**Configurando um ambiente virtual em python com reticulate para rodar tensorflow e keras usando GPU no MacOS modelo M1**

Vamos integrar o ambiente Conda configurado para o TensorFlow com aceleração pela GPU ao R, usando o pacote reticulate.

1 - Baixe o instalador do Miniforge específico para o Mac M1

https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-MacOSX-arm64.sh

**Abra o terminal (bash)**

2 – Execute o script de instalação:

2.1 - Abra o terminal e navegue até o diretório onde o instalador foi baixado (geralmente a pasta Downloads):

bash Miniforge3-MacOSX-arm64.sh

3 - Criar um ambiente Conda:

conda create -n tf-m1 python=3.9

3.1 - Ativando o ambiente conda tf-m1

conda activate tf-m1

4 - Instalar TensorFlow e dependências:

conda install -c apple tensorflow-deps

pip install tensorflow-macos

pip install tensorflow-metal

**Abra o R (configurando)**

5 - Instalar o pacote reticulate no R:

install.packages("reticulate")

6 - Configurar reticulate para usar o ambiente Conda:

library(reticulate)

use\_condaenv("tf-m1", required = TRUE)

7 - Instalar o pacote keras no R:

install.packages("keras")

8 - Carregar o pacote keras e usar o TensorFlow com GPU:

library(keras)

library(tensorflow)

# Verificar se a GPU está disponível

tf$config$list\_physical\_devices('GPU')

# Carregar o dataset MNIST

mnist <- dataset\_mnist()

# Definir e compilar o modelo

model <- keras\_model\_sequential() %>%

layer\_dense(units = 128, activation = 'relu', input\_shape = c(784)) %>%

layer\_dropout(rate = 0.2) %>%

layer\_dense(units = 10, activation = 'softmax')

model %>% compile(

loss = 'sparse\_categorical\_crossentropy',

optimizer = optimizer\_adam(),

metrics = c('accuracy')

)

# Preparar os dados

x\_train <- mnist$train$x / 255

x\_train <- array\_reshape(x\_train, c(nrow(x\_train), 784))

y\_train <- mnist$train$y

x\_test <- mnist$test$x / 255

x\_test <- array\_reshape(x\_test, c(nrow(x\_test), 784))

y\_test <- mnist$test$y

# Treinar o modelo

model %>% fit(x\_train, y\_train, epochs = 5, batch\_size = 32, validation\_split = 0.2)

# Avaliar o modelo

model %>% evaluate(x\_test, y\_test)

**Verificação Adicional**

Para garantir que o TensorFlow está usando a GPU, você pode verificar as mensagens de log durante a execução do modelo. Se a GPU estiver sendo usada, você verá mensagens indicando que os kernels estão sendo executados na GPU.

**Comandos extras**

Mostrar qual o ambiente virtual e versão de python que está sendo utilizada:

py\_config()

Verificar se a GPU está disponível

print(tf$config$list\_physical\_devices('GPU'))

Se o retorno for lista vazia = GPU não está funcionando

Se o retorno for lista 1 = GPU OK