TensorFlow v2.0

CHEAT SHEET

FEREBELL

TENSORFLOW

TensorFlow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que les permite a los investigadores innovar con el aprendizaje automático y, a los desarrolladores, compilar e implementar con facilidad aplicaciones con tecnología de AA.

GOOGLE COLABORATORY

Google Colab o también llamado Colaboratory es un servicio en la nube, ofrecido por Google de forma gratuita. Se basa en el entorno Jupyter Notebook y está destinado a la capacitación e investigación en aprendizaje automático.

ESTILOS DE MODELADO EN TF

En tf.keras existen tres formas básicas de modelar una red neuronal (RN), estas son:

- API secuencial

```
modelo = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(10, activation='softmax')
])
```

- API funcional

```
entradas = keras.Input(shape=(784,), name='img')
x = layers.Dense(64, activation='relu')(entradas)
x = layers.Dense(64, activation='relu')(x)
salidas = layers.Dense(10, activation='softmax')(x)

model = keras.Model(inputs=entradas, outputs=salidas)
```

- Subclasificador de modelos

```
class MyModel(tf.keras.Model):
  def __init__(self):
    super(MyModel, self).__init__()
    self.flatten = Flatten(input_shape=(28, 28))
    self.d1 = Dense(128, activation='relu')
    self.d2 = Dense(10, activation='softmax')

  def call(self, x):
    x = self.flatten(x)
    x = self.d1(x)
    return self.d2(x)

model = MyModel()
```

MAS EN NUESTRAS REDES:

COMENZANDO CON TF

Implementando este código crearás tu primera red neuronal que será capaz de clasificar entre imágenes de números del 0 al 9.

```
[1] import tensorflow as tf
    mnist = tf.keras.datasets.mnist
```

```
[2] (x_ent, y_ent),(x_val, y_val) = mnist.load_data()
    x_ent, x_val = x_ent / 255.0, x_val / 255.0
```

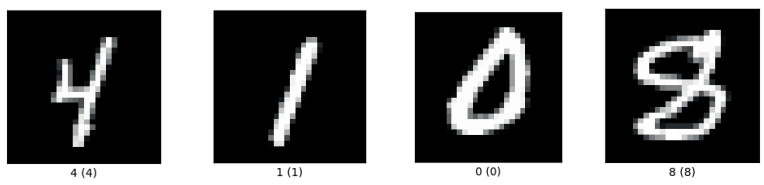
```
[3] modelo = tf.keras.models.Sequential([
      tf.keras.layers.Flatten(input_shape=(28, 28)),
      tf.keras.layers.Dense(128, activation='relu'),
      tf.keras.layers.Dense(10, activation='softmax')
    ])
```

```
[4] modelo.compile(optimizer='SGD',
                   loss='sparse_categorical_crossentropy',
                   metrics=['accuracy'])
```

```
[5] modelo.fit(x_ent, y_ent, epochs=5,
               validation_data= (x_val, y_val))
```

METIENDONOS AL CODIGO

[1] Importando la librería de tensorflow y la base de datos mnist, que contiene imágenes de números del 0 al 9, que se ven como:



[2] Se asignan los datos de entrenamiento a x_ent, que contiene las imágenes, y_ent corresponde a el número en la imagen. Se realiza el mismo procedimiento para los datos de validacion, con las variables x_val y y_val, respectivamente.

[3] En este paso creamos el modelo agregando dos capas ocultas, la primera contiene 128 neuronas y la segunda sólo 10 neuronas que representan la clasificación de los datos del 0 al 9.

[4] Para medir el error del modelo le asignamos la función pérdida sparse_categorical_crossentropy, para que el modelo aprenda utilizamos el optimizador SGD y como métrica extra medimos la exactitud (accuracy).

[5] Finalmente entrenamos el modelo con los datos x_ent, y_ent y probamos su desempeño con los datos x_val y y_val.