PONTIFICIA
UNIVERSIDAD
CATÓLICA DE
VALPARAÍSO

Introducción al Deep Learning

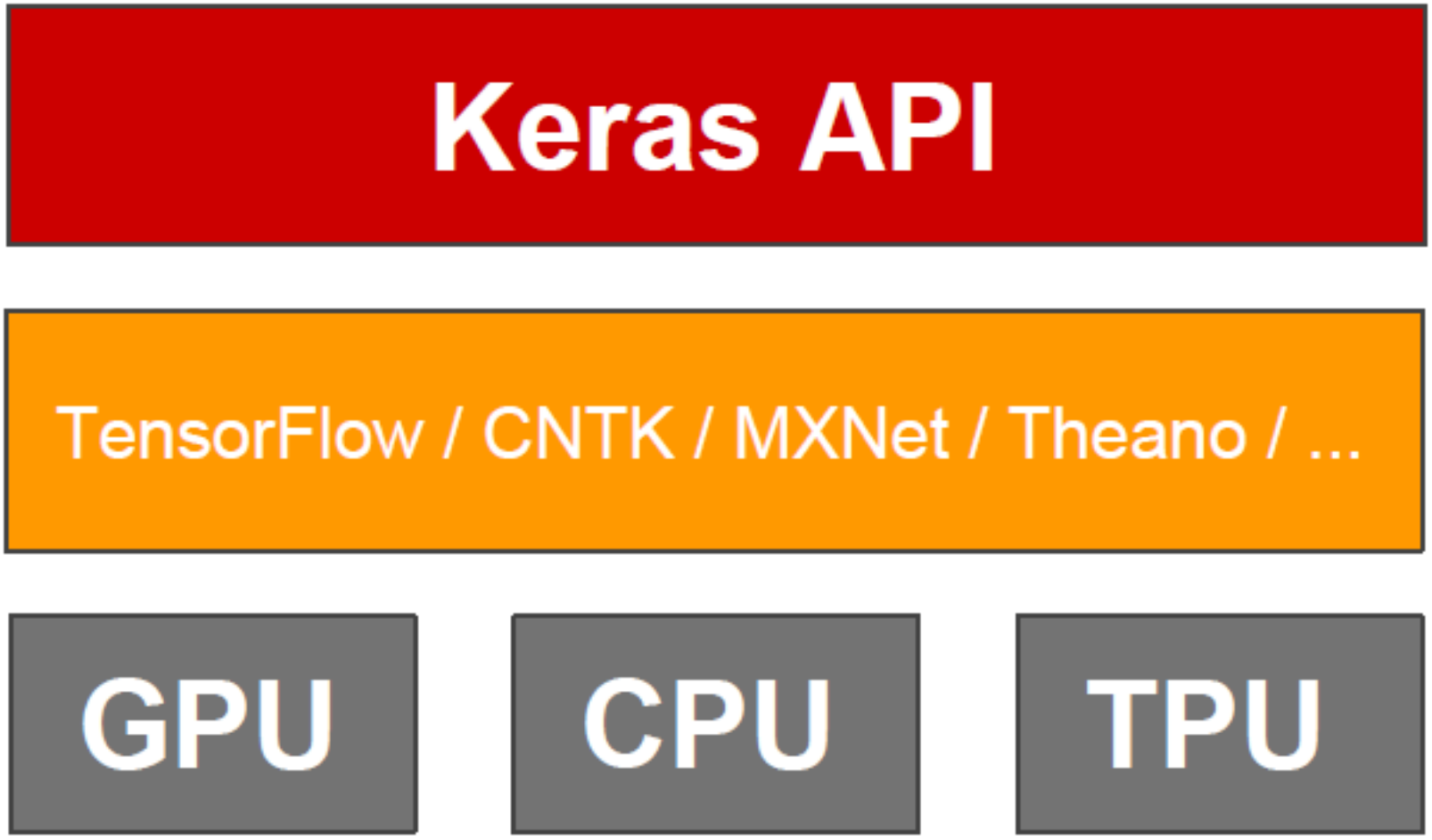
Dr. Ing. Gabriel Hermosilla Vigneau

Introducción a Keras

¿Qué es Keras?

- Keras es una API de redes neuronales de alto nivel, escrita en Python y capaz de ejecutarse sobre TensorFlow o Theano. Fue desarrollado con un enfoque que permite la experimentación rápida. El poder pasar de la idea al resultado con el menor tiempo posible es clave para hacer una buena investigación.
- Keras es una biblioteca de aprendizaje profundo que:
  - Permite realizar prototipos de forma fácil y rápida (a través de la facilidad de uso, la modularidad y la extensibilidad).
  - Admite redes convolucionales y redes recurrentes, así como combinaciones de las dos.
  - Funciona a la perfección en CPU y GPU.

¿Qué es Keras?



¿Qué es Keras?

- Principios:
- **Facilidad de uso**. Keras es una **API** (Application Programming Interface) diseñada para seres humanos, no máquinas. Keras sigue las mejores prácticas para reducir la carga cognitiva: ofrece APIs consistentes y simples, minimiza el número de acciones de usuario requeridas para los casos de uso comunes y proporciona comentarios claros y procesables en caso de error del usuario.
- **Modularidad**. Un modelo se entiende como una secuencia o un gráfico de módulos independientes totalmente configurables que se pueden conectar con la menor cantidad de restricciones posible. En particular, las capas neuronales, las funciones de costo, los optimizadores, los esquemas de inicialización, las funciones de activación, los esquemas de regularización son módulos independientes que puede combinar para crear nuevos modelos.

¿Qué es Keras?

- Principios:
- **Fácil extensibilidad**. Los nuevos módulos son fáciles de agregar (como nuevas clases y funciones), y los módulos existentes brindan amplios ejemplos. El poder crear fácilmente nuevos módulos permite una expresividad total, lo que hace que Keras sea adecuado para la investigación avanzada.
- **Trabaja con Python**. No hay archivos de configuración de modelos separados en un formato declarativo. Los modelos se describen en el código de Python, que es compacto, más fácil de depurar y permite la extensibilidad.

¿Quién hace Keras?

Colaboradores y patrocinadores

¿Qué tiene de especial Keras?

- Un enfoque en la experiencia del usuario.
- Gran adopción en la industria y comunidad de investigación.
- Multi-backend, multiplataforma.
- Fácil producción de modelos.

Empresas que utilizan Keras

etc...

Keras es multi-backend, multiplataforma.

- Desarrollado en Python, R
  - En Unix, Windows, OSX
- Corre el mismo código con ...
  - TensorFlow
  - CNTK
  - Theano
  - MXNet
  - PlaidML
  - Etc.
- CPU, NVIDIA GPU, AMD GPU, TPU ...

Estilos de API

- **El modelo secuencial**
  - Solo para pilas de capas secuenciales de entrada única, salida única
  - Bueno para más del 70% de los casos de uso.
- **La API funcional**
  - Topologías de gráficos estáticos arbitrarios, de múltiples entradas y múltiples salidas
  - Bueno para el 95% de los casos de uso.
- **Modelo de subclases**
  - Máxima flexibilidad.
  - Mayor superficie de error potencial.

API Secuencial

- El modelo secuencial es una pila lineal de capas.
- Puede crear un modelo secuencial pasando una lista de instancias de capa al constructor:

```
import keras
from keras import layers

model = keras.Sequential()
model.add(layers.Dense(20, activation='relu', input_shape=(10,)))
model.add(layers.Dense(20, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.fit(x, y, epochs=10, batch_size=32)
```

API Funcional

- La API funcional de Keras es el camino a seguir para definir modelos complejos, como los modelos de múltiples salidas, los gráficos acíclicos dirigidos o los modelos con capas compartidas.

```
import keras
from keras import layers

inputs = keras.Input(shape=(10,))
x = layers.Dense(20, activation='relu')(x)
x = layers.Dense(20, activation='relu')(x)
outputs = layers.Dense(10, activation='softmax')(x)

model = keras.Model(inputs, outputs)
model.fit(x, y, epochs=10, batch_size=32)
```

Modelo de subclases

```
import keras
from keras import layers

class MyModel(keras.Model):

    def __init__(self):
        super(MyModel, self).__init__()
        self.dense1 = layers.Dense(20, activation='relu')
        self.dense2 = layers.Dense(20, activation='relu')
        self.dense3 = layers.Dense(10, activation='softmax')

    def call(self, inputs):
        x = self.dense1(x)
        x = self.dense2(x)
        return self.dense3(x)

model = MyModel()
model.fit(x, y, epochs=10, batch_size=32)
```

Modelo secuencial de Keras

- El modelo secuencial es una pila lineal de capas nueronales.

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential([
    Dense(32, input_shape=(784,)),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

- También puede simplemente agregar capas a través del método .add ():

```
model = Sequential()
model.add(Dense(32, input_dim=784))
model.add(Activation('relu'))
```

Modelo secuencial de Keras

- El modelo necesita saber la forma de la entrada. Por esta razón, **la primera capa** en un modelo secuencial (y solo la primera, porque las siguientes capas pueden hacer inferencia de forma automática) necesita recibir información sobre su **forma de entrada**.
- Se debe agregar el argumento **input_shape** a la primera capa. Se representan en forma de tupla (es una secuencia ordenada de objetos, o una lista con un número limitado de objetos). En **input_shape**, la dimensión del lote no está incluida. En una imagen a color de 224x224 la forma sería (224, 224, 3).
- Algunas capas de 2 dimensiones, como Dense, admiten la especificación de su forma de entrada a través del argumento **input_dim**, y algunas capas temporales 3D admiten los argumentos **input_dim** y **input_length**.
- Si alguna vez se necesita especificar un tamaño de lote fijo para sus entradas, se puede pasar el argumento **batch_size** a una capa. Si pasa tanto batch_size = 32 como input_shape = (6, 8) a una capa, esperará que cada lote de entradas tenga la forma del lote (32, 6, 8).

Modelo secuencial de Keras

- De esta forma, los siguientes fragmentos son estrictamente equivalentes:

```
model = Sequential()
model.add(Dense(32, input_shape=(784,)))
```

```
model = Sequential()
model.add(Dense(32, input_dim=784))
```

Modelo secuencial de Keras

Compilación

- Antes de entrenar un modelo, debe configurar el proceso de aprendizaje, que se realiza a través del método de compilación.

Recibe tres argumentos:

- **Un optimizador**. Este podría ser un optimizador existente (como rmsprop o adagrad), o una instancia de la clase Optimizer (creado por uno mismo).
- **Una función de pérdida**. Este es la función loss objetivo que el modelo intentará minimizar. Puede ser una función de pérdida existente (como categorical_crossentropy o mse), o puede ser una nueva función objetivo.
- **Una lista de métricas**. Para cualquier problema de clasificación, querrá establecer esto en métricas = ['exactitud']. Una métrica podría ser una métrica existente o una función de métrica personalizada.

Modelo secuencial de Keras

Compilación

- Multiclases:

```
# For a multi-class classification problem
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

- 2 clases:

```
# For a binary classification problem
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

- Regresión:

```
# For a mean squared error regression problem
model.compile(optimizer='rmsprop',
              loss='mse')
```

- Custom:

```
# For custom metrics
import keras.backend as K

def mean_pred(y_true, y_pred):
    return K.mean(y_pred)

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy', mean_pred])
```

Modelo secuencial de Keras

Entrenamiento

- Los modelos Keras están entrenados con matrices Numpy de datos de entrada y etiquetas. Para entrenar un modelo normalmente se usa la función de ajuste (fit).
- Ejemplo para 2 clases:

```
# For a single-input model with 2 classes (binary classification):

model = Sequential()
model.add(Dense(32, activation='relu', input_dim=100))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Generate dummy data
import numpy as np
data = np.random.random((1000, 100))
labels = np.random.randint(2, size=(1000, 1))

# Train the model, iterating on the data in batches of 32 samples
model.fit(data, labels, epochs=10, batch_size=32)
```

Modelo secuencial de Keras

Entrenamiento

- Ejemplo para múltiples clases (10 clases):

```
# For a single-input model with 10 classes (categorical classification):

model = Sequential()
model.add(Dense(32, activation='relu', input_dim=100))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Generate dummy data
import numpy as np
data = np.random.random((1000, 100))
labels = np.random.randint(10, size=(1000, 1))

# Convert labels to categorical one-hot encoding
one_hot_labels = keras.utils.to_categorical(labels, num_classes=10)

# Train the model, iterating on the data in batches of 32 samples
model.fit(data, one_hot_labels, epochs=10, batch_size=32)
```