

Keras

Prof. Gheith Abandah

References:

Introduction to Keras by Francois Chollet, March 9th, 2018 ([pdf](#))

TensorFlow, *Keras*, <https://www.tensorflow.org/guide/keras>

Introduction

- YouTube Video: *Keras Explained* from Siraj Raval

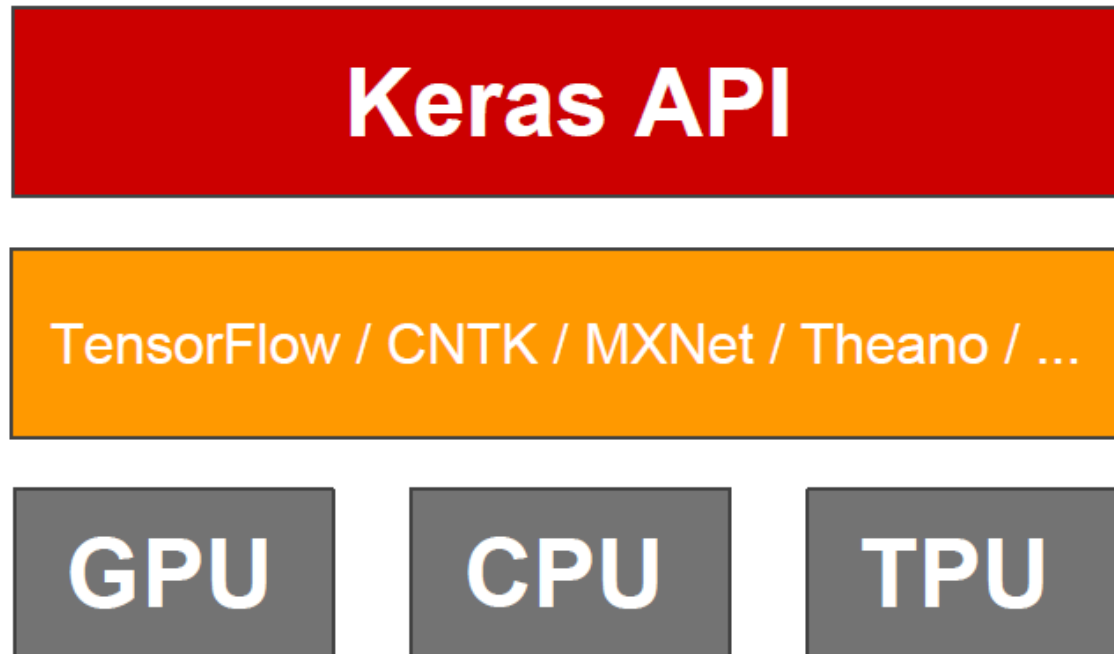
https://youtu.be/j_pJmXJwMLA

Outline

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. The Sequential Model
5. Example
6. Tutorials

1. Introduction

- **Keras** is a high-level API to build and train deep learning models.

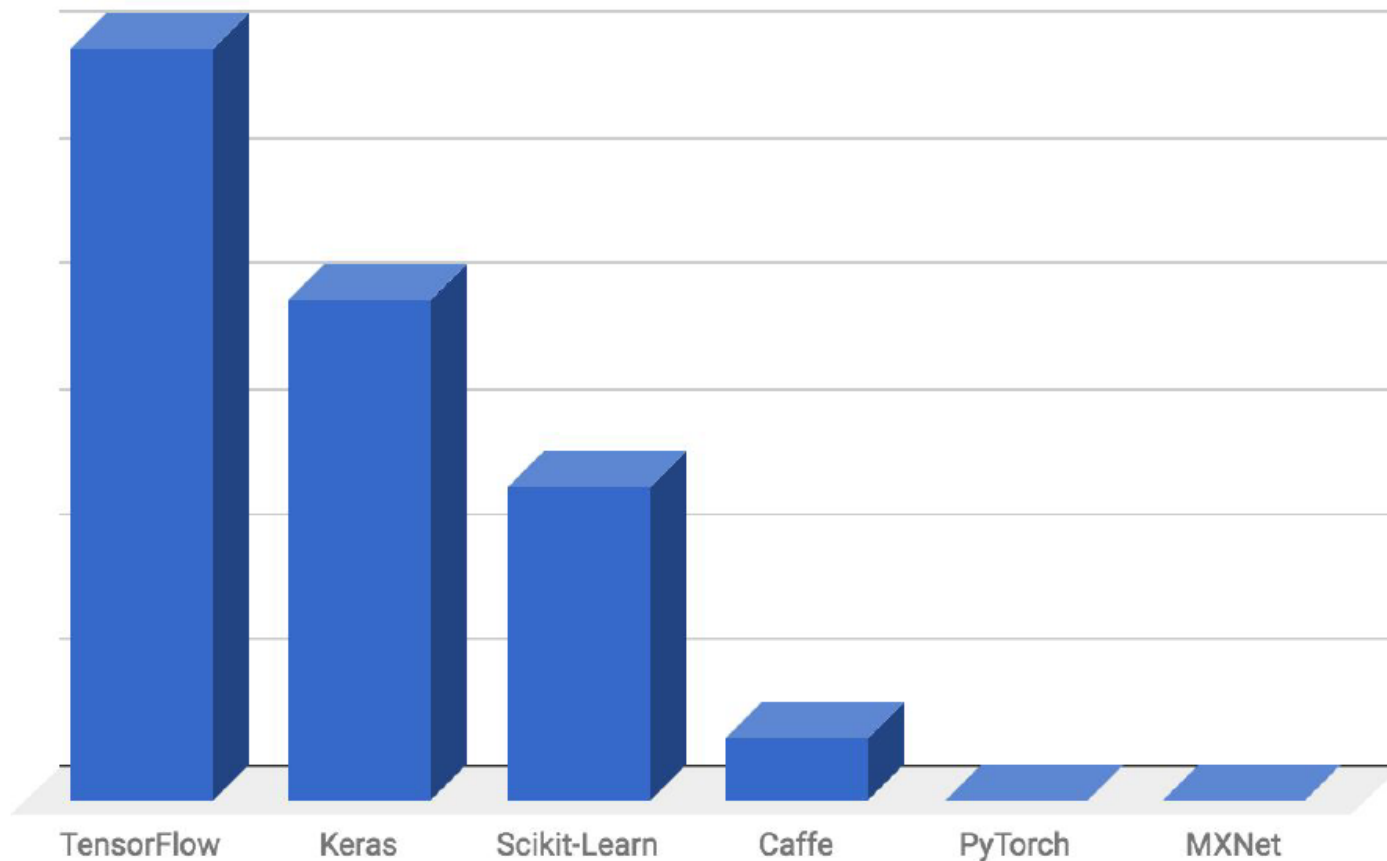


1. Introduction – Advantages

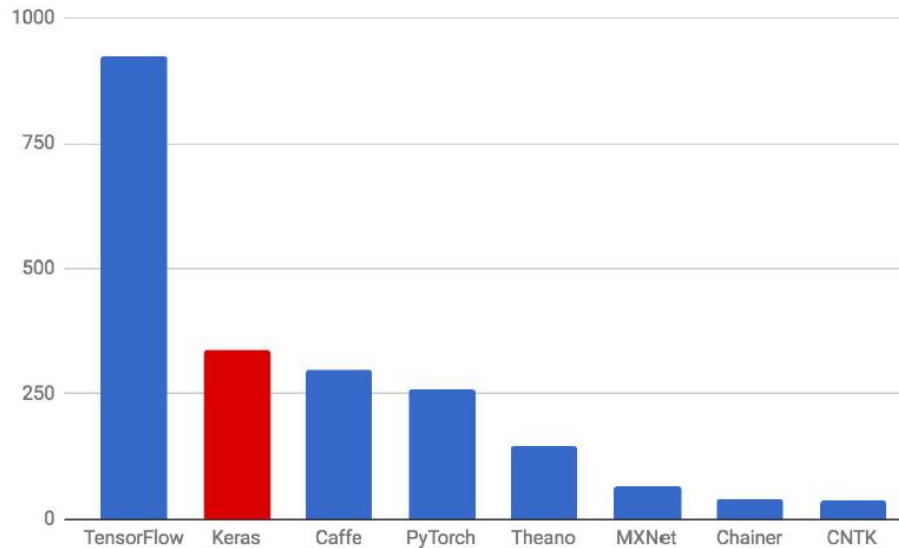
- *User friendly*: Keras has a simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors.
- *Modular and composable*: Keras models are made by connecting configurable building blocks together, with few restrictions.
- *Easy to extend*: Write custom building blocks to express new ideas for research. Create new layers, loss functions, and develop state-of-the-art models.

1. Introduction – Traction

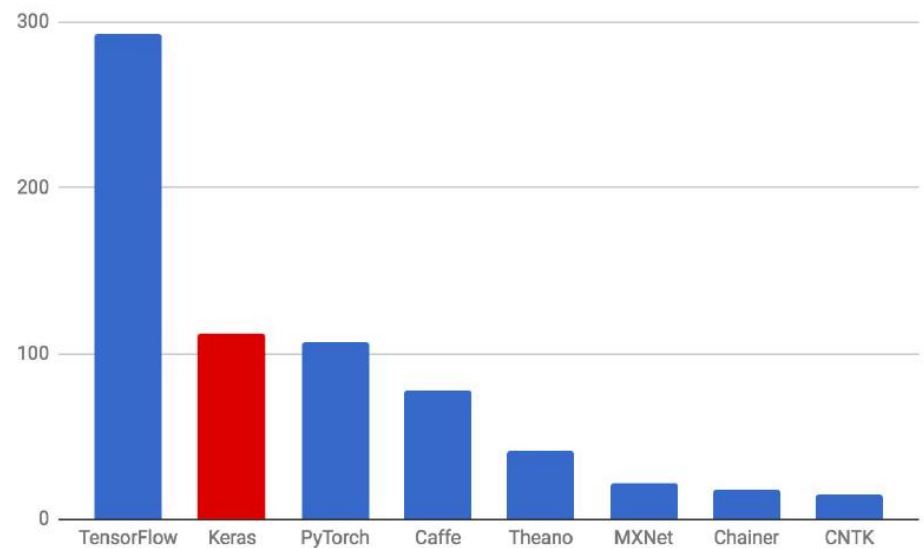
Hacker News jobs board mentions - out of 964 job postings



1. Introduction – Traction



arXiv mentions as of 2018/03/07 (past 3 months)



arXiv mentions as of 2018/03/07 (past 1 month)

2. Keras API Styles

1. The Sequential Model

- Dead simple
- Only for single-input, single-output, sequential layer stacks
- Good for 70+% of use cases

2. The functional API

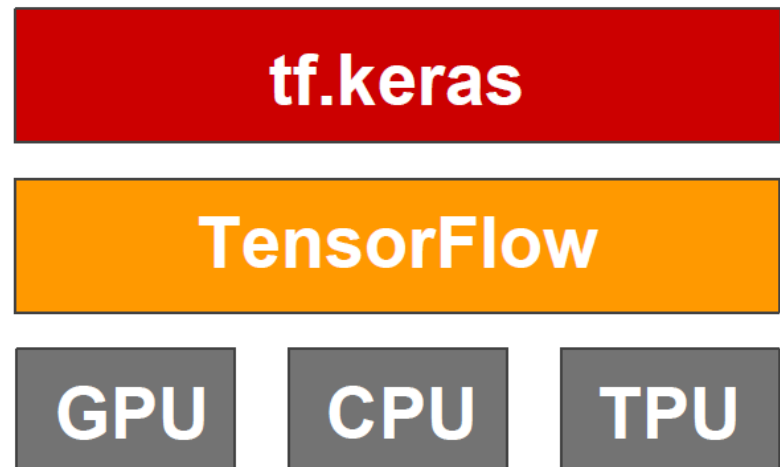
- Like playing with Lego bricks
- Multi-input, multi-output, arbitrary static graph topologies
- Good for 95% of use cases

3. Model subclassing

- Maximum flexibility
- Larger potential error surface

3. TensorFlow Keras

- Keras is the official high-level API of TensorFlow
- tensorflow.keras (tf.keras) module
- Part of core TensorFlow since v1.4
- Full Keras API
- Better optimized for TF
- Better integration with TF-specific



3. TensorFlow Keras

- To import Keras from TensorFlow

- **layers** has:

- Dense
- Activations
- Dropout
- Conv1D, 2D, 3D
- Polling
- RNN, LSTM, GRU
- ...

```
import tensorflow as tf
from tensorflow.keras import layers

print(tf.VERSION)
print(tf.keras.__version__)
```

```
1.12.0
2.1.6-tf
```

4. The Sequential Model

- In Keras, you assemble layers to build models. The most common type of model is a stack of layers: the `tf.keras.Sequential` model.
- To build a simple, fully-connected network (i.e. multi-layer perceptron):

```
model = tf.keras.Sequential()  
# Adds a densely-connected layer with 64 units to the model  
model.add(layers.Dense(64, activation='relu'))  
# Add another:  
model.add(layers.Dense(64, activation='relu'))  
# Add a softmax layer with 10 output units:  
model.add(layers.Dense(10, activation='softmax'))
```

4.1 Configuring the layers

- Configure by passing the name of a built-in function or a callable object.

```
# Create a sigmoid layer:  
layers.Dense(64, activation='sigmoid')  
# Or:  
layers.Dense(64, activation=tf.sigmoid)  
  
# A linear layer with L1 regularization of factor 0.01 applied to the kernel mat  
layers.Dense(64, kernel_regularizer=tf.keras.regularizers.l1(0.01))  
  
# A linear layer with L2 regularization of factor 0.01 applied to the bias vecto  
layers.Dense(64, bias_regularizer=tf.keras.regularizers.l2(0.01))  
  
# A linear layer with a kernel initialized to a random orthogonal matrix:  
layers.Dense(64, kernel_initializer='orthogonal')  
  
# A linear layer with a bias vector initialized to 2.0s:  
layers.Dense(64, bias_initializer=tf.keras.initializers.constant(2.0))
```

The default is no activation function, i.e., linear layer.

4.2 Set up training

- After the model is constructed, configure its learning process by calling the **compile** method:

```
model.compile(optimizer=tf.train.AdamOptimizer(0.001),  
              loss='categorical_crossentropy',  
              metrics=['accuracy'])
```

- Arguments
 - **optimizer**: The training procedure: AdamOptimizer, RMSPropOptimizer, or GradientDescentOptimizer.
 - **loss**: The function to minimize: mean square error (mse), categorical_crossentropy, and binary_crossentropy.
 - **metrics**: Used to monitor training.

4.3 Training

1. Input NumPy data

```
import numpy as np

data = np.random.random((1000, 32))
labels = np.random.random((1000, 10))

model.fit(data, labels, epochs=10, batch_size=32)
```

```
Epoch 1/10
1000/1000 [=====] - 0s 254us/step - loss: 11.5936 - categorical_accuracy: 0.0
Epoch 2/10
1000/1000 [=====] - 0s 64us/step - loss: 11.5301 - categorical_accuracy: 0.0
```

Can also monitor progress on validation data

```
model.fit(data, labels, epochs=10, batch_size=32,
          validation_data=(val_data, val_labels))
```

4.3 Training

2. Use the **Datasets API** to scale to large datasets or multi-device training.

```
# Instantiates a toy dataset instance:  
dataset = tf.data.Dataset.from_tensor_slices((data, labels))  
dataset = dataset.batch(32)  
dataset = dataset.repeat()  
  
# Don't forget to specify `steps_per_epoch` when calling `fit`  
model.fit(dataset, epochs=10, steps_per_epoch=30)
```

```
model.fit(dataset, epochs=10, steps_per_epoch=30,  
          validation_data=val_dataset,  
          validation_steps=3)
```

4.4 Evaluate

```
data = np.random.random((1000, 32))
labels = np.random.random((1000, 10))

model.evaluate(data, labels, batch_size=32)

model.evaluate(dataset, steps=30)
```

```
1000/1000 [=====] - 0s 74us/step
30/30 [=====] - 0s 3ms/step

[11.492062791188557, 0.178125]
```

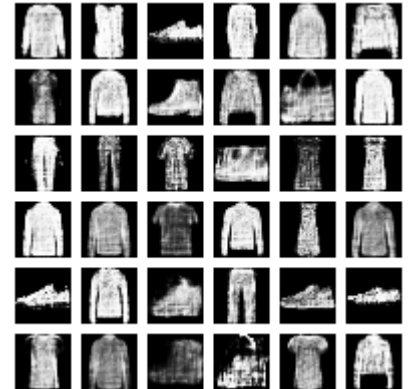

4.5 Predict

```
result = model.predict(data, batch_size=32)  
print(result.shape)
```

```
(1000, 10)
```

5. Example

- Fashion-MNIST with tf.keras by Yufeng Guo



- YouTube Video: *Getting Started with Keras (AI Adventures)* from Google Cloud Platform:
<https://youtu.be/J6Ok8p463C4>
- Code: <https://www.kaggle.com/yufenggg/fashion-mnist-with-keras>

6. Tutorials

- <https://keras.io/>
- <https://www.tensorflow.org/guide/keras>
- Keras Tutorial: Deep Learning in Python from DataCamp,
<https://www.datacamp.com/community/tutorials/deep-learning-python>
- Keras Tutorial: The Ultimate Beginner's Guide to Deep Learning in Python, from EliteDataScience,
<https://elitedatascience.com/keras-tutorial-deep-learning-in-python>

Summary

1. Introduction
2. Keras API Styles
3. TensorFlow Keras
4. The Sequential Model
5. Example
6. Tutorials