About Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result as fast as possible is key to doing good research.*

Keras is:

- **Simple** -- but not simplistic. Keras reduces developer *cognitive load* to free you to focus on the parts of the problem that really matter.
- **Flexible** -- Keras adopts the principle of *progressive disclosure of complexity*: simple workflows should be quick and easy, while arbitrarily advanced workflows should be *possible* via a clear path that builds upon what you've already learned.
- **Powerful** -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

Keras & TensorFlow 2

TensorFlow 2 is an end-to-end, open-source machine learning platform. You can think of it as an infrastructure layer for differentiable programming. It combines four key abilities:

- Efficiently executing low-level tensor operations on CPU, GPU, or TPU.
- Computing the gradient of arbitrary differentiable expressions.
- Scaling computation to many devices, such as clusters of hundreds of GPUs.
- Exporting programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices.

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

First contact with Keras

The core data structures of Keras are **layers** and **models**. The simplest type of model is the `Sequential` model, a linear stack of layers. For more complex architectures, you should use the Keras functional API, which allows to build arbitrary graphs of layers, or write models entirely from scratch via subclasssing.

Here is the `Sequential` model:

```
from tensorflow.keras.models import Sequential

model = Sequential()
```

Stacking layers is as easy as `.add()`:

```
from tensorflow.keras.layers import Dense

model.add(Dense(units=64, activation='relu'))
model.add(Dense(units=10, activation='softmax'))
```

Once your model looks good, configure its learning process with `.compile()`:

```
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

If you need to, you can further configure your optimizer. The Keras philosophy is to keep simple things simple, while allowing the user to be fully in control when they need to (the ultimate control being the easy extensibility of the source code via subclassing).

```
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.SGD(learning_rate=0.01, momentum=0.9,
nesterov=True))
```

You can now iterate on your training data in batches:

```
# x_train and y_train are Numpy arrays
model.fit(x_train, y_train, epochs=5, batch_size=32)
```

Evaluate your test loss and metrics in one line:

```
loss_and_metrics = model.evaluate(x_test, y_test, batch_size=128)
```

Or generate predictions on new data:

```
classes = model.predict(x_test, batch_size=128)
```

What you just saw is the most elementary way to use Keras.

However, Keras is also a highly-flexible framework suitable to iterate on state-of-the-art research ideas. Keras follows the principle of **progressive disclosure of complexity**: it makes it easy to get started, yet it makes it possible to handle arbitrarily advanced use cases, only requiring incremental learning at each step.

In much the same way that you were able to train & evaluate a simple neural network above in a few lines, you can use Keras to quickly develop new training procedures or exotic model architectures. Here's a low-level training loop example, combining Keras functionality with the TensorFlow `GradientTape`:

```
import tensorflow as tf

# Prepare an optimizer.
optimizer = tf.keras.optimizers.Adam()
# Prepare a loss function.
loss_fn = tf.keras.losses.kl_divergence

# Iterate over the batches of a dataset.
for inputs, targets in dataset:
    # Open a GradientTape.
    with tf.GradientTape() as tape:
        # Forward pass.
        predictions = model(inputs)
        # Compute the loss value for this batch.
        loss_value = loss_fn(targets, predictions)

    # Get gradients of loss wrt the weights.
    gradients = tape.gradient(loss_value, model.trainable_weights)
    # Update the weights of the model.
    optimizer.apply_gradients(zip(gradients, model.trainable_weights))
```

For more in-depth tutorials about Keras, you can check out:

- Introduction to Keras for engineers
- Introduction to Keras for researchers
- Developer guides

Installation & compatibility

Keras comes packaged with TensorFlow 2 as `tensorflow.keras`. To start using Keras, simply install TensorFlow 2.

Keras/TensorFlow are compatible with:

- Python 3.7–3.10
- Ubuntu 16.04 or later
- Windows 7 or later
- macOS 10.12.6 (Sierra) or later.

Support

You can ask questions and join the development discussion:

- In the TensorFlow forum.
- On the Keras Google group.
- On the Keras Slack channel. Use this link to request an invitation to the channel.

You can also post **bug reports and feature requests** (only) in GitHub issues. Make sure to read our guidelines first.

Why this name, Keras?

Keras (κέρας) means *horn* in Greek. It is a reference to a literary image from ancient Greek and Latin literature, first found in the *Odyssey*, where dream spirits (*Oneiroi*, singular *Oneiros*) are divided between those who deceive dreamers with false visions, who arrive to Earth through a gate of ivory, and those who announce a future that will come to pass, who arrive through a gate of horn. It's a play on the words κέρας (horn) / κραίνω (fulfill), and ἐλέφας (ivory) / ἐλεφαίρομαι (deceive).

Keras was initially developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System).

> *"Oneiroi are beyond our unravelling - who can be sure what tale they tell? Not all that men look for comes to pass. Two gates there are that give passage to fleeting Oneiroi; one is made of horn, one of ivory. The Oneiroi that pass through sawn ivory are deceitful, bearing a message that will not be fulfilled; those that come out through polished horn have truth behind them, to be accomplished for men who see them."* Homer, Odyssey 19. 562 ff (Shewring translation).
