

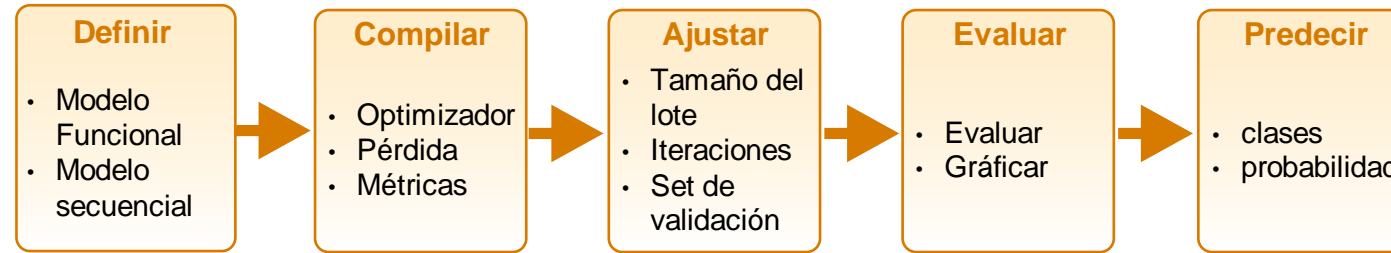
Aprendizaje Profundo con Keras3 : : GUÍA RÁPIDA



Introducción

Keras es una API de redes neuronales de alto nivel desarrollada con un enfoque en permitir una experimentación rápida. Es compatible con múltiples back-ends, incluidos TensorFlow, Jax y Torch.

Los backends como TensorFlow son bibliotecas matemáticas de nivel inferior para crear arquitecturas de redes neuronales profundas. El paquete keras3 R facilita el uso de Keras con cualquier backend en R.



<https://keras.posit.co>

<https://www.manning.com/books/deep-learning-with-r-second-edition>

INSTALACIÓN

El paquete keras3 R utiliza la biblioteca Python Keras. Puede instalar todos los requisitos previos directamente desde R.

Vea `?keras3::install_keras` para más detalles y opciones.

```
library(keras3)
reticulate::install_python()
install_keras()
```

Esto instala las bibliotecas necesarias en un entorno virtual llamado 'r-keras'. Detectará automáticamente si hay una GPU disponible.

ENTRENAMIENTO DE UN RECONOCEDOR DE IMÁGENES EN DATOS MNIST

5041

DEFINIR UN MODELO

API funcional: keras_input() y keras_model()
Definir un Modelo Funcional con entradas y salidas.
`inputs <- keras_input(<input-shape>)`
`outputs <- inputs |>`
`layer_dense() |> layer_...`
`model <- keras_model(inputs, outputs)`

API secuencial: keras_model_sequential()
Definir un Modelo Secuencial compuesto por una pila lineal de capas

```
model <-
  keras_model_sequential(<input-shape>) |>
  layer_dense() |> layer_...
```

API de subclases: Model()
Subclase de la base clase del Modelo

COMPILAR UN MODELO

compile(object, optimizer, loss, metrics, ...)
Configuración de un modelo de Keras para el entrenamiento

AJUSTAR UN MODELO

fit(object, x = NULL, y = NULL, batch_size = NULL, epochs = 10, verbose = 1, callbacks = NULL, ...)
Entrenar un modelo de Keras para un número fijo de iteraciones (epochs)

Personalizar entrenamiento:

- Proporcionar callbacks a **fit()**:
- Definir **Callback()** personalizados.
- Llamar **train_on_batch()** en un bucle de entrenamiento personalizado.
- Subclase de **Model()** e implemente un método **train_step** personalizado.
- Escriba un bucle de entrenamiento totalmente personalizado. Actualice los pesos con **model\$optimizer\$apply(gradients, weights)**

INSPECCIONAR UN MODELO

print(model) Imprimir un resumen de un modelo Keras

plot(model, show_shapes = FALSE, show_dtype = FALSE, show_layer_names = FALSE, ...)

Gráfica un modelo Keras

EVALUAR UN MODELO

evaluate(object, x = NULL, y = NULL, batch_size = NULL) Evaluar un modelo Keras

PREDECIR

predict() Generar predicciones usando un modelo Keras

predict_on_batch() Devuelve predicciones para un solo lote de muestras.

GUARDAR/CARGAR UN MODELO

save_model(); load_model()
Guardar/cargar modelos usando el formato ".keras".

save_model_weights(); load_model_weights()
Guardar/cargar pesos del modelo hacia/desde archivos ".h5".

save_model_config(); load_model_config()
Guardar/cargar arquitectura del modelo hacia/desde archivos ".json".

DESPLEGAR

Exporte solo el paso hacia adelante del modelo entrenado para el servicio de inferencia.

export_savedmodel(model, "my-saved-model/1")
Guarda un TF SavedModel para la inferencia.

rconnect::deployTFModel("my-saved-model")
Implementa un TF SavedModel en Connect para la inferencia.

CAPAS PRINCIPALES

layer_dense() Adición de una capa densamente conectada a una salida.

layer_einsum_dense() Adición de una capa densa con dimensionalidad arbitraria

layer_activation() Aplicar una función de activación a una salida.

layer_dropout() Elimina pesos de la entrada

layer_reshape() Cambia la forma de una salida a una forma determinada

layer_permute() Permutar las dimensiones de una entrada de acuerdo con un patrón determinado

layer_repeat_vector() Repite la entrada n veces

layer_lambda(object, f) Envuelve la expresión arbitraria como una capa







layer_activity_regularization() Capa que aplica una actualización a la actividad de entrada basada en la función de coste

layer_masking() Enmascara una secuencia mediante un valor de máscara para omitir periodos de tiempo





layer_flatten() Aplana una entrada

Más capas

CAPAS CONVOLUCIONALES

-  **layer_conv_1d()** 1D, e.g. convolución temporal
-  **layer_conv_2d_transpose()** Transpuesta 2D (desconvolución)
layer_conv_2d() 2D, e.g. convolución espacial sobre imágenes
-  **layer_conv_3d_transpose()** Transpuesta 3D
layer_conv_3d() 3D, e.g. convolución espacial sobre volumen
- layer_conv_lstm_2d()** LSTM concolucional
- layer_separable_conv_2d()** 2D separable en profundidad
-  **layer_upsampling_1d()**
layer_upsampling_2d()
layer_upsampling_3d() Capa de sobremuestreo
-  **layer_zero_padding_1d()**
layer_zero_padding_2d()
layer_zero_padding_3d() Capa de relleno 0
-  **layer_cropping_1d()**
layer_cropping_2d()
layer_cropping_3d() Capa de recorte

CAPAS DE AGRUPACIÓN

-  **layer_max_pooling_1d()**
layer_max_pooling_2d()
layer_max_pooling_3d() Agrupación máxima de 1D a 3D
-  **layer_average_pooling_1d()**
layer_average_pooling_2d()
layer_average_pooling_3d() Agrupación media de 1D a 3D
-  **layer_global_max_pooling_1d()**
layer_global_max_pooling_2d()
layer_global_max_pooling_3d() Agrupación máxima global
-  **layer_global_average_pooling_1d()**
layer_global_average_pooling_2d()
layer_global_average_pooling_3d() Agrupación media global

Preprocesamiento

PREPROCESAMIENTO DE IMÁGENES

- Cargar imágenes**
image_dataset_from_directory()
Cree un conjunto de datos TF a partir de archivos de imagen en un directorio.
- image_load(), image_from_array(), image_to_array(), image_array_save()**
Trabajar con instancias de imagen PIL
- Transformar imágenes**
op_image_crop()
op_image_extract_patches()
op_image_pad()
op_image_resize()
op_image_affine_transform()
op_image_map_coordinates()
op_image_rgb_to_grayscale()
Operaciones que transforman tensores de imagen de forma determinista.
- image_smart_resize()**
Cambiar el tamaño de las imágenes sin distorsión de la relación de aspecto.
- Capas de imagen**
Capas de preprocesamiento de imágenes integradas. Tenga en cuenta que cualquier función de operación de imagen también se puede usar como una capa en un modelo o en `layer_lambda()`.
- Capas de preprocesamiento de imágenes**
layer_resizing()
layer_rescaling()
layer_center_crop()
- Capas de aumento de imágenes**
Capas de preprocesamiento que aumentan aleatoriamente las entradas de imagen durante el entrenamiento.
layer_random_crop()
layer_random_flip()
layer_random_translation()
layer_random_rotation()
layer_random_zoom()
layer_random_contrast()
layer_random_brightness()

PREPROCESAMIENTO DE SECUENCIAS

- timeseries_dataset_from_array()**
Genere un conjunto de datos TF de ventanas deslizantes a lo largo de una serie temporal proporcionada como matriz.
- audio_dataset_from_directory()**
Genere un conjunto de datos TF a partir de archivos de audio.
- pad_sequences()**
Secuencias de relleno de la misma longitud

Preprocesamiento

PREPROCESAMIENTO DE TEXTO

- text_dataset_from_directory()**
Generar un conjunto de datos TF a partir de archivos de texto en un directorio.
- layer_text_vectorization(), get_vocabulary(), set_vocabulary()**
Asigne textos a secuencias enteras.

CARACTERÍSTICAS NUMÉRICAS

- PREPROCESAMIENTO**
layer_normalization()
Normaliza las características continuas.

- layer_discretization()**
Agrupa características continuas por rangos

PREPROCESAMIENTO DE CARACTERÍSTICAS

- CATEGÓRICAS**
layer_category_encoding()
Codificar características enteras

- layer_hashing()**
Características categóricas de hash y bin

- layer_hashed_crossing()**
Cruce de características usando el "truco de hashing"

- layer_string_lookup()**
Asigne cadenas a índices (posiblemente codificados)

- layer_integer_lookup()**
Asigne enteros a índices (posiblemente codificados)

DATOS TABULARES

- Utilidad integral para el preprocesamiento y la codificación de datos estructurados. Defina un espacio de entidades a partir de una lista de columnas de tabla (entidades).
feature_space <- layer_feature_space(features = list(<features>))

- Adaptar el espacio de entidades a un dataset
adapt(feature_space, dataset)

- Utilice la capa de preprocesamiento de `feature_space` adaptada como una capa en un modelo de Keras o en la canalización de entrada de datos con `tfdatasets::dataset_map()`

Características disponibles:

- feature_float()**
feature_float_rescaled()
feature_float_normalized()
feature_float_discretized()

- feature_integer_categorical()**
feature_integer_hashed()

- feature_string_categorical()**
feature_string_hashed()

- feature_cross()**
feature_custom()



Modelos entrenados

Las aplicaciones de Keras son modelos de aprendizaje profundo que están disponibles con pesos previamente entrenados. Estos modelos se pueden utilizar para la predicción, la extracción de características y el ajuste preciso.
application_mobilenet_v3_large()
application_mobilenet_v3_small()
MobileNetV3 Model, pre-entrenado en ImageNet

application_efficientnet_v2s()
application_efficientnet_v2m()
application_efficientnet_v2l()
EfficientNetV2 Model, pre-entrenado on ImageNet

application_inception_resnet_v2()
application_inception_v3()
Inception-ResNet v2 y v3 modelos, con pesos entrenados en ImageNet

application_vgg16(); application_vgg19()
VGG16 y VGG19 modelos

application_resnet50() ResNet50 modelo

application_nasnet_large()
application_nasnet_mobile()
NASNet arquitectura de modelo

IMAGENET

[ImageNet](https://www.image-net.org/) es una gran base de datos de imágenes con etiquetas, ampliamente utilizada para el aprendizaje profundo

application_preprocess_inputs()
application_decode_predictions()
Preprocesa un tensor que codifica un lote de imágenes para una aplicación y descodifica las predicciones de una aplicación

Callbacks

Un callback es un conjunto de funciones que se aplicarán en determinadas etapas del procedimiento de entrenamiento. Puede usar callbacks para obtener una vista de los estados internos y las estadísticas del modelo durante el entrenamiento.

callback_early_stopping() Detener el entrenamiento cuando una cantidad supervisada a dejado de mejorar
callback_learning_rate_scheduler() Agenda de tasa de aprendizaje
callback_tensorboard() TensorBoard visualizaciones básicas

