Nicolas Cari Rodríguez

Sistemas Inteligentes - Redes Neuronales Artificiales - Laboratorio

UCB - Cochabamba, 24 Abril del 2023

1. **Describe que hace este codigo**:

```
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

```
train_images.shape
```

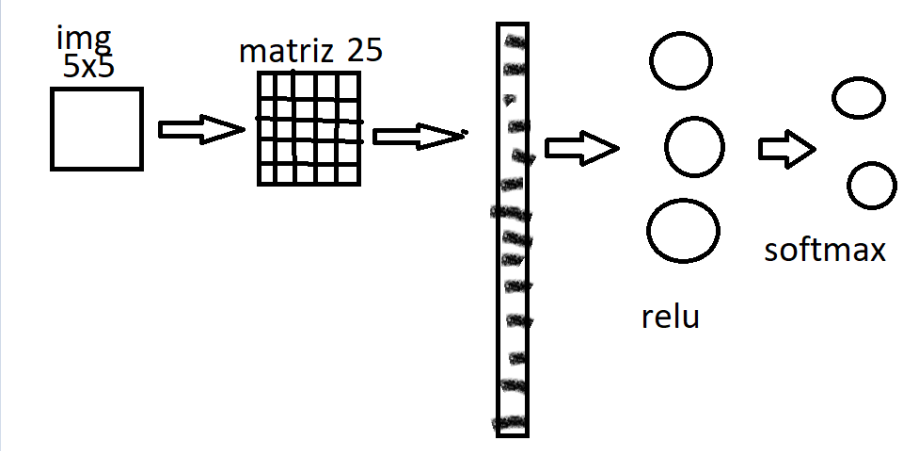
```
len(test_labels)
```

La primera parte agarra los valores del dataset de fashion mnist que son del dataset para encontrar ropa, separamos las imágenes en un dataset con train images y test images y las separamos.
Train images shape nos muestra el tamaño del dataset de entrenamiento
Len test labels nos muestra el tamaño de conjunto de etiquetas, en este caso son 10.

2. **Observa esta red neuronal y responde:**

```
model = keras. Sequential ([
keras. layers. Flatten (input_shape=(5, 5)),
keras. layers. Dense (3, activation='relu '),
keras. layers. Dense (2, activation='softmax')
])
```

- Dibuja la red neuronal. ¿Qué recibe como entrada? ¿Cuál es su salida?



Entra con una imagen que se hace flatten para tenerla como array, luego sale con softmax donde se muestra si es una cosa u otra.

- ¿En qué capa se usa relu y en cual softmax? ¿Por que?

Se usa relu en las capas ocultas porque son los mejores para evitar la perdida de gradiente.

- ¿Para qué sirve la función Flatten?

Convierte la imagen que tenemos en un array unidimensional para que tengamos los valores de los pixeles de la imagen.

- ¿Para qué sirve la función Dense?

La funcion Dense conecta la capa que se utiliza dense con todas las neuronas de la capa anterior, para que estén fully connected.

- Aumenta dos capas ocultas con 16 neuronas en cada una.

```
model = keras. Sequential ([
keras. layers. Flatten (input_shape=(5, 5)),
keras. layers. Dense (16, activation='relu '),
keras. layers. Dense (16, activation='relu '),
keras. layers. Dense (3, activation='relu '),
keras. layers. Dense (2, activation='softmax')
])
```

3. Observa este codigo y responde:

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

- ¿Qué hace un optimizer? ¿Cuál vimos en clase?

Un optimizer es el optimizador que utilizara el modelo para cambiar los pesos, el visto en clase es el descenso de gradiente.

- ¿Qué es el loss o la funcion de costo? ¿Cual vimos en clase?

Es la diferencia entre el resultado que tenemos y el resultado que queríamos. En clase vimos el mean square error.

- ¿Cuándo se usa sparse_categorical_crossentropy?

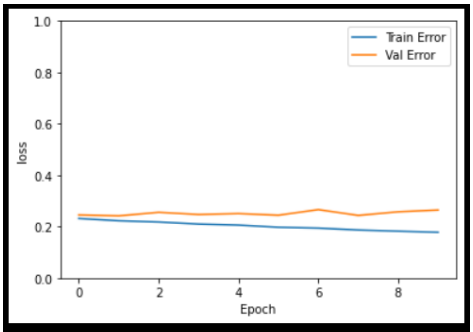
Se utiliza cuando las etiquetas de salida son categorías discretas, es decir para problemas con softmax donde tenemos muchas etiquetas.

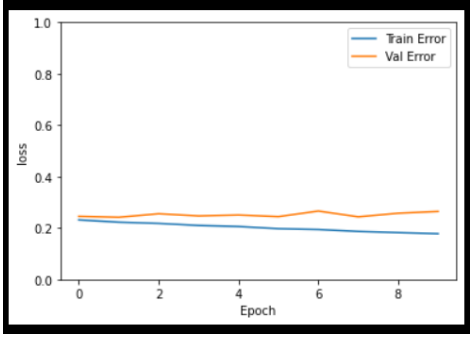
- ¿Qué otras metricas existen para clasificacion?

Accuracy, precision, recall, f1-score, especificidad por decir algunos.

- Experimenta con otros parametros, ¿que resultados obtienes?

Nuestros primeros resultados eran los siguientes:



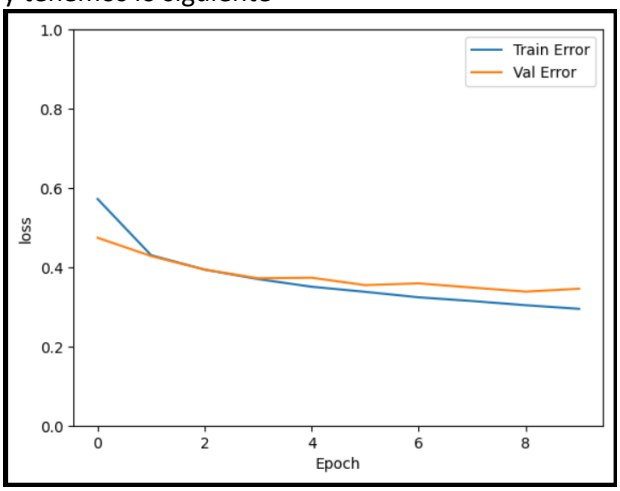


Test accuracy: 0.8770999908447266

Cambiamos lo siguiente

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

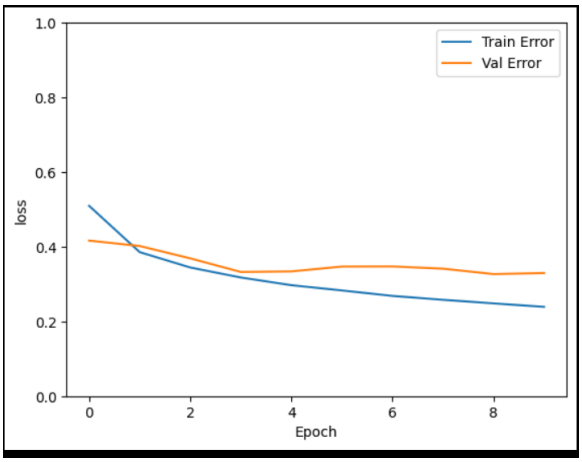
y tenemos lo siguiente



313/313 - 0s - loss: 0.3674 - accuracy: 0.8720 - 388ms/epoch - 1ms/step

Test accuracy: 0.871999979019165

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



313/313 - 0s - loss: 0.3599 - accuracy: 0.8779 - 340ms/epoch - 1ms/step

Test accuracy: 0.8779000043869019

En estos 2 ejemplos nuestros valores de acurracy son similares pero nuestro loss es mayor, dando a entender que utilizar solamente Adam es mejor.

Si cambiamos el loss algunos valores no acepta el modelo como ser categorical_crossentropy, que solo acepta 2 valores no 10 como en nuestro modelo.

4. Observa este código

```
hist = model. fit (train_images, train_labels, validation_split = 0.2, epochs=100)
```

- ¿Qué es lo que hace? ¿Que significan los parametros?

Esto entrena el modelo con nuestras train images, train labels que son nuestro conjunto de entrenamiento, con un validation Split de 0.2 que significa que utilizaremos el 20% de los datos cada epoch, y la cantidad de epochs que son la cantidad de veces que el modelo recorrerá todo el conjunto de entrenamiento.

- ¿Cuál es la diferencia entre loss y val _loss?

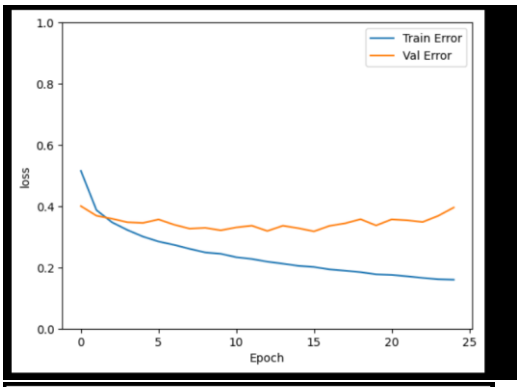
Loss es la perdida en nuestro conjunto de entrenamiento y val_loss es la perdida en nuestro conjunto de testeo, (val_loss es validation loss)

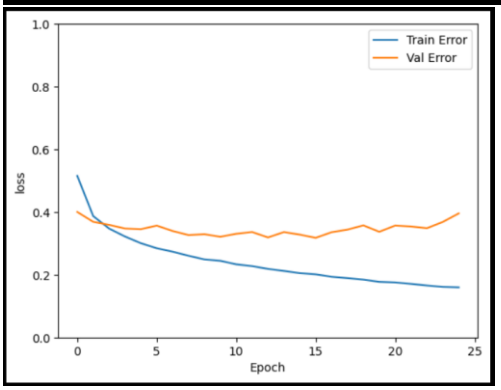
- ¿Cuál es la diferencia entre accuracy y val_accuracy?

Igual que el anterior es la acurracy de nuestro conjunto de entrenamiento y nuestro conjunto de testeo.

- Experimenta con el número de epochs.

Cambiando los epochs a 25:

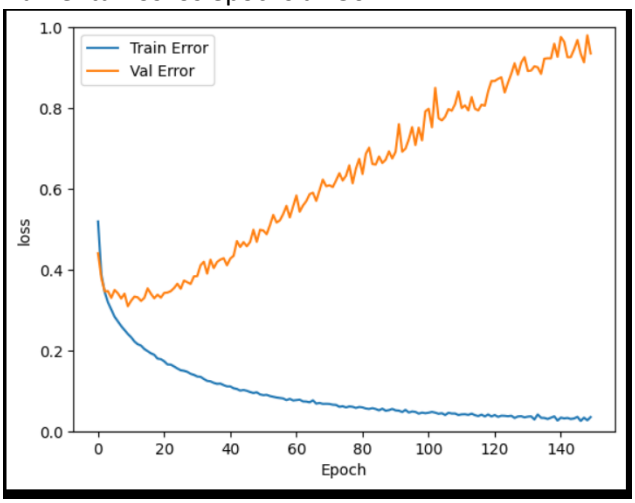




313/313 - 0s - loss: 0.4315 - accuracy: 0.8821 - 388ms/epoch - 1ms/step

Test accuracy: 0.882099986076355

Aumentamos los epochs a 150:



313/313 - 0s - loss: 1.0649 - accuracy: 0.8856 - 313ms/epoch - 999us/step

Test accuracy: 0.8855999708175659

Podemos ver que el test tiene mejor accuraccy pero el modelo esta sobreentrenado.

5. ¿Como puedes hacer predicciones dado un modelo entrenado?

Ejectutamos el modelo dándole un dato nuevo o un conjunto de datos que no se haya utilizado para entrenar y luego vemos si deduce de manera correcta que es nuestro dato, en estos casos podemos utilizar graficas para ver las predicciones y ajustar el modelo utilizándolas como base.

6. Que hace este codigo: class_names[np.argmax(predictions[10])]

De las class names que tenemos agarramos el valor de predicción más alta entre nuestros labels, para la predicción de la posición 10 de nuestros datos de testeo.