full-puzzle-model

March 10, 2022

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

import datetime
%load_ext tensorboard
```

We'll set early stopping and tensorboard call backs.

[3]:
```
# 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/full_puzzle" + datetime.datetime.now().
 ↪strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1)
```

[4]:
```
# 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
```

Train test split and data cleaning was performed in the `dataorg` notebook, so we can load our standard cleaned data. As a reminder, this is a list of FEN positions with the target as being the correct move. The data generator will turn this into 5 different potential positions, with the correct move labeled 1 and all other labeled 0.

[5]:
```
train = pd.read_csv('fens/train.csv')
val = pd.read_csv('fens/val.csv')
```

[6]:
```
train_gen = ChessPositionGen(train, batch_size=512)
val_gen = ChessPositionGen(val, batch_size=512)
```

The CNN is pretty much the same as was used in `misc-notebooks/baseline-models.ipynb`, but with the improved data generator, it can be trained on our full training set.

[7]:
```
full_puzzle_model = models.Sequential()
full_puzzle_model.add(layers.Conv2D(64, 4, padding='same',␣
 ↪input_shape=(8,8,13), activation='relu'))
full_puzzle_model.add(layers.MaxPooling2D(2))
full_puzzle_model.add(layers.Conv2D(32, 2, padding='same', activation='relu'))
full_puzzle_model.add(layers.Flatten())
full_puzzle_model.add(layers.Dense(64, activation='relu'))
full_puzzle_model.add(layers.Dense(1, activation='sigmoid'))
full_puzzle_model.compile(optimizer="adam", loss="binary_crossentropy",␣
 ↪metrics=['acc'])
full_puzzle_model.summary()

# Fitting the model
full_puzzle_history = full_puzzle_model.fit(x=train_gen,
                    validation_data=val_gen,
                    epochs=30,
                    callbacks=[earlystop, tensorboard_callback]
                    )
```

```
Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 8, 8, 64)          13376

_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 4, 4, 64)          0

_________________________________________________________________
conv2d_1 (Conv2D)            (None, 4, 4, 32)          8224

_________________________________________________________________
flatten (Flatten)            (None, 512)               0
```

```
_________________________________________________________________
dense (Dense)                (None, 64)                32832
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 54,497
Trainable params: 54,497
Non-trainable params: 0
_________________________________________________________________
Epoch 1/30
3953/3953 [==============================] - 2697s 678ms/step - loss: 0.4042 -
acc: 0.8451 - val_loss: 0.3744 - val_acc: 0.8628
Epoch 2/30
3953/3953 [==============================] - 2702s 683ms/step - loss: 0.3677 -
acc: 0.8653 - val_loss: 0.3647 - val_acc: 0.8667
Epoch 3/30
3953/3953 [==============================] - 2704s 684ms/step - loss: 0.3590 -
acc: 0.8691 - val_loss: 0.3559 - val_acc: 0.8703
Epoch 4/30
3953/3953 [==============================] - 2688s 680ms/step - loss: 0.3524 -
acc: 0.8716 - val_loss: 0.3518 - val_acc: 0.8712
Epoch 5/30
3953/3953 [==============================] - 2679s 678ms/step - loss: 0.3468 -
acc: 0.8731 - val_loss: 0.3440 - val_acc: 0.8737
Epoch 6/30
3953/3953 [==============================] - 2687s 680ms/step - loss: 0.3401 -
acc: 0.8744 - val_loss: 0.3381 - val_acc: 0.8745
Epoch 7/30
3953/3953 [==============================] - 2703s 684ms/step - loss: 0.3349 -
acc: 0.8755 - val_loss: 0.3356 - val_acc: 0.8748
Epoch 8/30
3953/3953 [==============================] - 2690s 680ms/step - loss: 0.3313 -
acc: 0.8762 - val_loss: 0.3298 - val_acc: 0.8769
Epoch 9/30
3953/3953 [==============================] - 2694s 682ms/step - loss: 0.3282 -
acc: 0.8772 - val_loss: 0.3319 - val_acc: 0.8762
Epoch 10/30
3953/3953 [==============================] - 2684s 679ms/step - loss: 0.3259 -
acc: 0.8778 - val_loss: 0.3253 - val_acc: 0.8781
Epoch 11/30
3953/3953 [==============================] - 2692s 681ms/step - loss: 0.3235 -
acc: 0.8785 - val_loss: 0.3230 - val_acc: 0.8786
Epoch 12/30
3953/3953 [==============================] - 2687s 679ms/step - loss: 0.3215 -
acc: 0.8791 - val_loss: 0.3235 - val_acc: 0.8787
Epoch 13/30
3953/3953 [==============================] - 2700s 683ms/step - loss: 0.3195 -
acc: 0.8796 - val_loss: 0.3200 - val_acc: 0.8794
```

```
Epoch 14/30
3953/3953 [==============================] - 2679s 678ms/step - loss: 0.3180 -
acc: 0.8801 - val_loss: 0.3189 - val_acc: 0.8796
Epoch 15/30
3953/3953 [==============================] - 2693s 681ms/step - loss: 0.3164 -
acc: 0.8807 - val_loss: 0.3173 - val_acc: 0.8808
Epoch 16/30
3953/3953 [==============================] - 2673s 676ms/step - loss: 0.3153 -
acc: 0.8810 - val_loss: 0.3183 - val_acc: 0.8804
Epoch 17/30
3953/3953 [==============================] - 2664s 674ms/step - loss: 0.3143 -
acc: 0.8813 - val_loss: 0.3172 - val_acc: 0.8806
Epoch 18/30
3953/3953 [==============================] - 2667s 674ms/step - loss: 0.3130 -
acc: 0.8817 - val_loss: 0.3155 - val_acc: 0.8818
Epoch 19/30
3953/3953 [==============================] - 2673s 676ms/step - loss: 0.3121 -
acc: 0.8819 - val_loss: 0.3153 - val_acc: 0.8815
Epoch 20/30
3953/3953 [==============================] - 2666s 674ms/step - loss: 0.3112 -
acc: 0.8822 - val_loss: 0.3128 - val_acc: 0.8822
Epoch 21/30
3953/3953 [==============================] - 2671s 676ms/step - loss: 0.3104 -
acc: 0.8826 - val_loss: 0.3150 - val_acc: 0.8815
Epoch 22/30
3953/3953 [==============================] - 2685s 679ms/step - loss: 0.3093 -
acc: 0.8829 - val_loss: 0.3121 - val_acc: 0.8819
Epoch 23/30
3953/3953 [==============================] - 2686s 679ms/step - loss: 0.3086 -
acc: 0.8831 - val_loss: 0.3099 - val_acc: 0.8831
Epoch 24/30
3953/3953 [==============================] - 2691s 681ms/step - loss: 0.3080 -
acc: 0.8834 - val_loss: 0.3114 - val_acc: 0.8827
Epoch 25/30
3953/3953 [==============================] - 2684s 679ms/step - loss: 0.3072 -
acc: 0.8836 - val_loss: 0.3091 - val_acc: 0.8831
Epoch 26/30
3953/3953 [==============================] - 2680s 678ms/step - loss: 0.3066 -
acc: 0.8837 - val_loss: 0.3080 - val_acc: 0.8839
Epoch 27/30
3953/3953 [==============================] - 2684s 679ms/step - loss: 0.3060 -
acc: 0.8840 - val_loss: 0.3106 - val_acc: 0.8828
Epoch 28/30
3953/3953 [==============================] - 2682s 678ms/step - loss: 0.3055 -
acc: 0.8842 - val_loss: 0.3084 - val_acc: 0.8836
Epoch 29/30
3953/3953 [==============================] - 2667s 675ms/step - loss: 0.3050 -
acc: 0.8844 - val_loss: 0.3071 - val_acc: 0.8839
```

```
Epoch 30/30
3953/3953 [==============================] - 2677s 677ms/step - loss: 0.3045 -
acc: 0.8844 - val_loss: 0.3108 - val_acc: 0.8833
```

```
[10]: # full_puzzle_model.save('full-puzzle-PB')
```

```
INFO:tensorflow:Assets written to: full-puzzle-PB/assets
```

```
[9]: # test = pd.read_csv('fens/test.csv')
     # test_gen = ChessPositionGen(test)
```

```
[ ]: %tensorboard --logdir logs/fit
```