Cutting Edge TensorFlow

Keras Tuner: hypertuning for humans

Elie Bursztein
Google, @elie
How do I get the best TensorFlow model?
Getting the optimal model requires to tune many inter-dependent parameters.
What if... hypertuning was as easy as 1, 2, 3?
Introducing Keras Tuner
Hypertuning for humans
Keras Tuner is a framework designed for:

- AI practitioners
- Hypertuner algorithm creators
- Model designers
With Keras Tuner, hyperparameter tuning is just a few lines of code away.
A basic MNIST model

```python
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(20, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer=Adam(0.001))
model.summary()
```
MNIST hypermodel is as easy as 1,2,3

1. Wrap model in a function
   def model_fn():
MNIST hypermodel is as easy as 1,2,3

1. Wrap model in a function
   ```python
   def model_fn():
   ```
   ```python
   LR = Choice('learning_rate', [0.001, 0.0005, 0.0001], group='optimizer')
   DROPOUT_RATE = Linear('dropout_rate', 0.0, 0.5, 5, group='dense')
   NUM_DIMS = Range('num_dims', 8, 32, 8, group='dense')
   NUM_LAYERS = Range('num_layers', 1, 3, group='dense')
   L2_NUM_FILTERS = Range('l2_num_filters', 8, 64, 8, group='cnn')
   L1_NUM_FILTERS = Range('l1_num_filters', 8, 64, 8, group='cnn')
   ```

2. Define hyper-parameters
MNIST hypermodel is as easy as 1,2,3

1. Wrap model in a function

   ```python
   def model_fn():
   ```

2. Define hyper-parameters

   ```python
   LR = Choice('learning_rate', [0.001, 0.0005, 0.0001], group='optimizer')
   DROPOUT_RATE = Linear('dropout_rate', 0.0, 0.5, 5, group='dense')
   NUM_DIMS = Range('num_dims', 8, 32, 8, group='dense')
   NUM_LAYERS = Range('num_layers', 1, 3, group='dense')
   L2_NUM_FILTERS = Range('l2_num_filters', 8, 64, 8, group='cnn')
   L1_NUM_FILTERS = Range('l1_num_filters', 8, 64, 8, group='cnn')
   ```

3. Replace static value with hyper-parameters

   ```python
   model = Sequential()
   model.add(Conv2D(L1_NUM_FILTERS, kernel_size=(3, 3), activation='relu'))
   model.add(Conv2D(L2_NUM_FILTERS, kernel_size=(3, 3), activation='relu'))
   model.add(Flatten())
   for _ in range(NUM_LAYERS):
     model.add(Dense(NUM_DIMS, activation='relu'))
   model.add(Dropout(DROPOUT_RATE))
   model.add(Dense(10, activation='softmax'))
   model.compile(loss='categorical_crossentropy', optimizer=Adam(LR))
   return model
   ```
MNIST hypermodel is as easy as 1,2,3

1. Wrap model in a function

   ```python
   def model_fn():
   ```

2. Define hyper-parameters

   ```python
   LR = Choice('learning_rate', [0.001, 0.0005, 0.0001], group='optimizer')
   DROPOUT_RATE = Linear('dropout_rate', 0.0, 0.5, 5, group='dense')
   NUM_DIMS = Range('num_dims', 8, 32, 8, group='dense')
   NUM_LAYERS = Range('num_layers', 1, 3, 1, group='dense')
   L2_NUM_FILTERS = Range('l2_num_filters', 8, 64, 8, group='cnn')
   L1_NUM_FILTERS = Range('l1_num_filters', 8, 64, 8, group='cnn')
   ```

3. Replace static value with hyper-parameters

   ```python
   model = Sequential()
   model.add(Conv2D(L1_NUM_FILTERS, kernel_size=(3, 3), activation='relu'))
   model.add(Conv2D(L2_NUM_FILTERS, kernel_size=(3, 3), activation='relu'))
   model.add(Flatten())
   for _ in range(NUM_LAYERS):
     model.add(Dense(NUM_DIMS, activation='relu'))
     model.add(Dropout(DROPOUT_RATE))
   model.add(Dense(10, activation='softmax'))
   model.compile(loss='categorical_crossentropy', optimizer=Adam(LR))
   return model
   ```
Intuitive API

State of the art hypertuner algorithms

Tunable architectures ready to go

Seamless experiments recording
Let's build a simple logo classifier!
from icongenerator.dataset import io19

ICON_SIZE = 100
NUM_EPOCHS = 5
BATCH_SIZE = 128
NUM_GEN_ICONS_PER_EPOCH = 50000
dataset = io19.download()
from icongenerator import TFGenerator

AUGMENTATION_PARAMS = {
    "background_frequency": 0.5,
    "shape_frequency": 0.5,
    "min_logo_size": 0.50,
    "max_logo_size": 0.95,
    "use_flips": True,
    "occlusion": "light",
    "use_logo_color_shift": True,
    "blur": 1.0,
    "max_rotation": 45,
    "max_affine_transformations": 2,
    "use_background_color_shift": True,
    "use_shape_color_shift": True,
    "min_logo_alpha": 230
}

tfg = TFGenerator(dataset, icon_size=ICON_SIZE, **AUGMENTATION_CONFIG)
Training ResNet101v2 as baseline

```python
base_model = ResNet101v2(input_shape=(100, 100, 3), weights=None, include_top=False)
x = GlobalAveragePooling2D()(base_model.output)

y = Dense(100 * 100 * 3, activation='sigmoid')(x)
decoded = Reshape((100, 100, 3), name='reconstruction')(y)

x = Flatten()(x)
prediction = Dense(kg.num_classes, activation='softmax', name='label')(x)
model = Model(base_model.input, [prediction, decoded])

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

model.fit_generator(kg, epochs=NUM_EPOCHS, validation_data=kg.val_data, callbacks=callbacks)
```
Training worked but model accuracy is low: 79.6% and it’s big: 44M parameters

Let’s use Keras Tuner to find a more accurate model and smaller model!
Keras Tuner TunableResNet

Use tunable ResNet as base

```python
def model_fn():
    base_model = TunableResNet(input_shape=(100, 100, 3))

    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    y = Dense(100 * 100 * 3, activation='sigmoid')(x)
    decoded = Reshape((100, 100, 3), name='reconstruction')(y)
    prediction = Dense(generator.num_classes, activation='softmax')(x)

    model = Model(base_model.input, [prediction, decoded])

    model.compile(optimizer=optimizers.Adam(LEARNING_RATE),
                  loss=['categorical_crossentropy', 'mse'],
                  metrics=['accuracy'])

    return model
```

Customize it to support our multi-head output

Init hypertuner

```python```
tuner = Tuner(model_fn, 'val_accuracy' epoch_budget=500, max_epochs=5)
```

Tuning

```python```
tuner.search(tfg, validation_data=validation_data)
```
```
Yay! Keras Tuner found a better model with 100% accuracy (+20%) and only 24M parameters (-45%)

Dataset is small so there is a possibility of overfit despite using augmented icons in training
Ecosystem integration

TensorBoard    Colab    BigQuery    Command line    More to come!
One more thing!

Because everyone like surprises
Online dashboard

Monitoring on the go!
Hypertuning

Overall epoch budget
- 2h 51m 2s left
- 848/1000

Current model
- 2m 25s left
- 48/50

Overview

Trained models

<table>
<thead>
<tr>
<th>objective: val_acc</th>
<th>model size</th>
<th>learning rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000000</td>
<td>24,517,863</td>
<td>0.1</td>
</tr>
<tr>
<td>1.000000</td>
<td>25,133,015</td>
<td>0.1</td>
</tr>
<tr>
<td>1.000000</td>
<td>25,694,167</td>
<td>0.01</td>
</tr>
<tr>
<td>1.000000</td>
<td>48,058,327</td>
<td>0.01</td>
</tr>
<tr>
<td>1.000000</td>
<td>46,936,023</td>
<td>0.1</td>
</tr>
<tr>
<td>0.968750</td>
<td>33,517,527</td>
<td>0.01</td>
</tr>
<tr>
<td>0.968750</td>
<td>35,200,983</td>
<td>0.01</td>
</tr>
<tr>
<td>0.968750</td>
<td>29,491,735</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Sign up today for Keras Tuner alpha to help us making it awesome!

https://g.co/research/kerastunereap