keras.Sequential()

`Sequential` groups a linear stack of layers into a `Model`.

```
tf.keras.Sequential(
    layers=None
)
```

`layers`	The list of layers composing the network.

Methods

A list of useful methods.

compile

```
compile(
    optimizer='rmsprop',
    loss=None,
    metrics=None
)
```

Configures the model for training.

`optimizer`	String (name of optimizer) or optimizer instance.
`loss`	Loss function. A string (name of loss function). May be `'categorical_crossentropy'` or `'binary_crossentropy'`.
`metrics`	List of metrics to be evaluated by the model during training and testing. Typically you will use `metrics=['accuracy']`.

fit

```
fit(
    x=None,
    y=None,
    batch_size=None,
    epochs=1,
    verbose='auto',
    validation_data=None
)
```

Trains the model for a fixed number of epochs (dataset iterations).

x	Input data. It could be: • A NumPy array (or array-like), or a list of arrays (in case the model has multiple inputs). • A tensor, or a list of tensors (in case the model has multiple inputs). • A dict mapping input names to the corresponding array/tensors, if the model has named inputs. • A pandas Dataframe
y	Target data. Like the input data x, it could be either NumPy array(s) or a pandas Dataframe.
validation_data	Data on which to evaluate the loss and any model metrics at the end of each epoch. The model will not be trained on this data. Like the input data x, it could be either NumPy array(s) or a pandas Dataframe.
batch_size	Integer or None. Number of samples per gradient update. If unspecified, batch_size will default to 32.
epochs	Integer. Number of epochs to train the model. An epoch is an iteration over the entire x and y data provided.
verbose	"auto", 0, 1, or 2. Verbosity mode. 0 = silent, 1 = progress bar, 2 = one line per epoch. "auto" becomes 1 for most cases. Note that the progress bar is not particularly useful when logged to a file, so verbose=2 is recommended when not running interactively (e.g., in a production environment). Defaults to "auto".

evaluate

```
evaluate(
    x=None,
    y=None,
    verbose='auto'
)
```

Returns the loss value & metrics values for the model in test mode.

x	Input data. It could be: • A NumPy array (or array-like), or a list of arrays (in case the model has multiple inputs). • A tensor, or a list of tensors (in case the model has multiple inputs).

	• A dict mapping input names to the corresponding array/tensors, if the model has named inputs. • A pandas Dataframe
`y`	Target data. Like the input data `x`, it could be either NumPy array(s) or a pandas Dataframe.
`verbose`	`"auto"`, 0, 1, or 2. Verbosity mode. 0 = silent, 1 = progress bar, 2 = single line. `"auto"` becomes 1 for most cases. Note that the progress bar is not particularly useful when logged to a file, so `verbose=2` is recommended when not running interactively (e.g. in a production environment). Defaults to `"auto"`.

predict

```
predict(
    x, verbose='auto'
)
```

Generates output predictions for the input samples.

`x`	Input data. It could be: • A NumPy array (or array-like), or a list of arrays (in case the model has multiple inputs). • A tensor, or a list of tensors (in case the model has multiple inputs). • A dict mapping input names to the corresponding array/tensors, if the model has named inputs. • A pandas Dataframe
`verbose`	`"auto"`, 0, 1, or 2. Verbosity mode. 0 = silent, 1 = progress bar, 2 = single line. `"auto"` becomes 1 for most cases. Note that the progress bar is not particularly useful when logged to a file, so `verbose=2` is recommended when not running interactively (e.g. in a production environment). Defaults to `"auto"`.

keras.layers.Input()

Used to instantiate a Keras tensor.

```
tf.keras.Input( shape=None )
```

shape	A shape tuple (tuple of integers or None objects). For instance, `shape=(32,)` indicates that the expected input will be 32-dimensional vectors. Elements of this tuple can be None; None elements represent dimensions where the shape is not known and may vary (e.g. sequence length).

keras.layers.Dense()

Just your regular densely-connected NN layer.

```
tf.keras.layers.Dense(
    units,
    activation=None
)
```

`units`	Positive integer, dimensionality of the output space.
`activation`	Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: `a(x) = x`). Possible values: 'relu', ' sigmoid', softmax.

keras.optimizers.Adam()

Optimizer that implements the Adam algorithm.

```
tf.keras.optimizers.Adam( learning_rate=0.001 )
```

Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. According to Kingma et al., 2014, the method is "*computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters*".

`learning_rate`	A float. The learning rate. Defaults to `0.001`.