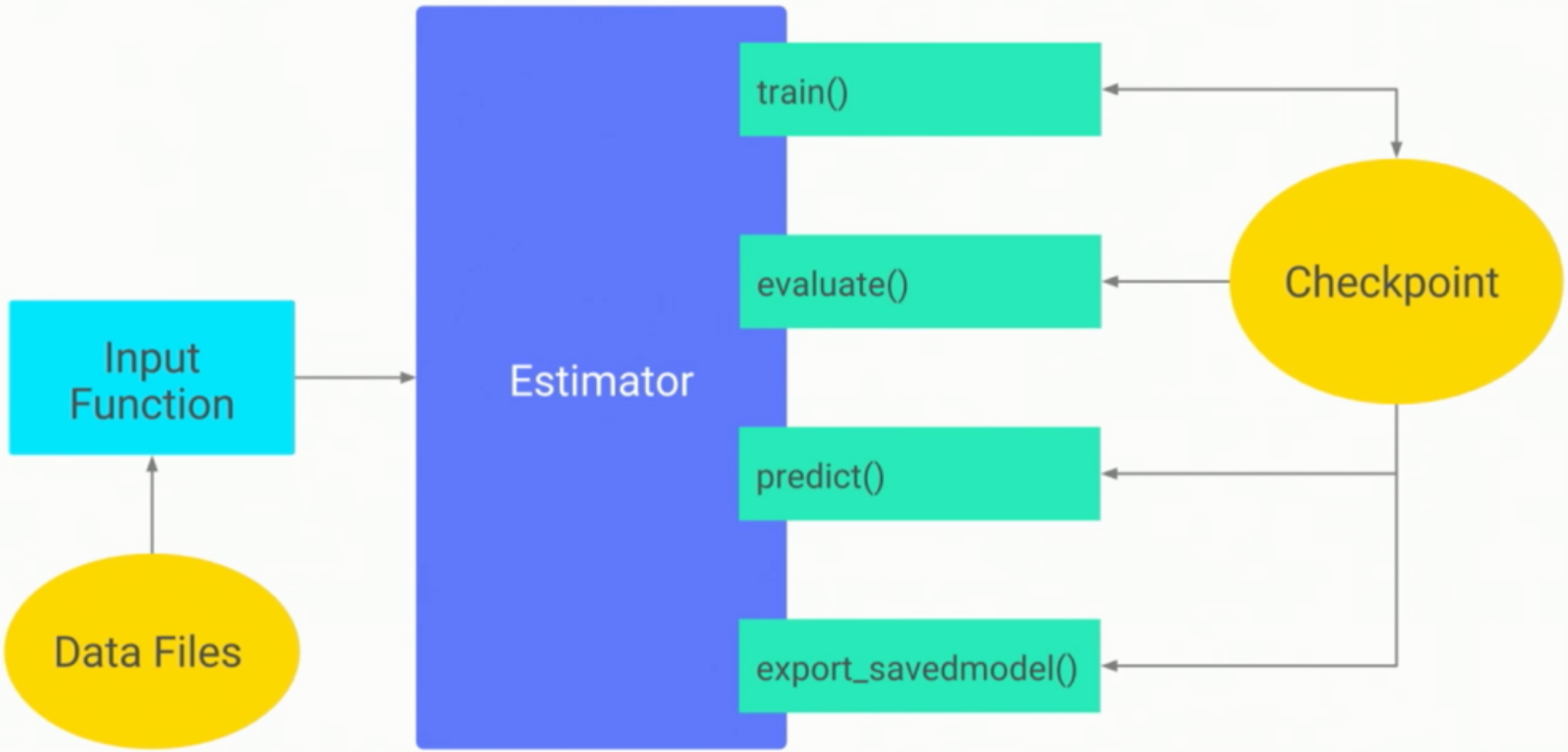
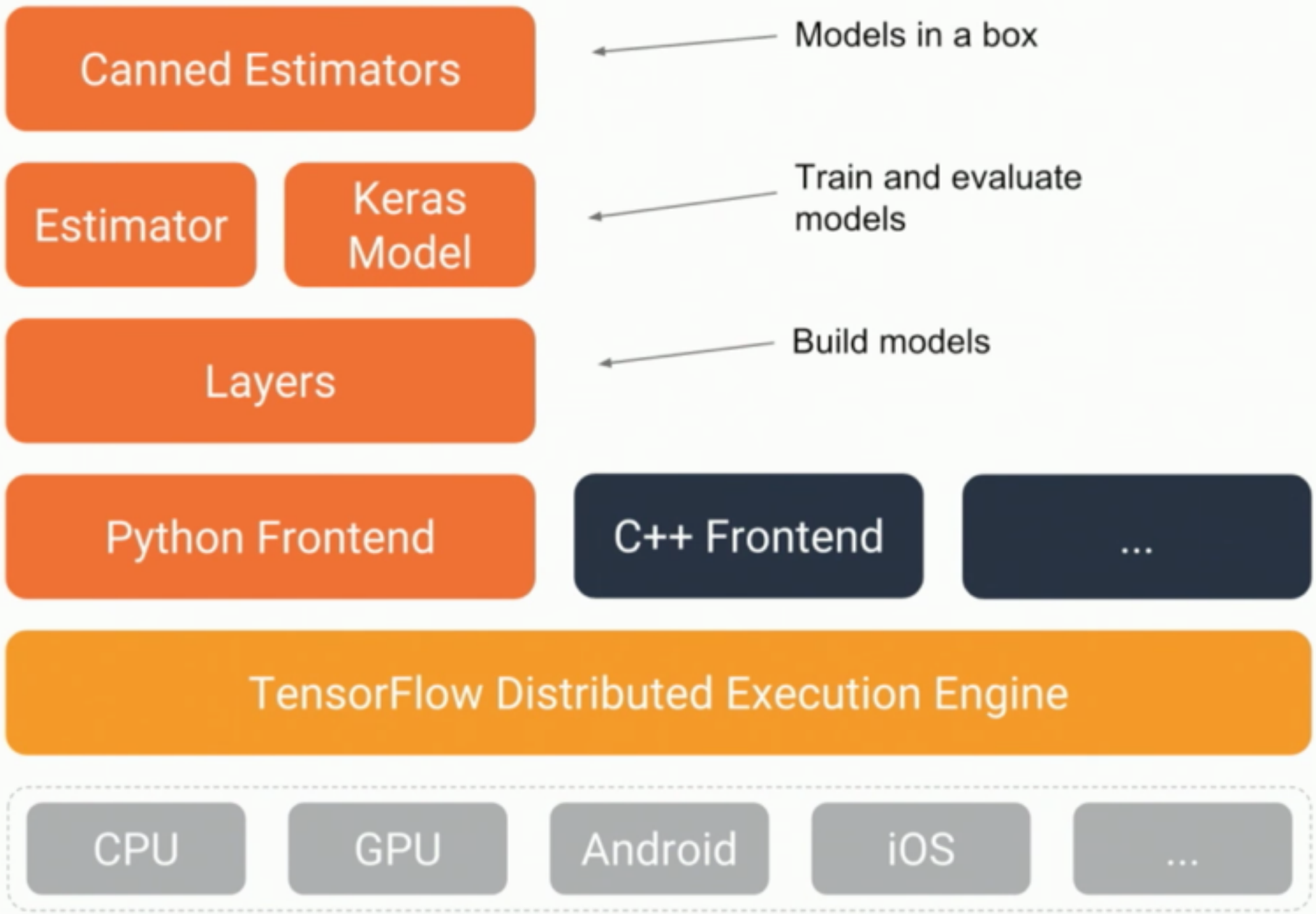
Effective TensorFlow 非专家入门指南

廖祜秋


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


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
...

```
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, ...])
```

```
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)
```

TensorBoard SCALARS IMAGES AUDIO GRAPHS DISTRIBUTIONS HISTOGRAMS EMBEDDINGS TEXT

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

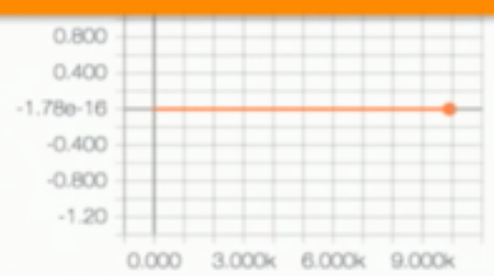
Write a regex to filter runs

.

eval

TOGGLE ALL RUNS

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



dnn/regression_head/loss

enqueue_input 1

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

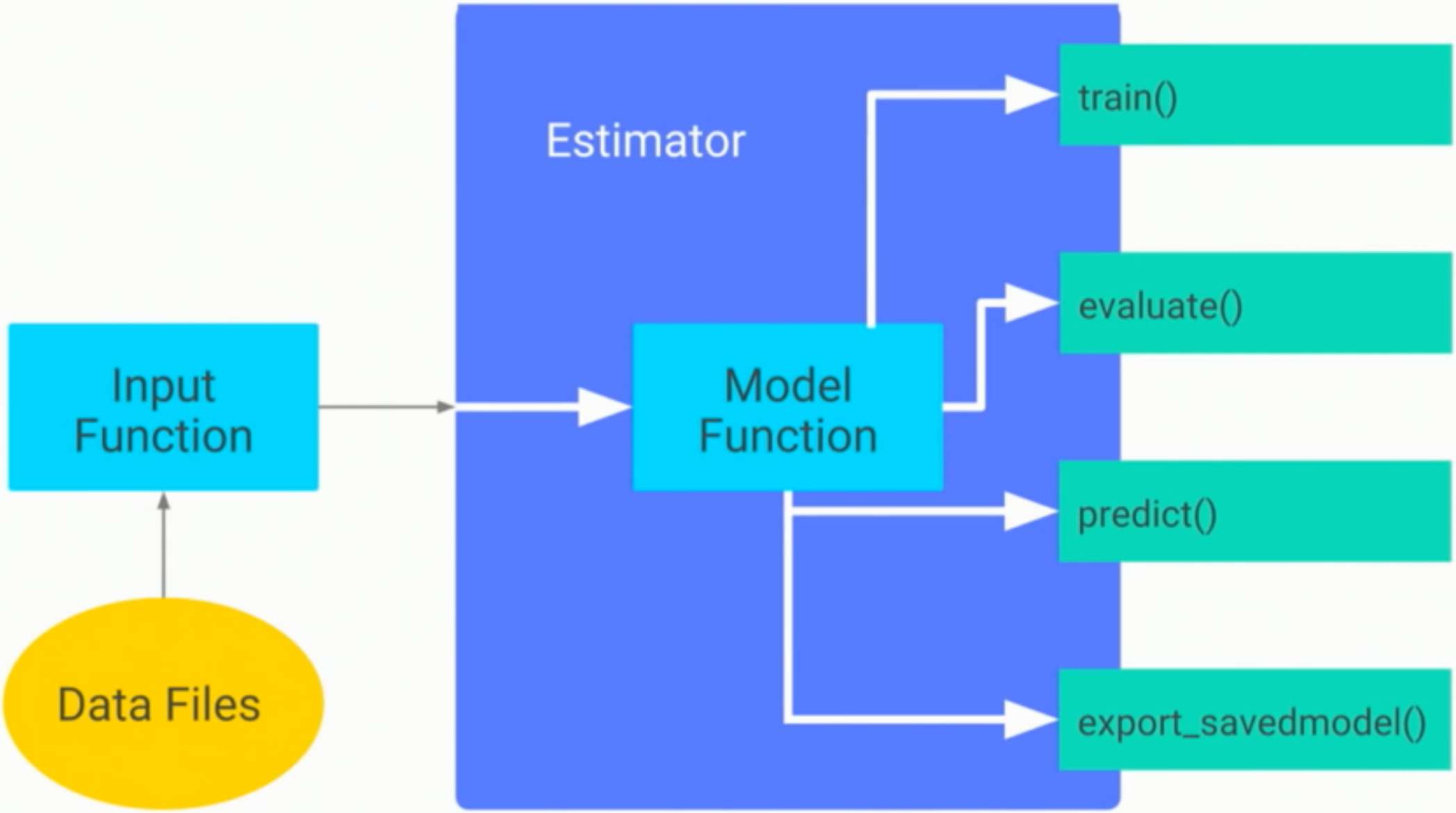
```
numpy_input_fn
pandas_input_fn
generator_input_fn
...
```

```
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])
```


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

```
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"))
```

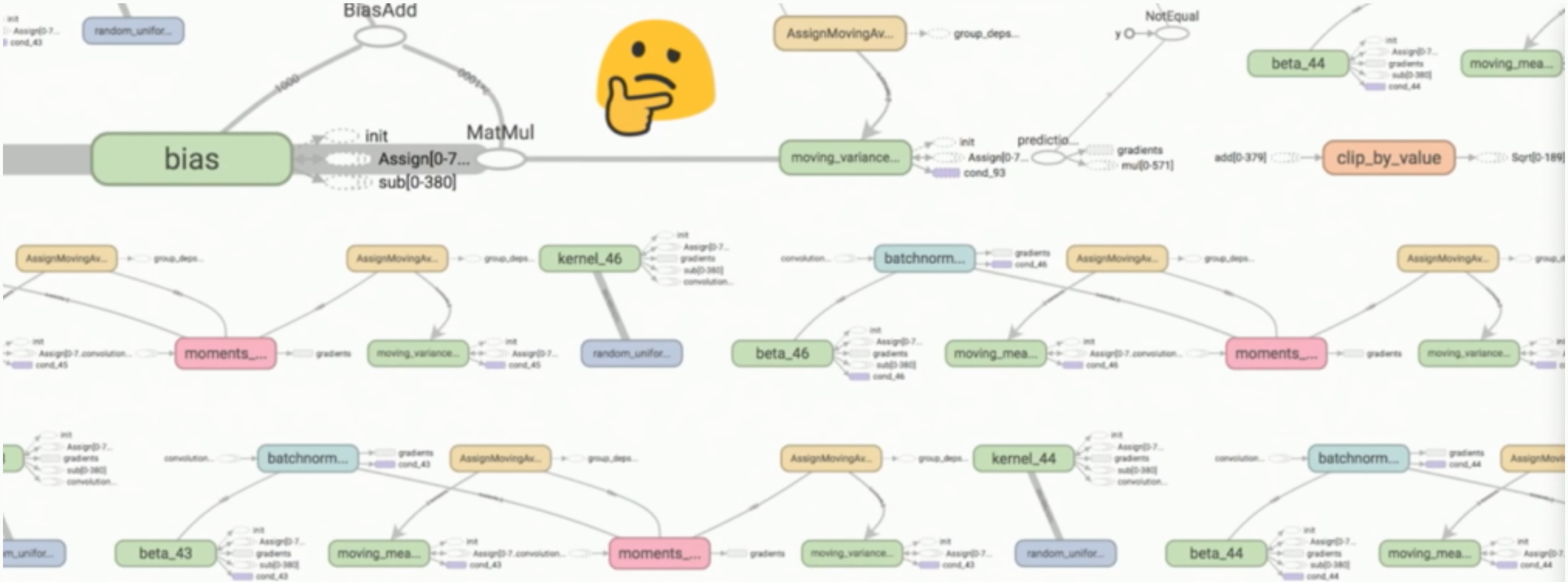
```
cluster_spec = {
    'ps': ['host1:2222', 'host2:2222', ...],
    'worker': ['host3:2222', 'host4:2222', ...],
}
```

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

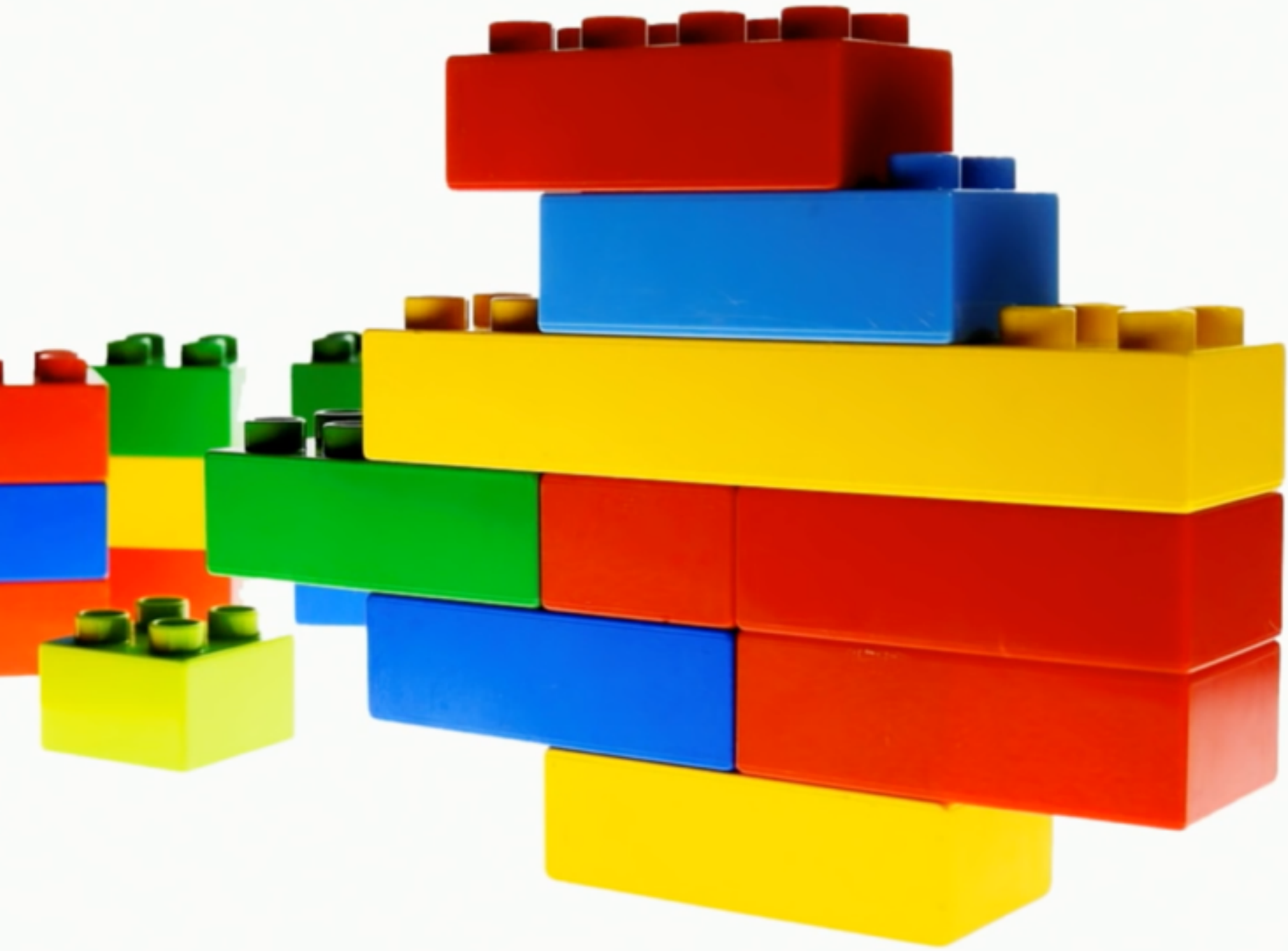
https://github.com/tensorflow/ecosystem

https://gist.github.com/martinwicke/6838c23abdc53e6bcda36ed9f40cff39

- Complete ML models quickly
- Distributed training



Building TensorFlow models from scratch



- Keras: Deep learning for everyone
- A high-level API specification

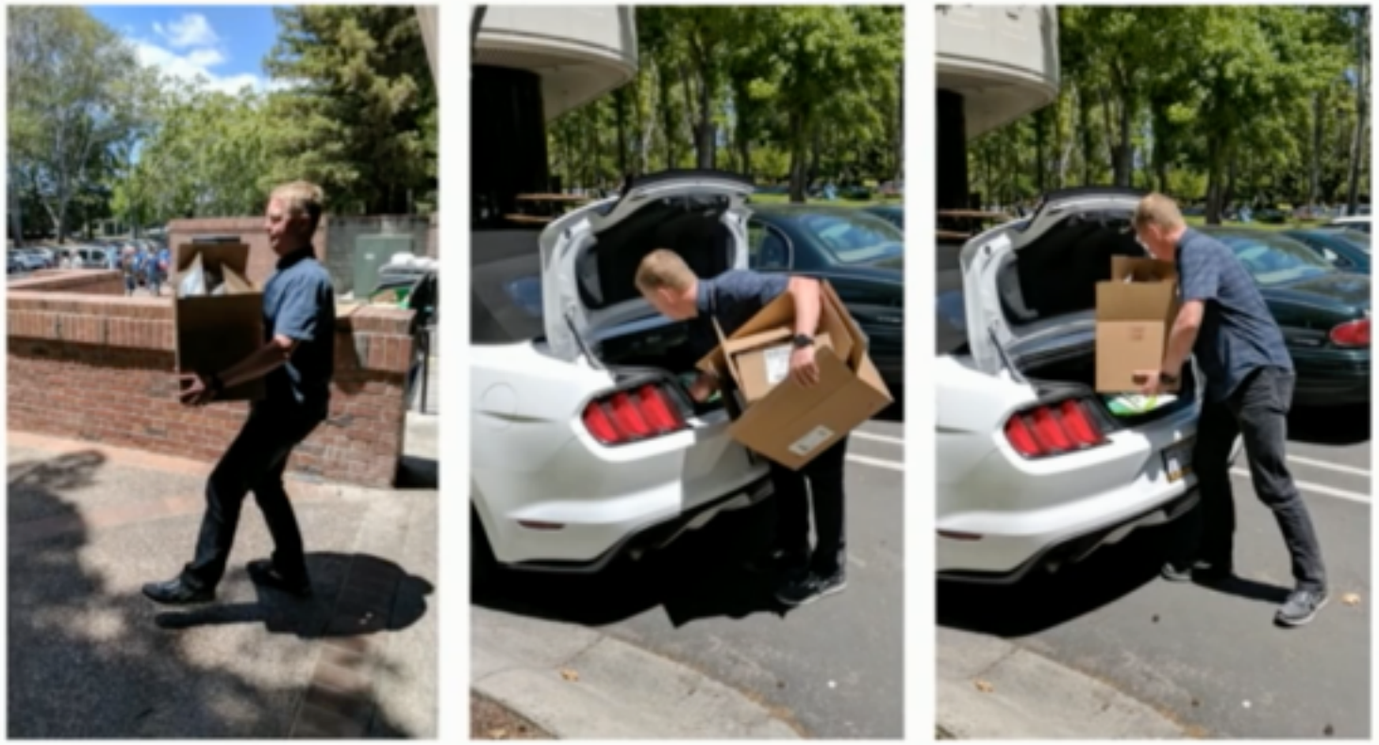
- Focus on User Experience

- Keras API as tf.keras

- For TensorFlow Users: a simplified workflow

- For Keras users: access to more powerful features: distributed training, hyper parameter optimization, training on Cloud ML

Toy video-QA problem

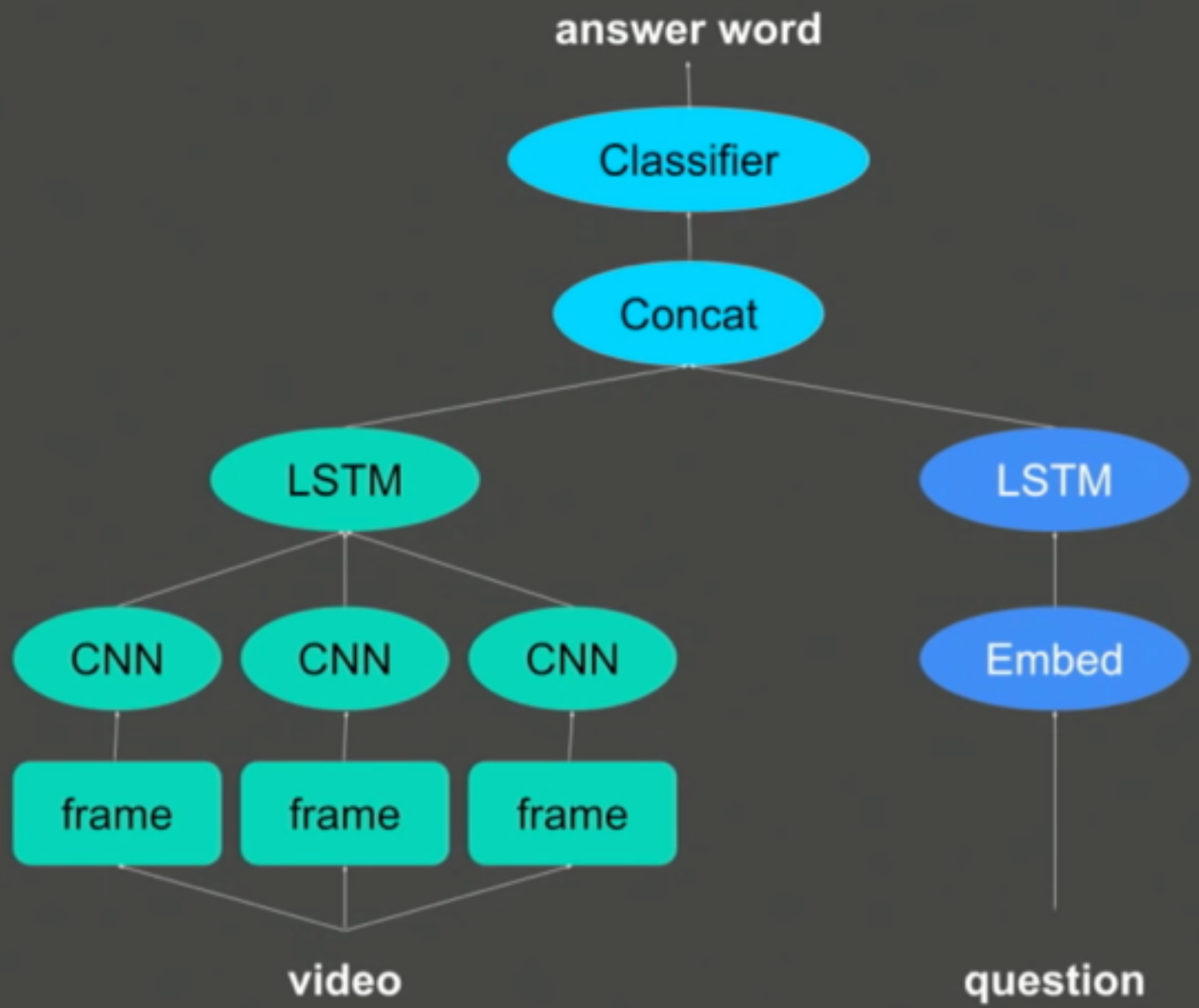


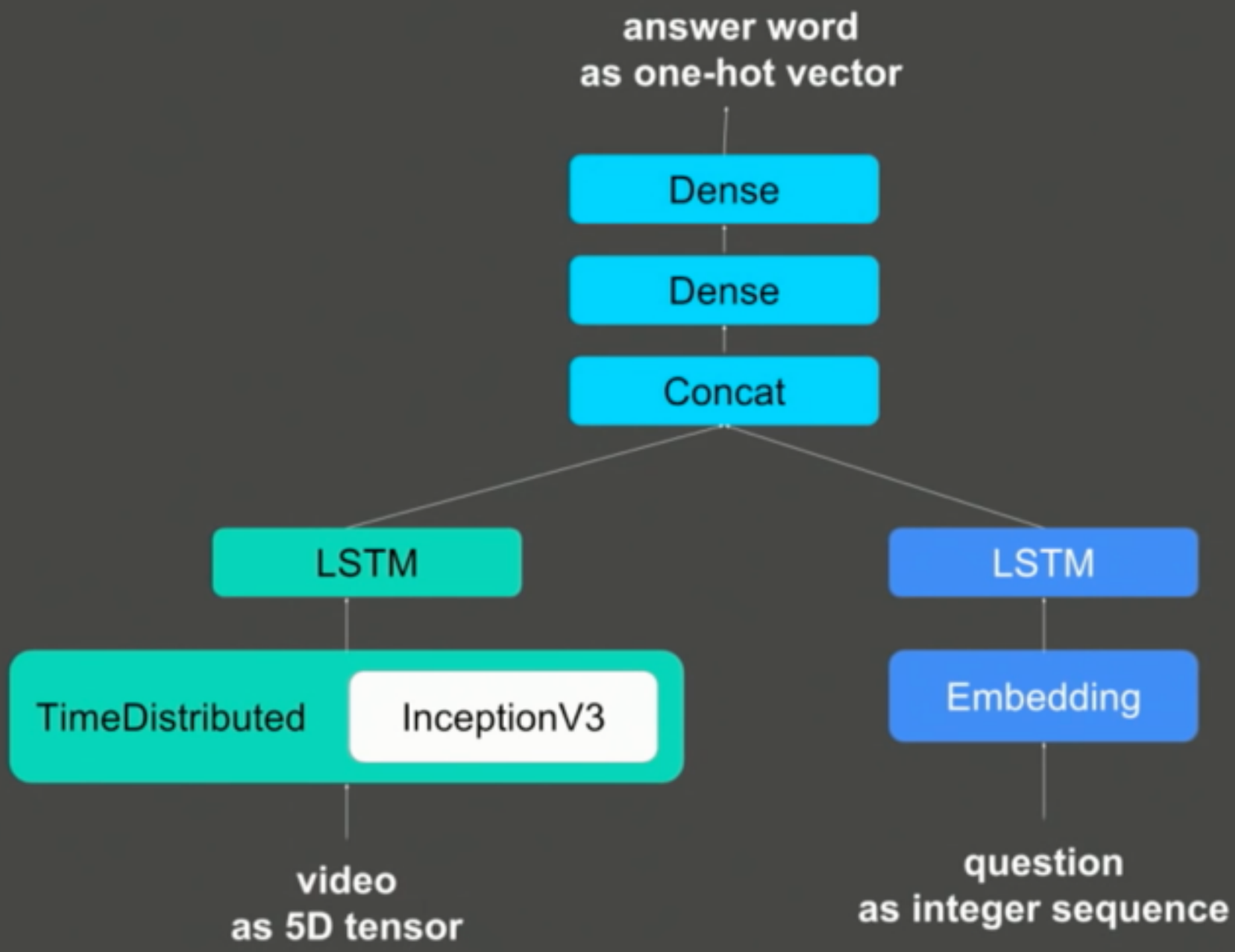
> ***"What is the man doing?"***
> packing

> ***"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


answer word
as one-hot vector
Dense
Dense
Concat
LSTM
TimeDistributed
InceptionV3
video
as 5D tensor
LSTM
Embedding
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 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)
```

Best practice is the default config.

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)
```

- Eras: a high-level API with focus on UX.

- Now part of TensorFlow

- To use with Estimators and Experiments for distributed training

A big step towards

making deep learning accessible to everyone.