Proyecto Fashion-MNIST
Informes de Avances.

El presente documento abarca los avances solicitados, durante el desarrollo del *Curso: Deep Learning,* asociados a la elaboración y la implementación progresiva de los conocimientos adquiridos en el Curso a través de la ejecución del proyecto Fashion-MNIST.

Dicho lo anterior, el desglose del informe engloba tres apartados; ***Avance 1*** incluye los objetivos del proyecto, una introducción y una breve descripción de los datos. Mientras que el ***Avance 2***, consiste en una descripción más detallada de los datos a utilizar (cantidad de datos, de dónde vienen, sus distintos atributos y etiquetas, etc.), con el objetivo de tener un mejor entendimiento de ellos. El cual facilitará la elaboración del ***Avance 3***, correspondiente a la propuesta del modelo con su respectiva arquitectura asociada y el análisis y/o evaluación del modelo propuesto.

AVANCE 1

03 Julio 2020

En el siguiente apartado se procede a presentar los objetivos del proyecto, su alcance y una breve contextualización, la cual será conducente a la descripción del conjunto de datos.

Objetivos

Construir un modelo de clasificación que mejor funcione para el data set entregado. Para tales efectos, se propone implementar distintos tipos de clasificadores y en función al porcentaje de error que arroje el modelo, será el seleccionado.

El presente proyecto cuenta con 70 mil imágenes (28x28 píxeles) de artículos de vestuario, agrupado en 2 conjuntos; de entrenamiento y de prueba, con 60 mil y 10 mil imágenes respectivamente.

Alcance

El alcance del proyecto es extrapolar el modelo de clasificación seleccionado, a la organización y/o clasificación de las prendas por temporada (invierno, verano, etc.) en bodegas de almacenamiento del rubro.

Contextualización Data

Zalando es una tienda de moda en línea alemana. Su modelo de negocio es la venta online al por menor, de zapatos y ropa para mujer, hombre y niño. Fue creada en 2008, cuenta con sede en Berlín y está presente en varios países europeos.

Fashion-MNIST es un conjunto de datos de imágenes de artículos de Zalando, el cual cuenta un set de datos de entrenamiento con 60,000 muestras y un set de prueba con 10,000 muestras. Cada muestra es una imagen en escala de grises de 28x28, asociada con una etiqueta de 10 clases.

Zalando pretende que Fashion-MNIST sirva como un reemplazo directo para el conjunto original de datos MNIST para la evaluación comparativa de algoritmos de aprendizaje automático. Comparte el mismo tamaño de imagen y estructura de entrenamiento y divisiones de pruebas.

Descripción Dataset

El tamaño de cada imagen es de 28 x 28 píxeles (alto x ancho), abarcando un total de 784 píxeles. Cada píxel tiene un único valor de píxel asociado, que indica la claridad u oscuridad de ese píxel, con números más altos que significan más oscuro. Este valor de píxel es un número entero entre 0 y 255.

Los conjuntos de datos de entrenamiento y prueba tienen 785 columnas. La primera columna consta de las etiquetas de clase (T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag y Ankle boot) asociadas a un tipo de prenda de vestir. El resto de las columnas contienen los valores de píxeles de la imagen asociada. Para ubicar un píxel en la imagen, suponga que x = i * 28 + j, donde i y j son enteros entre 0 y 27.

El píxel se encuentra en la fila i y columna j de una matriz de 28 x 28, por lo tanto, la matriz se compone de:

- Cada fila es una imagen separada
- La columna 1 es la etiqueta de clase.
- Las columnas restantes son números de píxeles (784 en total).
- Cada valor es la oscuridad del píxel (1 a 255)

A su vez, cada muestra de entrenamiento y de prueba es asignado a una de las siguientes etiquetas:

0 T-shirt/top	**4** Coat	**8** Bag
1 Trouser	**5** Sandal	**9** Ankle boot
2 Pullover	**6** Shirt	
3 Dress	**7** Sneaker	

AVANCE 2

22 Julio 2020

El siguiente apartado tiene el carácter de complementario a la descripción del Dataset entregada en el anterior *Avance 1*. Por lo tanto, a continuación, se entrega una descripción más detallada y/o específica del Dataset a utilizar (cantidad de datos, de dónde vienen, sus distintos atributos y etiquetas, etc.), con el objetivo de tener un mejor entendimiento de ellos. Para tales efectos, se desarrolla una exploración de ellos utilizando Python en Google Colaboratory (Colab) con acceso a GPUs dada la cantidad de datos.

Los datos son extraídos directamente de la API Keras de TensorFlow v2.2.0 (tf.keras.datasets.fashion_mnist.load_data).

Fashion-MNIST puede usarse como reemplazo directo para el conjunto de datos MNIST original (10 categorías de dígitos escritos a mano). Comparte el mismo tamaño de imagen (28x28) y estructura de entrenamiento (60,000) y divisiones de prueba (10,000). Por su lado, Keras es una API de aprendizaje profundo de alto nivel popular y bien considerada. Está integrado directamente en TensorFlow, además de ser un proyecto de código abierto independiente. Su importación es a través de “tf.keras”, permitiendo aprovechar la funcionalidad en su ejecución y la utilidad de “tf.data”.

La exploración y visualización del Dataset se ejecuta en Jupyter Notebook en Drive de Google Colab, y se desarrolla de la siguiente forma:

1.Conexión e Importación del Dataset

Para conectar con el Drive de Google se utiliza la siguiente línea de código “from google.colab import drive”. Luego, cargamos los datos de fashion_mnist con la API keras.datasets con solo una línea de código “tf.keras.datasets.fashion_mnist.load_data”.

2. Exploración y Visualización de los Datos

De acuerdo con la figura 1, los siguientes códigos tiene por objetivo explorar los conjuntos de datos de entrenamiento y de prueba. Hay que recordar que cada imagen está en la escala de grises es 28x28.

```
[1] # para conectar con Drive
    from google.colab import drive

    drive.mount('/content/drive/')

    PATH = '/content/drive/My Drive/ML_Practicas_2020/'

    Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force_remount=True).

[2] #Descargar los datos de fashion_mnist

    import tensorflow as tf
    from tensorflow import keras

    import numpy as np
    import matplotlib.pyplot as plt

    (x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()

    # Verificación carga completa del Dataset:
    ## Conjunto de Entrenamiento
    print("x_train shape:", x_train.shape, "y_train shape:", y_train.shape)
    print("x_train Dimensiones:", x_train.ndim, "y_train Dimensiones:", y_train.ndim)
    print("x_train Tipo de Dato:", x_train.dtype, "y_train Tipo de Dato:", y_train.dtype)
    print("x_train Tamaño:", x_train.size, "y_train Tamaño:", y_train.size)

    print('------------------------------------------------------------')

    ## Conjunto de Test
    print("x_test shape:", x_test.shape, "y_test shape:", y_test.shape)
    print("x_test Dimensiones:", x_test.ndim, "y_test Dimensiones:", y_test.ndim)
    print("x_test Tipo de Dato:", x_test.dtype, "y_test Tipo de Dato:", y_test.dtype)
    print("x_test Tamaño:", x_test.size, "y_test Tamaño:", y_test.size)
```

Figura 1: Código Exploración del Dataset

De la codificación presentada se obtiene como resultados, el siguiente resumen:

x_train shape: (60000, 28, 28) y_train shape: (60000,)
x_train Dimensiones: 3 y_train Dimensiones: 1
x_train Tipo de Dato: uint8 y_train Tipo de Dato: uint8
x_train Tamaño: 47040000 y_train Tamaño: 60000
---
x_test shape: (10000, 28, 28) y_test shape: (10000,)
x_test Dimensiones: 3 y_test Dimensiones: 1
x_test Tipo de Dato: uint8 y_test Tipo de Dato: uint8
x_test Tamaño: 7840000 y_test Tamaño: 10000

Por consiguiente, sé constata que hay 60,000 imágenes en el conjunto de datos de entrenamiento y 10,000 en el conjunto de datos de prueba y que las imágenes están en la escala de grises con 28x28 píxeles.

2.1 Visualización de los Datos

Para tales efectos, utilizó la librería matplotlib (import matplotlib.pyplot as plt), con el fin de visualizar una de las imágenes de los conjuntos de datos de entrenamiento haciendo uso de "imshow ()" para visualizar. (ver figura 2)

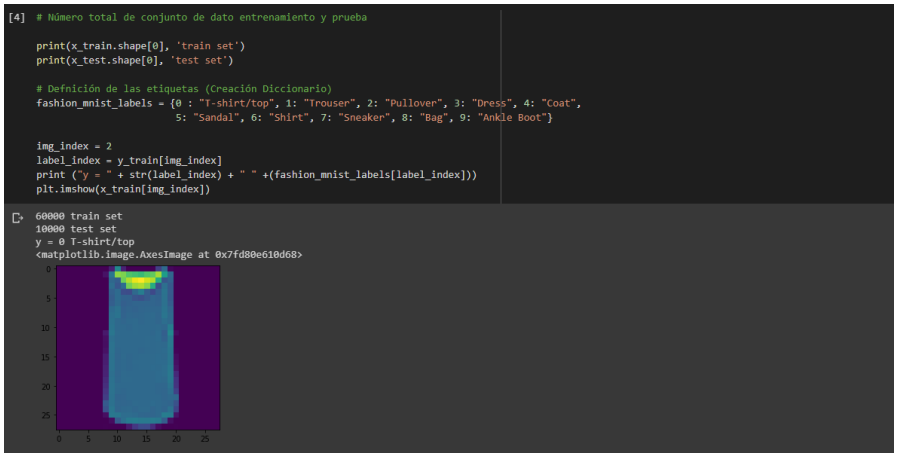


Figura 2: Código y Visualización – Imagen del Conjunto de Datos de Entrenamiento

Posteriormente, de acuerdo con la figura 3, se elabora un gráfico de las primeras nueve imágenes del mismo conjunto de datos, mostrando que las imágenes son fotografías en escala de grises de prendas de vestir.

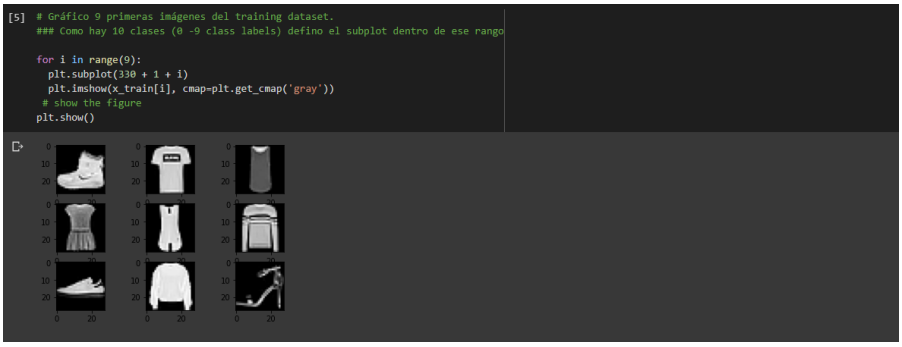


Figura 3: Código y Visualización – Primeras 9 Imagen del Conjunto de Datos de Entrenamiento

Finalmente, se realiza una cuantificación de imágenes por clase (10 clases) para cada conjunto de dato, de manera tal, de comprobar y visualizar la distribución de las imágenes por clase. De acuerdo con la figura 4, para el caso del conjunto de datos de entrenamiento se desarrolla la siguiente codificación:

```
[6] # Cuántas imagenes hay por clase en cada conjunto de dato

    labels = {0 : "T-shirt/top", 1: "Trouser", 2: "Pullover", 3: "Dress", 4: "Coat",
              5: "Sandal", 6: "Shirt", 7: "Sneaker", 8: "Bag", 9: "Ankle Boot"}

    # Para conjunto de entrenamiento
    unique, counts = np.unique(y_train, return_counts=True)

    print(unique, counts)

    [0 1 2 3 4 5 6 7 8 9] [6000 6000 6000 6000 6000 6000 6000 6000 6000 6000]

[ ] unique= labels
    print(unique, counts)

    {0: 'T-shirt/top', 1: 'Trouser', 2: 'Pullover', 3: 'Dress', 4: 'Coat', 5: 'Sandal', 6: 'Shirt', 7: 'Sneaker', 8: 'Bag', 9: 'Ankle Boot'} [6000 6000 6000 6000 6000 6000 6000 6000 6000 6000]

[7] %matplotlib inline
    import matplotlib.pyplot as plt; plt.rcdefaults()
    import numpy as np
    # Plot
    label ={"T-shirt/top","Trouser", "Pullover","Dress","Coat",
            "Sandal","Shirt","Sneaker","Bag","Ankle Boot")
    objects = (labels)
    y_pos=np.arange(len(objects))
    counts = [6000,6000,6000,6000,6000,6000,6000,6000,6000,6000]
    plt.subplots(1,1, figsize=(12,4))
    plt.bar (y_pos, counts, align='center', alpha=0.5, color='rgbkymc')
    plt.xticks(y_pos, objects)
    plt.ylabel('counts')
    plt.title('Número de Imágenes por Clase - Conjunto de Entrenamiento')
    plt.savefig(PATH+'my_figure.png')

    plt.show()
```

Figura 4: Código y Visualización – Número de Imágenes por Clase para el Conjunto de Datos de Entrenamiento

Obteniendo en la figura 5 la visualización la distribución de las imágenes por cada clase presente en el conjunto de datos de entrenamiento. Con lo cual se confirma el Dataset-Entrenamiento está perfectamente balanceado:

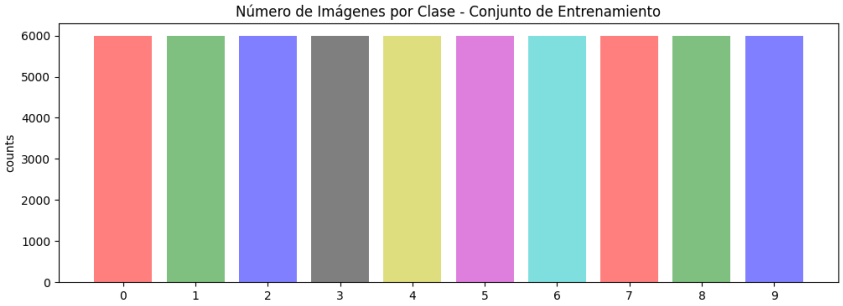


Figura 5: Visualización – Distribución de Imágenes por Clase para el Conjunto de Datos de Entrenamiento

A continuación, en la figura 6 se muestra la misma codificación para el conjunto de datos de prueba:

```
[8] # Para conjunto de test
    unique, counts = np.unique(y_test, return_counts=True)
    print(unique, counts)

    [0 1 2 3 4 5 6 7 8 9] [1000 1000 1000 1000 1000 1000 1000 1000 1000 1000]

[9] unique= labels
    print(unique, counts)

    {0: 'T-shirt/top', 1: 'Trouser', 2: 'Pullover', 3: 'Dress', 4: 'Coat', 5: 'Sandal', 6: 'Shirt', 7: 'Sneaker', 8: 'Bag', 9: 'Ankle Boot'} [1000 1000 1000 1000 1000 1000 1000 1000 1000 1000]

[10] # Plot
    label ={"T-shirt/top","Trouser", "Pullover","Dress","Coat",
            "Sandal","Shirt","Sneaker","Bag","Ankle Boot")
    objects = (labels)
    y_pos=np.arange(len(objects))
    counts = [1000,1000,1000,1000,1000,1000,1000,1000,1000,1000]
    plt.subplots(1,1, figsize=(12,4))
    plt.bar (y_pos, counts, align='center', alpha=0.5, color='rgbkymc')
    plt.xticks(y_pos, objects)
    plt.ylabel('counts')
    plt.title('Número de Imágenes por Clase - Conjunto de Prueba')
    plt.savefig(PATH+'my_figure2.png')

    plt.show()
```

Figura 6: Visualización – Distribución de Imágenes por Clase para el Conjunto de Datos de Prueba

Obteniendo en la figura 7 la visualización la distribución de las imágenes por cada clase presente en el conjunto de datos de prueba. Con lo cual se confirma el Dataset - Prueba está perfectamente balanceado:

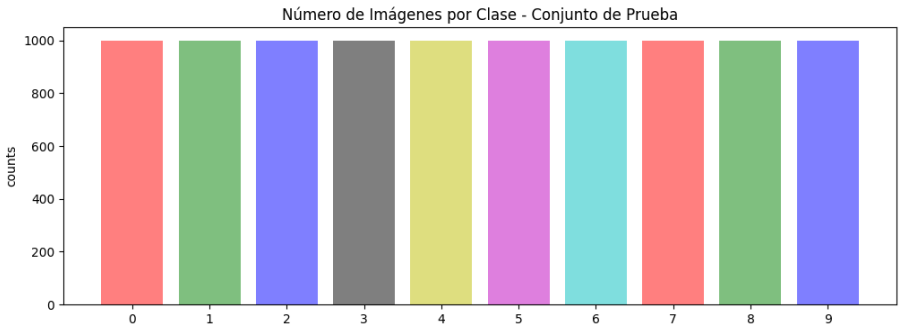


Figura 7: Visualización – Distribución de Imágenes por Clase para el Conjunto de Datos de Prueba

AVANCE 3

28 Julio 2020

El siguiente apartado entrega la propuesta del o los modelos con sus respectivas arquitecturas asociadas.

Los modelos propuestos para el conjunto de datos Fashion-MNIST son de tipo Clasificación utilizando arquitectura de *Redes Neuronales Convolucionales (CNN).*

Antes de definir el o los modelos a implementar se realizan algunos pasos previos; primero se *Normaliza el Dataset*, normalizando las dimensiones de estos con el objetivo que queden en la misma escala. Dicho en otras palabras, dado que los valores de píxeles de cada imagen son enteros en el rango de 0 y 255 en la escala de grises, estos son reescalados al rango 0 y 1, lo que se hace convirtiendo el tipo de datos en entero sin signos a flotantes y luego se dividen los valores de píxeles por el valor máximo.

Posteriormente *se Divide Dataset en Conjunto de Datos de Entrenamiento* (Imágenes utilizadas para entrenar el modelo)*, Validación* (Imágenes para ajustar los hiperparámetros y evaluar los modelos) *y Prueba* (para probar el modelo entregando Imágenes nuevas nunca vistas antes por el modelo), para tales efectos, el conjunto de entrenamiento de 60.000 imágenes es dividido en 50.000 imágenes para Entrenamiento y 10.000 para Validación.

Adicionalmente, se sabe que el Dataset presenta algunas características predeterminadas al minuto de bajar la data, tales como;

- Las imágenes están previamente segmentadas, es decir, cada imagen contiene una sola prenda de vestir.
- Las imágenes tienen el mismo tamaño 28 × 28 píxeles.
- Las imágenes están en escala de grises, entre 0 y 255.
- El Dataset contiene 10 clases y que estas se representan como enteros únicos.

Dicho lo anterior, es recomendable, por un lado, remodelar (reshape) las matrices de datos para que tengan un solo canal de color, actualizando así los datos de entrada de (28,28) a (28, 28,1) de acuerdo con la nueva división realizada. Por el otro lado, usar "*one hot encoding*" para las clases,

transformando el entero en un vector binario de 10 elementos con un 1 para el índice del valor de la clase.

Una vez terminada la preparación del Dataset anteriormente descrita, se procede a definir la arquitectura del modelo CNN.

1. Arquitectura Modelo

A continuación, se define un modelo de red neuronal convolucional de referencia para Dataset Fashion-MNIST. El modelo tiene dos aspectos principales; la primera es “the feature extraction front end” (la parte frontal de extracción de características) y la segunda es ”the classifier backend” backend clasificador que hará una predicción, esta son las capas de salidas.

1.1 Modelo 1 (ver figura 8 y 9)

Se utiliza el modelo Secuencial (model = Sequential ()).

- La primera capa convolucional (Conv2D ()) tendrá un filtro de (32) con una función de activación (relu) y se define la forma de los datos de entrada (imput_shape = (28,28,1)).
- La segunda tendrá la misma característica que la anterior capa convolucional (Conv2D ()), pero con un filtro de (64).
- Luego se le entrega un mapa de filtro (Flatten ()) para aplanar la imagen y así proporcionar características al clasificador.
- Finalmente, se le entrega una capa de salida densa (Dense ()) con una función de activación (softmax) y 10 nodos (10 clases) que clasificará las 10 categorías de datos en Fashion-MNIST.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
flatten_1 (Flatten)          (None, 36864)             0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                368650
=================================================================
Total params: 387,466
Trainable params: 387,466
Non-trainable params: 0
_________________________________________________________________
None
```

Figura 8: Resumen Arquitectura Modelo 1 – Python_Colab

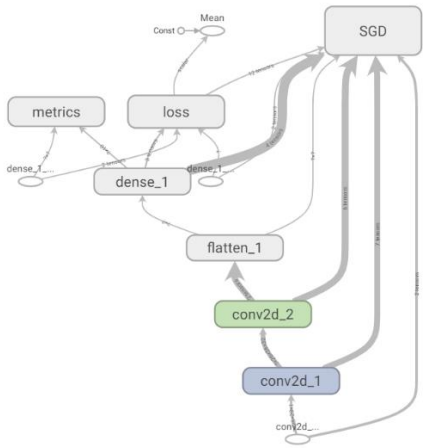


Figura 9: Grafo Modelo 1 – TensorBoard

1.2 Modelo 2 (ver figura 10 - 12)

Nuevamente se utiliza el modelo Secuencial (model = Sequential ()). Sin embargo, se adicionan las siguientes modificaciones:

- Se trabaja con *un Generador de Imagen (ImageDataGenerator)*: Este generador ofrece la capacidad de adaptarse a modelos utilizando el aumento de datos de imagen, cuyo objetivo es expandir el conjunto de datos de entrenamiento con el fin de mejorar el rendimiento y la capacidad del modelo para generalizar.
- Se agrega *Capa de Agrupación MaxPooling2D* seguida de c/capa convolucional; Reduce la muestra de entrada tomando el valor máximo para cada dimensión a lo largo del eje de características. Además, reduce el costo computacional al reducir el número de parámetros para aprender.

El modelo 2 queda de la siguiente forma:

- La primera capa convolucional (Conv2D ()) tendrá un filtro de (32) con una función de activación (relu) y se define la forma de los datos de entrada (imput_shape = (28,28,1)), seguida de una capa de agrupación máxima (MaxPooling2D ()).
- La segunda tendrá la misma característica que la anterior capa convolucional (Conv2D ()), pero con un filtro de (64), seguida por otra capa de agrupación máxima (MaxPooling2D ()) de igual característica que la anterior.
- Luego se le entrega un mapa de filtro (Flatten ()) para aplanar la imagen y así proporcionar características al clasificador.
- Finalmente, se le entrega una capa de salida densa (Dense ()) con una función de activación (softmax) y 10 nodos (10 clases) que clasificará las 10 categorías de datos en Fashion-MNIST.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                16010
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
_________________________________________________________________
None
```

Figura 10: Resumen Arquitectura Modelo 2 – Python_Colab

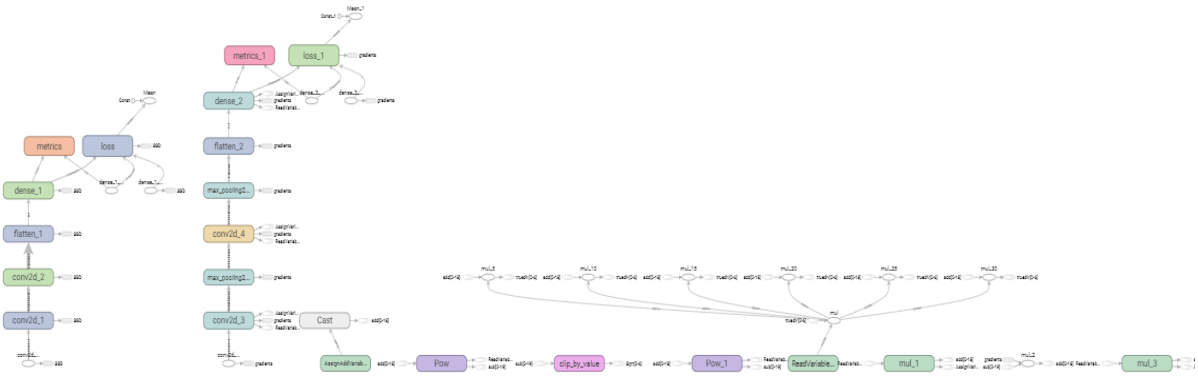


Figura 11: Parte del Grafo Modelo 2 - TensorBoard

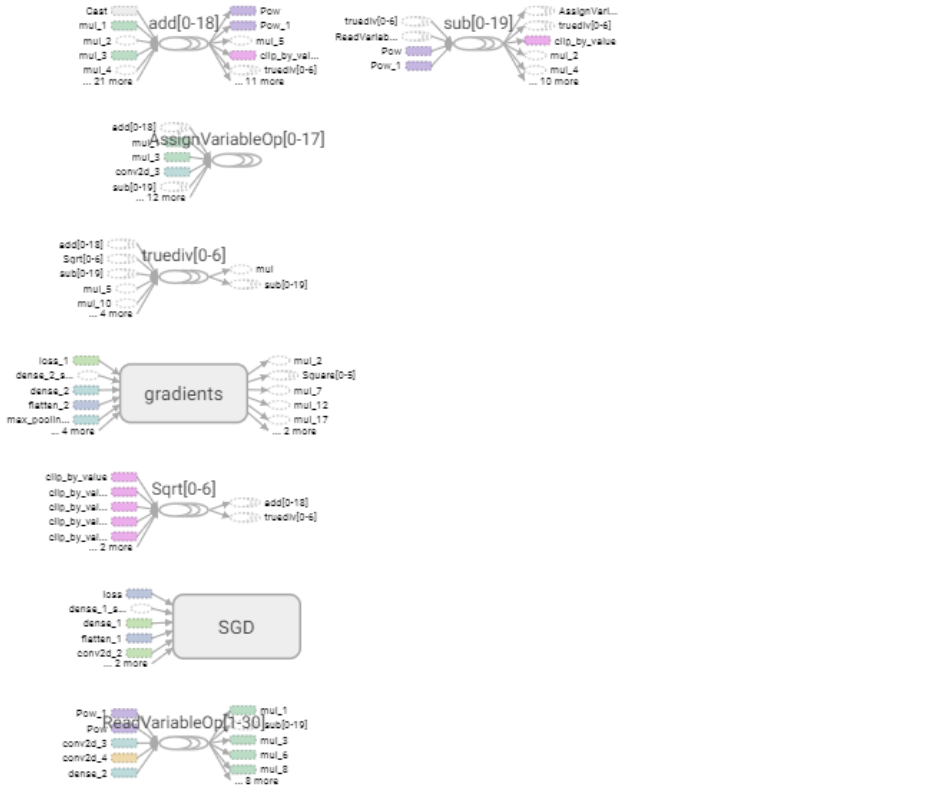


Figura 12: Grafo_Gradientes - Modelo 2 - TensorBoard

1.3 *Modelo 3 (ver figura 13 - 16)*

Nuevamente se utiliza el modelo Secuencial (model = Sequential ()), se mantiene las mismas modificaciones realizadas al modelo 2, con las siguientes adiciones:

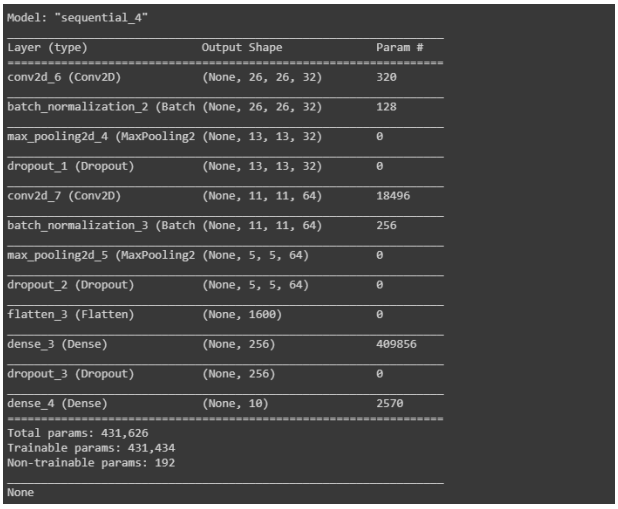
- Se agrega *Capa de BatchNormalization* seguida de c/capa convolucional; Transforma los datos que pasan a través de ella, de manera que estén estandarizados (media 0 y desviación estándar 1). Por lo tanto, la capa sigue las estadísticas de los datos de entrada y las utiliza para estandarizar los datos, por lo que la media y la varianza se va actualizando por cada

mini batch. Por otro lado, el hiperparámetro “momentum” permite controlar cuanto de las estadísticas del mini batch anterior son consideradas para la actualización en el mini batch actual. El valor por default es 0.99, si lo dejamos en 0 se utilizan sólo las estadísticas del mini batch actual. En resumen, permite mejorar la estabilidad del modelo y ayuda a que logre converger en menos épocas.

- Se agrega *Capa de Dropout* seguida de c/capa de agrupación máxima (MaxPooling2D); Método de regularización que "apaga" neuronas aleatoriamente de acuerdo con una probabilidad de apagado. Tiene el efecto de simular muchas redes neuronales con diferentes estructuras, haciendo la red neuronal más robusta a los datos de entrada. En resumen, esta capa permite obtener una mejor generalización del modelo.
- Se agrega *Capa Densa con 256 nodos* seguida del filtro (Flatten ()); Con el objetivo de suavizar el salto de 1600 a 10 presente en el modelo 2. El objetivo es asegurar menor perdida de información entre la capa Flatten y la capa de salida.

El modelo 3 queda de la siguiente forma:

- La primera capa convolucional (Conv2D ()) tendrá un filtro de (32) con una función de activación (relu) y se define la forma de los datos de entrada (imput_shape = (28,28,1)), seguida por una capa de BatchNormalization (BatchNormalization ()), luego le sigue una capa de agrupación máxima (MaxPooling2D ()) y termina con la capa de Dropout (Dropout ()).
- La segunda tendrá la misma característica que la anterior capa convolucional (Conv2D ()), pero con un filtro de (64), seguida por las capas de BatchNormalization (BatchNormalization ()), agrupación máxima (MaxPooling2D ()) y Dropout (Dropout ()), todas de igual característica que las anteriores.
- Luego se le entrega un mapa de filtro (Flatten ()) para aplanar la imagen y así proporcionar características al clasificador, seguida de una capa densa (Dense ()) con 256 nodos y función de activación (relu) y termina con la capa de Dropout (Dropout ()) de igual características que las anteriores.
- Finalmente, se le entrega una capa de salida densa (Dense ()) con una función de activación (softmax) y 10 nodos (10 clases) que clasificará las 10 categorías de datos en Fashion-MNIST.



```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
batch_normalization_2 (Batch (None, 26, 26, 32)        128
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
batch_normalization_3 (Batch (None, 11, 11, 64)        256
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               409856
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570
=================================================================
Total params: 431,626
Trainable params: 431,434
Non-trainable params: 192
_________________________________________________________________
None
```

Figura 13: Resumen Arquitectura Modelo 3 – Python_Colab

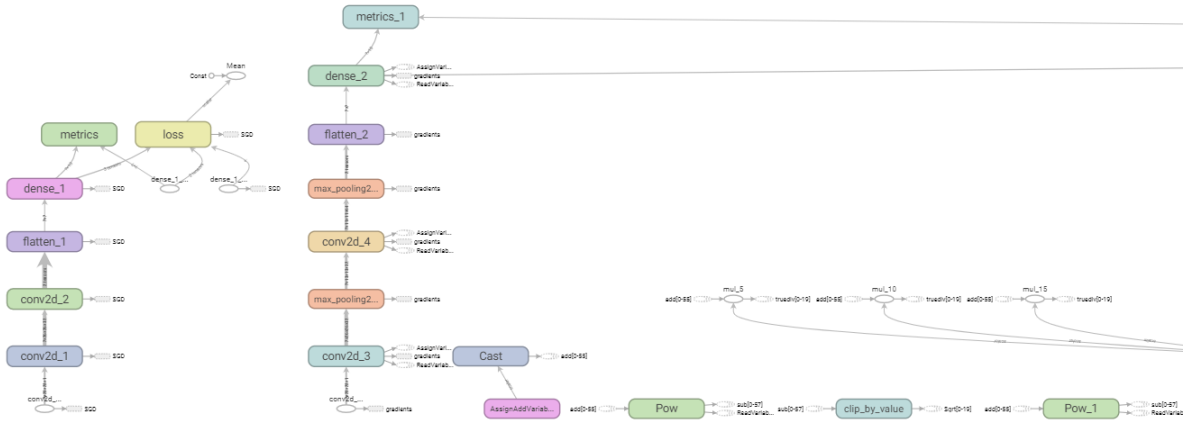


Figura 14: Parte de Inicio del Grafo Modelo 3 – TensorBoard

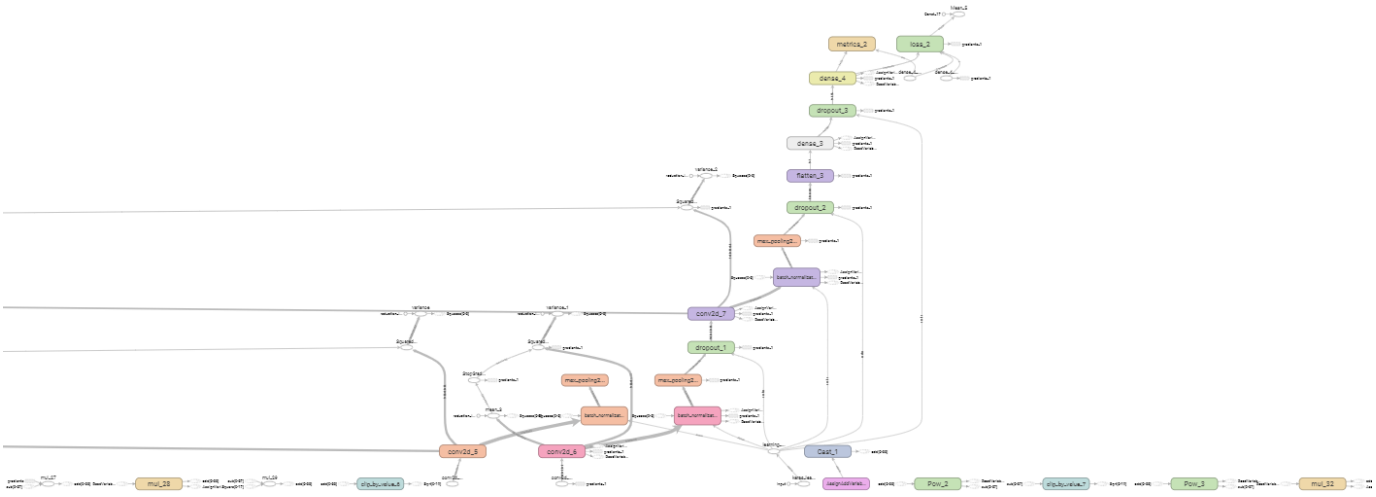


Figura 15: Parte Intermedia del Grafo Modelo 3 - TensorBoard

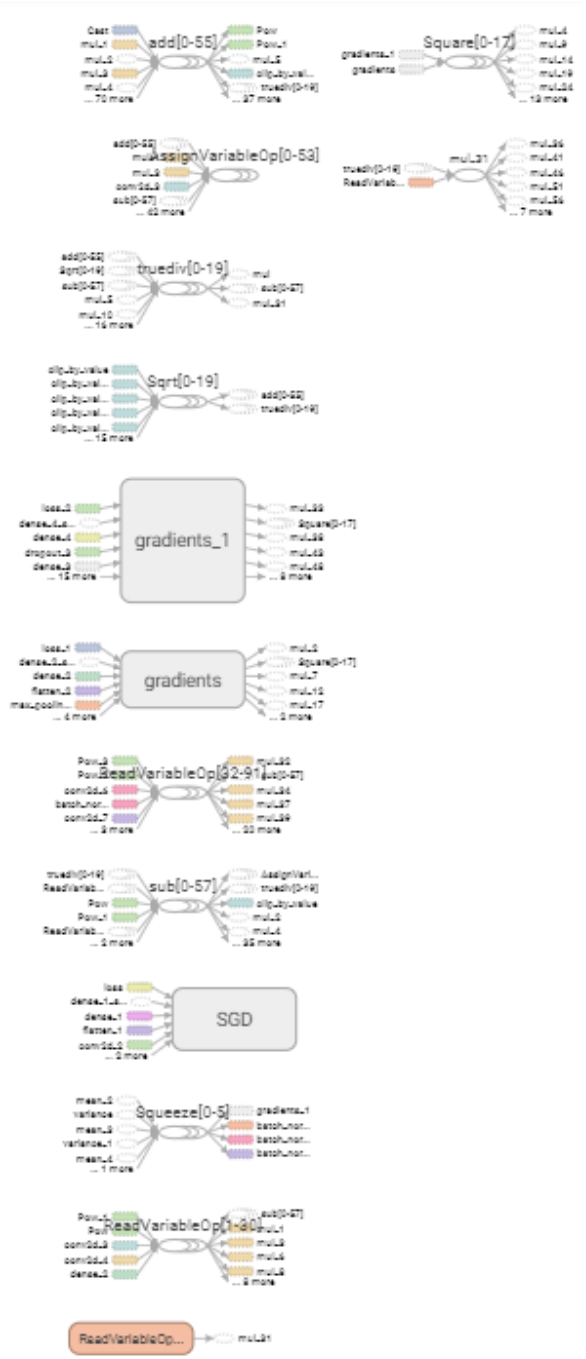


Figura 16: Grafo_Gradientes - Modelo 3- TensorBoard

Es importante aclarar que como parte de los objetivos se declaró probar varios modelos de clasificación, tales como Sequential, VGG 19 y AlexNet, sin embargo, luego del estudio de cada uno de los modelos, se llega a la conclusión, qué, dada las características del Dataset Fashion-MNIST (tamaño imagen 28 x28 - escala gris, es decir de 1 dimensión), los modelos VGG 19 y AlexNet no son aplicables. En el primer caso (VVG 19) este tipo de modelo están diseñados para imágenes RGB, es decir, imágenes en colores de tres dimensiones. Mientras que para el segundo caso (AlexNet) el modelo requiere de imágenes de tamaño 227 x 227.

Si bien, se podrían hacer las trasformaciones requeridas en cada modelo al Dataset Fashion-MNIST, este pierde sus características de origen y consigo la idea principal del proyecto que es encontrar un clasificador para ese tipo de data.

Por consiguiente, se decide aplicar el modelo secuencial con diferentes arquitecturas propuestas.

AVANCE 4 – ENTREGA

14 agosto 2020

El siguiente apartado se presentan los ajustes realizados a los modelos previamente propuestos en función a la retroalimentación recibida en la presentación del avance2. Posteriormente se presenta los resultados, evaluación y visualización de función de costo y accuracy de cada arquitectura de modelo. Finalmente, se muestra las predicciones de los modelos seleccionados.

1. Resultados y Evaluación de la Arquitectura del Modelo

A continuación, se presenta la arquitectura utilizada para cada modelo, seguida de los resultados obtenidos del entrenamiento en conjunto con las gráficas asociadas a la Función de Costo y Accuracy.

Cabe mencionar que para todos los entrenamientos de los diferentes modelos se trabaja con: Generador de imágenes (**ImageDataGenerator**), con Callbacks, en particular con **ModelCheckpoint** y **EarlyStopping** (patience= 50), se ocupa **500 épocas** para ejecutar el entrenamiento y son compilados con **optimizador ADM**.

Asimismo, entre los cambios incorporados en todos los modelos convolucional es la incorporación del parámetro **padding =´same'**, el cual cumple ayuda a obtener un mejor rendimiento del modelo. Este parámetro de manera predeterminada es padding= "valid", lo que significa que las convoluciones solo se aplican cuando es posible, al ser cambiado a padding "same" agrega valores cero alrededor de la entrada, de modo que la salida tenga el mismo tamaño que la entrada. Por consiguiente, del mapa de características de la imagen de entrada, una porción mayor tiene la oportunidad de participar o contribuir a la salida.

1.1 Entrenamiento Modelo Secuencial 1

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
flatten_1 (Flatten)          (None, 36864)             0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                368650
=================================================================
Total params: 387,466
Trainable params: 387,466
Non-trainable params: 0
_________________________________________________________________
None
```

Resultados:
Train: 99.81%
Val: 90.61%
Test: 90.47%

Visualización: Función de Costo y Accuracy

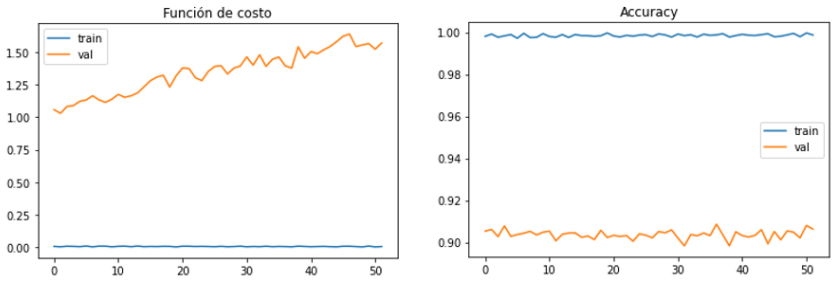


Figura 17: Función de Costo y Accuracy - Modelo 1

Las siguientes arquitecturas de modelos se les agrega Capa de Dropout de parámetro (0,5) a la salida con el objetivo de obtener una mejor generalización del modelo. Cabe mencionar que si bien, es recomendable usar este método de regularización lo más bajo posible (entre 0,2 y 0,3) al ser probados en los modelos los resultados entregados por el test accuracy no superaban al 50%.

Asimismo, con el objetivo de mejorar el rendimiento del modelo, en las siguientes arquitecturas se puede observar la variación que se realizara en el número de filtro utilizados en la capa convolucional. La más común es el aumento de Filtros en la capa, ya que permite más oportunidades de extraer características simples de las imágenes de entrada.

Dicho lo anterior, se comienza con un aumento gradual del número de filtros utilizados en las capas convolucional (pasando de 8 a 16 a 32 a 64) para así no perder tanta información de capa en capa. Luego, siguiendo la misma lógica, se hacen pruebas de aumento un poco más dramático como de 32 pasar a 64 filtros, o bien, pasar de 16 a 32 filtros.

Finalmente, se hacen dos pruebas haciendo el ejercicio inverso, es decir, realizar una disminución del número de filtros, por ejemplo, se prueba pasar de 64 a 32 a 16 a 8 filtros, y otra prueba de pasar de 32 a 16 filtro.

1.2 Entrenamiento Modelo Secuencial 2

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_10 (Conv2D)           (None, 28, 28, 8)         80
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 28, 28, 16)        1168
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 14, 14, 32)        4640
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 14, 14, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 3136)              0
_________________________________________________________________
batch_normalization_2 (Batch (None, 3136)              12544
_________________________________________________________________
dense_3 (Dense)              (None, 256)               803072
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570
=================================================================
Total params: 842,570
Trainable params: 836,298
Non-trainable params: 6,272
_________________________________________________________________
None
```

Resultados:
Train: 71.51%
Val: 70.32%
Test: 70.02%

Visualización: Función de Costo y Accuracy

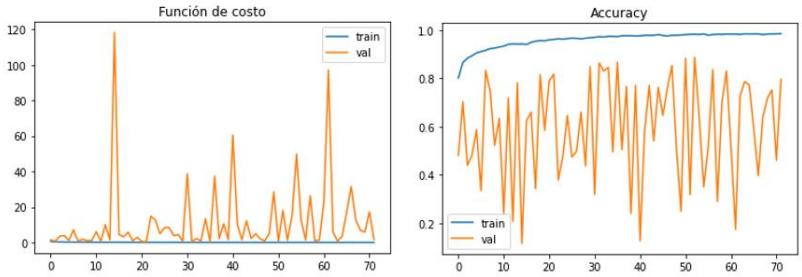


Figura 18: Función de Costo y Accuracy - Modelo 2

1.3 Entrenamiento Modelo Secuencial 3

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_55 (Conv2D)           (None, 28, 28, 32)        320
_________________________________________________________________
batch_normalization_9 (Batch (None, 28, 28, 32)        128
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_56 (Conv2D)           (None, 14, 14, 64)        18496
_________________________________________________________________
batch_normalization_10 (Batc (None, 14, 14, 64)        256
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 7, 7, 64)          0
_________________________________________________________________
flatten_16 (Flatten)         (None, 3136)              0
_________________________________________________________________
dense_29 (Dense)             (None, 256)               803072
_________________________________________________________________
dropout_21 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_30 (Dense)             (None, 10)                2570
=================================================================
Total params: 824,842
Trainable params: 824,650
Non-trainable params: 192
_________________________________________________________________
None
```

Resultados:
Train: 86.3%
Val: 84.39%
Test: 84.1%

Visualización: Función de Costo y Accuracy

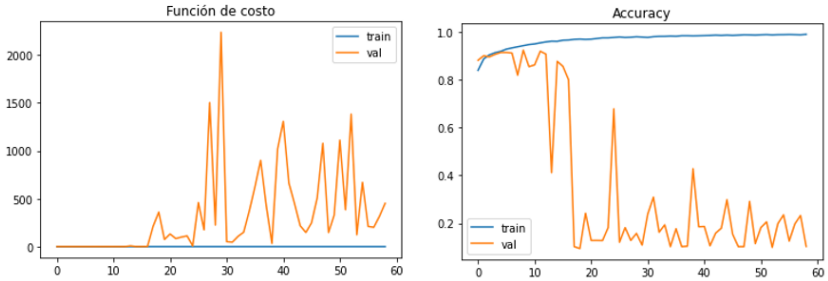


Figura 19: Función de Costo y Accuracy - Modelo 3

1.4 Entrenamiento Modelo Secuencial 4

```
Model: "sequential_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_57 (Conv2D)           (None, 28, 28, 64)        640
_________________________________________________________________
conv2d_58 (Conv2D)           (None, 28, 28, 32)        18464
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_59 (Conv2D)           (None, 14, 14, 16)        4624
_________________________________________________________________
conv2d_60 (Conv2D)           (None, 14, 14, 8)         1160
_________________________________________________________________
max_pooling2d_34 (MaxPooling (None, 7, 7, 8)           0
_________________________________________________________________
flatten_17 (Flatten)         (None, 392)               0
_________________________________________________________________
dense_31 (Dense)             (None, 256)               100608
_________________________________________________________________
dropout_22 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_32 (Dense)             (None, 10)                2570
=================================================================
Total params: 128,066
Trainable params: 128,066
Non-trainable params: 0
_________________________________________________________________
None
```

Resultados:
Train: 84.7%
Val: 83.45%
Test: 82.52%

Visualización: Función de Costo y Accuracy

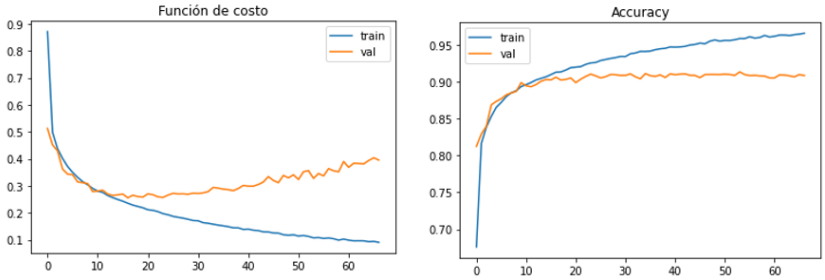


Figura 20: Función de Costo y Accuracy - Modelo 4

1.5 Entrenamiento Modelo Secuencial 5

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_39 (Conv2D)           (None, 28, 28, 16)        160
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 28, 28, 16)        2320
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 14, 14, 32)        4640
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 14, 14, 32)        9248
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 7, 7, 32)          0
_________________________________________________________________
flatten_11 (Flatten)         (None, 1568)              0
_________________________________________________________________
dense_19 (Dense)             (None, 256)               401664
_________________________________________________________________
dropout_12 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_20 (Dense)             (None, 10)                2570
=================================================================
Total params: 420,602
Trainable params: 420,602
Non-trainable params: 0
_________________________________________________________________
None
```

Resultados:
Train: 83.08%
Val: 82.26%
Test:80.94%

Visualización: Función de Costo y Accuracy

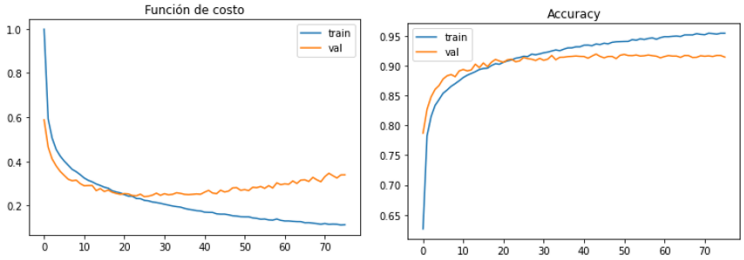


Figura 21: Función de Costo y Accuracy - Modelo 5

1.6 Entrenamiento Modelo Secuencial 6

Model: "sequential_14"

Layer (type)	Output Shape	Param #
conv2d_43 (Conv2D)	(None, 28, 28, 32)	320
conv2d_44 (Conv2D)	(None, 28, 28, 32)	9248
max_pooling2d_23 (MaxPooling	(None, 14, 14, 32)	0
conv2d_45 (Conv2D)	(None, 14, 14, 16)	4624
conv2d_46 (Conv2D)	(None, 14, 14, 16)	2320
max_pooling2d_24 (MaxPooling	(None, 7, 7, 16)	0
flatten_12 (Flatten)	(None, 784)	0
dense_21 (Dense)	(None, 256)	200960
dropout_13 (Dropout)	(None, 256)	0
dense_22 (Dense)	(None, 10)	2570

Total params: 220,042
Trainable params: 220,042
Non-trainable params: 0

Resultados:
Train: 84.52%
Val: 83.46%
Test: 82.41%

Visualización: Función de Costo y Accuracy

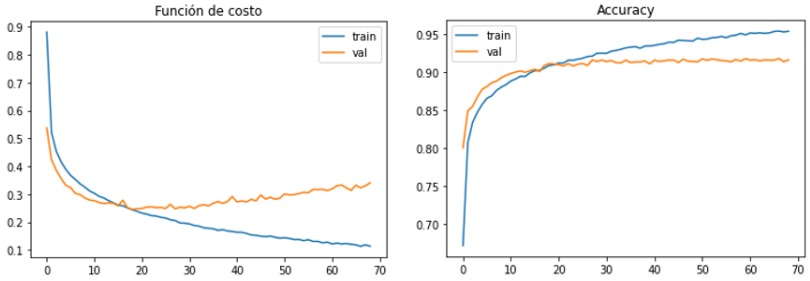


Figura 22: Función de Costo y Accuracy - Modelo 6

1.7 Análisis Visualización de Función de Costo y Accuracy de los Modelo

De acuerdo con las figuras 20,21 y 22 se puede observar en las curvas de aprendizaje de los 3 modelos que, la curva del conjunto de validación ya convergió (curva casi plana), por lo tanto, no tiene sentido seguir entrenando el modelo, dado que no va a mejorar. En cuanto a la función de costo de los 3 modelos, se puede ver que la curva de entrenamiento en la función de costo comienza a descender cada vez más, siendo conducente a la conclusión que hay un sobreajuste. Por lo tanto, los 3 modelos no se deben entrenarse más, de lo contrario los modelos se seguirán sobre ajustándose.

Por otro lado, la figura 19, se observa en la curva de validación que el accuracy baja más en la medida que más se entrene el modelo 3 lo que se puede deber a un mega sobreajuste, por lo tanto, nuevamente no tiene sentido seguir entrenando dado que este no va a mejorar. Una posible

solución a esto es aumentar el tamaño de los batch cosa que existan menos etapas de actualización de parámetros así la curva de accuracy subirá más lento al principio dando más espacio antes de llegar al 80% como se puede apreciar en la figura.

En relación con la figura 18, la cual corresponde al modelo 2, los resultados obtenidos, dada la arquitectura propuesta, no es congruente con la visualización obtenida de la función de costo y accuracy, por consiguiente, la información es insuficiente para hacer un análisis certero. En este contexto, pese al reiterado entrenamiento con diferente número de épocas, la incongruencia entre los resultados y la visualización de la función de costo y accuracy se mantuvo.

Finalmente, de acuerdo con los resultados obtenidos del entrenamiento de los modelos, se procede a realizar las predicciones solo en aquellos modelos que tuvieron mejor porcentaje de Test Accuracy, por lo tanto, se presentan las predicciones asociadas a los modelos 1, 3 y 6.

2. Visualización Predicciones de los Modelos Seleccionados

Se realizan las predicciones de los modelos a partir del conjunto de datos de prueba. Para tales efectos, se imprimen 15 imágenes del conjunto de datos de prueba en conjunto con sus respectivos títulos, previamente configurados con etiqueta de predicción. Para efectos de visualización se configura de manera tal que, si la predicción coincide con la etiqueta verdadera, el título ***será de color verde***, de lo contrario, se muestra en ***color rojo***

2.1 Predicción Modelo 1

Figura 24: Predicción - Modelo 1

De acuerdo con la figura 24, la predicción del modelo fue bastante certera, puesto que todas las imágenes están en verde indicando que la clasificación por el modelo fue correcta en relación con la etiqueta verdadera.

2.2 Predicción Modelo 3

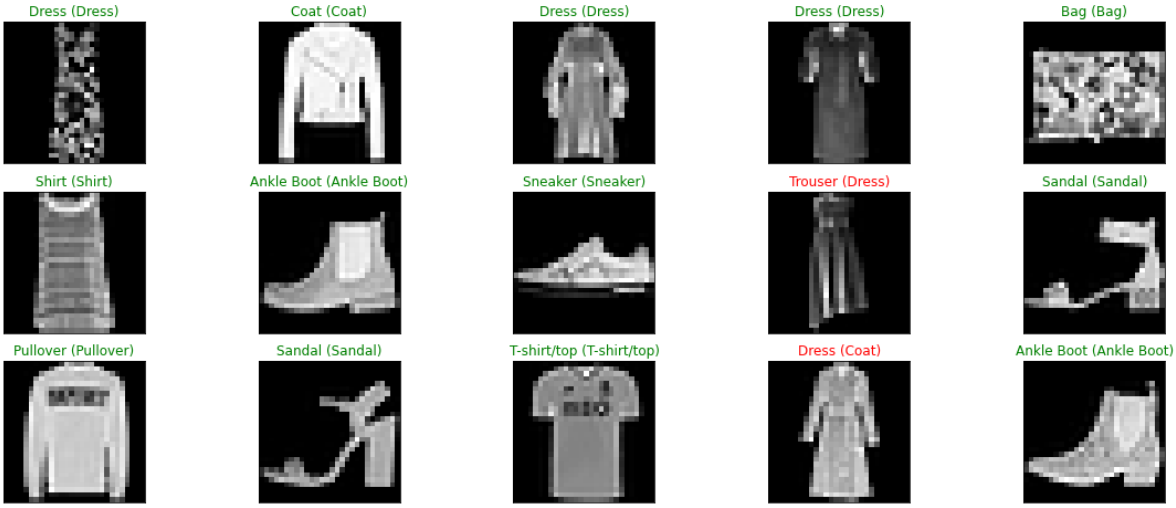


Figura 25: Predicción - Modelo 3

De acuerdo con la figura 25, la clasificación realizada por el modelo no fue los suficientemente buena. De las 15 imágenes, se observan **2** de ellas en donde la predicción fue **incorrecta,** las cuales se visualizan en color rojo; por ejemplo, la etiqueta entregada por el modelo dice “Dress” debiendo decir “Coat”.

2.3 Predicción Modelo 6

Figura 26: Predicción - Modelo 6

De acuerdo con la figura 26, la clasificación realizada por el modelo no fue los suficientemente buena. De las 15 imágenes, se observan **3** de ellas en donde la predicción fue **incorrecta**, las cuales se visualizan en color rojo; por ejemplo, la etiqueta entregada por el modelo dice “Coat” debiendo decir “Shirt”.

CONCLUSIONES

En términos de predicción y resultado obtenido por cada uno de los modelos propuestos, se puede concluir que el primer modelo logra cumplir con el objetivo de clasificar con un bajo porcentaje de error. Sin embargo, la arquitectura aún debe ser depurada para ser extrapolada como modelo clasificador de prendas de vestir a aplicar como sistema en alguna bodega de almacenamiento del rubro.