Tensors, layers and autoencoders

DEEP LEARNING WITH KERAS IN PYTHON

Miguel Esteban
Data Scientist & Founder

Accessing Keras layers

```
# Acessing the first layer of a Keras model
first_layer = model.layers[0]

# Printing the layer, and its input, output and weights
print(first_layer.input)
print(first_layer.output)
print(first_layer.weights)
```

```
<tf.Tensor 'dense_1_input:0' shape=(?, 3) dtype=float32>
```

```
<tf.Tensor 'dense_1/Relu:0' shape=(?, 2) dtype=float32>
```

```
[<tf.Variable 'dense_1/kernel:0' shape=(3, 2) dtype=float32_ref>,
 <tf.Variable 'dense_1/bias:0' shape=(2,) dtype=float32_ref>]
```

What are tensors?

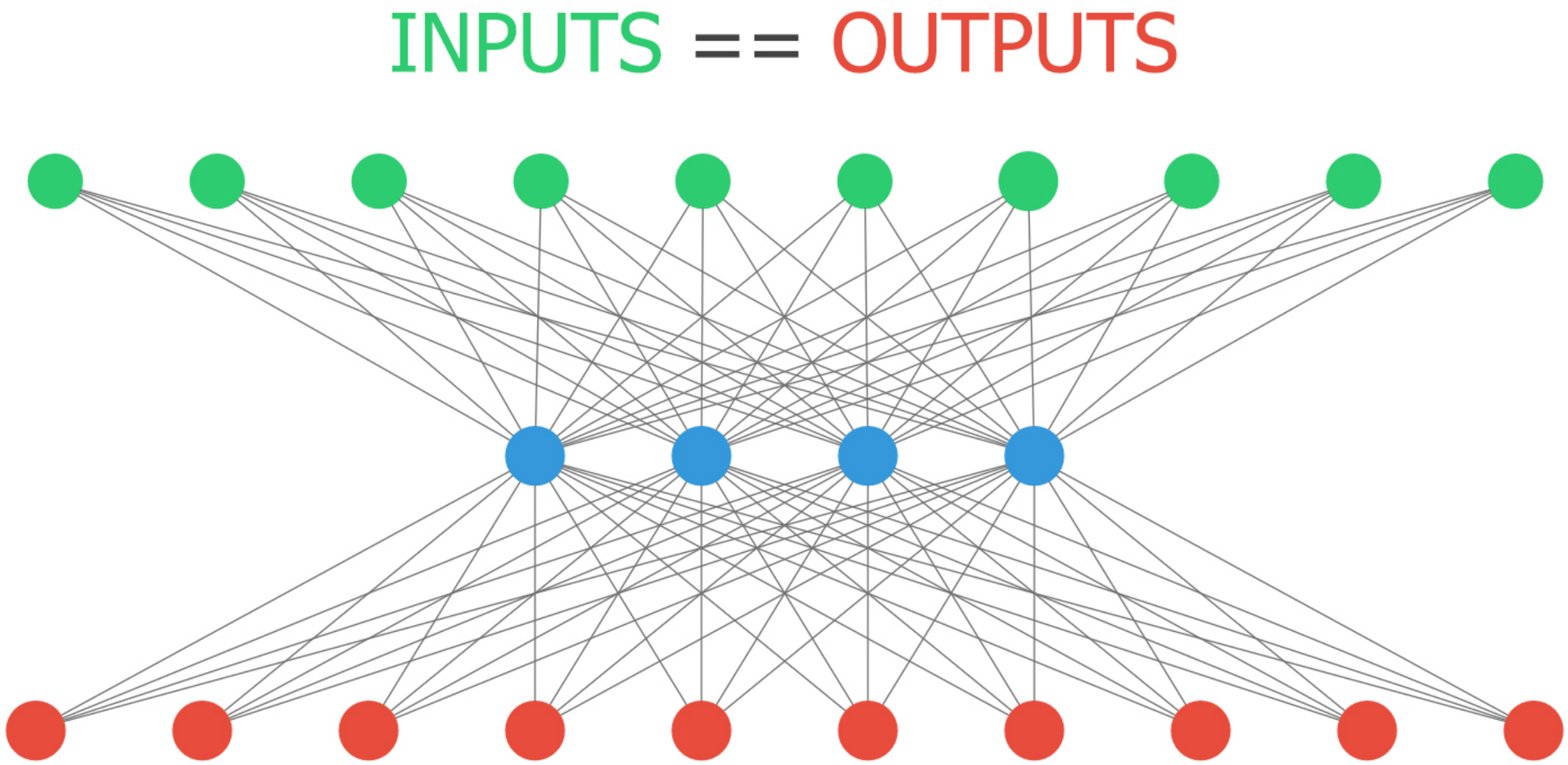
```
# Defining a rank 2 tensor (2 dimensions)
T2 = [[1,2,3],
      [4,5,6],
      [7,8,9]]
# Defining a rank 3 tensor (3 dimensions)
T3 = [[1,2,3],
      [4,5,6],
      [7,8,9],

      [10,11,12],
      [13,14,15],
      [16,17,18],

      [19,20,21],
      [22,23,24],
      [25,26,27]]
```

```
# Import Keras backend
import keras.backend as K

# Get the input and output tensors of a model layer
inp = model.layers[0].input
out = model.layers[0].output

# Function that maps layer inputs to outputs
inp_to_out = K.function([inp], [out])

# We pass and input and get the output we'd get in that first layer
print(inp_to_out([X_train])
```

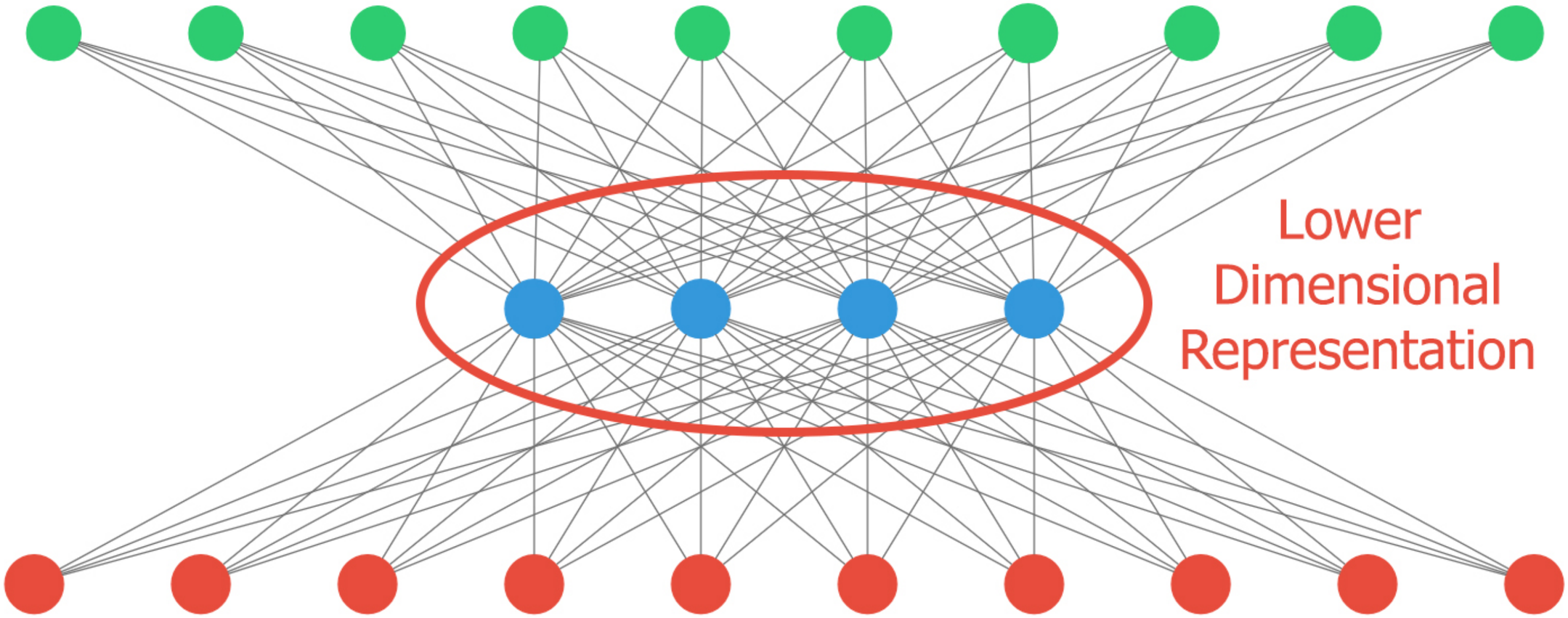
```
# Outputs of the first layer per sample in X_train
[array([[0.7, 0],...,[0.1, 0.3]])]
```

it's time to introduce a new architecture...

Autoencoders!



Autoencoders!



Autoencoder use cases

- Dimensionality reduction:
  - Smaller dimensional space representation of our inputs.
- De-noising data:
  - If trained with clean data, irrelevant noise will be filtered out during reconstruction.
- Anomaly detection:
  - A poor reconstruction will result when the model is fed with unseen inputs.
- ...

Building a simple autoencoder

```
# Instantiate a sequential model
autoencoder = Sequential()

# Add a hidden layer of 4 neurons and an input layer of 100
autoencoder.add(Dense(4, input_shape=(100,), activation='relu'))

# Add an output layer of 100 neurons
autoencoder.add(Dense(100, activation='sigmoid'))

# Compile your model with the appropiate loss
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

Breaking it into an encoder

```
# Building a separate model to encode inputs
encoder = Sequential()
encoder.add(autoencoder.layers[0])

# Predicting returns the four hidden layer neuron outputs
encoder.predict(X_test)
```

```
# Four numbers for each observation in X_test
array([10.0234375, 5.833543, 18.90444, 9.20348],...)
```

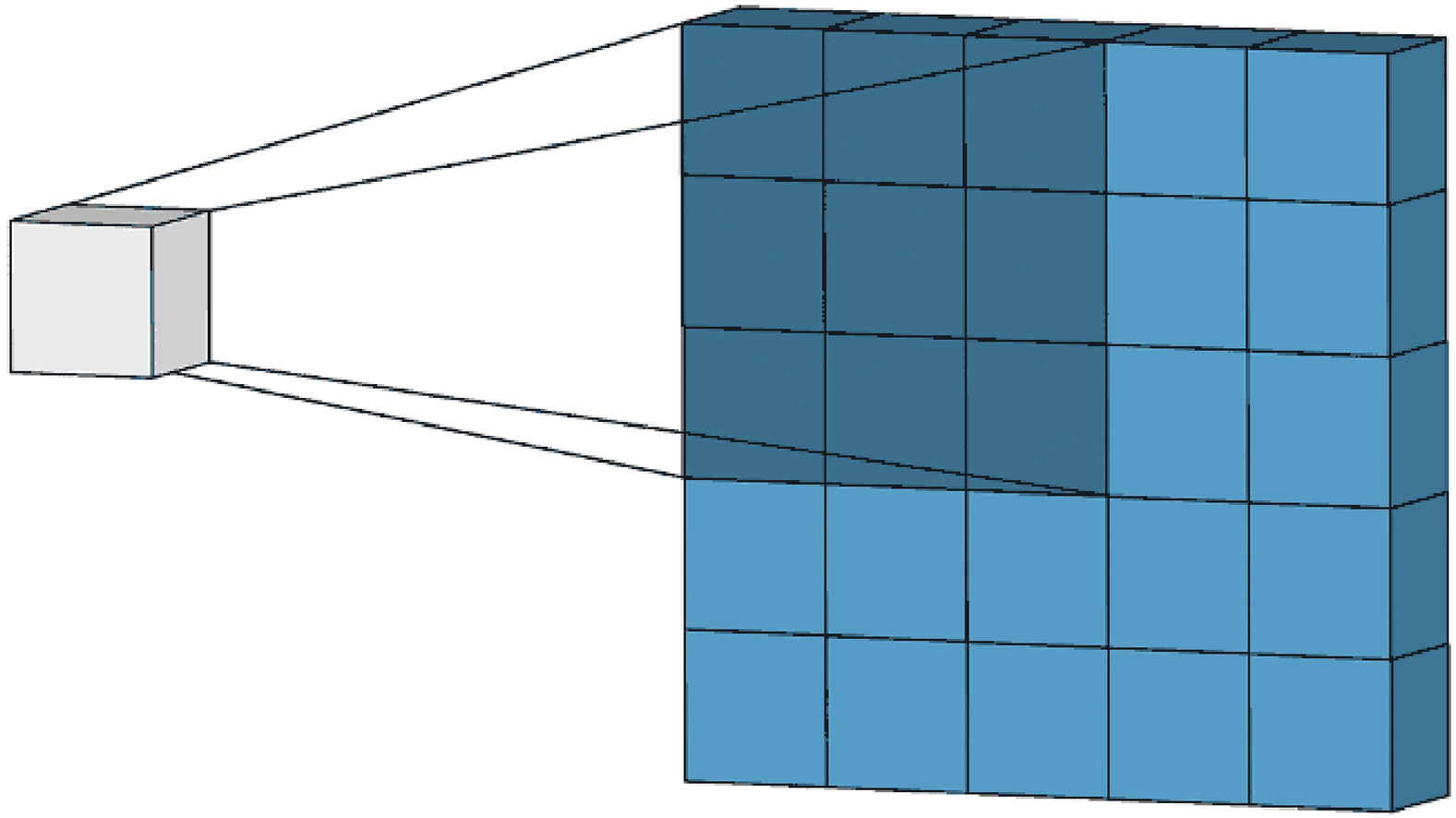
Let's experiment!

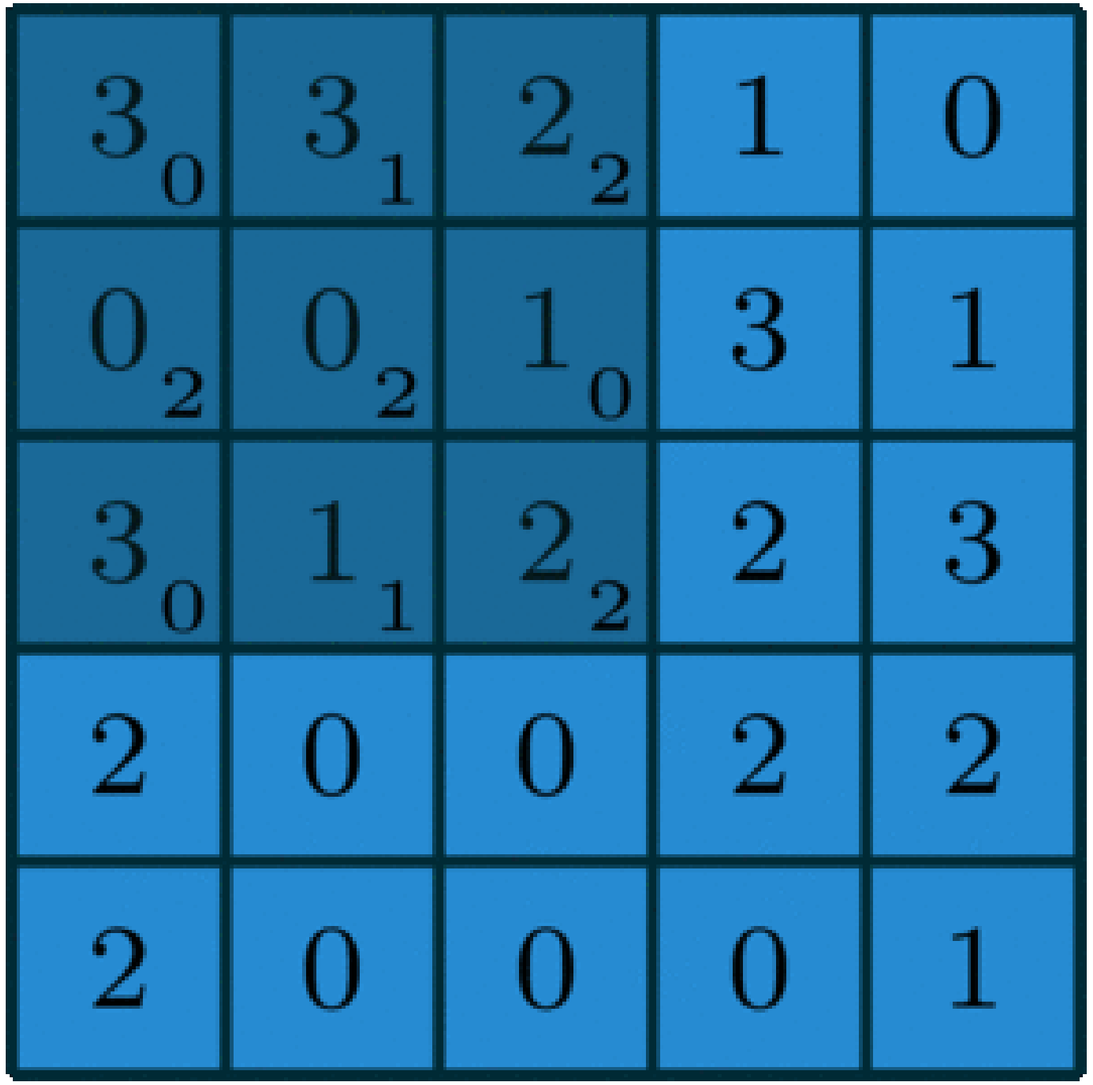
DEEP LEARNING WITH KERAS IN PYTHON

Intro to CNNs

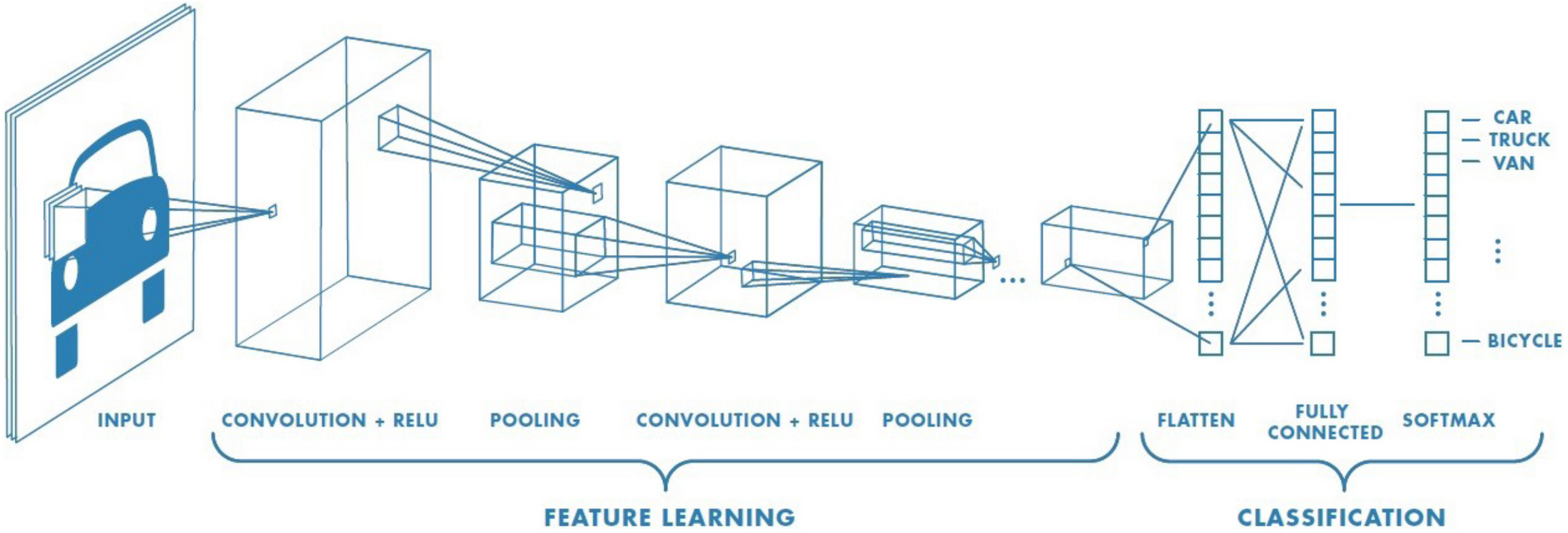
DEEP LEARNING WITH KERAS IN PYTHON

Miguel Esteban
Data Scientist & Founder





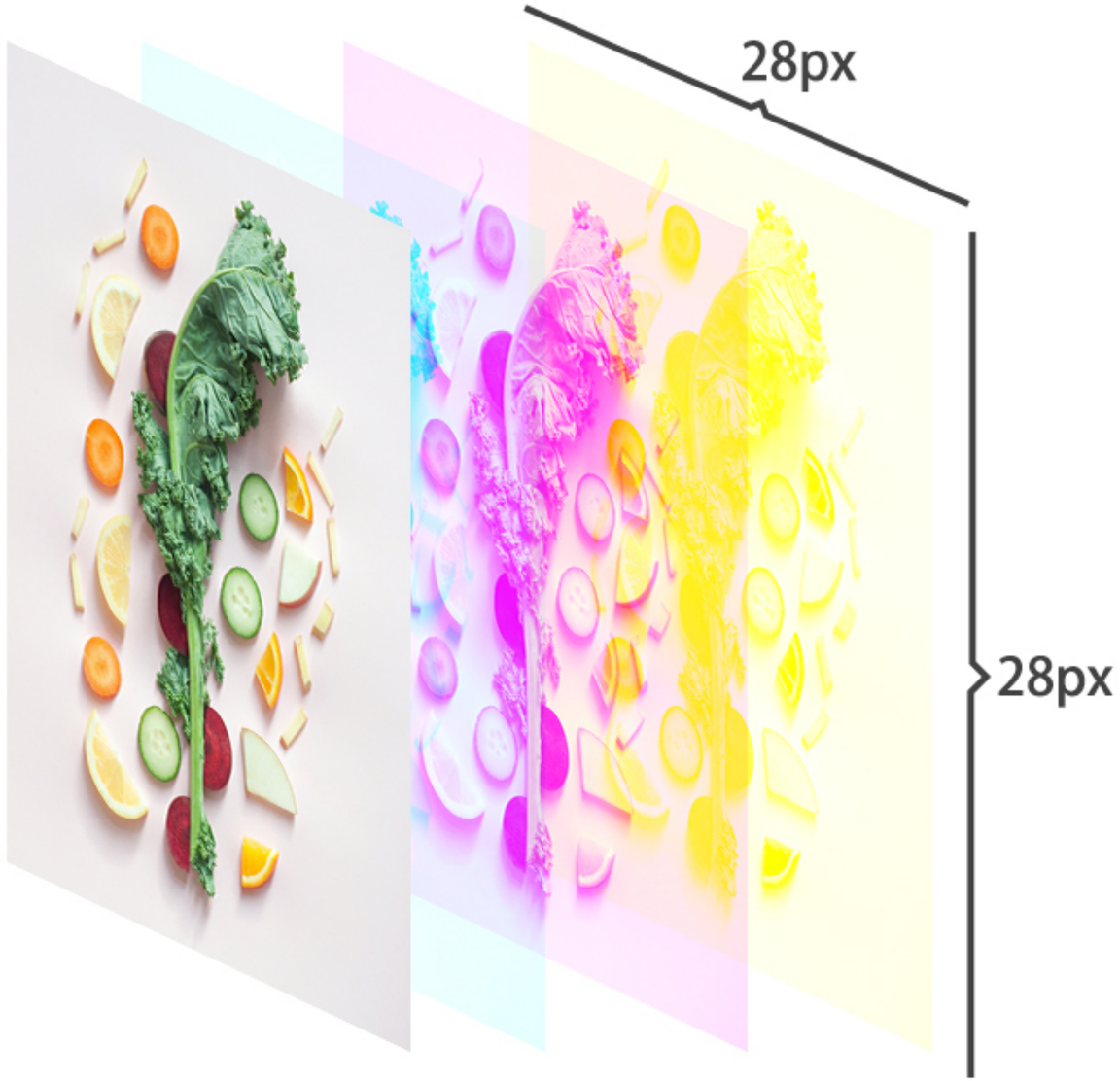
12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

input_shape
(WIDTH,HEIGHT,CHANNELS)

input_shape = (28,28,3)

Color Images have 3 channels
RGB (Red Green Blue)



```
# Import Conv2D layer and Flatten from keras layers
from keras.layers import Dense, Conv2D, Flatten

# Instantiate your model as usual
model = Sequential()

# Add a convolutional layer with 32 filters of size 3x3
model.add(Conv2D(filters=32,
                 kernel_size=3,
                 input_shape=(28, 28, 1),
                 activation='relu'))

# Add another convolutional layer
model.add(Conv2D(8, kernel_size=3, activation='relu'))

# Flatten the output of the previous layer
model.add(Flatten())

# End this multiclass model with 3 outputs and softmax
model.add(Dense(3, activation='softmax'))
```

1

2

3

ResNet50
(50 Layers)

4 Predicted

("cheeseburger', 0.99),
('bagel', 0.01),
('bakery', 0.00)

Pre-processing images for ResNet50

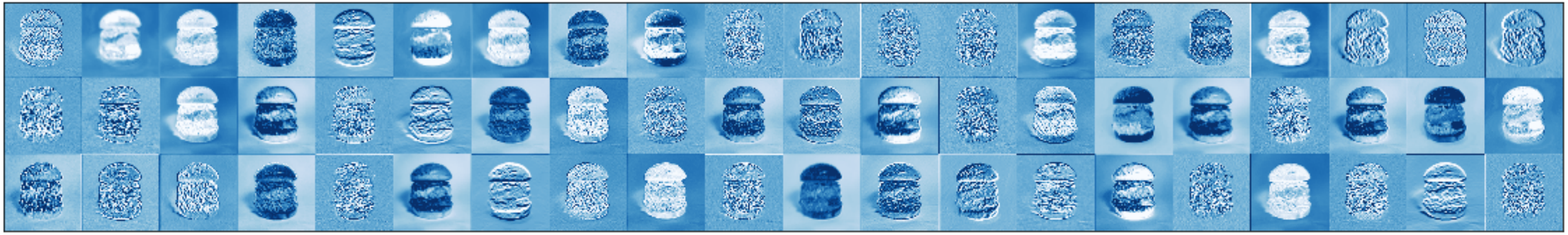
```
# Import image from keras preprocessing
from keras.preprocessing import image

# Import preprocess_input from keras applications resnet50
from keras.applications.resnet50 import preprocess_input

# Load the image with the right target size for your model
img = image.load_img(img_path, target_size=(224, 224))

# Turn it into an array
img = image.img_to_array(img)

# Expand the dimensions so that it's understood by our network:
# img.shape turns from (224, 224, 3) into (1, 224, 224, 3)
img = np.expand_dims(img, axis=0)

# Pre-process the img in the same way training images were
img = preprocess_input(img)
```

Using the ResNet50 model in Keras

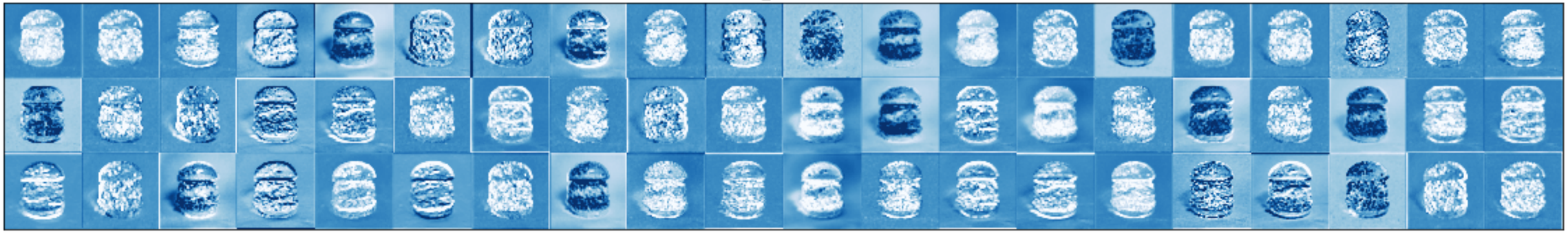
```
# Import ResNet50 and decode_predictions from keras.applications.resnet50
from keras.applications.resnet50 import ResNet50, decode_predictions

# Instantiate a ResNet50 model with imagenet weights
model = ResNet50(weights='imagenet')

# Predict with ResNet50 on our img
preds = model.predict(img)

# Decode predictions and print it
print('Predicted:', decode_predictions(preds, top=1)[0])
```

```
Predicted: [('n07697313', 'cheeseburger', 0.9868016)]
```

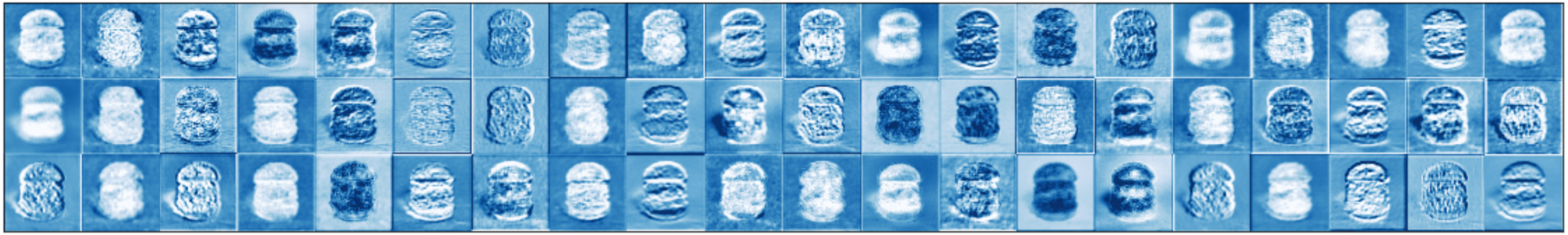
conv1



res2a_branch2a



res2a_branch2b



Let's experiment!

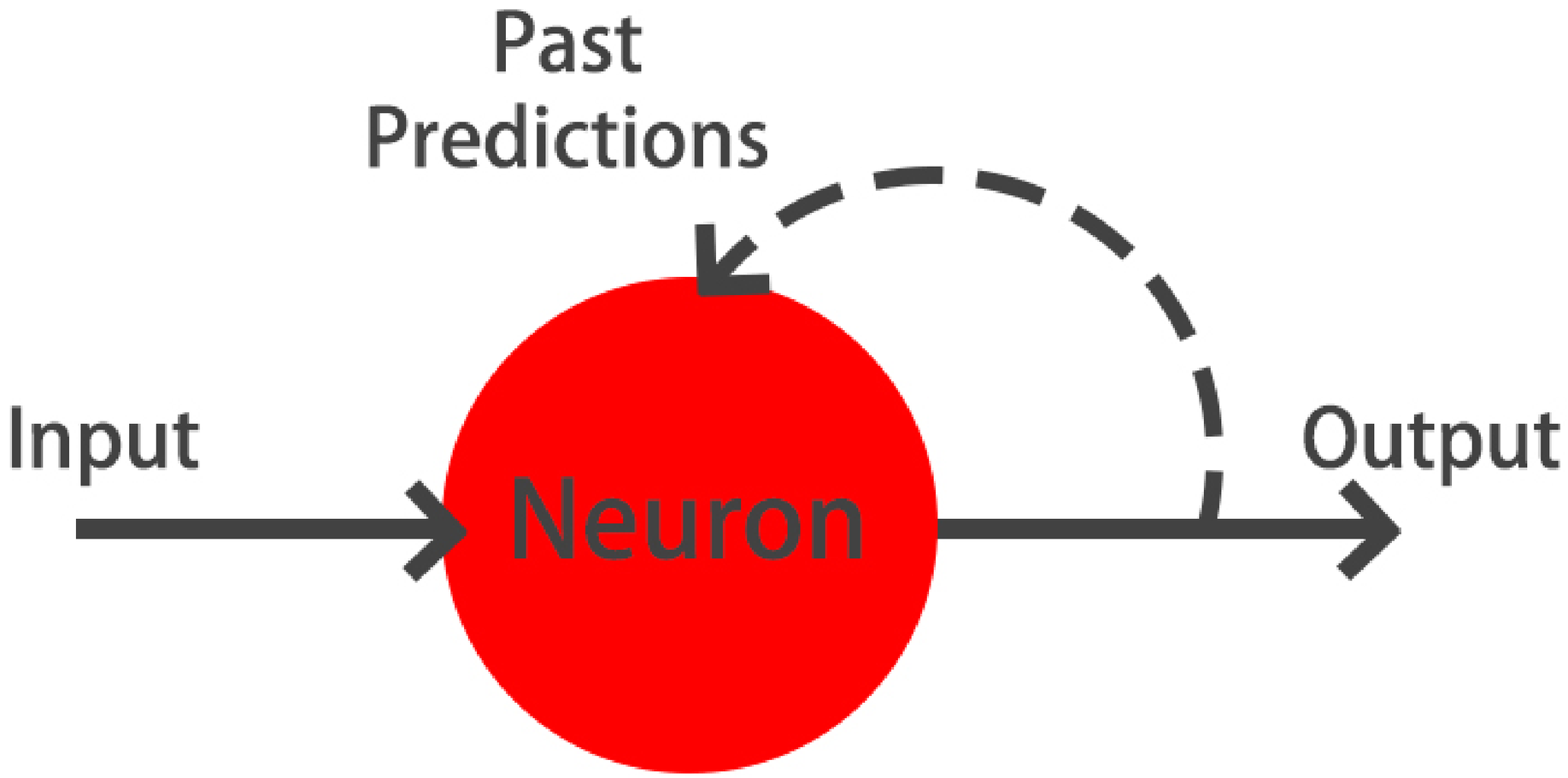
DEEP LEARNING WITH KERAS IN PYTHON

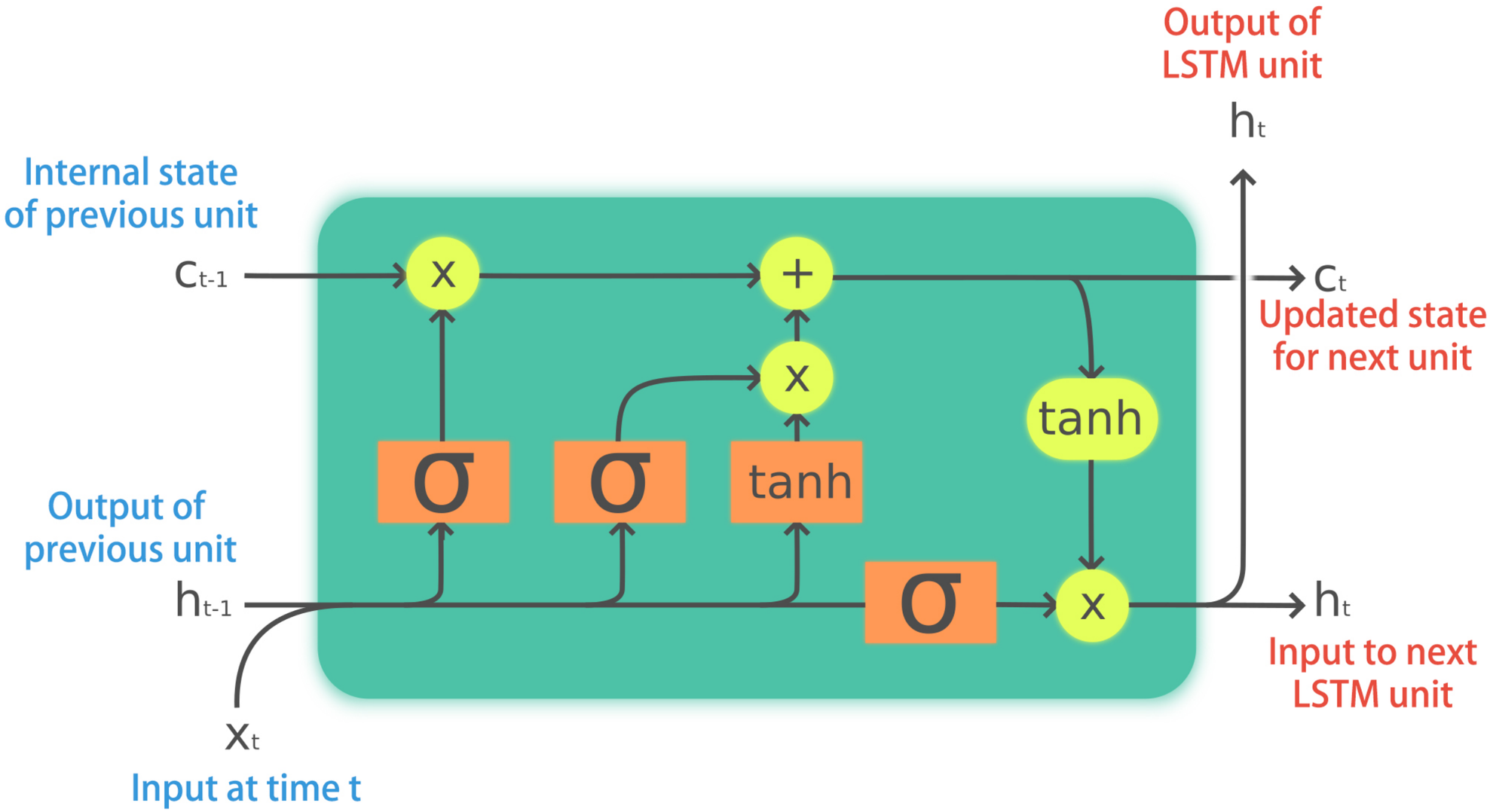
Intro to LSTMs

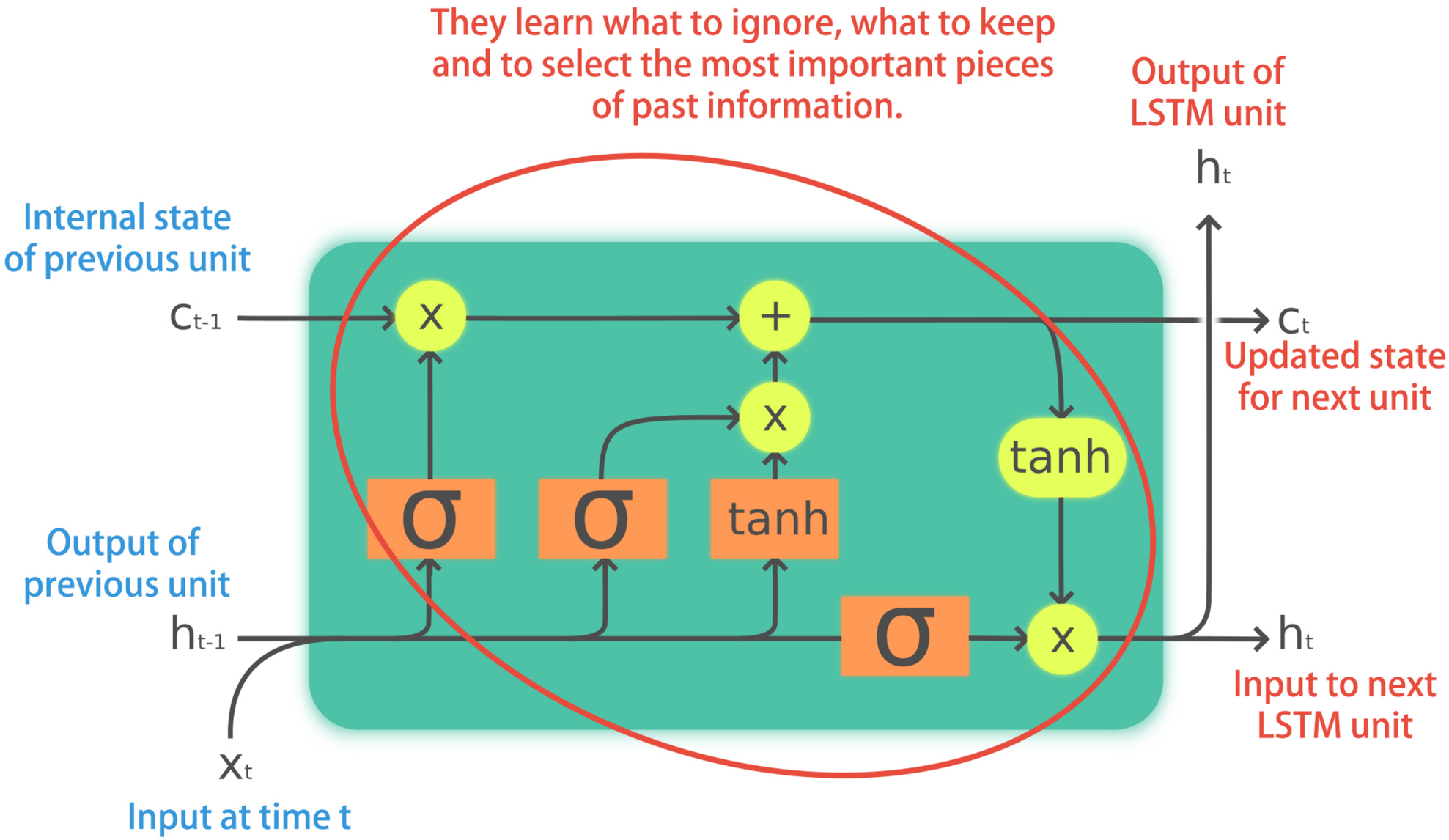
DEEP LEARNING WITH KERAS IN PYTHON

Miguel Esteban
Data Scientist & Founder

What are RNNs?

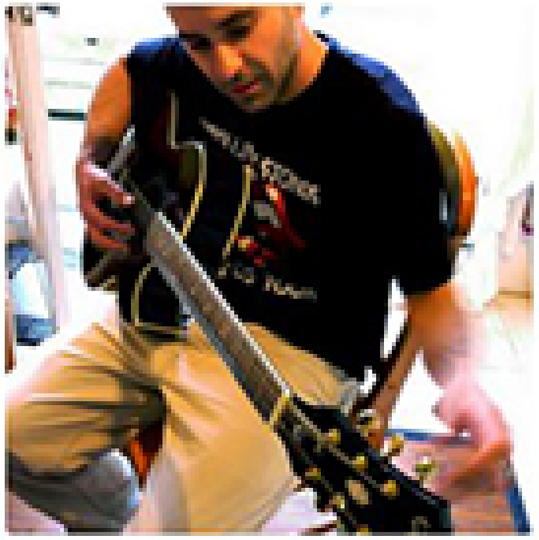



Output of
LSTM unit
h_t
Internal state
of previous unit
C_{t-1}
x
+
C_t
Updated state
for next unit
x
tanh
σ
σ
tanh
σ
x
Output of
previous unit
h_{t-1}
h_t
Input to next
LSTM unit
x_t
Input at time t

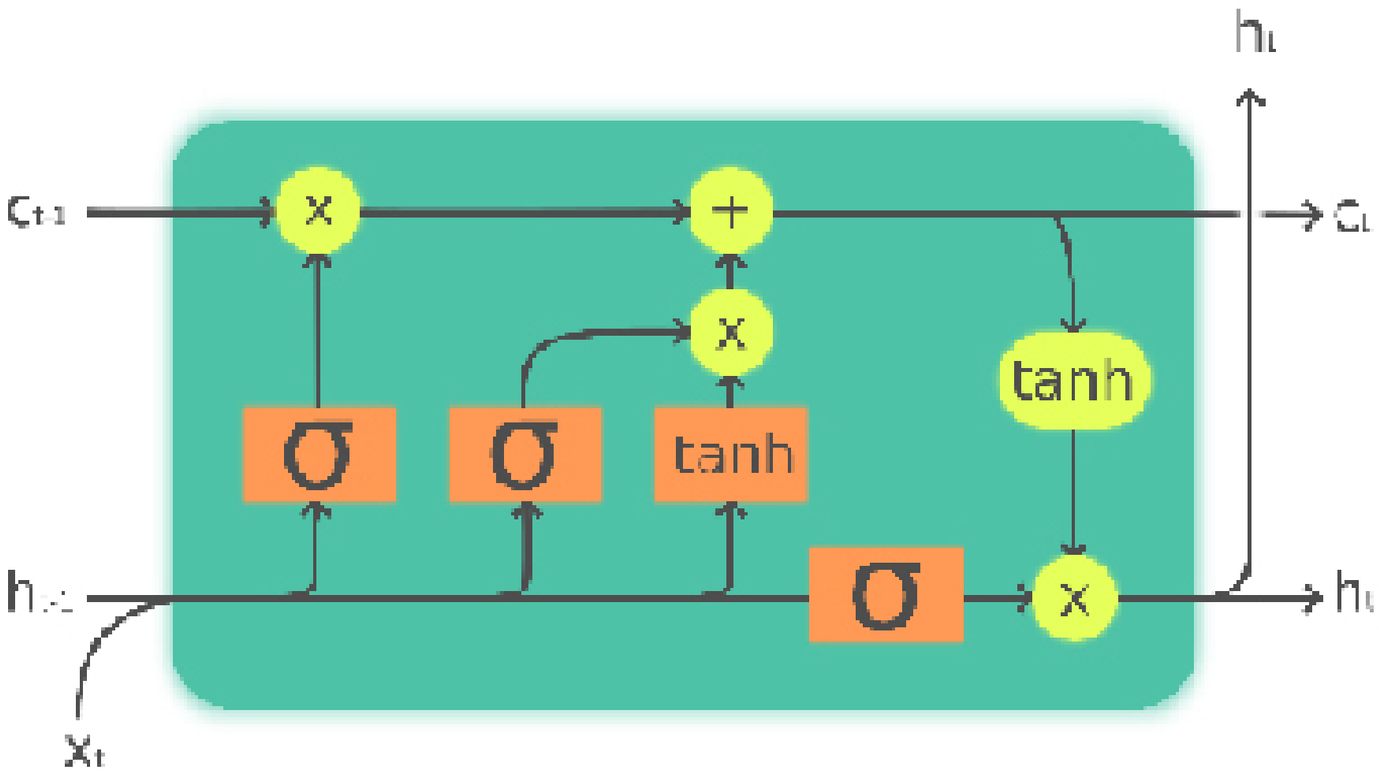

They learn what to ignore, what to keep and to select the most important pieces of past information.
Output of LSTM unit
h_t
Internal state of previous unit
C_{t-1}
x
+
C_t
Updated state for next unit
x
tanh
σ
σ
tanh
σ
x
Output of previous unit
h_{t-1}
h_t
Input to next LSTM unit
x_t
Input at time t

When to use LSTMs?

- Image captioning
- Speech to text
- Text translation
- Document summarization
- Text generation
- Musical composition
- ...



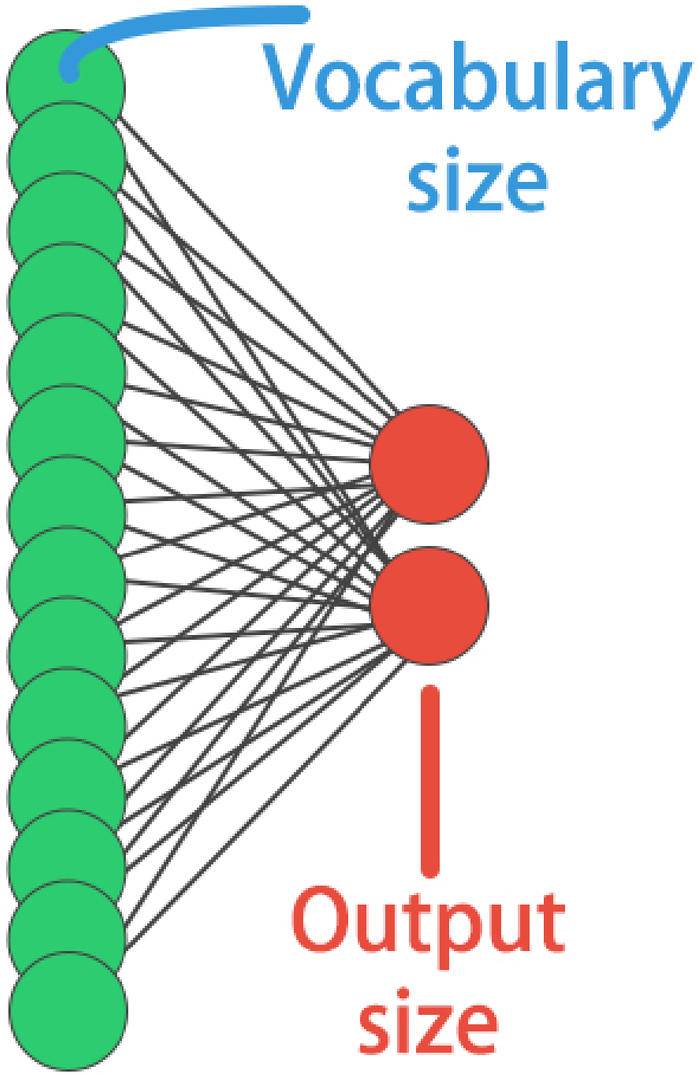
[1] Karpathy, A., & Fei [2] Fei, L. (2015). Deep visual [3] semantic alignments for generating image descriptions.

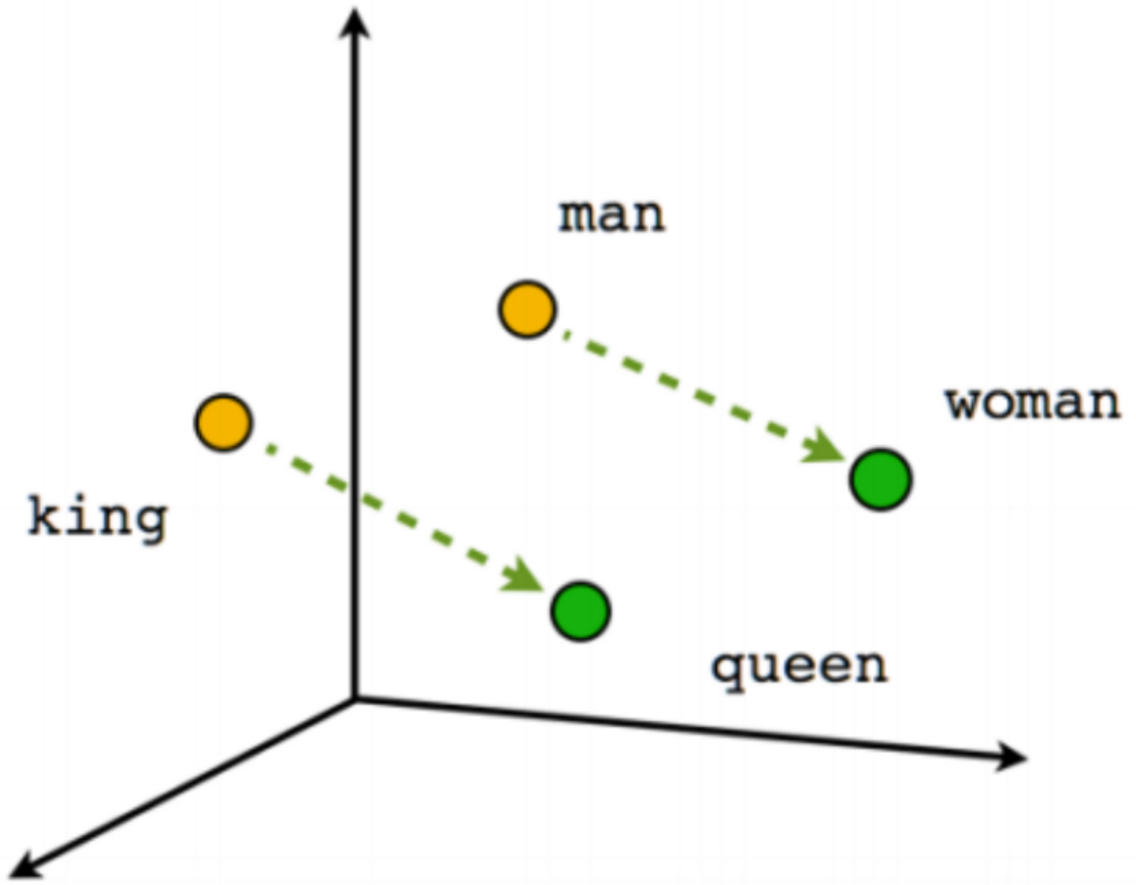


+ Text

this is a sentence

42 1 1 23 1


Vocabulary size
Output size


man
woman
king
queen

```
text = 'Hi this is a small sentence'

# We choose a sequence length
seq_len = 3

# Split text into a list of words
words = text.split()
```

```
['Hi', 'this', 'is', 'a', 'small', 'sentence']
```

```
# Make lines
lines = []
for i in range(seq_len, len(words) + 1):
  line = ' '.join(words[i-seq_len:i])
  lines.append(line)
```

```
['Hi this is', 'this is a', 'is a small', 'a small sentence']
```

```
# Import Tokenizer from keras preprocessing text
from keras.preprocessing.text import Tokenizer

# Instantiate Tokenizer
tokenizer = Tokenizer()

# Fit it on the previous lines
tokenizer.fit_on_texts(lines)

# Turn the lines into numeric sequences
sequences = tokenizer.texts_to_sequences(lines)
```

```
array([[5, 3, 1], [3, 1, 2], [1, 2, 4], [2, 4, 6]])
```

```
print(tokenizer.index_word)
```

```
{1: 'is', 2: 'a', 3: 'this', 4: 'small', 5: 'hi', 6: 'sentence'}
```

```
# Import Dense, LSTM and Embedding layers
from keras.layers import Dense, LSTM, Embedding
model = Sequential()

# Vocabulary size
vocab_size = len(tokenizer.index_word) + 1

# Starting with an embedding layer
model.add(Embedding(input_dim=vocab_size, output_dim=8, input_length=2)

# Adding an LSTM layer
model.add(LSTM(8))


# Adding a Dense hidden layer
model.add(Dense(8), activation='relu')

# Adding an output layer with softmax
model.add(Dense(vocab_size, activation='softmax'))
```

Let's do it!

DEEP LEARNING WITH KERAS IN PYTHON

You're done!

DEEP LEARNING WITH KERAS IN PYTHON

Miguel Esteban
Data Scientist & Founder

Congratulations!

You've learned a lot

What you've learned

- Basics of neural networks
- Building sequential models
- Building models for regression
- Approaching binary classification, multi-class and multi-label problems with neural networks
- Activation functions
- Hyperparameter optimization
- Autoencoders
- De-noising images
- CNN concepts
- Use pre-trained models
- Visualize convolutions
- LSTMs concepts
- Work with LSTMs and text
- All this by using many different datasets and learning a lot of Keras utility functions.

What could you learn next?

- Go deeper into CNNs
- Go deeper into LSTMs
- Keras Functional API
- Models that share layers, models with several branches
- GANs: Generative Adversarial Networks
- Deeplearning projects

Goodbye!

Have a good one!

DEEP LEARNING WITH KERAS IN PYTHON