# Modern Deep NLP - Session 1

Instructor: Dr. Ehsan Amjadian

Teaching Staff: Preston Engstrom, Dr. Florian Goebels, Werner Chao, Masoud Hoveidar

27 June 2019

aisc

# License



aisc

# Logistics

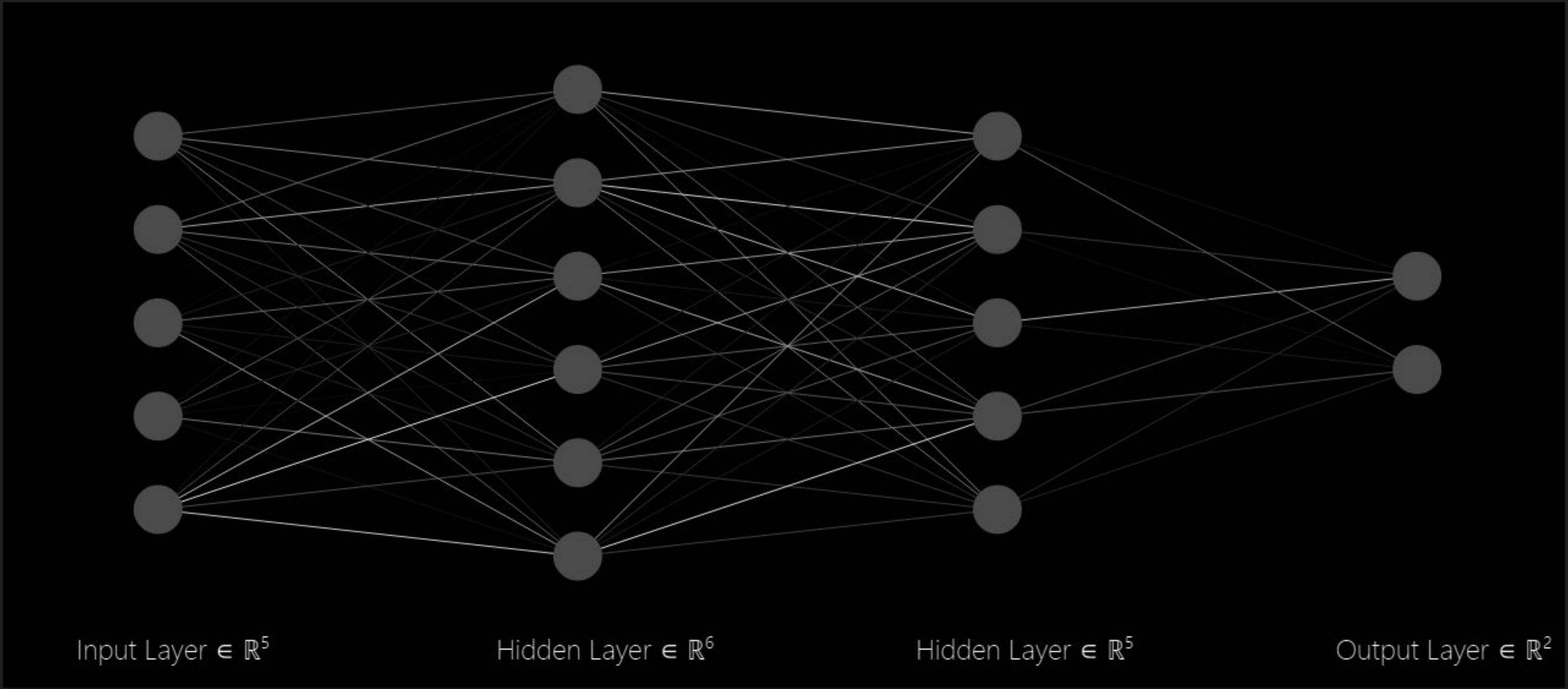
1. The purpose of the workshop
    a. Internalize Cutting Edge Deep NLP
    b. The Reasons behind these Innovations
    c. A Comprehensive Map of Modern Algorithms
2. This Session:
    a. Intro and evolution of modern deep NLP
    b. Transfer Learning using a Modern Deep NLP Algorithm
    c. Encoder-Decoder Architecture

# Prerequisites

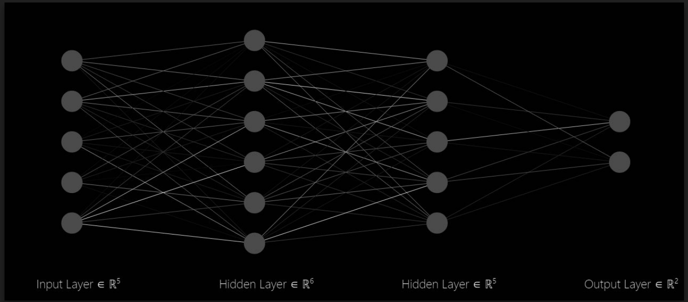
Prerequisites

1. Knowledge of Python
2. Familiarity with Linear Algebra
3. Knowledge of Machine Learning
4. Familiarity with TensorFlow and PyTorch is a plus but not a hard requirement

# Multilayer Perceptron: Architecture



# Multilayer Perceptron: Equations



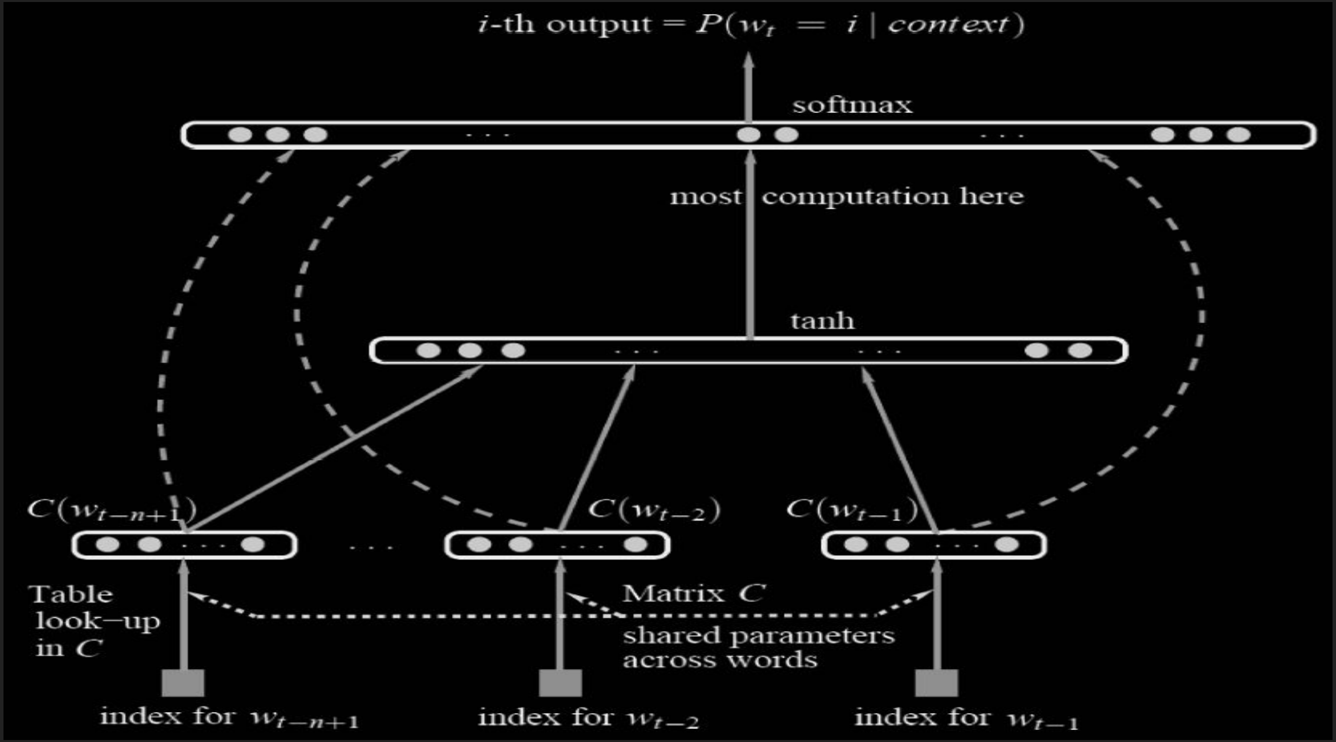

Linear Projection

1) z = Wx + b

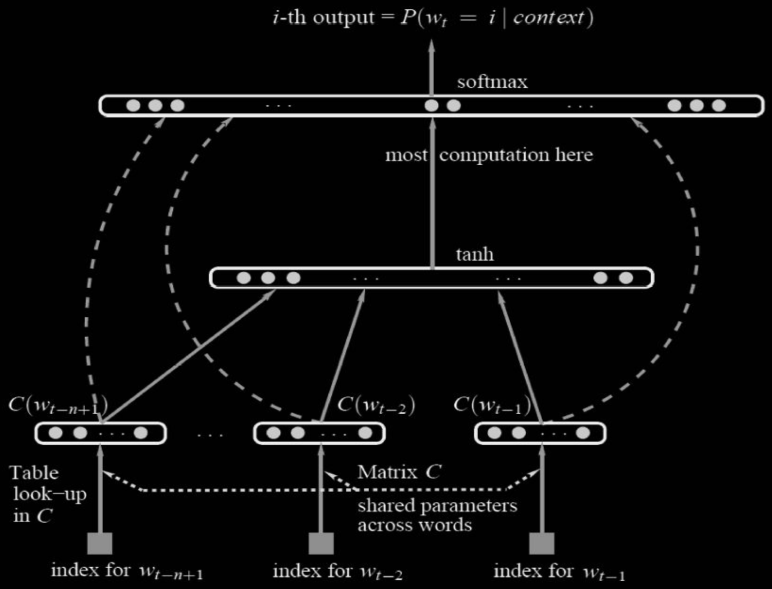
Common Nonlinearities

2) Sig: a = 1 / (1 + e^-z)
3) Tanh: (e^z - e^-z) / (e^z + e^-z)
4) ReLU: z^+ = max(0, z)
5) Leaky ReLU: {
    a) 0.01z for z < 0,
    b) z for z >= 0

# Neural Language Model



# Neural Language Model: Equations



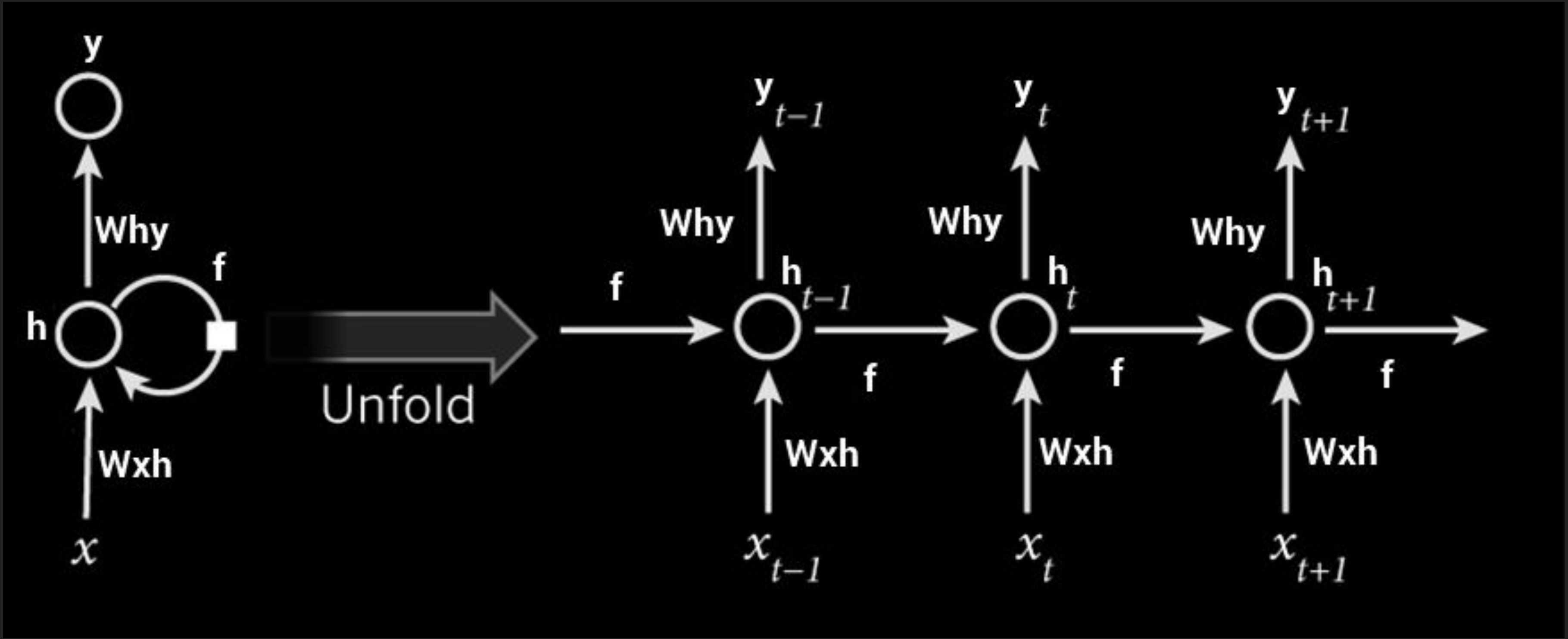

$$f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$$

$$\hat{P}(w_t | w_{t-1}, \ldots, w_{t-n+1}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

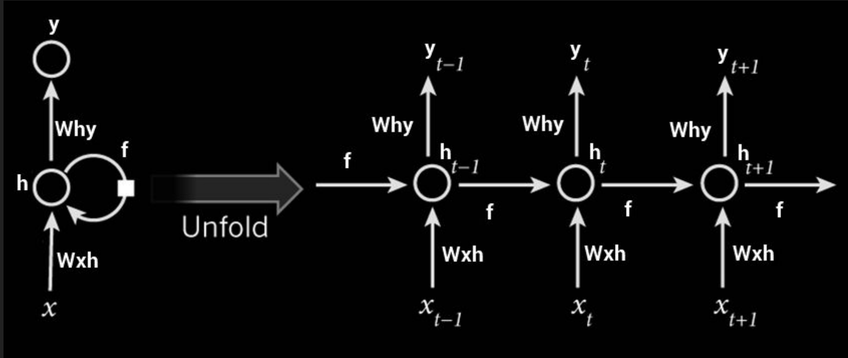
$$L = \frac{1}{T} \sum_t \log f(w_t, w_{t-1}, \cdots, w_{t-n+1}; \theta) + R(\theta).$$

Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where $g$ is the neural network and $C(i)$ is the $i$-th word feature vector.

# Vanilla RNN: Architecture



# Vanilla RNN: Equations




**Elman network**

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

**Jordan network**

$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h)$$
$$y_t = \sigma_y(W_y h_t + b_y)$$

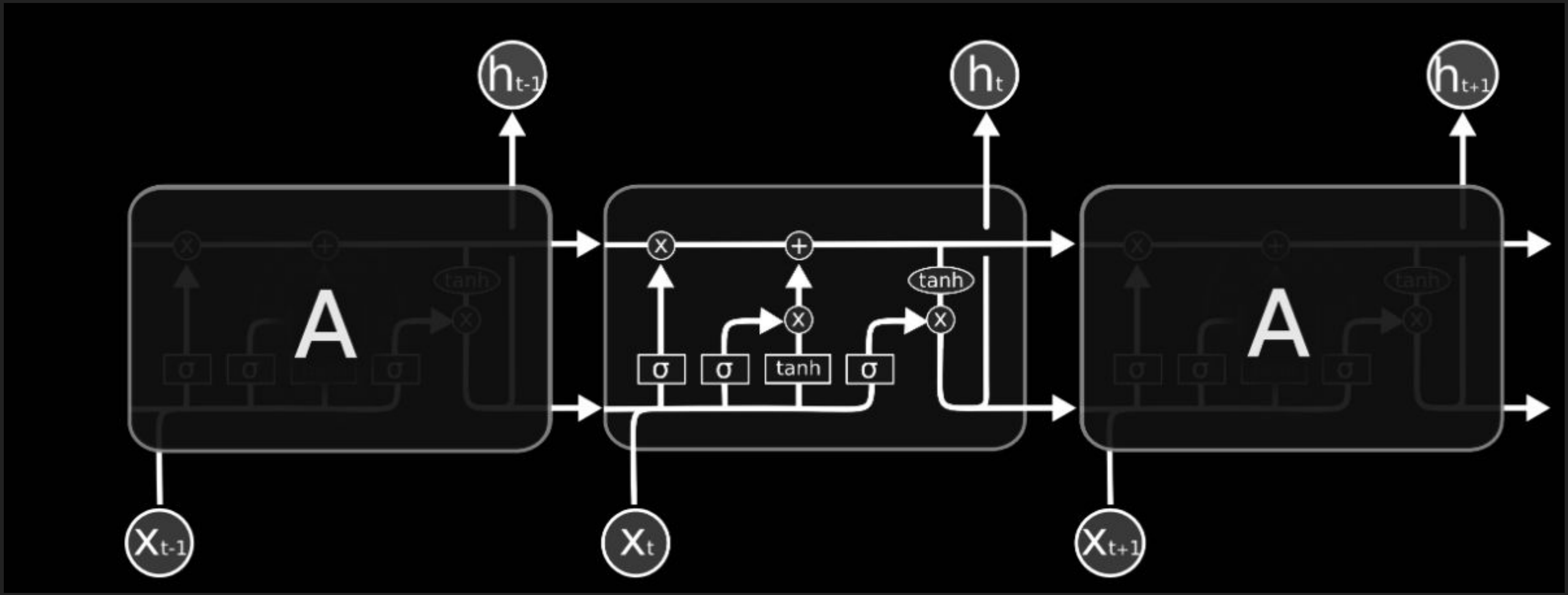
Variables and functions

- $x_t$ : input vector
- $h_t$ : hidden layer vector
- $y_t$ : output vector
- $W$, $U$ and $b$: parameter matrices and vector
- $\sigma_h$ and $\sigma_y$: Activation functions

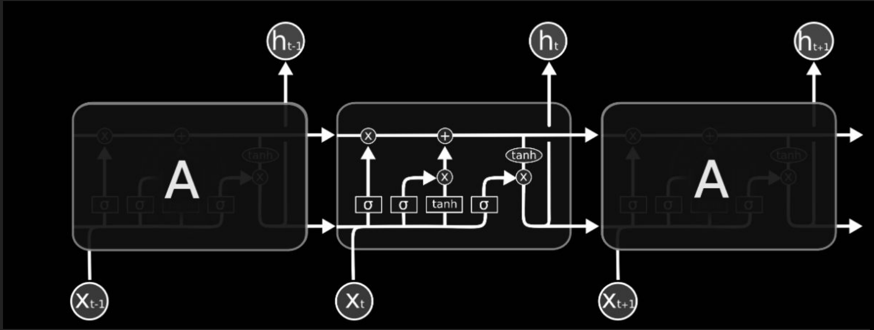
Can you tell me what the difference is?

Which one is the diagram?

# LSTM: Architecture



# LSTM: Equations




$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$

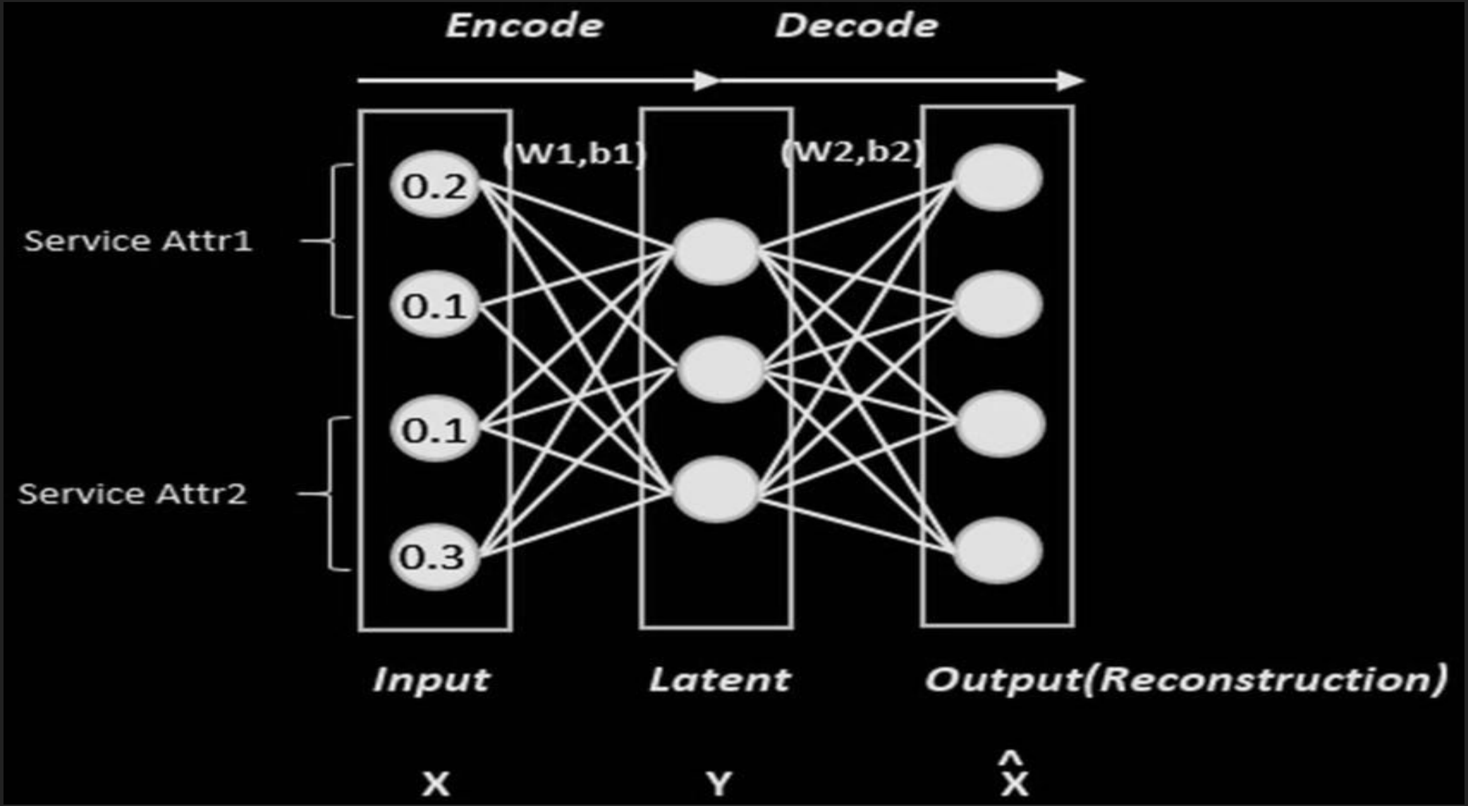
$$h_t = o_t * \tanh\left(C_t\right)$$

- The significance of Cell State
- Gates
- Operations
  - Addition
  - Pointwise Mult.
  - Sigmoid

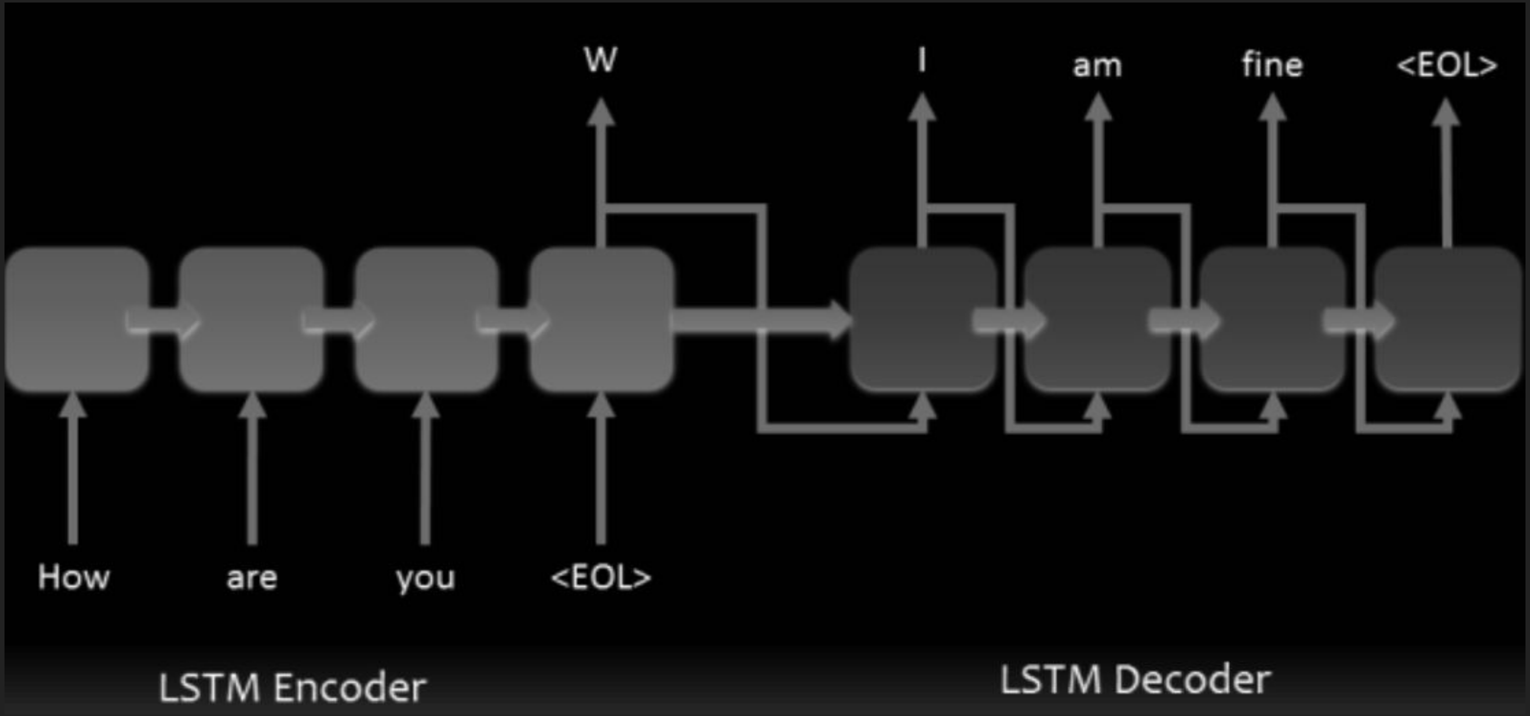
# Questions So Far?

-

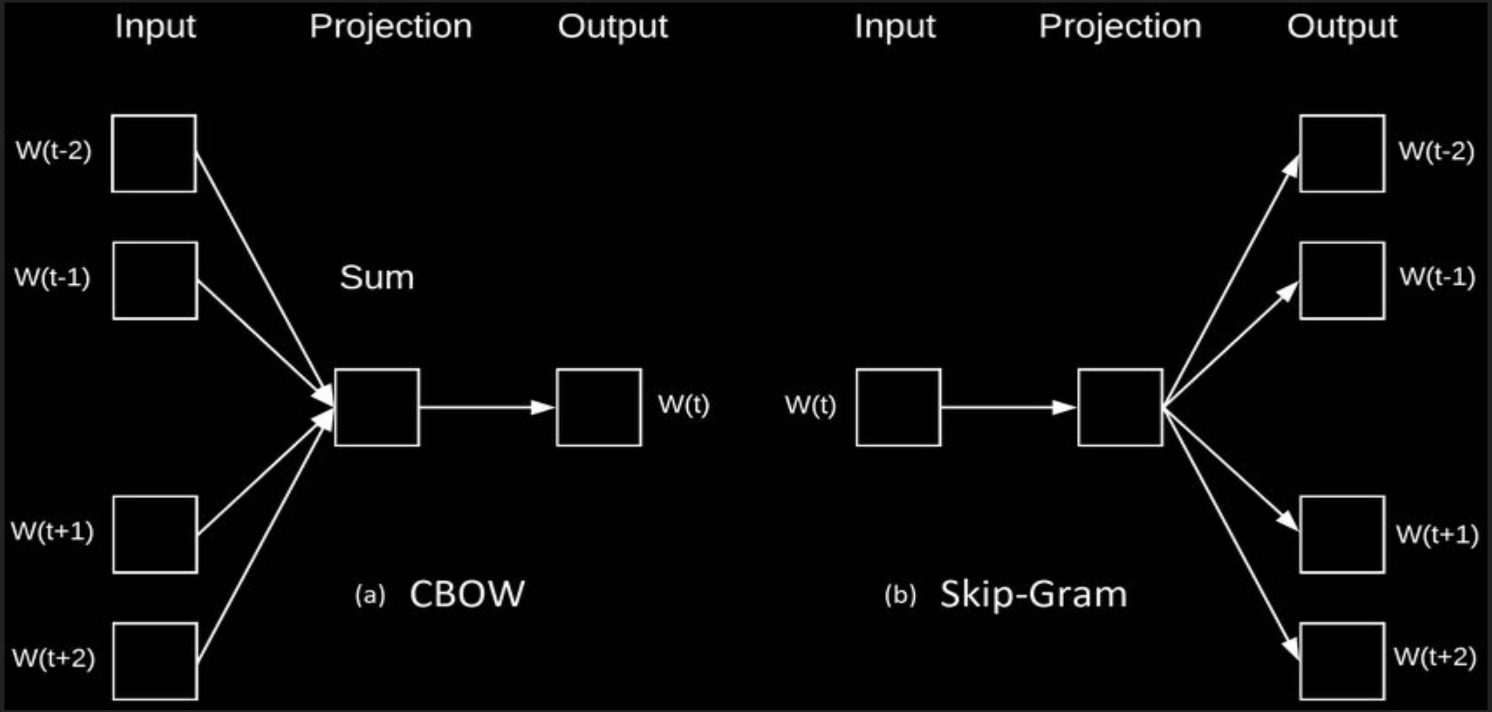
# Autoencoding: Representation Learning



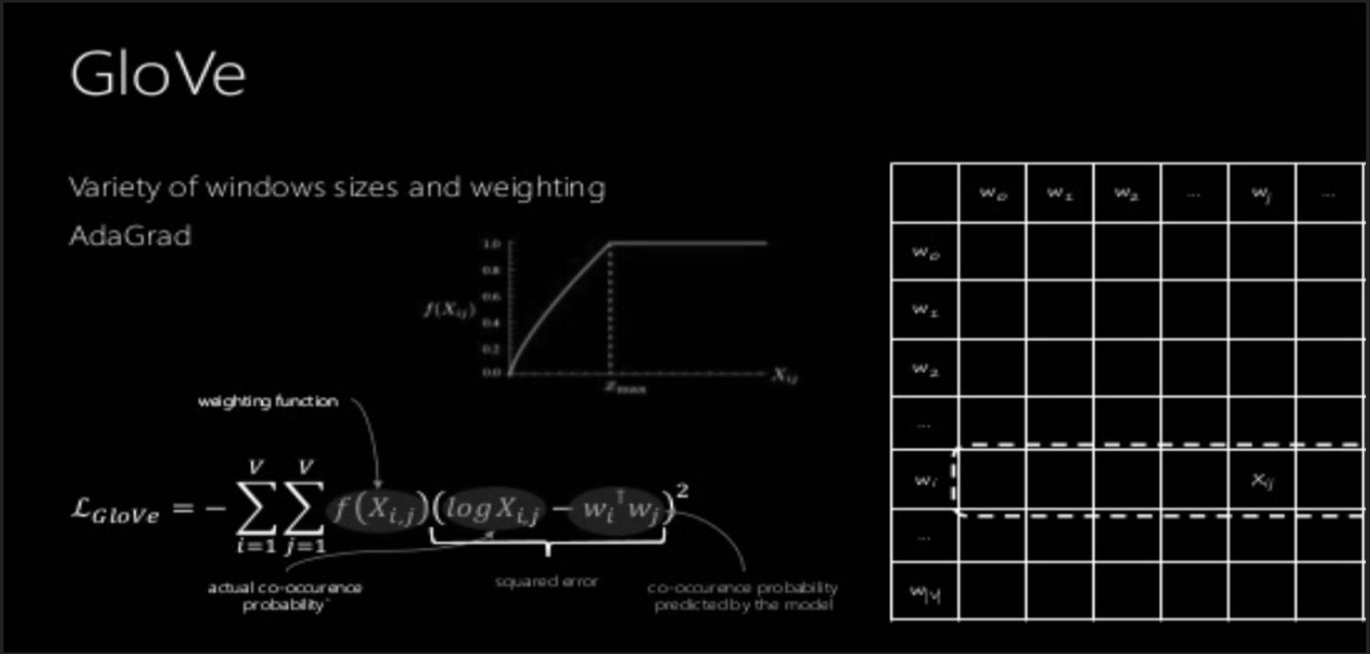
# Encoder-Decoder Architecture: Seq2Seq Models



# Word-to-Vec


Input
Projection
Output
W(t-2)
W(t-1)
Sum
W(t+1)
W(t+2)
W(t)
(a) CBOW
Input
Projection
Output
W(t)
W(t-2)
W(t-1)
W(t+1)
W(t+2)
(b) Skip-Gram

# Word-to-Vec


GloVe
Variety of windows sizes and weighting
AdaGrad
weighting function
squared error
actual co-occurence probability
co-occurence probability predicted by the model
$\mathcal{L}_{GloVe} = -\sum_{i=1}^{V}\sum_{j=1}^{V} f(X_{i,j})(\log X_{i,j} - w_i^{\top} w_j)^2$
$f(X_{ij})$
$X_{ij}$
$x_{max}$
$X_{ij}$

# Questions?