

Compiling the Model

Compile defines the loss function, the optimizer, and the metrics. That's all. It has nothing to do with the weights and you can compile a model as many times as you want without causing any problems with pre-trained weights. You need a compiled model to train (because training uses the loss function and the optimizer).

Adding Activation Function to the model layer

The activation function is defined in the dense layer of the model and is used to squeeze the value within a particular range. In simple terms, it is a function that is used to convert the input signal of a node to an output signal. **tf.Keras** comes with the following predefined activation functions to choose from

```
model = tf.keras.Sequential()  
model.add()  
model.add()
```

For Example:

```
model.add(Dense(10, input_shape=(784,))  
# This is same as:  
model.add(Dense(10, input_dim=784,))  
# And to the following:  
model.add(Dense(10, batch_input_shape=(None, 784)))
```

Compiling the model

Syntax:

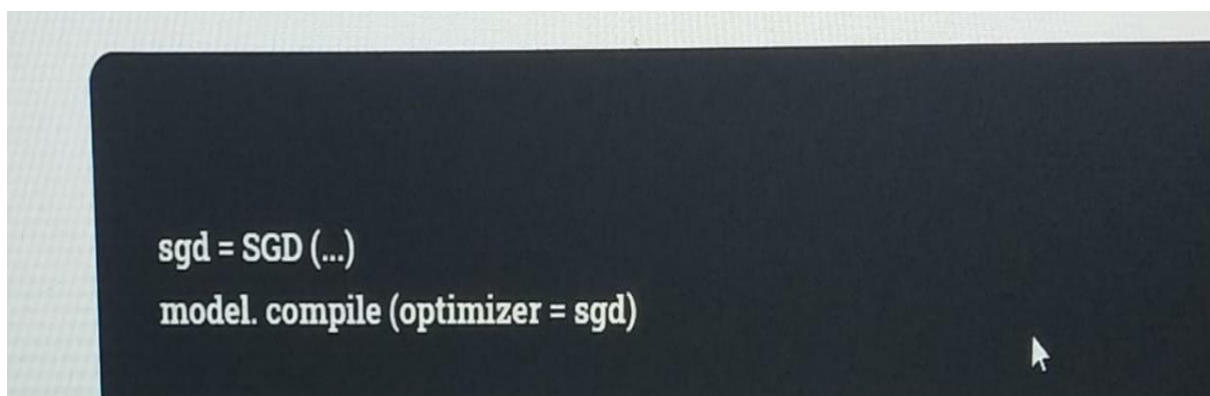
```
model.compile(optimizer=..., loss=..., metrics = ...)
```

1. Optimizer: While training a deep learning model, we need to alter the weights of each epoch and minimize the loss function. An optimizer is a function or algorithm that adjusts the neural network's properties such as weights and learning rate. As a result, it helps to reduce total loss and enhance the accuracy of your model.

Some of the popular *Gradient Descent Optimizers* are:

- **SGD:** Stochastic gradient descent, to reduce the computation cost of gradient
- **RMSprop:** Adaptive learning rate optimization method which utilizes the magnitude of recent gradients to normalize the gradients
- **Adam:** Adaptive Moment Estimation (Adam) leverages the power of adaptive learning rates methods to find individual learning rates for each parameter.

For Example:



```
sgd = SGD (...)  
model.compile(optimizer = sgd)
```

2. Loss: Loss functions are a measure of how well your model predicts the predicted outcome.

Some of the popular *Model Loss Functions* are:

- **mese:** for mean squared error
- **binary_crossentropy:** for binary logarithmic loss (log loss)
- **categorical_crossentropy:** for multi-class logarithmic loss (log loss)

For Example:

```
model.compile(optimizer= adam ",loss='mse',metrics=['accuracy']  
)
```

Let's put them together in the code:

```
# Compiling the model.  
model_1.compile(optimizer='SGD',  
                loss='categorical_crossentropy',  
                metrics=['accuracy'])
```