Recurrent Neural Networks Part 1: Introduction

Lecture 13

Subir Varma

So Far ...

$$Y = f(X)$$

$X = (x_1, x_2, \ldots, x_n)$

Deep Learning Network

$P(Y = cat|X)$

$P(Y = dog|X)$

.
.
.

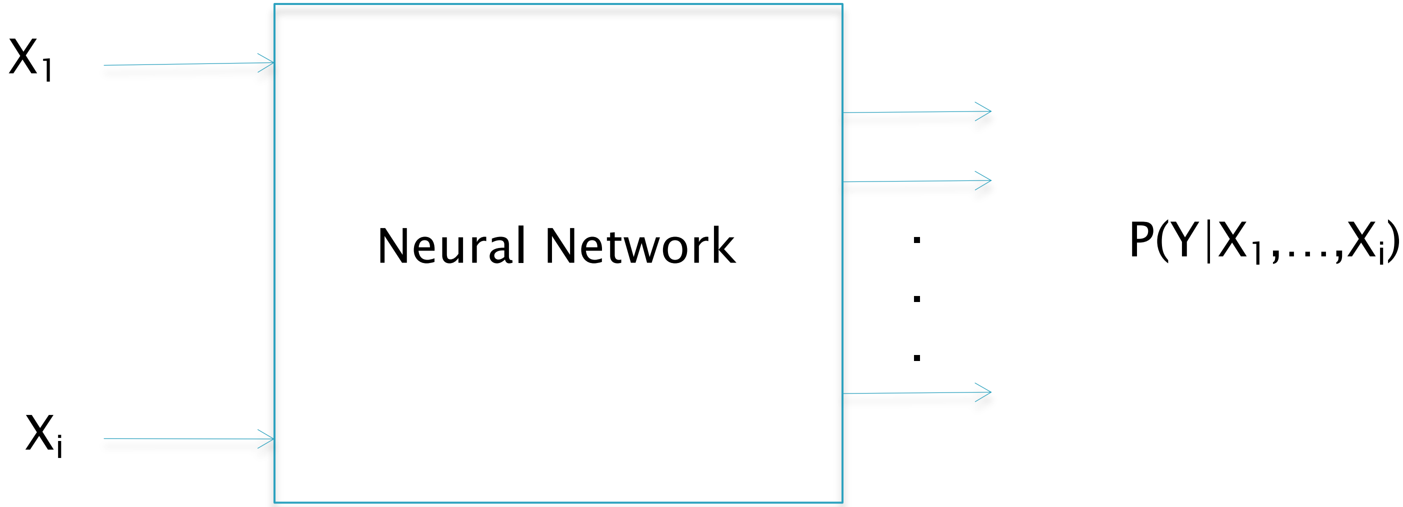
$P(Y = car|X)$

Input: Tensor

Output: Probability Distribution

What About Sequences?

Y depends on the sequence $(X_1,\ldots,X_i)$
We need to Estimate $P(Y|X_1,\ldots,X_i)$

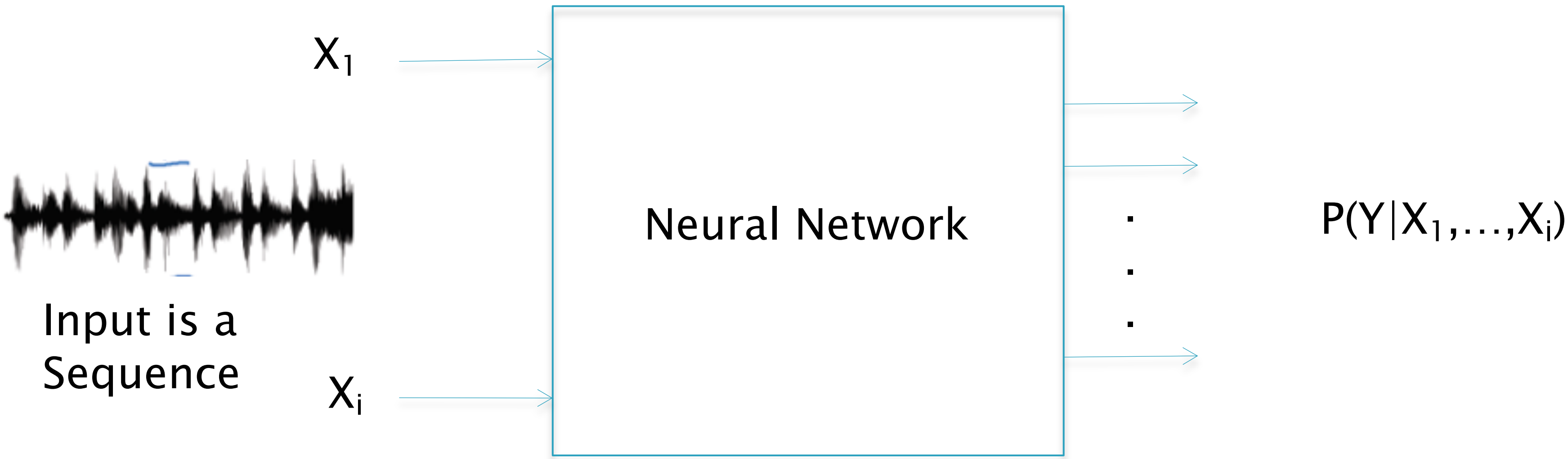


What is wrong with this?

DFNs and CNNs are not Modular: Larger input requires a larger network

Is there a single network that works irrespective of the length of the sequence?

Y depends on the sequence $(X_1,\ldots,X_i)$
We need to Estimate $P(Y|X_1,\ldots,X_i)$

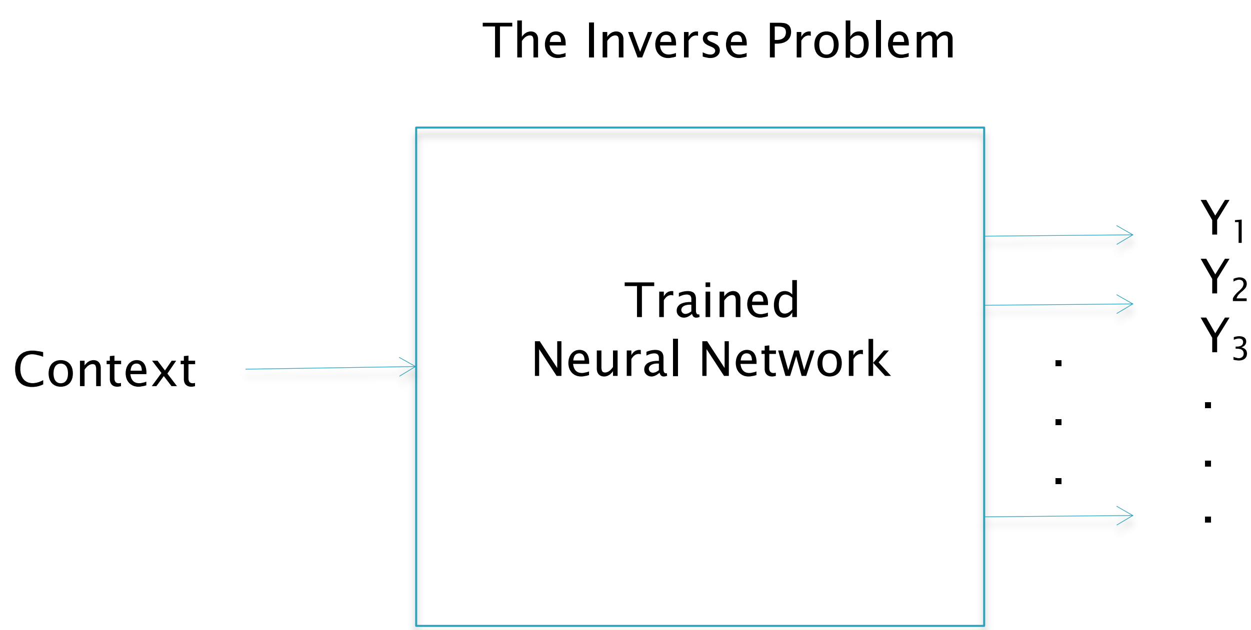


Can a single network to estimate $P(Y|X_1,\ldots,X_i)$ irrespective of the length of the sequence?

Examples:

X = word ➔ Sequence of X = sentence
X = image ➔ Sequence of X = Video clip
X = audio sample ➔ Sequence of X =Audio Clip

Generating Sequences



Given a Context, can the Network generate a Sequence associated with the Context?

If the context is another sequence (X_1, X_2, …, X_n) then this corresponds to Machine Translation

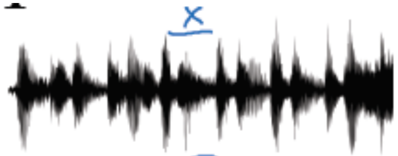
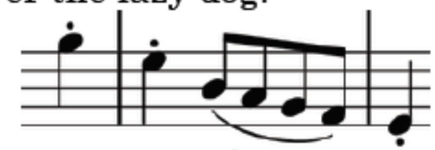
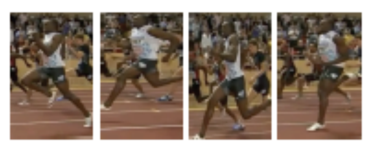
What Problems do RNNs Solve?

ConvNets finds patterns in space, but what about patterns in time?

Why is this important?

- Language – Word Sequences
- Video – Picture Sequences
- Sound
- Musical Notes
- Financial Data

Examples of Sequence Processing

	x		y
Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	Ø	→	
Sentiment classification	"There is nothing to like in this movie."	→	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

Where do RNNs shine?

Prediction Problems involving Sequences

Predict the next word in a sentence

Predict the next musical note

Predict the stock price for tomorrow

Classification
Prediction
Generation

History of RNNs

- RNNs were first proposed in the 1970s
- The Back Propagation Through Time (BPTT) training algorithm was discovered in the 1980s
- Progress in the area was held up due to the difficulty in training RNNs – The Vanishing Gradient Problem
- LSTM (Long Short Term Memories) were introduced in the early 1990s – Solution to Vanishing Gradients
- Transformers: Introduced in 2016, a generalization of the RNN architecture
- Several recent successes:
  - Google Translate: Now entirely based on LSTMs
  - Speech Transcription Systems: State of the Art Performance
  - Image Captioning

RNN Architecture

Problem

How can we compute $P(Y|X_1,..,X_i) = h(X_1,X_2,\ldots X_i)$ for variable number of inputs, using a single model?

Define a State Variable (or Hidden Variable Z), such that
$Z_0 = X_0$
$Z_i = f(Z_{i-1},X_i)$, $i = 1, 2, \ldots$

Define the output as a function of the State Variable
$Y_i = h(Z_i)$, $i = 0, 1, 2, \ldots$

Then
$Y_0 = h(Z_0) = h(X_0)$
$Y_1 = h(Z_1) = h(f(Z_0,X_1)) = h(f(X_0,X_1))$
$Y_2 = h(Z_2) = h(f(Z_1,X_2)) = h(f(f(X_0,X_1),X_2))$

The sequence Y_n can be described using only two functions f and h

State Equations

$Z_0 = X_0$

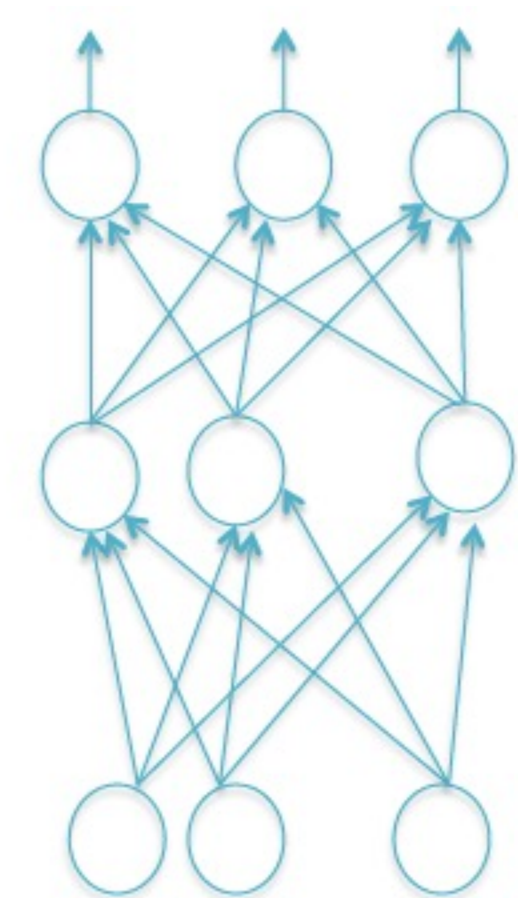
$Z_{i+1} = f(Z_i, X_{i+1}),\ i = 0, 1, \ldots$

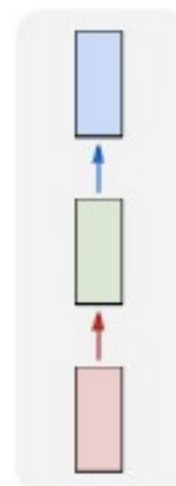
$Y_{i+1} = h(Z_{i+1}),\ i = 0, 1, 2, \ldots$

$$Z_{i+1} = f(WZ_i + UX_{i+1})$$
$$Y_{i+1} = h(VZ_{i+1})$$

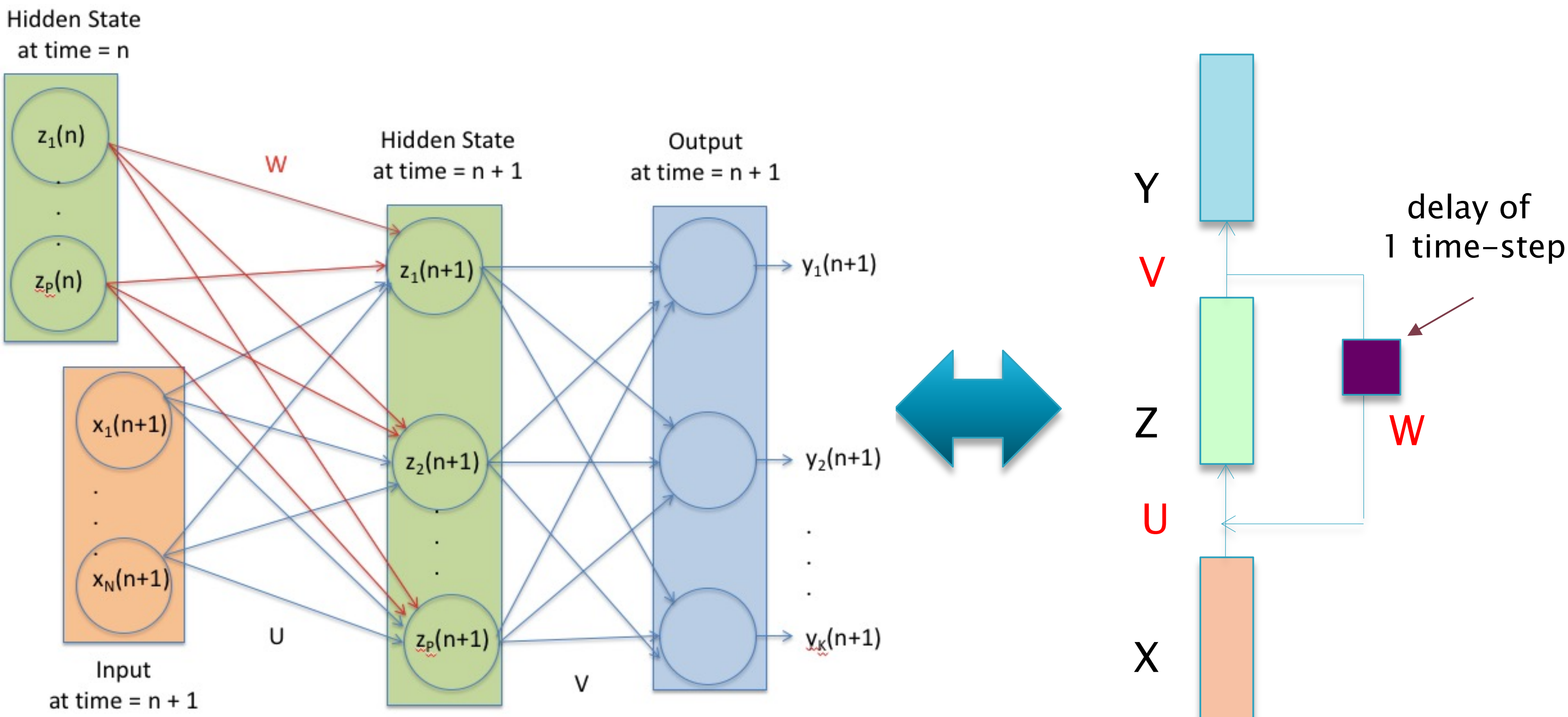
How can we represent these equations as a Neural Network?

Dense Feed Forward Neural Networks





RNN Parametrization



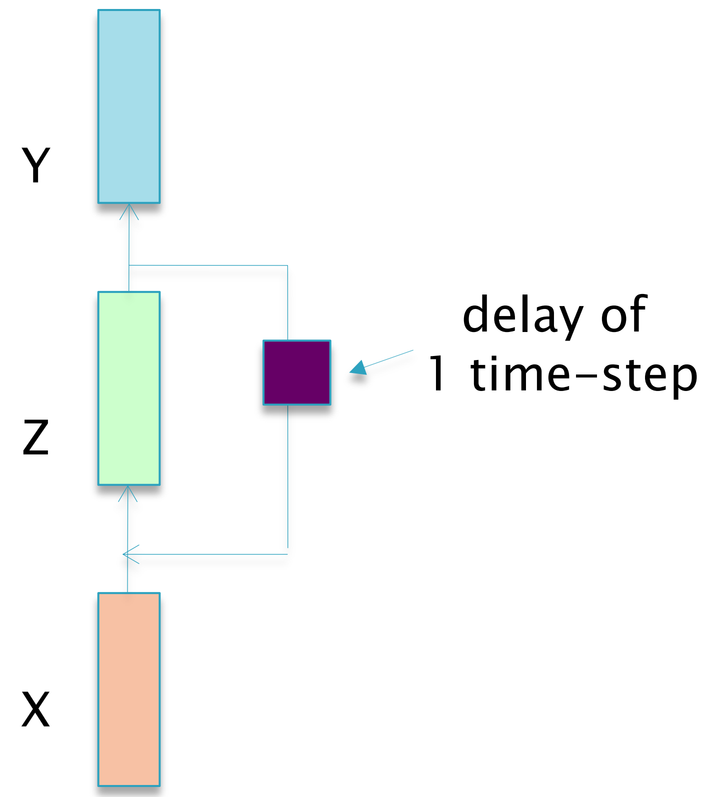
$$Z_{i+1} = f(WZ_i + UX_{i+1})$$

$$Y_{i+1} = h(VZ_{i+1})$$

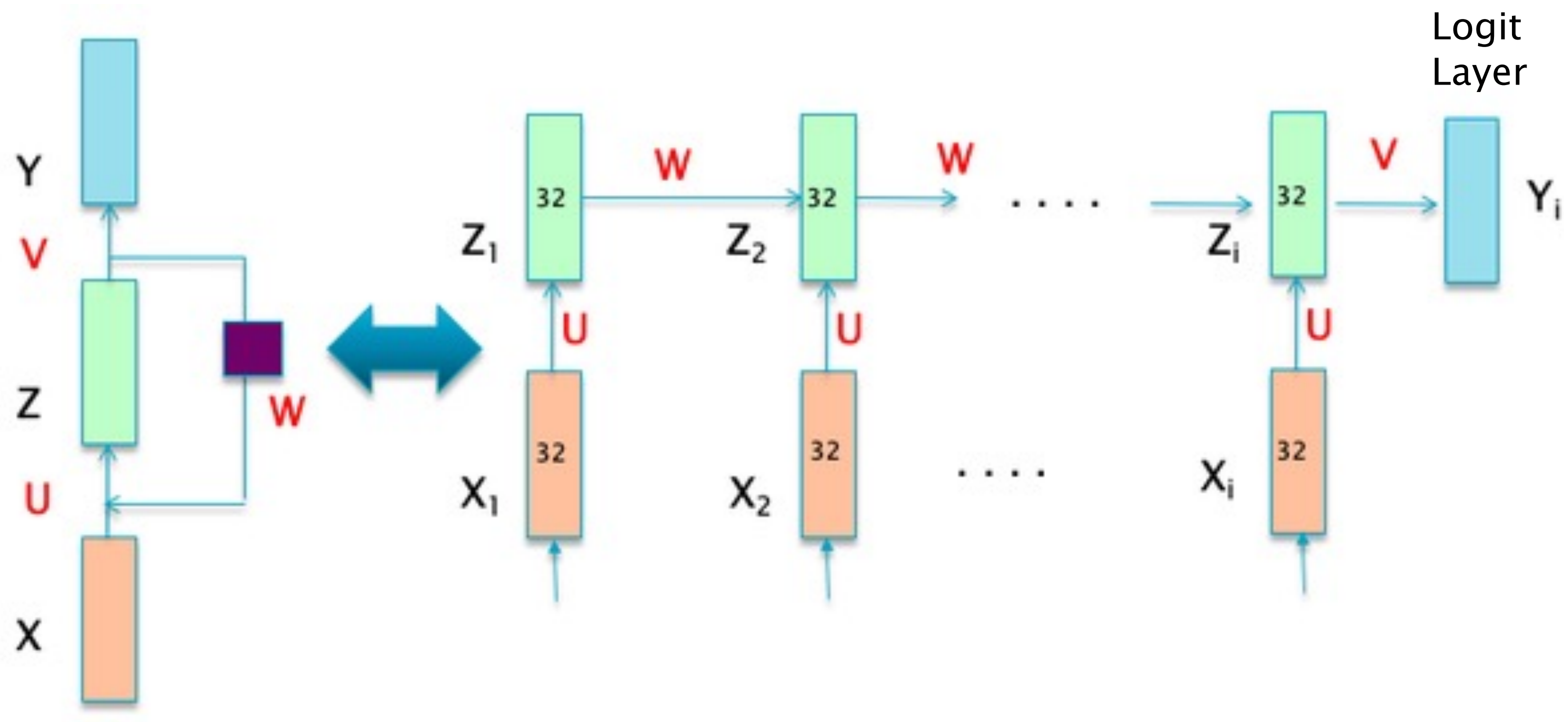
Turning the Recursion into a Neural Network

$$Z_{i+1} = f(WZ_i + UX_{i+1})$$
$$Y_{i+1} = h(VZ_{i+1})$$

- The state Z becomes a hidden layer in a Neural Network
- Z is a function of not just input X but also the previous value of the state



RNN Unfolding In Time: Equivalent Feed Forward Network



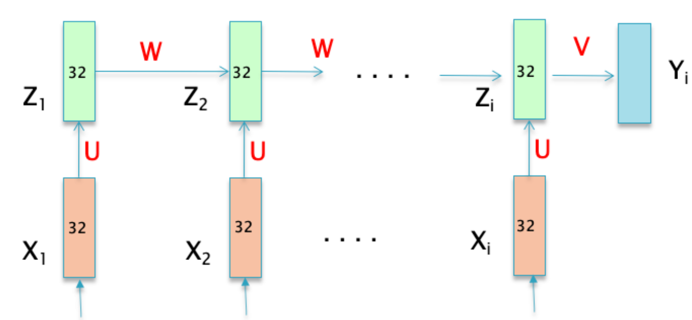
$$Z_{i+1} = f(WZ_i + UX_{i+1})$$
$$Y_{i+1} = h(VZ_{i+1})$$

Represents the network at different instants in time

How Do RNNs Work?

- ConvNets detect patterns in space. Since the same filter is used at all spatial locations, this results in translational invariance
- RNNs detect two types of patterns:
  - Patterns that occur at a particular instant in time
  - Patterns that are spread over time

Hence when a RNN makes a classification, its decision is influenced not only by the current input, but what has happened in the past



Contrasting RNNs with ConvNets

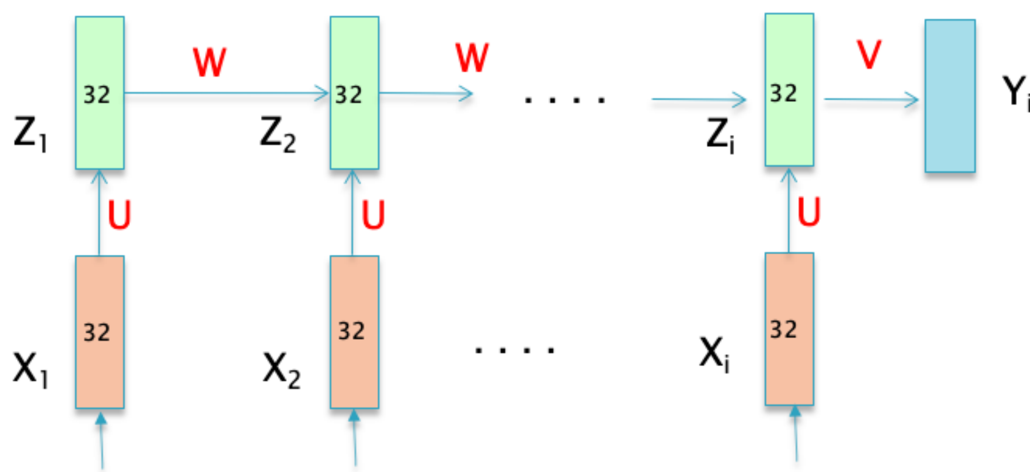
- How pattern recognition in Convnets differs from that in RNNs
  - ConvNets slide a single filter over the entire image
    RNNs slide a single filter over the entire input sequence => Translational Invariance in time.

ConvNets ⟷ Translational Invariance in Space

RNNs ⟷ Translational Invariance in Time

Contrasting RNNs with ConvNets

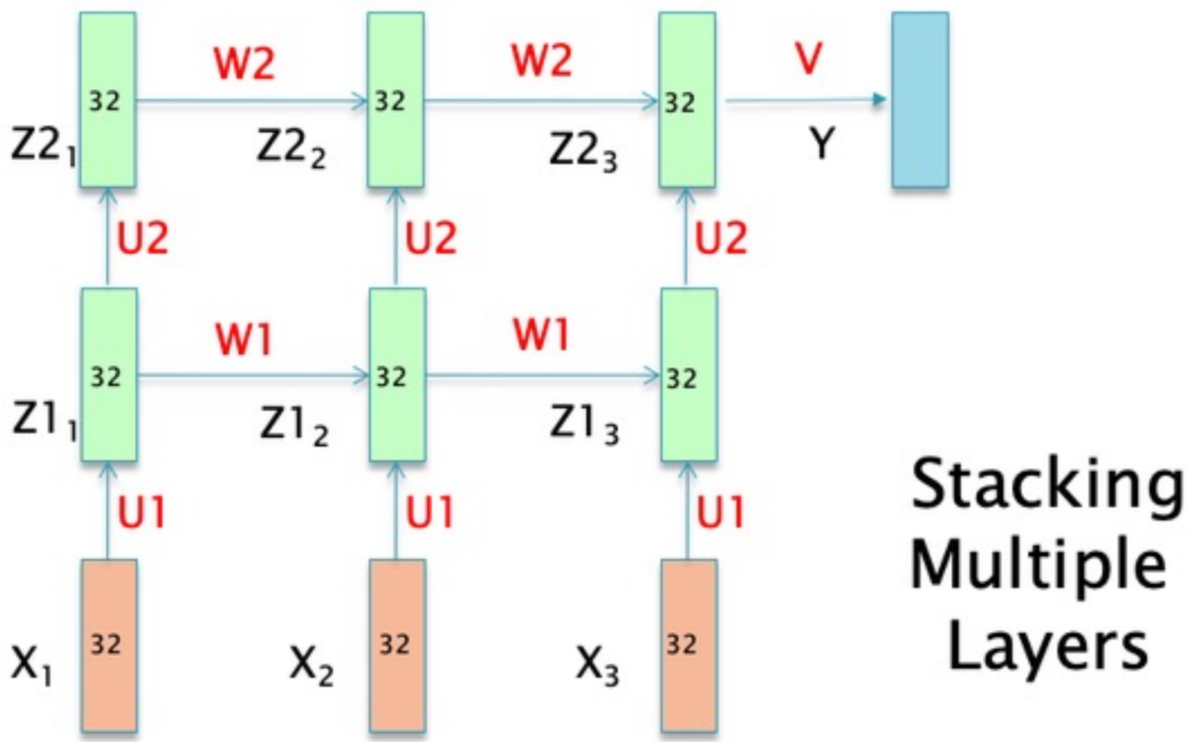
- How data representation in the Hidden Layer in RNNs is different from that in ConvNets
  - Higher Layers in ConvNets create representations at higher levels of abstraction
  - The Hidden Layer in RNNs captures patterns that are spread in time, but at the same level of abstraction



Deep RNNs

Looks for temporal
Patterns at the second
Level of abstraction

Looks for temporal
Patterns at the first
Level of abstraction

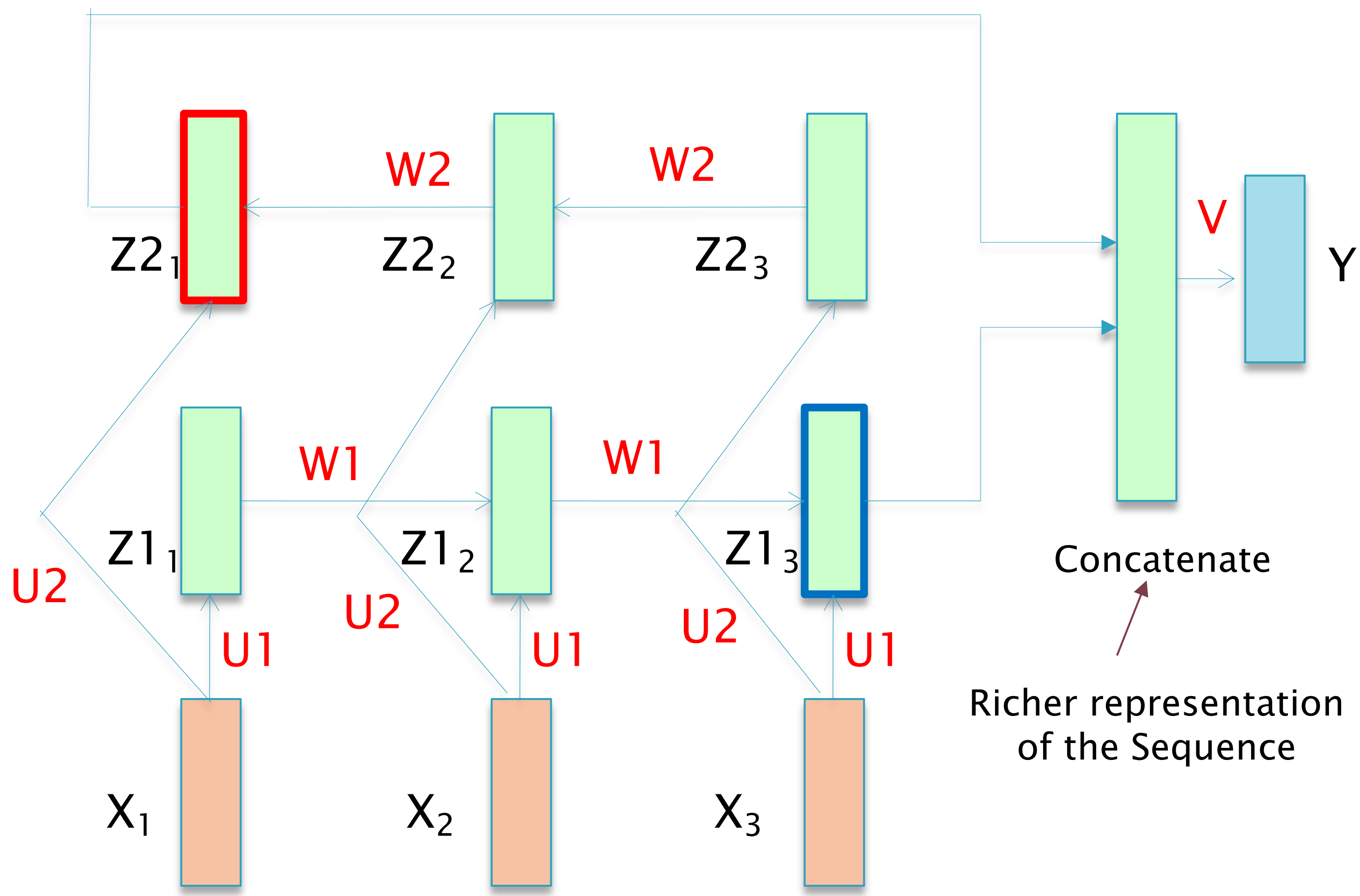


Bi-Directional RNNs

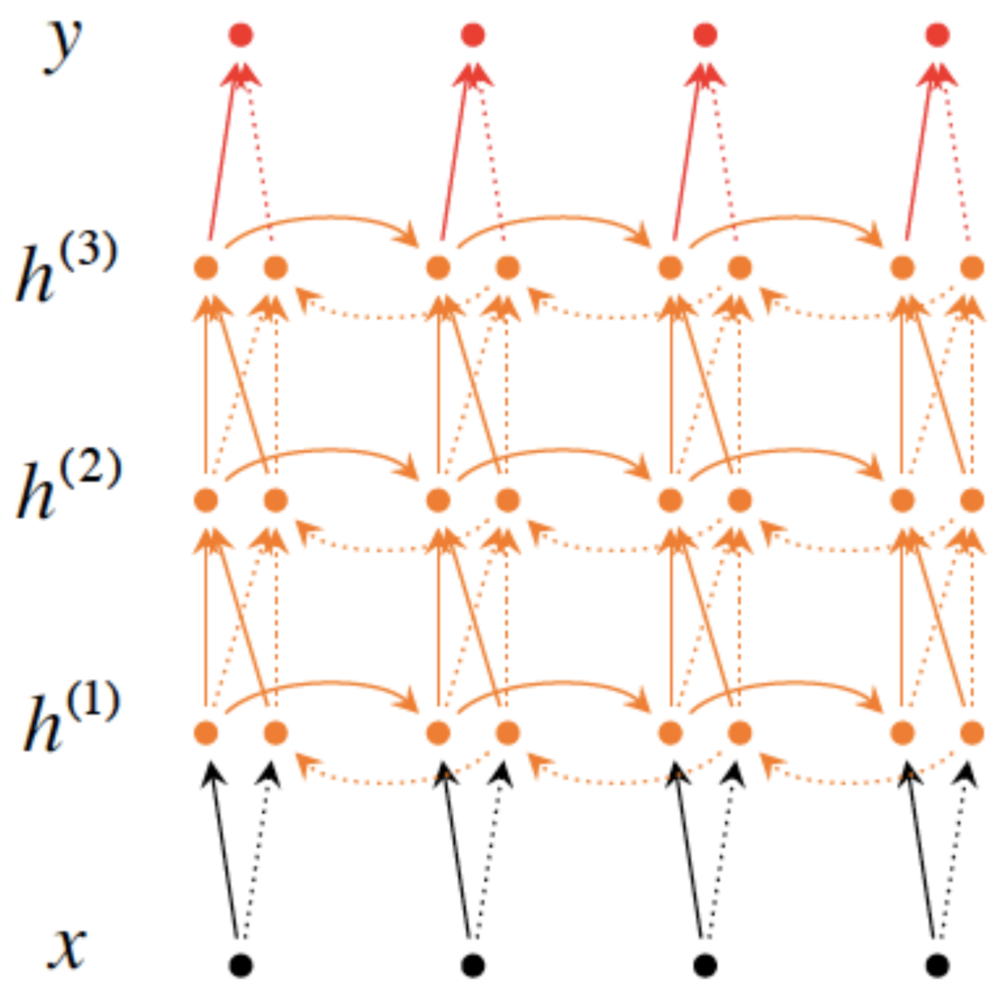
Output is determined By Past as well as Future Inputs

Detects temporal patterns from past and future

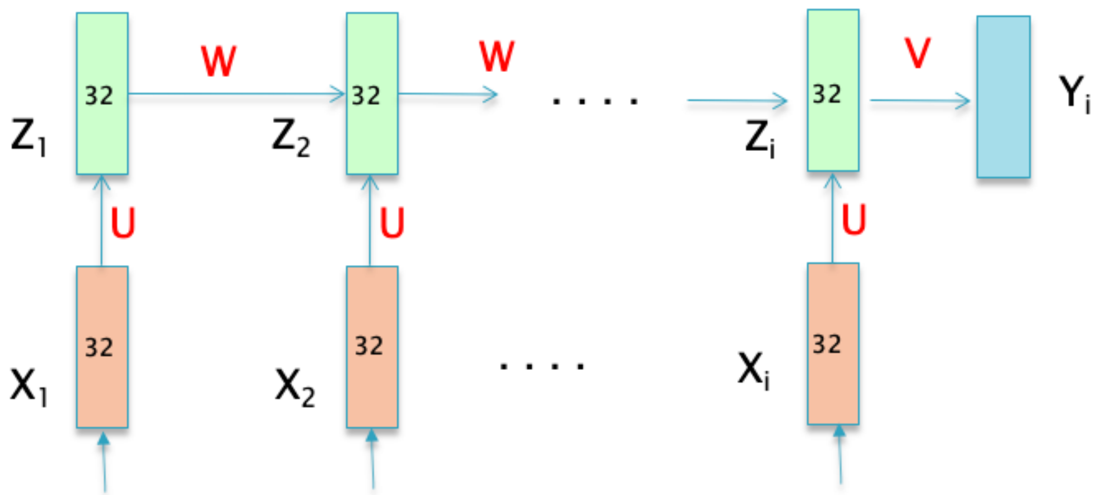
Example: Natural Language Processing



Deep Bi-Directional RNNs



Types of RNNs: Multiple Inputs and Single Output

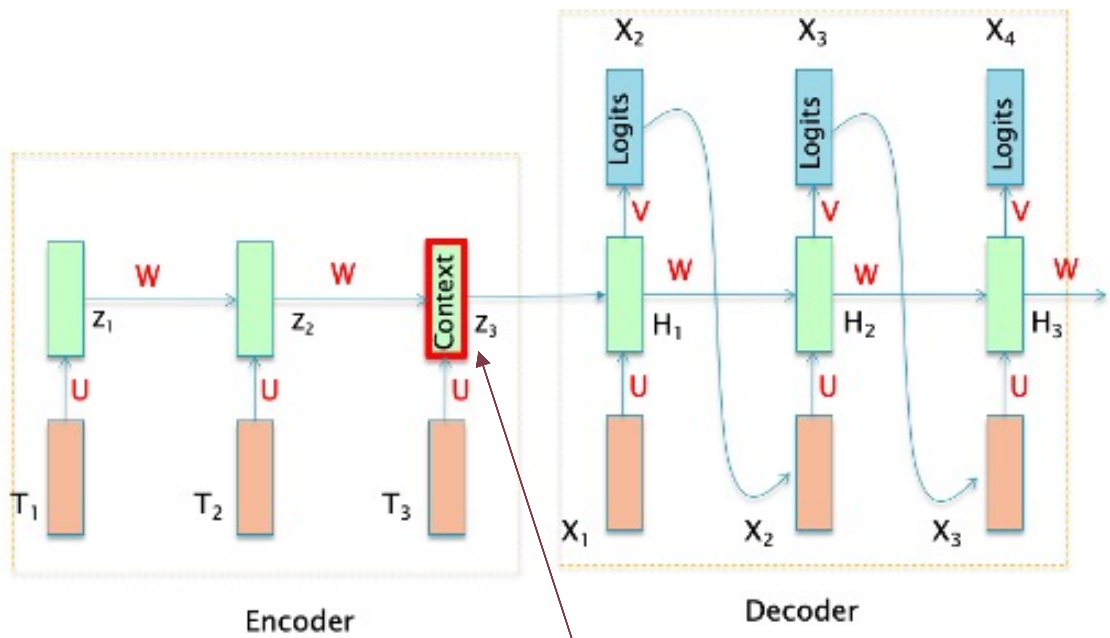


Applications:
- Prediction Problems
- Sentiment Classification
- Video Activity Recognition
- DNA Sequence Analysis

Types of RNN: Multiple Inputs and Multiple Outputs – Encoder Decoder Systems

The output of the network is a Word Sequence

Auto Regressive Generation



Summarizes Input Sequence

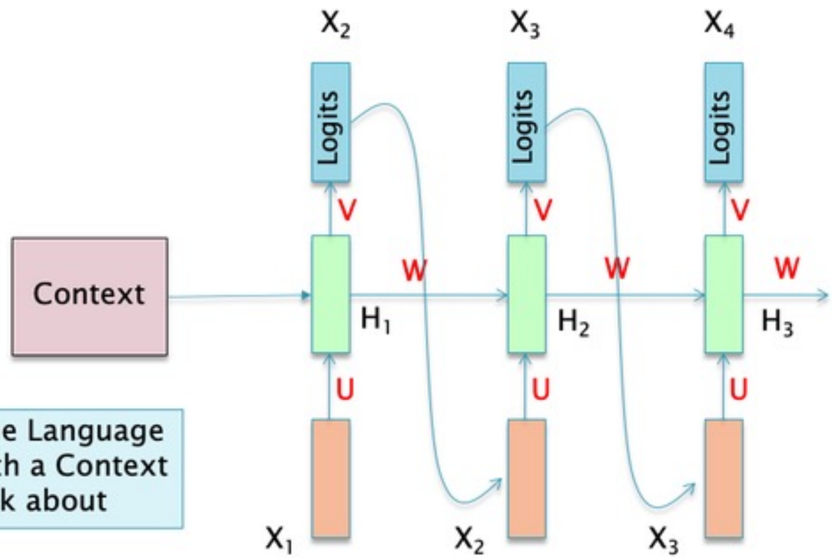
Applications:

- Machine Translation
- Speech Transcription
- Auto Reply
- Question Answering

Types of RNNs: Multiple Outputs Language Models

A sentence related to the context
Using a Language Model

Auto Regressive Generation



The output of the network is a Word Sequence

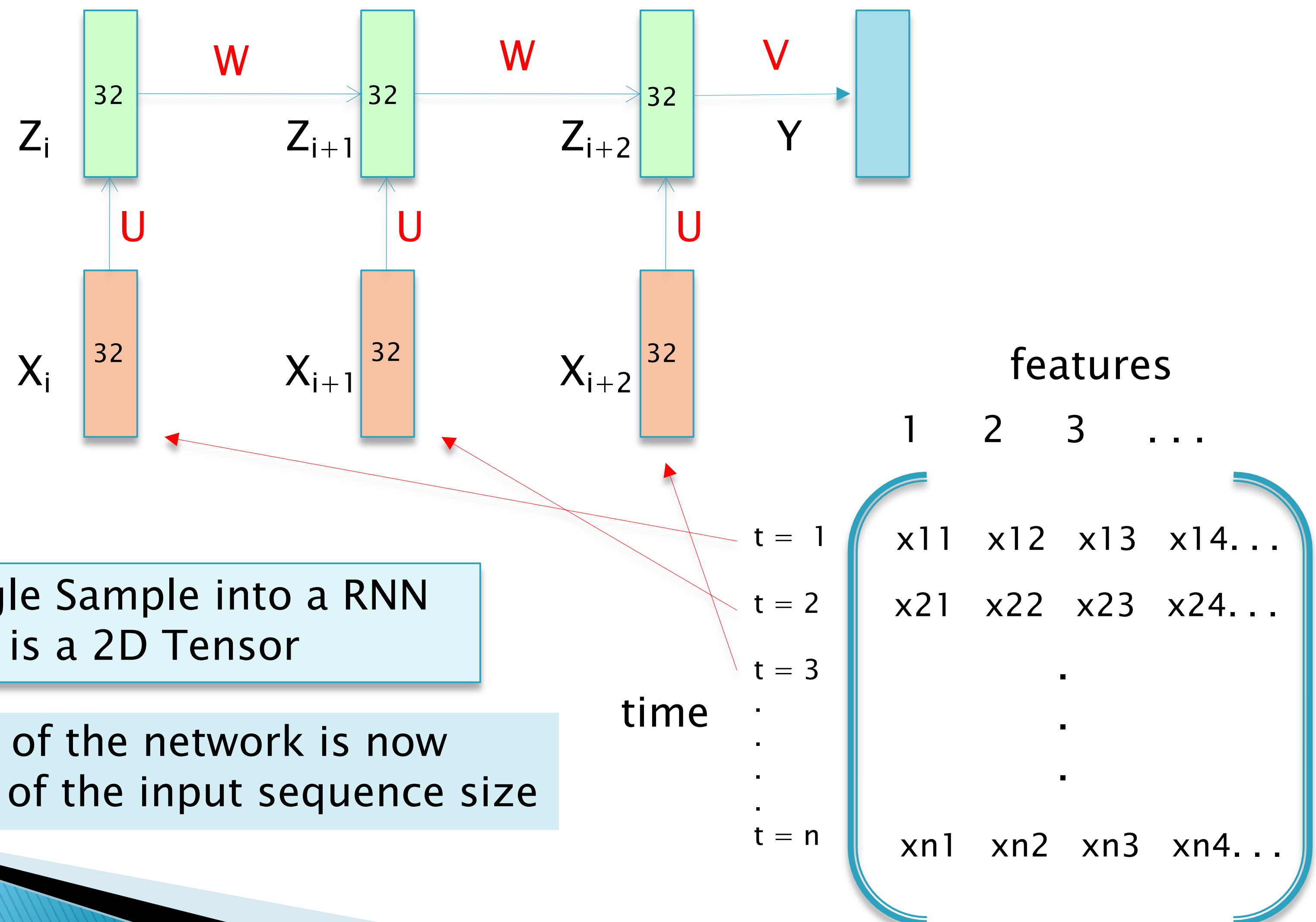
Supply the Language Model with a Context to talk about

Context – A high level representation

Examples:
- An Image
- A sound waveform

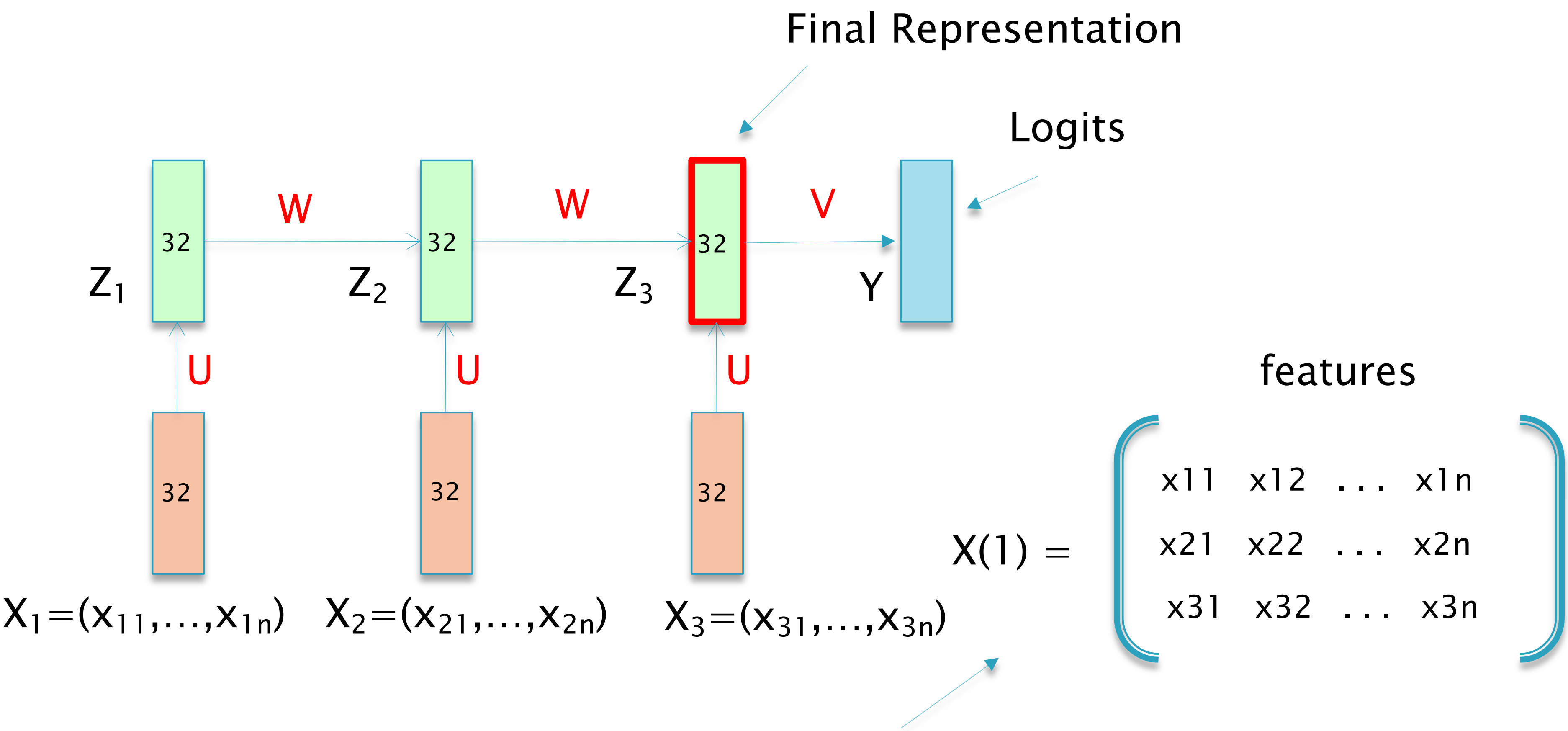
Modeling RNNs with Keras
(Chollet, Chapter 10-Deep Learning for Time Series)

Loading Data into a RNN

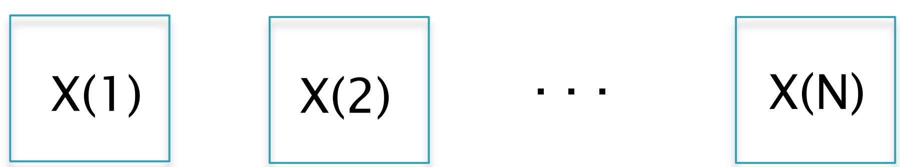


A Single Sample into a RNN
is a 2D Tensor

Depth of the network is now
A function of the input sequence size

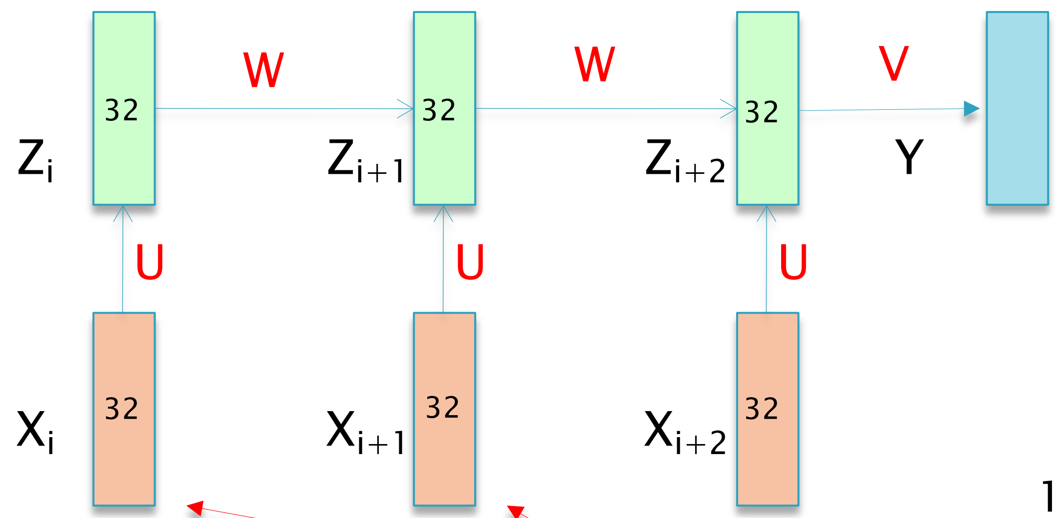


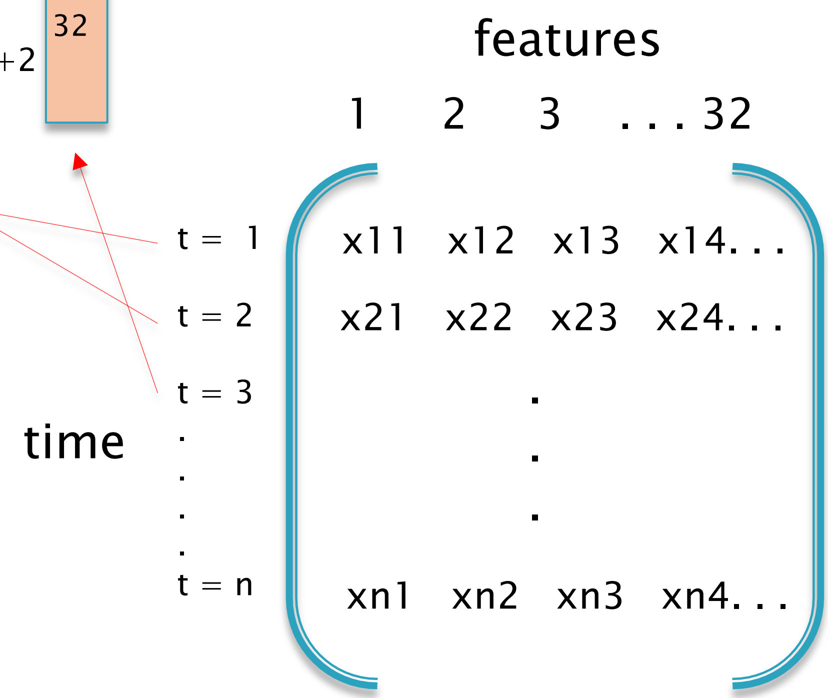
The matrix X(1) forms a single training sample



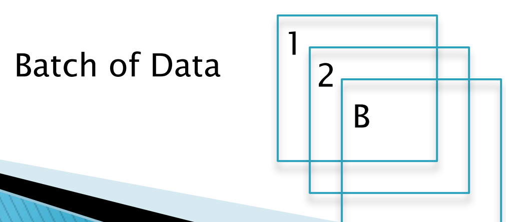
Training Data Set

Loading Data into a RNN





A Batch of Data Samples form a
3D Tensor: (sample x time x features)



Loading Data into a RNN: IMDB

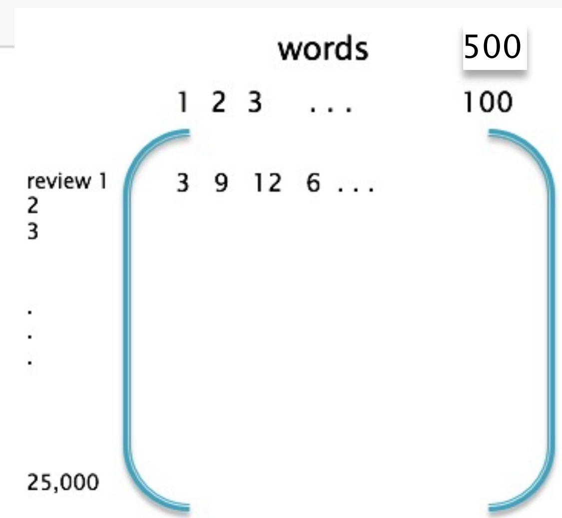
```python
from keras.datasets import imdb
from keras.preprocessing import sequence

max_features = 10000  # number of words to consider as features
maxlen = 500  # cut texts after this number of words (among top max_features most common words)
batch_size = 32

print('Loading data...')
(input_train, y_train), (input_test, y_test) = imdb.load_data(num_words=max_features)
print(len(input_train), 'train sequences')
print(len(input_test), 'test sequences')

print('Pad sequences (samples x time)')
input_train = sequence.pad_sequences(input_train, maxlen=maxlen)
input_test = sequence.pad_sequences(input_test, maxlen=maxlen)
print('input_train shape:', input_train.shape)
print('input_test shape:', input_test.shape)
```

Size of Dictionary

Load the data as lists of integers

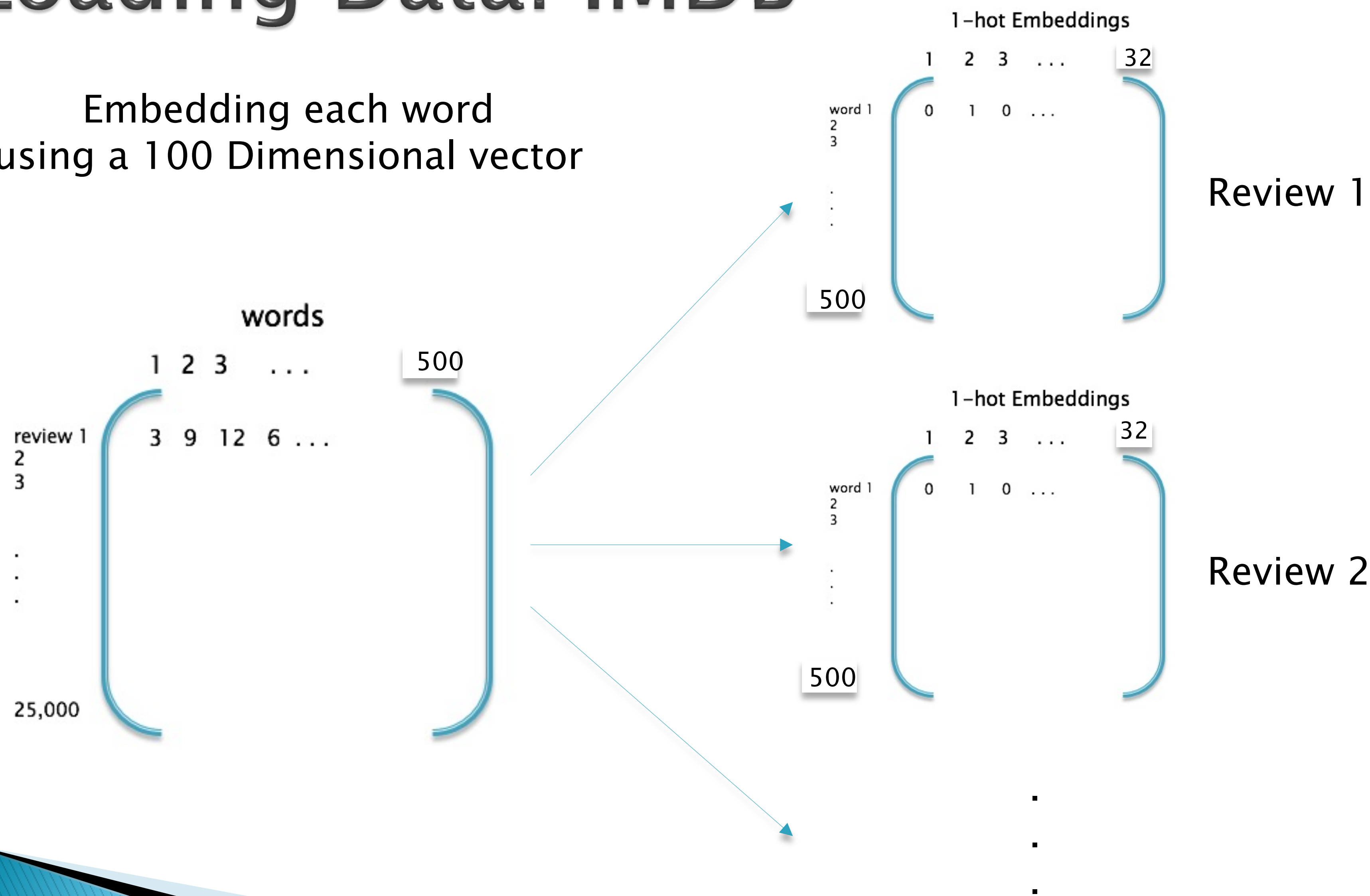
Turns the lists of integers into a 2D integer tensor of shape (samples,maxlen)

```
Loading data...
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
input_train shape: (25000, 500)
input_test shape: (25000, 500)
```

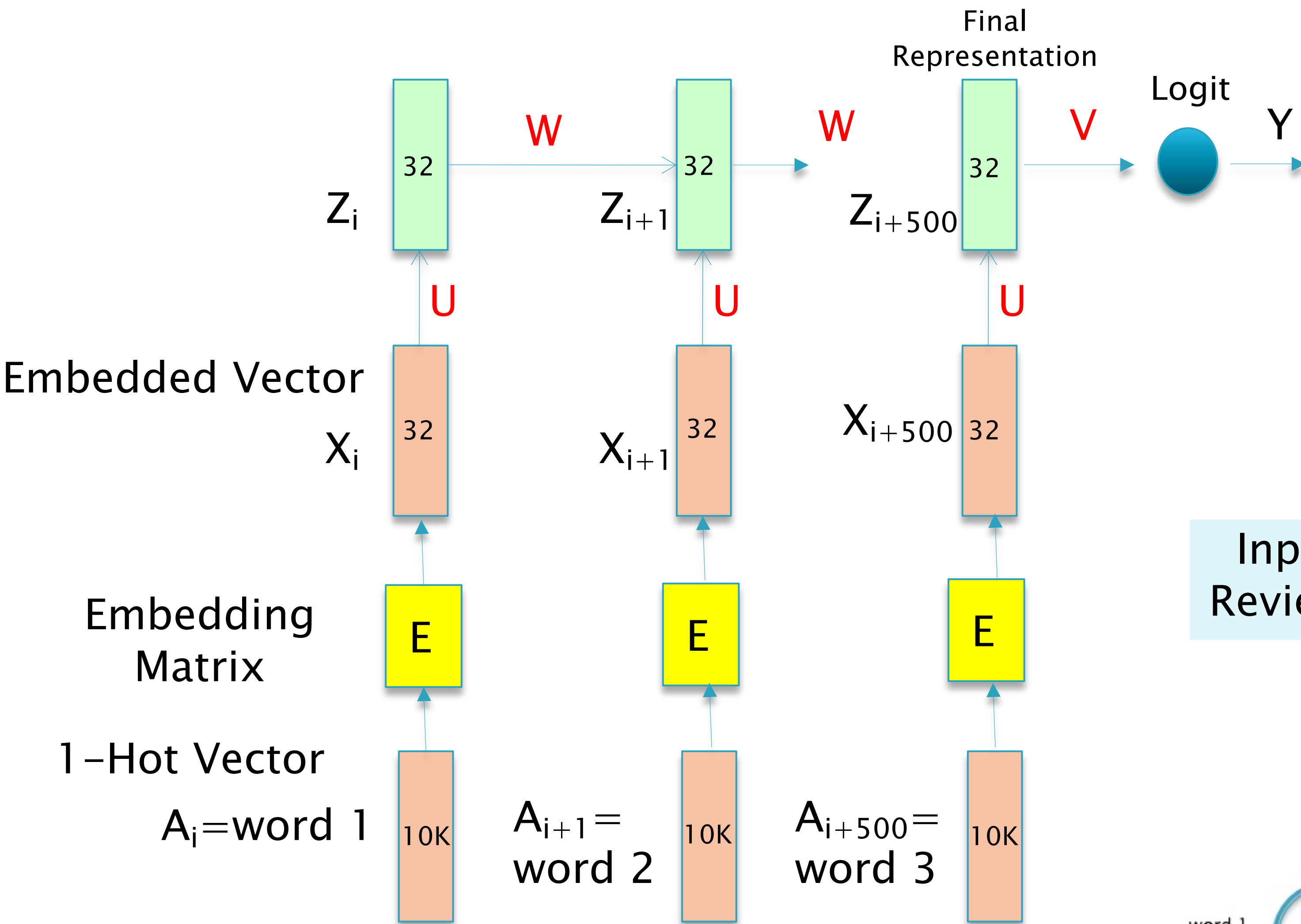


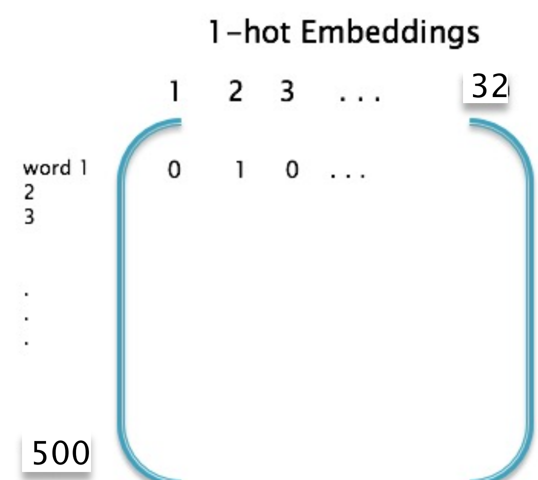
Loading Data: IMDB

Embedding each word
using a 100 Dimensional vector



Inputting a Single Review into the RNN



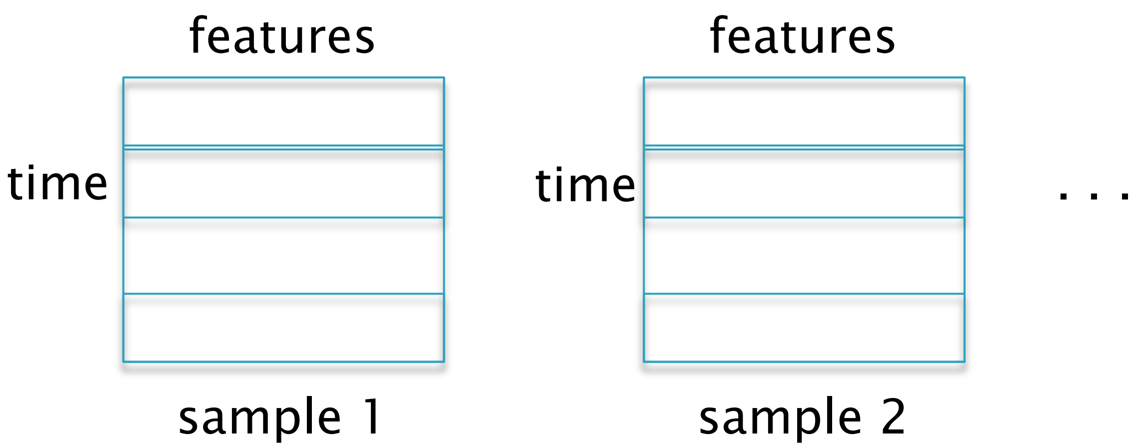


Specifying the Model

```python
from keras.layers import Dense

model = Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(input_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

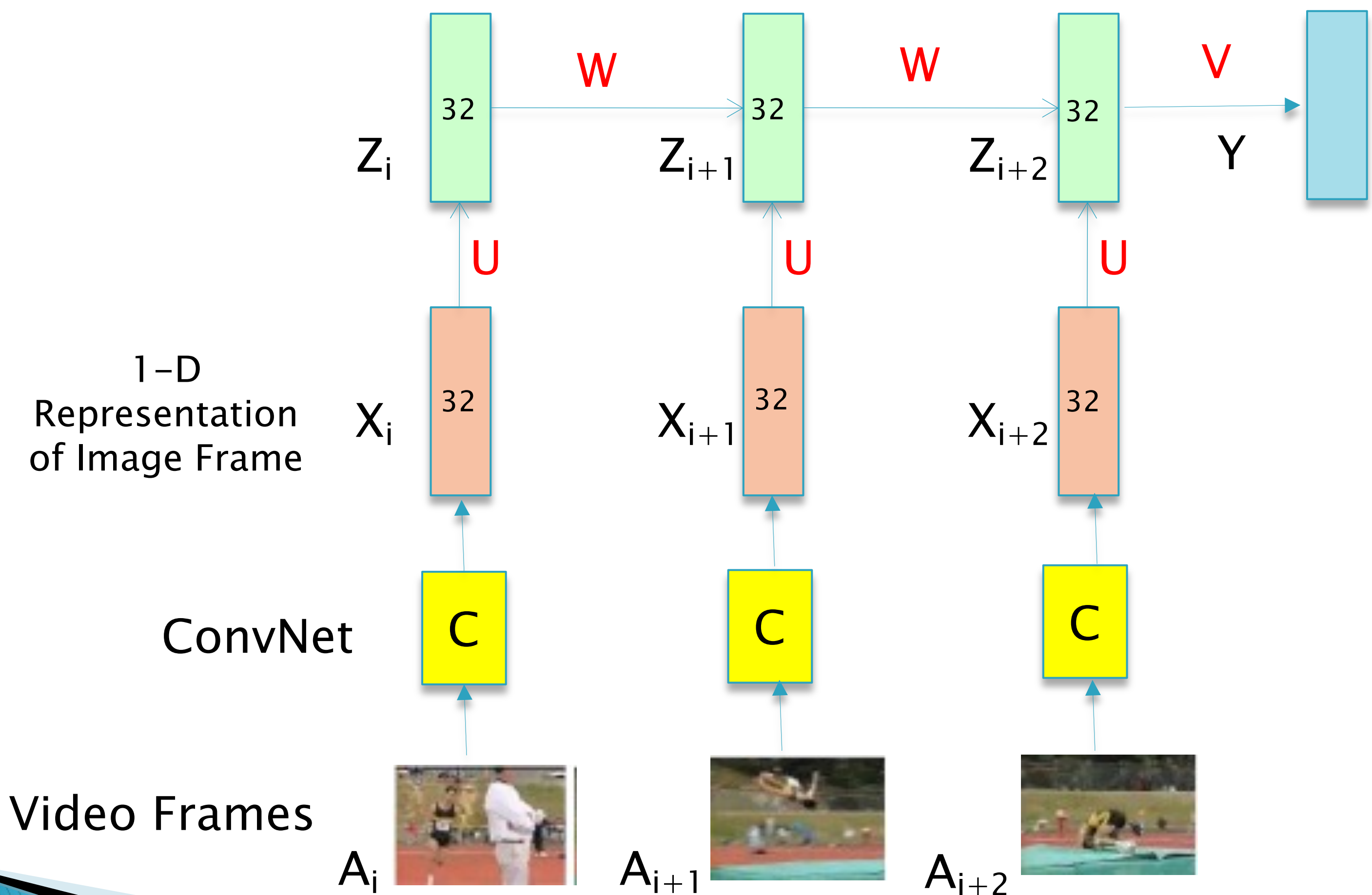
Loading Data: The General Case

- In general: The RNN is fed with input data of shape
  (# samples, time, features)



What about higher dimensional sequences such as video?
Each video clip has shape (time, height, width, depth)

Example: Video

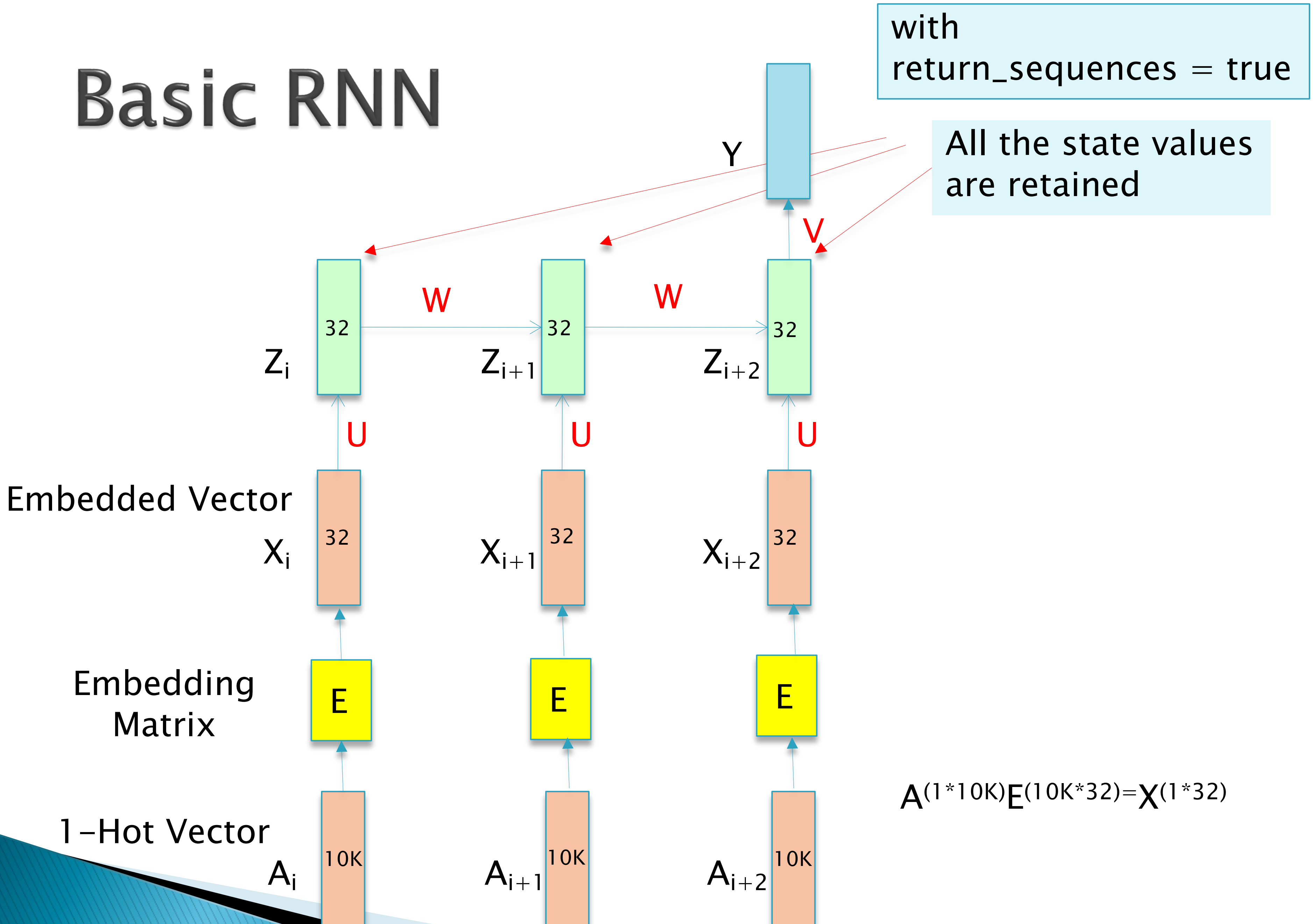


Basic RNN (for NLP)

```
from keras.models import Sequential
from keras.layers import Embedding, SimpleRNN

model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 32)          320000
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, 32)                2080
=================================================================
Total params: 322,080
Trainable params: 322,080
Non-trainable params: 0
_________________________________________________________________
```

Basic RNN



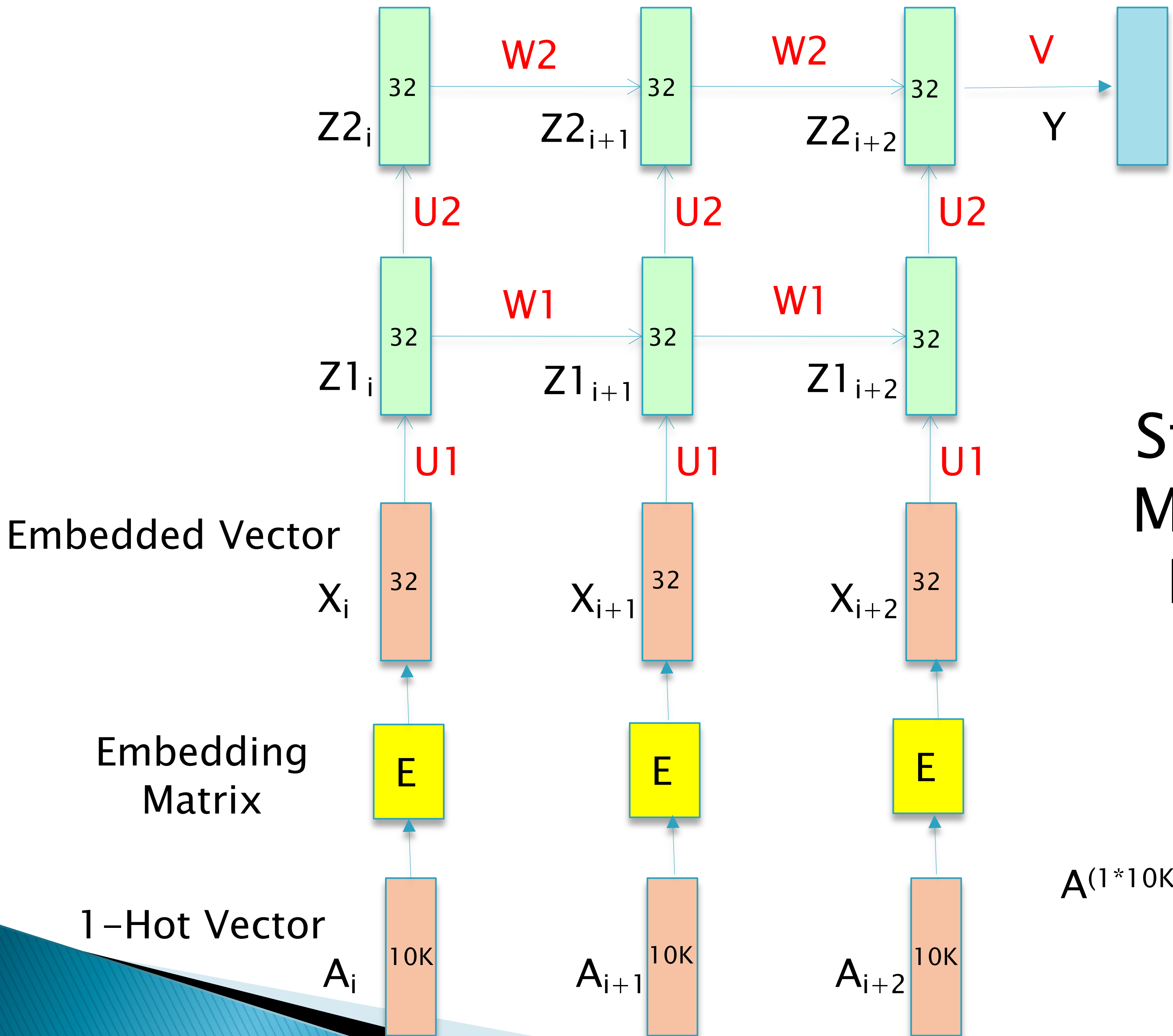
$A^{(1*10K)}E^{(10K*32)}=X^{(1*32)}$

Basic RNN

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 32)          320000
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, None, 32)          2080
=================================================================
Total params: 322,080
Trainable params: 322,080
Non-trainable params: 0
_________________________________________________________________
```

Stacking Multiple Layers



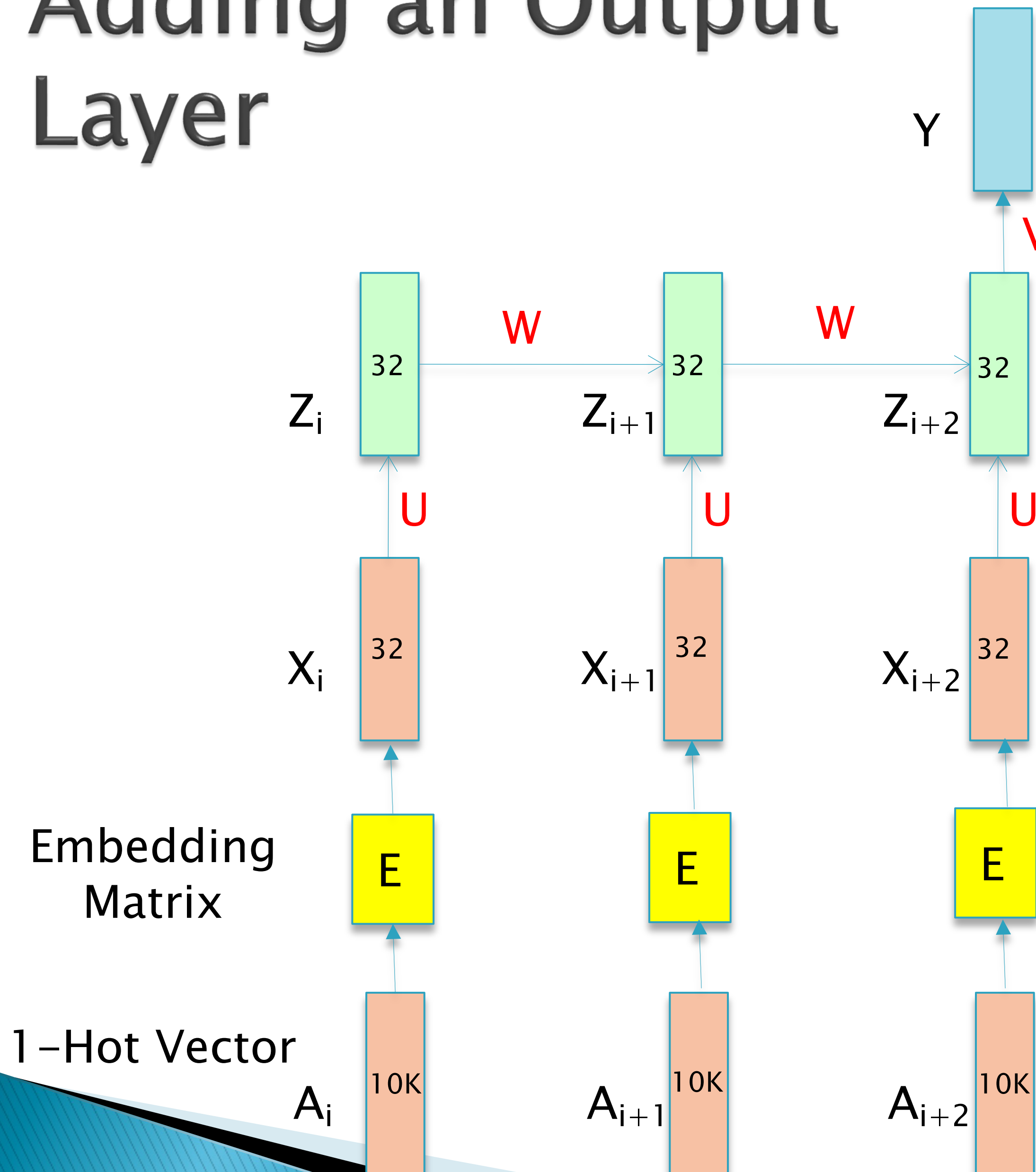
$A^{(1*10K)}E^{(10K*32)}=X^{(1*32)}$

Stacking Multiple Layers

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))  # This last layer only returns the last outputs.
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, None, 32)          320000
_________________________________________________________________
simple_rnn_3 (SimpleRNN)     (None, None, 32)          2080
_________________________________________________________________
simple_rnn_4 (SimpleRNN)     (None, None, 32)          2080
_________________________________________________________________
simple_rnn_5 (SimpleRNN)     (None, None, 32)          2080
_________________________________________________________________
simple_rnn_6 (SimpleRNN)     (None, 32)                2080
=================================================================
Total params: 328,320
Trainable params: 328,320
Non-trainable params: 0
_________________________________________________________________
```

Adding an Output Layer

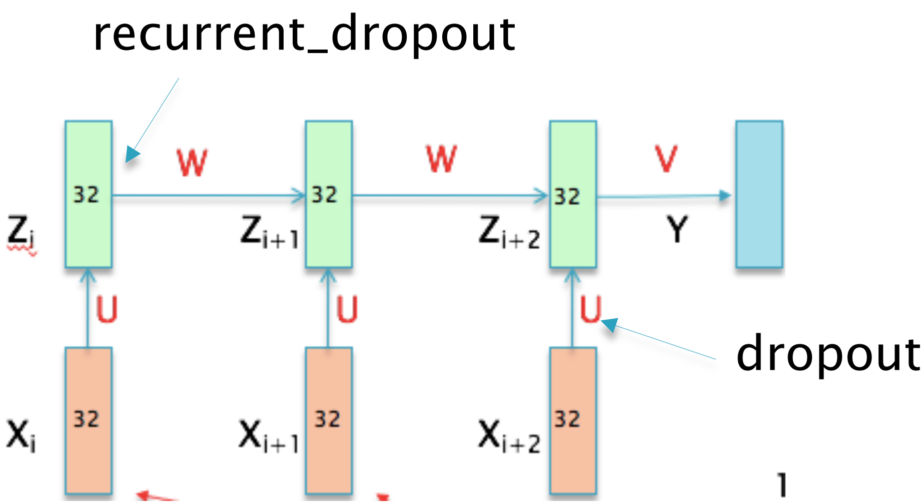


$A^{(1*10K)}E^{(10K*32)}=X^{(1*32)}$

Adding an Output Layer

```
from keras.layers import Dense

model = Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(input_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

Specifying Dropout for RNNs




```
model = Sequential()
model.add(layers.GRU(32,
                     dropout=0.2,
                     recurrent_dropout=0.2,
                     input_shape=(None, float_data.shape[-1])))
model.add(layers.Dense(1))

model.compile(optimizer=RMSprop(), loss='mae')
history = model.fit_generator(train_gen,
                              steps_per_epoch=500,
                              epochs=40,
                              validation_data=val_gen,
                              validation_steps=val_steps)
```

Further Reading

- Das and Varma: Chapter RNNs
- Chollet, Chapter 10