tuning

March 10, 2022

[1]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras import models, layers
from tensorflow import keras
from tensorflow.keras.callbacks import EarlyStopping
from data_generation import DenseGenerator, ChessPositionGen
import keras_tuner as kt

import datetime
%load_ext tensorboard
```

We'll start with the same setup as in `full-puzzle-model.ipynb`

[2]:
```
# Setting paramaters on early stopping
earlystop = EarlyStopping(monitor='val_loss',
                          min_delta=0,
                          patience=5,
                          verbose=1,
                          mode='min',
                          restore_best_weights=True)

log_dir = "logs/fit/tuned" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1)
```

[3]:
```
# Memory management, likely not necessary, but used as a safety as per the␣
 ↪documentation recommendations on using GPUS

gpus = tf.config.list_physical_devices('GPU')
if gpus:
  try:
    # Currently, memory growth needs to be the same across GPUs
    for gpu in gpus:
        tf.config.experimental.set_memory_growth(gpu, True)
    logical_gpus = tf.config.list_logical_devices('GPU')
    print(len(gpus), "Physical GPUs,", len(logical_gpus), "Logical GPUs")
  except RuntimeError as e:
```

```
    # Memory growth must be set before GPUs have been initialized
    print(e)
```

```
1 Physical GPUs, 1 Logical GPUs
```

[4]:
```
train = pd.read_csv('fens/train.csv')
val = pd.read_csv('fens/val.csv')
```

[5]:
```
train_dense_gen = DenseGenerator(train, batch_size=1024)
val_dense_gen = DenseGenerator(val, batch_size=1024)
```

[5]:
```
train_tune_gen = ChessPositionGen(train, batch_size=512)
val_tune_gen = ChessPositionGen(val, batch_size=512)
```

0.1 Multi Layer Perceptron model for comparison with the CNN versions

This model uses basic densely connected layers and uses a slight variation of the previously used data generator with no reshaping of the array.

[6]:
```
dense_model = models.Sequential()
dense_model.add(layers.Dense(832, input_shape=(832,), activation='relu'))
dense_model.add(layers.Dense(64, activation='relu'))
dense_model.add(layers.Dense(64, activation='relu'))
dense_model.add(layers.Dense(1, activation='sigmoid'))
dense_model.compile(optimizer="adam", loss="binary_crossentropy",␣
 ↪metrics=['acc'])
dense_model.summary()

# Fitting the model
dense_history = dense_model.fit(x=train_dense_gen,
                    validation_data=val_dense_gen,
                    # steps_per_epoch=100,
                    epochs=15,
                    callbacks=[earlystop, tensorboard_callback]
                    )
```

```
Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 832)               693056
_________________________________________________________________
dense_1 (Dense)              (None, 64)                53312
_________________________________________________________________
dense_2 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
```

```
Total params: 750,593
Trainable params: 750,593
Non-trainable params: 0
_________________________________________________________________
Epoch 1/15
1977/1977 [==============================] - 2590s 1s/step - loss: 0.4226 - acc:
0.8320 - val_loss: 0.3911 - val_acc: 0.8487
Epoch 2/15
1977/1977 [==============================] - 2598s 1s/step - loss: 0.3766 - acc:
0.8544 - val_loss: 0.3697 - val_acc: 0.8579
Epoch 3/15
1977/1977 [==============================] - 2574s 1s/step - loss: 0.3586 - acc:
0.8620 - val_loss: 0.3572 - val_acc: 0.8626
Epoch 4/15
1977/1977 [==============================] - 2570s 1s/step - loss: 0.3450 - acc:
0.8670 - val_loss: 0.3475 - val_acc: 0.8660
Epoch 5/15
1977/1977 [==============================] - 2565s 1s/step - loss: 0.3335 - acc:
0.8710 - val_loss: 0.3412 - val_acc: 0.8686
Epoch 6/15
1977/1977 [==============================] - 2572s 1s/step - loss: 0.3249 - acc:
0.8740 - val_loss: 0.3370 - val_acc: 0.8703
Epoch 7/15
1977/1977 [==============================] - 2562s 1s/step - loss: 0.3183 - acc:
0.8763 - val_loss: 0.3307 - val_acc: 0.8719
Epoch 8/15
1977/1977 [==============================] - 2564s 1s/step - loss: 0.3129 - acc:
0.8783 - val_loss: 0.3280 - val_acc: 0.8732
Epoch 9/15
1977/1977 [==============================] - 2562s 1s/step - loss: 0.3083 - acc:
0.8798 - val_loss: 0.3256 - val_acc: 0.8743
Epoch 10/15
1977/1977 [==============================] - 2569s 1s/step - loss: 0.3043 - acc:
0.8814 - val_loss: 0.3233 - val_acc: 0.8749
Epoch 11/15
1977/1977 [==============================] - 2570s 1s/step - loss: 0.3007 - acc:
0.8826 - val_loss: 0.3214 - val_acc: 0.8754
Epoch 12/15
1977/1977 [==============================] - 2571s 1s/step - loss: 0.2975 - acc:
0.8838 - val_loss: 0.3194 - val_acc: 0.8766
Epoch 13/15
1977/1977 [==============================] - 2570s 1s/step - loss: 0.2948 - acc:
0.8848 - val_loss: 0.3193 - val_acc: 0.8766
Epoch 14/15
1977/1977 [==============================] - 2570s 1s/step - loss: 0.2919 - acc:
0.8857 - val_loss: 0.3180 - val_acc: 0.8761
Epoch 15/15
1977/1977 [==============================] - 2568s 1s/step - loss: 0.2897 - acc:
```

```
0.8864 - val_loss: 0.3165 - val_acc: 0.8773
```

[9]:
```
# dense_model.save('MLPmodel-Long-PB')
```

```
INFO:tensorflow:Assets written to: DenseModel-PB/assets
```

Initially this model was only trained for 15 epochs, then when it appeared that it might be roughly comparable with the CNN model, it was trained for another 15 (early stopped after 11, 26 epochs in total) for a more direct comparison.

[10]:
```
dense_history = dense_model.fit(x=train_dense_gen,
                    validation_data=val_dense_gen,
                    epochs=15,
                    callbacks=[earlystop, tensorboard_callback])
```

```
Epoch 1/15
1977/1977 [==============================] - 2595s 1s/step - loss: 0.2872 - acc:
0.8874 - val_loss: 0.3181 - val_acc: 0.8767
Epoch 2/15
1977/1977 [==============================] - 2590s 1s/step - loss: 0.2852 - acc:
0.8881 - val_loss: 0.3158 - val_acc: 0.8776ETA: 2:49 - l - ETA: 22s
Epoch 3/15
1977/1977 [==============================] - 2582s 1s/step - loss: 0.2833 - acc:
0.8888 - val_loss: 0.3150 - val_acc: 0.8778
Epoch 4/15
1977/1977 [==============================] - 2581s 1s/step - loss: 0.2816 - acc:
0.8895 - val_loss: 0.3151 - val_acc: 0.8781
Epoch 5/15
1977/1977 [==============================] - 2569s 1s/step - loss: 0.2800 - acc:
0.8899 - val_loss: 0.3153 - val_acc: 0.8774
Epoch 6/15
1977/1977 [==============================] - 2583s 1s/step - loss: 0.2784 - acc:
0.8904 - val_loss: 0.3144 - val_acc: 0.8785
Epoch 7/15
1977/1977 [==============================] - 2601s 1s/step - loss: 0.2770 - acc:
0.8909 - val_loss: 0.3159 - val_acc: 0.8780
Epoch 8/15
1977/1977 [==============================] - 2595s 1s/step - loss: 0.2756 - acc:
0.8914 - val_loss: 0.3156 - val_acc: 0.8782
Epoch 9/15
1977/1977 [==============================] - 2589s 1s/step - loss: 0.2743 - acc:
0.8919 - val_loss: 0.3155 - val_acc: 0.8783
Epoch 10/15
1977/1977 [==============================] - 2587s 1s/step - loss: 0.2731 - acc:
0.8923 - val_loss: 0.3175 - val_acc: 0.8785
Epoch 11/15
1977/1977 [==============================] - 2588s 1s/step - loss: 0.2721 - acc:
0.8926 - val_loss: 0.3160 - val_acc: 0.8781
```

```
Restoring model weights from the end of the best epoch.
Epoch 00011: early stopping
```

[12]:
```
# dense_model.save('MLPmodel-Long.h5')
```

0.2 Autotuning

For further information on the Keras autotuner, consult the documentation.

[6]:
```
def model_builder(hp):
    """
    Autotuner modeling function, based off the CNN model from the␣
 ↪full-puzzle-model notebook.
    """
    model = models.Sequential()

    # Set Convolutional layer parameters
    hp_filters = hp.Int('filters', min_value=16, max_value=128, step=8)
    hp_ksize = hp.Int('kernel_size', min_value=2, max_value=8, step=2)

    model.add(layers.Conv2D(filters=hp_filters, kernel_size=hp_ksize,␣
 ↪padding='same', input_shape=(8,8,13), activation='relu'))
    model.add(layers.MaxPooling2D(2))
    model.add(layers.Conv2D(filters=hp_filters, kernel_size=hp_ksize,␣
 ↪padding='same', activation='relu'))
    model.add(layers.Flatten())

    # Add dense tuning parameters
    hp_units = hp.Int('units', min_value=16, max_value=128, step=8)
    model.add(layers.Dense(units=hp_units, activation='relu'))
    model.add(layers.Dense(1, activation='sigmoid'))
    model.compile(optimizer="adam", loss="binary_crossentropy", metrics=['acc'])

    return model

tuner = kt.Hyperband(model_builder, objective='val_acc', max_epochs=10,␣
 ↪directory='tuner', project_name='CNN_tuning')
```

With the `model_builder` function and tuner created, let's start the search. The steps per epoch have been reduced in order to finish searching in a reasonable amount of time.

[7]:
```
tuner.search(x=train_tune_gen, validation_data=val_tune_gen,␣
 ↪steps_per_epoch=1000, callbacks=[earlystop])

best_hps=tuner.get_best_hyperparameters()[0]
print(best_hps)
```

```
Trial 30 Complete [02h 09m 36s]
```

```
val_acc: 0.8775654435157776

Best val_acc So Far: 0.8832182884216309
Total elapsed time: 02h 04m 53s
INFO:tensorflow:Oracle triggered exit
<keras_tuner.engine.hyperparameters.HyperParameters object at 0x7f4850260a90>
```

With parameters found, we can now train our best model.

[16]:
```
tuned_model = tuner.get_best_models(num_models=1)[0]
tuned_model.build()
tuned_model.summary()

tuned_history = tuned_model.fit(x=train_tune_gen,
                    validation_data=val_tune_gen,
                    epochs=30,
                    callbacks=[earlystop, tensorboard_callback])
```

```
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_1
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.beta_2
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
WARNING:tensorflow:Unresolved object in checkpoint:
(root).optimizer.learning_rate
WARNING:tensorflow:A checkpoint was restored (e.g. tf.train.Checkpoint.restore
or tf.keras.Model.load_weights) but not all checkpointed values were used. See
above for specific issues. Use expect_partial() on the load status object, e.g.
tf.train.Checkpoint.restore(…).expect_partial(), to silence these warnings, or
use assert_consumed() to make the check explicit. See
https://www.tensorflow.org/guide/checkpoint#loading_mechanics for details.
Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 8, 8, 120)         99960
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 4, 4, 120)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 4, 4, 120)         921720
_________________________________________________________________
flatten (Flatten)            (None, 1920)              0
_________________________________________________________________
dense (Dense)                (None, 96)                184416
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 97
=================================================================
Total params: 1,206,193
Trainable params: 1,206,193
```

```
Non-trainable params: 0
_________________________________________________________________
Epoch 1/30
3953/3953 [==============================] - 2677s 677ms/step - loss: 0.2935 -
acc: 0.8893 - val_loss: 0.2870 - val_acc: 0.8919
Epoch 2/30
3953/3953 [==============================] - 2696s 682ms/step - loss: 0.2733 -
acc: 0.8959 - val_loss: 0.2728 - val_acc: 0.8965
Epoch 3/30
3953/3953 [==============================] - 2697s 682ms/step - loss: 0.2610 -
acc: 0.9002 - val_loss: 0.2653 - val_acc: 0.8992
Epoch 4/30
3953/3953 [==============================] - 2693s 681ms/step - loss: 0.2524 -
acc: 0.9032 - val_loss: 0.2617 - val_acc: 0.9006
Epoch 5/30
3953/3953 [==============================] - 2691s 681ms/step - loss: 0.2457 -
acc: 0.9055 - val_loss: 0.2578 - val_acc: 0.9014
Epoch 6/30
3953/3953 [==============================] - 2684s 679ms/step - loss: 0.2399 -
acc: 0.9074 - val_loss: 0.2564 - val_acc: 0.9021
Epoch 7/30
3953/3953 [==============================] - 2691s 681ms/step - loss: 0.2353 -
acc: 0.9091 - val_loss: 0.2547 - val_acc: 0.9031
Epoch 8/30
3953/3953 [==============================] - 2691s 681ms/step - loss: 0.2310 -
acc: 0.9106 - val_loss: 0.2568 - val_acc: 0.9032
Epoch 9/30
3953/3953 [==============================] - 2688s 680ms/step - loss: 0.2268 -
acc: 0.9120 - val_loss: 0.2552 - val_acc: 0.9033
Epoch 10/30
3953/3953 [==============================] - 2685s 679ms/step - loss: 0.2233 -
acc: 0.9131 - val_loss: 0.2560 - val_acc: 0.9034
Epoch 11/30
3953/3953 [==============================] - 2686s 680ms/step - loss: 0.2201 -
acc: 0.9142 - val_loss: 0.2550 - val_acc: 0.9034
Epoch 12/30
3953/3953 [==============================] - 2668s 675ms/step - loss: 0.2174 -
acc: 0.9151 - val_loss: 0.2561 - val_acc: 0.9033
Restoring model weights from the end of the best epoch.
Epoch 00012: early stopping
```

[18]:
```
# tuned_model.save("tuned_model-PB")
```

```
INFO:tensorflow:Assets written to: tuned_model-PB/assets
```