Deep Learning Training DN

by Sanjeev Gupta

Training a DNN using the sequntial API on the Boston housing dataset

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.layers import Dense, Flatten
from keras.callbacks import TensorBoard, ModelCheckpoint,EarlyStopping
from tensorflow.keras.datasets import imdb
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.utils import plot_model
from keras import models
from keras import layers
```

```
# Load Data

from keras.datasets import boston_housing
```

```
#check train data and test data

(train_data, train_labels), (test_data, test_labels) =  boston_housing.load_data()
print(f"train_data[0] = {train_data[0]}")   # encoded review
print(f"len(train_data[0] ) = {len(train_data[0])}")
print(f"train_labels[0] ={train_labels[0]}")
```

```
train_data[0] = [  1.23247   0.        8.14      0.        0.538     6.142    91.7
   3.9769    4.      307.       21.      396.9      18.72   ]
len(train_data[0] ) = 13
train_labels[0] =15.2
```

```
train_data.shape
test_data.shape
print(f"train_data.shape ={train_data.shape}")
print(f"test_data.shape ={test_data.shape}")
```

```
train_data.shape =(404, 13)
test_data.shape =(102, 13)
```

```
# Normalize the data

mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /= std
test_data -= mean
test_data /= std
```

```
# split in train and validation data set

train_x = train_data[101:] #-- from 102 onwards - train set i.e. 75%
train_y = train_data[101:]
val_x = train_data[:101] # 1st 101 items -- i.e. 25% of original training data set of 404 rows
val_y = train_data[:101]
```

```
print(f"train_x.shape ={train_x.shape}")
print(f"val_x.shape ={val_x.shape}")
```

```
train_x.shape =(303, 13)
val_x.shape =(101, 13)
```

```
model = models.Sequential()
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1, activation= 'linear'))
model.build(input_shape=(None, 13))
```

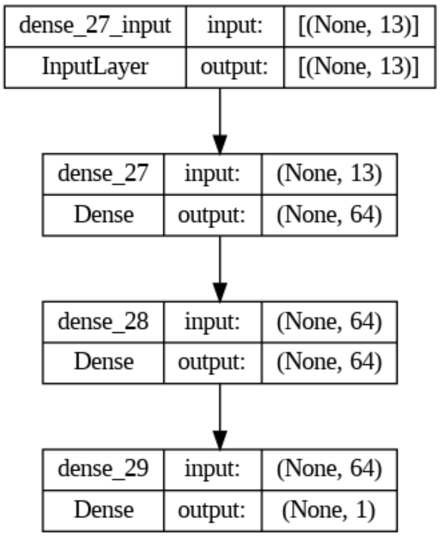
```
model.summary()
plot_model(model, show_shapes=True)
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_27 (Dense)            (None, 64)                896

 dense_28 (Dense)            (None, 64)                4160

 dense_29 (Dense)            (None, 1)                 65

=================================================================
Total params: 5,121
Trainable params: 5,121
Non-trainable params: 0
_________________________________________________________________
```



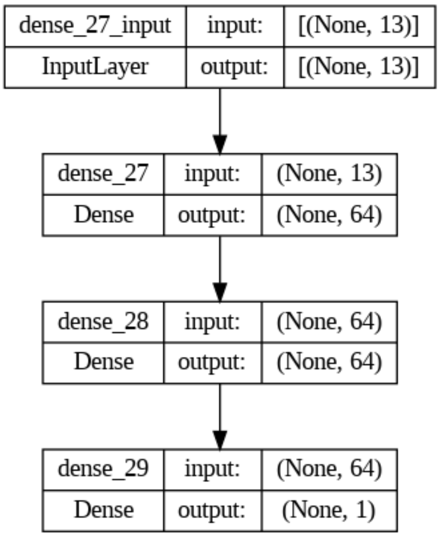
```
callbacks = [EarlyStopping(monitor="val_loss", patience=2),
             ModelCheckpoint("BHC_model_checkpoint",save_best_only=True),
             TensorBoard(log_dir="/tensorboard_files")]
```

```
model.compile(optimizer='rmsprop',
             loss='mse',
             metrics=['mae'])
```

```
history = model.fit(train_x,
                   train_y,
                   epochs=26,
                   batch_size=16,
                   validation_data=(val_x, val_y),
                   callbacks=callbacks)
```

```
Epoch 1/26
18/19 [===========================>..] - ETA: 0s - loss: 0.9227 - mae: 0.6931WARNING:absl:Found untraced functions such as _update_
19/19 [==============================] - 2s 64ms/step - loss: 0.9167 - mae: 0.6926 - val_loss: 0.9464 - val_mae: 0.7211
Epoch 2/26
14/19 [=====================>........] - ETA: 0s - loss: 0.8830 - mae: 0.6823WARNING:absl:Found untraced functions such as _update_
19/19 [==============================] - 1s 54ms/step - loss: 0.8821 - mae: 0.6765 - val_loss: 0.9442 - val_mae: 0.7163
Epoch 3/26
 1/19 [>.............................] - ETA: 0s - loss: 0.8760 - mae: 0.6830WARNING:absl:Found untraced functions such as _update_
19/19 [==============================] - 1s 41ms/step - loss: 0.8785 - mae: 0.6750 - val_loss: 0.9432 - val_mae: 0.7215
Epoch 4/26
 1/19 [>.............................] - ETA: 0s - loss: 0.8282 - mae: 0.6684WARNING:absl:Found untraced functions such as _update_
19/19 [==============================] - 1s 35ms/step - loss: 0.8770 - mae: 0.6737 - val_loss: 0.9403 - val_mae: 0.7102
Epoch 5/26
19/19 [==============================] - 0s 4ms/step - loss: 0.8767 - mae: 0.6727 - val_loss: 0.9438 - val_mae: 0.7079
Epoch 6/26
19/19 [==============================] - 0s 5ms/step - loss: 0.8769 - mae: 0.6731 - val_loss: 0.9434 - val_mae: 0.7231
```

```
plot_model(model, show_shapes=True)
```


dense_27_input | input: | [(None, 13)]
InputLayer | output: | [(None, 13)]
dense_27 | input: | (None, 13)
Dense | output: | (None, 64)
dense_28 | input: | (None, 64)
Dense | output: | (None, 64)
dense_29 | input: | (None, 64)
Dense | output: | (None, 1)

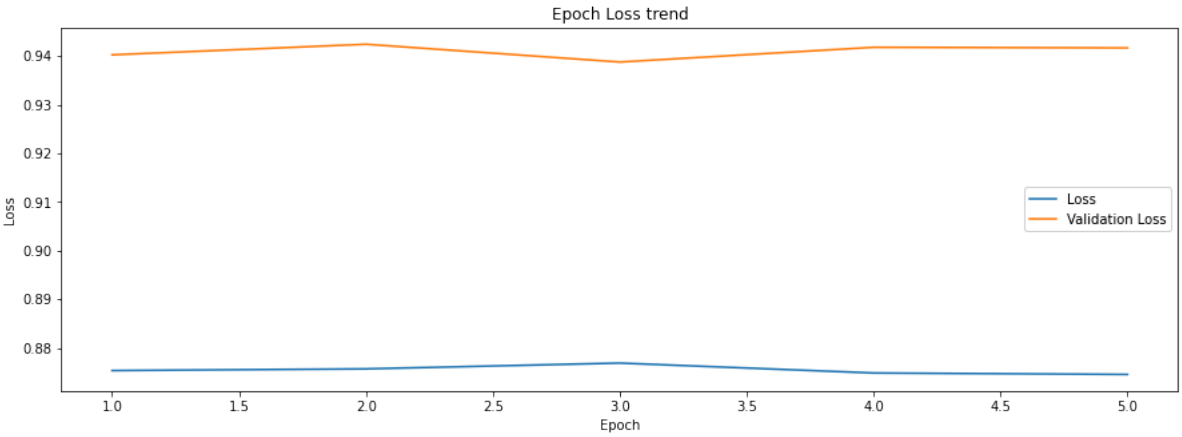
```
new_fit = model.fit(train_x,
                    train_y,
                    epochs=700,
                    batch_size=16,
                    verbose=1,
                    validation_data=(val_x, val_y),
                    callbacks=callbacks)
```

```
Epoch 1/700
 1/19 [>.............................] - ETA: 0s - loss: 1.2974 - mae: 0.7742WARNING:absl:Found untraced functions such as _update_
19/19 [==============================] - 1s 38ms/step - loss: 0.8754 - mae: 0.6730 - val_loss: 0.9402 - val_mae: 0.7110
Epoch 2/700
19/19 [==============================] - 0s 5ms/step - loss: 0.8757 - mae: 0.6727 - val_loss: 0.9424 - val_mae: 0.7078
Epoch 3/700
 1/19 [>.............................] - ETA: 0s - loss: 0.9807 - mae: 0.7324WARNING:absl:Found untraced functions such as _update_
19/19 [==============================] - 1s 65ms/step - loss: 0.8769 - mae: 0.6726 - val_loss: 0.9387 - val_mae: 0.7116
Epoch 4/700
19/19 [==============================] - 0s 5ms/step - loss: 0.8749 - mae: 0.6721 - val_loss: 0.9418 - val_mae: 0.7176
Epoch 5/700
19/19 [==============================] - 0s 6ms/step - loss: 0.8746 - mae: 0.6717 - val_loss: 0.9416 - val_mae: 0.7194
```

```
data = pd.DataFrame(new_fit.history)
data.head()
```

	loss	mae	val_loss	val_mae
0	0.875364	0.672966	0.940221	0.710972
1	0.875712	0.672663	0.942398	0.707757
2	0.876902	0.672593	0.938742	0.711625
3	0.874869	0.672132	0.941772	0.717585
4	0.874567	0.671658	0.941646	0.719385

```
figure = plt.gcf()
figure.set_size_inches((15, 5))
plt.title("Epoch Loss trend")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.plot(range(1, len(new_fit.history["loss"]) + 1), new_fit.history["loss"])
plt.plot(range(1, len(new_fit.history["val_loss"]) + 1), new_fit.history["val_loss"])
plt.legend(["Loss", "Validation Loss"])
plt.show()
```



```
test_x = test_data[:102] #-- from 1st item onwards
test_y = test_data[:102]

test_loss1, test_mae = model.evaluate(test_x, test_y)

print("Test LOSS:", test_loss1)
print("Test MeanAbsoluteErr:", test_mae)
print("\n")

pred=model.predict(test_x[0:10])
pred
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.8471 - mae: 0.6811
Test LOSS: 0.8470749258995056
Test MeanAbsoluteErr: 0.6811405420303345


1/1 [==============================] - 0s 72ms/step
array([[ 0.5891278 ],
       [-0.09123248],
       [-0.25890428],
       [ 0.10903289],
       [-0.30970424],
       [-0.1941256 ],
       [-0.0870996 ],
       [-0.21474262],
       [ 0.01392661],
       [ 0.44068208]], dtype=float32)
```

```
def build_model():

  model = keras.Sequential([
                          Dense(64, activation='relu', input_shape=(None, 13)),
                          Dense(64, activation='relu'),
                          Dense(1, activation='linear')])

#  model.build(input_shape=(None, 13))
  model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
  return model


k = 4
num_val_samples = len(test_x) // k
num_epochs = 500
all_scores = []


all_val_mae_histories = []
all_val_loss_histories = []
all_train_loss_histories = []
all_train_mae_histories = []


for i in range(k):
    print(f'Processing fold # {i}')

    val_data = train_x[i * num_val_samples: (i+1) * num_val_samples]
    val_targets = train_y[i * num_val_samples: (i+1) * num_val_samples]

    partial_train_data = np.concatenate(
                            [train_x[:i * num_val_samples],
                            train_x[(i+1) * num_val_samples:]],
                            axis=0)
    partial_train_targets = np.concatenate(
                            [train_x[:i * num_val_samples],
                            train_y[(i+1)*num_val_samples:]],
                            axis=0)


    model1=build_model()

    history2=model1.fit(partial_train_data,
              partial_train_targets, validation_data=(val_data, val_targets),
              epochs=num_epochs,batch_size=16, verbose=0)

    val_mae_history = history2.history["val_mae"]
    val_loss_history = history2.history["val_loss"]
    train_loss_history=history2.history["loss"]
    train_mae_history=history2.history["mae"]

    all_val_mae_histories.append(val_mae_history)
    all_val_loss_histories.append(val_loss_history)
    all_train_loss_histories.append(train_loss_history)
    all_train_mae_histories.append(train_mae_history)
```

```
Processing fold # 0
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
Processing fold # 1
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
Processing fold # 2
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
Processing fold # 3
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
WARNING:tensorflow:Model was constructed with shape (None, None, 13) for input KerasTensor(type_spec=TensorSpec(shape=(None, None, 1
```

```
data=pd.DataFrame(history2.history)
data.head()
```

	loss	mae	val_loss	val_mae
0	0.912282	0.688862	0.761905	0.629218
1	0.890953	0.678993	0.761082	0.619528
2	0.888472	0.678107	0.757470	0.615590
3	0.886607	0.676534	0.760383	0.617682
4	0.885658	0.677491	0.759515	0.613651

```
test_loss_kfold, test_mae_kfold = model1.evaluate(test_x, test_y)

print("\n")
print("Loss Before K-Fold validation:", test_loss1)
print("MeanAbsoluteError Before K-Fold validation:", test_mae)
print("\n")
print("Loss after K-Fold validation:", test_loss_kfold)
print("MeanAbsoluteError after K-Fold validation:", test_mae_kfold)
```

```
4/4 [==============================] - 0s 5ms/step - loss: 0.8418 - mae: 0.6724


Loss Before K-Fold validation: 0.8470749258995056
MeanAbsoluteError Before K-Fold validation: 0.6811405420303345


Loss after K-Fold validation: 0.8417602181434631
MeanAbsoluteError after K-Fold validation: 0.6723980903625488
```

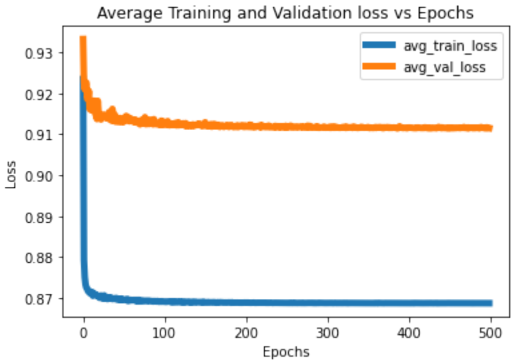
```
val_loss_histories_matrix = np.array(all_val_loss_histories)
avg_val_loss = val_loss_histories_matrix.mean(axis=0)

train_loss_histories_matrix = np.array(all_train_loss_histories)
avg_train_loss = train_loss_histories_matrix.mean(axis=0)

plt.plot(avg_train_loss, label='avg_train_loss', linewidth=5, zorder=-10)
plt.plot(avg_val_loss, label='avg_val_loss',linewidth=5, zorder=-10)
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Average Training and Validation loss vs Epochs")
plt.legend()

best_epoch = np.argmin(avg_val_loss)
print(f"minimum val loss at epoch: {best_epoch}")
```

```
minimum val loss at epoch: 386
```



New Section

New Section