

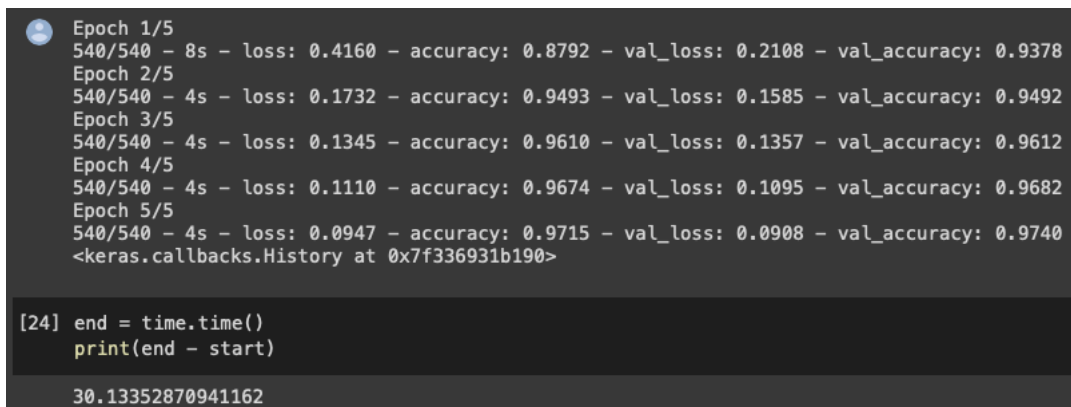
Universidad del Valle de Guatemala
Departamento de Matemática
Licenciatura en Matemática Aplicada

Estudiante: Rudik Roberto Rompich
Correo: rom19857@uvg.edu.gt
Carné: 19857

CC3066 - Data Science I - Catedrático: Luis Furlan
9 de septiembre de 2021

Laboratorio 6 - MNIST

Instrucciones: El modelo que se desarrolló en clase tiene una precisión ya bastante alta. Sin embargo, hay varios ajustes que se pueden intentar para mejorarlo. Es importante poner atención al tiempo que se tarda cada época en ejecutar. Utilizando el código visto en clase, experimenten con los hiperparámetros del algoritmo.



```
Epoch 1/5  
540/540 - 8s - loss: 0.4160 - accuracy: 0.8792 - val_loss: 0.2108 - val_accuracy: 0.9378  
Epoch 2/5  
540/540 - 4s - loss: 0.1732 - accuracy: 0.9493 - val_loss: 0.1585 - val_accuracy: 0.9492  
Epoch 3/5  
540/540 - 4s - loss: 0.1345 - accuracy: 0.9610 - val_loss: 0.1357 - val_accuracy: 0.9612  
Epoch 4/5  
540/540 - 4s - loss: 0.1110 - accuracy: 0.9674 - val_loss: 0.1095 - val_accuracy: 0.9682  
Epoch 5/5  
540/540 - 4s - loss: 0.0947 - accuracy: 0.9715 - val_loss: 0.0908 - val_accuracy: 0.9740  
<keras.callbacks.History at 0x7f336931b190>  
  
[24] end = time.time()  
      print(end - start)  
  
30.13352870941162
```

Figura 1: Caso base hecho en clase.

1. Problemas

Problema 1. *El ancho (tamaño de la capa escondida) del algoritmo. Intenten con un tamaño de 200.*

```

Epoch 1/5
540/540 - 9s - loss: 0.2760 - accuracy: 0.9198 - val_loss: 0.1266 - val_accuracy: 0.9632
Epoch 2/5
540/540 - 4s - loss: 0.1059 - accuracy: 0.9676 - val_loss: 0.0978 - val_accuracy: 0.9697
Epoch 3/5
540/540 - 4s - loss: 0.0699 - accuracy: 0.9790 - val_loss: 0.0655 - val_accuracy: 0.9810
Epoch 4/5
540/540 - 4s - loss: 0.0517 - accuracy: 0.9838 - val_loss: 0.0535 - val_accuracy: 0.9840
Epoch 5/5
540/540 - 4s - loss: 0.0393 - accuracy: 0.9875 - val_loss: 0.0511 - val_accuracy: 0.9840
<keras.callbacks.History at 0x7f3d78146dd0>

[24] end = time.time()
    print(end - start)

30.20947504043579

```

Figura 2: Tamaño de capa escondida de 200.

1. ¿Cómo cambia la precisión de validación del modelo?

Solución. Comparado al modelo original, en la quinta época se obtiene una mejora de 2% en la precisión del modelo. ☐

2. ¿Cuánto tiempo se tardó el algoritmo en entrenar?

Solución. 30.209 segundos en entrenar; levemente más tardado que en el caso original. ☐

3. ¿Puede encontrar un tamaño de capa escondida que funcione mejor?

Solución. Se compararon los siguientes casos:

Tamaño capa	Épocas	Tiempo (s)	Precisión
50	5	30.13	97.4 %
200	5	30.20	98.4 %
300	5	42.14	98.65 %
400	5	50.94	98.33 %

Cuadro 1: Tamaños de capa

En conclusión, a partir de la capa 200 ya se cuenta con una precisión excelente y agregar capas no necesariamente mejorar el modelo, como sucede en la el tamaño de capa de 400 que presenta un porcentaje más bajo que en el tamaño de capa de 300. Por lo que se recomendaría usar un tamaño de capa levemente superior a 200. ☐

Problema 2. La profundidad del algoritmo. Agreguen una capa escondida más al algoritmo. ¡Este es un ejercicio extremadamente importante!

```
Epoch 1/5
540/540 - 10s - loss: 0.2665 - accuracy: 0.9206 - val_loss: 0.1330 - val_accuracy: 0.9595
Epoch 2/5
540/540 - 5s - loss: 0.1018 - accuracy: 0.9685 - val_loss: 0.0869 - val_accuracy: 0.9735
Epoch 3/5
540/540 - 5s - loss: 0.0673 - accuracy: 0.9787 - val_loss: 0.0756 - val_accuracy: 0.9758
Epoch 4/5
540/540 - 5s - loss: 0.0499 - accuracy: 0.9839 - val_loss: 0.0701 - val_accuracy: 0.9790
Epoch 5/5
540/540 - 5s - loss: 0.0408 - accuracy: 0.9868 - val_loss: 0.0599 - val_accuracy: 0.9820
<keras.callbacks.History at 0x7f0a127e4a10>

▶ end = time.time()
  print(end - start)

30.371798276901245
```

Figura 3: Capa extra.

1. ¿Cómo cambia la precisión de validación?

Solución. Comparado al caso anterior, hubo una disminución del 0.2% en la precisión del algoritmo. ☐

2. ¿Qué hay del tiempo que se tarda en ejecutar? Pista: deben tener cuidado con las formas de los pesos y los sesgos.

Solución. El tiempo es prácticamente el mismo; un poco mayor por centésimas de segundo. ☐

Problema 3. El ancho y la profundidad del algoritmo. Agregue cuantas capas sean necesarias para llegar a 5 capas escondidas. Es más, ajusten el ancho del algoritmo conforme lo encuentre más conveniente.

```
Epoch 1/5
540/540 - 11s - loss: 0.2690 - accuracy: 0.9179 - val_loss: 0.1566 - val_accuracy: 0.9523
Epoch 2/5
540/540 - 6s - loss: 0.1104 - accuracy: 0.9659 - val_loss: 0.1006 - val_accuracy: 0.9705
Epoch 3/5
540/540 - 6s - loss: 0.0747 - accuracy: 0.9766 - val_loss: 0.0960 - val_accuracy: 0.9697
Epoch 4/5
540/540 - 6s - loss: 0.0610 - accuracy: 0.9811 - val_loss: 0.0681 - val_accuracy: 0.9798
Epoch 5/5
540/540 - 6s - loss: 0.0502 - accuracy: 0.9849 - val_loss: 0.0682 - val_accuracy: 0.9813
<keras.callbacks.History at 0x7f8e25a8d750>

24] end = time.time()
    print(end - start)

51.91294002532959
```

Figura 4: 5 capas.

1. ¿Cómo cambia la precisión de validación?

Solución. Es levemente peor al caso anterior; lo que quiere decir que agregar capas no hace al modelo más efectivo. ☐

2. ¿Qué hay del tiempo de ejecución?

Solución. Llega a casi 52 segundos; por lo cual no es tan eficiente comparado a los casos anteriores. \square

Problema 4. Experimenten con las funciones de activación. Intenten aplicar una transformación sigmoidal a ambas capas. La activación sigmoidal se obtiene escribiendo “sigmoid”.

Solución. Los resultados aparentan ser levemente peores y el tiempo no tiene ningún cambio.

```
Epoch 1/5
540/540 - 9s - loss: 0.5800 - accuracy: 0.8453 - val_loss: 0.2688 - val_accuracy: 0.9232
Epoch 2/5
540/540 - 5s - loss: 0.2247 - accuracy: 0.9335 - val_loss: 0.1975 - val_accuracy: 0.9415
Epoch 3/5
540/540 - 5s - loss: 0.1671 - accuracy: 0.9506 - val_loss: 0.1613 - val_accuracy: 0.9532
Epoch 4/5
540/540 - 5s - loss: 0.1318 - accuracy: 0.9609 - val_loss: 0.1239 - val_accuracy: 0.9653
Epoch 5/5
540/540 - 5s - loss: 0.1039 - accuracy: 0.9695 - val_loss: 0.1030 - val_accuracy: 0.9698
<keras.callbacks.History at 0x7f78d3696350>

[24] end = time.time()
print(end - start)

30.118937015533447
```

Figura 5: Función «sigmoid».

Problema 5. Continúen experimentando con las funciones de activación. Intenten aplicar un ReLu a la primera capa escondida y tanh a la segunda. La activación tanh se obtiene escribiendo “tanh”.

```
Epoch 1/5
540/540 - 9s - loss: 0.2550 - accuracy: 0.9246 - val_loss: 0.1175 - val_accuracy: 0.9640
Epoch 2/5
540/540 - 5s - loss: 0.0956 - accuracy: 0.9712 - val_loss: 0.0907 - val_accuracy: 0.9725
Epoch 3/5
540/540 - 5s - loss: 0.0649 - accuracy: 0.9796 - val_loss: 0.0622 - val_accuracy: 0.9828
Epoch 4/5
540/540 - 5s - loss: 0.0482 - accuracy: 0.9851 - val_loss: 0.0560 - val_accuracy: 0.9840
Epoch 5/5
540/540 - 5s - loss: 0.0365 - accuracy: 0.9885 - val_loss: 0.0438 - val_accuracy: 0.9887
<keras.callbacks.History at 0x7f67ccf68990>

▶ end = time.time()
print(end - start)

🌐 30.308727741241455
```

Figura 6: Activación ReLu y tanh.

Solución. En cuestiones de eficiencia; el tiempo es similar a los demás casos. Por otra parte, la precisión de la validación es la más alta de todos los casos con 98.87% de precisión. \square

Problema 6. Ajusten el tamaño de la tanda. Prueben con un tamaño de tanda de 10,000.

```

Epoch 1/5
6/6 - 8s - loss: 1.8323 - accuracy: 0.4910 - val_loss: 1.1630 - val_accuracy: 0.7562
Epoch 2/5
6/6 - 3s - loss: 0.9392 - accuracy: 0.7941 - val_loss: 0.6457 - val_accuracy: 0.8425
Epoch 3/5
6/6 - 3s - loss: 0.5687 - accuracy: 0.8518 - val_loss: 0.4509 - val_accuracy: 0.8705
Epoch 4/5
6/6 - 3s - loss: 0.4243 - accuracy: 0.8792 - val_loss: 0.3669 - val_accuracy: 0.8905
Epoch 5/5
6/6 - 3s - loss: 0.3519 - accuracy: 0.8976 - val_loss: 0.3216 - val_accuracy: 0.9043
<keras.callbacks.History at 0x7f1b84ba3990>

[24] end = time.time()
print(end - start)

22.497343063354492

```

Figura 7: Tamaño de tanda de 10,000.

1. ¿Cómo cambia el tiempo requerido?

Solución. El tiempo es de 22.50 segundos, siendo el caso más eficiente. ☐

2. ¿Cómo cambia la precisión?

Solución. La precisión es la más baja hasta ahora; con un 90 % de precisión. ☐

Problema 7. Ajusten el tamaño de la tanda a 1. Eso corresponde al SGD.

```

Epoch 1/5
54000/54000 - 101s - loss: 0.2447 - accuracy: 0.9280 - val_loss: 0.1465 - val_accuracy: 0.9610
Epoch 2/5
54000/54000 - 95s - loss: 0.1382 - accuracy: 0.9628 - val_loss: 0.1218 - val_accuracy: 0.9658
Epoch 3/5
54000/54000 - 94s - loss: 0.1194 - accuracy: 0.9676 - val_loss: 0.0985 - val_accuracy: 0.9727
Epoch 4/5
54000/54000 - 103s - loss: 0.1031 - accuracy: 0.9715 - val_loss: 0.1192 - val_accuracy: 0.9692
Epoch 5/5
54000/54000 - 96s - loss: 0.1049 - accuracy: 0.9714 - val_loss: 0.0949 - val_accuracy: 0.9743
<keras.callbacks.History at 0x7f9af9a5a6d0>

[24] end = time.time()
print(end - start)

670.8687813282013

```

Figura 8: Tamaño de la tanda 1.

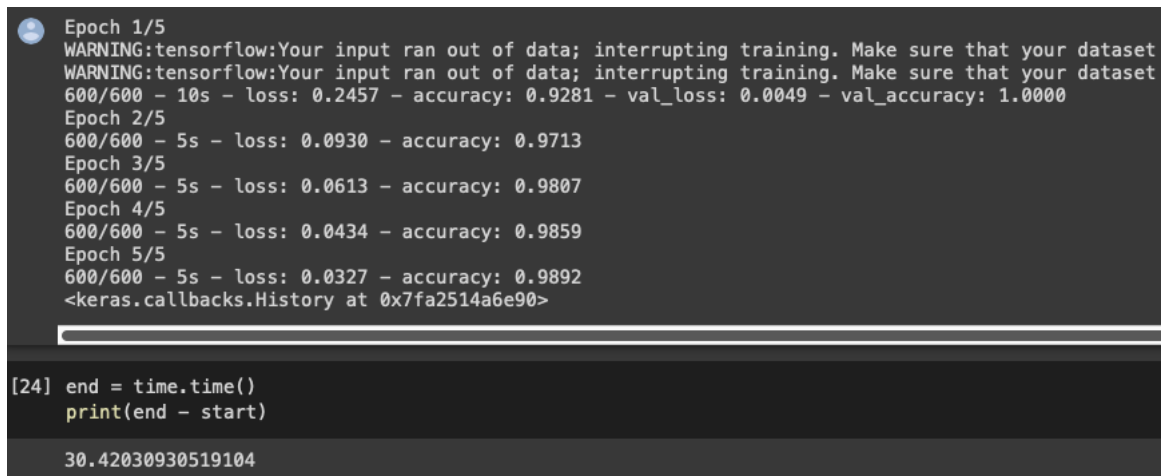
1. ¿Cómo cambian el tiempo y la precisión?

Solución. El tiempo se dispara drásticamente (671 segundos); aunque la precisión es bastante normal, alrededor de 97%. ☐

2. ¿Es el resultado coherente con la teoría?

Solución. En efecto, la teoría se cumple; ya que el descenso del gradiente está al principio y por eso se tarda un largo rato en cargar el modelo. ☐

Problema 8. *Ajusten la tasa de aprendizaje. Prueben con un valor de 0.0001. ¿Hace alguna diferencia?*



```
Epoch 1/5
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset
600/600 - 10s - loss: 0.2457 - accuracy: 0.9281 - val_loss: 0.0049 - val_accuracy: 1.0000
Epoch 2/5
600/600 - 5s - loss: 0.0930 - accuracy: 0.9713
Epoch 3/5
600/600 - 5s - loss: 0.0613 - accuracy: 0.9807
Epoch 4/5
600/600 - 5s - loss: 0.0434 - accuracy: 0.9859
Epoch 5/5
600/600 - 5s - loss: 0.0327 - accuracy: 0.9892
<keras.callbacks.History at 0x7fa2514a6e90>

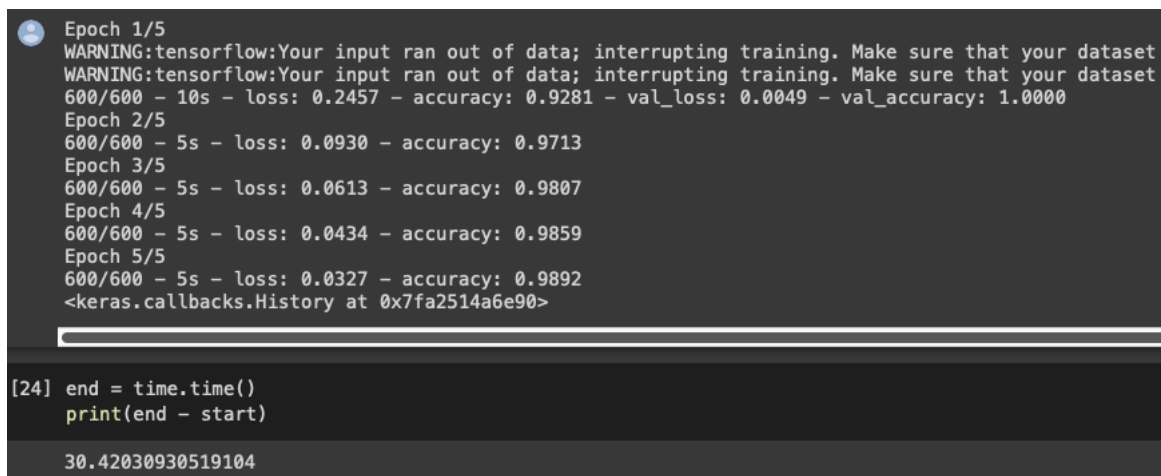
[24] end = time.time()
print(end - start)

30.42030930519104
```

Figura 9: Valor de 0.0001.

Solución. El tiempo es básicamente promedio; mientras que la precisión llega a 98.92 % de precisión; aunque la librería lanza un error respecto a posibles datos erróneos. ☐

Problema 9. *Ajusten la tasa de aprendizaje a 0.02. ¿Hay alguna diferencia?*



```
Epoch 1/5
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset
600/600 - 10s - loss: 0.2457 - accuracy: 0.9281 - val_loss: 0.0049 - val_accuracy: 1.0000
Epoch 2/5
600/600 - 5s - loss: 0.0930 - accuracy: 0.9713
Epoch 3/5
600/600 - 5s - loss: 0.0613 - accuracy: 0.9807
Epoch 4/5
600/600 - 5s - loss: 0.0434 - accuracy: 0.9859
Epoch 5/5
600/600 - 5s - loss: 0.0327 - accuracy: 0.9892
<keras.callbacks.History at 0x7fa2514a6e90>

[24] end = time.time()
print(end - start)

30.42030930519104
```

Figura 10: Valor de 0.02.

Solución. Hasta el momento, esta elección representa el modelo más preciso con 99.33 % y 30.42 segundos. ☐

Problema 10. *Combinen todos los métodos indicados arriba e intenten llegar a una precisión de validación de 98.5 % o más.*

Solución. Las siguientes opciones son las mejores para obtener una precisión de 99.33 %:

1. Dos capas:
 - a) Primera capa con inicialización ReLu.
 - b) Segunda capa con inicialización tanh.

2. Tasa de aprendizaje de 0.02.
3. Tamaño de tanda: 100.
4. Tamaño de capa escondida: 200.

□