

Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

## Actividad 1: Conceptos generales de redes neuronales

En esta actividad vamos a revisar algunos de los conceptos básicos de las redes neuronales, pero no por ello menos importantes.

El dataset a utilizar es Fashion MNIST, un problema sencillo con imágenes pequeñas de ropa, pero más interesante que el dataset de MNIST. Puedes consultar más información sobre el dataset en este enlace.

El código utilizado para contestar tiene que quedar claramente reflejado en el Notebook. Puedes crear nuevas cells si así lo deseas para estructurar tu código y sus salidas. A la hora de entregar el notebook, asegúrate de que los resultados de ejecutar tu código han quedado guardados (por ejemplo, a la hora de entrenar una red neuronal tiene que verse claramente un log de los resultados de cada epoch).

```
[1]: import tensorflow as tf
      print(tf.__version__)
```

2.9.1

En primer lugar vamos a importar el dataset Fashion MNIST (recordad que este es uno de los dataset de entrenamiento que están guardados en keras) que es el que vamos a utilizar en esta actividad:

```
[2]: mnist = tf.keras.datasets.fashion_mnist
```

Llamar a **load\_data** en este dataset nos dará dos conjuntos de dos listas, estos serán los valores de entrenamiento y prueba para los gráficos que contienen las prendas de vestir y sus etiquetas.

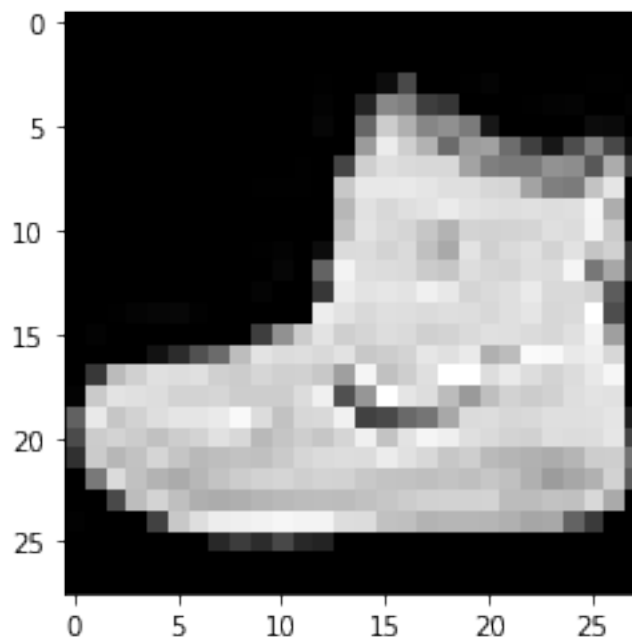
Nota: Aunque en esta actividad lo veis de esta forma, también lo vais a poder encontrar como 4 variables de esta forma: training\_images, training\_labels, test\_images, test\_labels = mnist.load\_data()

```
[3]: (training_images, training_labels), (test_images, test_labels) = mnist.
      load_data()
```

Antes de continuar vamos a dar un vistazo a nuestro dataset, para ello vamos a ver una imagen de entrenamiento y su etiqueta o clase.

```
[4]: import numpy as np
      np.set_printoptions(linewidth=200)
      import matplotlib.pyplot as plt
      plt.imshow(training_images[0], cmap="gray") # recordad que siempre es
      #preferible trabajar en blanco y negro
      #
      print(training_labels[0])
      print(training_images[0])
```

Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022



Habreis notado que todos los valores numericos están entre 0 y 255. Si estamos entrenando una red neuronal, una buena practica es transformar todos los valores entre 0 y 1, un proceso llamado “normalización” y afortunadamente en Python es fácil normalizar una lista. Lo puedes hacer de esta manera:

```
[5]: training_images = training_images / 255.0
test_images = test_images / 255.0
```

Se agrega preprocesamiento sobre las etiquetas, será útil para hacer legible la clasificación hecha con la red neuronal.

```
[6]: # Es necesario preprocesar las etiquetas de salida para que sean
      # reconocidas como categorías
training_labels = tf.keras.utils.to_categorical(training_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)
```

También se procesarán los resultados finales del modelo, así las predicciones del clasificador serán fácilmente interpretables. Para ello se usará una función.

```
[7]: # Convertir las predicciones a sus respectivas etiquetas
def pred_to_label(predictions):
    pred = list()
    for i in range(len(predictions)):
        pred.append(np.argmax(predictions[i]))
    return pred
```

Por ejemplo, en seguida se muestra el primer valor de la variable `test_labels`, luego se pasa por la función antes creada y se muestra de nuevo, así se puede ver la diferencia

Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```
[8]: print("El valor de test_labels[0]:")
print(test_labels[0])
print("El valor de la etiqueta test_labels[0]:")
test_labels_redable = pred_to_label(test_labels)
print(test_labels_redable[0])
```

El valor de test\_labels[0]:  
[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]  
El valor de la etiqueta test\_labels[0]:  
9

Ahora vamos a definir el modelo, pero antes vamos a repasar algunos comandos y conceptos muy útiles: \* **Sequential**: Eso define una SECUENCIA de capas en la red neuronal \* **Dense**: Añade una capa de neuronas \* **Flatten**: ¿Recuerdas que las imágenes cómo eran las imágenes cuando las imprimiste para poder verlas? Un cuadrado, Flatten sólo toma ese cuadrado y lo convierte en un vector de una dimensión.

Cada capa de neuronas necesita una función de activación. Normalmente se usa la función relu en las capas intermedias y softmax en la última capa \* **Relu** significa que "Si  $X > 0$  devuelve  $X$ , si no, devuelve 0", así que lo que hace es pasar sólo valores 0 o mayores a la siguiente capa de la red. \* **Softmax** toma un conjunto de valores, y escoge el más grande.

**Pregunta 1 (3.5 puntos)**. Utilizando Keras, y preparando los datos de  $X$  e  $y$  como fuera necesario, define y entrena una red neuronal que sea capaz de clasificar imágenes de Fashion MNIST con las siguientes características:

- Una hidden layer de tamaños 128, utilizando unidades sigmoid Optimizador Adam.
- Durante el entrenamiento, la red tiene que mostrar resultados de loss y accuracy por cada epoch.
- La red debe entrenar durante 10 epochs y batch size de 64.
- La última capa debe de ser una capa softmax.
- Tu red tendría que ser capaz de superar fácilmente 80 % de accuracy.

```
[9]: ### Tu código para la red neuronal de la pregunta 1 aquí ###
nn = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=[28,28]), # Según
    ↪ training_images.shape
    tf.keras.layers.Dense(128, activation="sigmoid"),
    tf.keras.layers.Dense(10, activation="softmax") # número de
    ↪ clases np.unique(training_labels)
])
nn.compile(loss="categorical_crossentropy", optimizer="adam",
    ↪ metrics=["accuracy"])
nn.fit(training_images, training_labels, batch_size=64, epochs=10)
```

Epoch 1/10  
938/938 [=====] - 3s 3ms/step - loss: 0.5957 - accuracy: 0.8008  
Epoch 2/10  
938/938 [=====] - 3s 3ms/step - loss: 0.4117 - accuracy: 0.8537

Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```
Epoch 3/10
938/938 [=====] - 3s 3ms/step - loss: 0.3740 -
accuracy: 0.8673
Epoch 4/10
938/938 [=====] - 3s 3ms/step - loss: 0.3502 -
accuracy: 0.8743
Epoch 5/10
938/938 [=====] - 5s 5ms/step - loss: 0.3311 -
accuracy: 0.8807
Epoch 6/10
938/938 [=====] - 6s 6ms/step - loss: 0.3164 -
accuracy: 0.8845
Epoch 7/10
938/938 [=====] - 4s 4ms/step - loss: 0.3029 -
accuracy: 0.8903
Epoch 8/10
938/938 [=====] - 6s 6ms/step - loss: 0.2920 -
accuracy: 0.8924
Epoch 9/10
938/938 [=====] - 8s 8ms/step - loss: 0.2821 -
accuracy: 0.8973
Epoch 10/10
938/938 [=====] - 7s 7ms/step - loss: 0.2721 -
accuracy: 0.9005
```

[9]: <keras.callbacks.History at 0x7f4362640af0>

Para concluir el entrenamiento de la red neuronal, una buena practica es evaluar el modelo para ver si la precisión de entrenamiento es real

**pregunta 2 (0.5 puntos):** evalua el modelo con las imagenes y etiquetas test.

[10]: `### Tu código para la evaluación de la red neuronal de la pregunta 2`  
`↪aquí ###`  
`score = nn.evaluate(test_images, test_labels)`

```
313/313 [=====] - 1s 1ms/step - loss: 0.3596 -
accuracy: 0.8706
```

Ahora vamos a explorar el código con una serie de ejercicios para alcanzar un grado de comprensión mayor sobre las redes neuronales y su entrenamiento.

## 1. Ejercicio 1: Funcionamiento de las predicción de la red neuronal

Para este primer ejercicio sigue los siguientes pasos:

- Crea una variable llamada **classifications** para construir un clasificador para las imágenes de prueba, para ello puedes utilizar la función predict sobre el conjunto de test
- Imprime con la función print la primera entrada en las clasificaciones.

Asignatura	Datos del alumno	Fecha
Sistemas cognitivos artificiales	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

**pregunta 3.1 (0.25 puntos)**, el resultado al imprimirlo es un vector de números, \* ¿Por qué crees que ocurre esto, y qué representa este vector de números?

**pregunta 3.2 (0.25 puntos)** \* ¿Cuál es la clase de la primera entrada# de la variable **classifications**? La respuesta puede ser un número o su etiqueta/clase equivalente.

```
[11]: ### Tu código del clasificador de la pregunta 3 aquí ###
classifications = nn.predict(test_images)
print("El primer valor de salida de la red neuronal es:")
print(classifications[0])
classifications_redable = pred_to_label(classifications)
print("El primer valor predicho es:", classifications_redable[0], "y su
valor esperado es:", test_labels_redable[0])
```

```
313/313 [=====] - 1s 1ms/step
El primer valor de salida de la red neuronal es:
[6.0270954e-06 4.8652731e-07 1.7790466e-05 2.1860997e-05 2.8347647e-05
5.0766636e-02 6.2340114e-05 1.2115291e-02 1.6463656e-04 9.3681657e-01]
El primer valor predicho es: 9 y su valor esperado es: 9
```

**Tu respuesta a la pregunta 3.1 aquí:** La razón es que cada elemento es un vector que representa un valor de la función real que la red neuronal está imitando, en una palabra, son probabilidades. En este caso es un vector de diez componentes porque ese es el número posible de categorías.

**Tu respuesta a la pregunta 3.2 aquí:** Si bien varía respecto a la ejecución, es posible ver el valor en el código de la pregunta 3, suele ser 0 o 9 (el correcto y más común).

## 2. Ejercicio 2: Impacto variar el número de neuronas en las capas ocultas

En este ejercicio vamos a experimentar con nuestra red neuronal cambiando el número de neuronas por 512 y por 1024. Para ello, utiliza la red neuronal de la pregunta 1, y su capa oculta cambia las 128 neuronas:

- **pregunta 4.1 (0.25 puntos):** 512 neuronas en la capa oculta
- **pregunta 4.2 (0.25 puntos):** 1024 neuronas en la capa oculta

y entrena la red en ambos casos.

**pregunta 4.3 (0.5 puntos):** ¿Cuál es el impacto que tiene la red neuronal?

```
[12]: ### Tu código para 512 neuronas aquí ###
nn_512 = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=[28,28]), # Según
    training_images.shape
    tf.keras.layers.Dense(512, activation="sigmoid"),
    tf.keras.layers.Dense(10, activation="softmax") # número de clases
    np.unique(training_labels)
])
nn_512.compile(loss="categorical_crossentropy", optimizer="adam",
metrics=["accuracy"])
```

Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```
h512 = nn_512.fit(training_images, training_labels, batch_size=64,
    epochs=10, validation_data=(test_images, test_labels))
```

```
Epoch 1/10
938/938 [=====] - 5s 5ms/step - loss: 0.5284 -
accuracy: 0.8144 - val_loss: 0.4469 - val_accuracy: 0.8379
Epoch 2/10
938/938 [=====] - 5s 6ms/step - loss: 0.4014 -
accuracy: 0.8558 - val_loss: 0.4299 - val_accuracy: 0.8445
Epoch 3/10
938/938 [=====] - 5s 6ms/step - loss: 0.3639 -
accuracy: 0.8674 - val_loss: 0.3885 - val_accuracy: 0.8631
Epoch 4/10
938/938 [=====] - 5s 5ms/step - loss: 0.3355 -
accuracy: 0.8773 - val_loss: 0.3671 - val_accuracy: 0.8693
Epoch 5/10
938/938 [=====] - 5s 5ms/step - loss: 0.3148 -
accuracy: 0.8845 - val_loss: 0.3814 - val_accuracy: 0.8608
Epoch 6/10
938/938 [=====] - 5s 5ms/step - loss: 0.2987 -
accuracy: 0.8895 - val_loss: 0.3476 - val_accuracy: 0.8738
Epoch 7/10
938/938 [=====] - 5s 5ms/step - loss: 0.2829 -
accuracy: 0.8948 - val_loss: 0.3439 - val_accuracy: 0.8788
Epoch 8/10
938/938 [=====] - 5s 5ms/step - loss: 0.2682 -
accuracy: 0.9010 - val_loss: 0.3396 - val_accuracy: 0.8772
Epoch 9/10
938/938 [=====] - 5s 5ms/step - loss: 0.2564 -
accuracy: 0.9054 - val_loss: 0.3372 - val_accuracy: 0.8798
Epoch 10/10
938/938 [=====] - 10s 11ms/step - loss: 0.2447 -
accuracy: 0.9097 - val_loss: 0.3255 - val_accuracy: 0.8837
```

```
[13]: score_512 = nn_512.evaluate(test_images, test_labels)
```

```
313/313 [=====] - 1s 4ms/step - loss: 0.3255 -
accuracy: 0.8837
```

```
[14]: ### Tu código para 1024 neuronas aquí ###
nn_1024 = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=[28,28]), # Según
    training_images.shape
    tf.keras.layers.Dense(1024, activation="sigmoid"),
    tf.keras.layers.Dense(10, activation="softmax") # número de
    clases np.unique(training_labels)
])
```

Asignatura	Datos del alumno	Fecha
Sistemas cognitivos artificiales	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```
nn_1024.compile(loss="categorical_crossentropy", optimizer="adam",  
               metrics=["accuracy"])  
h1024 = nn_1024.fit(training_images, training_labels, batch_size=64,  
                   epochs=10, validation_data=(test_images, test_labels))
```

Epoch 1/10

938/938 [=====] - 12s 12ms/step - loss: 0.5229 - accuracy: 0.8127 - val\_loss: 0.4502 - val\_accuracy: 0.8397

Epoch 2/10

938/938 [=====] - 16s 17ms/step - loss: 0.4040 - accuracy: 0.8537 - val\_loss: 0.4313 - val\_accuracy: 0.8428

Epoch 3/10

938/938 [=====] - 14s 15ms/step - loss: 0.3637 - accuracy: 0.8672 - val\_loss: 0.3874 - val\_accuracy: 0.8602

Epoch 4/10

938/938 [=====] - 14s 15ms/step - loss: 0.3348 - accuracy: 0.8780 - val\_loss: 0.3796 - val\_accuracy: 0.8663

Epoch 5/10

938/938 [=====] - 14s 15ms/step - loss: 0.3122 - accuracy: 0.8850 - val\_loss: 0.3633 - val\_accuracy: 0.8713

Epoch 6/10

938/938 [=====] - 12s 13ms/step - loss: 0.2924 - accuracy: 0.8917 - val\_loss: 0.3765 - val\_accuracy: 0.8604

Epoch 7/10

938/938 [=====] - 13s 14ms/step - loss: 0.2779 - accuracy: 0.8959 - val\_loss: 0.3441 - val\_accuracy: 0.8732

Epoch 8/10

938/938 [=====] - 13s 13ms/step - loss: 0.2615 - accuracy: 0.9024 - val\_loss: 0.3760 - val\_accuracy: 0.8626

Epoch 9/10

938/938 [=====] - 19s 20ms/step - loss: 0.2481 - accuracy: 0.9084 - val\_loss: 0.3315 - val\_accuracy: 0.8831

Epoch 10/10

938/938 [=====] - 14s 15ms/step - loss: 0.2353 - accuracy: 0.9114 - val\_loss: 0.3198 - val\_accuracy: 0.8882

[15]: score\_1024 = nn\_1024.evaluate(test\_images, test\_labels)

313/313 [=====] - 2s 5ms/step - loss: 0.3198 - accuracy: 0.8882

**Tu respuesta a la pregunta 4.3 aquí:** Aumentó el accuracy. Es posible verlo al evaluar las redes resultantes. Sin embargo, esta mejora tiene un límite, en el caso de las 1024 neuronas el cambio es menor. En algunos casos se puede ver también una pequeña disminución de la pérdida, pero ese cambio no siempre ocurre.

Si ahora entrenais el modelo de esta forma (con 512 y 1024 neuronas en la capa oculta) y volveis a ejecutar el predictor guardado en la variable **classifications**, escribir el código del clasificador del ejercicio 1 de nuevo e imprimid el primer objeto guardado en la variable **classifications**.

Asignatura	Datos del alumno	Fecha
Sistemas cognitivos artificiales	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

**pregunta 5.1 (0.25 puntos):**

- ¿En que clase esta clasificado ahora la primera prenda de vestir de la variable classifications?

**pregunta 5.1 (0.25 puntos):**

- ¿Porque crees que ha ocurrido esto?

```
[16]: ### Tu código del clasificador de la pregunta 5 aquí ###
classifications_512 = nn_512.evaluate(test_images, test_labels)
classifications_512 = pred_to_label(classifications_512)
classifications_1024 = nn_1024.evaluate(test_images, test_labels)
classifications_1024 = pred_to_label(classifications_1024)
print("Para la red con 512 neuronas:")
print("El primer valor predicho es", classifications_512[0], "y el valor_
    esperado es", test_labels_redable[0])
print("Para la red con 1024 neuronas:")
print("El primer valor predicho es", classifications_1024[0], "y el_
    valor esperado es", test_labels_redable[0])
```

```
313/313 [=====] - 1s 4ms/step - loss: 0.3255 -
accuracy: 0.8837
```

```
313/313 [=====] - 2s 5ms/step - loss: 0.3198 -
accuracy: 0.8882
```

Para la red con 512 neuronas:

El primer valor predicho es 0 y el valor esperado es 9

Para la red con 1024 neuronas:

El primer valor predicho es 0 y el valor esperado es 9

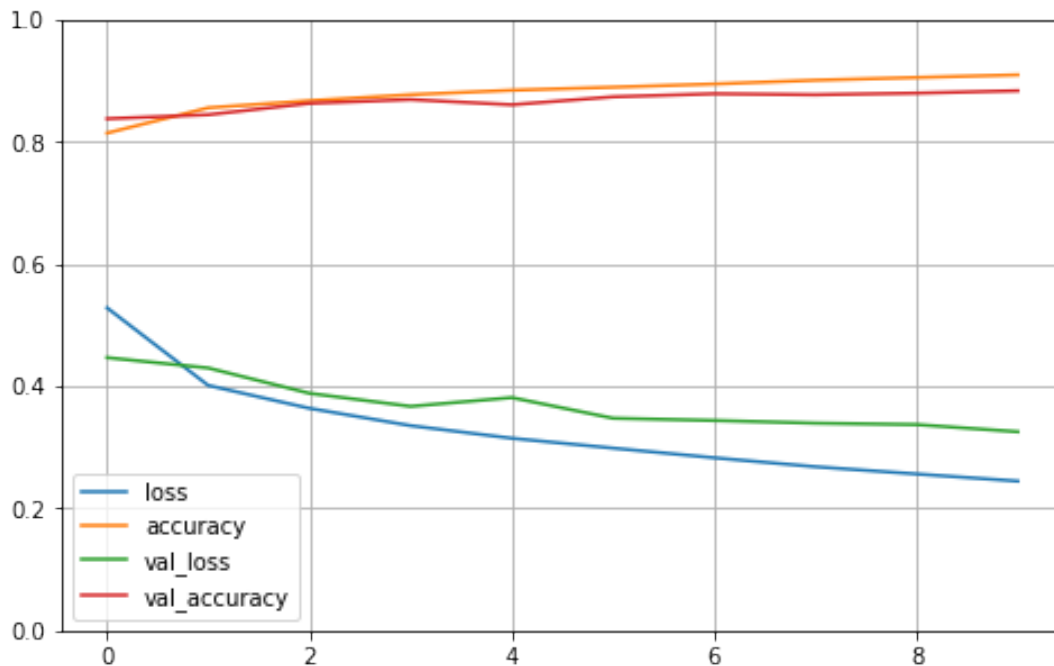
**Tu respuesta a la pregunta 5.1 aquí:** Si bien, el valor sigue variando con cada ejecución del código, lo más común parece ser 0 y no 9. Cabe resaltar que 0 es el valor incorrecto.

**Tu respuesta a la pregunta 5.2 aquí:** Para estas redes neuronales se agregaron datos de validación con el objetivo de contestar esta pregunta. En la gráfica siguiente se puede observar que el valor del accuracy se comporta correctamente, pero el valor de la pérdida no es tan bueno en los datos de validación, existe un pico y el valor se va alejando según avanzan las epoch, es posible atribuir el valor de la red neuronal a este comportamiento.

```
[17]: import pandas as pd
import matplotlib.pyplot as plt
pd.DataFrame(h512.history).plot(figsize=(8,5))
plt.grid(True)
plt.gca().set_ylim(0,1)
plt.show()
```



Asignatura	Datos del alumno	Fecha
Sistemas cognitivos artificiales	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022



### 3. Ejercicio 3: ¿por qué es tan importante la capa Flatten?

En este ejercicio vamos a ver que ocurre cuando quitamos la capa flatten, para ello, escribe la red neuronal de la pregunta 1 y no pongas la capa Flatten.

**pregunta 6 (0.5 puntos):** ¿puedes explicar porque da el error que da?

```
[18]: ### Tu código de la red neuronal sin capa flatten de la pregunta 6
      < aquí ###
nn = tf.keras.models.Sequential([
    tf.keras.layers.Dense(128, activation="sigmoid"),
    tf.keras.layers.Dense(10, activation="softmax") # número de
    < clases np.unique(training_labels)
])
nn.compile(loss="categorical_crossentropy", optimizer="adam",
    < metrics=["accuracy"])
nn.fit(training_images, training_labels, batch_size=64, epochs=10)
```

Epoch 1/10

```
ValueError                                Traceback (most recent call
    < last)
<ipython-input-18-0561f009f7dd> in <cell line: 7>()
    5 ])
    6 nn.compile(loss="categorical_crossentropy", optimizer="adam",
    < metrics=["accuracy"])
```

Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```

----> 7 nn.fit(training_images, training_labels, batch_size=64,
↳ epochs=10)

/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/utils/
↳ traceback_utils.py in error_handler(*args, **kwargs)
    65     except Exception as e: # pylint: disable=broad-exception
    66         filtered_tb = _process_traceback_frames(e.__traceback__)
---> 67         raise e.with_traceback(filtered_tb) from None
    68     finally:
    69         del filtered_tb

/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/engine/
↳ training.py in tf__train_function(iterator)
    13         try:
    14             do_return = True
---> 15             retval_ = ag__.converted_call(ag__.
↳ ld(step_function), (ag__.ld(self), ag__.ld(iterator)), None, fscope)
    16         except:
    17             do_return = False

```

ValueError: in user code:

```

File "/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/training.py", line 1051, in train_function *
    return step_function(self, iterator)
File "/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/training.py", line 1040, in step_function **
    outputs = model.distribute_strategy.run(run_step, args=(data,))
File "/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/training.py", line 1030, in run_step **
    outputs = model.train_step(data)
File "/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/training.py", line 890, in train_step
    loss = self.compute_loss(x, y, y_pred, sample_weight)
File "/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/training.py", line 948, in compute_loss
    return self.compiled_loss(
File "/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/compile_utils.py", line 201, in __call__
    loss_value = loss_obj(y_t, y_p, sample_weight=sw)
File "/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/
↳ losses.py", line 139, in __call__
    losses = call_fn(y_true, y_pred)
File "/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/
↳ losses.py", line 243, in call **
    return ag_fn(y_true, y_pred, **self._fn_kwargs)
File "/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/
↳ losses.py", line 1787, in categorical_crossentropy

```

Asignatura	Datos del alumno	Fecha
Sistemas cognitivos artificiales	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```

        return backend.categorical_crossentropy(
File "/shared-libs/python3.9/py/lib/python3.9/site-packages/keras/
↪backend.py", line 5119, in categorical_crossentropy
        target.shape.assert_is_compatible_with(output.shape)

ValueError: Shapes (None, 10) and (None, 28, 10) are incompatible

```

**Tu respuesta a la pregunta 6 aquí:** Lo que ocurre es que la dimensión de los datos de entrada superan los datos esperados por la red. La capa eliminada sirve para adaptar esas dimensiones, crea datos unidimensionales.

#### 4. Ejercicio 4: Número de neuronas de la capa de salida

Considerad la capa final, la de salida de la red neuronal de la pregunta 1.

**pregunta 7.1 (0.25 puntos):** ¿Por qué son 10 las neuronas de la última capa?

**pregunta 7.2 (0.25 puntos):** ¿Qué pasaría si tuvieras una cantidad diferente a 10?

Por ejemplo, intenta entrenar la red con 5, para ello utiliza la red neuronal de la pregunta 1 y cambia a 5 el número de neuronas en la última capa.

```

[19]: ### Tu código de la red neuronal con 5 neuronas en la capa de salida
      ↪de la pregunta 7 aquí ###
nn = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=[28,28]), # Según
    ↪training_images.shape
    tf.keras.layers.Dense(1024, activation="sigmoid"),
    tf.keras.layers.Dense(5, activation="softmax") # número de clases
    ↪np.unique(training_labels)
])
nn.compile(loss="categorical_crossentropy", optimizer="adam",
    ↪metrics=["accuracy"])
nn.fit(training_images, training_labels, batch_size=64, epochs=10)

```

Epoch 1/10

```

-----
ValueError                                Traceback (most recent call
    ↪last)
<ipython-input-19-a16e008f006d> in <cell line: 8>()
      6 ]
      7 nn.compile(loss="categorical_crossentropy", optimizer="adam",
    ↪metrics=["accuracy"])
----> 8 nn.fit(training_images, training_labels, batch_size=64,
    ↪epochs=10)

```

Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```

/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/utils/
↳ traceback_utils.py in error_handler(*args, **kwargs)
    65     except Exception as e: # pylint: disable=broad-except
    66         filtered_tb = _process_traceback_frames(e.__traceback__)
--> 67         raise e.with_traceback(filtered_tb) from None
    68     finally:
    69         del filtered_tb

```

```

/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/engine/
↳ training.py in tf__train_function(iterator)
    13         try:
    14             do_return = True
--> 15             retval_ = ag__.converted_call(ag__.
↳ ld(step_function), (ag__.ld(self), ag__.ld(iterator)), None, fscope)
    16         except:
    17             do_return = False

```

ValueError: in user code:

```

File "/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/training.py", line 1051, in train_function *
    return step_function(self, iterator)
File "/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/training.py", line 1040, in step_function **
    outputs = model.distribute_strategy.run(run_step, args=(data,))
File "/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/training.py", line 1030, in run_step **
    outputs = model.train_step(data)
File "/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/training.py", line 890, in train_step
    loss = self.compute_loss(x, y, y_pred, sample_weight)
File "/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/training.py", line 948, in compute_loss
    return self.compiled_loss(
File "/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/
↳ engine/compile_utils.py", line 201, in __call__
    loss_value = loss_obj(y_t, y_p, sample_weight=sw)
File "/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/
↳ losses.py", line 139, in __call__
    losses = call_fn(y_true, y_pred)
File "/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/
↳ losses.py", line 243, in call **
    return ag_fn(y_true, y_pred, **self._fn_kwargs)
File "/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/
↳ losses.py", line 1787, in categorical_crossentropy
    return backend.categorical_crossentropy(
File "/shared-lib/python3.9/py/lib/python3.9/site-packages/keras/
↳ backend.py", line 5119, in categorical_crossentropy

```

Asignatura	Datos del alumno	Fecha
Sistemas cognitivos artificiales	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```
target.shape.assert_is_compatible_with(output.shape)
```

```
ValueError: Shapes (None, 10) and (None, 5) are incompatible
```

**Tu respuestas a la pregunta 7.1 aquí:** Es el tamaño del vector de salida porque es el número de categorías existentes.

**Tu respuestas a la pregunta 7.2 aquí:** El vector de salida no será compatible en tamaño y causará un error. Es un problema de dimensionalidad.

## 5. Ejercicio 5: Aumento de epoch y su efecto en la red neuronal

En este ejercicio vamos a ver el impacto de aumentar los epoch en el entrenamiento. Usando la red neuronal de la pregunta 1:

**pregunta 8.1 (0.20 puntos)** \* Intentad 15 epoch para su entrenamiento, probablemente obtendras un modelo con una pérdida mucho mejor que el que tiene 5.

**pregunta 8.2 (0.20 puntos)** \* Intenta ahora con 30 epoch para su entrenamiento, podrás ver que el valor de la pérdida deja de disminuir, y a veces aumenta.

**pregunta 8.3 (0.60 puntos)** \* ¿Porque que piensas que ocurre esto? Explica tu respuesta y da el nombre de este efecto si lo conoces.

```
[20]: ### Tu código para 15 epoch aquí ###
nn_15 = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=[28,28]), # Según
    training_images.shape
    tf.keras.layers.Dense(128, activation="sigmoid"),
    tf.keras.layers.Dense(10, activation="softmax") # número de
    clases np.unique(training_labels)
])
nn_15.compile(loss="categorical_crossentropy", optimizer="adam",
    metrics=["accuracy"])
nn_15.fit(training_images, training_labels, batch_size=64, epochs=15)
```

```
Epoch 1/15
938/938 [=====] - 3s 2ms/step - loss: 0.5948 -
accuracy: 0.8012
Epoch 2/15
938/938 [=====] - 2s 3ms/step - loss: 0.4098 -
accuracy: 0.8534
Epoch 3/15
938/938 [=====] - 2s 3ms/step - loss: 0.3730 -
accuracy: 0.8662
Epoch 4/15
938/938 [=====] - 2s 2ms/step - loss: 0.3497 -
accuracy: 0.8730
Epoch 5/15
```

Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```

938/938 [=====] - 2s 2ms/step - loss: 0.3311 -
accuracy: 0.8800
Epoch 6/15
938/938 [=====] - 3s 3ms/step - loss: 0.3153 -
accuracy: 0.8866
Epoch 7/15
938/938 [=====] - 4s 4ms/step - loss: 0.3029 -
accuracy: 0.8902
Epoch 8/15
938/938 [=====] - 3s 3ms/step - loss: 0.2921 -
accuracy: 0.8929
Epoch 9/15
938/938 [=====] - 4s 4ms/step - loss: 0.2823 -
accuracy: 0.8964
Epoch 10/15
938/938 [=====] - 4s 4ms/step - loss: 0.2725 -
accuracy: 0.8999
Epoch 11/15
938/938 [=====] - 3s 3ms/step - loss: 0.2636 -
accuracy: 0.9032
Epoch 12/15
938/938 [=====] - 3s 3ms/step - loss: 0.2570 -
accuracy: 0.9052
Epoch 13/15
938/938 [=====] - 3s 3ms/step - loss: 0.2480 -
accuracy: 0.9087
Epoch 14/15
938/938 [=====] - 2s 3ms/step - loss: 0.2413 -
accuracy: 0.9115
Epoch 15/15
938/938 [=====] - 2s 3ms/step - loss: 0.2360 -
accuracy: 0.9136

```

```
[21]: score_15 = nn_15.evaluate(test_images, test_labels)
```

```

313/313 [=====] - 1s 1ms/step - loss: 0.3298 -
accuracy: 0.8807

```

```

[22]: ### Tu código para 30 epoch aquí ###
nn_30 = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=[28,28]), # Según
    ↪ training_images.shape
    tf.keras.layers.Dense(128, activation="sigmoid"),
    tf.keras.layers.Dense(10, activation="softmax") # número de
    ↪ clases np.unique(training_labels)
])
nn_30.compile(loss="categorical_crossentropy", optimizer="adam",
    ↪ metrics=["accuracy"])

```

Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```
nn_30.fit(training_images, training_labels, batch_size=64, epochs=30)
```

```
Epoch 1/30
938/938 [=====] - 3s 2ms/step - loss: 0.5838 -
accuracy: 0.8061
Epoch 2/30
938/938 [=====] - 2s 3ms/step - loss: 0.4102 -
accuracy: 0.8536
Epoch 3/30
938/938 [=====] - 2s 2ms/step - loss: 0.3735 -
accuracy: 0.8656
Epoch 4/30
938/938 [=====] - 2s 2ms/step - loss: 0.3486 -
accuracy: 0.8740
Epoch 5/30
938/938 [=====] - 2s 2ms/step - loss: 0.3303 -
accuracy: 0.8805
Epoch 6/30
938/938 [=====] - 2s 3ms/step - loss: 0.3148 -
accuracy: 0.8857
Epoch 7/30
938/938 [=====] - 2s 2ms/step - loss: 0.3015 -
accuracy: 0.8903
Epoch 8/30
938/938 [=====] - 2s 2ms/step - loss: 0.2901 -
accuracy: 0.8951
Epoch 9/30
938/938 [=====] - 2s 2ms/step - loss: 0.2812 -
accuracy: 0.8984
Epoch 10/30
938/938 [=====] - 2s 3ms/step - loss: 0.2720 -
accuracy: 0.9012
Epoch 11/30
938/938 [=====] - 2s 2ms/step - loss: 0.2630 -
accuracy: 0.9039
Epoch 12/30
938/938 [=====] - 2s 2ms/step - loss: 0.2566 -
accuracy: 0.9052
Epoch 13/30
938/938 [=====] - 2s 2ms/step - loss: 0.2479 -
accuracy: 0.9110
Epoch 14/30
938/938 [=====] - 2s 2ms/step - loss: 0.2404 -
accuracy: 0.9115
Epoch 15/30
938/938 [=====] - 2s 3ms/step - loss: 0.2338 -
accuracy: 0.9141
Epoch 16/30
```

Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```

938/938 [=====] - 2s 2ms/step - loss: 0.2282 -
accuracy: 0.9170
Epoch 17/30
938/938 [=====] - 2s 3ms/step - loss: 0.2232 -
accuracy: 0.9190
Epoch 18/30
938/938 [=====] - 2s 2ms/step - loss: 0.2170 -
accuracy: 0.9202
Epoch 19/30
938/938 [=====] - 2s 2ms/step - loss: 0.2111 -
accuracy: 0.9234
Epoch 20/30
938/938 [=====] - 2s 2ms/step - loss: 0.2077 -
accuracy: 0.9248
Epoch 21/30
938/938 [=====] - 2s 2ms/step - loss: 0.2009 -
accuracy: 0.9275
Epoch 22/30
938/938 [=====] - 2s 2ms/step - loss: 0.1974 -
accuracy: 0.9279
Epoch 23/30
938/938 [=====] - 2s 3ms/step - loss: 0.1928 -
accuracy: 0.9305
Epoch 24/30
938/938 [=====] - 2s 3ms/step - loss: 0.1878 -
accuracy: 0.9330
Epoch 25/30
938/938 [=====] - 2s 2ms/step - loss: 0.1836 -
accuracy: 0.9345
Epoch 26/30
938/938 [=====] - 2s 2ms/step - loss: 0.1802 -
accuracy: 0.9352
Epoch 27/30
938/938 [=====] - 2s 2ms/step - loss: 0.1757 -
accuracy: 0.9374
Epoch 28/30
938/938 [=====] - 2s 3ms/step - loss: 0.1718 -
accuracy: 0.9391
Epoch 29/30
938/938 [=====] - 3s 3ms/step - loss: 0.1698 -
accuracy: 0.9391
Epoch 30/30
938/938 [=====] - 2s 2ms/step - loss: 0.1653 -
accuracy: 0.9406

```

```
[23]: score_30 = nn_30.evaluate(test_images, test_labels)
```

```

313/313 [=====] - 1s 1ms/step - loss: 0.3338 -
accuracy: 0.8858

```



Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

**Tu respuesta a la pregunta 8.3 aquí:** En realidad, se puede observar que epoch a epoch el valor de loss continúa bajando. Pese a ello, el valor resultado de la evaluación no se ve disminuido sustancialmente entre las redes de 15 y 30 epoch (a veces puede aumentar). Todo lo anterior indica que el logaritmo de la probabilidad de clasificación correcta disminuye con cada lote de datos que avanza en la red neuronal, pero en el total de los datos procesados este valor se estanca luego de determinado número epoch y luego es posible que aumente, señal de la existencia de overfit, es decir, la red neuronal memoriza pero no aprende.

## 6. Ejercicio 6: Early stop

En el ejercicio anterior, cuando entrenabas con epoch extras, tenías un problema en el que tu pérdida podía cambiar. Puede que te haya llevado un poco de tiempo esperar a que el entrenamiento lo hiciera, y puede que hayas pensado “¿no estaría bien si pudiera parar el entrenamiento cuando alcance un valor deseado?”, es decir, una precisión del 85 % podría ser suficiente para ti, y si alcanzas eso después de 3 epoch, ¿por qué sentarte a esperar a que termine muchas más épocas? Como cualquier otro programa existen formas de parar la ejecución

A partir del ejemplo de código que se da, hacer una nueva función que tenga en cuenta la pérdida (loss) y que pueda parar el código para evitar que ocurra el efecto secundario que vimos en el ejercicio 5.

```
[ ]: ### Ejemplo de código

class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('accuracy') > 0.85):
            print("\nAlcanzado el 85% de precisión, se cancela el
            entrenamiento!!")
            self.model.stop_training = True
```

**Pregunta 9 (2 puntos):** Completa el siguiente código con una clase callback que una vez alcanzado el 40 % de pérdida detenga el entrenamiento.

```
[24]: import tensorflow as tf
print(tf.__version__)

### Tu código de la función callback para parar el entrenamiento de
la red neuronal al 40% de loss aquí: ###
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('loss') > 0.4):
            print("\nPérdida del 40%. Fin del entrenamiento.")
            self.model.stop_training = True

callbacks = myCallback()
mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (test_images, test_labels) =
mnist.load_data()
```

Asignatura	Datos del alumno	Fecha
<b>Sistemas cognitivos artificiales</b>	Apellidos: Domínguez Espinoza Nombre: Edgar Uriel	27 de julio de 2022

```

training_images = training_images/255.0
test_images = test_images/255.0

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(),
                                     tf.keras.layers.Dense(512,
                                     ↪activation=tf.nn.relu),
                                     tf.keras.layers.Dense(10,
                                     ↪activation=tf.nn.softmax)])

model.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(training_images, training_labels, epochs=50,
          ↪callbacks=[callbacks])

```

2.9.1

Epoch 1/50

1868/1875 [=====>.] - ETA: 0s - loss: 0.4766 -  
↪accuracy:

0.8288

Pérdida del 40%. Fin del entrenamiento.

1875/1875 [=====] - 11s 6ms/step - loss: 0.4765  
↪-

accuracy: 0.8288

## 7. Conclusión

En esta práctica se ha creado una red neuronal sencilla de una sola capa oculta. Se han ido cambiando los hiperparámetros de la red, con el objetivo de ver la repercusión de estos cambios en el aprendizaje de la red.

Es importante decir que la práctica se comenzó a escribir en la plataforma de [My Binder](#), pero se tuvieron problemas debido al limitado hardware que pone a disposición del usuario. Para hacer la práctica fue necesario que python detectara una tarjeta gráfica, proceso que el sistema realiza de manera automática. Lo anterior es relevante porque pese a la sencillez de la red implementada existió el problema, es posible suponer entonces que redes más complejas tendrán requisitos realmente elevados a nivel hardware, de ahí el uso de servicios en línea más profesionales.

En tanto a los resultados, lo más notable es que aunque el resultado del notebook puede considerarse constante, pueden observarse ocasiones en donde hay una variación visible en los resultados, principalmente en las predicciones hechas por la red neuronal, dichas variaciones son comentadas en las respuestas presentadas en el documento.