

## **KERAS Y TENSORFLOW**

**FREDDY ALEJANDRO FLOREZ BOHORQUEZ**

**ORLANDO CORREA MARTINEZ**

**LAURA LOPEZ ARBELAEZ**

**SEMESTRE 7**

**PROFESOR**

**INGENIERO ARIEL RUIZ**

**MATERIA**

**INTELIGENCIA ARTIFICIAL**

**UNIVERSIDAD SANTIAGO DE CALI**

**CALI-VALLE**

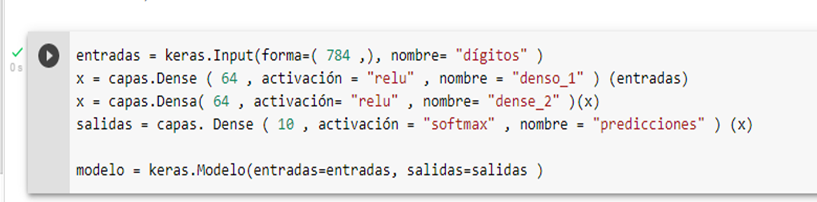
**2022**

**TALLER KERAS Y TENSORFLOW**

1. **Verificar el funcionamiento de los siguientes ejemplos y concluya sobre cada uno de los parámetros de configuración de los modelos, especialmente las características de los modelos (ejemplo Dense) y los optimizadores (ejemplo SDG) que ofrece Keras:**

**Desarrollo**

En el siguiente documento explicaremos los diferentes modelos además los diferentes parámetros que usan como lo son los optimizadores, las funciones de pérdida y las métricas asociado lo de el como lo realizan. Además hay modelos de entrenamiento, evaluación y predicción.

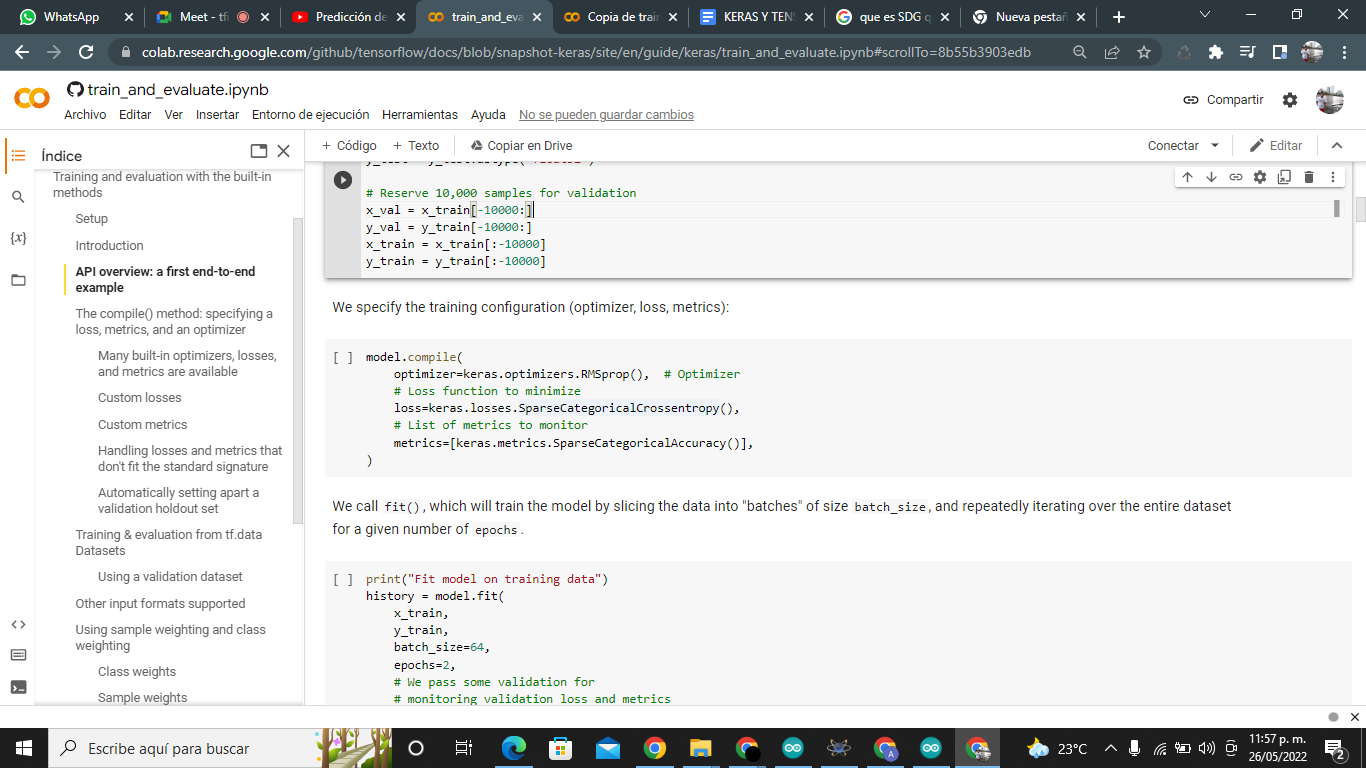


**figura 1 entradas, salidas capas dense y softmax**

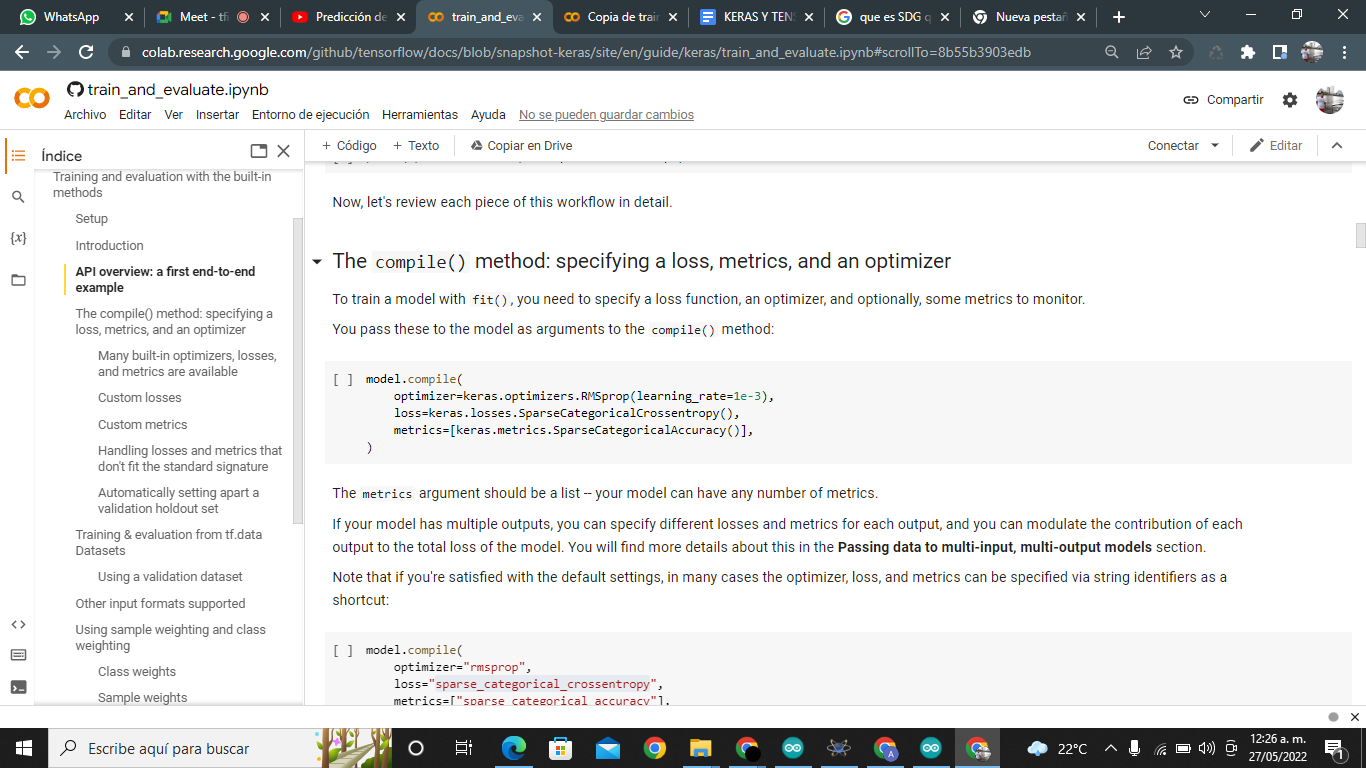


**figura 2 Datos de MNIST y preparando el dataset de entrenamiento y pruebas**

En el siguiente ejemplo se inicia con la importación de librerías de keras y tensor flow esto debido a que se usa matrices en el conjunto de datos MNIST ver figura 2 para la parte de parámetro y argumentos. Siguiendo con nuestra red comenzamos a definir la arquitectura que va a usar en la cual se empieza definiendo las entradas y las salidas ver figura 1, además de eso el modelo cuenta con dos capas dense la cuales se componen de sesenta y cuatro neuronas además que vienen asociado con la función de activación relu este proceso mencionado anteriormente se realiza para cada capa dense para finalizar una capa del softmax con 10 neuronas, luego de haber definido el modelo se procede a su respectiva compilación y carga de los datos a usar además de eso se define o declaran las variables a usar para el entrenamiento y los de prueba además esta es la arquitectura que se usa para los ejemplos posteriores.

****

**figura 3 Modelo de compilación**

****

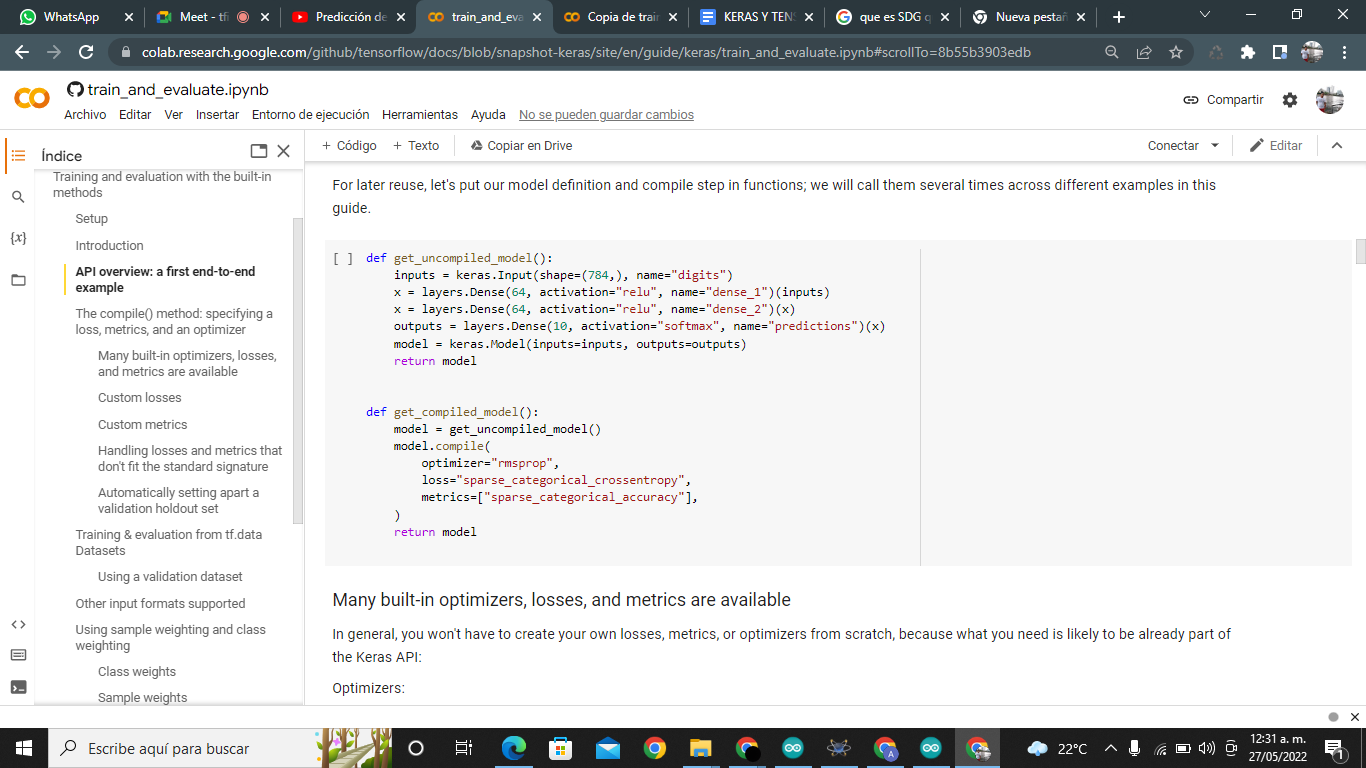
**Figura 3 Modelo de compilación**

Para el modelo de compilación ver la figura 3 en este modelo lo que busca es antes de entrenar la red es configurar la arquitectura de la red. Debido a lo anterior el modelo de compilación posee tres parámetros a continuación los veremos empezamos con los optimizadores, las funciones de pérdida loss y las métricas usadas por los modelos, cada ejemplo posee unas características en específico para ciertas aplicaciones en sí y las iremos viendo a lo largo de las explicaciones de los ejemplos del modelo.

En la siguiente figura 4 se ve como se prepara el modelo de compilación previo al entrenamiento definiendo su arquitectura por medio de sus capas Dense y las funciones de activación relu y la capa de softmax.

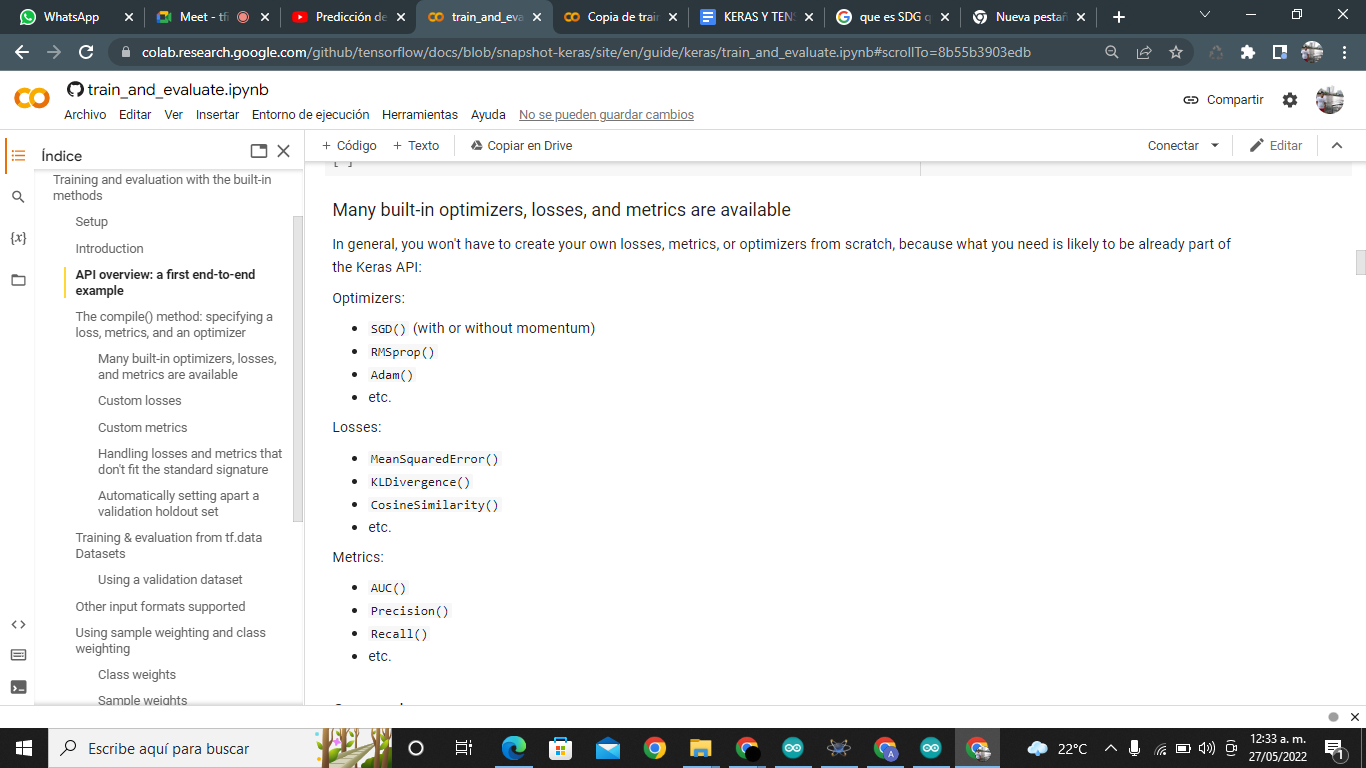
Para el optimizador lo que busca es ajustar los pesos para disminuir el error en la red el cual se utiliza en el ejemplo que es RMSprop que en común los optimizadores que hiperparametros como lo es learnig rete, es en este que puede definir el rango de entrenamiento en el que aprender la red y uno puede definir para que esté en el radio adecuado.

Este tipo de optimizador RMSprop ver figura 4 es el de mantener un promedio móvil (descontado) del cuadrado de los gradientes, también divide el gradiente por la raíz de este promedio este tipo de optimizador, implementa un impulso simple, no un impulso de Nesterov. La versión centrada además mantiene un promedio móvil de los gradientes y usa ese promedio para estimar la varianza y tiene argumentos como (tasa de aprendizaje, rho, impulso, epsilon, centrado, name, \*\*kwargs) cada uno tiene un valor para más detalle ver <https://keras.io/api/optimizers/rmsprop/>

****

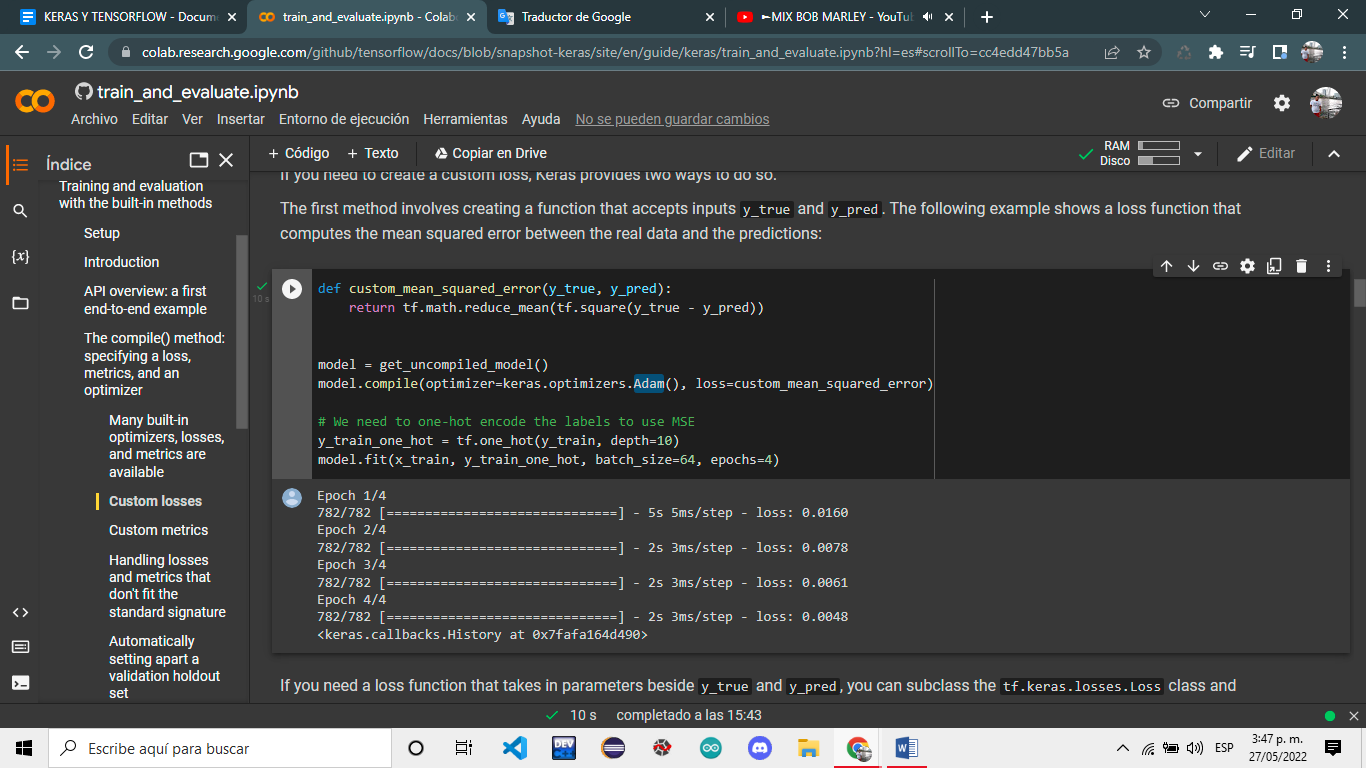
**Figura 4 Arquitectura de la red a la compilación del modelo con las características**

En la siguiente figura 5 es la de los diferentes elementos del modelo que podemos encontrar en el API de keras además en estas explican que aportan al modelo y en qué casos se debe de tener en cuenta el funcionamiento en específico de cada uno de los elementos para esto se da a entender en cada una de las siguiente imagen hay se ve los optimizadores, las funciones de pérdida y la s métricas cada una se puede ver en la API para mayor claridad de los modelos y así ver cuales se ajusta a mi modelo de acuerdo a sus características de entreno y compilación.

****

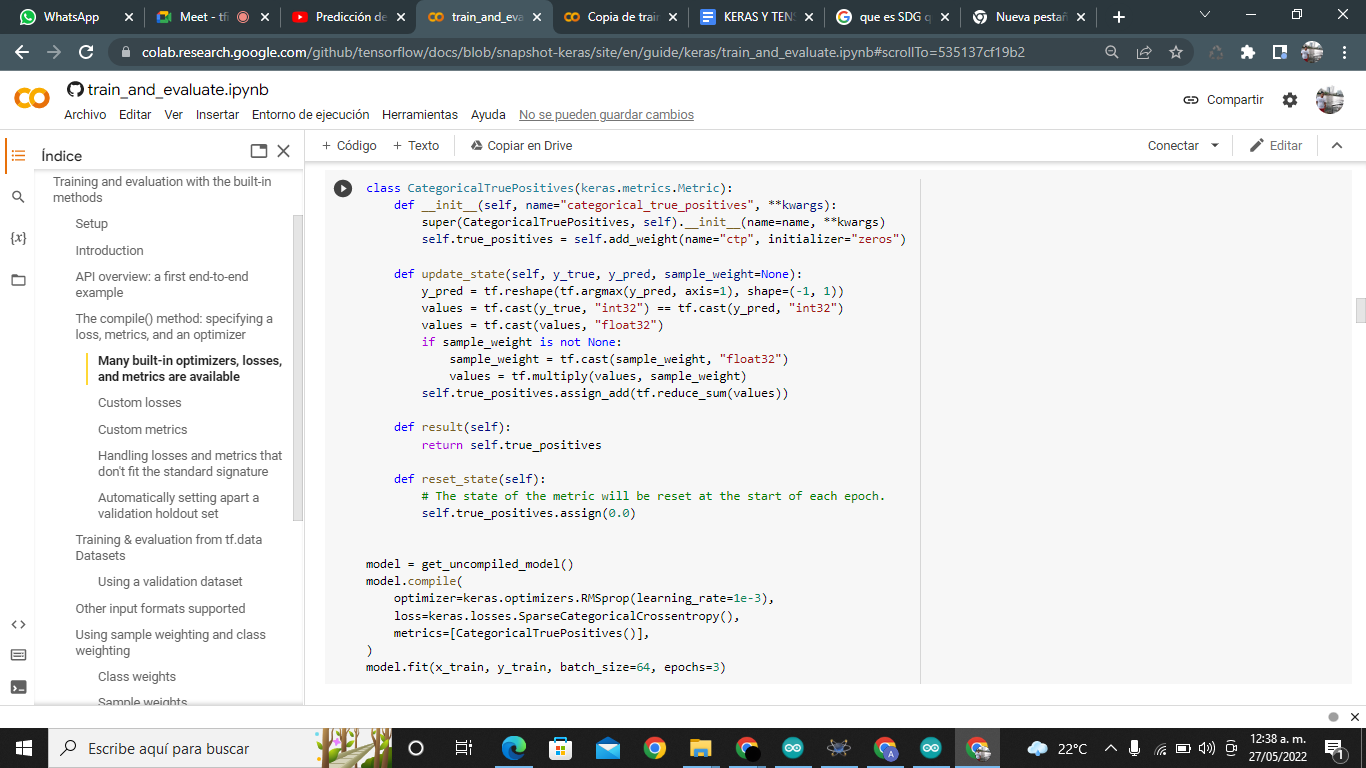
**Figura 5 Funciones de pérdida, métricas u optimizadores que son propias de keras**

En keras encontramos gran variedad en cuanto a funciones de pérdida pero no son las únicas, como vemos en la figura 6 una de perdida personalizada para esta funciones depende los valore reales y\_true y los valores de predicción y\_pred, los modelos se compone de esos dos argumentos en su estructura pero lo que nos refleja es que si son posibles partiendo de los datos reales se iran mejorando la perdida de acuerdo a cada época con respecto a la pérdida y la precisión la gráfica que vimos en los ejemplos mejora con respecto a ello y el número de épocas

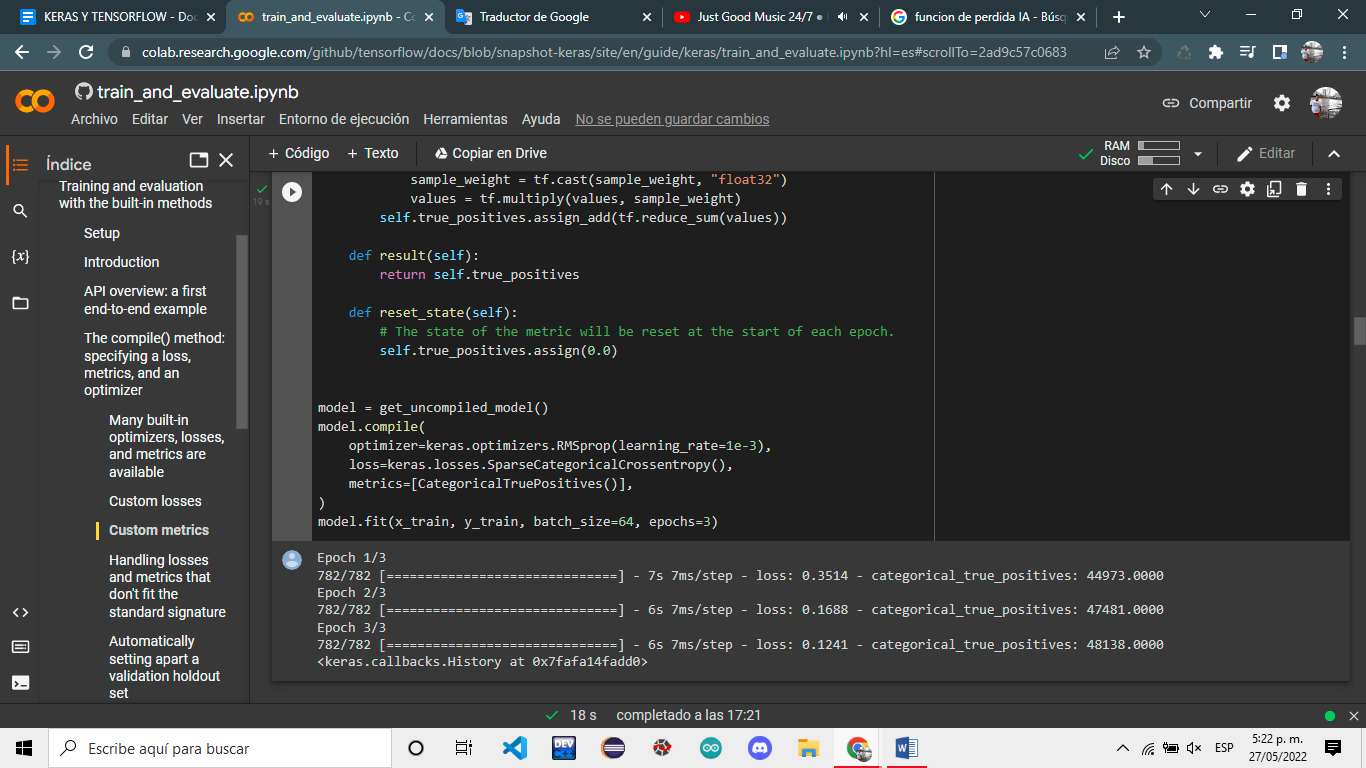
****

**figura 6 Ejemplo de pérdida personalizada**

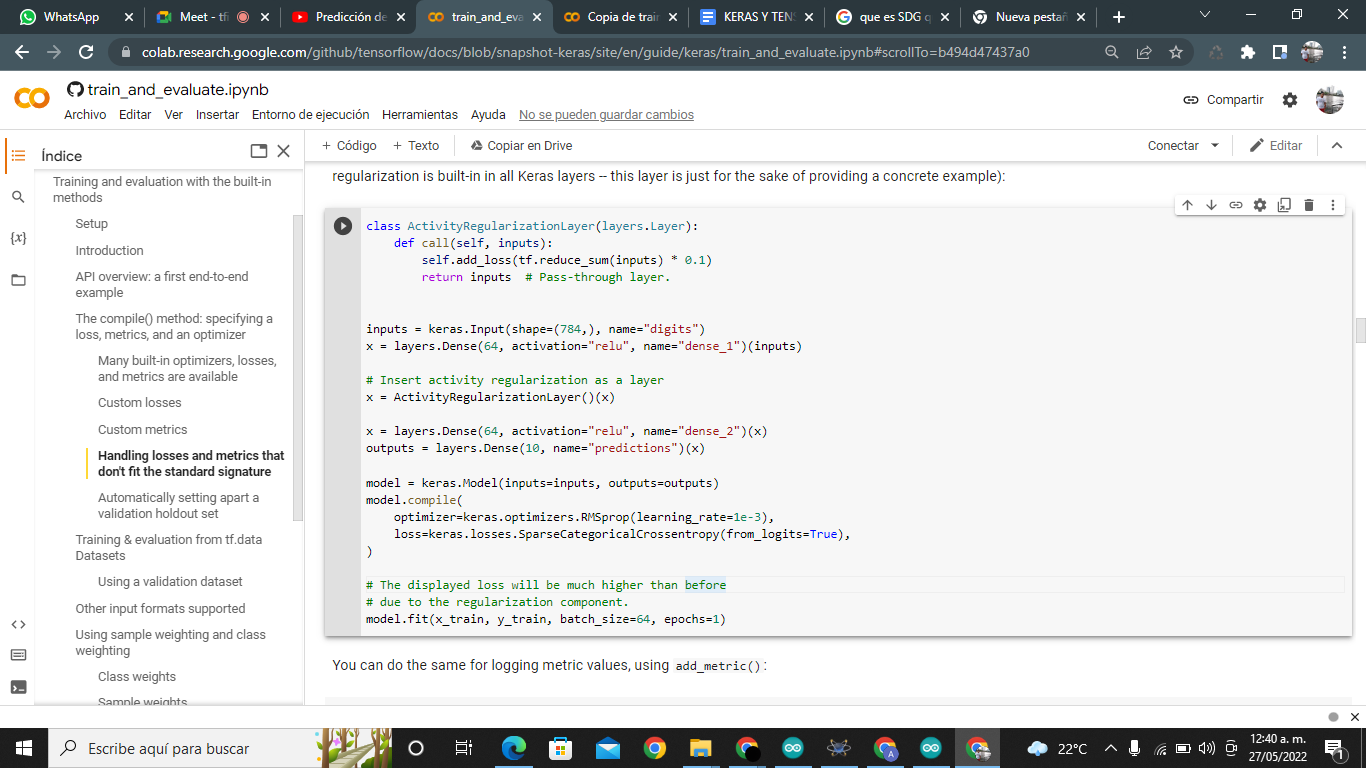
En las pérdidas se pueden colocar argumentos para la red que vienen de keras

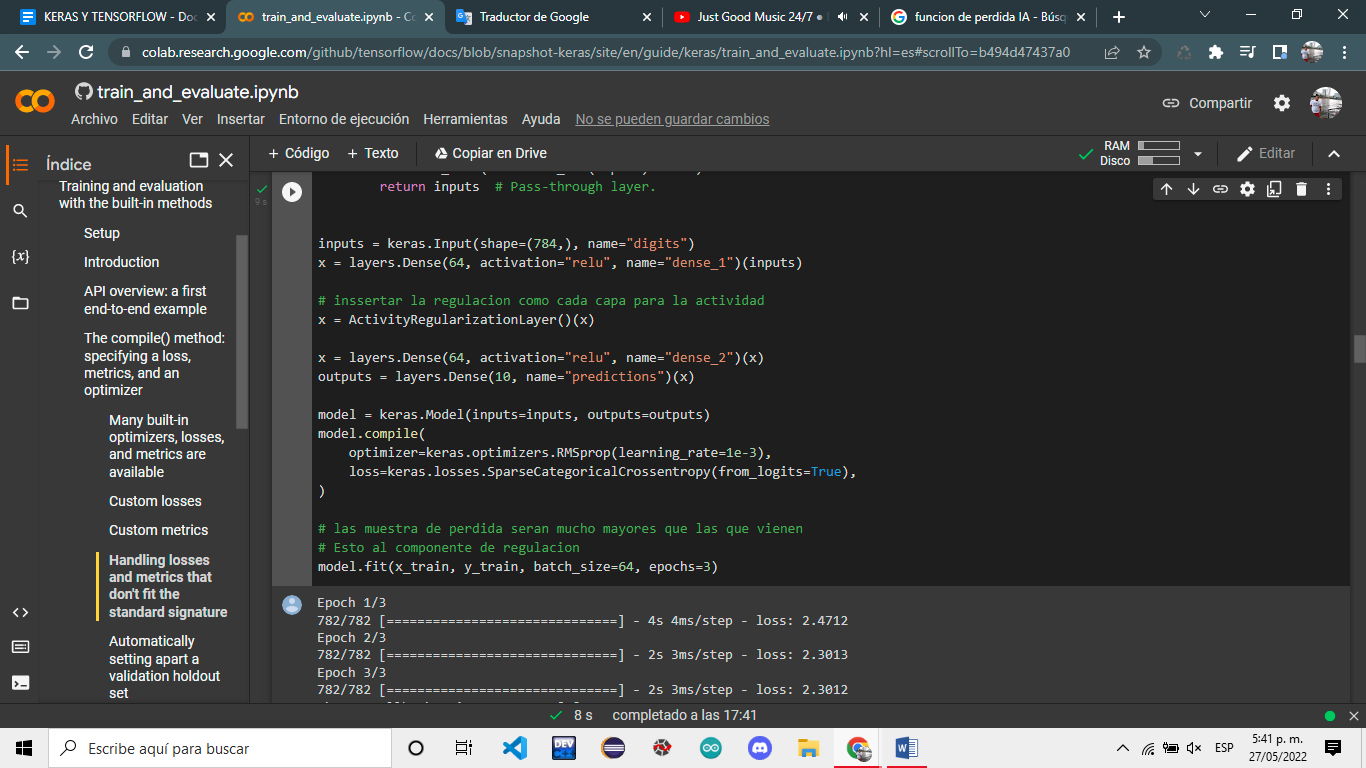
****

El por que de que se adiciona un nuevo valor es porque puede ocurrir que en los datos que tengamos halla un par de datos que se desvían mucho de su predicción, pero solo unos pocos además eso permite que en la precisión nunca mantenga dando cambios atípicos por tanto, este valor extra se usa para reducir la desviación de esos datos con sus predicciones. en las siguientes figuras se podrá apreciar que después de cierta épocas el cambio de esta va a ser ligero a medida que mejora al agregar ese nuevo valor o argumento.



En la siguiente figura si no se implementa el nuevo parámetro lo que ocurre es que se presente de esa manera en cada época lo de la función de pérdida y lo de predicción.

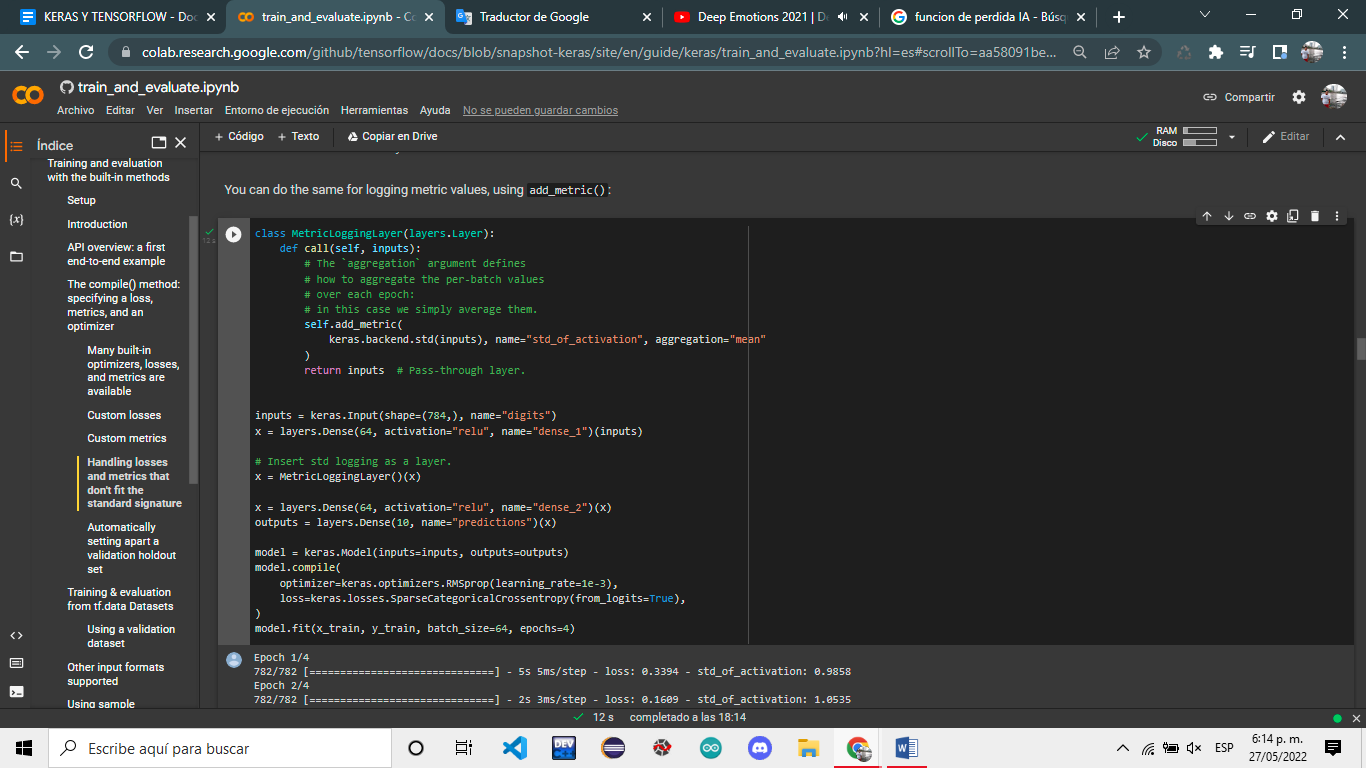
****

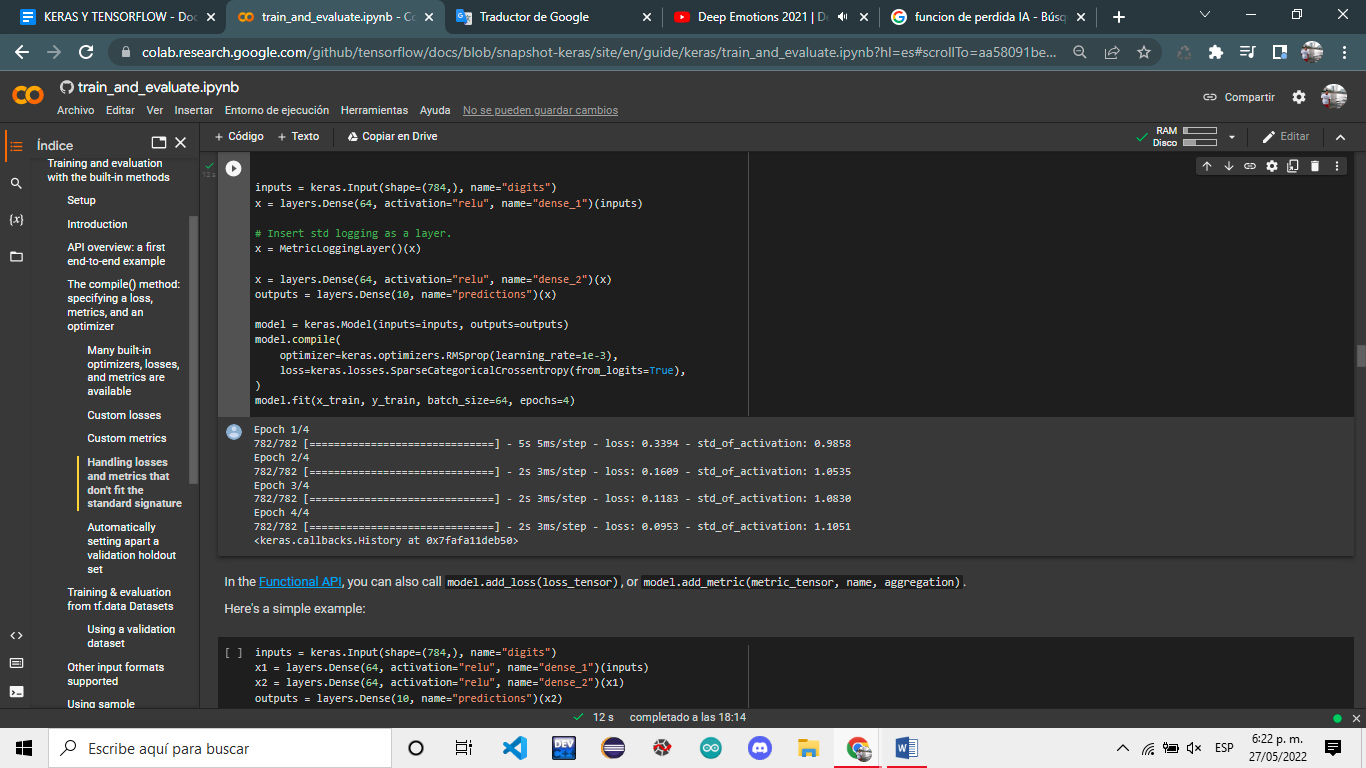
****

se puede ver aquí agregamos una pérdida dentro de otra perdida dentro de la función de pérdida customizada que arriba la mencionamos que se podía también en caso contrario del tradicional del de keras y así va reduciendo pero se ve muy grande todavía errores muy grandes. Esto sucede porque cuando se agrega una pérdida esta se suma a la ponderación que ya viene de las pérdidas y hace aumentando por cada época.

Adicional a lo anterior existen unas métricas que me permiten ver el desempeño de la red por medio de los algoritmos. Además keras ofrece diferentes métricas que me permiten ver el comportamiento de la red adicional a ello hay métricas para el aprendizaje de clasificación y aprendizaje regresivo. También se pueden crear métricas personalizadas que se acomoden a la necesidad del modelo.

Si las métricas son personalizadas, en esta se puede agregar métricas específicas para determinados valores como vemos en la siguiente figura.

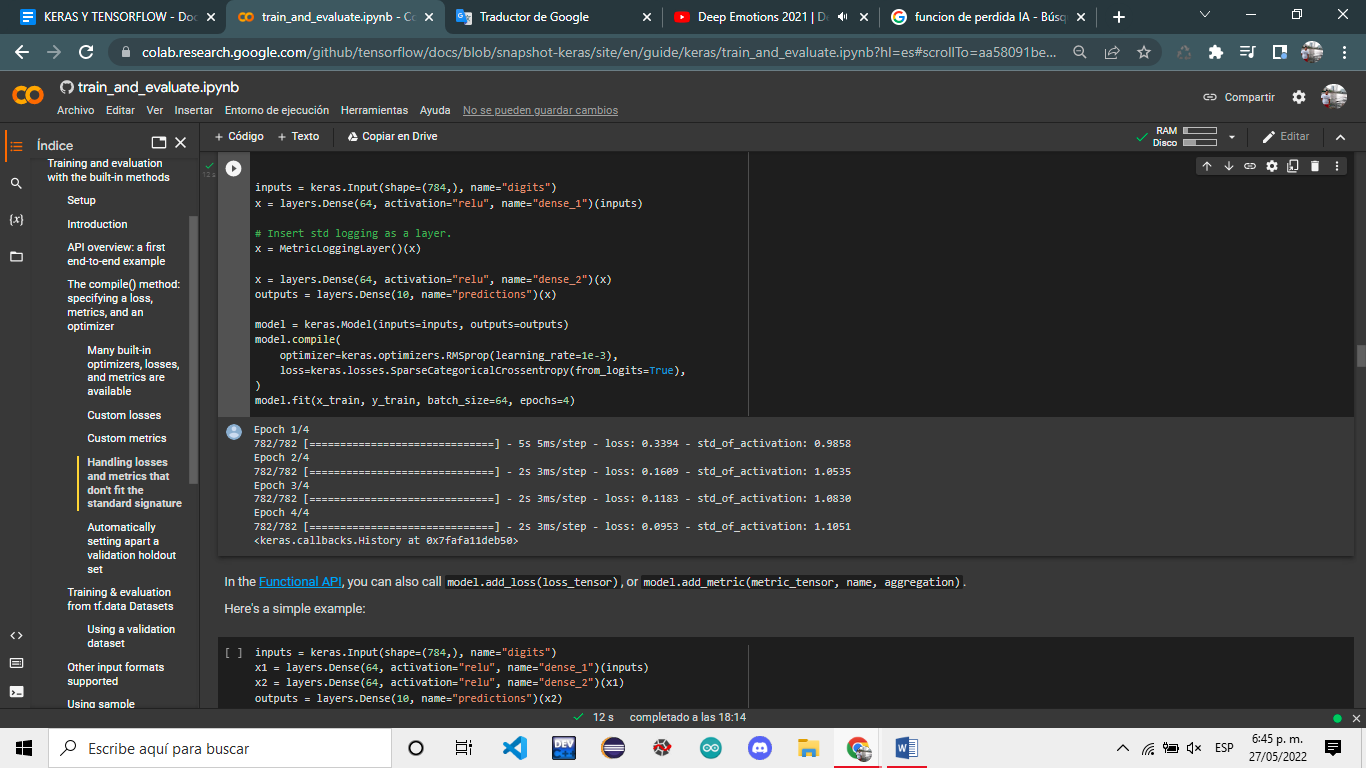


****

Como podemos vemos en el modelo de evaluación en las imágenes y como se ve antes con las pérdidas loss quienes también tiene la característica de agregar en medio de sus funciones, se agrega una nueva métrica y esta se va a promediar con las otras métricas, además en los resultados de las épocas vemos que debido a eso los valores son mucho más bajos a pesar de todavía conservar su característica de ir creciendo con el paso de las épocas. además esto comparte debido a la adición de la ponderación que lleva el modelo.

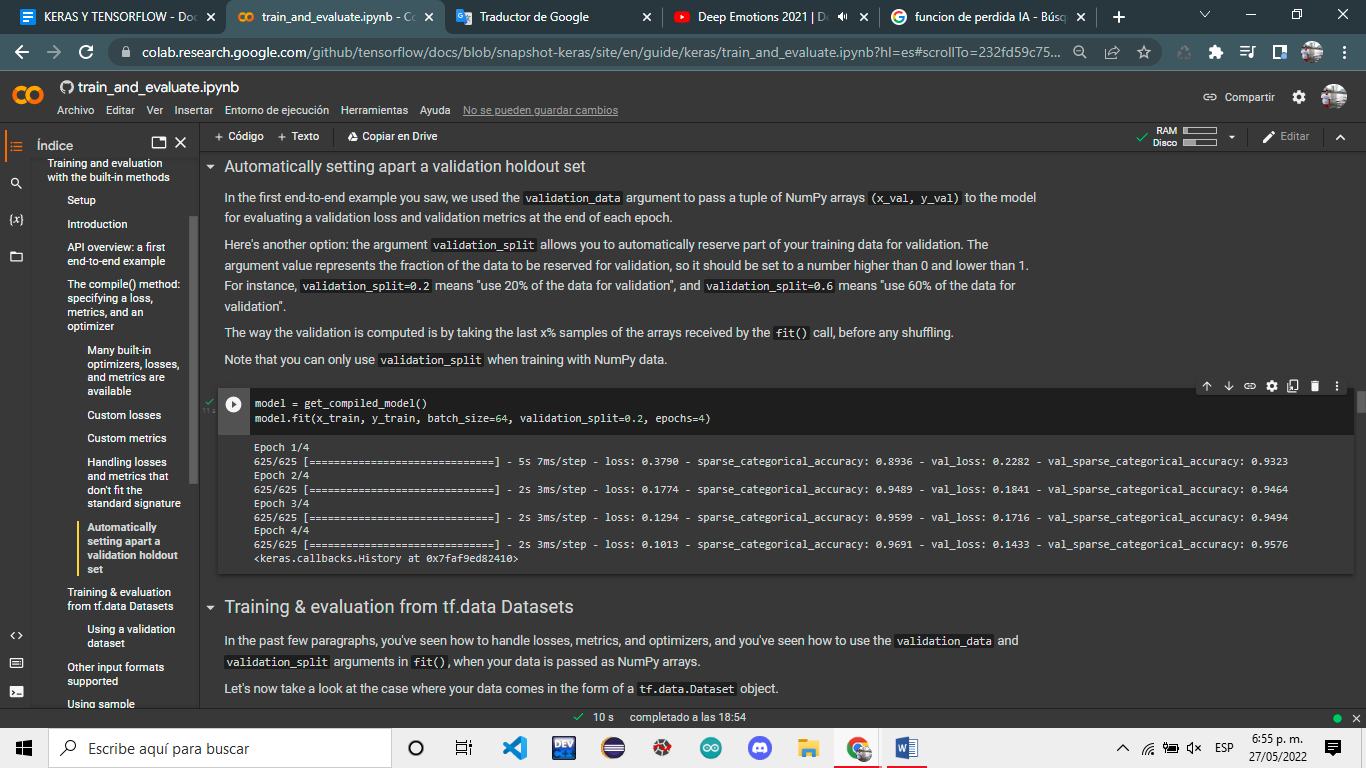
Lo que se lleva hasta ahora en el documento podemos pasar al entrenamiento, en las figuras anteriores se ha visto, este también tiene parámetros algunos que siempre deben ir y otros que se usan ocasiones para mejorar el entrenamiento del modelo.

Por otro lado tenemos los argumentos que siempre aparecen los x\_train y los y\_train que pueden ser valores de entrada y salida , continuando con otro lado los batch\_size que es un número que nos indica las muestras por actualización de gradiente y por otro lado las epochs o épocas que será la cantidad de iteraciones que tendremos para entrenar el modelo . estos son los argumentos básicos



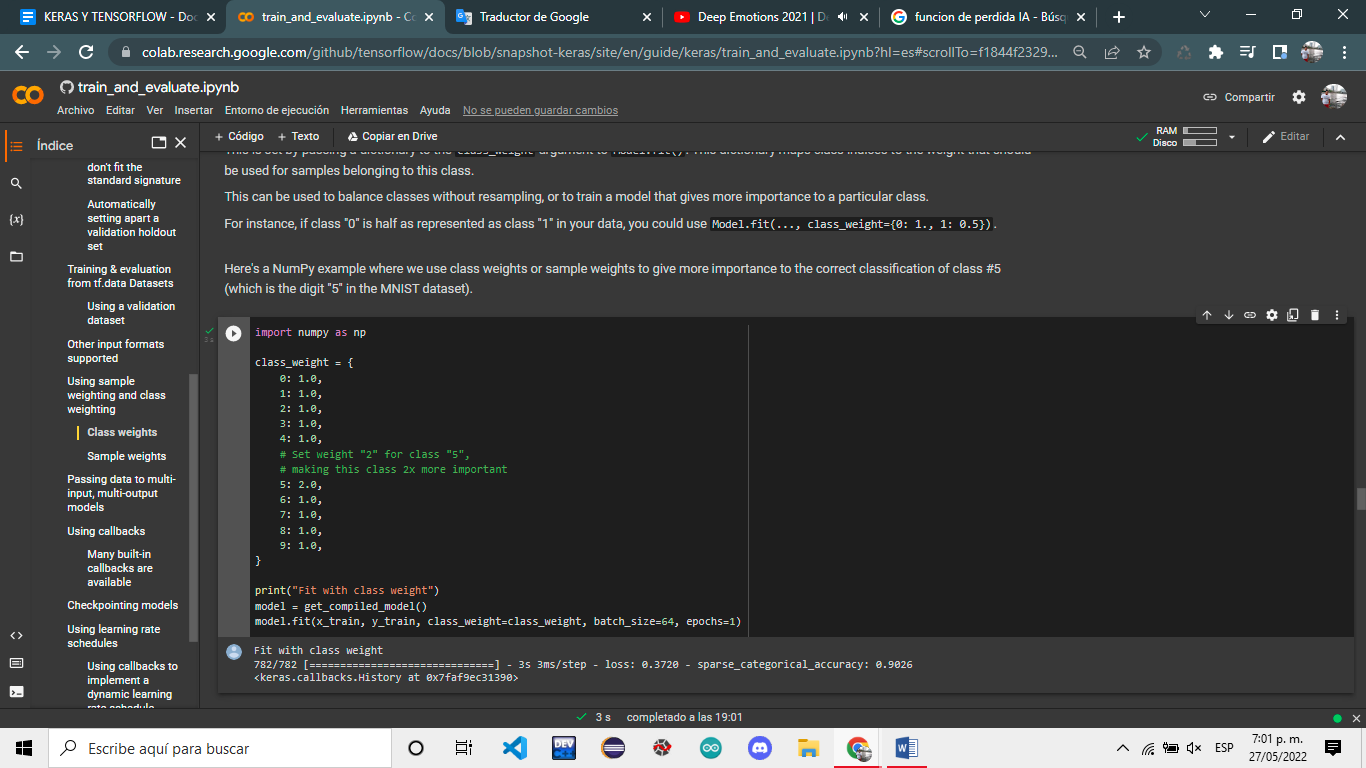
Si observamos podemos ver que con solo estos valores al compilarlo el resultado es una perdida y un métrica y podemos concluir que gracias a valores como el batch \_size o el tamaño y las épocas se puede hacer evidente cómo esta nuestro modelo ya que en un principio en este ejemplo solo se ejecuta en 1 iteración , pero ahora que incrementamos las iteraciones a 4 podemos observar la diferencia entre valores de métricas y valores de pérdidas previas y como se observa está se ve un correcto cambio.

Al analizar otros argumentos para verificar aspectos como validar la pérdida y la métrica como veremos.

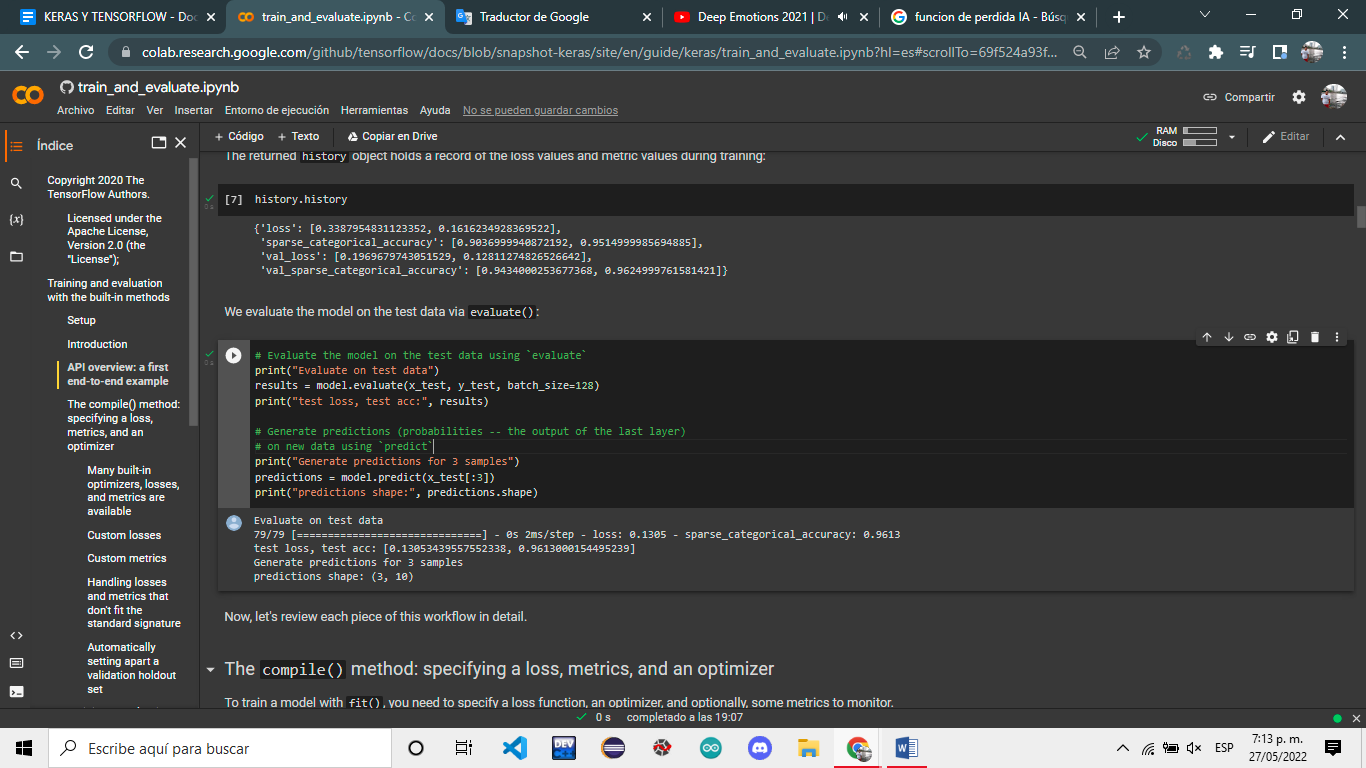


El argumento validation\_split es un argumento que ayuda a evaluar la pérdida y la precisión pero con la cualidad de que no toma todos los datos si no una parte , con esto lo que se busca es tener un mejor control, además de evitar el manejo de datos dispares dentro de los datos.

En el enteramiento es que si el usuario lo prefiere se puede definir un argumento llamada class\_weight , el cual no es más que asignar pesos para los datos pero que se puede utilizar para favorecer a alguna clase de datos en particular que queremos que la red considere como más importante o peso y nosotros ver su comportamiento. ver la siguiente figura



Este modelo de predicción es mas simples a diferencia de los otros modelos pero también muy importante, para hacer uso de esto primero tenemos que verificar que los resultados del modelo de evaluación sean los deseados, continuando lo que se hace es que se toma como el argumento los X\_test lo que hace que influya en el modelo de predicción.



**Ejemplo:**

[**https://colab.research.google.com/github/tensorflow/docs/blob/snapshot-keras/site/en/guide/keras/train\_and\_evaluate.ipynb#scrollTo=dfdc6f08988e**](https://colab.research.google.com/github/tensorflow/docs/blob/snapshot-keras/site/en/guide/keras/train_and_evaluate.ipynb)