**IABD – PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL**

*PRÁCTICA EVALUABLE – 5.3 – Mnist: Zalando: Un modelo de datos exponencialmente más grandeº*

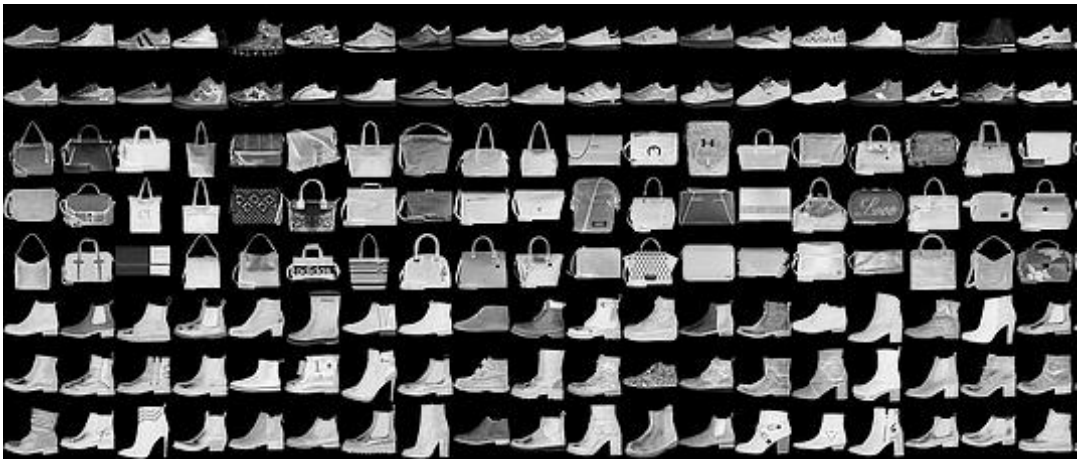
Una vez que hemos experimentado con keras y con problemas de regresión, clasificación binaria y clasificación multiclase, vamos a trabajar con un conjunto de datos más complejo.

Los expertos en data science de la empresa Zalando Research, Han Xiao y Kashif Rasul, han creado un dataset de imágenes de sus productos para experimentar con algoritmos de aprendizaje automático:

Este dataset fue bautizada como Fashion MNIST y se han subido este dataset a multitud de fuentes, como Github o Kaggle. Incluso está disponible dentro de los datasets (keras.datasets.fashion_mnist).

¿Os acordáis de los vídeos de la serie Animated Maths de 3blue1brown? En estos vídeos se explicaba cómo clasificar dígitos manuscritos que la MNIST ("Modified National Institute of Standards and Technology") había recopilado de cientos de estudiantes.

La idea detrás Fashion-MNIST surgió como una necesidad de proporcionar a la comunidad de aprendizaje automático un conjunto de datos más desafiante que el MNIST original, con el fin de impulsar el desarrollo de modelos de aprendizaje automático más sofisticados. Los artículos de moda presentan una complejidad mayor en términos de variabilidad de patrones, formas y texturas, lo que ofrece un desafío más significativo para los algoritmos de clasificación. En lugar de dígitos, contiene imágenes de artículos de moda de Zalando, una empresa de moda en línea. Este conjunto de datos incluye 70,000 imágenes de 10 categorías diferentes de ropa y accesorios, como camisetas/tops, pantalones, vestidos, abrigos, sandalias, camisas, zapatillas, bolsos, y botas al tobillo. Al igual que MNIST, las imágenes de Fashion-MNIST también son en blanco y negro y de 28x28 píxeles:



En esta práctica entrenaremos una red neuronal para que reconozca prendas de vestir a partir de un conjunto de datos común llamado Fashion MNIST. Puede obtener más información sobre este conjunto de datos aquí.

Contiene 70.000 prendas de vestir en 10 categorías diferentes. Cada prenda de vestir está en una imagen en escala de grises de 28x28.

Resuelve los siguientes ejercicios. Entrega todo en un fichero llamado prc5.3_ fullname.py

**IABD – PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL**

*PRÁCTICA EVALUABLE – 5.3 – Mnist: Zalando: Un modelo de datos exponencialmente más grandeº*

1. Los datos de Fashion MNIST están disponibles directamente en la API de conjuntos de datos de tf.keras. Los puedes cargar así:

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

   ¿Qué representan cada una de las variables que devuelve mnist.load_data()? ¿Cuántos elementos y qué forma tiene cada uno de ellos?

   - training_images: contiene 60.000 imágenes de entrenamiento, en blanco y negro con una resolución de 28x28. Su forma es (60000, 28, 28).
   - training_labels: son las etiquetas a las imágenes de entrenamiento. Cada una va del 0 al 9. Su forma es (60000,).
   - test_images: tiene 10.000 imágenes para evaluar el modelo. Tienen la misma forma que las de entrenamiento: (10000, 28, 28).
   - test_labels: son las etiquetas a las imágenes de prueba. Su forma es (10000,).

2. Visualiza el contenido de training_images[0] con un print. ¿Cuál es su forma? ¿Qué tipo de dato contiene? Puedes visualizar una imagen en matplotlib.pyplot con el método imshow, que recibe como parámetro la imagen que quieres mostrar.
   - Contenido: En el código adjunto, línea 30
   - Forma: (28, 28)
   - Tipo de dato: uint8
   - Visualización en el código, líneas 37-39

3. ¿Cuántas clases diferentes de ropa hay? Visualiza en una figura de matplotlib, 10 elementos de cada clase.
   - Las clases son: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag y Ankle boot
   - Fuente: https://keras.io/api/datasets/fashion_mnist/

4. De las etiquetas verdaderas, crea la siguiente lista de clases:

```
class_names = ['Camiseta', 'Pantalón', 'Jersey', 'Vestido', 'Abrigo', 'Sandalia', 'Camisa', 'Zapatilla', 'Bolso', 'Bota']
x,idx = tf.unique(training_labels)
for i in x:
  print(class_names[i])
```

   - Línea 83 en el código

5. Cuando trabajamos con imágenes, para pasarlas a una RNA, es mejor escalar los valores de sus píxeles. Normalizar es simplemente convertir los valores de los píxeles a un rango 0-1 en lugar de un rango 0-255 (valores posibles de un byte). Por tanto, tan sólo tienes que dividir todos los pixeles entre 255. Puedes investigar sobre lo que es el MinMaxScaler. Escala los valores de los píxeles de todas las imágenes.
   - Línea 97 en el código

6. Crea un modelo de keras para que aprenda las diferencias entre las clases de ropa. ¿Qué forma tendrá la capa de entrada? Para que la RNA procese mejor la primera capa, hay que aplicar una capa llamada Flatten, que "aplana la matriz que representa todos los píxeles de la imagen. Por tanto, la primera capa de tu modelo será:

```
#define la primera capa, junto con la forma de la entrada
tf.keras.layers.Flatten(input_shape=(28,28))
```

   Introduce también en el modelo una capa oculta de 128 neuronas con la función de activación 'relu'. ¿Cómo debe ser la capa de salida? ¿Qué función de activación debe tener?

- La capa de salida debe tener 10 neuronas, porque hay 10 clases de ropa. Cada neurona representa una posible prenda.
- La función de activación debe ser softmax, para que el modelo dé una probabilidad para cada clase, para que elija la clase con más probabilidad.
- Línea 107 en el código.

7. Compila el modelo utilizando el método compile con el optimizador Adam, función de pérdida sparse_categorical_crossentropy y utilizando como métrica "Accuracy".
   - Línea 117 en el código

8. Entrena el conjunto training_images durante 5 épocas. ¿Qué resultados obtienes? ¿Te parecen buenos?
   - Se obtuvo un 88% de precisión. El modelo es preciso con pocas épocas. La pérdida fue disminuyendo con cada época.
   - Línea 126

9. Evalúa el modelo con el conjunto de test y comenta los resultados que has obtenidos
   - La precisión es del 86% y la pérdida de 0.35. El modelo ha sido capaz de generalizar correctamente a datos que nunca vio. No hay signos de sobreajuste.
   - Línea 131

10. Ejecuta el siguiente código:

```
classifications = model.predict(test_images)
print(classifications[0])
```

    La salida, después de ejecutarla, es una lista de números. ¿Por qué crees que es así y qué representan esos números? ¿Qué representa esta lista? Elige la respuesta de entre las siguientes opciones
    a) Los 10 valores aleatorios menos significativos
    b) Las 10 primeras clasificaciones que hizo la RNA
    c) La probabilidad de que el elemento sea de cada una de las 10 clases
       - La probabilidad de que el elemento sea de cada una de las 10 clases.
       - La salida es una lista de 10 números porque cada número representa la probabilidad de que la imagen de entrada pertenezca a esa clase.
       - Línea 145

11. ¿Cómo sabes que esta lista te dice que tu elemento es una bota? Elige la respuesta de entre las siguientes opciones:
    a) No hay suficiente información para responder esa pregunta
    b) El décimo elemento de la lista es el más grande y la bota tiene la etiqueta 9
    c) La bota es la etiqueta 9 y hay 0->9 elementos en la lista
       - El décimo elemento de la lista es el más grande y la bota tiene la etiqueta 9

12. Prueba a cargar una imagen que hayas diseñado tú, escribe el código necesario para cargar la imagen y predecir su clase. Puedes utilizar la función imread del paquete cv2 (Computer Vision 2):

```
import cv2
mi_imagen=cv2.imread('miImagen.png',cv2.IMREAD_GRAYSCALE)
plt.imshow(mi_imagen)
```

13. Veamos ahora las capas en tu modelo. Experimenta con diferentes valores para la capa densa con 512 neuronas. ¿Qué resultados diferentes obtienes para la pérdida, el tiempo de entrenamiento, etc.? ¿Por qué crees que es así?

**IABD – PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL**

*PRÁCTICA EVALUABLE – 5.3 – Mnist: Zalando: Un modelo de datos exponencialmente más grandeº*

Incrementa la capa oculta a 1024 neuronas -- ¿Cuál es el impacto?

a) El entrenamiento lleva más tiempo, pero es más preciso
b) El entrenamiento lleva más tiempo, pero no afecta la precisión
c) El entrenamiento lleva el mismo tiempo, pero es más preciso

- El entrenamiento lleva más tiempo, pero es más preciso
- Con 512 neuronas aumenta la precisión en un 89%.
- Con 1024 neuronas también tiene un 89%, pero tarda un poco mas que con 512 neuronas.
- Si duplicamos el número de neuronas, el modelo aprende patrones más complejos. También reduce la pérdida.
- Línea 160

14. ¿Qué pasaría si eliminas la capa Flatten()? ¿Por qué crees que es así?
    - Saltará un error al entrenar el modelo, no entiende el formato de los datos porque hay que aplanar la imagen. Espera un vector y no una matriz 2D:
      File "c:\Users\Eurobeater\Documents\Curso IA y Big Data Git\Programacion de Inteligencia Artificial\2EV\Tema 5\Python\Ejercicios\Practica 5.3\prc5.3_ antonio_gallego_peñalver.py", line 127, in <module>
        model.fit(training_images, training_labels, epochs=5)
      File "C:\Users\Eurobeater\anaconda3\Lib\site-packages\keras\src\utils\traceback_utils.py", line 122, in error_handler
        raise e.with_traceback(filtered_tb) from None
      File "C:\Users\Eurobeater\anaconda3\Lib\site-packages\keras\src\backend\tensorflow\nn.py", line 725, in sparse_categorical_crossentropy
        raise ValueError(
      ValueError: Argument `output` must have rank (ndim) `target.ndim - 1`. Received: target.shape=(32,), output.shape=(32, 28, 10)

      El código esta comentado para que no salte ningún error, se pueden quitar los comentarios de la línea en cuestión para comprobar

15. Considera las capas finales (de salida). ¿Por qué hay 10? ¿Qué pasaría si tuvieras una cantidad diferente a 10? Por ejemplo, intenta entrenar la red con 5
    - Hay 10 capas de salida porque hay 10 clases posibles. Si no ponemos 10, el modelo no clasificará correctamente.
    - Cada una representa la probabilidad de pertenecer a una clase.
    - El error en cuestión:
      Received a label value of 9 which is outside the valid range of [0, 5).  Label values: 2 6 3 7 2 0 4 4 3 3 8 9 8 2 4 8 1 2 9 0 6 4 5 5 0 0 6 0 5 5 9 4
      [[{{node compile_loss/sparse_categorical_crossentropy/SparseSoftmaxCrossEntropyWithLogits/SparseSoftmaxCrossEntropyWithLogits}}]] [Op:__inference_multi_step_on_iterator_137584]

      El código esta comentado para que no salte ningún error, se pueden quitar los comentarios de la línea en cuestión para comprobar

**IABD – PROGRAMACIÓN DE INTELIGENCIA ARTIFICIAL**

*PRÁCTICA EVALUABLE – 5.3 – Mnist: Zalando: Un modelo de datos exponencialmente más grandeº*

16. Considera los efectos de capas adicionales en la red. ¿Qué pasará si agregas otra capa entre la que tiene 512 y la capa final con 10?
    - El entrenamiento tarda un poco por aumentar los parámetros. Aumenta la precisión ligeramente, aprendiendo patrones más complejos.
    - Alcanzó aproximadamente un 89,5% de precisión.
    - Línea 223

17. ¿Cuál es el impacto de poner más o menos épocas en el entrenamiento? ¿Por qué crees que sería así?
    - Si se aumentan las épocas, el modelo tarda más en entrenar, haciendo que aprenda mejor.
    - Si entrenamos con muchas épocas, hay riesgo de que se produzca sobreajuste. Si entrenamos con muchas épocas, el modelo puede sufrir underfitting.

18. Prueba con 15 épocas: probablemente obtendrás un modelo con una pérdida mucho mayor que el que tiene 5.
    - Con 15 épocas la precisión del modelo ha aumentado y la pérdida ha disminuido en comparación con 5 épocas.
    - Línea 239.

19. Prueba 30 épocas: es posible que vea que el valor de pérdida deje de disminuir y, a veces, aumente. Este es un efecto secundario de algo llamado "sobreajuste" y es algo que debes tener en cuenta al entrenar redes neuronales. No tiene sentido perder el tiempo entrenando si no estás mejorando la pérdida. Es como si la RNA se lo hubiera aprendido de memoria.
    - No hay sobreajuste. El loss siguió disminuyendo y la precisión mejoró.
    - Línea 238

20. Antes de entrenar, se normalizaron los datos, pasando de valores que eran 0-255 a valores que eran 0-1. ¿Cuál sería el impacto de eliminar ese proceso? Aquí está el código completo para probarlo. ¿Por qué crees que obtienes resultados diferentes?
    - Sin normalizar, el modelo rinde peor
    - Línea 274

21. Anteriormente, cuando entrenaste con épocas adicionales, probablemente te fijaras en que a veces, el valor de loss deja de disminuir o el valor de accuracy deja de aumentar. Aun así, el entrenamiento no finalizó hasta pasado un buen rato. Es posible que hayas pensado "¿no sería genial que pudiera detener el entrenamiento cuando se alcance un valor de accuracy deseado?" Pregunta al chat gpt cómo puedes hacerlo utilizando una función de callback y añade a tu modelo que, una vez alcanzado el 91% de efectividad, deje de entrenar.
    - Para detener el entrenamiento, usé un callback de Keras. Si la precisión baja del 91%, se detiene el entrenamiento con self.model.stop_training = True.
    - Línea 301.