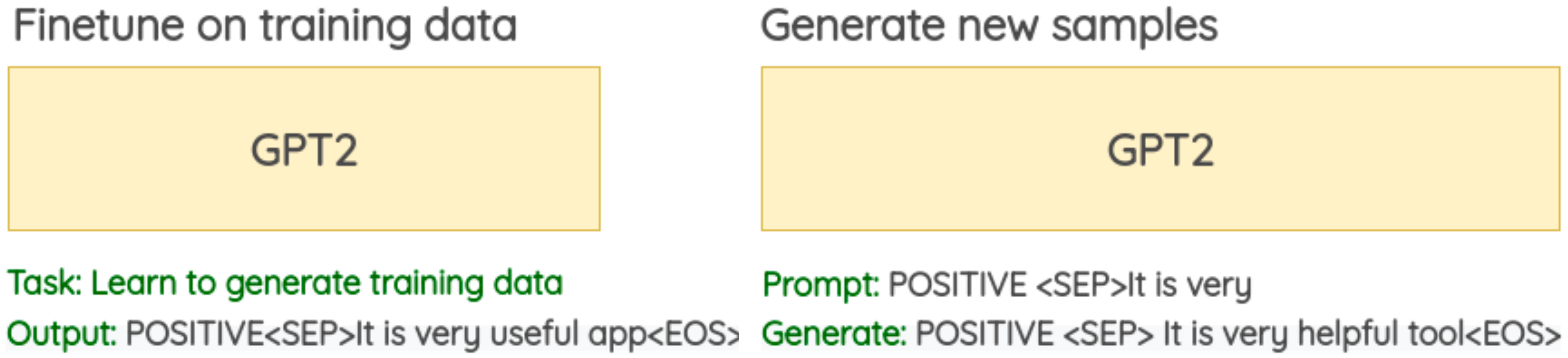
L2 Penalty



Data augmentation

Nearest neighbors in word2vec





Source for NLP illustrations: Amit Chaudhary (2020) "A Visual Survey of Data Augmentation in NLP.

Dropout & Regularization (Text Classification Notebook 2)

Optimizers https://cs231n.github.io/neural-networks-3/

Received Wisdom on Building Neural Nets

Architecture

- Transfer learning, if possible, otherwise start with copying the architecture of others who have worked on the problem.
- Experiment, and make decisions based on validation error.
- Deeper (more layers) and thinner (fewer nodes per layer) networks are (a) more difficult to optimize, but (b) more likely to generalize well.
- Some say start with 2 hidden layers, number of nodes a power of 2, second layer 1/2 the size of the first.

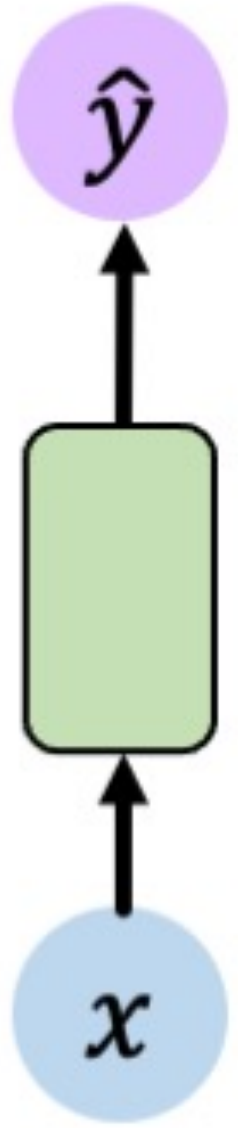
Training

- Always use early stopping.
- Dropout is often advisable. < 50% on hidden layers, 0-20% on input layers
- 5,000+ observations per category for acceptable performance (this advice is now too conservative for problems that can be informed by pretained embeddings or language models).
- Use k-fold validation (instead of validation/train/test split) for smaller datasets.
- Use as large a batch size as the GPU can handle. Start at 16 for really large models and increase in powers of 2.
- For classification with unbalanced data, set class weights in your loss functions.
- Monitor activation histograms. (e.g., TensorBoard)

Embeddings (Text Classification Notebooks 3 & 4)

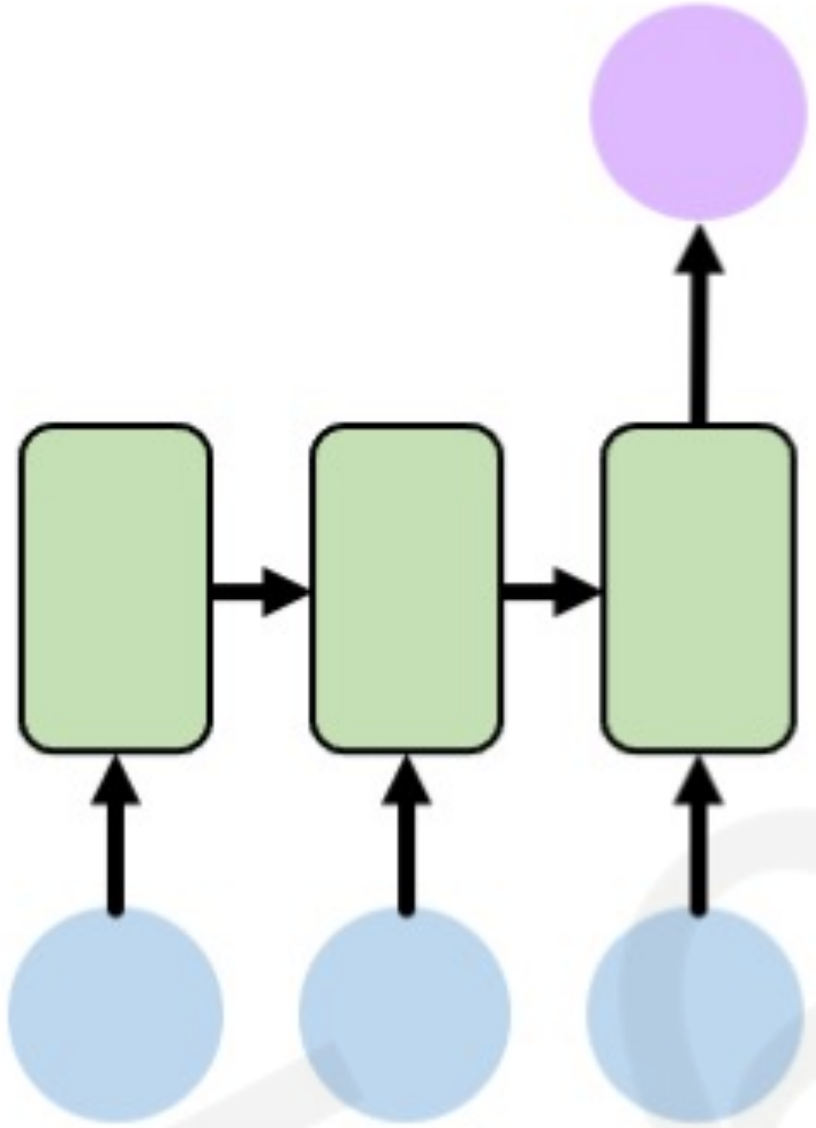
Modeling sequence with recurrence

Sequence Modeling Applications

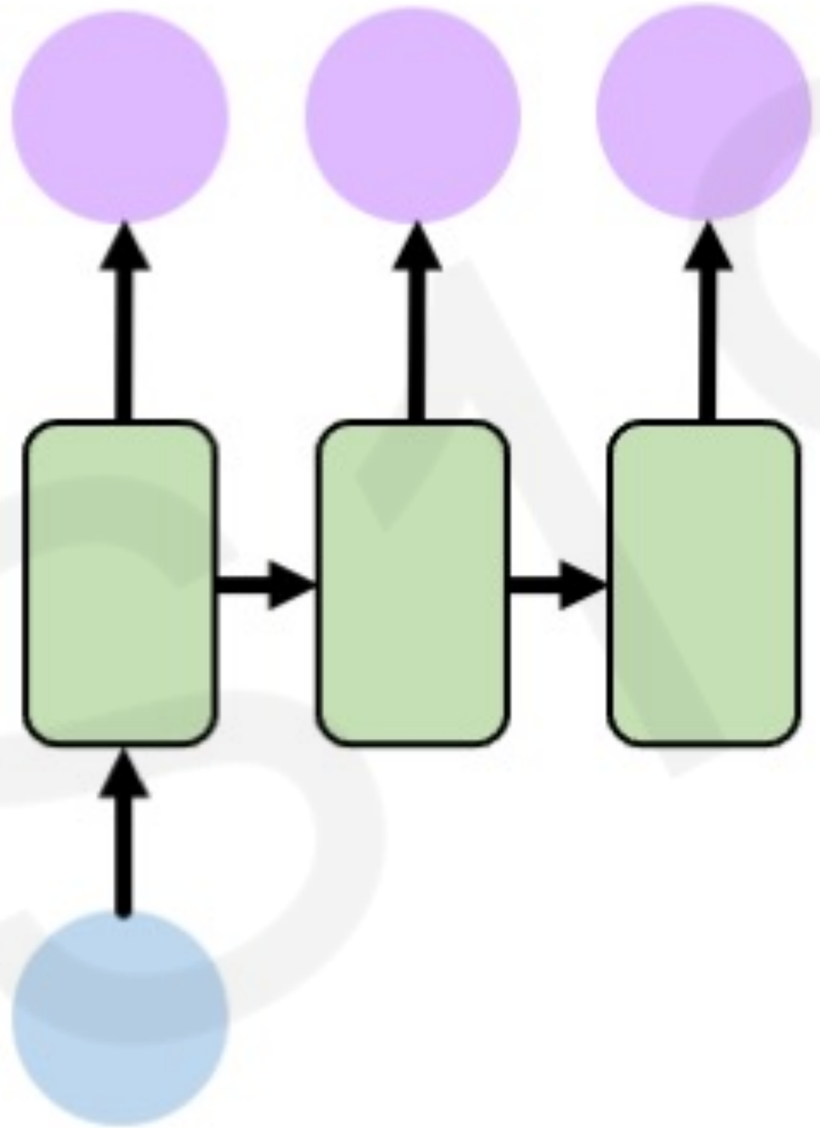


One to One
Binary Classification

"Will I pass this class?"
Student → Pass?

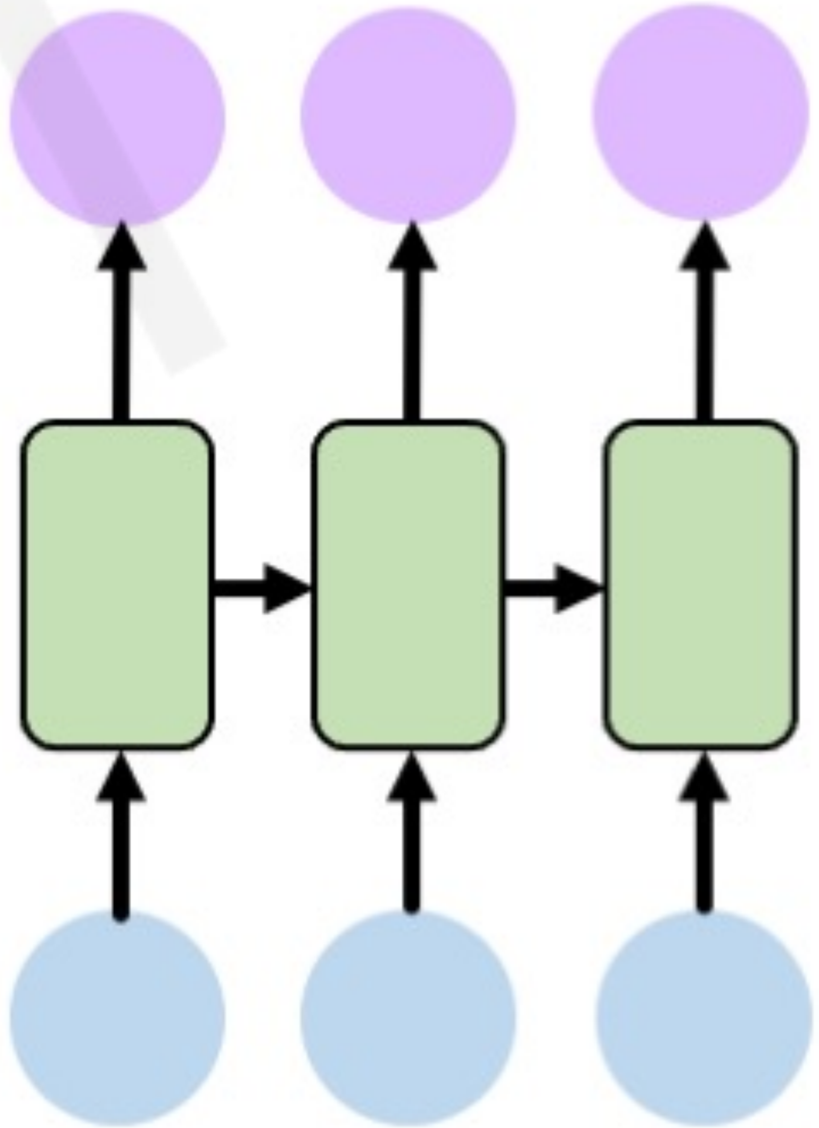


Many to One
Sentiment Classification



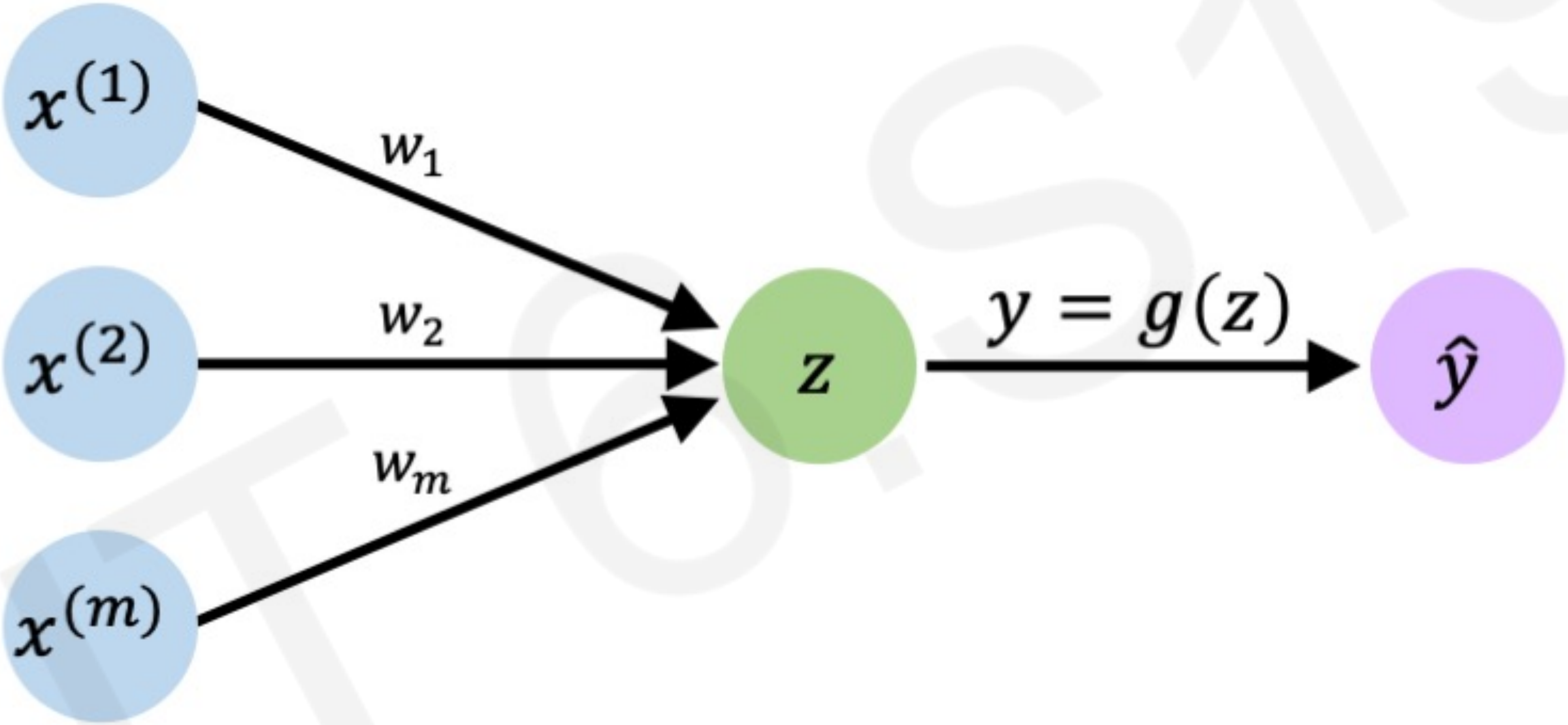
One to Many
Image Captioning

"A baseball player throws a ball."

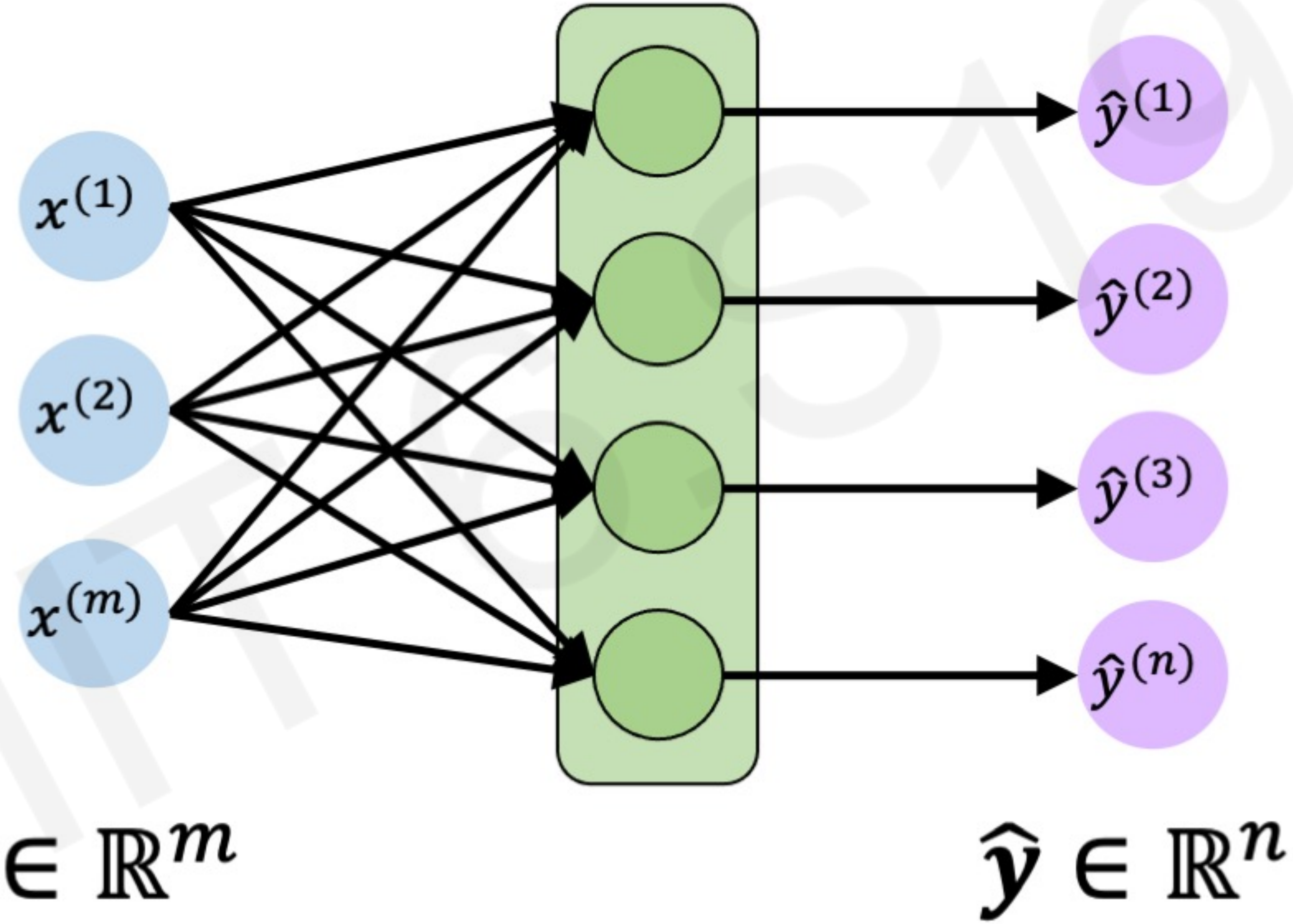


Many to Many
Machine Translation

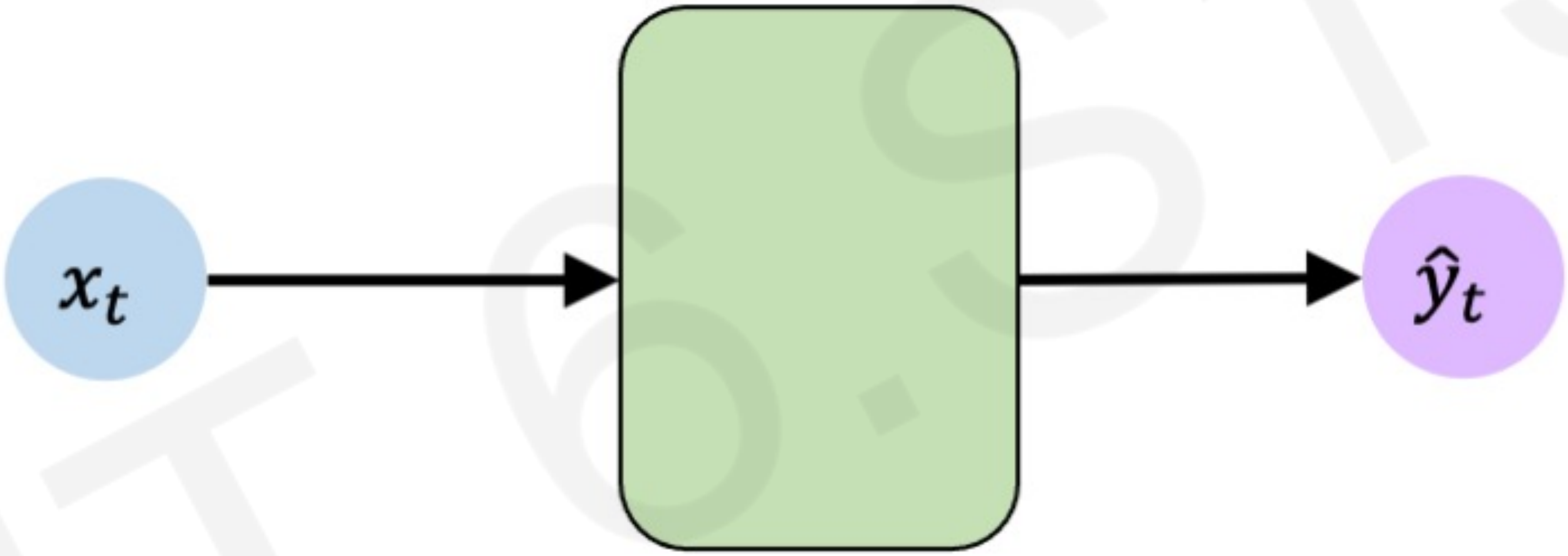
The Perceptron Revisited



Feed-Forward Networks Revisited

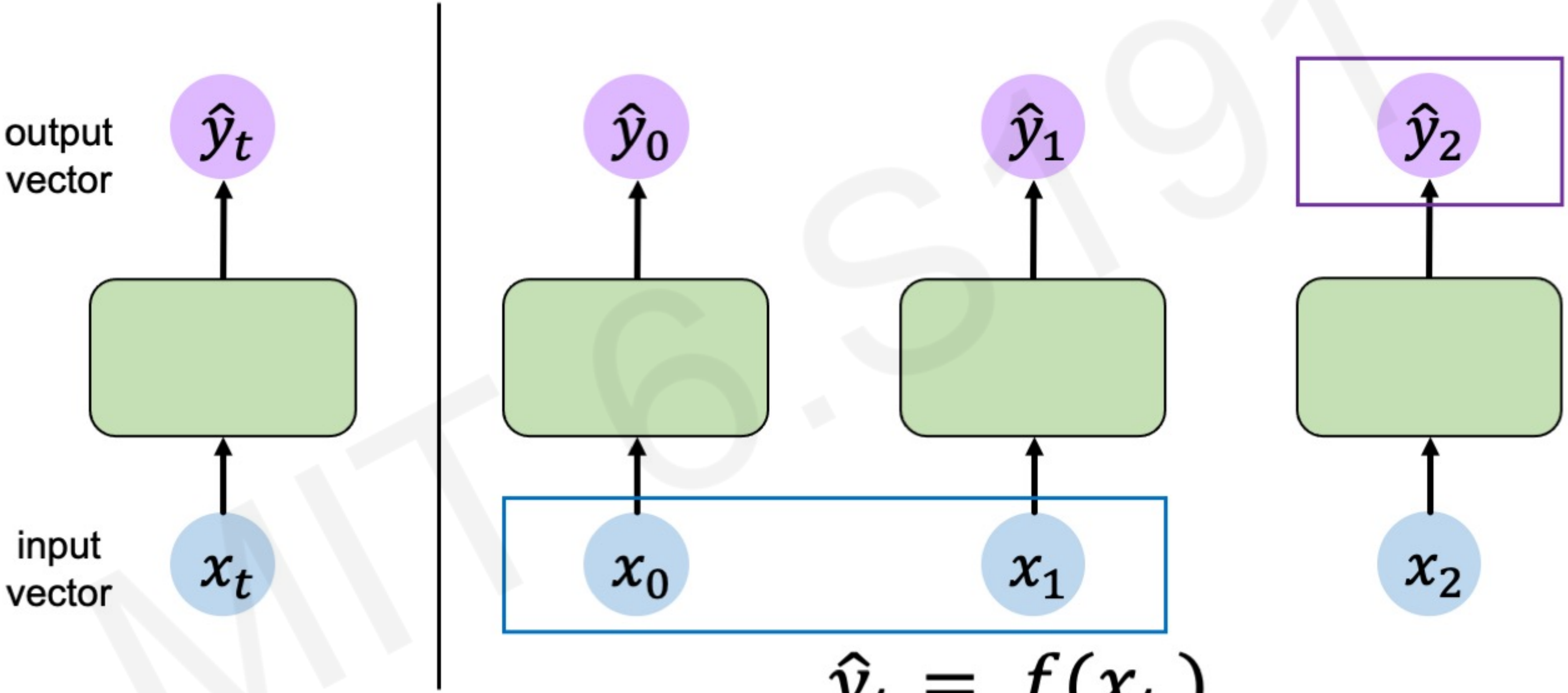


Feed-Forward Networks Revisited



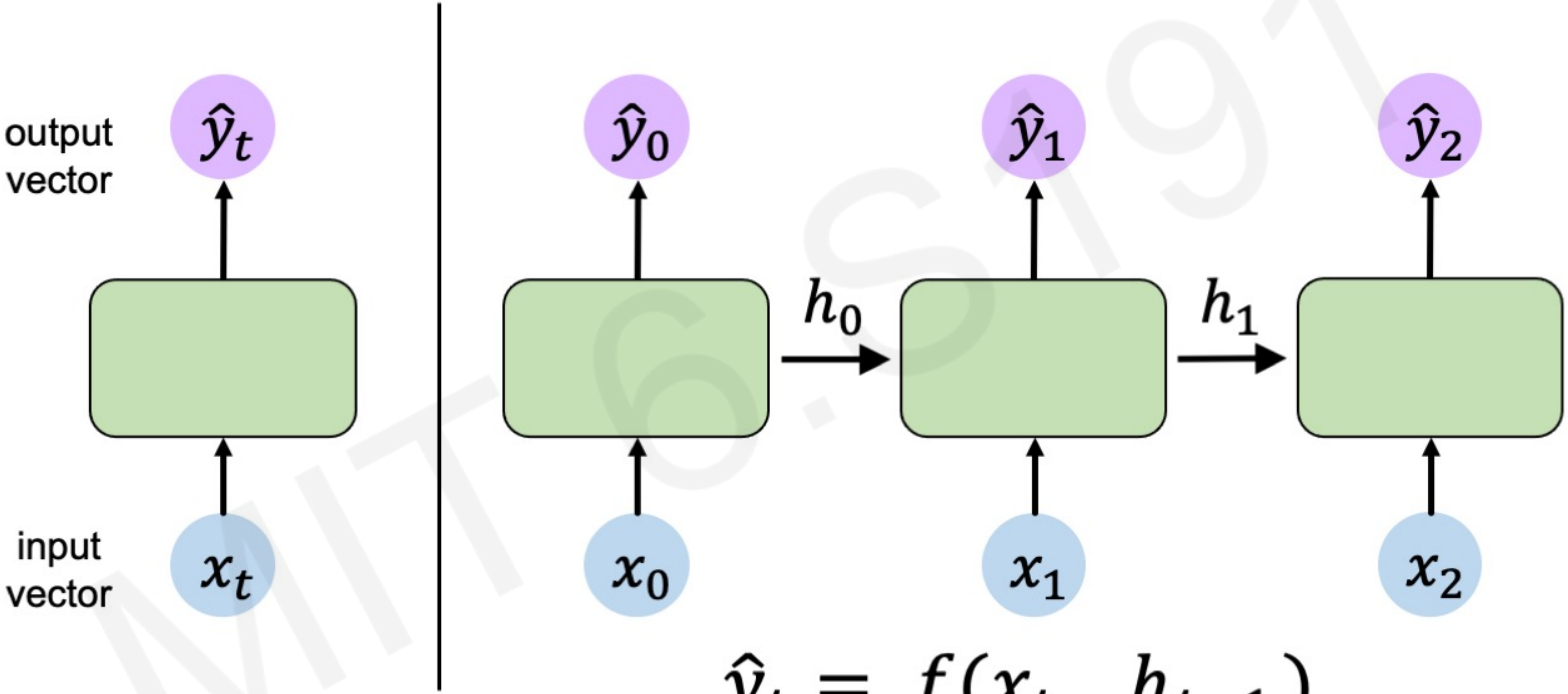
$\boldsymbol{x}_t \in \mathbb{R}^m$ $\qquad$ $\hat{\boldsymbol{y}}_t \in \mathbb{R}^n$

Handling Individual Time Steps



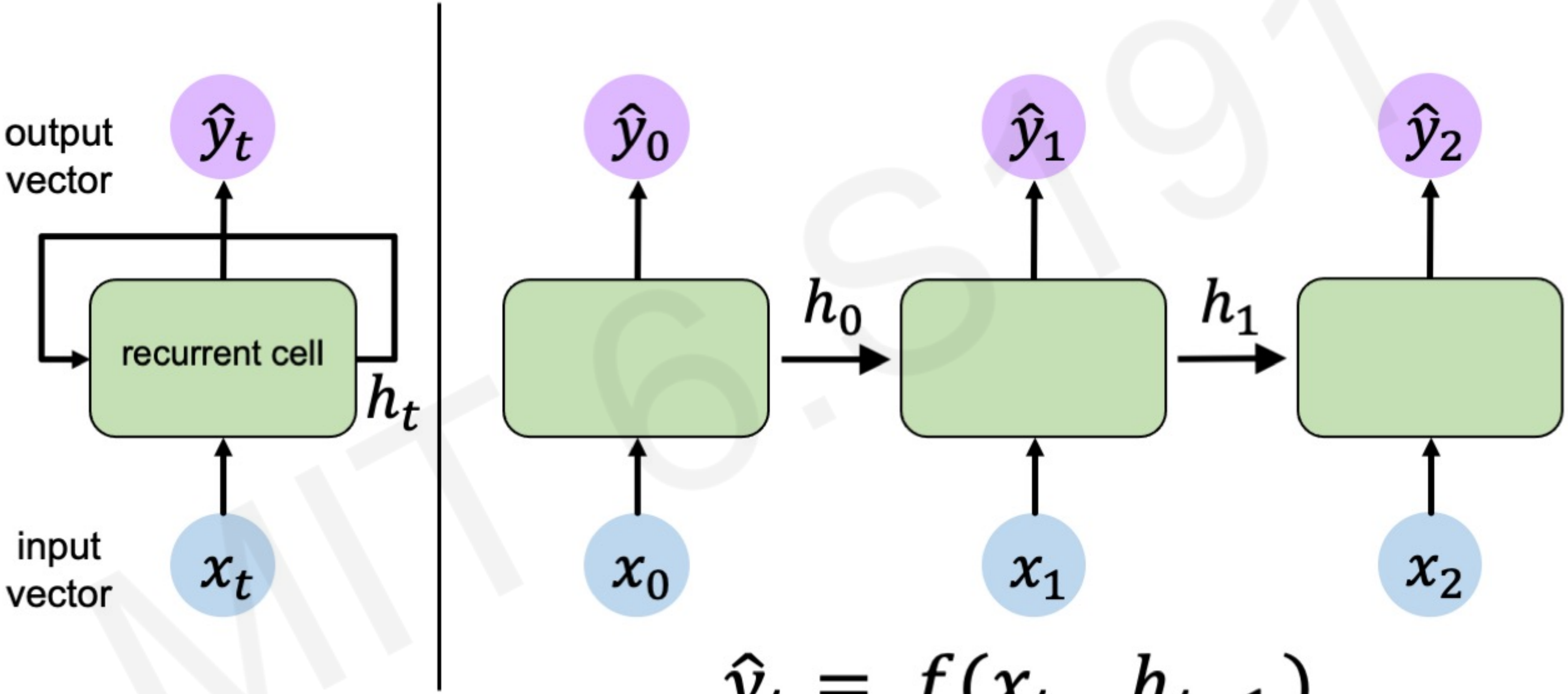
$$\hat{y}_t = f(x_t)$$

Neurons with Recurrence



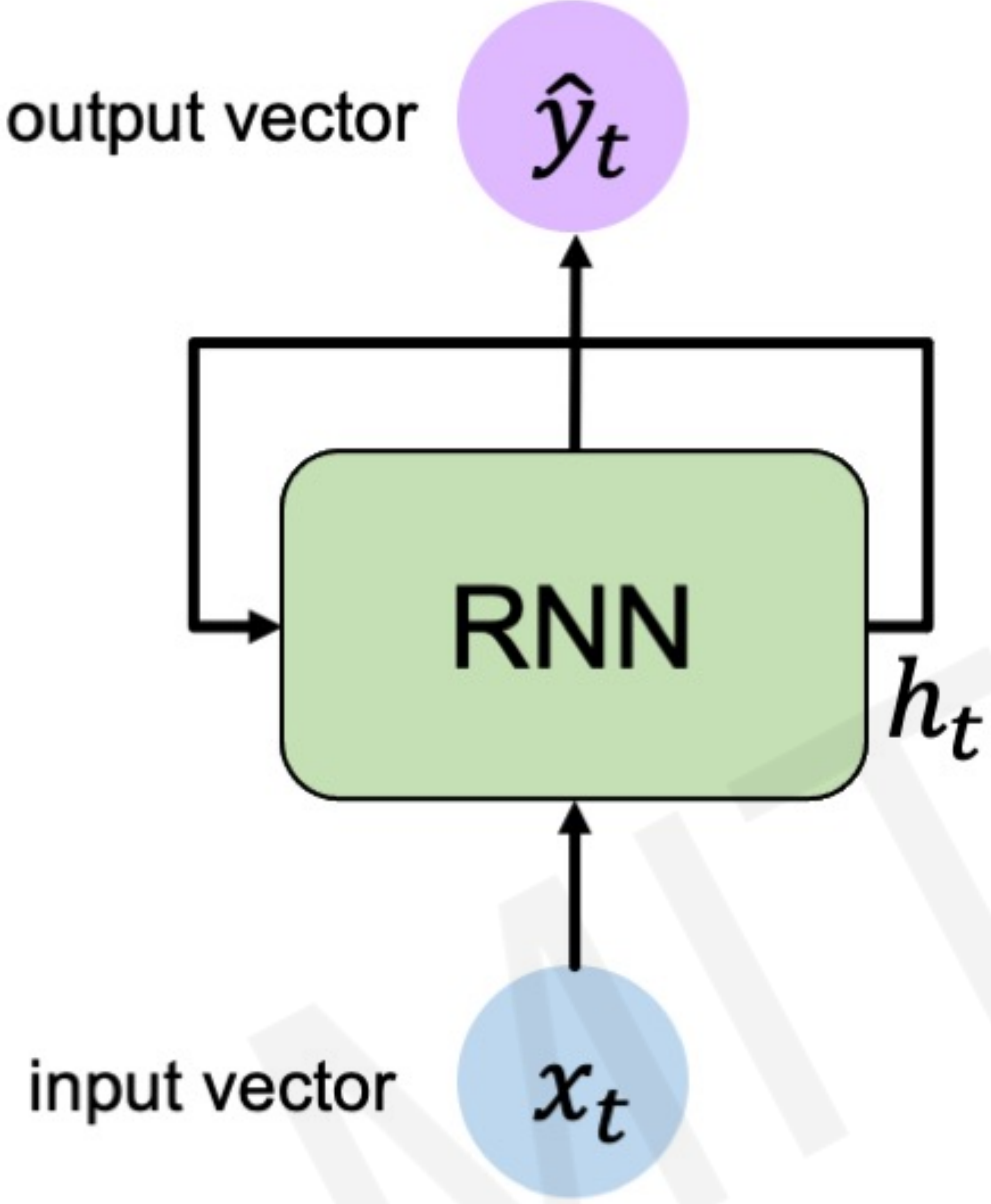
$$\underbrace{\hat{y}_t}_{\text{output}} = f(\underbrace{x_t}_{\text{input}}, \underbrace{h_{t-1}}_{\text{past memory}})$$

Neurons with Recurrence



$$\underbrace{\hat{y}_t}_{\text{output}} = f(\underbrace{x_t}_{\text{input}},\ \underbrace{h_{t-1}}_{\text{past memory}})$$

Recurrent Neural Networks (RNNs)



Apply a **recurrence relation** at every time step to process a sequence:

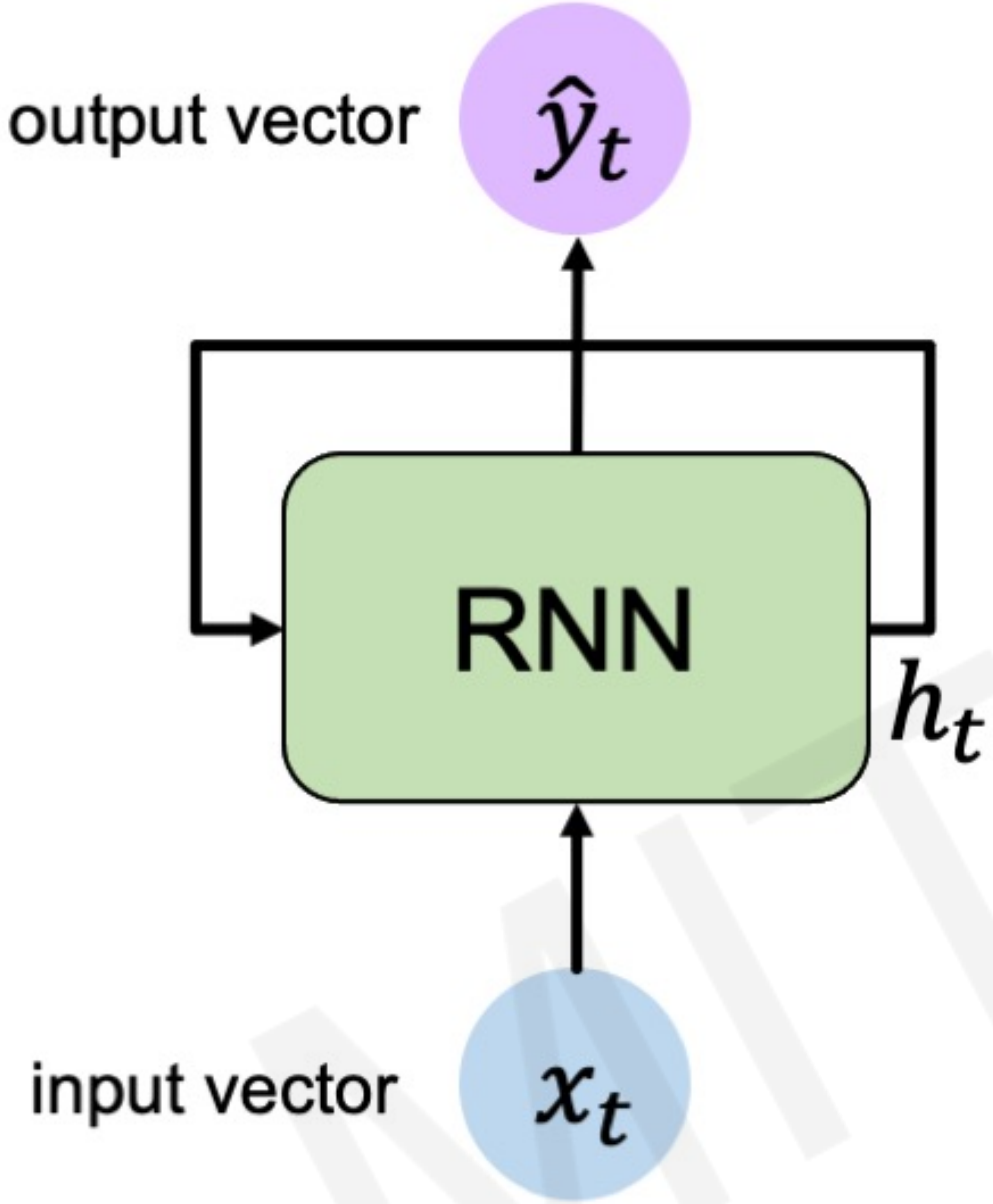
$$\boxed{h_t} = \boxed{f_W}(\boxed{x_t}, \boxed{h_{t-1}})$$

cell state — function with weights W — input — old state

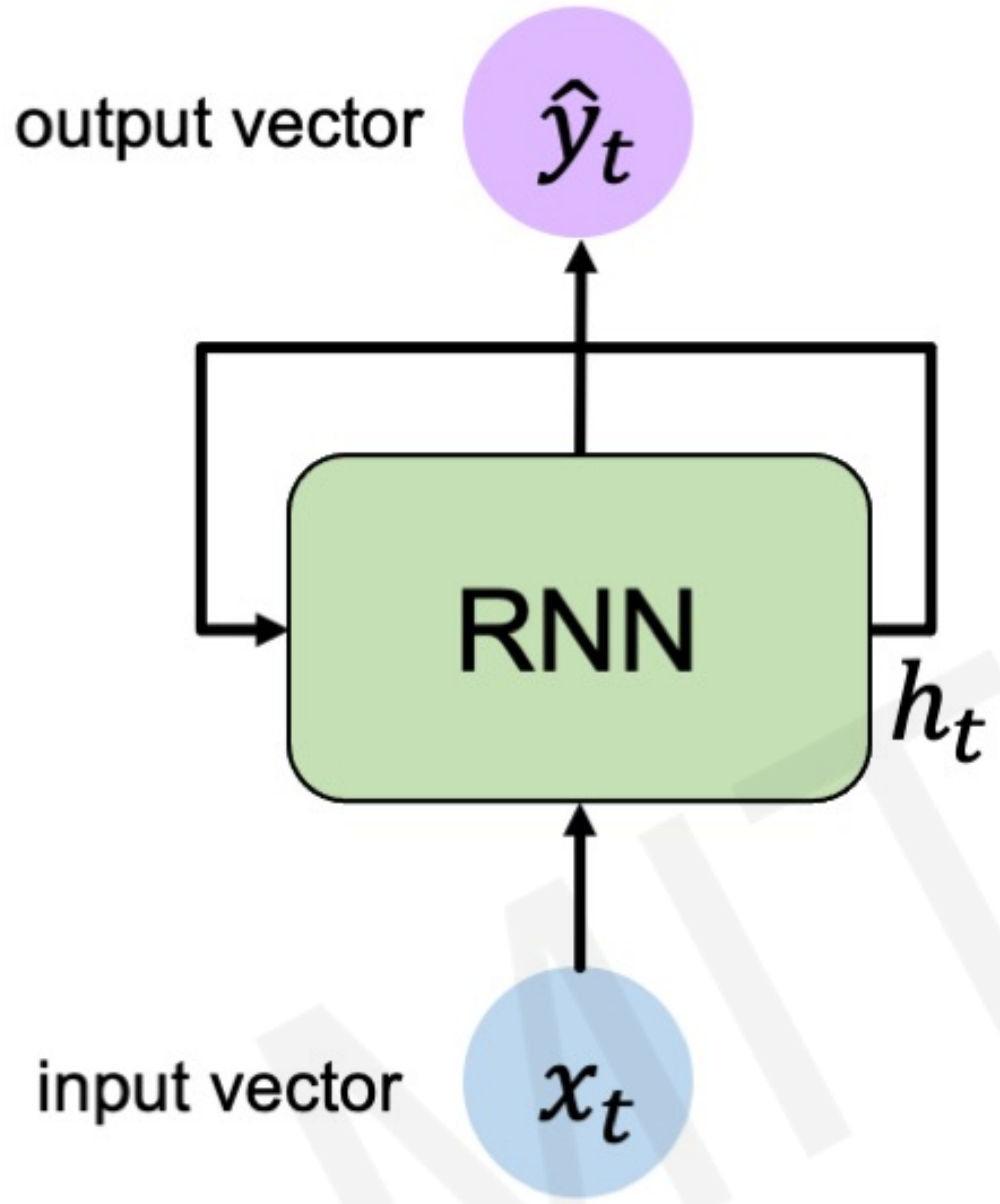
Note: the same function and set of parameters are used at every time step

RNNs have a **state**, h_t, that is updated **at each time step** as a sequence is processed

RNN State Update and Output



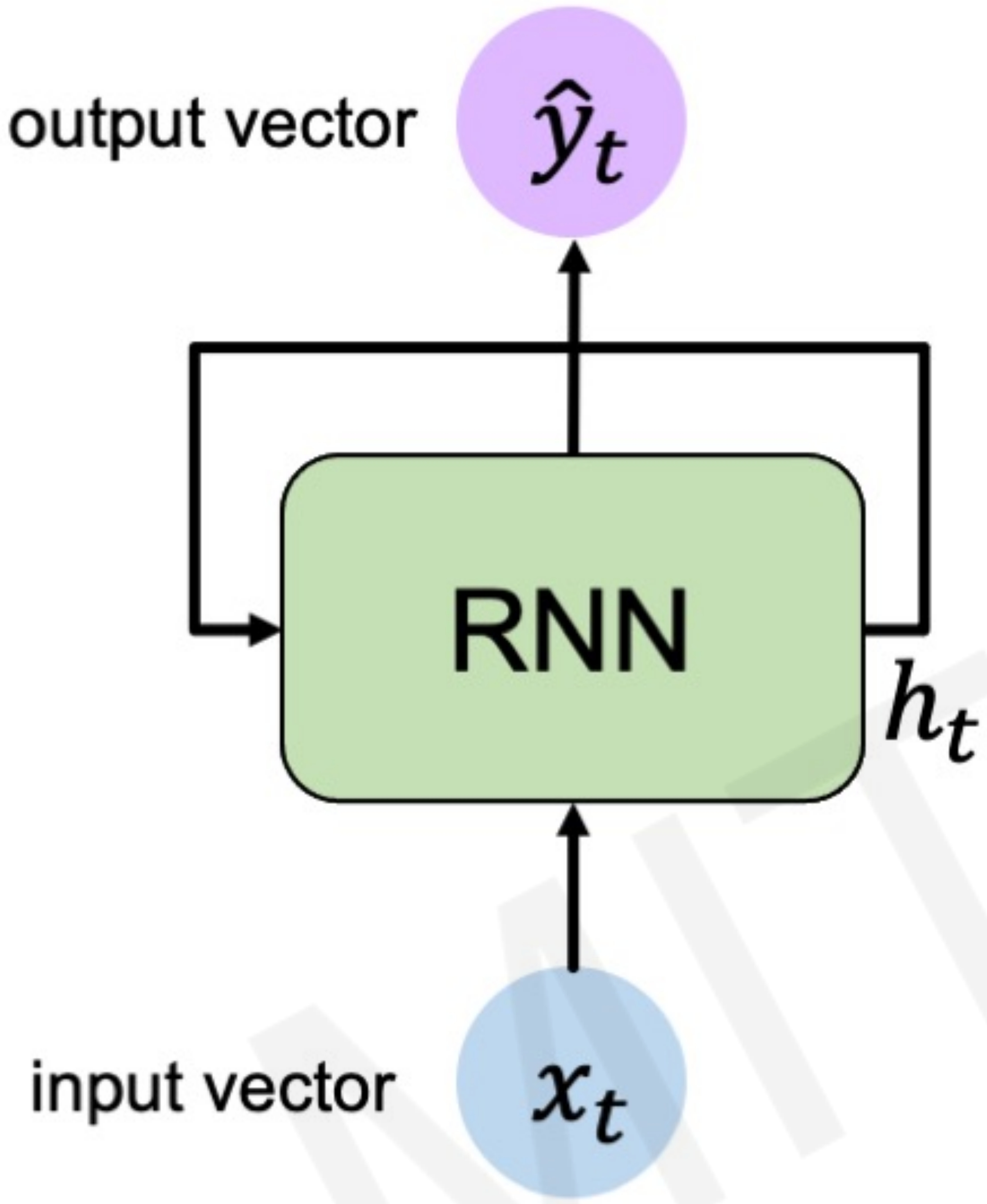
RNN State Update and Output



Input Vector

x_t

RNN State Update and Output



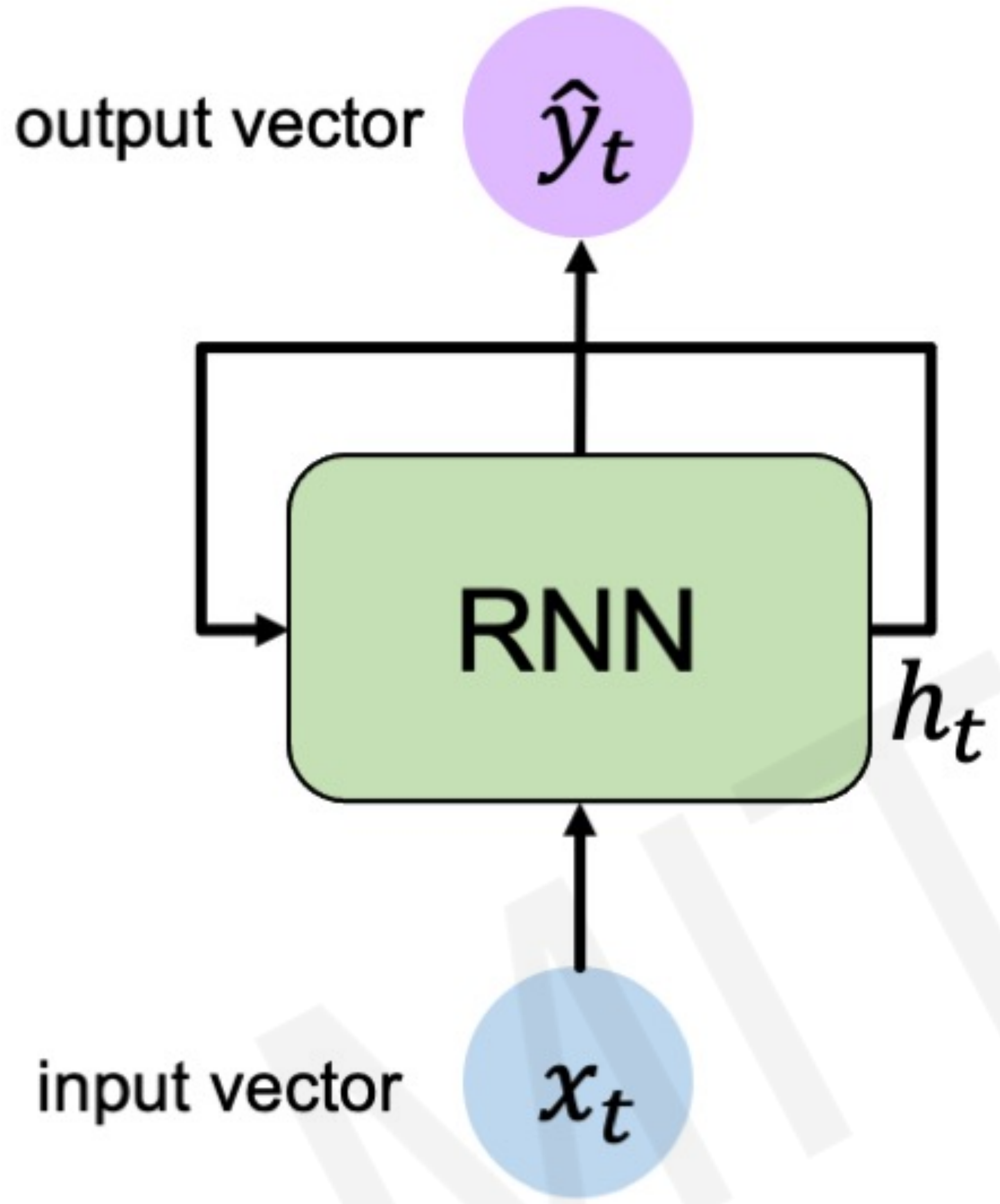
Update Hidden State

$$h_t = \tanh(\boldsymbol{W}_{hh}^T h_{t-1} + \boldsymbol{W}_{xh}^T x_t)$$

Input Vector

$$x_t$$

RNN State Update and Output



Output Vector

$$\hat{y}_t = \boldsymbol{W}_{\boldsymbol{hy}}^{\boldsymbol{T}} h_t$$

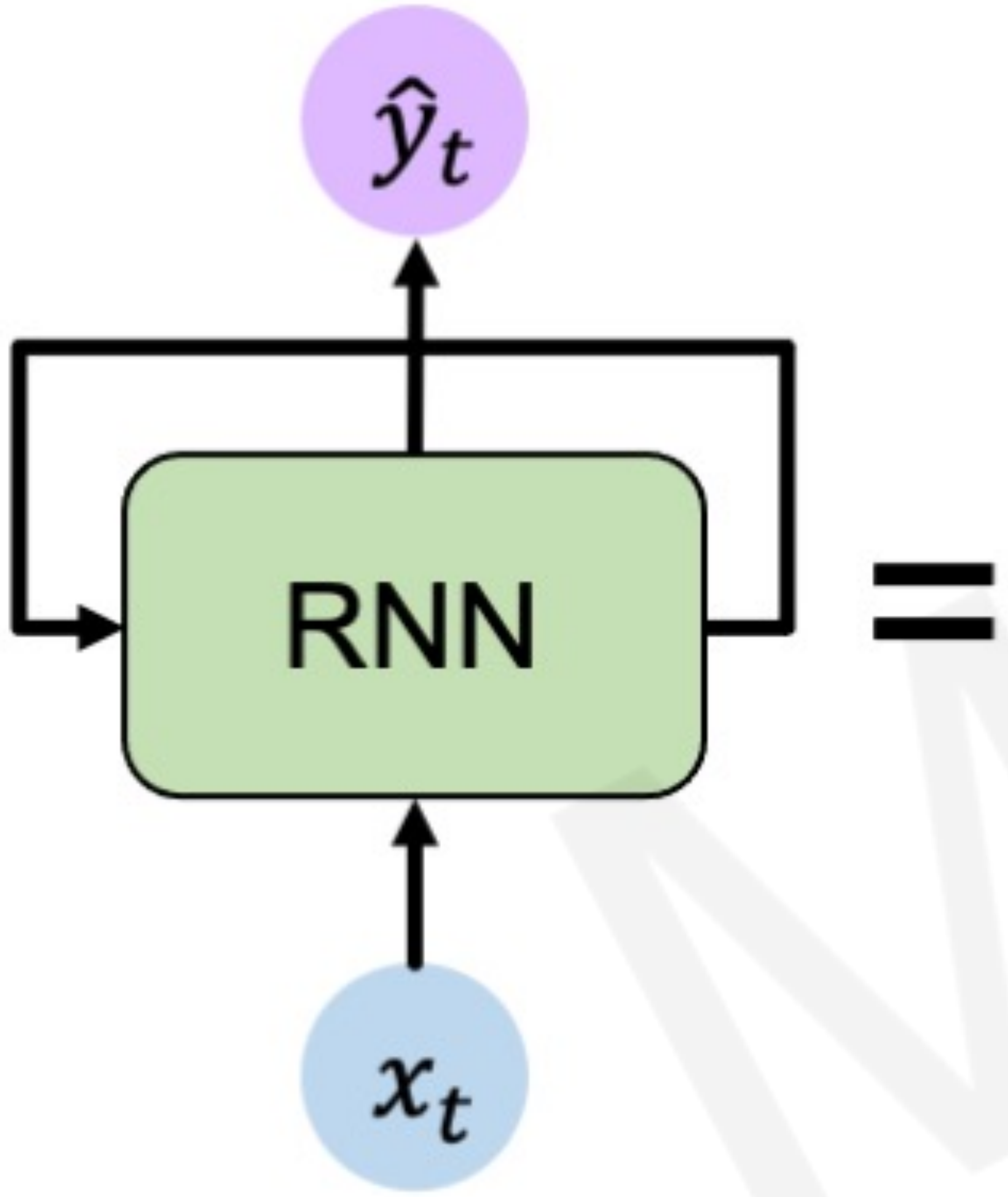
Update Hidden State

$$h_t = \tanh(\boldsymbol{W}_{\boldsymbol{hh}}^{\boldsymbol{T}} h_{t-1} + \boldsymbol{W}_{\boldsymbol{xh}}^{\boldsymbol{T}} x_t)$$

Input Vector

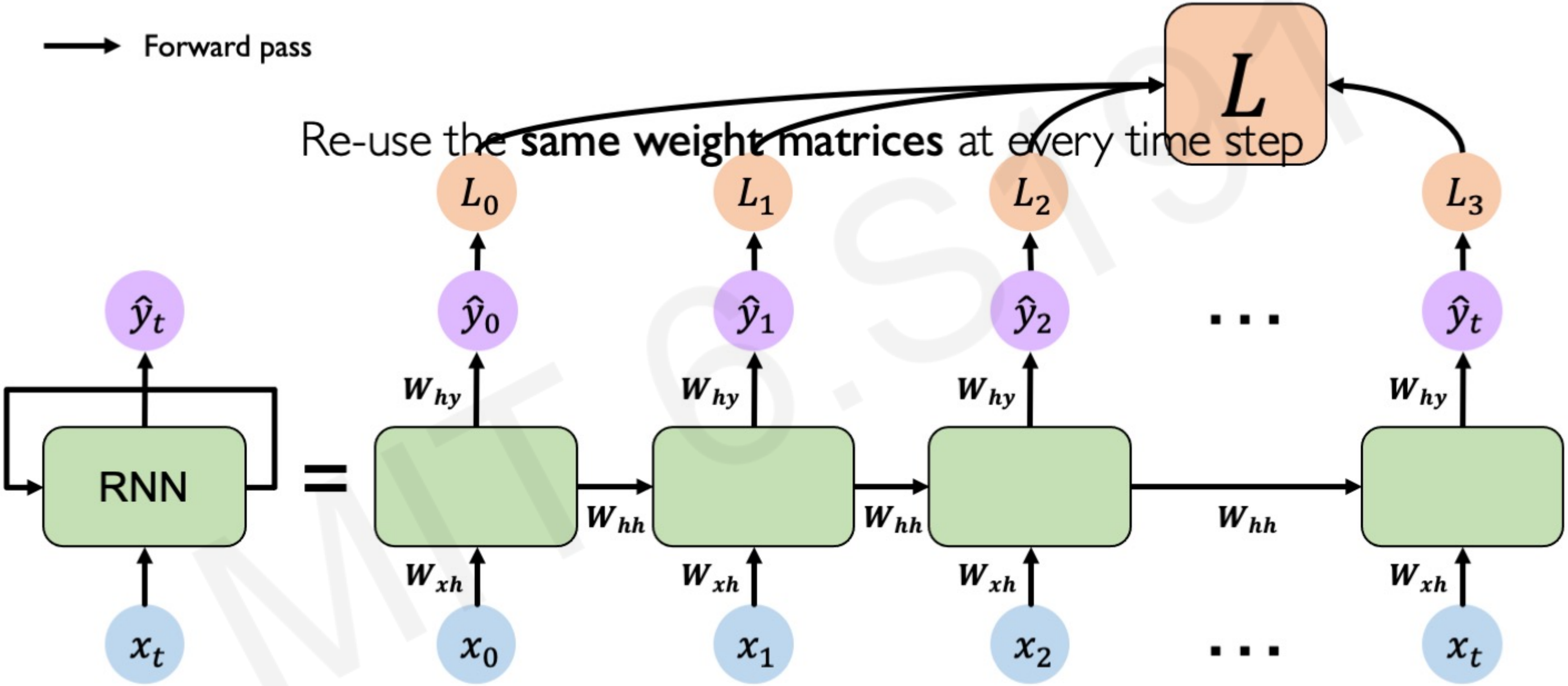
$$x_t$$

RNNs: Computational Graph Across Time



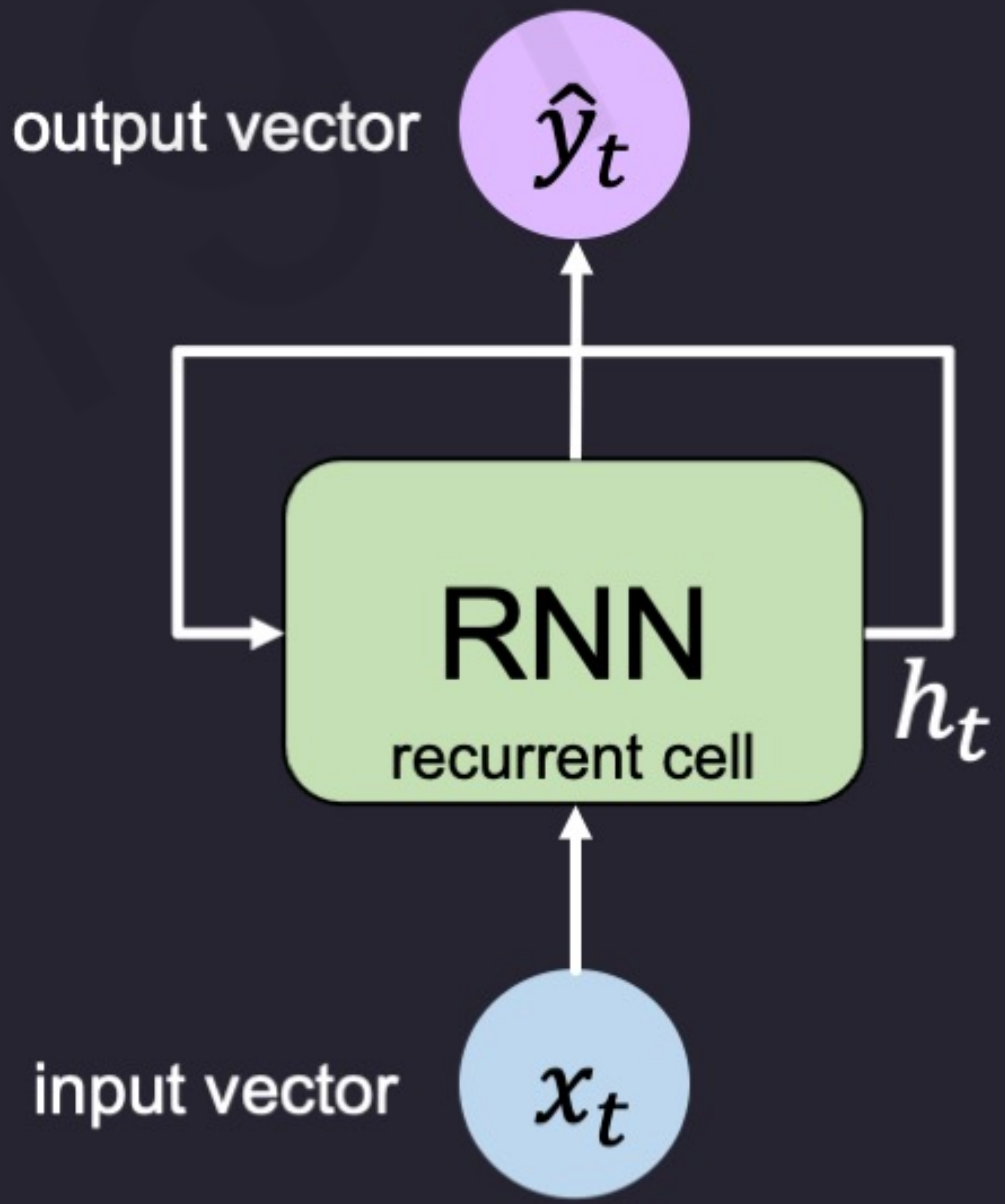
Represent as computational graph unrolled across time

RNNs: Computational Graph Across Time

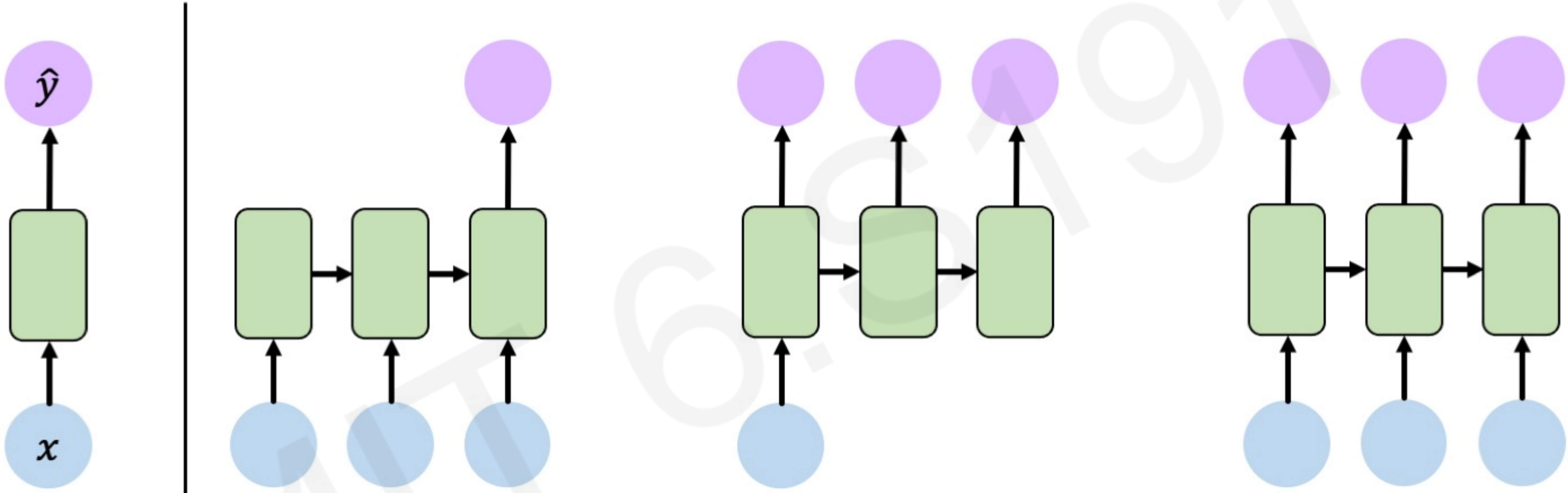


RNN Implementation in TensorFlow

```
tf.keras.layers.SimpleRNN(rnn_units)
```



RNNs for Sequence Modeling



One to One
"Vanilla" NN
Binary classification

Many to One
Sentiment Classification

One to Many
Text Generation
Image Captioning

Many to Many
Translation & Forecasting
Music Generation

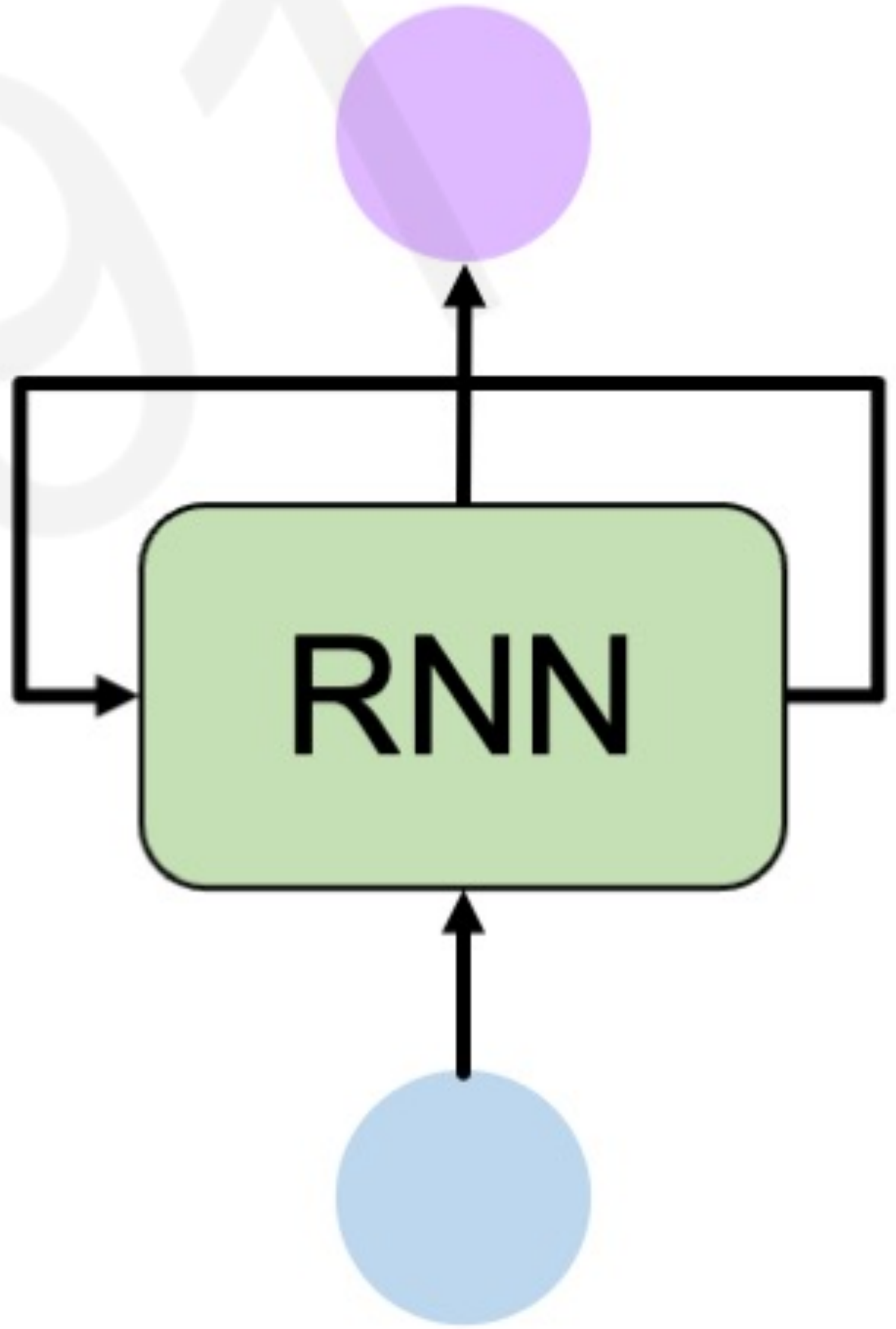
… and many other architectures and applications

6.S191 Lab!

Sequence Modeling: Design Criteria

To model sequences, we need to:

1. Handle **variable-length** sequences
2. Track **long-term** dependencies
3. Maintain information about **order**
4. **Share parameters** across the sequence



Recurrent Neural Networks (RNNs) meet these sequence modeling design criteria

A Sequence Modeling Problem: Predict the Next Word

"This morning I took my cat for a walk."

A Sequence Modeling Problem: Predict the Next Word

"This morning I took my cat for a walk."

given these words

A Sequence Modeling Problem: Predict the Next Word

"This morning I took my cat for a walk."

given these words

predict the next word

A Sequence Modeling Problem: Predict the Next Word

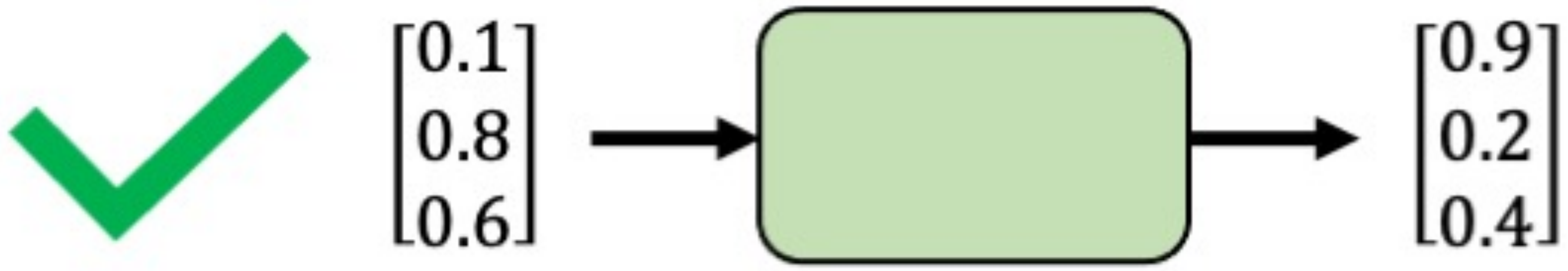
"This morning I took my cat for a walk."

given these words

predict the next word

Representing Language to a Neural Network

Neural networks cannot interpret words



Neural networks require numerical inputs

A Sequence Modeling Problem: Predict the Next Word

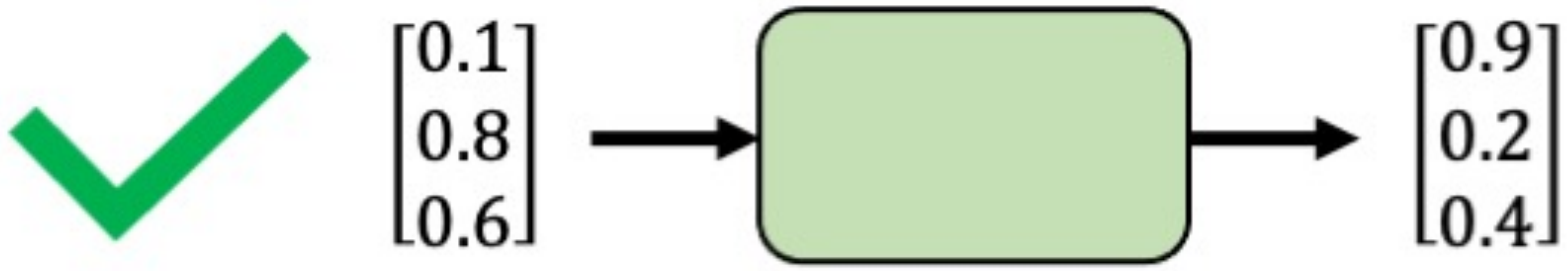
"This morning I took my cat for a walk."

given these words

predict the next word

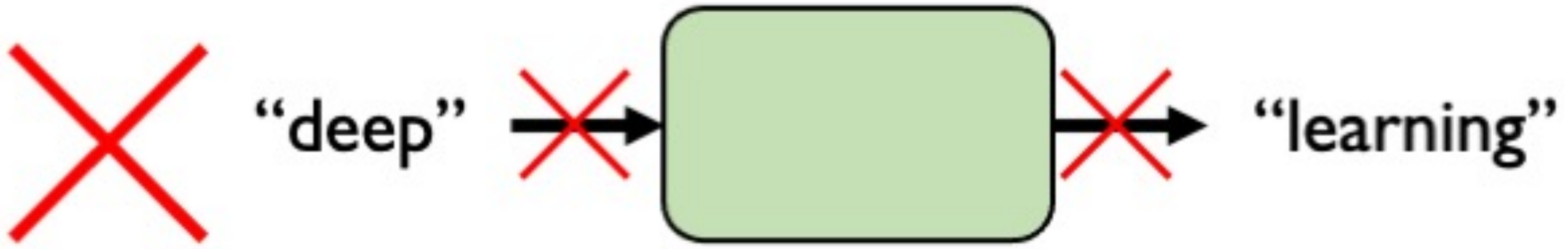
Representing Language to a Neural Network

Neural networks cannot interpret words

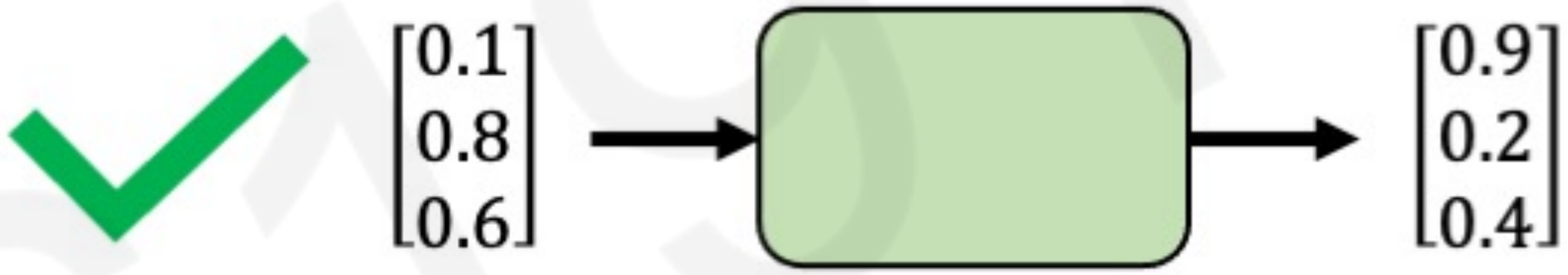


Neural networks require numerical inputs

Encoding Language for a Neural Network

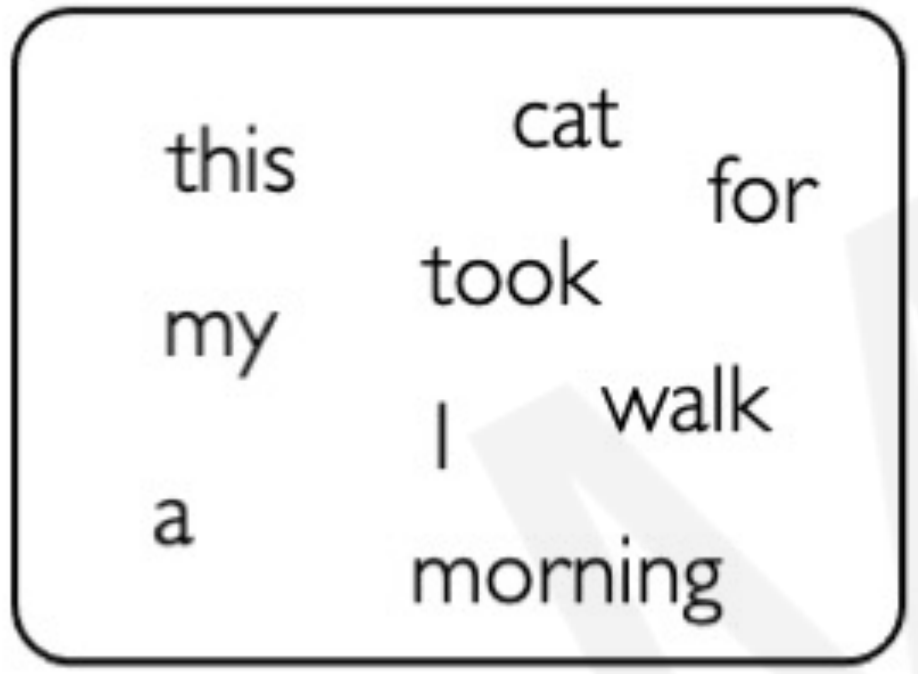


Neural networks cannot interpret words

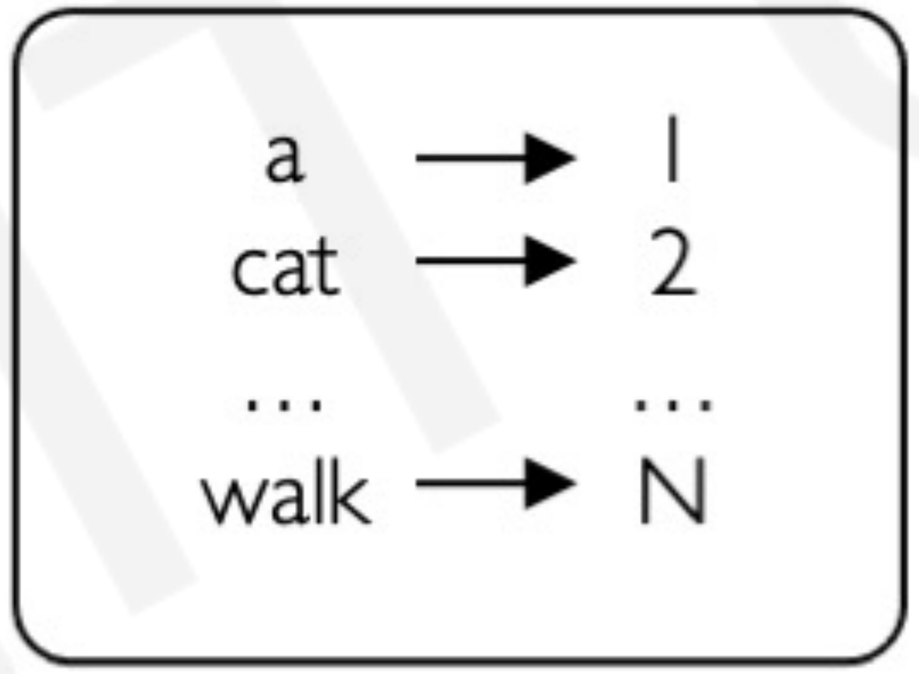


Neural networks require numerical inputs

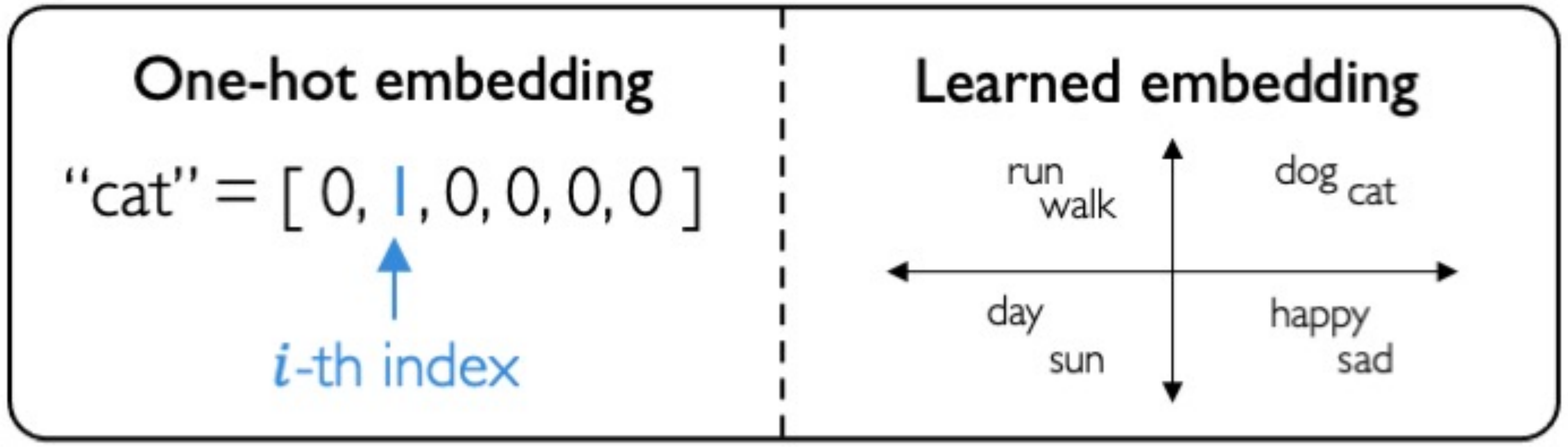
Embedding: transform indexes into a vector of fixed size.



1. Vocabulary:
Corpus of words



2. Indexing:
Word to index



3. Embedding:
Index to fixed-sized vector

Handle Variable Sequence Lengths

The food was <u>great</u>

vs.

We visited a restaurant for <u>lunch</u>

vs.

We were hungry but cleaned the house before <u>eating</u>

Model Long-Term Dependencies

"**France** is where I grew up, but I now live in Boston. I speak fluent ___."

J'aime 6.S191!

We need information from **the distant past** to accurately predict the correct word.

Capture Differences in Sequence Order

The food was good, not bad at all.

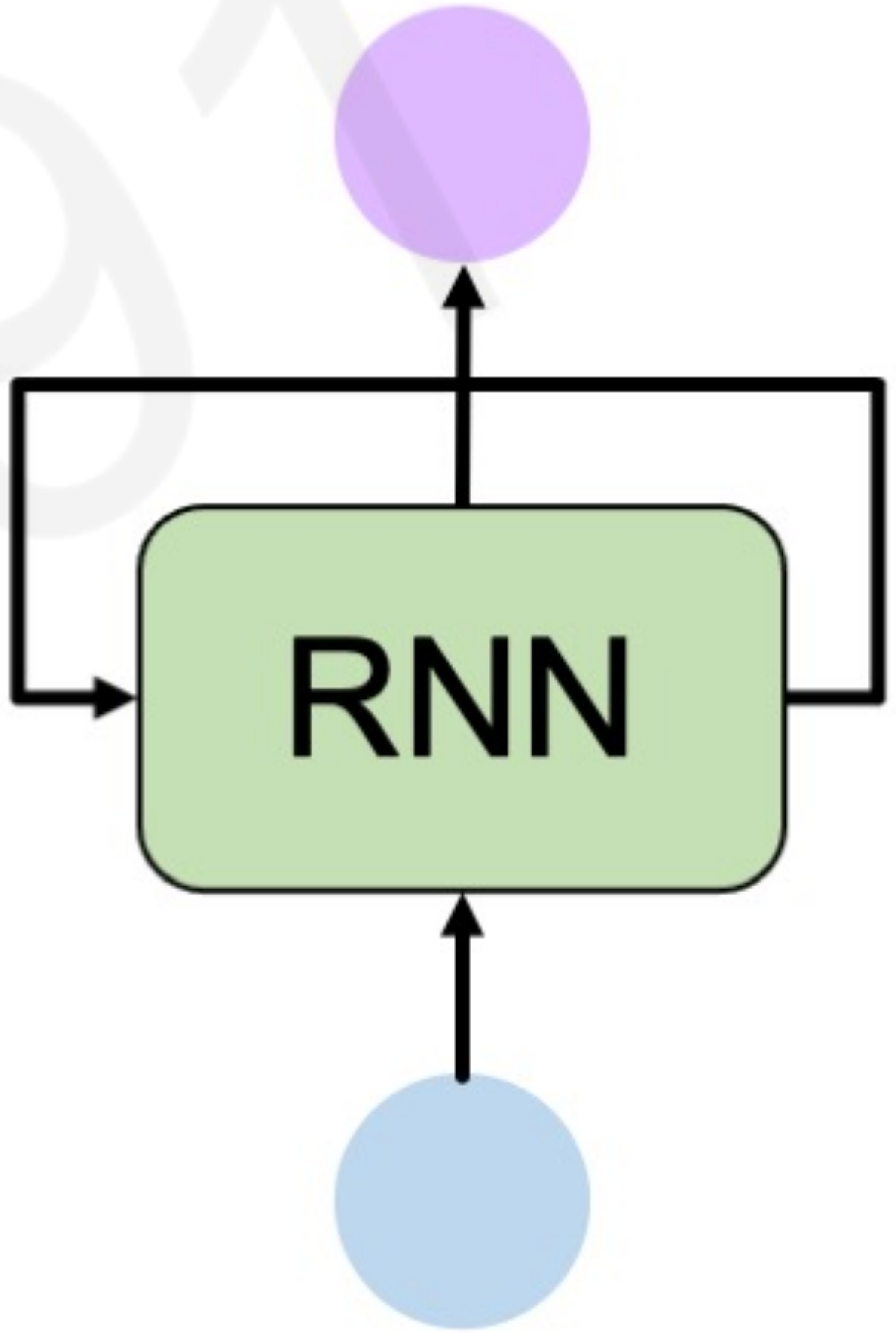
vs.

The food was bad, not good at all.

Sequence Modeling: Design Criteria

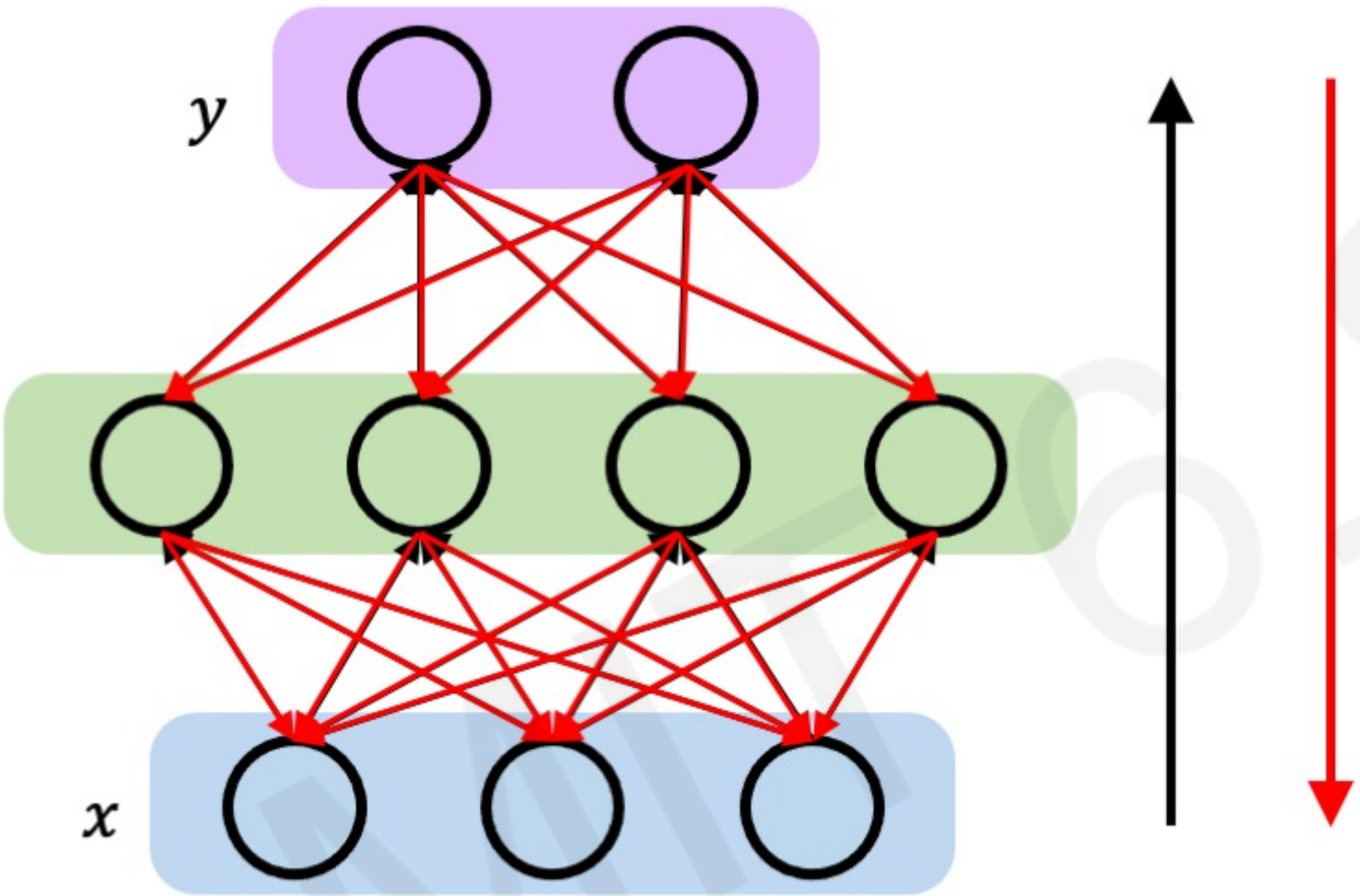
To model sequences, we need to:

1. Handle **variable-length** sequences
2. Track **long-term** dependencies
3. Maintain information about **order**
4. **Share parameters** across the sequence



Recurrent Neural Networks (RNNs) meet these sequence modeling design criteria

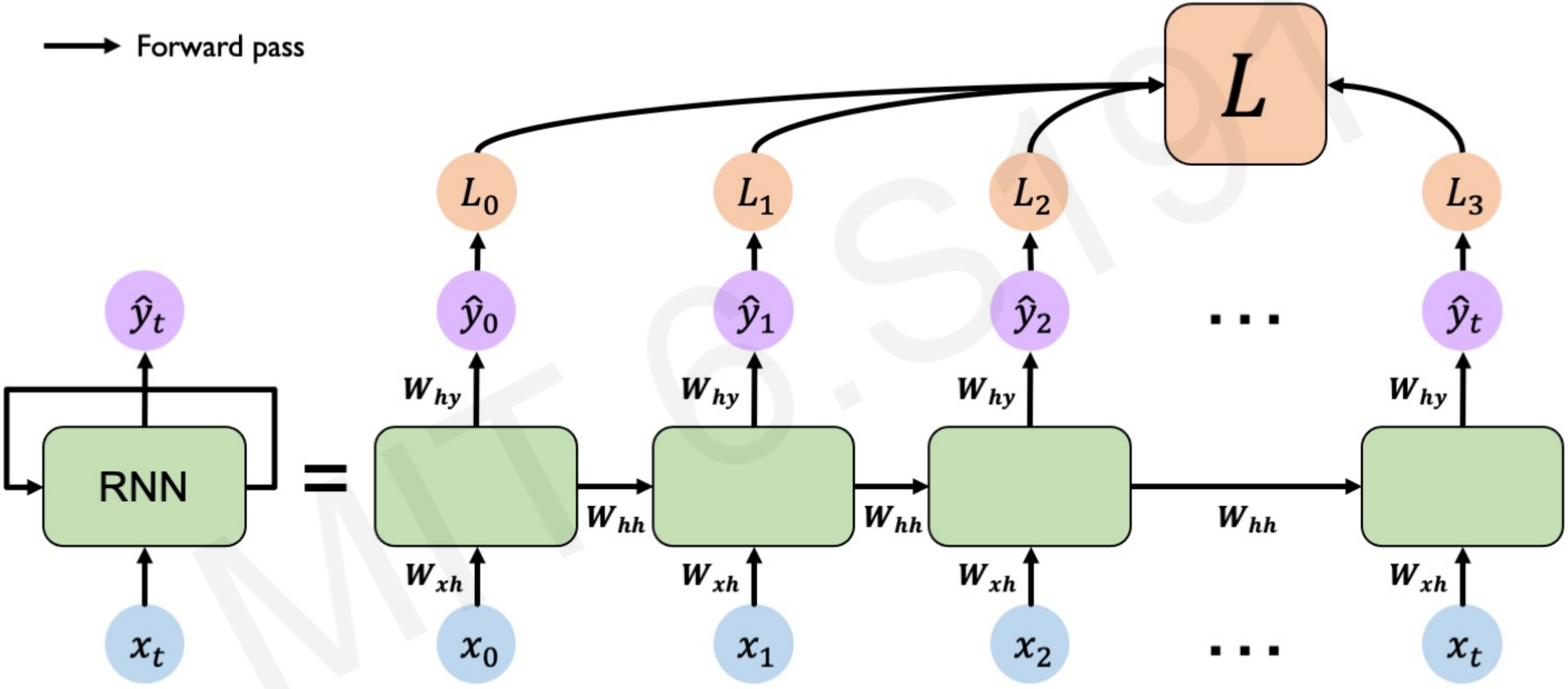
Recall: Backpropagation in Feed Forward Models



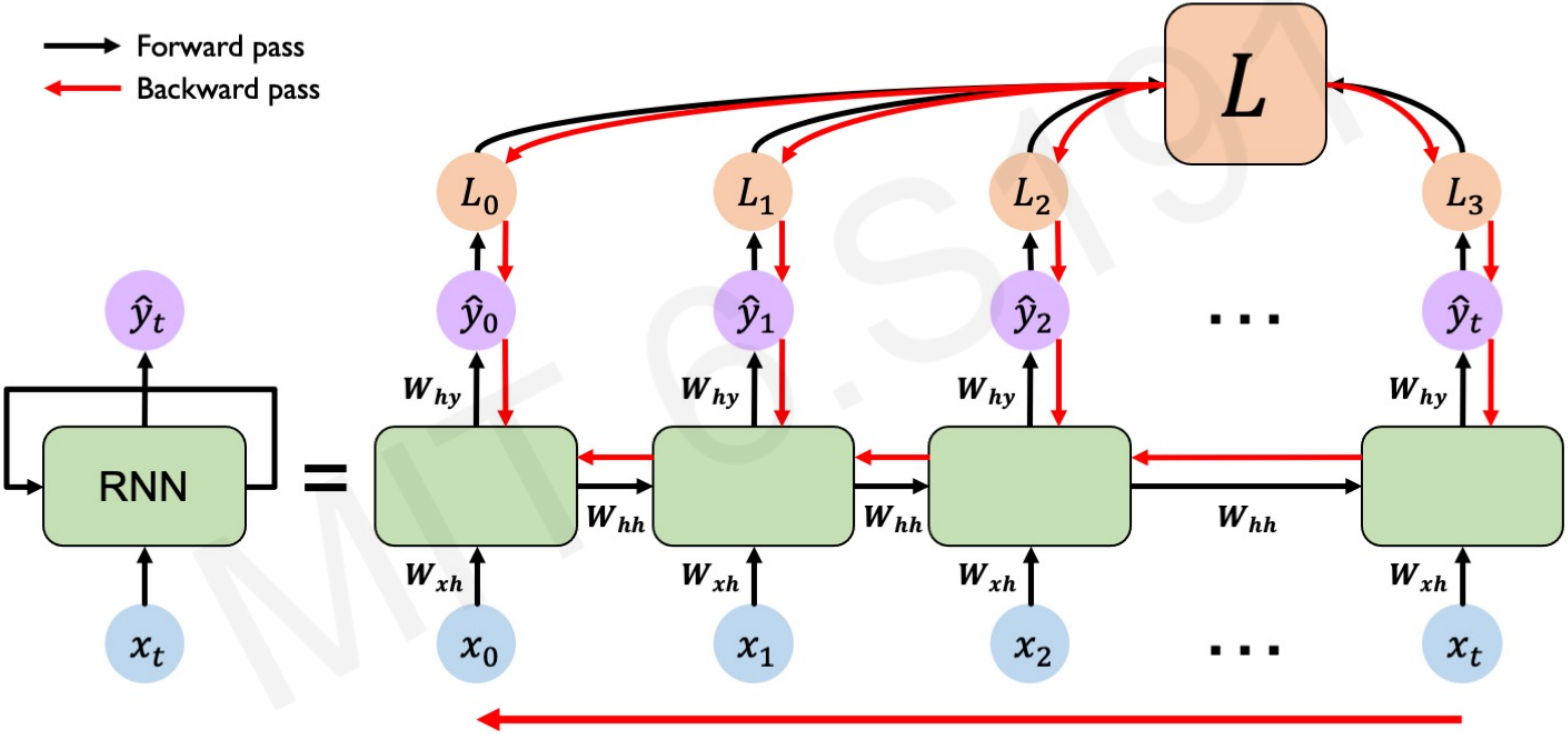
Backpropagation algorithm:

1. Take the derivative (gradient) of the loss with respect to each parameter
2. Shift parameters in order to minimize loss

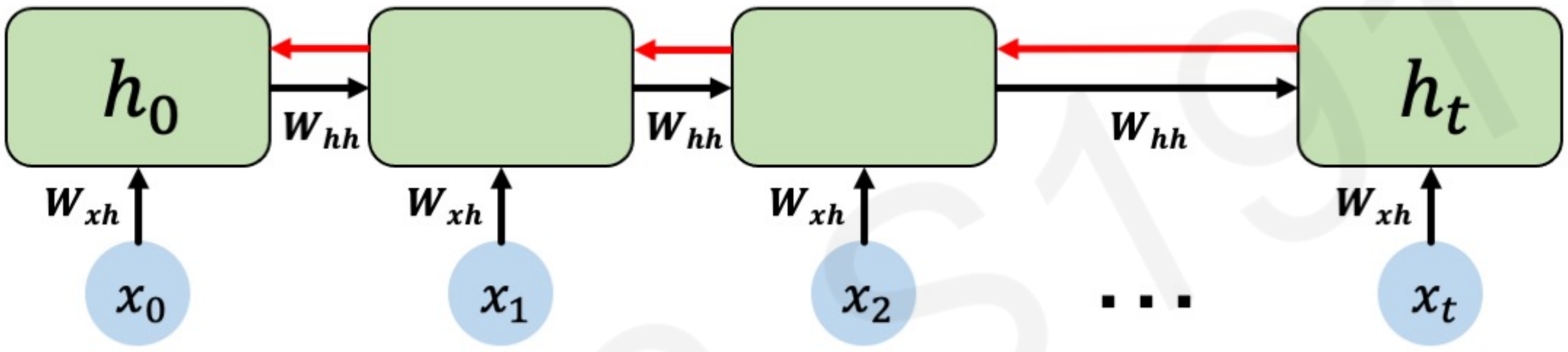
RNNs: Backpropagation Through Time



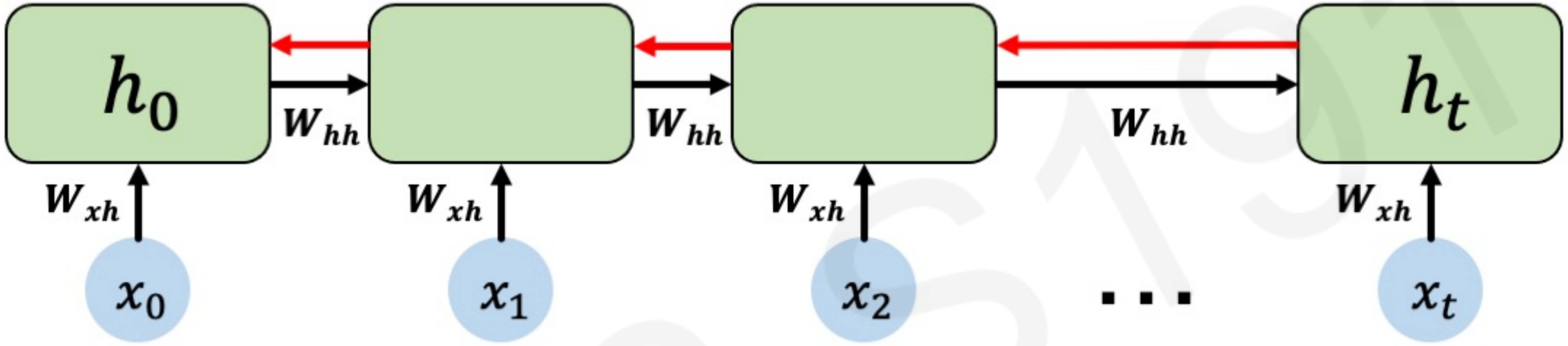
RNNs: Backpropagation Through Time



Standard RNN Gradient Flow

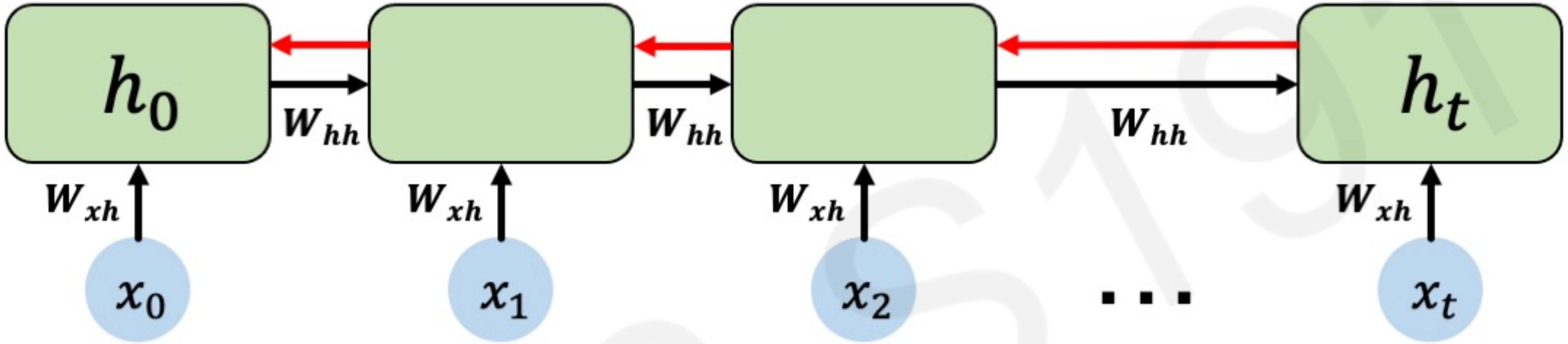


Standard RNN Gradient Flow



Computing the gradient wrt h_0 involves **many factors of** W_{hh} + **repeated gradient computation**!

Standard RNN Gradient Flow: Exploding Gradients

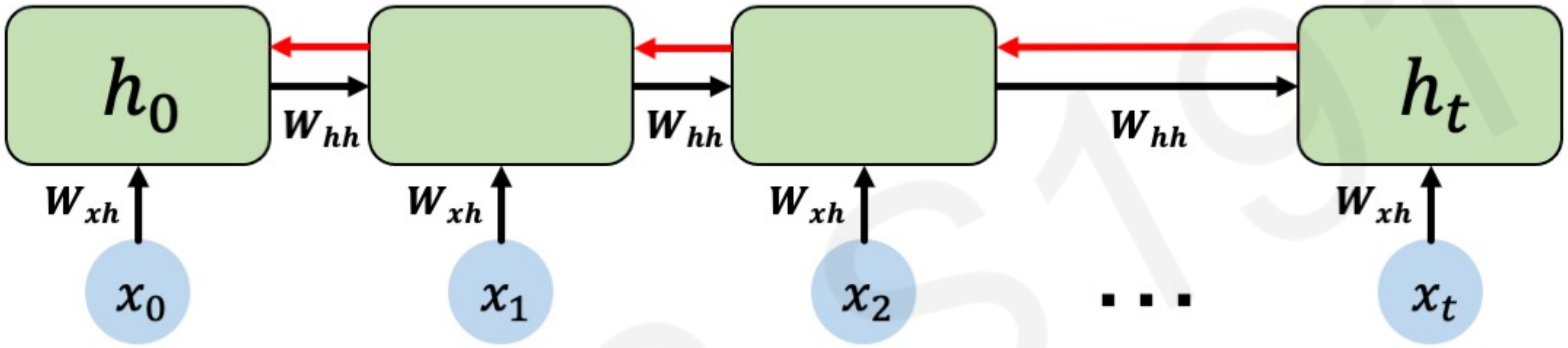


Computing the gradient wrt h_0 involves **many factors of W_{hh} + repeated gradient computation**!

Many values > 1:
exploding gradients

Gradient clipping to scale big gradients

Standard RNN Gradient Flow: Vanishing Gradients



Computing the gradient wrt h_0 involves **many factors of W_{hh} + repeated gradient computation**!

Many values > 1:
exploding gradients

Gradient clipping to scale big gradients

Many values < 1:
vanishing gradients

1. **Activation function**
2. **Weight initialization**
3. **Network architecture**

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

↓

Errors due to further back time steps have smaller and smaller gradients

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

↓

Errors due to further back time steps have smaller and smaller gradients

↓

Bias parameters to capture short-term dependencies

The Problem of Long-Term Dependencies

"The clouds are in the ___"

Why are vanishing gradients a problem?

Multiply many **small numbers** together

↓

Errors due to further back time steps have smaller and smaller gradients

↓

Bias parameters to capture short-term dependencies

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

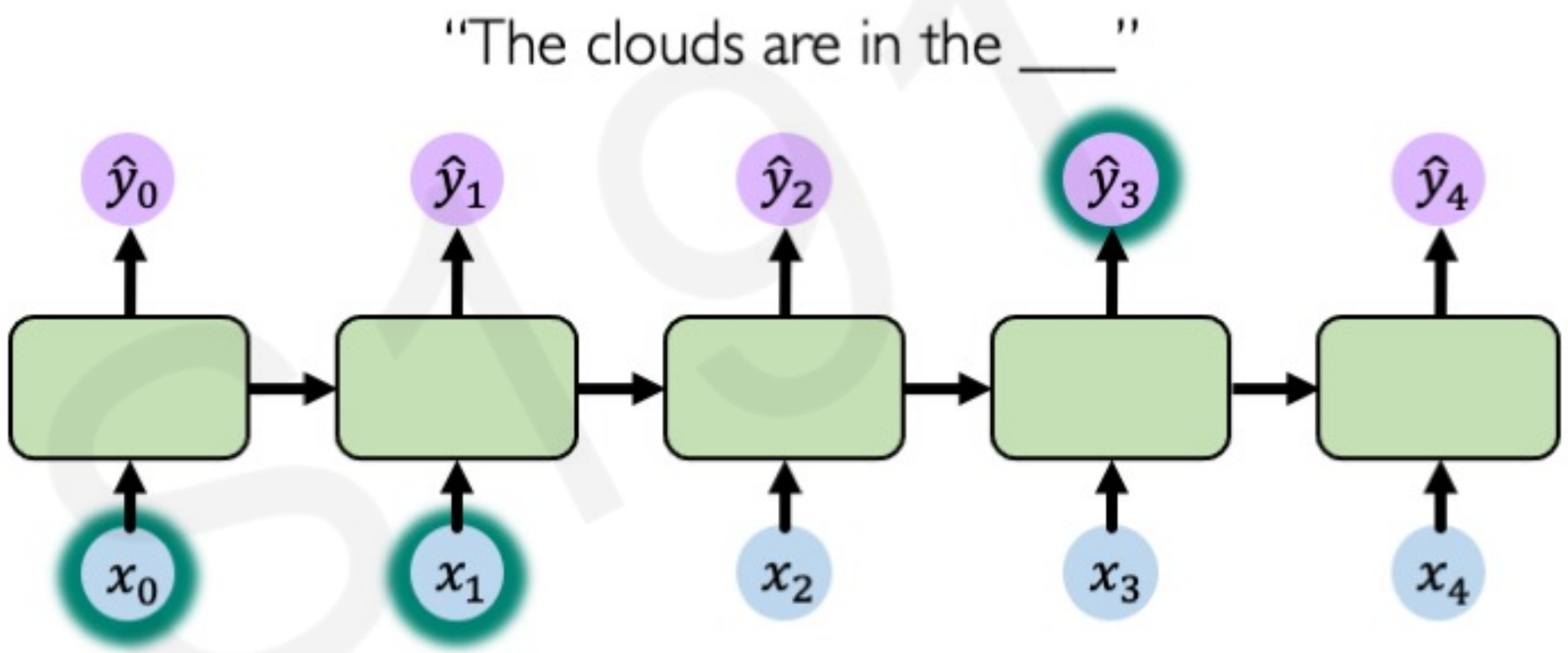
Multiply many **small numbers** together

↓

Errors due to further back time steps have smaller and smaller gradients

↓

Bias parameters to capture short-term dependencies



The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

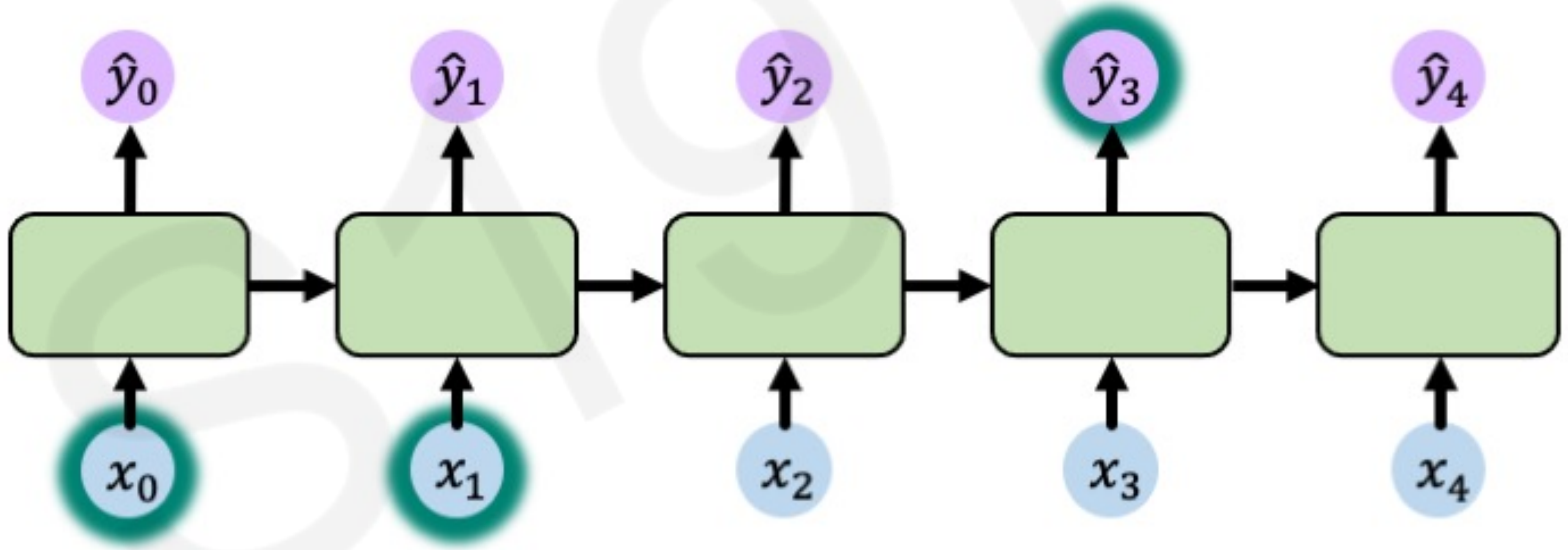
↓

Errors due to further back time steps have smaller and smaller gradients

↓

Bias parameters to capture short-term dependencies

"The clouds are in the ___"



"I grew up in France, … and I speak fluent___ "

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

↓

Errors due to further back time steps have smaller and smaller gradients

↓

Bias parameters to capture short-term dependencies