SAMSUNG

Together for Tomorrow!
Enabling People
Education for Future Generations

Samsung Innovation Campus

Artificial Intelligence Course

Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 9.

Deep Learning - Part II

AI Course

Samsung Innovation Campus

Samsung Internal Use Only

Update History

Version	Update Details
v1.0	• 1st draft completed
v1.01	• 'Guide for Instructors' page inserted
v1.02	• Front and Cover page layout updated • 'Coding Exercise' replaced 'Coding Practice' • 'Quiz.' replaced 'Coding Problem'
v1.03	• Samsung, SIC logo updated
V1.04	• Chapter info. updated

Together for Tomorrow!
Enabling People
Education for Future Generations

Chapter 9.

Deep Learning - Part II

UNIT 3.

Deep Learning with Keras

Samsung Innovation Campus

UNIT 3.
Deep Learning with Keras

What this unit is about:

- Building artificial intelligence models with the Keras library.
- Utilization of the DNN, CNN, RNN and LSTM networks.
- Embedding representation of words.
- Natural language processing with the Keras library.

Expected outcome:

- Proficiency in building artificial intelligence models.

How to check your progress:

- Coding Exercises.
- Quiz.

Chapter 9.

Deep Learning - Part II

Together for Tomorrow!
Enabling People
Education for Future Generations

UNIT 3. Deep Learning with Keras.

3.1. Keras Basics.

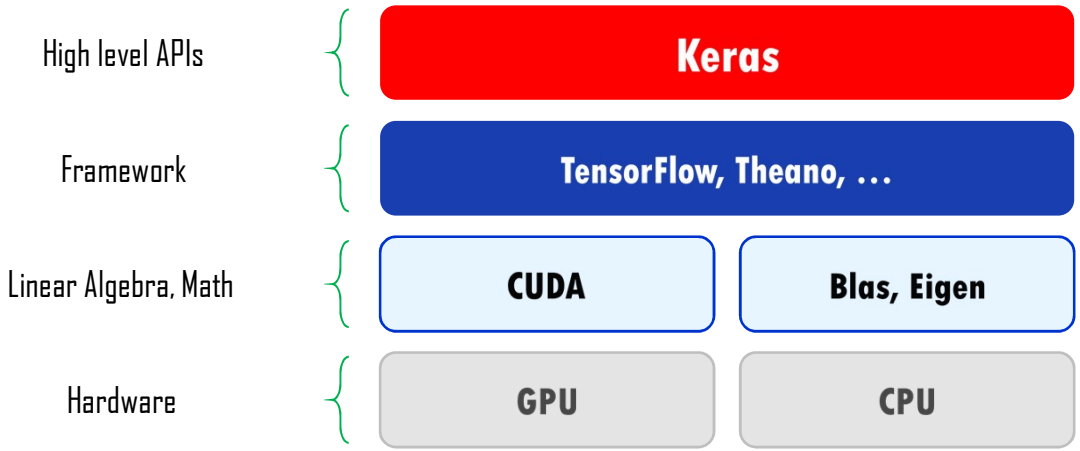
3.2. AI with Keras.

3.3. Natural Language Processing with Keras.

Keras Basics (1/9)

About the Keras library:

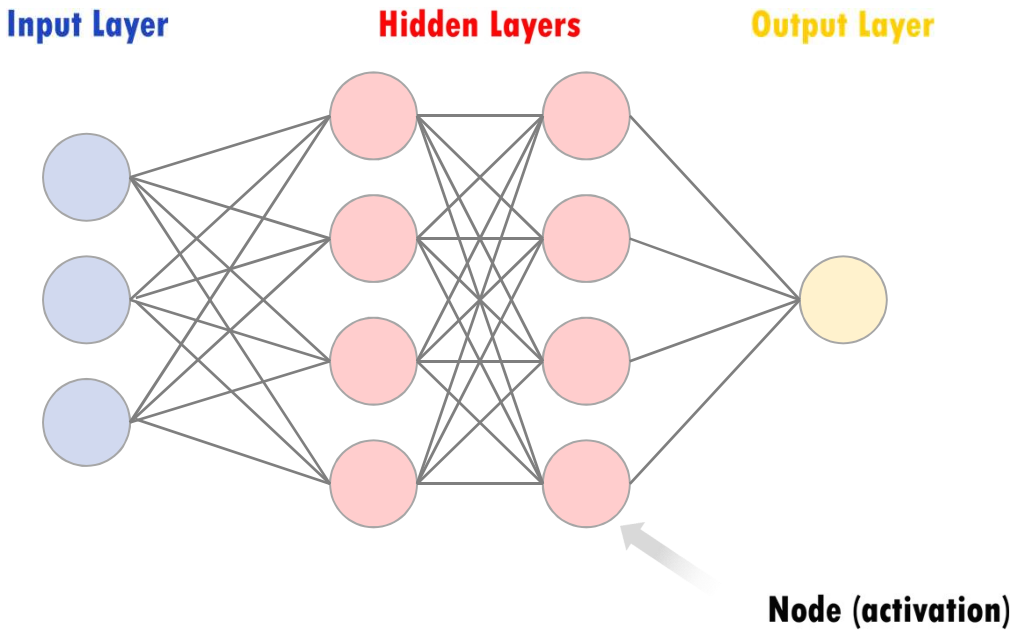
- A high-end deep learning library.
- Does not support the low-end tensor operations.
- Runs on top of TensorFlow.



Keras Basics (2/9)

About the Keras library:

- It is easy to model deep neural networks with Keras.



Keras Basics (3/9)

Training work flow:

Define a model's layer structure (specify the activation functions).

Define the loss function and the optimization method.

Compile the model.

Specify the batch learning method and start the training (gradient descent).

Keras Basics (4/9)

Loss (cost) functions in Keras:

Name	Formula	Keras Expression
Squared Error (L2 Error)	$L = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$	loss = "mse"
Absolute Error (L1 Error)	$L = \sum_{i=1}^{n} \lvert y_i - \hat{y}_i \rvert$	loss = "mae"
Binary Cross Entropy	$L = -\frac{1}{n} \sum_{i=1}^{n} (y_i log(\hat{p}_i) + (1 - y_i)\ log(1 - \hat{p}_i))$	loss = "binary_crossentropy"
Multi-Class Cross Entropy	$L = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik}\ log(\hat{p}_{ik})$	loss = "categorical_crossentropy"

- Cross entropy is the appropriate loss function for the classification.

Keras Basics (5/9)

Gradient descent algorithms:

Algorithm	Explanation	Keras
Stochastic Gradient Descent	Gradient descent algorithm with batch learning.	SGD(lr=0.1)
Momentum	Takes into account the gradient from the previous step.	SGD(momentum=0.9)
Nesterov Momentum	Minimizes unnecessary movements.	SGD(lr=0.01, momentum=0.9, nesterov=True)
Adagrad	Adaptative step size.	Adagrad(lr=0.01)
RMSProp	Improves upon the Adagrad.	RMSProp(lr=0.01)
Adam	Combines the best properties of Adagrad and RMSProp.	Adam(lr=0.01)

Keras Basics (6/9)

Activation functions:

Name	Formula	Keras
Linear	$linear(x) = x$	activation="linear"
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	activation="sigmoid"
Tanh	$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	activation="tanh"
ReLu	$ReLu(x) = \max(0, x)$	activation="relu"
Softmax	$\sigma(\boldsymbol{x})_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$	activation="softmax"

Keras Basics (7/9)

Layers:

Keras	Explanation
Dense()	Dense layer.
Conv1D()	1D convolution layer.
Conv2D()	2D convolution layer.
MaxPooling1D()	1D max pooling layer.
MaxPooling2D()	2D max pooling layer.
Dropout()	Applies dropout to the input.
Flatten()	Flattens the input.
Embedding()	Turns positive integers (label encoding) into dense vectors.
SimpleRNN()	RNN where the output is fed back to input.
LSTM()	LSTM layer.
TimeDistributed()	A wrapper that applies a layer to every time slice.
Input()	The first layer for a functional API model.

Keras Basics (8/9)

Two ways of building models with Keras:

1). Sequential API:

- Layers are added to a "Sequential" object.
- Easy but less flexible.

2). Functional API:

- The first layer must be defined as "Input" where the data shape is specified.
- Each layer defined as a function that takes in as argument the previous layer.
- A "Model" object takes in as arguments the first and the last layers.
- More flexible.

Keras Basics (9/9)

Datasets provided by Keras:

- Useful for learning/practicing purpose.
- Provided by the "keras.datasets" module.

Name	Description
cifar10	32×32 color images labeled over 10 categories. 50000 training and 10000 testing images.
cifar100	32×32 color images labeled over 100 categories. 50000 training and 10000 testing images.
imdb	25000 movie reviews from IMDB with binary labeling (positive or negative).
reuters	11228 newswires from Reuters, labeled over 46 topics.
mnist	28×28 grayscale images of handwritten digits. 60000 training and 10000 testing images.
fashion_mnist	28×28 grayscale images of 10 fashion categories. 60000 training and 10000 testing images.
boston_housing	13 attributes of houses at different locations around Boston suburbs in the late 1970s.

Deep Learning - Part II

UNIT 3. Deep Learning with Keras.

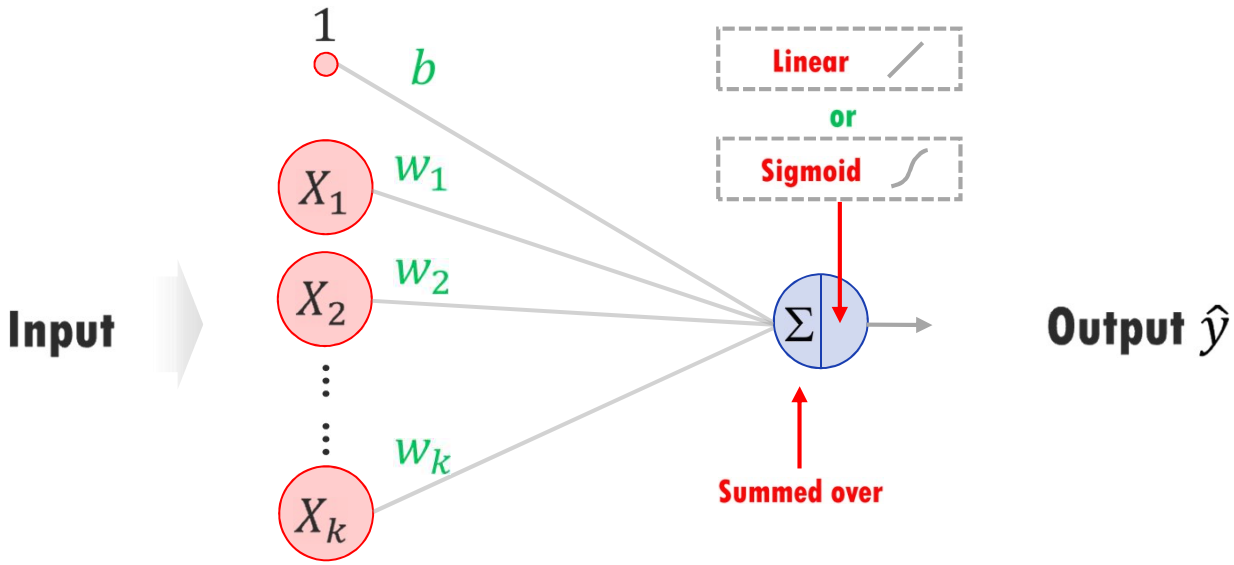
3.1. Keras Basics.

3.2. AI with Keras.

3.3. Natural Language Processing with Keras.

AI with Keras (1/8)

Linear/Logistic regression with Keras:



- The major difference between linear and logistic regression lies in the choice of activation function.

AI with Keras (2/8)

Linear/Logistic regression with Keras:

- Sequential API example:

```
# Import the necessary classes.
from keras.models import Sequential
from keras.layers import Dense
```

```
# Build the model by adding a layer.
my_model = Sequential()
my_model.add(Dense(input_dim = n_vars, units = 1, activation="linear"))       # activation="linear" for linear regression.
```

```
# Define the optimization method and the loss function. Then, compile.
my_optimizer=Adam(lr=learn_rate)
my_model.compile(loss = "mse", optimizer = my_optimizer, metrics=["mse"])
```

```
# Train.
my_model.fit(X_train, y_train, epochs=n_epochs, batch_size = n_batch_size, validation_split = 0.2, verbose = 0)
```

AI with Keras (3/8)

Linear/Logistic regression with Keras:

- Functional API example:

```
# Import the necessary classes.
from keras.models import Model
from keras.layers import Input, Dense
```

```
# Build the model by adding a layer.
my_input = Input(shape=(n_vars,))                    # The first layer with the shape specified.
my_output = Dense(units=1,activation='linear')(my_input)   # Takes in the previous layer as function argument.
my_model = Model(inputs=my_input, outputs=my_output)   #  The first and the last layers entered.
```

- Optimization, compilation, training, etc. steps are the same as before.

Coding Exercise #0801

Follow practice steps on 'ex_0801.ipynb' file.

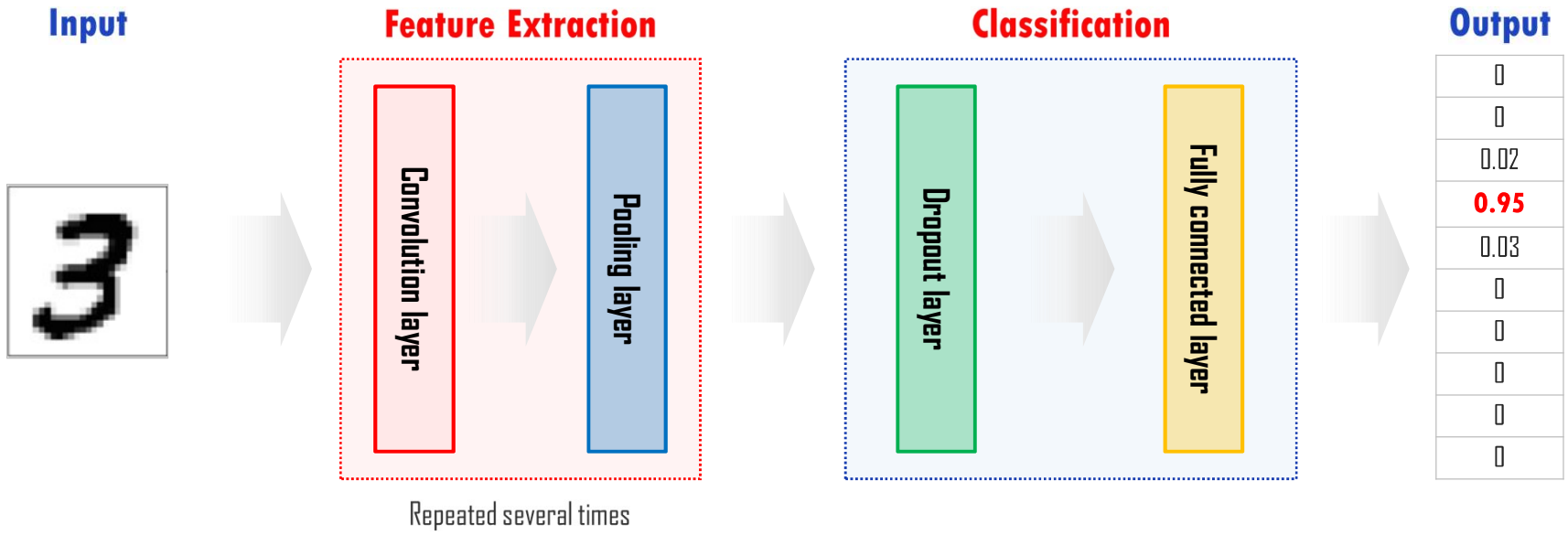
Coding Exercise #0802

Follow practice steps on 'ex_0802.ipynb' file.

AI with Keras (4/8)

Diagram of a Convolutional Neural Network (CNN):

- The convolution and pooling layers are repeated several times.
- There is a fully connected layer just before the output.



AI with Keras (5/8)

CNN with Keras: a code example.

```
# Import the necessary classes.
from keras.models import Sequential                                    # We will use the Sequential API.
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
```

```
# Build the CNN model by adding the layers.
hold_prob = 0.5
my_model = Sequential()
my_model.add(Conv2D(input_shape=(28,28,1),filters=32,kernel_size=(6,6), activation="relu"))
my_model.add(MaxPooling2D(pool_size=2,padding='same'))
my_model.add(Conv2D(filters=64,kernel_size=(6,6), activation="relu"))
my_model.add(MaxPooling2D(pool_size=2,padding='same'))
my_model.add(Flatten())
my_model.add(Dense(units = 1024, activation="relu"))
my_model.add(Dropout(rate=hold_prob))
my_model.add(Dense(units = 10, activation="softmax"))
```

Coding Exercise #0803

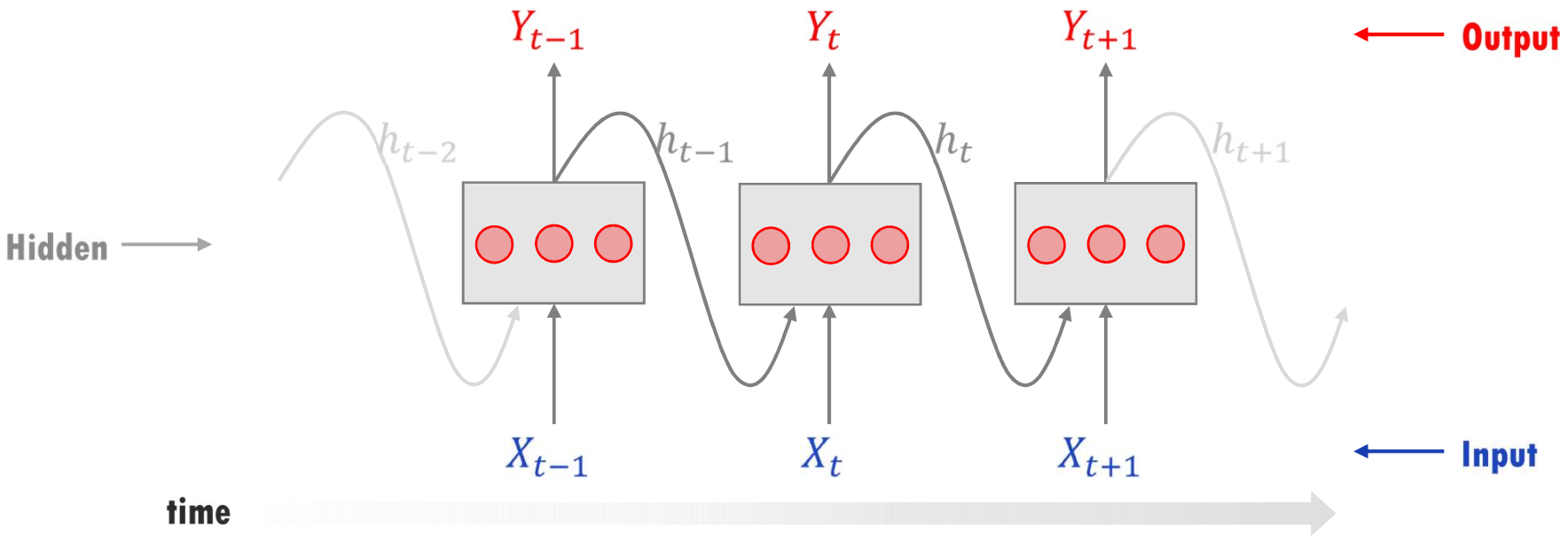
Follow practice steps on 'ex_0803.ipynb' file.

Coding Exercise #0804

Follow practice steps on 'ex_0804.ipynb' file.

AI with Keras (6/8)

Recurrent Neural Network (RNN):

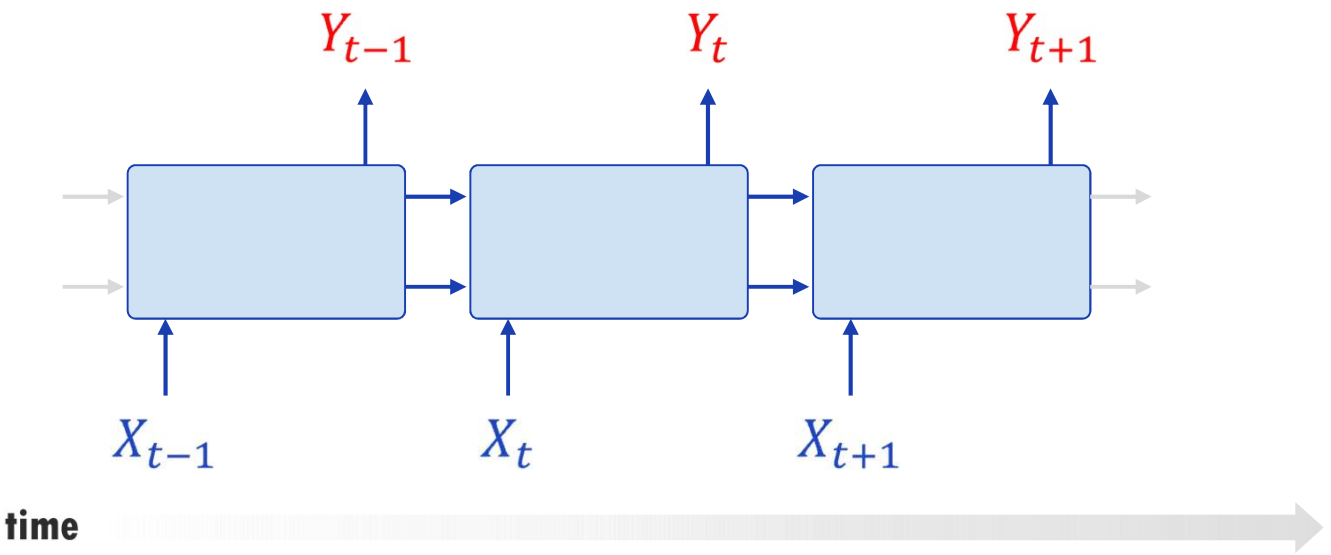


- This is a "Sequence in and Sequence out" model.

AI with Keras (7/8)

Long Short Term Memory (LSTM) network:

- LSTM cells can form a sequence just like the regular RNN cells.
- The following is a "Sequence in and Sequence out" model.

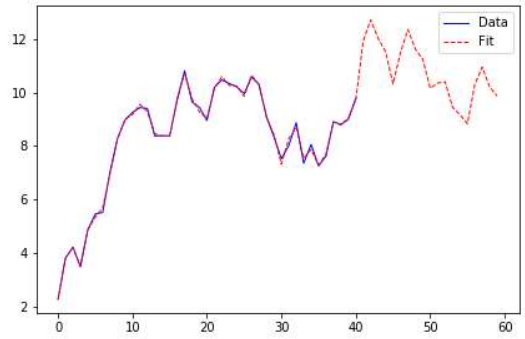


AI with Keras (8/8)

RNN/LSTM with Keras: a code example.

```
# Import the necessary classes.
from keras.models import Sequential                                    # We will use the Sequential API.
from keras.layers import SimpleRNN, LSTM, Dense, TimeDistributed
```

```
# "sequence in and sequence out" model : at SimpleRNN() return_sequences=True.
my_model = Sequential()
my_model.add(SimpleRNN(units=n_neurons, return_sequences=True, input_shape=(None, n_input)))    # RNN.
my_model.add(TimeDistributed(Dense(units=n_output, activation="linear")))                        # Wrapped.
```



Coding Exercise #0805

Follow practice steps on 'ex_0805.ipynb' file.

Chapter 9.

Deep Learning - Part II

Together for Tomorrow!
Enabling People
Education for Future Generations

UNIT 3. Deep Learning with Keras.

3.1. Keras Basics.

3.2. AI with Keras.

3.3. Natural Language Processing with Keras.

Natural Language Processing with Keras (1/16)

One-hot-encoding vs word embedding:

Ex). Given a sentence "I eat an apple every morning", let's suppose that the words are indexed as:

I : 3
Eat : 0
An : 2
Apple : 1
Every : 4
Morning : 5

The words would have the following one-hot-encoding representations:

I : [0 0 0 1 0 0]
Eat : [1 0 0 0 0 0]
An : [0 0 1 0 0 0]
Apple : [0 1 0 0 0 0]
Every : [0 0 0 0 1 0]
Morning : [0 0 0 0 0 1]

Natural Language Processing with Keras (2/16)

One-hot-encoding vs word embedding:

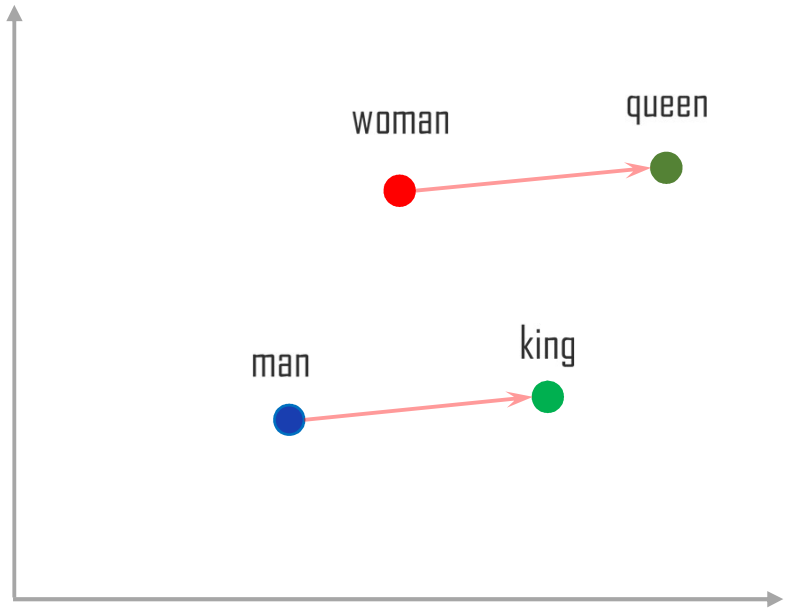
- We notice obvious problems with the one-hot-encoding representation.
- To improve, the "word embedding" is introduced which is a distributed representation method.

One-Hot-Encoding	Embedding
The dimension of the vector space is large. The dimension is as large as the vocabulary size.	The dimension of the vector space is limited.
Vectors are sparse; they are mostly filled with 0s that carry no information.	Vectors are dense. Every vector element carries some information.
No semantic relationship among the vectors. The vectors are orthogonal to each other.	Semantic relationship among the vectors.

- There are also "paragraph embedding" and "document embedding" representations.
- We will call "dense vector" or "embedding vector" interchangeably.

Natural Language Processing with Keras (3/16)

- Word embedding (Word2Vec):



  - Among the dense vectors, relationships such as following are established:

$$queen - woman = king - man$$

Natural Language Processing with Keras (4/16)

Word embedding (Word2Vec):

- Use CBOW (Continuous Bag of Words) and/or Skip-Gram to build the embedding vectors.

  1). Build a predictive model based on the Softmax regression (multi-class logistic regression).

  2). We assume one-hot-encoded input and output vectors.

  3). Extract the embedding vectors from the trained weight matrices.

Natural Language Processing with Keras (5/16)

Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Training Sentence	Center Word	Context Words
I eat an apple every morning	I	eat
I eat an apple every morning	eat	I, an
I eat an apple every morning	an	eat, apple
I eat an apple every morning	apple	an, every
I eat an apple every morning	every	apple, morning
I eat an apple every morning	morning	every

- We assumed a "sliding window" over the training sentence.

Natural Language Processing with Keras (6/16)

Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Ex). Let's suppose that the vector dimension of the one-hot-encoded words = 6.

Let's also suppose that we would like to find dense vectors of dimension = 3.

We will consider two context words (one from the left and another from the right).

So, we have a situation as the following:

I eat an apple every morning

I eat an ? ____ every morning

Natural Language Processing with Keras (7/16)

Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Ex). (continues from the previous page)

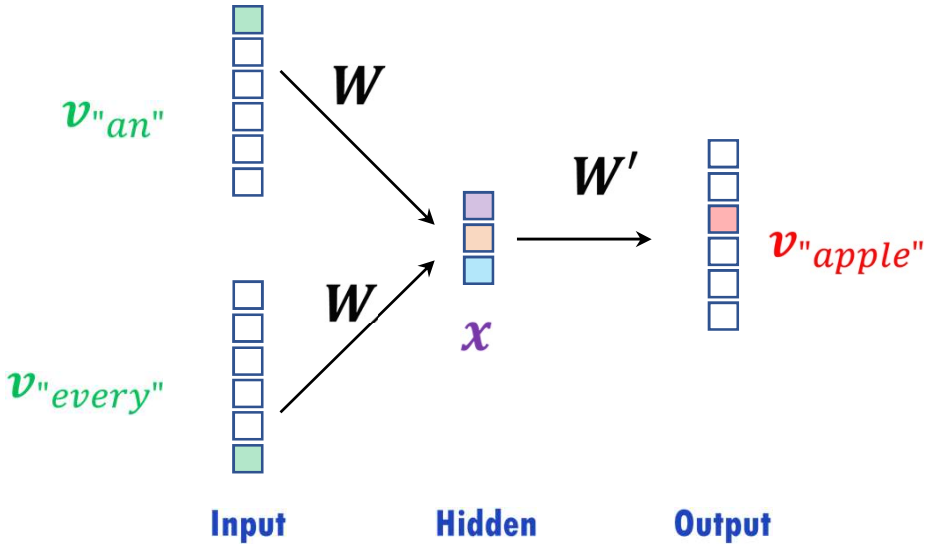
Then, we build a Softmax regression model.

For the vector inputs of "an" and "every",

we would like to train the weights W

and W' such that the predicted output is

the vector "apple".



One-hot-encoded words : $v_{"an"}$ =[1 0 0 0 0 0] , $v_{"every"}$ =[0 0 0 0 0 1] , $v_{"apple"}$ =[0 0 1 0 0 0]

Natural Language Processing with Keras (8/16)

Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Ex). (continues from the previous page)

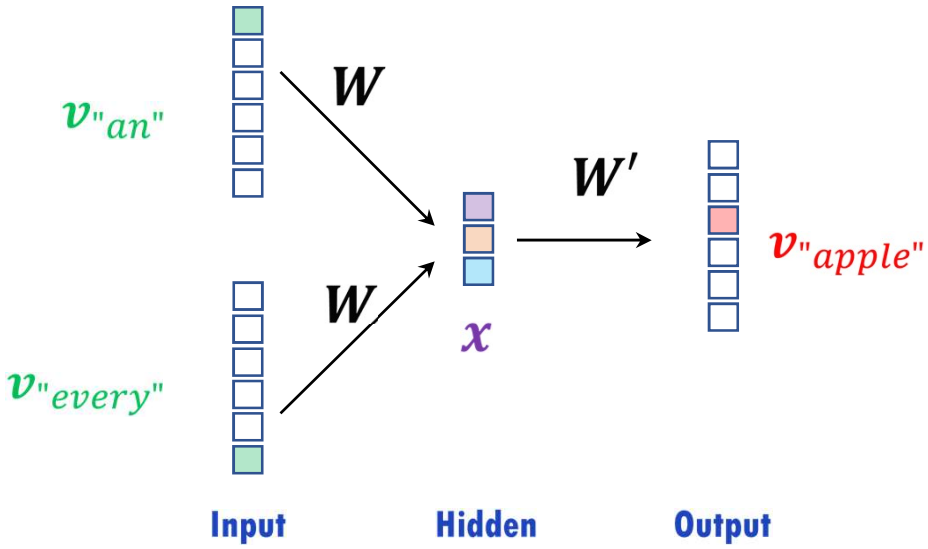
We have the following sizes:

Size of the matrix $\boldsymbol{W}$=3×6

Size of the matrix $\boldsymbol{W}'$=6×3

Dimension of the vector $\boldsymbol{x}$= 3.

Dimension of the input and output = 6.



Natural Language Processing with Keras (9/16)

Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Ex). (continues from the previous page)

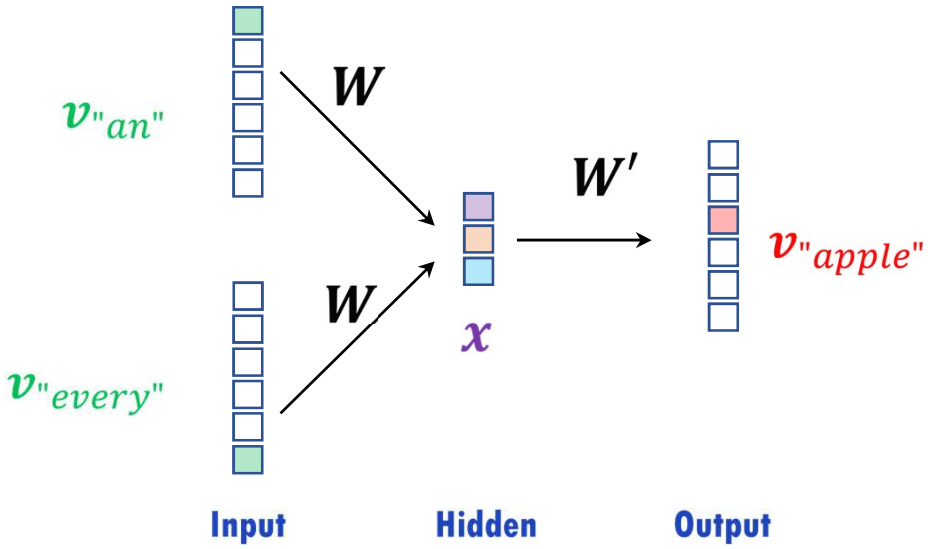
We propagate forward from the input layer

to the hidden layer (a single node):

$$x_{\text{"an"}} = W \cdot v_{"an"}$$

$$x_{\text{"every"}} = W \cdot v_{"every"}$$

$$x = \frac{x_{\text{"an"}} + x_{\text{"every"}}}{2}$$ ⇐ Average for the hidden node.



Natural Language Processing with Keras (10/16)

Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

Ex). (continues from the previous page)

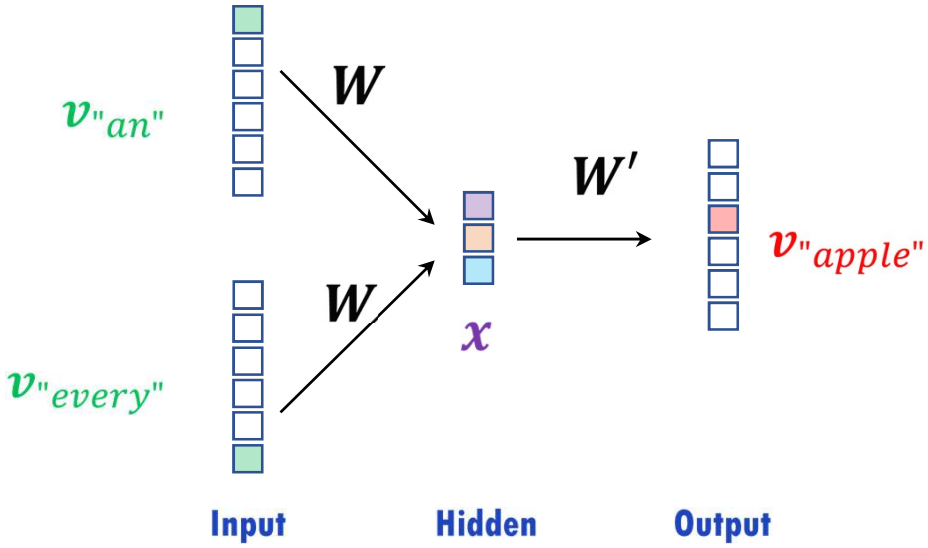
We propagate forward to the output layer:

$\hat{v} = W' \cdot x$

We should train the weights W and W'

argmax($\hat{v}$) and argmax($v_{"apple"}$)

is minimized.



Natural Language Processing with Keras (11/16)

Word embedding (Word2Vec):

- CBOW: Using the context words, predict the (missing) center word.

  Ex). (continues from the previous page)

  Now, let's interpret the result.

  a). When we propagate from the input layer to the hidden layer (by matrix multiplication),

  the one-hot-encoded input vectors $\boldsymbol{v}_{"an"}$ and $\boldsymbol{v}_{"every"}$ are picking the columns 0 and 5

  of $\boldsymbol{W}$ and projecting them to the hidden layer with the target dimension = 3.

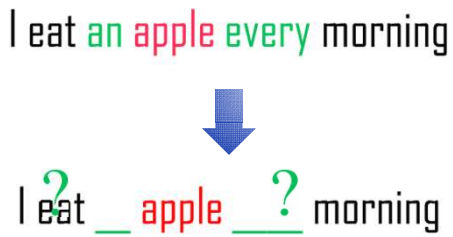
  b). So, the dense vectors for "an", "every" are the **columns** 0 and 5 of the **trained** $\boldsymbol{W}$.

  c). Analogously, we can extract dense vectors from the **rows** of the **trained** $\boldsymbol{W}'$.

Natural Language Processing with Keras (12/16)

Word embedding (Word2Vec):

- Skip-Gram: Using a center word, predict the (missing) context words.

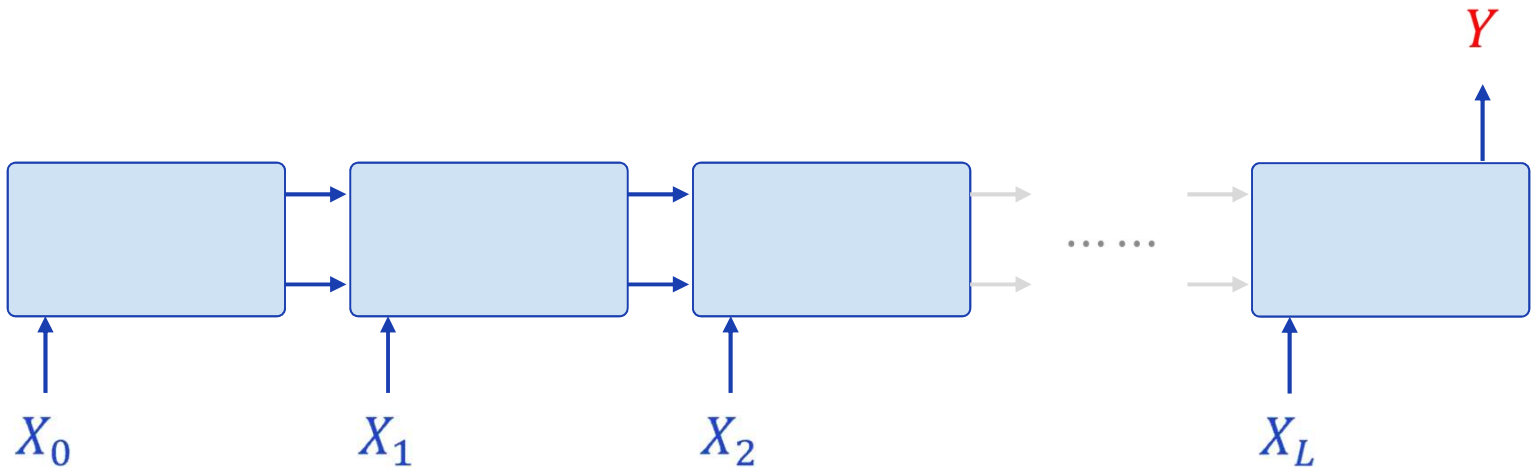


- Similar to the CBOW, here also we train a Softmax regression to predict the missing words.
- We extract the dense vectors (embedding vectors) from the trained weight matrices.

Natural Language Processing with Keras (13/16)

LSTM network for document classification:

- "Sequence in and Vector out" model.
- Embedding representation of the words.



Natural Language Processing with Keras (14/16)

LSTM network for document classification: a code example.

```
# Import the necessary classes.
from keras.models import Sequential                    # We will use the Sequential API.
from keras.layers import Dense, LSTM, Embedding
```

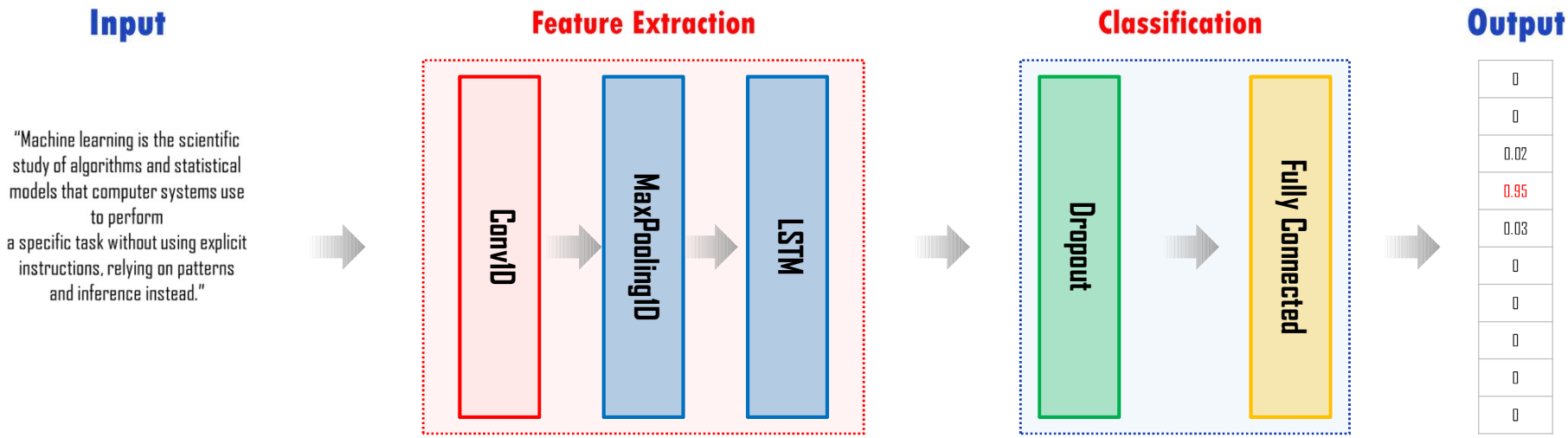
```
# Build a model by adding the layers.
my_model = Sequential()
my_model.add(Embedding(n_words,n_input))
my_model.add(LSTM(units=n_neurons, return_sequences=False, input_shape=(None, n_input), activation='tanh'))
my_model.add(Dense(1, activation='sigmoid'))
```

- In LSTM(), we should set *return_sequences=False* for a "Sequence in and Vector out" model.

Natural Language Processing with Keras (15/16)

Deep learning model for document classification:

- 1D convolution + 1D max pooling + LSTM for the feature extraction.
- Localized sequence patterns picked up by the 1D convolution.



Natural Language Processing with Keras (16/16)

Deep learning model for document classification: a code example.

```
# Import the necessary classes.
from keras.models import Sequential                                    # We will use the Sequential API.
from keras.layers import Dense, LSTM, Embedding, Conv1D, MaxPool1D, Dropout
```

```
# Build a model by adding the layers.
my_model = Sequential()
my_model.add(Embedding(n_words, n_input))                    # Embedding layer.
my_model.add(Conv1D(filters=n_filters, kernel_size = k_size, strides=stride_size,padding='valid',activation='relu'))
my_model.add(MaxPool1D(pool_size = 2))
my_model.add(LSTM(units=n_neurons, return_sequences=False, input_shape=(None, n_input), activation='tanh'))
my_model.add(Dropout(rate=hold_prob))
my_model.add(Dense(1, activation='sigmoid'))
```

Coding Exercise #0806

Follow practice steps on 'ex_0806.ipynb' file.

Coding Exercise #0807

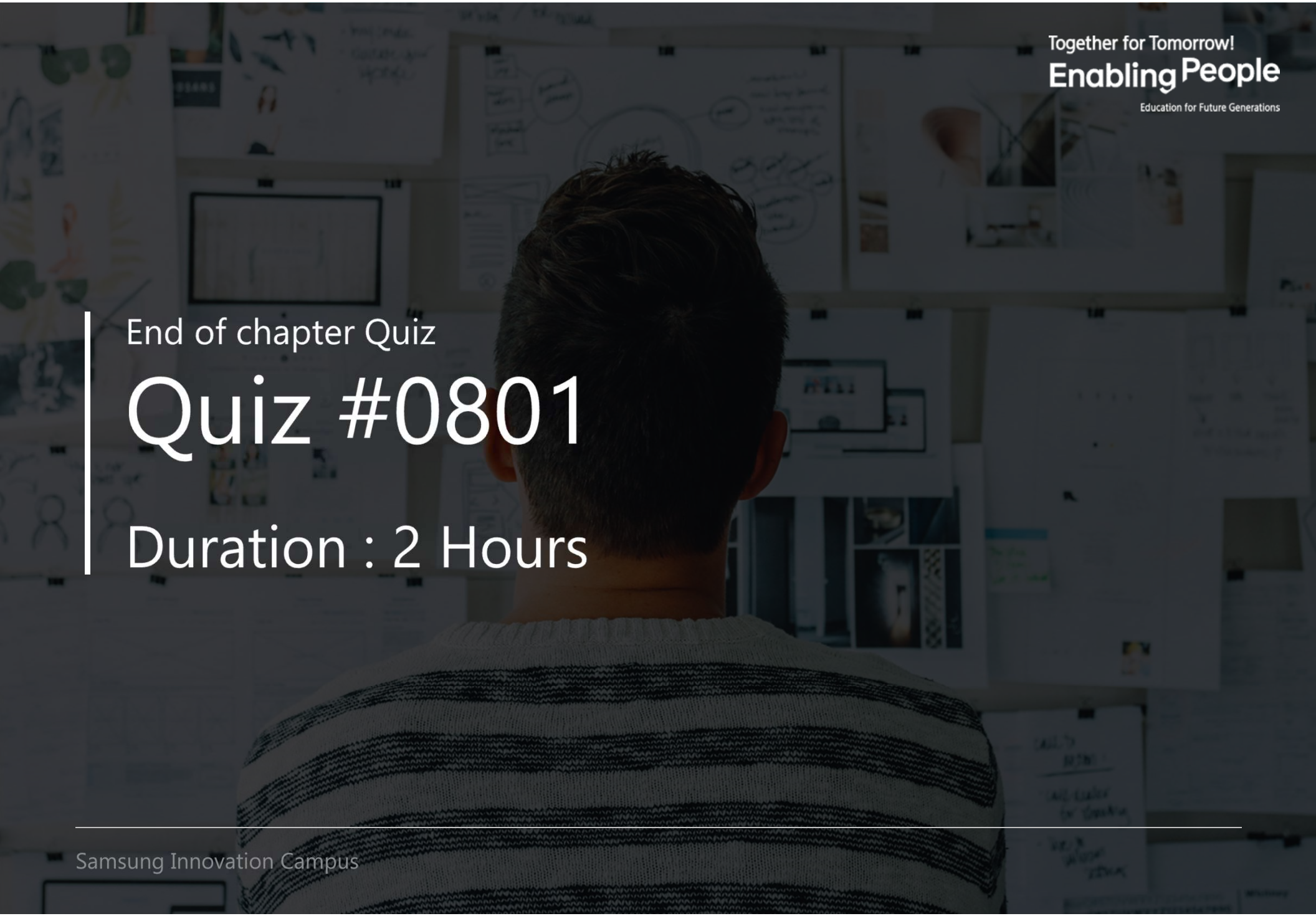
Follow practice steps on 'ex_0807.ipynb' file.

Coding Exercise #0808

Follow practice steps on 'ex_0808.ipynb' file.

Coding Exercise #0809

Follow practice steps on 'ex_0809.ipynb' file.

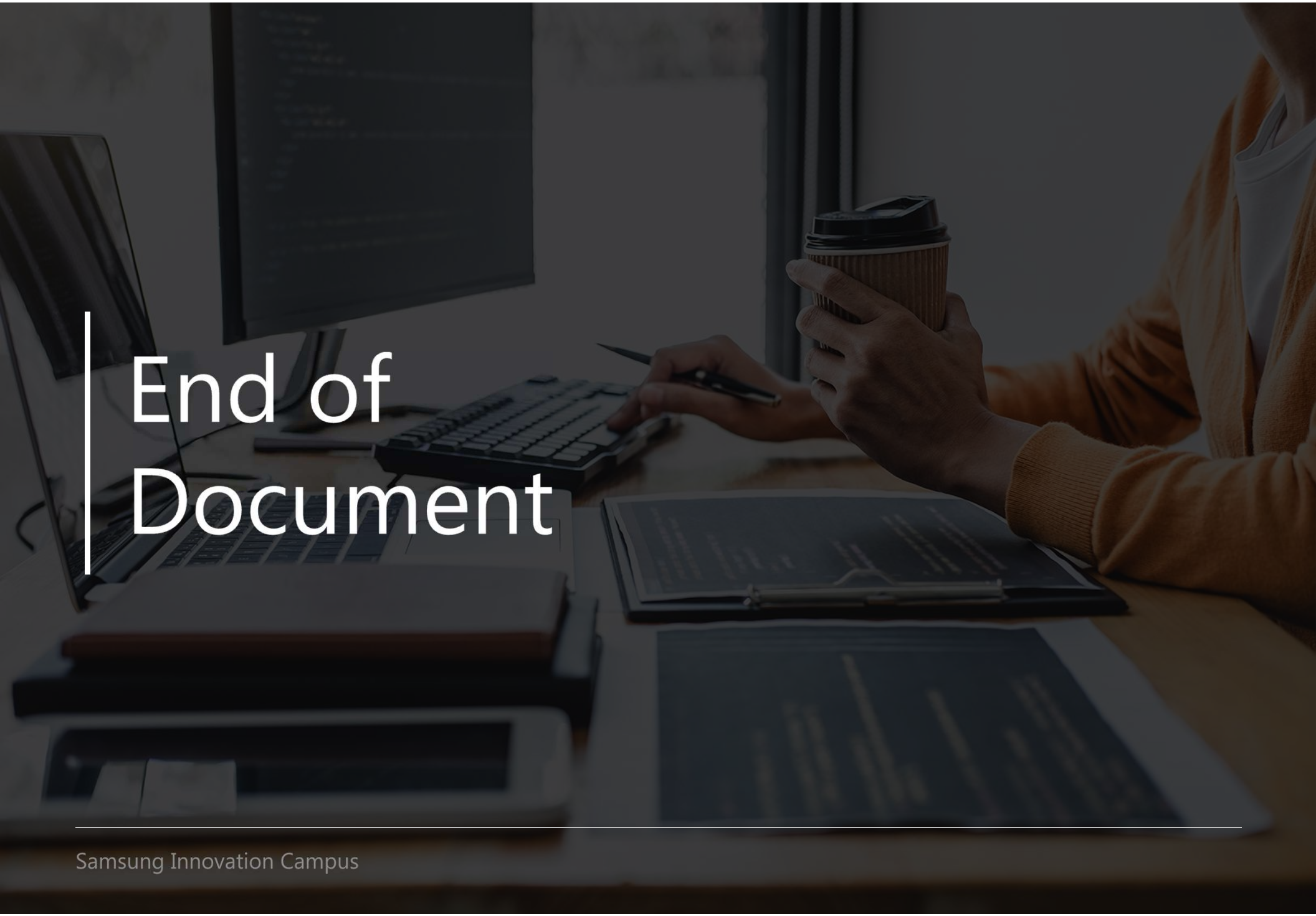


Together for Tomorrow!
Enabling People
Education for Future Generations

End of chapter Quiz

Quiz #0801

Duration : 2 Hours


End of
Document

Together for Tomorrow!

Enabling People

Education for Future Generations