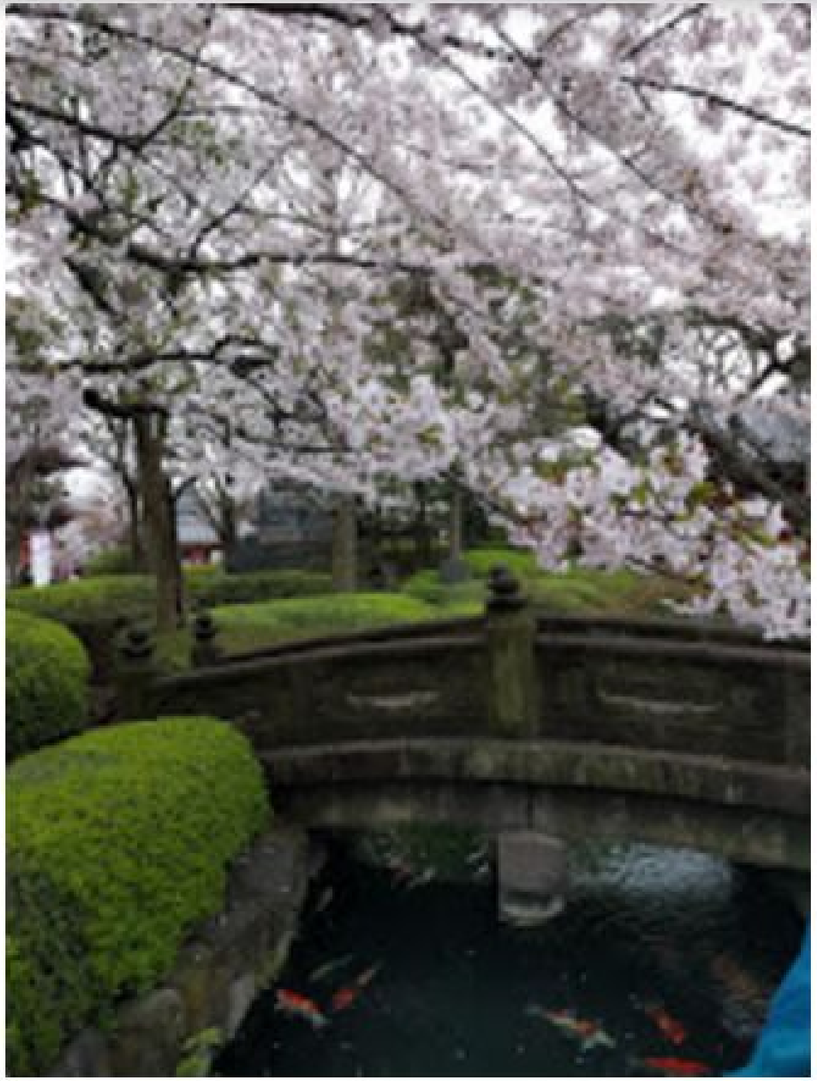
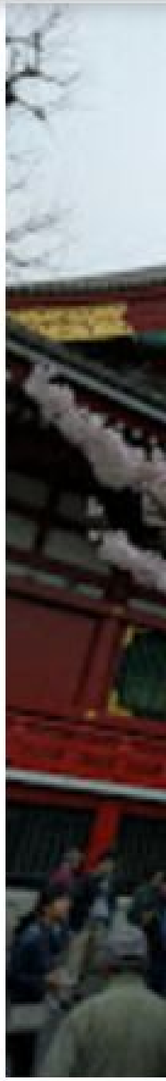
Effective TensorFlow for Non-Experts

Baohua Liao
liaobaohua@gmail.com

#io17

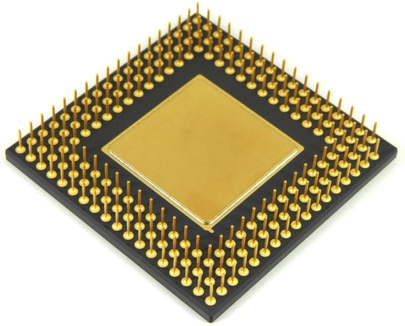
cherry blossom



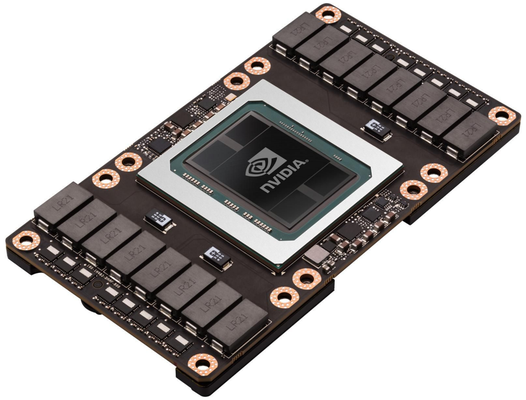



EXIT

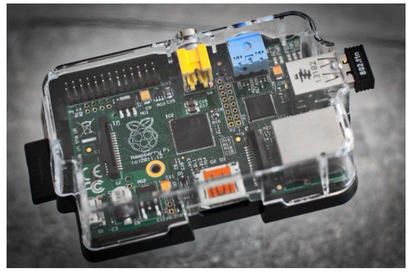
Complexity



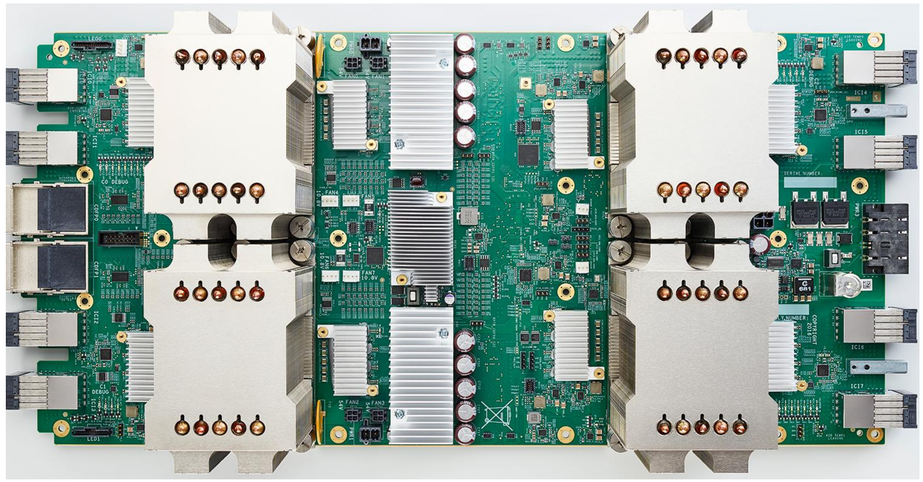
CPU



GPU

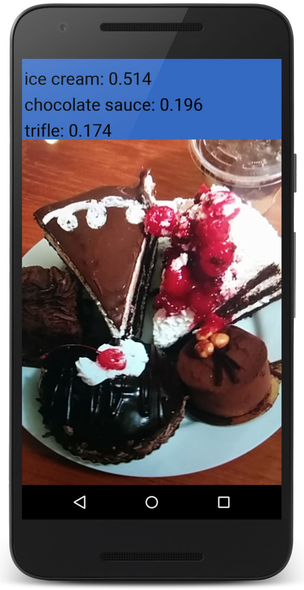


Raspberry Pi

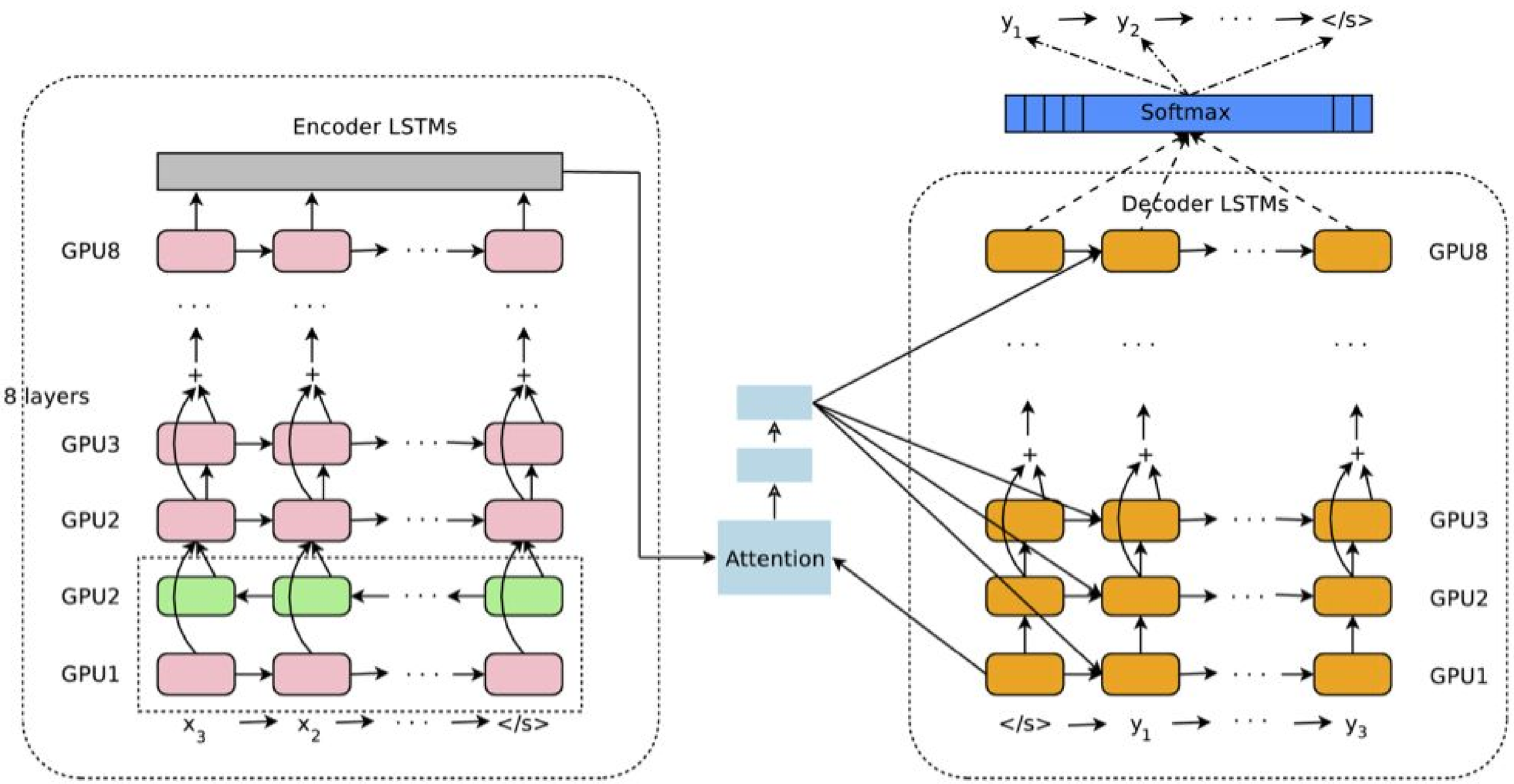


Cloud TPU

iOS



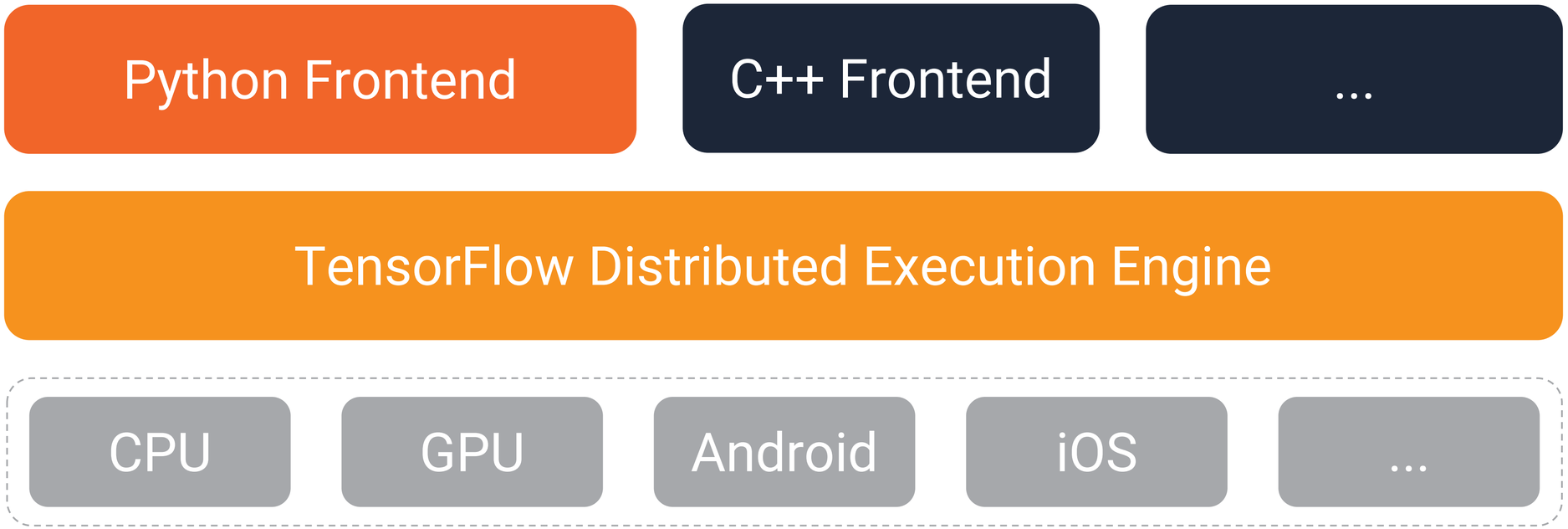
Android

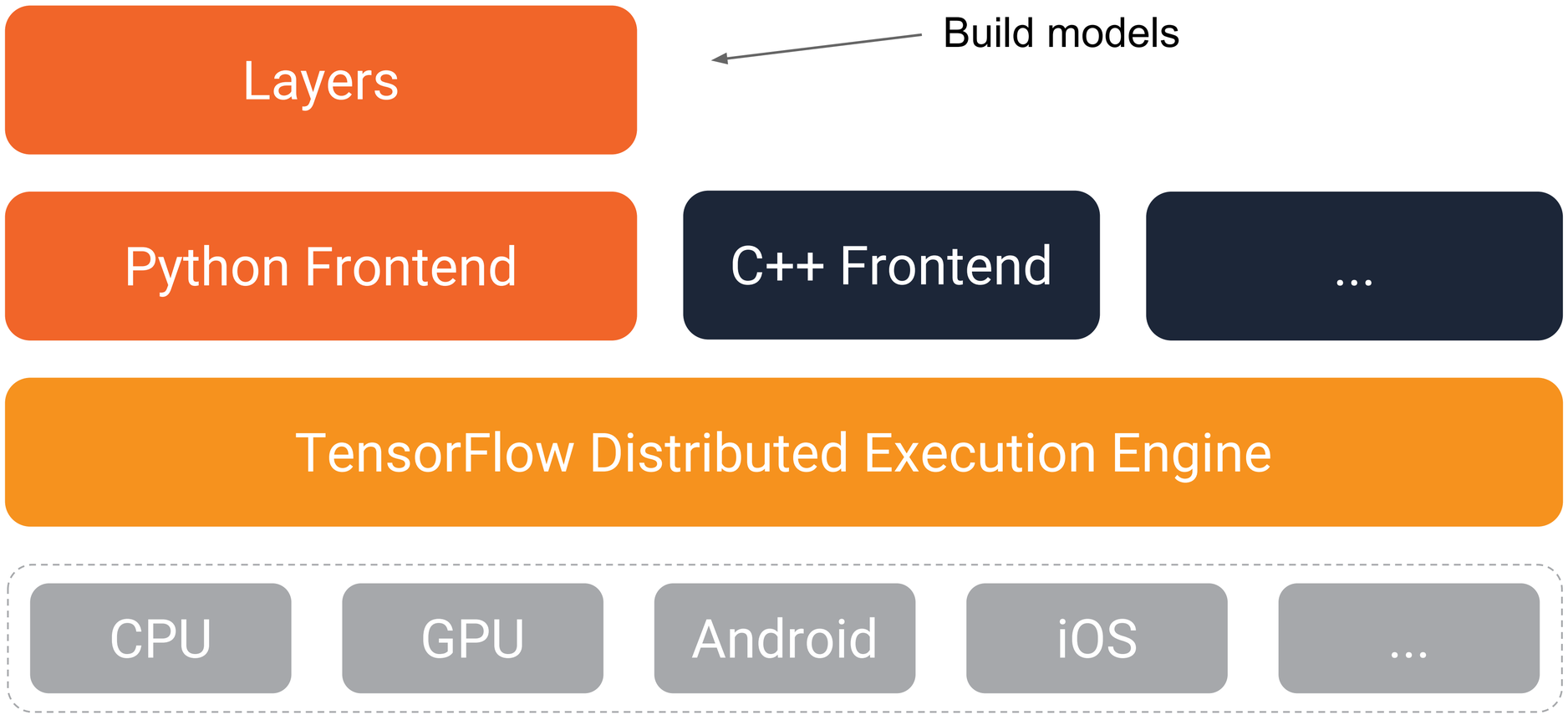

Encoder LSTMs
GPU8
8 layers
GPU3
GPU2
GPU2
GPU1
x3
x2
</s>
Attention
y1
y2
</s>
Softmax
Decoder LSTMs
GPU8
GPU3
GPU2
GPU1
</s>
y1
y3

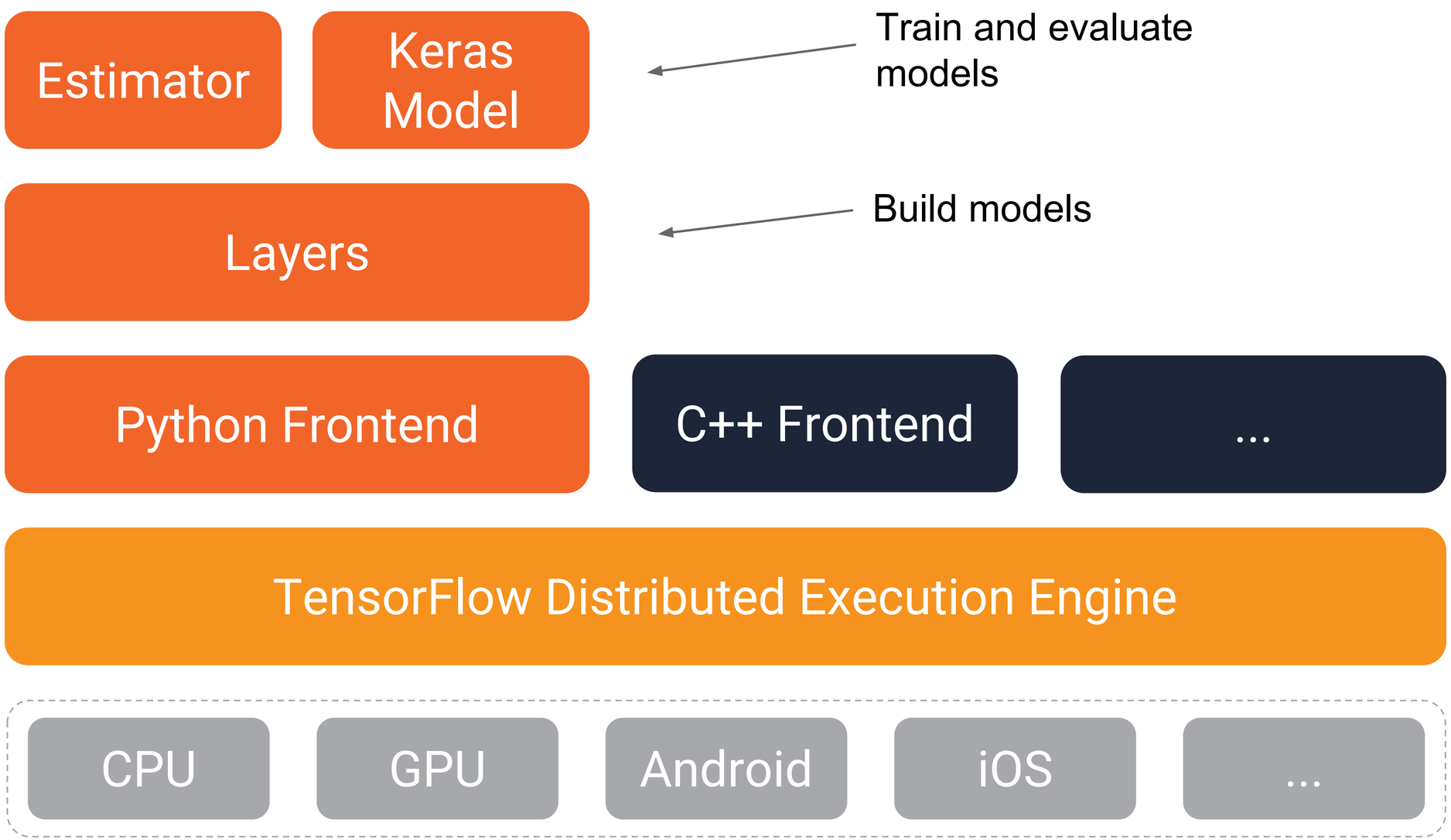
Distributed out of the box

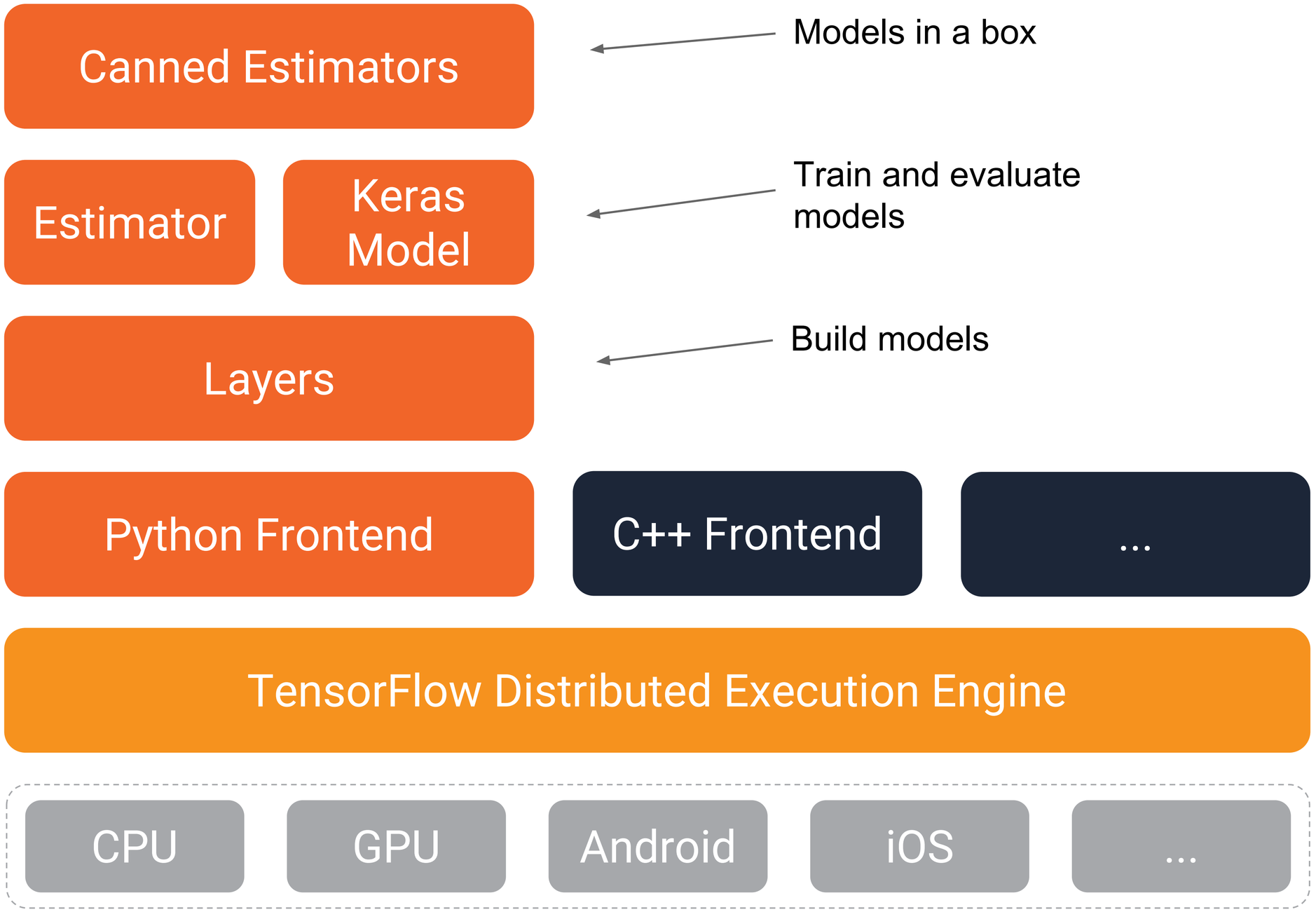
- Distributed out of the box
- Runs on CPU/GPU/TPU/mobile

- Distributed out of the box
- Runs on CPU/GPU/TPU/mobile
- Fast and Flexible


Python Frontend
C++ Frontend
...
TensorFlow Distributed Execution Engine
CPU
GPU
Android
iOS
...


Build models
Layers
Python Frontend
C++ Frontend
...
TensorFlow Distributed Execution Engine
CPU
GPU
Android
iOS
...


Estimator
Keras Model
Train and evaluate models
Layers
Build models
Python Frontend
C++ Frontend
...
TensorFlow Distributed Execution Engine
CPU
GPU
Android
iOS
...


Canned Estimators
Models in a box
Estimator
Keras Model
Train and evaluate models
Layers
Build models
Python Frontend
C++ Frontend
...
TensorFlow Distributed Execution Engine
CPU
GPU
Android
iOS
...

My first Model

goo.gl/0OgXiL

```
make = categorical_column_with_hash_bucket('make', 100)
horsepower = numeric_column('horsepower', shape=[])
cylinders = categorical_column_with_vocabulary_list(
    'num-of-cylinders', ['two', 'three', 'four', ...])
...

regressor = LinearRegressor(feature_columns=[make, horsepower,
                                             cylinders, ...])
```

```
make = categorical_column_with_hash_bucket('make', 100)
horsepower = numeric_column('horsepower', shape=[])
cylinders = categorical_column_with_vocabulary_list(
    'num-of-cylinders', ['two', 'three', 'four', ...])
...

regressor = LinearRegressor(feature_columns=[make, horsepower,
                                             cylinders, ...])
```

```
make = categorical_column_with_hash_bucket('make', 100)
horsepower = numeric_column('horsepower', shape=[])
cylinders = categorical_column_with_vocabulary_list(
    'num-of-cylinders', ['two', 'three', 'four', ...])
...

regressor = LinearRegressor(feature_columns=[make, horsepower,
                                             cylinders, ...])
```

```
...
regressor = LinearRegressor(feature_columns=[make, horsepower,
                                             cylinders, ...])

train_input_fn = pandas_input_fn(x=input_data, y=input_label,
                                 batch_size=64, shuffle=True,
                                 num_epochs=None)
```

```
...
regressor = LinearRegressor(feature_columns=[make, horsepower,
                                             cylinders, ...])

train_input_fn = pandas_input_fn(x=input_data, y=input_label,
                                 batch_size=64, shuffle=True,
                                 num_epochs=None)

regressor.train(train_input_fn, steps=10000)
```

```
INFO:tensorflow:loss = 6.07866e+07, step = 102 (0.884 sec)
INFO:tensorflow:global_step/sec: 131.905
INFO:tensorflow:loss = 7.26533e+07, step = 202 (0.758 sec)
INFO:tensorflow:global_step/sec: 125.879
INFO:tensorflow:loss = 3.64895e+07, step = 302 (0.794 sec)
INFO:tensorflow:global_step/sec: 131.558
INFO:tensorflow:loss = 4.84076e+07, step = 402 (0.760 sec)
INFO:tensorflow:global_step/sec: 124.394
INFO:tensorflow:loss = 4.316e+07, step = 502 (0.804 sec)
INFO:tensorflow:global_step/sec: 126.16
INFO:tensorflow:loss = 2.32498e+07, step = 602 (0.793 sec)
INFO:tensorflow:global_step/sec: 122.751
INFO:tensorflow:loss = 4.53009e+07, step = 702 (0.817 sec)
INFO:tensorflow:global_step/sec: 118.297
INFO:tensorflow:loss = 4.56962e+07, step = 802 (0.843 sec)
INFO:tensorflow:global_step/sec: 126.641
INFO:tensorflow:loss = 3.3779e+07, step = 902 (0.790 sec)
INFO:tensorflow:global_step/sec: 109.258
INFO:tensorflow:loss = 5.32131e+07, step = 1002 (0.915 sec)
INFO:tensorflow:global_step/sec: 135.997
INFO:tensorflow:loss = 3.2445e+07, step = 1102 (0.735 sec)
INFO:tensorflow:global_step/sec: 123.69
INFO:tensorflow:loss = 2.43225e+07, step = 1202 (0.811 sec)
INFO:tensorflow:global_step/sec: 136.363
INFO:tensorflow:loss = 2.86487e+07, step = 1302 (0.730 sec)
INFO:tensorflow:global_step/sec: 129.674
```

Write a regex to create a tag group

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

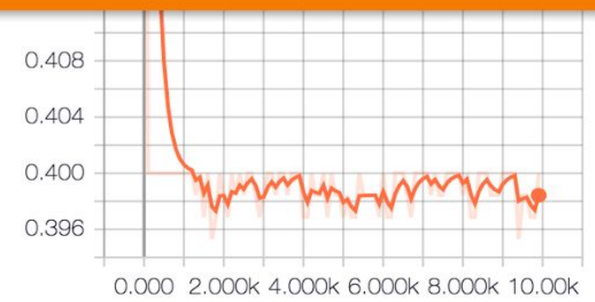
Write a regex to filter runs

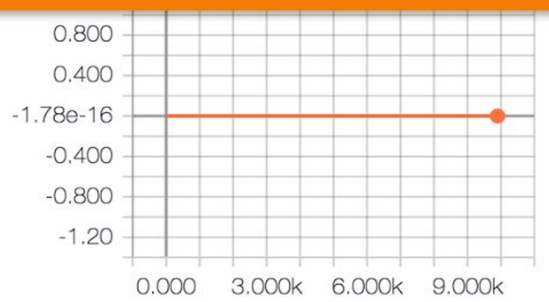
.

eval

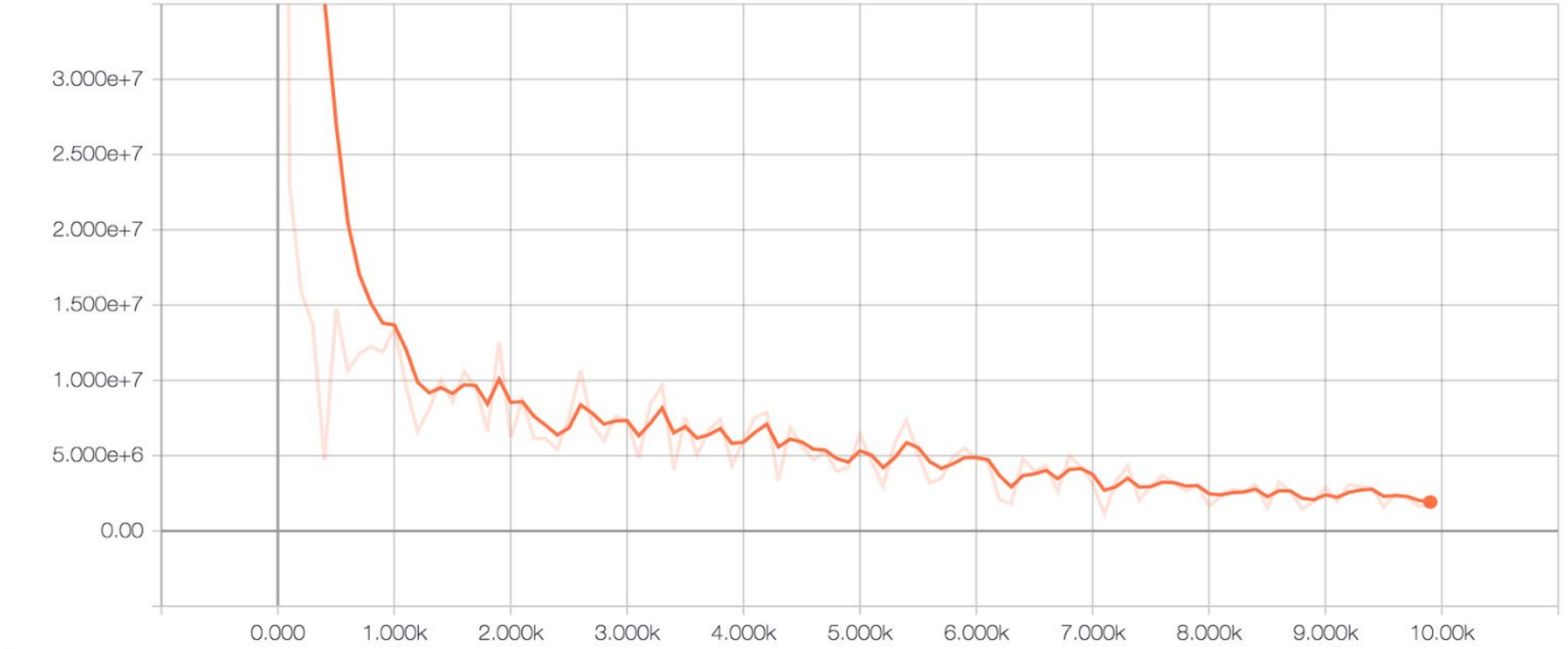
TOGGLE ALL RUNS

/var/folders/qr/4w8_972d0w17wlh9xnlly3y800_glr/T/tmpvSZZhV





dnn/regression_head/loss



enqueue_input 1

Fit to screen

Download PNG

Run (2) .

Session runs (0)

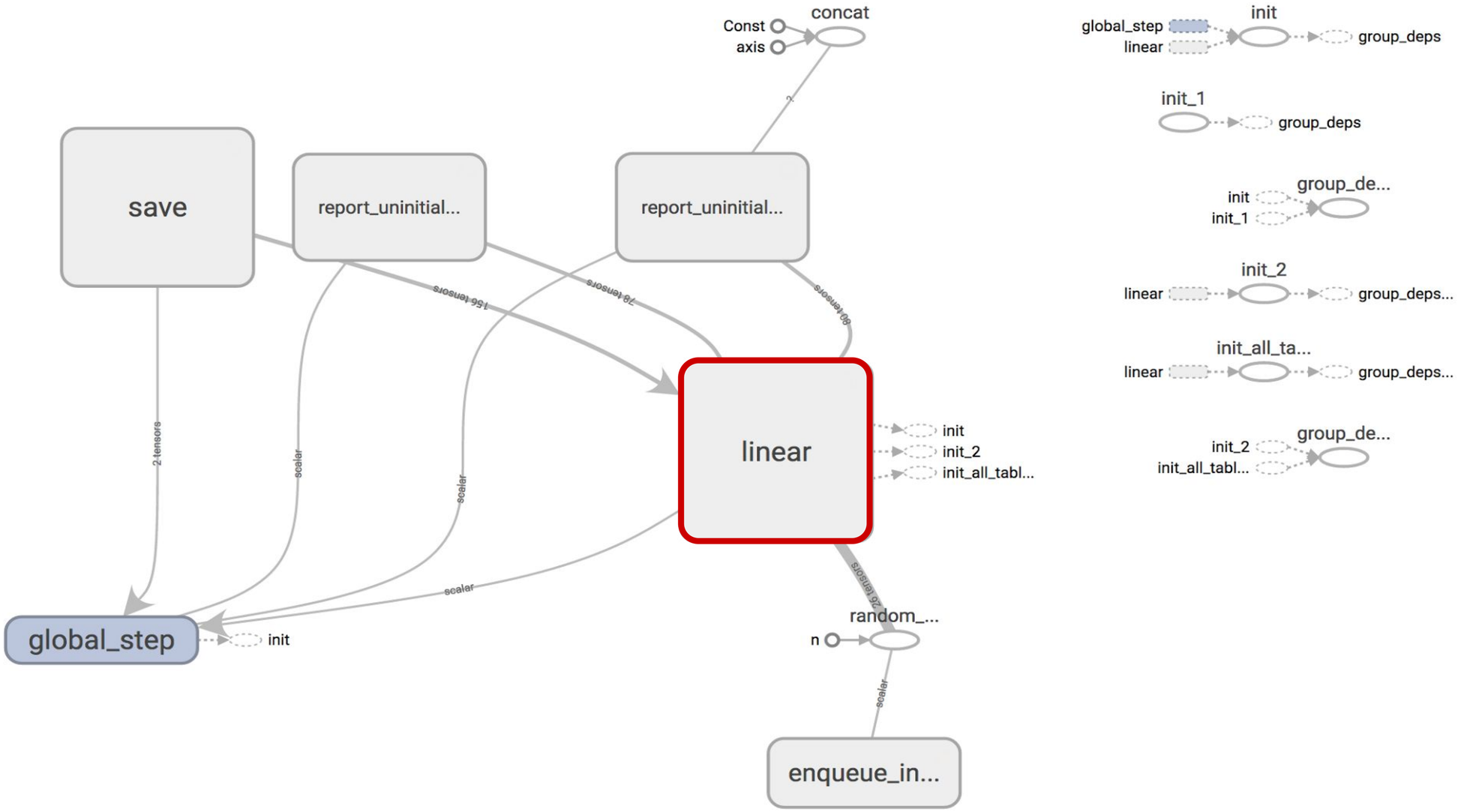
Upload Choose File

Trace inputs

Color
- Structure
- Device
- XLA Cluster
- Compute time
- Memory

colors
- same substructure
- unique substructure

Graph (* = expandable)
- Namespace*
- OpNode
- Unconnected series*
- Connected series*
- Constant
- Summary
- Dataflow edge
- Control dependency edge
- Reference edge



Fit to screen
Download PNG

Run (2) .

Session runs (0)

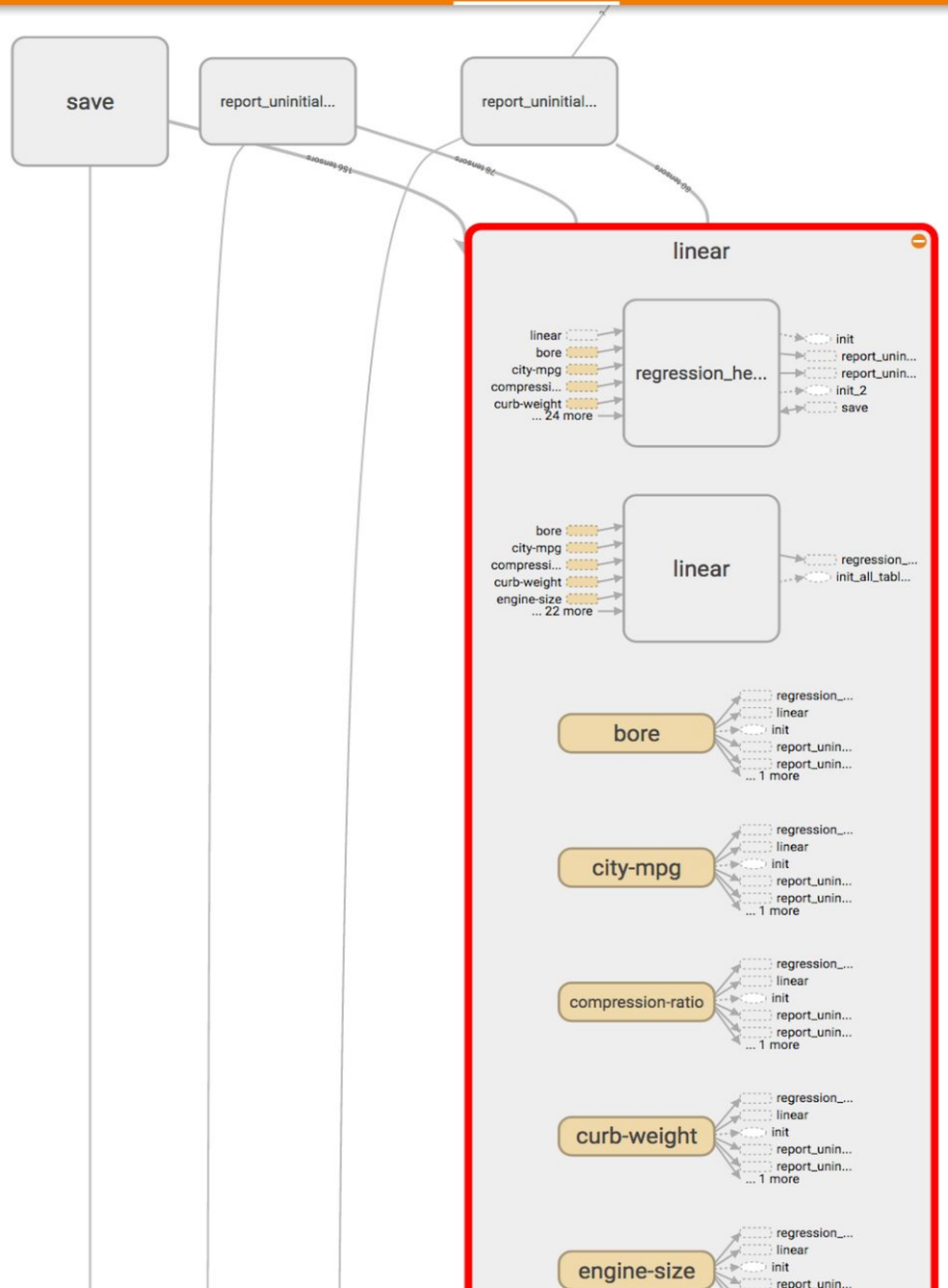
Upload Choose File

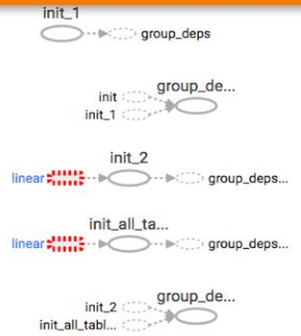
Trace inputs

Color
- Structure
- Device
- XLA Cluster
- Compute time
- Memory

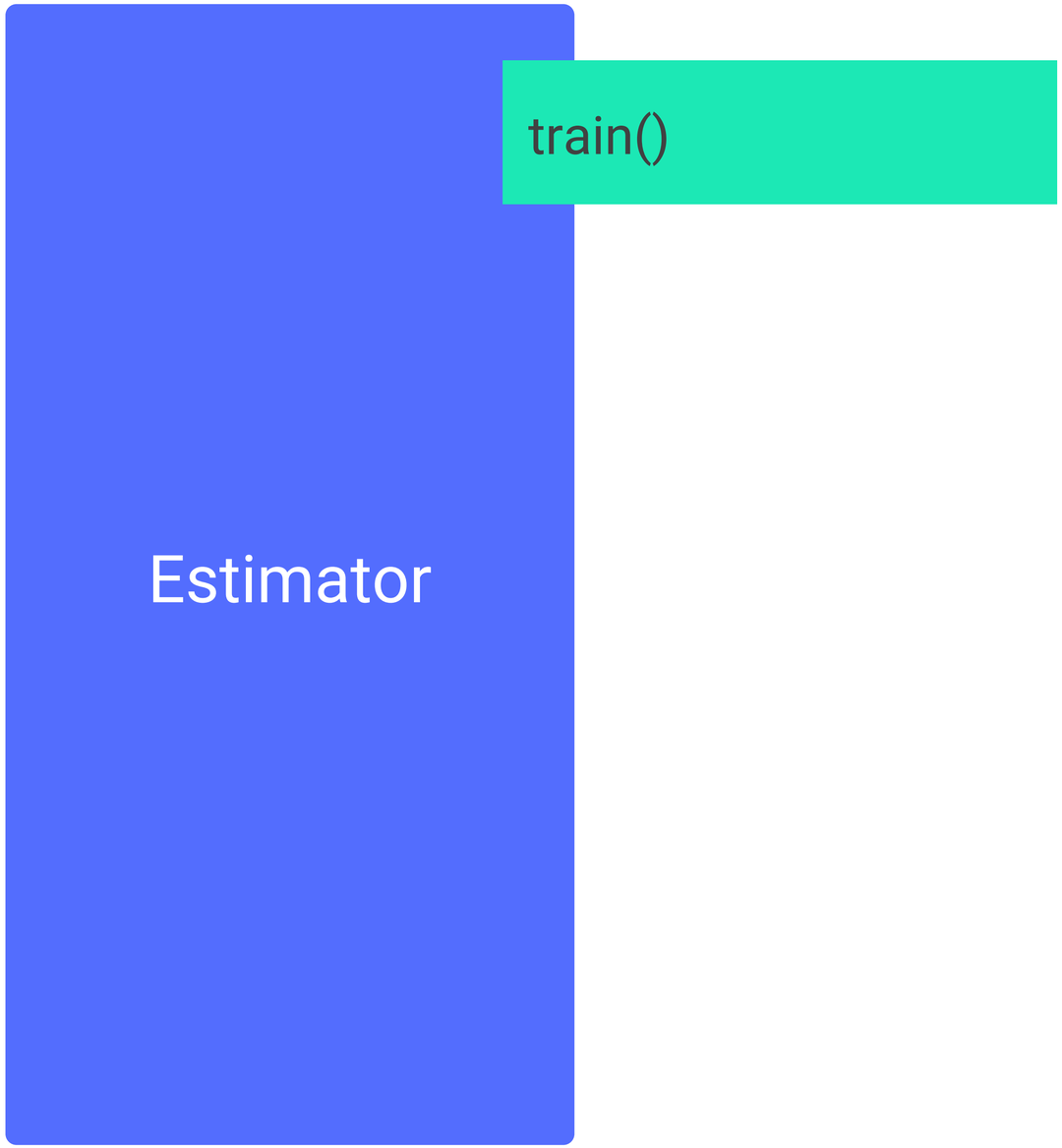
colors same substructure
unique substructure

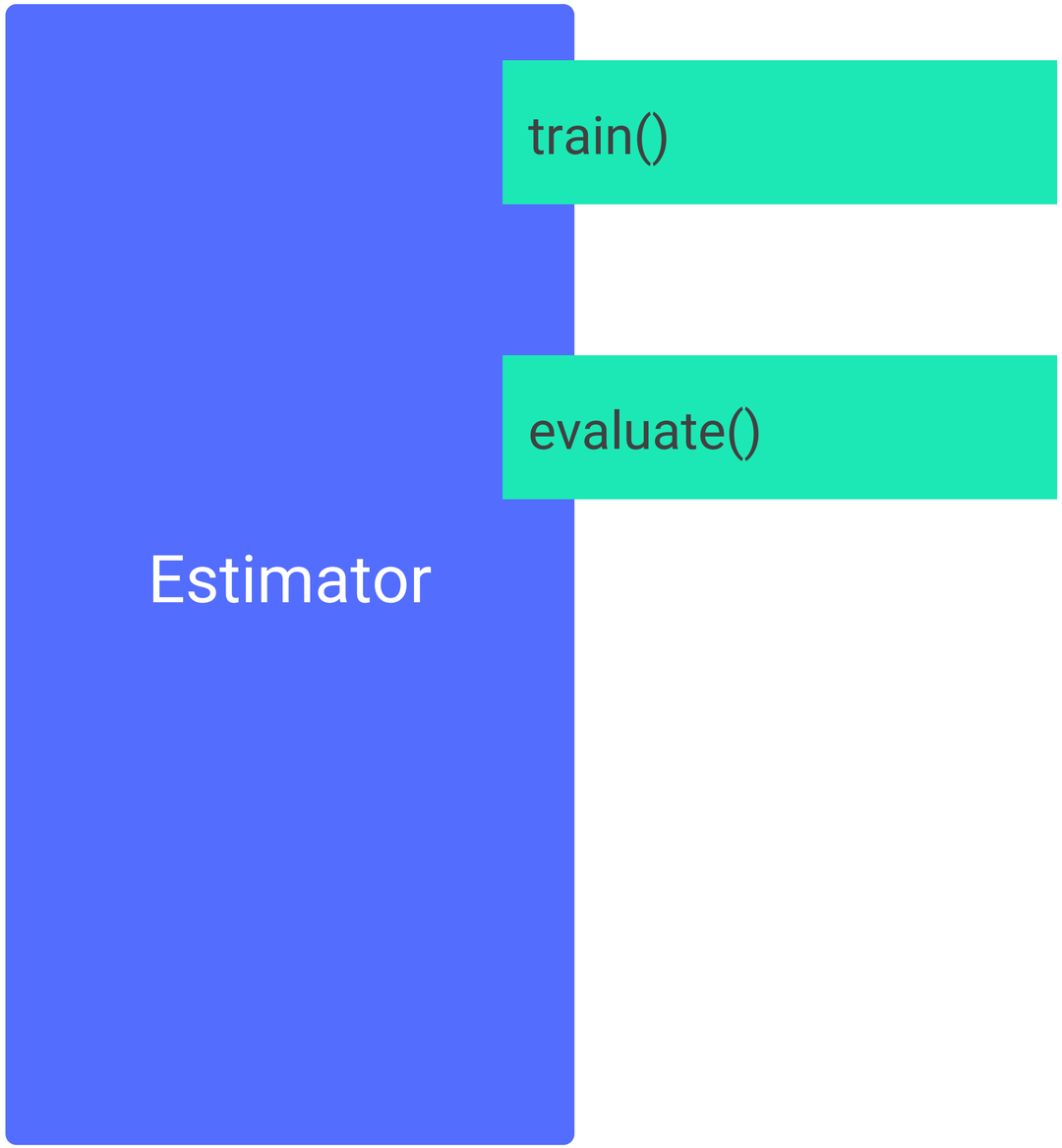
Graph (* = expandable)
- Namespace*
- OpNode
- Unconnected series*
- Connected series*
- Constant
- Summary
- Dataflow edge
- Control dependency edge
- Reference edge

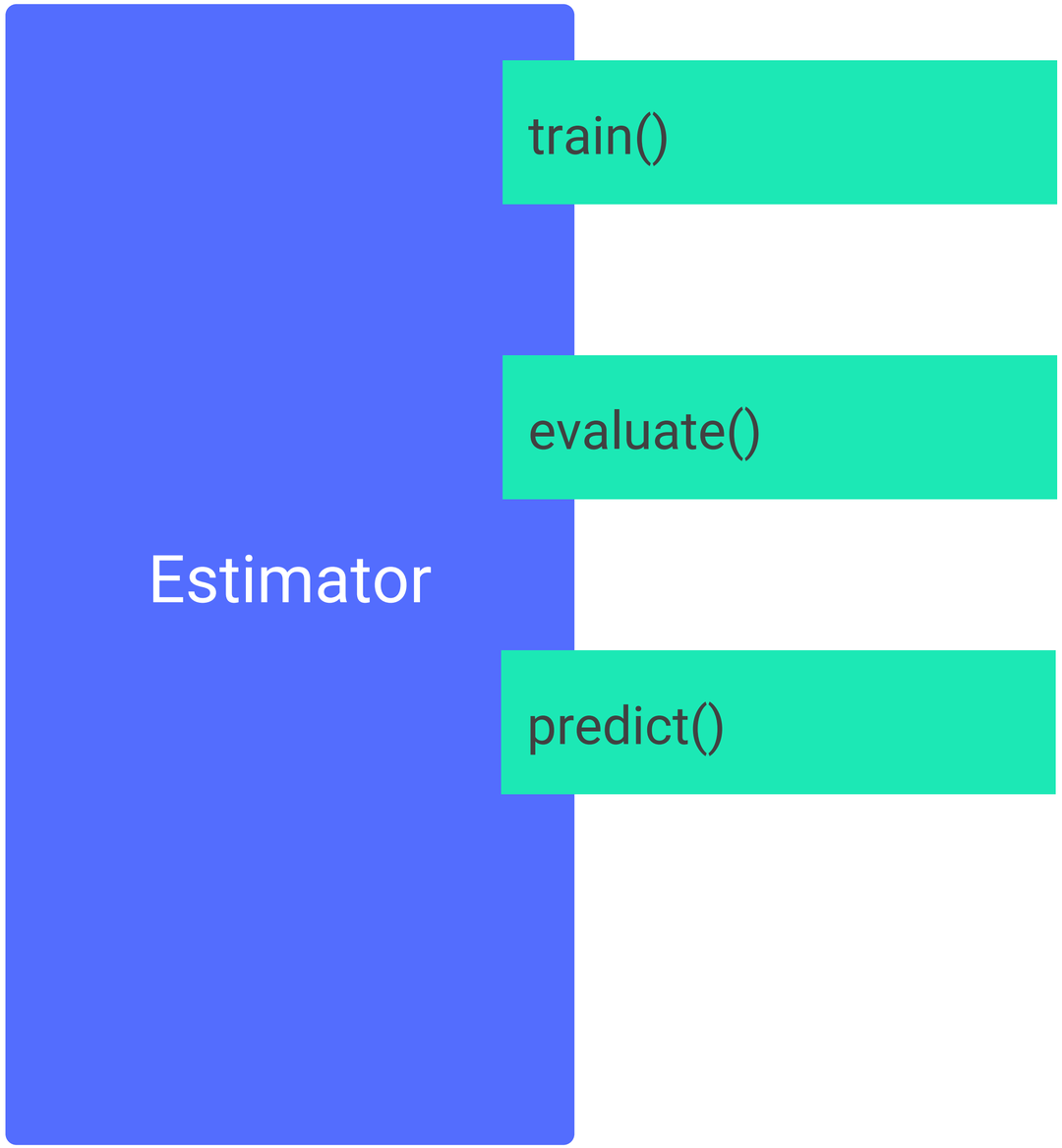


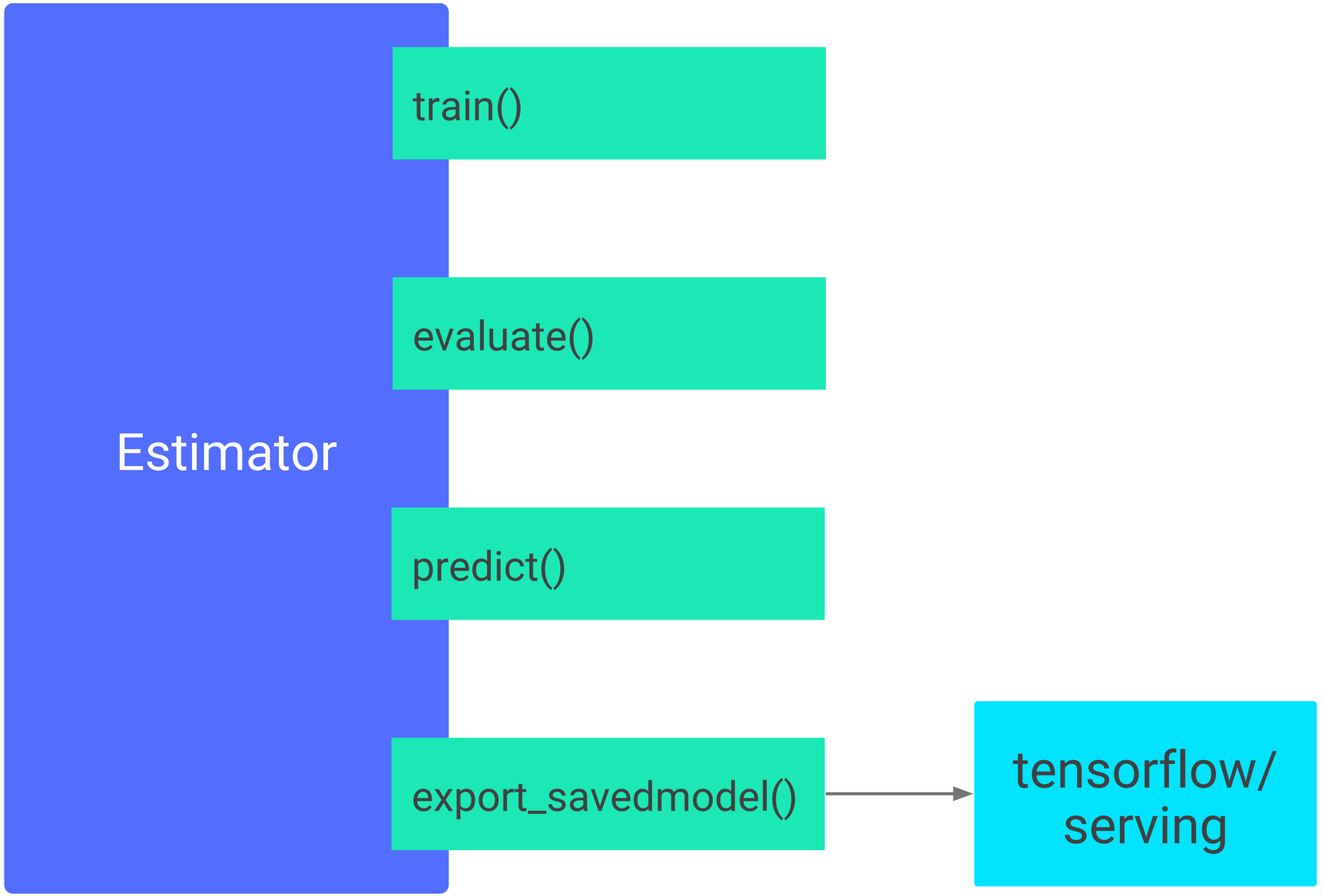


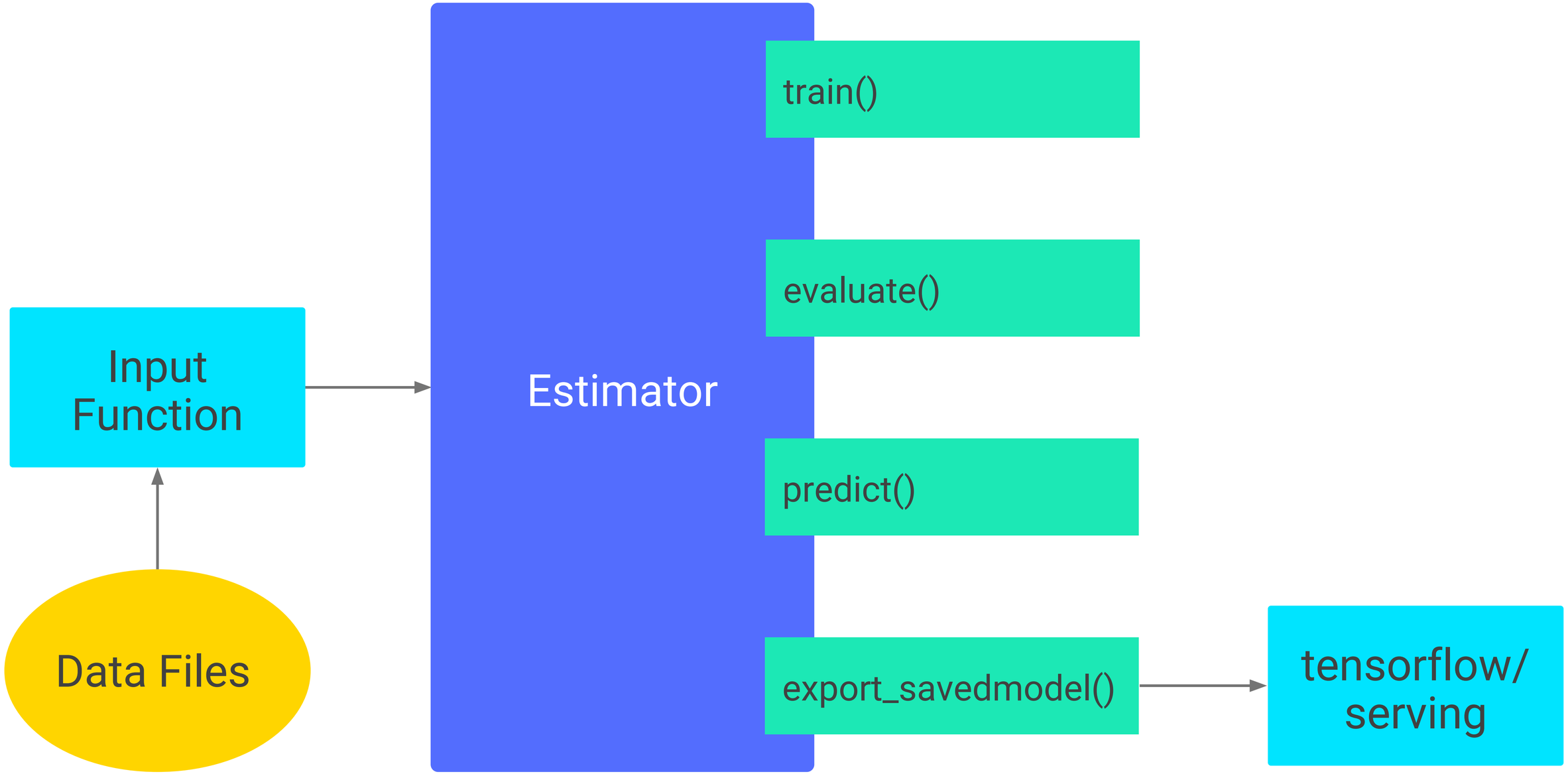
linear
Subgraph: 1364 nodes

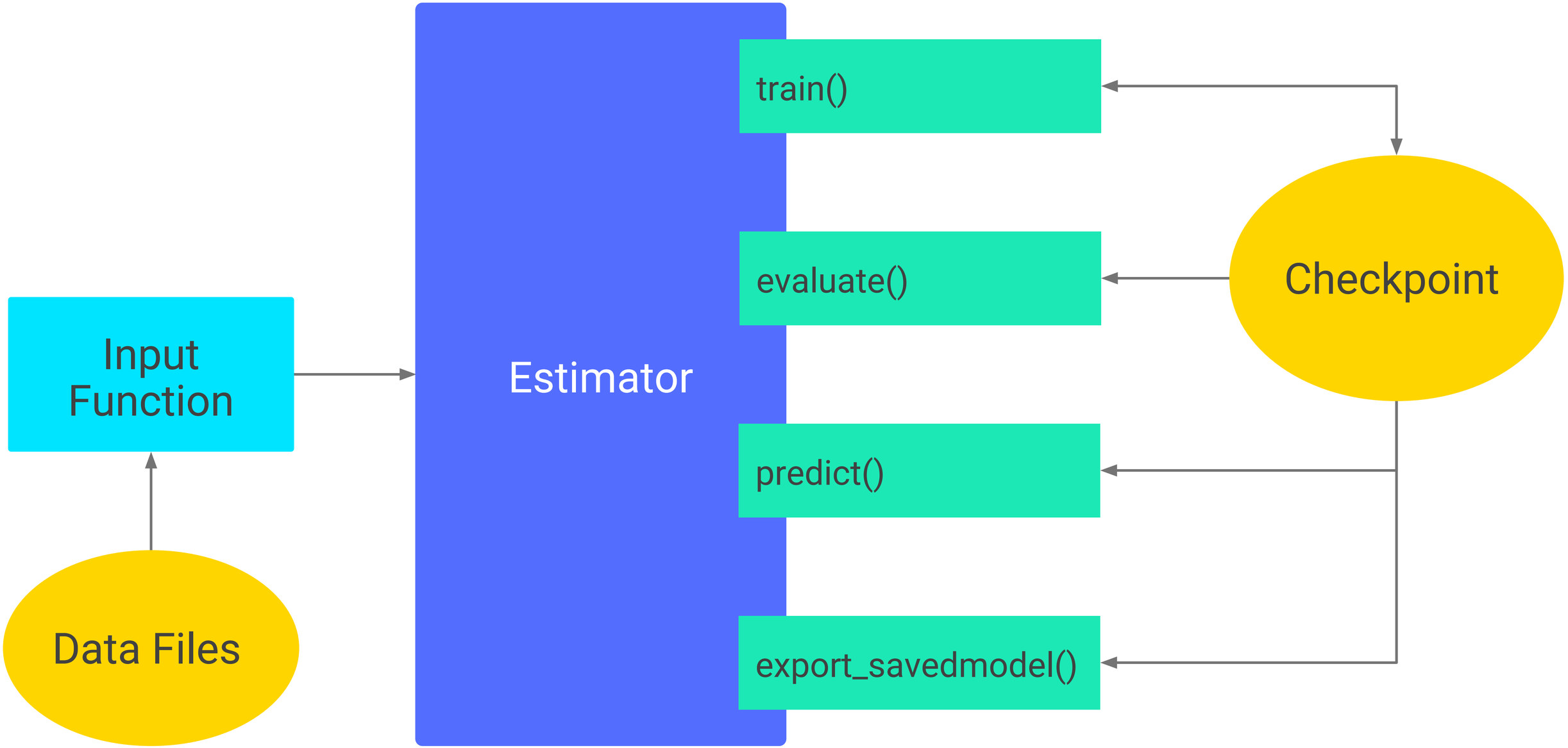

train()
Estimator


train()
evaluate()
Estimator


Estimator
train()
evaluate()
predict()


Estimator
train()
evaluate()
predict()
export_savedmodel()
tensorflow/
serving


Input Function
Data Files
Estimator
train()
evaluate()
predict()
export_savedmodel()
tensorflow/ serving


Input Function
Data Files
Estimator
train()
evaluate()
predict()
export_savedmodel()
Checkpoint

I want
... something else

```
...
regressor = LinearRegressor(feature_columns=[make, horsepower,
                                             cylinders, ...])

train_input_fn = pandas_input_fn(x=input_data, y=input_label,
                                 batch_size=64, shuffle=True,
                                 num_epochs=None)
```

numpy_input_fn
pandas_input_fn
generator_input_fn
...

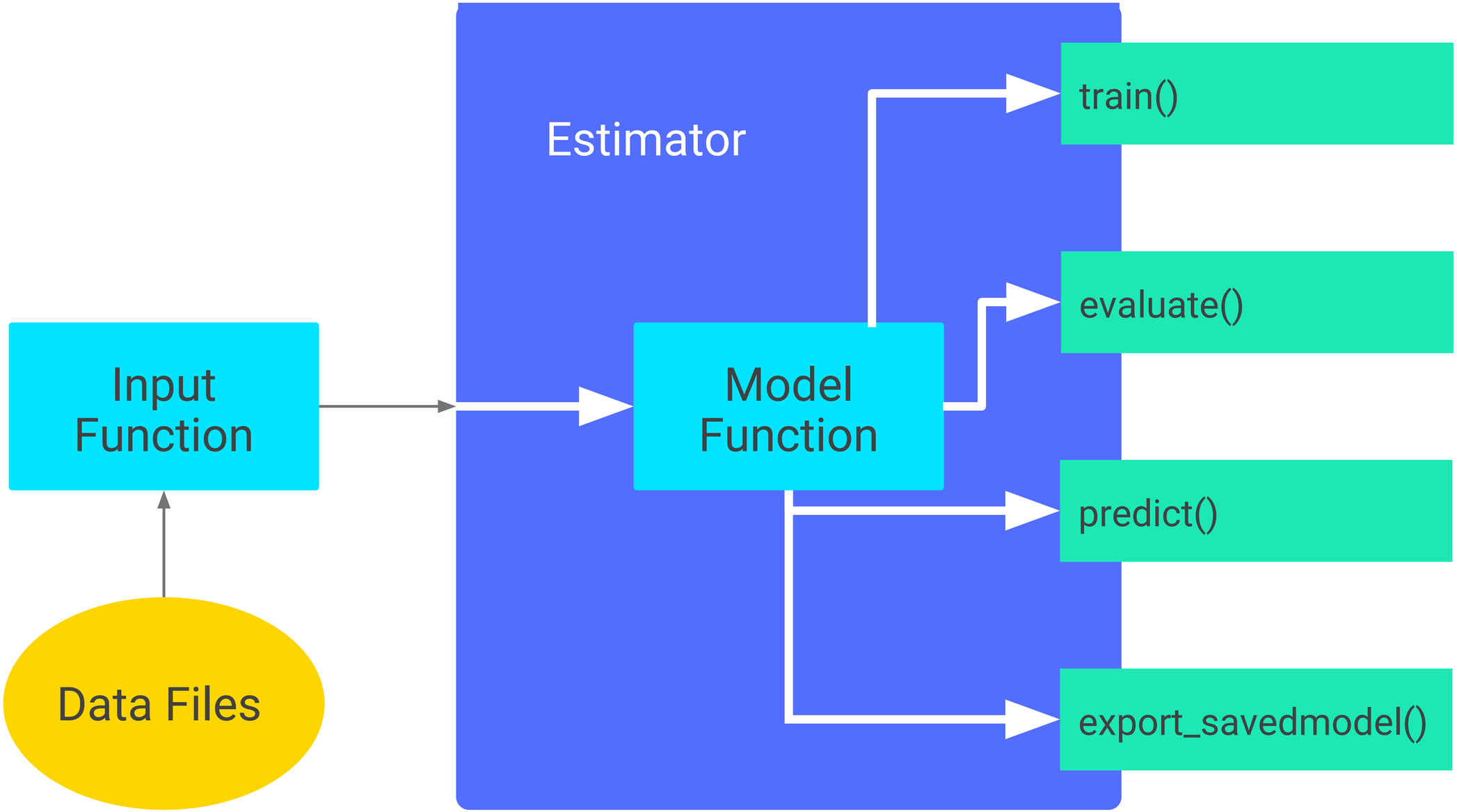
```
cylinders = categorical_column_with_vocabulary_list(
    'num-of-cylinders', ['two', 'three', 'four', ...])
...

regressor = LinearRegressor(feature_columns=[make, horsepower,
                                             cylinders, ...])
```

LinearClassifier
DNNRegressor
DNNClassifier
TensorForest
KMeans
...

```
cylinders = categorical_column_with_vocabulary_list(
    'num-of-cylinders', ['two', 'three', 'four', ...])
...

regressor = DNNRegressor(feature_columns=[embedding_column(make, 10),
                                          horsepower,
                                          indicator_column(cylinders, 3),
                                          ...],
                         hidden_units=[50, 30, 10])
```

```
cylinders = categorical_column_with_vocabulary_list(
    'num-of-cylinders', ['two', 'three', 'four', ...])
...

regressor = DNNRegressor(feature_columns=[embedding_column(make, 10),
                                          horsepower,
                                          indicator_column(cylinders, 3),
                                          ...],
                         hidden_units=[50, 30, 10])
```

```
cylinders = categorical_column_with_vocabulary_list(
    'num-of-cylinders', ['two', 'three', 'four', ...])
...

regressor = DNNRegressor(feature_columns=[embedding_column(make, 10),
                                          horsepower,
                                          indicator_column(cylinders, 3),
                                          ...],
                         hidden_units=[50, 30, 10])
```


Estimator
train()
evaluate()
Input Function
Model Function
predict()
export_savedmodel()
Data Files

Distributed TensorFlow

```
def experiment_fn(run_config, hparams):
  # Make Estimator
  regressor = DNNRegressor(..., config=run_config,
                           hidden_units=hparams['units'])

  # Collect information for training
  return Experiment(estimator=regressor,
                    train_input_fn=pandas_input_fn(...),
                    eval_input_fn=pandas_input_fn(...))

if __name__ == '__main__':
  learn_runner.run(experiment_fn,
                   run_config=RunConfig(model_dir="/tmp/output_dir"))
```

```
def experiment_fn(run_config, hparams):
  # Make Estimator
  regressor = DNNRegressor(..., config=run_config,
                           hidden_units=hparams['units'])

  # Collect information for training
  return Experiment(estimator=regressor,
                    train_input_fn=pandas_input_fn(...),
                    eval_input_fn=pandas_input_fn(...))

if __name__ == '__main__':
  learn_runner.run(experiment_fn,
                   run_config=RunConfig(model_dir="/tmp/output_dir"))
```

```
def experiment_fn(run_config, hparams):
  # Make Estimator
  regressor = DNNRegressor(..., config=run_config,
                           hidden_units=hparams['units'])

  # Collect information for training
  return Experiment(estimator=regressor,
                    train_input_fn=pandas_input_fn(...),
                    eval_input_fn=pandas_input_fn(...))

if __name__ == '__main__':
  learn_runner.run(experiment_fn,
                   run_config=RunConfig(model_dir="/tmp/output_dir"))
```

```
def experiment_fn(run_config, hparams):
  # Make Estimator
  regressor = DNNRegressor(..., config=run_config,
                           hidden_units=hparams['units'])

  # Collect information for training
  return Experiment(estimator=regressor,
                    train_input_fn=pandas_input_fn(...),
                    eval_input_fn=pandas_input_fn(...))

if __name__ == '__main__':
  learn_runner.run(experiment_fn,
                   run_config=RunConfig(model_dir="/tmp/output_dir"))
```

```
cluster_spec = {
    'ps': ['host1:2222', 'host2:2222', ...],
    'worker': ['host3:2222', 'host4:2222', ...],
}

os.environ['TF_CONFIG'] = json.dumps({
    'cluster': cluster_spec,
    'task': {'type': 'worker', 'index': 1}
})
```

Setting up a cluster

`github.com/tensorflow/ecosystem`


A
B
C
D

Complete ML models, quickly

- Complete ML models, quickly
- Distributed training

- Complete ML models, quickly
- Distributed training
- Runs on CPU/GPU/TPU/mobile

Full Code available here

goo.gl/0OgXiL

TensorFlow Tutorials

https://www.tensorflow.org/tutorials

Estimator Tutorial

https://www.tensorflow.org/extend/estimators

From Research to Production with TensorFlow Serving

Tomorrow, 1:30p, Amphitheatre

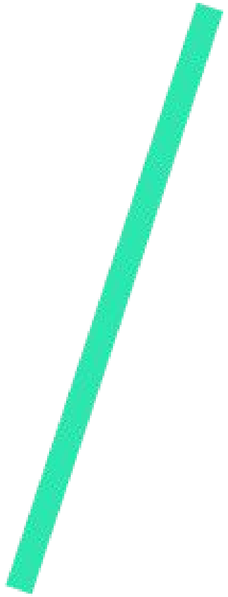
#io17

The Keras API

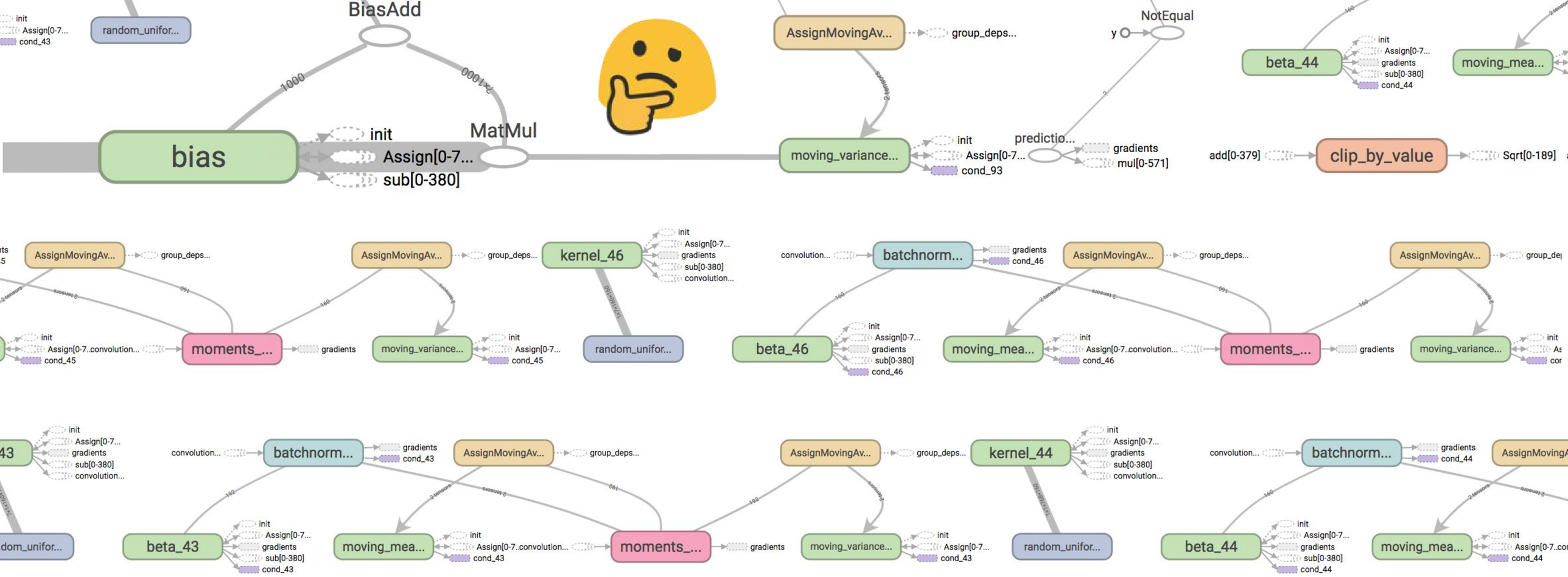
A simple, intuitive interface for TensorFlow

Baohua Liao
liaobaohua@gmail.com

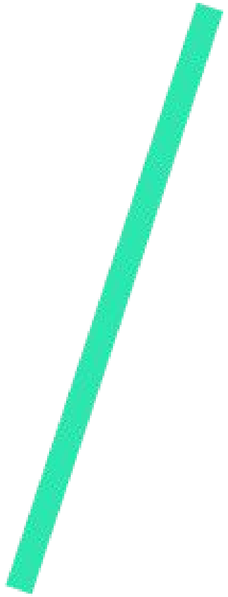
#io17



Keras is a high-level API for building TensorFlow models, to use with Estimators and Experiments.



Building TensorFlow models from scratch



The core ideas behind deep learning are simple, and so should be their implementation.

Intuitive affordances

Expressive:
build anything

Fast experimentation,
continuous learning

Instant, clear feedback

Fun, accessible!

Keras

Deep learning for everyone

A **high-level** API specification

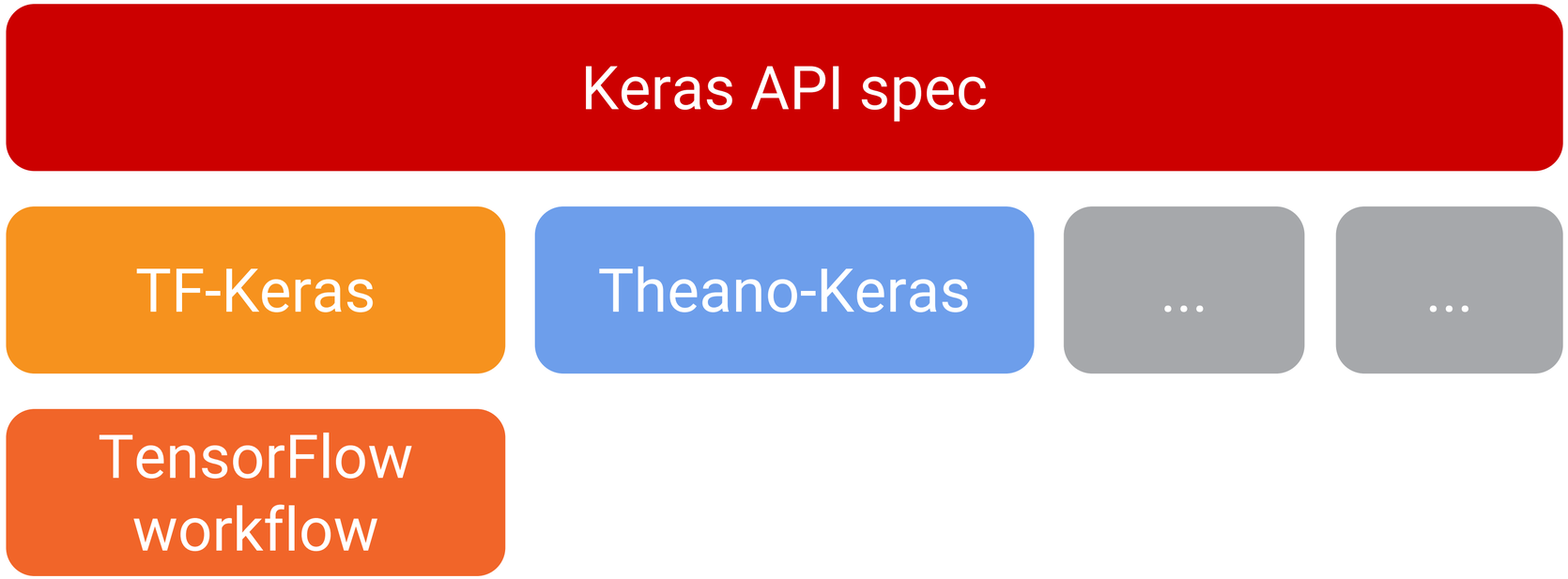
A **high-level** API specification

Built with a focus on **User Experience**

A **high-level** API specification

Built with a focus on **User Experience**

“**Deep learning accessible to everyone**”


Keras API spec
TF-Keras
Theano-Keras
...
...
TensorFlow workflow

The Keras API as **tf.keras**

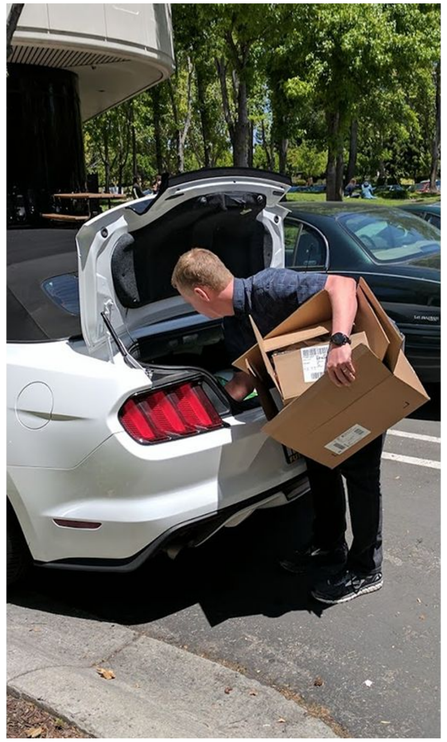
For TensorFlow users: a simplified workflow

For Keras users: access to more powerful features

Example workflow

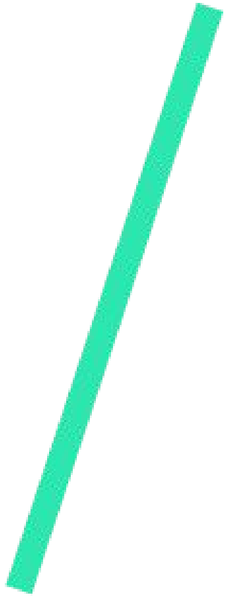
A toy video-QA problem

Toy video-QA problem

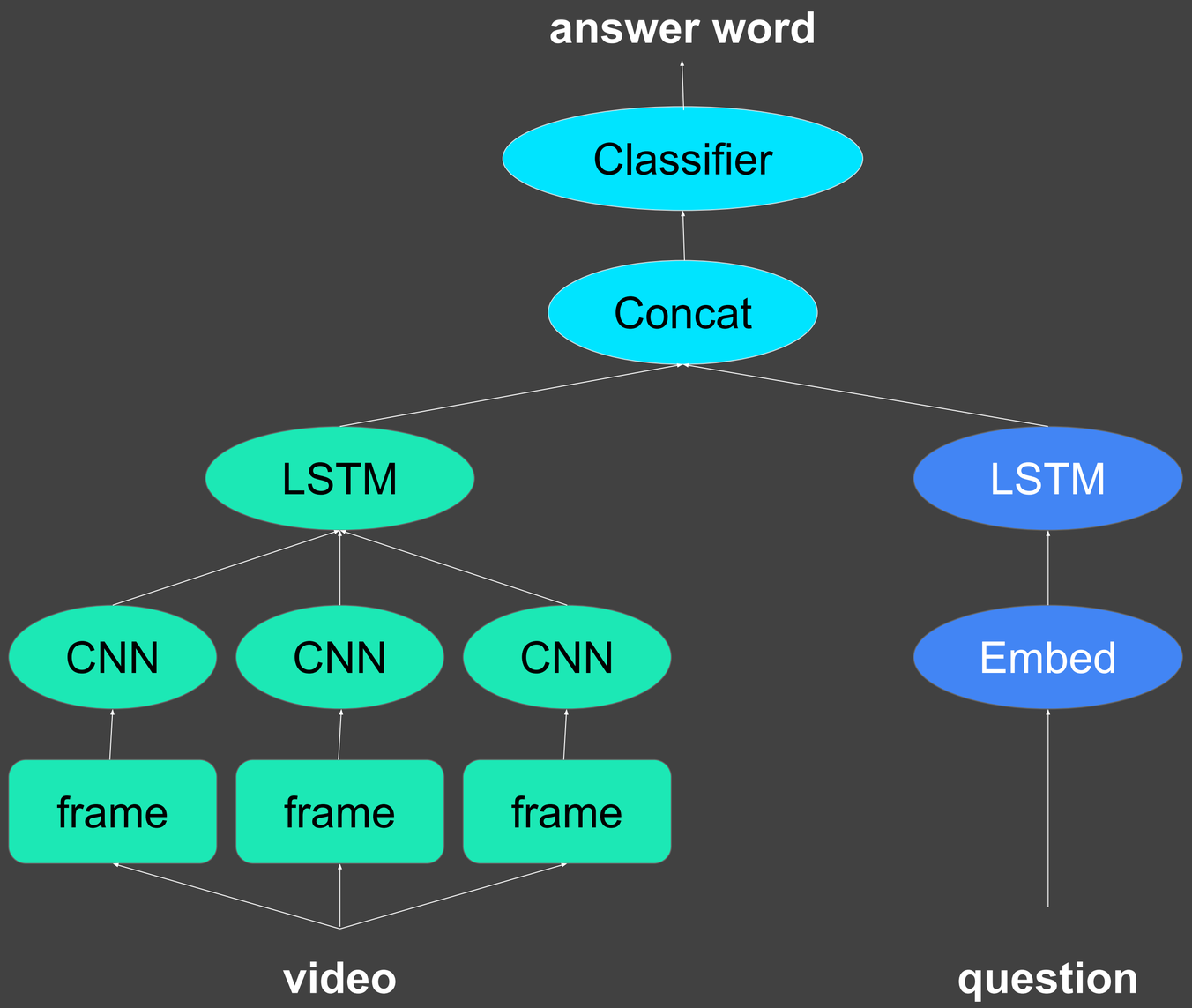




> ***“What is the man doing?”***
> packing

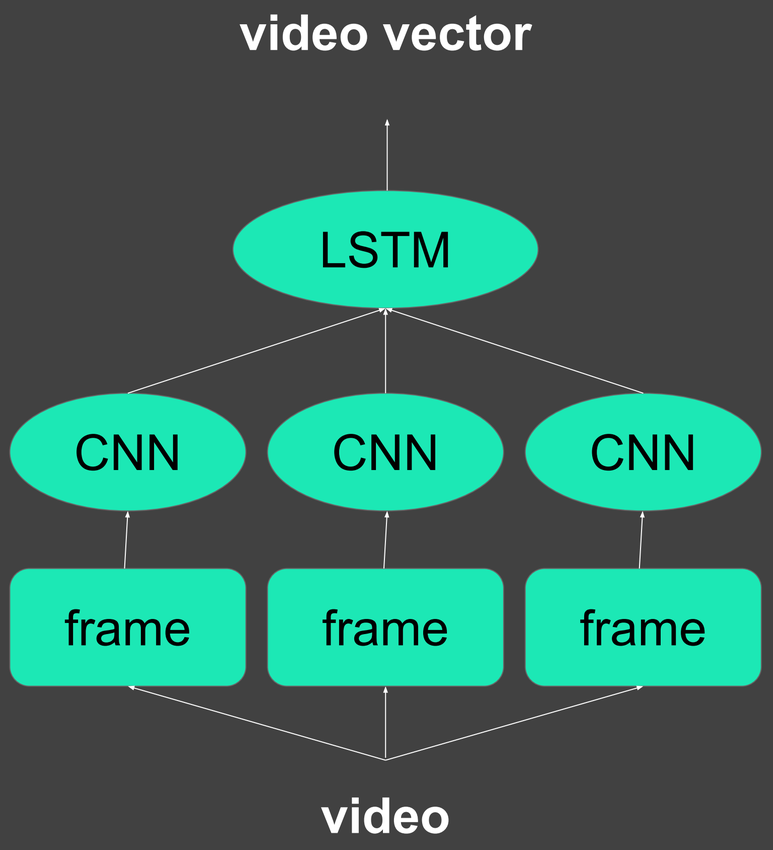
> ***“What color is his shirt?”***
> blue

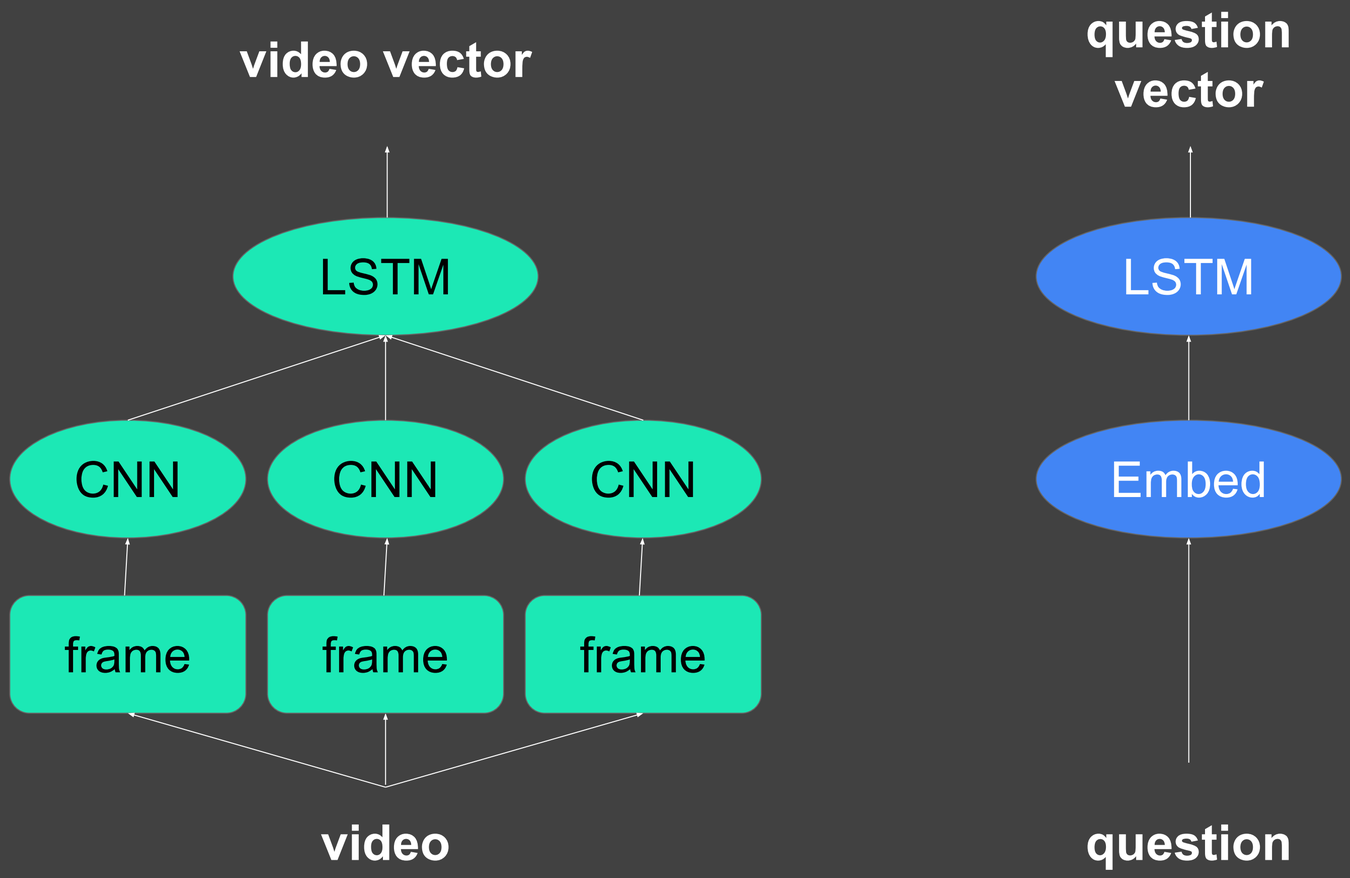


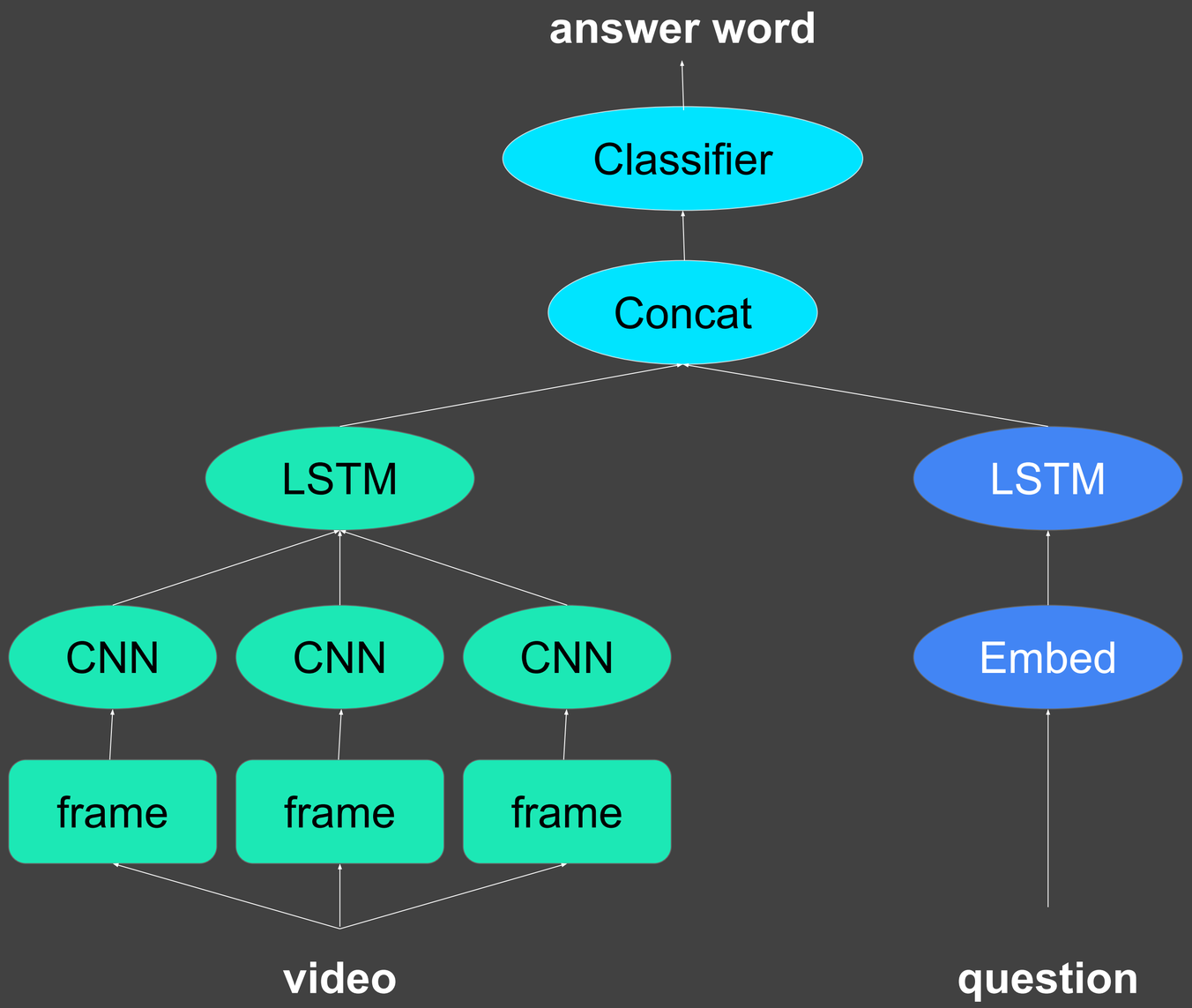
A previously very hard problem, made accessible to anyone with basic Python scripting abilities

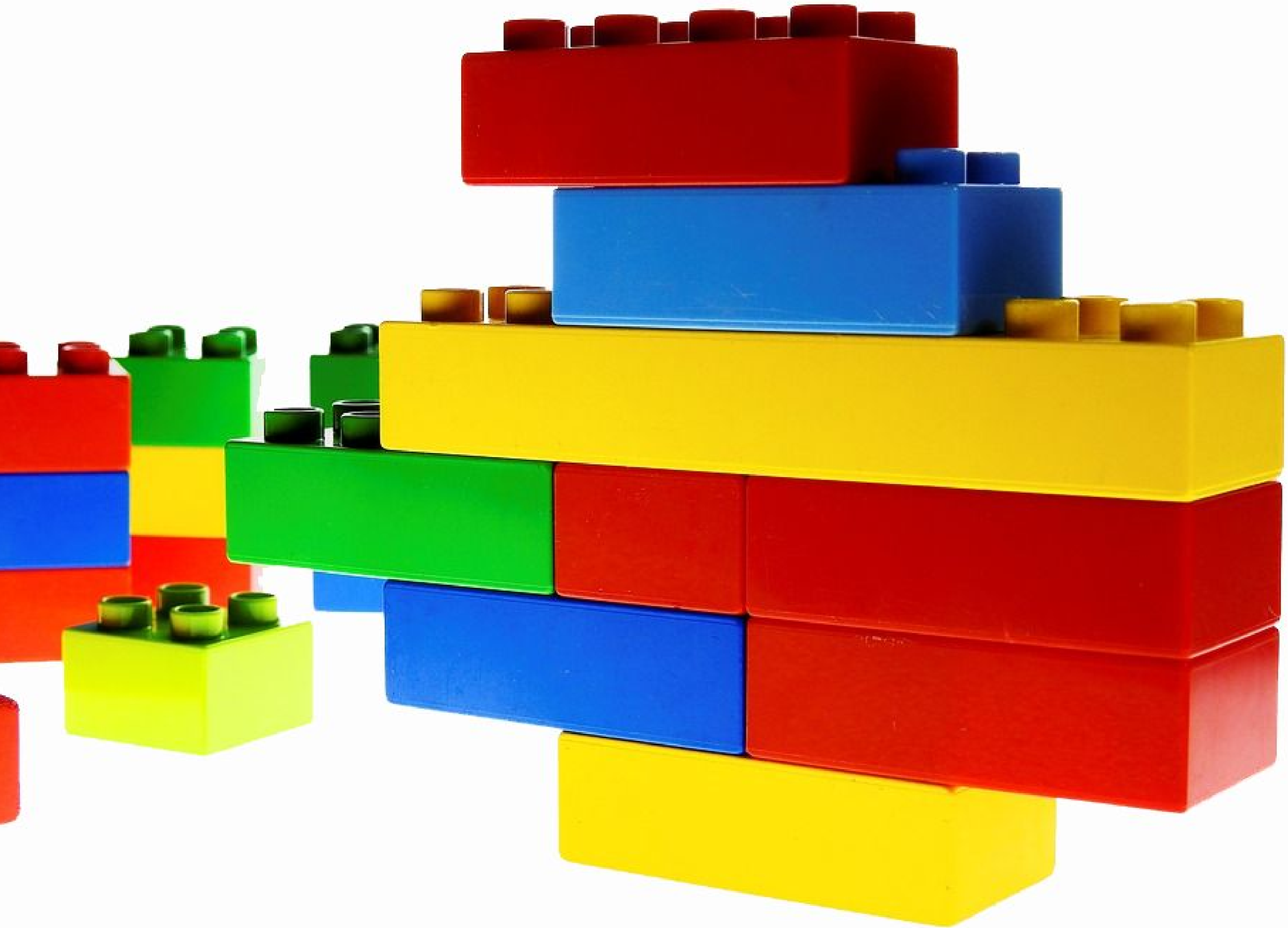

answer word
Classifier
Concat
LSTM
LSTM
CNN
CNN
CNN
Embed
frame
frame
frame
video
question

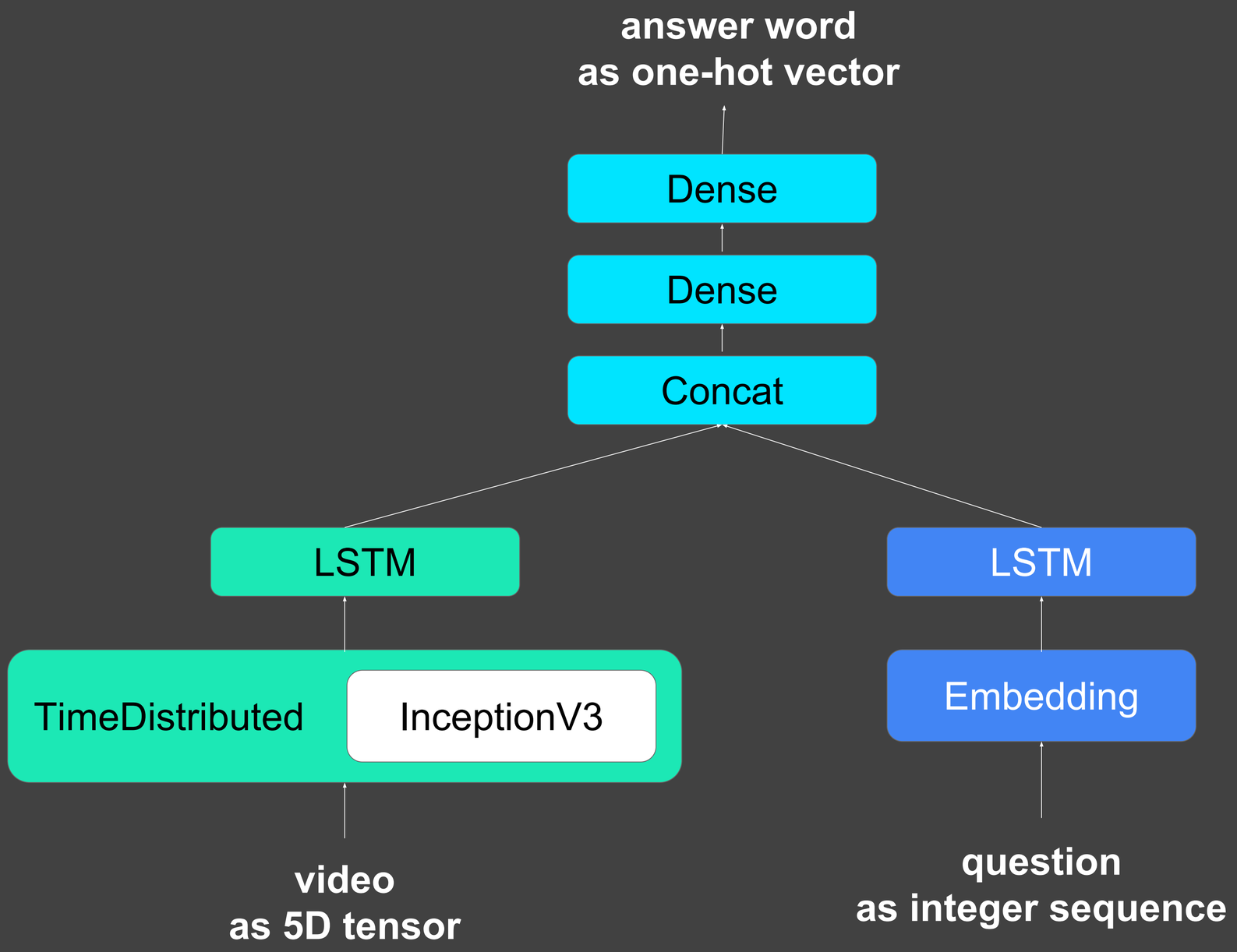
from frames to a vector




video vector
LSTM
CNN
CNN
CNN
frame
frame
frame
video
question vector
LSTM
Embed
question


answer word
Classifier
Concat
LSTM
LSTM
CNN
CNN
CNN
Embed
frame
frame
frame
video
question




answer word
as one-hot vector
Dense
Dense
Concat
LSTM
LSTM
TimeDistributed
InceptionV3
Embedding
video
as 5D tensor
question
as integer sequence

Turning frames into a vector, with pre-trained representations

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import InceptionV3

video = keras.Input(shape=(None, 150, 150, 3), name='video')
cnn = InceptionV3(weights='imagenet',
                  include_top=False,
                  pooling='avg')
cnn.trainable = False
frame_features = layers.TimeDistributed(cnn)(video)
video_vector = layers.LSTM(256)(frame_features)
```

Turning frames into a vector, with pre-trained representations

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import InceptionV3

video = keras.Input(shape=(None, 150, 150, 3), name='video')
cnn = InceptionV3(weights='imagenet',
                  include_top=False,
                  pooling='avg')
cnn.trainable = False
frame_features = layers.TimeDistributed(cnn)(video)
video_vector = layers.LSTM(256)(frame_features)
```

Turning frames into a vector, with pre-trained representations

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import InceptionV3

video = keras.Input(shape=(None, 150, 150, 3), name='video')
cnn = InceptionV3(weights='imagenet',
                  include_top=False,
                  pooling='avg')
cnn.trainable = False
frame_features = layers.TimeDistributed(cnn)(video)
video_vector = layers.LSTM(256)(frame_features)
```

Turning frames into a vector, with pre-trained representations

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import InceptionV3

video = keras.Input(shape=(None, 150, 150, 3), name='video')
cnn = InceptionV3(weights='imagenet',
                  include_top=False,
                  pooling='avg')
cnn.trainable = False
frame_features = layers.TimeDistributed(cnn)(video)
video_vector = layers.LSTM(256)(frame_features)
```

Turning frames into a vector, with pre-trained representations

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.applications import InceptionV3

video = keras.Input(shape=(None, 150, 150, 3), name='video')
cnn = InceptionV3(weights='imagenet',
                  include_top=False,
                  pooling='avg')
cnn.trainable = False
frame_features = layers.TimeDistributed(cnn)(video)
video_vector = layers.LSTM(256)(frame_features)
```

Turning a sequence of words into a vector

```
question = keras.Input(shape=(None,), dtype='int32', name='question')
embedded_words = layers.Embedding(input_voc_size, 256)(question)
question_vector = layers.LSTM(128)(embedded_words)
```

Predicting an answer word

```
x = layers.concatenate([video_vector, question_vector])
x = layers.Dense(128, activation=tf.nn.relu)(x)
predictions = layers.Dense(output_voc_size, name='predictions')(x)
```

Setting up the training configuration

```
model = keras.models.Model([video, question], predictions)
model.compile(optimizer=tf.AdamOptimizer(),
              loss=tf.softmax_crossentropy_with_logits)
```

Leveraging Experiment for distributed training

```
def experiment_fn(config, params):
  model = ...
  estimator = model.get_estimator(config=...)
  return Experiment(estimator,
                    train_input_fn=pandas_input_fn(...),
                    eval_input_fn=pandas_input_fn(...))

if __name__ == '__main__':
  tf.contrib.train.run_experiment(experiment_fn)
```

Takeaways

Keras: a high-level API with focus on UX

Now part of TensorFlow as **tf.keras**

To use with **Estimators** and **Experiments** for distributed training

A big step towards making deep learning accessible to everyone

Thank you!

Baohua Liao
liaobaohua@gmail.com

#io17