Early Stopping

- Early Stopping is userd to stop the program earlier then specified number of many epochs.
- This is implemented in fit() as the model is trained in the fit function.
- For early stopping the fit function can have many parameter like:
    - Callbacks
    - Patience
    - Min Delta

Without Early Stopping

```
import numpy as np
from keras.layers import Input, concatenate
from keras.models import Model

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([[0], [1], [1], [1]])

Inp = Input(shape = (2))
h1_1 = Dense(units = 2, activation = 'relu')(Inp)
h1_2 = Dense(units = 2, activation = 'relu')(Inp)
c = concatenate([h1_1, h1_2])
h2 = Dense(units = 2, activation = 'relu')(c)
Output = Dense(units = 1, activation='sigmoid')(h2)
functional_model = Model(Inp, Output)

functional_model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
functional_model.fit(x_train, y_train, epochs=10)

x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_pred = functional_model.predict(x_test)

y_pred = np.round(y_pred)
print("Prediction: ", y_pred)
```

```
Epoch 1/10
1/1 [==============================] - 1s 774ms/step - loss: 0.2494 - accuracy: 1.0000
Epoch 2/10
1/1 [==============================] - 0s 29ms/step - loss: 0.2491 - accuracy: 0.7500
Epoch 3/10
1/1 [==============================] - 0s 19ms/step - loss: 0.2485 - accuracy: 1.0000
Epoch 4/10
1/1 [==============================] - 0s 17ms/step - loss: 0.2482 - accuracy: 1.0000
Epoch 5/10
1/1 [==============================] - 0s 18ms/step - loss: 0.2477 - accuracy: 1.0000
Epoch 6/10
1/1 [==============================] - 0s 19ms/step - loss: 0.2472 - accuracy: 1.0000
Epoch 7/10
1/1 [==============================] - 0s 13ms/step - loss: 0.2468 - accuracy: 1.0000
Epoch 8/10
1/1 [==============================] - 0s 30ms/step - loss: 0.2463 - accuracy: 1.0000
Epoch 9/10
1/1 [==============================] - 0s 30ms/step - loss: 0.2460 - accuracy: 1.0000
Epoch 10/10
1/1 [==============================] - 0s 18ms/step - loss: 0.2455 - accuracy: 1.0000
1/1 [==============================] - 0s 343ms/step
Prediction:  [[0.]
 [1.]
 [1.]
 [1.]]
```

With Early Stopping

```
#format of fit() function
fit(train,  target, epochs = ----, verbose = 0, callbacks = mycallback)
```

FACT: Verbose by default is 1

CALLBACKS

- Callbacks can monitor either loss or accuracy.
- Accuracy 1 can come even when the model is marginally confident so it is suggested to monitor loss.
- When taking the accuracy, mode is taken as "**max**" and when taking loss, mode is taken as "**min**".
- Callback function should be defined before the fit function.

```
myCallback = keras.callbacks.EarlyStopping(monitor = 'loss', mode = 'min', patience = 10)
functional_model.fit(x_train, y_train, epochs=10000, callbacks = myCallback)
```

```
import tensorflow as tf
tf.random.set_seed(34)

import keras
import numpy as np
from keras.layers import Input, concatenate, Dense
from keras.models import Model

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([[0], [1], [1], [1]])

Inp = Input(shape = (2))
h1_1 = Dense(units = 2, activation = 'relu')(Inp)
h1_2 = Dense(units = 2, activation = 'relu')(Inp)
c = concatenate([h1_1, h1_2])
h2 = Dense(units = 2, activation = 'relu')(c)
Output = Dense(units = 1, activation='sigmoid')(h2)
functional_model = Model(Inp, Output)

functional_model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
myCallback = keras.callbacks.EarlyStopping(monitor = 'loss', mode = 'min')
functional_model.fit(x_train, y_train, epochs=10000,callbacks = myCallback)

x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_pred = functional_model.predict(x_test)
y_pred = np.round(y_pred)
print("Prediction: ", y_pred)

pred = functional_model.predict(x_train)
pred
```

```
Epoch 1/10000
WARNING:tensorflow:6 out of the last 19 calls to <function Model.make_train_function.<locals>.train_function at 0x7f2b1ee3ad30> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings
1/1 [==============================] - 0s 472ms/step - loss: 0.2494 - accuracy: 1.0000
Epoch 2/10000
1/1 [==============================] - 0s 12ms/step - loss: 0.2491 - accuracy: 0.7500
WARNING:tensorflow:6 out of the last 10 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f2b1fece9d0> triggered tf.function retracing. Tracing is expensive and the excessive number of trac
1/1 [==============================] - 0s 76ms/step
Prediction:  [[0.]
 [1.]
 [1.]
 [1.]]
1/1 [==============================] - 0s 24ms/step
array([[0.4998407 ],
       [0.94797933],
       [0.79942334],
       [0.97621727]], dtype=float32)
```

Patience

- Now the problem with early stopping is that it may get too excited as soon as the loss gets low and it stop earlier than required. here comes the paramter of "Patience".
- While measuring the loss using SGD we are basically looking for minimas.
- With early stopping the program can stop at a **Local Minima** which is not what we want. We want the program to stop at **Global Minima** instead.
- Due to this reason we want the callback to have some patience. So we can use the **Patience** parameter in the callback
- **Eg**: Patience = 5 means that when a minima is attained it will check for the next 5 more epochs if there is a change and the mimima is a global minima. If not then the program will keep on running other it will stop after running 5 more epochs.
- One can check the last 5 consecutive losses to match the result.

```
import tensorflow as tf
tf.random.set_seed(34)

import keras
import numpy as np
from keras.layers import Input, concatenate, Dense
from keras.models import Model

x_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_train = np.array([[0], [1], [1], [1]])

Inp = Input(shape = (2))
h1_1 = Dense(units = 2, activation = 'relu')(Inp)
h1_2 = Dense(units = 2, activation = 'relu')(Inp)
c = concatenate([h1_1, h1_2])
h2 = Dense(units = 2, activation = 'relu')(c)
Output = Dense(units = 1, activation='sigmoid')(h2)
functional_model = Model(Inp, Output)

functional_model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
myCallback = keras.callbacks.EarlyStopping(monitor = 'loss', mode = 'min', patience =5)
functional_model.fit(x_train, y_train, epochs=10000,callbacks = myCallback)

x_test = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_pred = functional_model.predict(x_test)

y_pred = np.round(y_pred)
print("Prediction: ", y_pred)

pred = functional_model.predict(x_train)
pred
```

```
Streaming output truncated to the last 5000 lines.
1/1 [==============================] - 0s 10ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6376/10000
1/1 [==============================] - 0s 9ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6377/10000
1/1 [==============================] - 0s 6ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6378/10000
1/1 [==============================] - 0s 6ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6379/10000
1/1 [==============================] - 0s 7ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6380/10000
1/1 [==============================] - 0s 10ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6381/10000
1/1 [==============================] - 0s 6ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6382/10000
1/1 [==============================] - 0s 9ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6383/10000
1/1 [==============================] - 0s 6ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6384/10000
1/1 [==============================] - 0s 6ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6385/10000
1/1 [==============================] - 0s 6ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6386/10000
1/1 [==============================] - 0s 9ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6387/10000
1/1 [==============================] - 0s 6ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6388/10000
1/1 [==============================] - 0s 12ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6389/10000
1/1 [==============================] - 0s 8ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6390/10000
1/1 [==============================] - 0s 9ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6391/10000
1/1 [==============================] - 0s 7ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6392/10000
1/1 [==============================] - 0s 9ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6393/10000
1/1 [==============================] - 0s 6ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6394/10000
1/1 [==============================] - 0s 30ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6395/10000
1/1 [==============================] - 0s 121ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6396/10000
1/1 [==============================] - 0s 29ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6397/10000
1/1 [==============================] - 0s 11ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6398/10000
1/1 [==============================] - 0s 9ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6399/10000
1/1 [==============================] - 0s 11ms/step - loss: 0.0184 - accuracy: 1.0000
```

```
Epoch 6400/10000
1/1 [==============================] - 0s 8ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6401/10000
1/1 [==============================] - 0s 7ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6402/10000
1/1 [==============================] - 0s 11ms/step - loss: 0.0184 - accuracy: 1.0000
Epoch 6403/10000
1/1 [==============================] - 0s 6ms/step - loss: 0.0183 - accuracy: 1.0000
```