Saving models —

1) Traditional —

model.save('my_model.keras').h5')

2) Saving weights

model.save_weights('my_model.weights.h5')

model.load_weights('my_model.weights.h5')

model.load_weights('my_model.weights.h5', by_name = True)

only the 1st tag layer will use the weights of previous model by name

3) Configuration
Saves JSon

4) Customs Saving

Attention layer = tells specific data to be important so machine doesn't forget

Functional API

makes flexible model

keras.utils.plot_model(~~model~~,)

↑ gi plots ~~and~~ model in flow chart

Dense layers

weights : frequency
(tuning the data)

Biases : Volume
(output of data)

initializer = keras.initializers.
GlorotNormal()

↑ assigns weight

line = $cx + c$ ← bias constant

↑ weights

Dropout

self.seed_generator
= keras.random.SeedGenerator(1337)
↑
reproducible output

Task = changes value in
seed Generator and see
change in output.
Source code will be given.
~~Chage~~ Change read.me on github

~~Auto encoder~~:
assembles (encoder and decoder)
Images will be vectorized
and when given input will be
reconstructed

Encoder

no dense layer since no classification
is done

Decoder

x = layers.UpSampling2D(3)(x)
↑
opposite of pooling

System with multiple inputs / outputs.

embedding :-
makes graph of different elements
(multi-dimensional)

LSTM :- (Long-Short Term Memory)
Memorizes words from embedding
(as vectors)