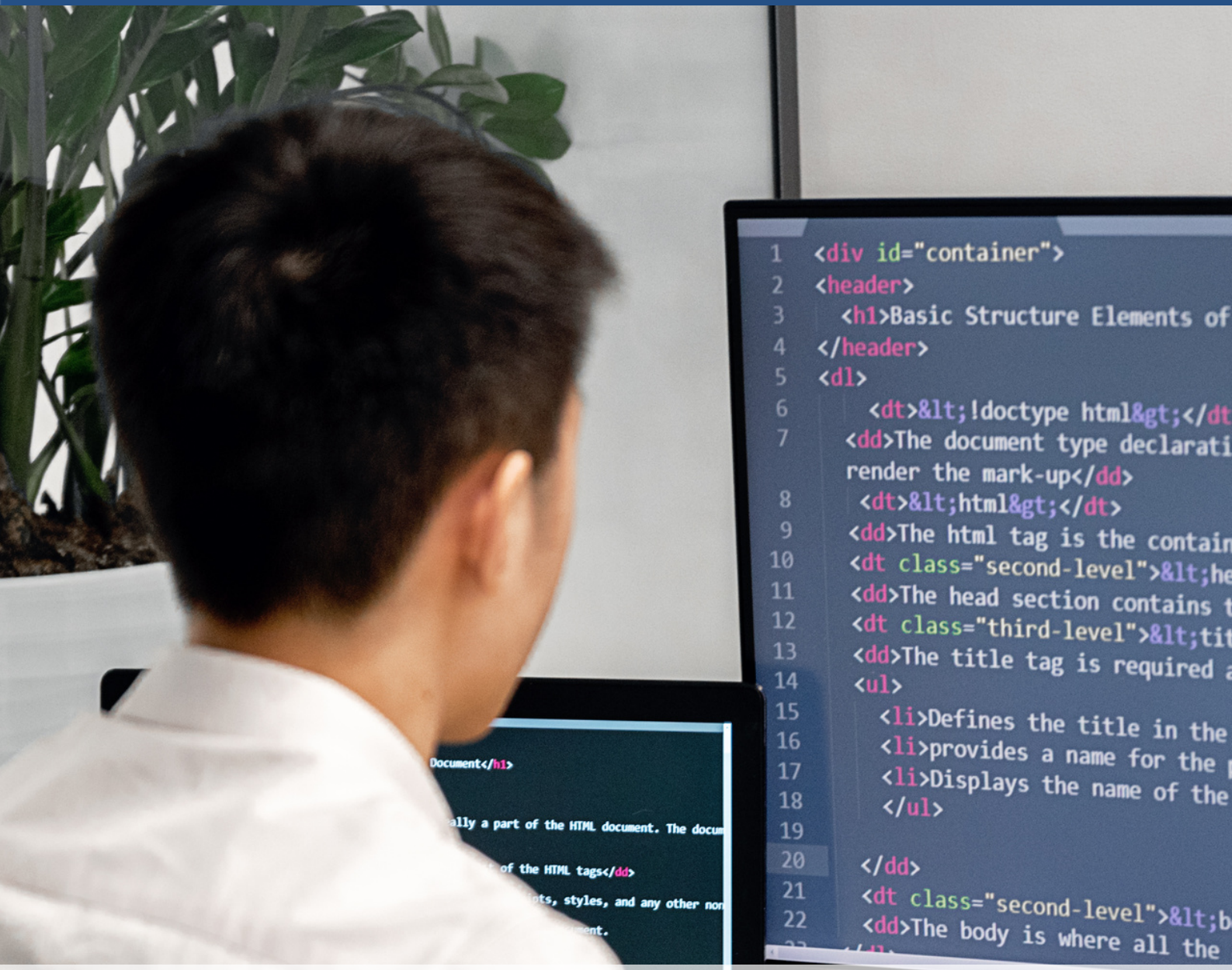
COLOMBIA POTENCIA DE LA VIDA

TIC



Introducción e instalación de Keras

TALENTO TECH | AZ PROYECTOS EDUCATIVOS | UTP Universidad Tecnológica de Pereira

1. Introducción a Keras

Keras es una biblioteca de aprendizaje profundo de alto nivel escrita en Python que facilita la creación y el entrenamiento de modelos de redes neuronales de forma rápida y sencilla. Aunque Keras no es un framework de IA en sí mismo, sino más bien una interfaz de programación de aplicaciones (API), juega un papel crucial en el campo de la inteligencia artificial por varias razones:

• Facilidad de uso

Keras está diseñado para ser fácil de usar y accesible para desarrolladores de todos los niveles de experiencia. Proporciona una API simple y coherente que permite a los usuarios construir y entrenar modelos de redes neuronales con muy pocas líneas de código. Esta simplicidad hace que Keras sea una opción popular para estudiantes, investigadores y profesionales que desean experimentar con el aprendizaje profundo.

• Flexibilidad y modularidad

Keras está diseñado para ser altamente modular y extensible. Permite a los desarrolladores construir modelos complejos utilizando una variedad de capas, funciones de activación, funciones de pérdida y optimizadores. Además, Keras es compatible con múltiples backends de cálculo, incluyendo TensorFlow, Theano y Microsoft Cognitive Toolkit (CNTK), lo que permite a los usuarios elegir la plataforma que mejor se adapte a sus necesidades.

• Integración con otros frameworks

Aunque Keras se puede utilizar de forma independiente, también se integra perfectamente con otros frameworks de aprendizaje profundo, como TensorFlow. De hecho, TensorFlow 2.0 adoptó a Keras como su API de alto nivel predeterminada, lo que significa que los usuarios pueden acceder a todas las capacidades de TensorFlow mientras aprovechan la simplicidad y la facilidad de uso de Keras.

• Comunidad y soporte

Keras cuenta con una gran comunidad de usuarios y desarrolladores que contribuyen con código, documentación y recursos educativos. Esto garantiza un desarrollo continuo y un amplio apoyo para la biblioteca. Además, Keras ofrece una excelente documentación y una serie de tutoriales y ejemplos que facilitan el aprendizaje y la resolución de problemas.

Keras es una herramienta invaluable en el campo del aprendizaje profundo, proporcionando una interfaz intuitiva y fácil de usar para construir y entrenar modelos de redes neuronales. Su simplicidad, flexibilidad y extensibilidad lo hacen indispensable para una amplia gama de aplicaciones en inteligencia artificial, como:

La investigación académica

El desarrollo de aplicaciones comerciales

Componentes o partes importantes de Keras

Ver documentación oficial de Keras en **https://keras.io/api/**

• Models API

La API de modelos de Keras proporciona una forma intuitiva de definir y construir modelos de aprendizaje profundo. Con esta API, puedes crear modelos secuenciales (Sequential) o modelos con topología más compleja (funcionales) utilizando capas (layers) y conexiones entre ellas. Esta API también incluye métodos para compilar, entrenar y evaluar modelos, así como para predecir nuevas muestras.

• Layers API

La API de capas de Keras ofrece una amplia variedad de capas que se pueden utilizar para construir modelos de aprendizaje profundo. Estas capas incluyen:

- **Capas de convolución**
- **Capas de pooling**
- **Capas totalmente conectadas**
- **Capas de activación**
- **Capas de regularización**
- **Capas de normalización**

Las capas se pueden apilar y conectar de forma secuencial o funcional para construir arquitecturas de modelos complejas.

• Callbacks API

Proporciona una forma de personalizar y controlar el proceso de entrenamiento de un modelo. Los callbacks son objetos que se pueden pasar al método fit() de un modelo y que se ejecutan en diferentes puntos durante el entrenamiento, como al inicio o al final de cada época. Esto permite realizar acciones como:

- **Guardar el mejor modelo**
- **Detener el entrenamiento temprano**
- **Ajustar la tasa de aprendizaje**
- **Visualizar métricas en tiempo real**

• Ops API

La API de operaciones (Ops) de Keras proporciona una interfaz para definir y ejecutar operaciones de bajo nivel en tensores. Esto incluye:

- **Operaciones matemáticas**
- **Operaciones de manipulación de tensores**
- **Operaciones de indexación y segmentación**

Estas operaciones se usan internamente en las capas y funciones de Keras para realizar cálculos en los datos de entrada y en los pesos del modelo.

• Optimizadores

Son algoritmos para minimizar la función de pérdida durante el entrenamiento de un modelo. Keras proporciona una variedad de optimizadores como el **descenso de gradiente estocástico (SGD)**, el **algoritmo Adam**, el **algoritmo RMSprop**, y más. Estos pueden ser configurados con diferentes parámetros, como la tasa de aprendizaje, la tasa de decaimiento, el momento, y otros.

• Métricas

Las métricas en Keras son funciones utilizadas para evaluar el rendimiento de un modelo durante el entrenamiento y la evaluación. Keras proporciona una variedad de métricas comunes, como:

- **La precisión (accuracy)**
- **La pérdida (loss)**
- **La precisión top-K**
- **El área bajo la curva ROC (AUC-ROC)**

Estas métricas se pueden utilizar para supervisar el rendimiento del modelo y ajustar los parámetros del entrenamiento.

• Losses

Las funciones de pérdida en Keras son funciones utilizadas para calcular la pérdida entre las predicciones del modelo y las etiquetas verdaderas durante el entrenamiento. Keras proporciona una variedad de funciones de pérdida comunes, como:

- **La entropía cruzada categórica (categorical crossentropy)**
- **La entropía cruzada binaria (binary crossentropy)**
- **El error cuadrático medio (mean squared error)**

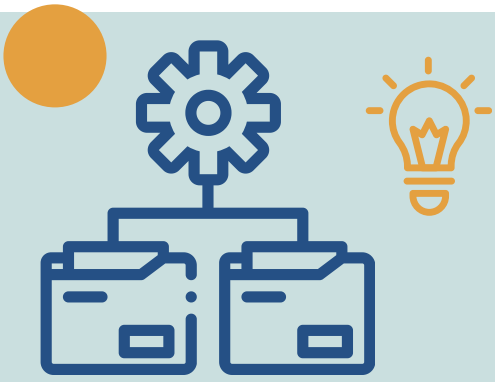
Estas funciones se utilizan para guiar el proceso de optimización y mejorar el rendimiento del modelo.

• Built-in small datasets

Keras incluye una serie de conjuntos de datos pequeños integrados que se pueden utilizar para fines de demostración, experimentación y aprendizaje. Estos conjuntos de datos incluyen:

- MNIST (reconocimiento de dígitos manuscritos),
- CIFAR-10 y CIFAR-100 (clasificación de imágenes),
- IMDB (análisis de sentimientos de críticas de películas),
- Fashion MNIST conjunto de datos, una alternativa a MNIST
- California Housing conjunto de datos de regresión de precios

Estos conjuntos de datos son útiles para familiarizarse con el uso de Keras y para desarrollar y probar modelos de forma rápida y sencilla.



Estas son algunas de las principales componentes o partes importantes de Keras que hacen que sea una biblioteca tan poderosa y versátil para el desarrollo de modelos de aprendizaje profundo en Python. Su API intuitiva, su amplia gama de capas y funciones, y su soporte integrado para optimizadores, métricas y bases de datos.

2. Comenzar a trabajar con Keras

- **Instalar Keras:**

Puedes instalar Keras utilizando pip:

```
pip install keras
```

Esto instalará Keras y todas sus dependencias en tu sistema.

- **Importar Keras:**

Después de instalar Keras, puedes importarlo en tu script de Python usando la siguiente declaración:

```
import keras
```

Implementar una red neuronal utilizando Keras sigue varios pasos generales:

1 Preparación de los datos:

Asegúrate de tener tus datos correctamente estructurados en matrices NumPy u otros formatos compatibles con Keras. Divide tus datos en conjuntos de entrenamiento, validación y prueba si es necesario.

2 Importación de la biblioteca:

Importa las clases y funciones necesarias de Keras, incluyendo **Sequential** para modelos secuenciales y capas específicas como **Dense**, **Conv2D**, **LSTM**, etc., dependiendo del tipo de red neuronal que estés construyendo.

3 Configuración del modelo:

Crea una instancia de un modelo secuencial (**Sequential**) o funcional de Keras. Añade capas al modelo, especificando el número de neuronas, la función de activación, la regularización, entre otros, según tus necesidades. Puedes hacerlo usando el método add del modelo.

4 Compilación del modelo:

Utiliza el método **compile** para compilar el modelo, especificando el optimizador, la función de pérdida y las métricas que se utilizarán para evaluar el rendimiento del modelo durante el entrenamiento.

Ver documentación oficial:
https://keras.io/api/models/model_training_apis/

5 Entrenamiento del modelo:

Utiliza el método **fit** para entrenar el modelo con los datos de entrenamiento. Especifica el número de épocas de entrenamiento y el tamaño del lote (batch size).

6 Evaluación del modelo:

Utiliza el método **evaluate** para evaluar el rendimiento del modelo en los datos de validación o prueba.

7 Predicciones:

Utiliza el método **predict** para realizar predicciones en nuevos datos utilizando el modelo entrenado.

Es importante tener en cuenta que la implementación exacta puede variar según el tipo de red neuronal que estés construyendo (redes totalmente conectadas, convolucionales, recurrentes, etc.) y los requisitos específicos de tu problema. Sin embargo, estos pasos generales proporcionan una guía básica para implementar redes neuronales utilizando Keras.

Ejemplo básico de cómo implementar una red neuronal utilizando Keras en Python:

- **Importación de bibliotecas:**

```
import numpy as np
import keras
```

Importamos las bibliotecas necesarias. **keras** es una API de alto nivel para construir y entrenar modelos de redes neuronales.

- **Construcción del modelo:**

```
# Build a simple Sequential model+

model = keras.Sequential([keras.layers.Dense(units=1,
input_shape=[1])])
```

Creamos un modelo secuencial utilizando **Sequential** de Keras, lo que significa que las capas se apilan en secuencia.

Agregamos una capa densa (**Dense**) al modelo. Esta capa tiene una sola unidad (**units=1**) y espera una entrada de una dimensión (**input_shape=[1]**).

- **Compilación del modelo:**

```
# Compile the model
model.compile(optimizer='sgd', loss='mean_squared_error')
```

Compilamos el modelo utilizando el optimizador de descenso de gradiente estocástico (**sgd**) y la función de pérdida de error cuadrático medio (**mean_squared_error**). Esto configura el modelo para la fase de entrenamiento.

- **Declaración de entradas y salidas para el entrenamiento:**

Para este ejemplo, vamos a asumir que tenemos como entradas **X** los números del 1 al 5, y como salidas **Y** vamos asumir que tenemos una función que toma cada x la multiplica por 10 y le suma 1 así:

X	Y
1	11
2	21
3	31
4	41
5	51

```
# Declare model inputs and outputs for training
X = np.array([1.0, 2.0, 3.0, 4.0, 5.0], dtype=float)
y = np.array([11.0,21.0,31.0,41.0,51.0], dtype=float)
```

Declaramos los datos de entrada (**X**) y los resultados deseados (**Y**) para el entrenamiento del modelo.

- **Entrenamiento del modelo:**

```
# Train the model
model.fit(X, y, epochs=100)
```

Entrenamos el modelo utilizando los datos de entrada (**X**) y los resultados deseados (**Y**) durante 100 épocas. Durante cada época, el modelo ajusta sus pesos para minimizar la función de pérdida especificada en el paso de compilación.

- **Realización de una predicción:**

```
# Make a prediction
print(model.predict([10.0]))
```

Realizamos una predicción utilizando el modelo entrenado, pasando una entrada de prueba de 10.0. El modelo utiliza los pesos ajustados durante el entrenamiento para predecir el resultado correspondiente.

</>

Este código implementa y entrena un modelo de red neuronal muy simple utilizando Keras. El modelo tiene una sola capa densa y se entrena para predecir una salida a partir de una única entrada.