Alumno: Jorge Luis Toral Gamez

Repositorio: https://github.com/Jorge-1501/Redes-Neuronales.git

Ingreso a Drive y GitHub

[ ] ↳ *9 celdas ocultas*

Inicio de la Tarea 3

Parte 1

Red con Keras

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import RMSprop


# Cargar el conjunto de datos MNIST
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()


# Reestructuración los datos
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)


# Creación del modelo [784, 30, 10]
model = models.Sequential()
model.add(layers.Dense(784, activation='sigmoid', input_shape=(28 * 28,)))
model.add(layers.Dense(30, activation='sigmoid', input_shape=(28 * 28,)))
model.add(layers.Dense(10, activation='softmax'))


# Compilar el modelo
```

```
# Compilar el modelo
custom_optimizer = RMSprop(learning_rate=0.001, rho=0.9, epsilon=1e-09)

model.compile(optimizer = custom_optimizer,
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'])


# Entrenar el modelo
history = model.fit(train_images, train_labels, epochs=30, batch_size=10, validati
```

```
Epoch 1/30
4800/4800 [==============================] - 22s 4ms/step - loss: 0.3658 - ac
Epoch 2/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.1560 - ac
Epoch 3/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.1088 - ac
Epoch 4/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0855 - ac
Epoch 5/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0691 - ac
Epoch 6/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0560 - ac
Epoch 7/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0491 - ac
Epoch 8/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0415 - ac
Epoch 9/30
4800/4800 [==============================] - 17s 3ms/step - loss: 0.0363 - ac
Epoch 10/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0302 - ac
Epoch 11/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0266 - ac
Epoch 12/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0229 - ac
Epoch 13/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0197 - ac
Epoch 14/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0170 - ac
Epoch 15/30
4800/4800 [==============================] - 16s 3ms/step - loss: 0.0141 - ac
Epoch 16/30
4800/4800 [==============================] - 16s 3ms/step - loss: 0.0135 - ac
Epoch 17/30
4800/4800 [==============================] - 16s 3ms/step - loss: 0.0111 - ac
Epoch 18/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0099 - ac
Epoch 19/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0083 - ac
Epoch 20/30
4800/4800 [==============================] - 16s 3ms/step - loss: 0.0074 - ac
Epoch 21/30
4800/4800 [==============================] - 17s 3ms/step - loss: 0.0066 - ac
Epoch 22/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0056 - ac
```

```
Epoch 23/30
4800/4800 [==============================] - 24s 5ms/step - loss: 0.0046 - ac
Epoch 24/30
4800/4800 [==============================] - 20s 4ms/step - loss: 0.0038 - ac
Epoch 25/30
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0033 - ac
Epoch 26/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0033 - ac
Epoch 27/30
4800/4800 [==============================] - 16s 3ms/step - loss: 0.0020 - ac
Epoch 28/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0020 - ac
Epoch 29/30
4800/4800 [==============================] - 17s 3ms/step - loss: 0.0017 - ac
```

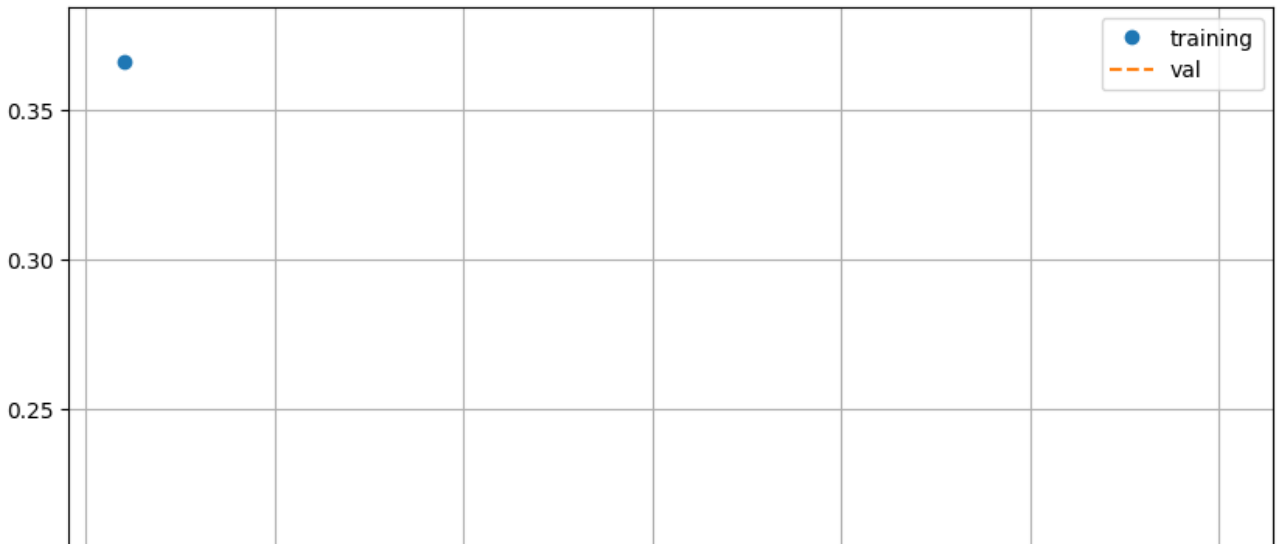
```
# Evaluación el modelo en el conjunto de prueba
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Precisión en el conjunto de prueba:', test_acc)
```

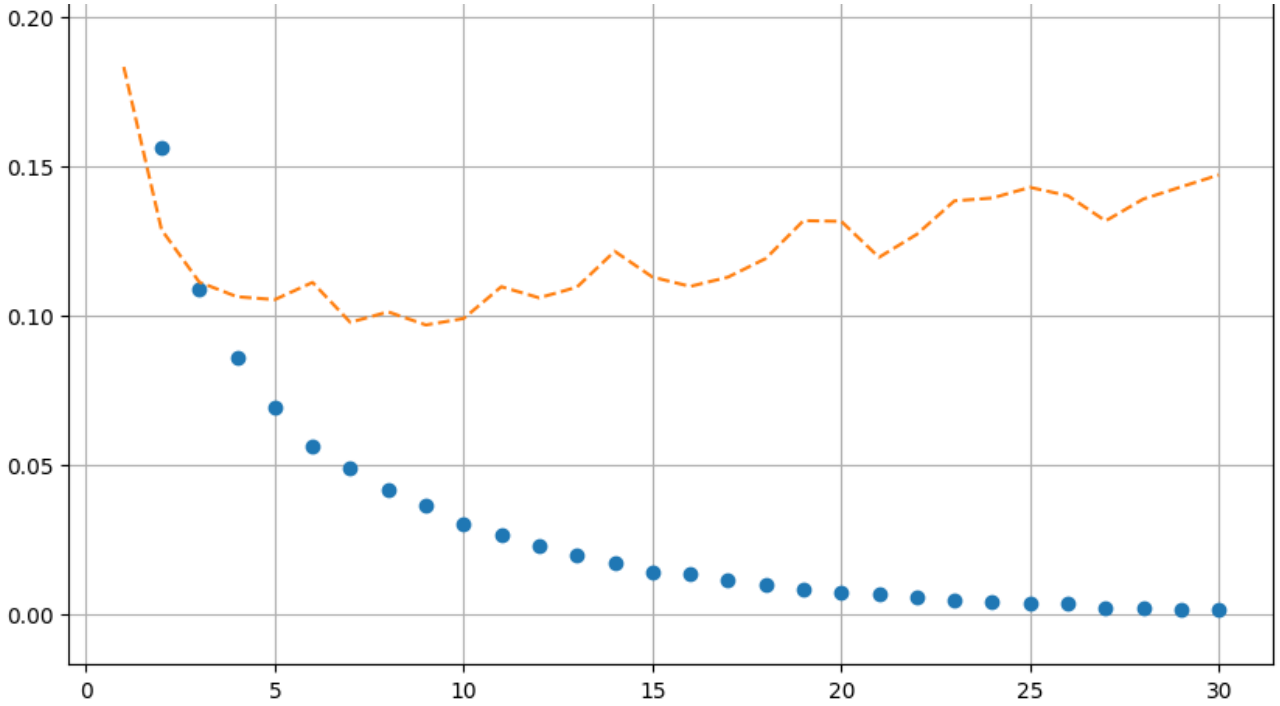
```
313/313 [==============================] - 1s 4ms/step - loss: 0.1202 - accur
Precisión en el conjunto de prueba: 0.9818999767303467
```

```
import matplotlib.pyplot as plt


history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']

fig = plt.figure(figsize=(10,10))
epoch = range(1,len(loss_values)+1)
plt.plot(epoch,loss_values, 'o',label='training')
plt.plot(epoch,val_loss_values, '--',label='val')
plt.legend()
plt.grid()
plt.show()
```





¿Obtuviste resultados similares? No, fueron notablemente mejores. Se obtuvo un accurancy del 0.9815 contra un 0.8412 del anterior modelo utilizando CPU. Mientras que al cambiar el entorno a una T4 GPU el tiempo se redujo a 9 minutos y un accurancy de 0.9818. El sobreajuste se mantiene igual en cada caso.

¿Tardó lo mismo para entrenar el mismo número de epocas? Tardó menos, un aproximado de 21 minutos contra los casi 30 anteriormente.

```
!git status
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Tarea3-Redes.ipynb

no changes added to commit (use "git add" and/or "git commit -a")
```

```
!git status
!git add .
!git commit -m "Modificación de entorno y corrección de número de capas en parte 1
!git push
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Tarea3-Redes.ipynb

no changes added to commit (use "git add" and/or "git commit -a")
[main 1e1f601] Modificación de entorno y corrección de número de capas en par
 1 file changed, 1 insertion(+), 1 deletion(-)
 rewrite Digitos/src/Tarea3-Redes.ipynb (98%)
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 29.68 KiB | 2.97 MiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Jorge-1501/Redes-Neuronales
   fb88598..1e1f601  main -> main
```

Parte 2

Modificacion de parámetros 1

```
#import tensorflow as tf
#import matplotlib.pyplot as plt
#from tensorflow.keras import layers, models
#from tensorflow.keras.datasets import mnist
#from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam


# Cargar el conjunto de datos MNIST
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
11490434/11490434 [==============================] - 2s 0us/step
```

```
# Reestructuración los datos
train_images = train_images.reshape((60000, 28 * 28))
```

```
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)


# Creación del modelo
model2 = models.Sequential()
model2.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
model2.add(layers.Dense(30, activation='relu', input_shape=(28 * 28,)))
model2.add(layers.Dense(10, activation='softmax'))


# Compilar el modelo
custom_optimizer2 = Adam(learning_rate=0.0001, beta_1=0.9, beta_2=0.999, epsilon=1

model2.compile(optimizer = custom_optimizer2,
               loss = 'categorical_crossentropy',
               metrics = ['accuracy'])


# Entrenar el modelo
history2 = model2.fit(train_images, train_labels, epochs=50, batch_size=10, valida
```

```
Epoch 1/50
4800/4800 [==============================] - 24s 5ms/step - loss: 0.3773 - ac
Epoch 2/50
4800/4800 [==============================] - 17s 4ms/step - loss: 0.1574 - ac
Epoch 3/50
4800/4800 [==============================] - 17s 4ms/step - loss: 0.1097 - ac
Epoch 4/50
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0826 - ac
Epoch 5/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0655 - ac
Epoch 6/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0514 - ac
Epoch 7/50
4800/4800 [==============================] - 17s 3ms/step - loss: 0.0413 - ac
Epoch 8/50
4800/4800 [==============================] - 21s 4ms/step - loss: 0.0329 - ac
Epoch 9/50
4800/4800 [==============================] - 17s 3ms/step - loss: 0.0262 - ac
Epoch 10/50
4800/4800 [==============================] - 17s 3ms/step - loss: 0.0212 - ac
Epoch 11/50
4800/4800 [==============================] - 20s 4ms/step - loss: 0.0171 - ac
Epoch 12/50
4800/4800 [==============================] - 24s 5ms/step - loss: 0.0137 - ac
Epoch 13/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0108 - ac
Epoch 14/50
```

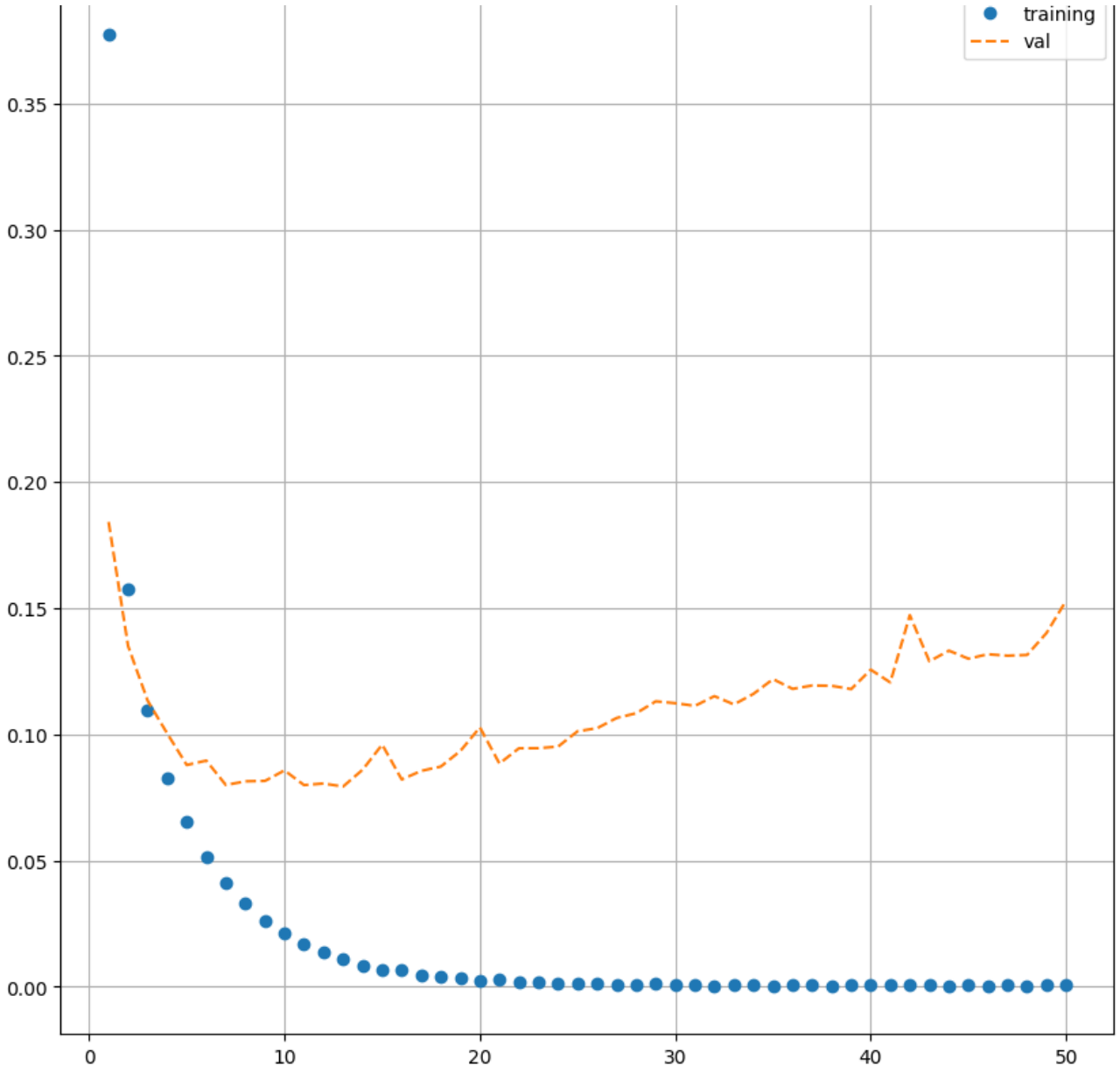
```
Epoch 14/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0084 - ac
Epoch 15/50
4800/4800 [==============================] - 17s 3ms/step - loss: 0.0069 - ac
Epoch 16/50
4800/4800 [==============================] - 20s 4ms/step - loss: 0.0064 - ac
Epoch 17/50
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0047 - ac
Epoch 18/50
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0039 - ac
Epoch 19/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0034 - ac
Epoch 20/50
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0023 - ac
Epoch 21/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0028 - ac
Epoch 22/50
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0019 - ac
Epoch 23/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0019 - ac
Epoch 24/50
4800/4800 [==============================] - 17s 3ms/step - loss: 0.0015 - ac
Epoch 25/50
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0014 - ac
Epoch 26/50
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0012 - ac
Epoch 27/50
4800/4800 [==============================] - 20s 4ms/step - loss: 8.3067e-04
Epoch 28/50
4800/4800 [==============================] - 17s 3ms/step - loss: 9.3600e-04
Epoch 29/50
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0012 - ac
```

```
# Evaluación el modelo en el conjunto de prueba
test_loss2, test_acc2 = model2.evaluate(test_images, test_labels)
print('Precisión en el conjunto de prueba:', test_acc2)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.1383 - accur
Precisión en el conjunto de prueba: 0.9797999858856201
```

```
history_dict2 = history2.history
loss_values2 = history_dict2['loss']
val_loss_values2 = history_dict2['val_loss']

fig = plt.figure(figsize=(10,10))
epoch = range(1,len(loss_values2)+1)
plt.plot(epoch,loss_values2, 'o',label='training')
plt.plot(epoch,val_loss_values2, '--',label='val')
plt.legend()
plt.grid()
plt.show()
```



En esta red seleccioné un activación 'relu' para las capas ocultas y una softmax para la última capa. Aumenté el número de epocas por 50 y trabajé con una $\eta = 0.0001$ y Modifiqué el número de neuronas utilizadas de ([784,30,10]) por ([512,30,10]). Los resultados fueron un mayor sobreajuste y saltos en la convergencia final, por estar muy cerca del accurancy de 1.000.

```
!git status
!git add .
!git commit -m "Cambio de función de activación, número de neuronas y optimizador,
!git push
```

```
Refresh index: 100% (96/96), done.
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Tarea3-Redes.ipynb

no changes added to commit (use "git add" and/or "git commit -a")
[main f050d97] Cambio de función de activación, número de neuronas y optimiza
 1 file changed, 1 insertion(+), 1 deletion(-)
 rewrite Digitos/src/Tarea3-Redes.ipynb (97%)
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 30.74 KiB | 2.56 MiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Jorge-1501/Redes-Neuronales
   1e1f601..f050d97  main -> main
```

Modificacion de parámetros 2

```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam


# Cargar el conjunto de datos MNIST
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()


# Reestructuración los datos
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
```

```
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)


# Creación del modelo
model3 = models.Sequential()
model3.add(layers.Dense(784, activation='relu', input_shape=(28 * 28,)))
#model3.add(layers.Dense(128, activation='relu', input_shape=(28 * 28,)))
model3.add(layers.Dense(24, activation='relu', input_shape=(28 * 28,)))
model3.add(layers.Dense(10, activation='softmax'))


# Compilar el modelo
custom_optimizer3 = Adam(learning_rate=0.00001, beta_1=0.95, beta_2=0.999, epsilon

model3.compile(optimizer = custom_optimizer3,
               loss = 'categorical_crossentropy',
               metrics = ['accuracy'])


# Entrenar el modelo
history3 = model3.fit(train_images, train_labels, epochs=30, batch_size=10, valida
```

```
Epoch 1/30
4800/4800 [==============================] - 23s 4ms/step - loss: 1.0514 - ac
Epoch 2/30
4800/4800 [==============================] - 20s 4ms/step - loss: 0.4117 - ac
Epoch 3/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.3162 - ac
Epoch 4/30
4800/4800 [==============================] - 19s 4ms/step - loss: 0.2719 - ac
Epoch 5/30
4800/4800 [==============================] - 19s 4ms/step - loss: 0.2429 - ac
Epoch 6/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.2213 - ac
Epoch 7/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.2040 - ac
Epoch 8/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1895 - ac
Epoch 9/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1770 - ac
Epoch 10/30
4800/4800 [==============================] - 19s 4ms/step - loss: 0.1663 - ac
Epoch 11/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1564 - ac
Epoch 12/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1476 - ac
Epoch 13/30
4800/4800 [==============================] - 19s 4ms/step - loss: 0.1398 - ac
Epoch 14/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1328 - ac
Epoch 15/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.1259 - ac
Epoch 16/30
4800/4800 [==============================] - 20s 4ms/step - loss: 0.1199 - ac
```

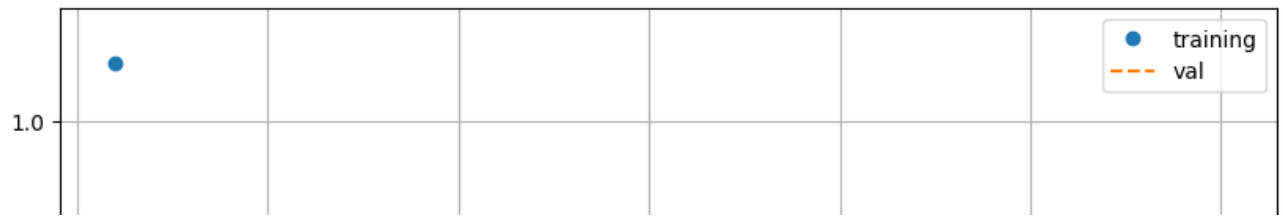
```
Epoch 17/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1143 - ac
Epoch 18/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1089 - ac
Epoch 19/30
4800/4800 [==============================] - 19s 4ms/step - loss: 0.1039 - ac
Epoch 20/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0993 - ac
Epoch 21/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0950 - ac
Epoch 22/30
4800/4800 [==============================] - 20s 4ms/step - loss: 0.0909 - ac
Epoch 23/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0873 - ac
Epoch 24/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0835 - ac
Epoch 25/30
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0800 - ac
Epoch 26/30
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0769 - ac
Epoch 27/30
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0738 - ac
Epoch 28/30
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0708 - ac
Epoch 29/30
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0679 - ac
```

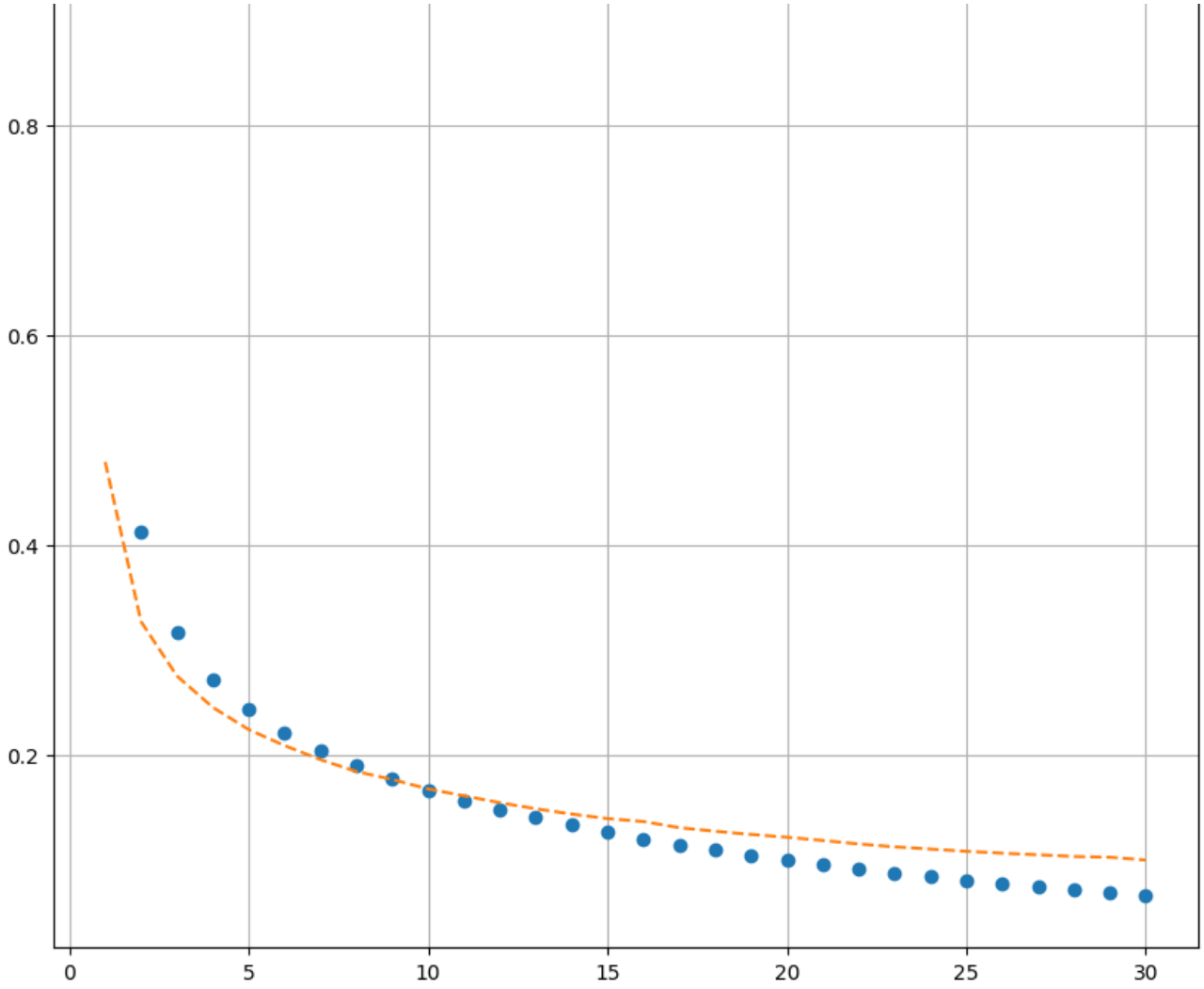
```
# Evaluación el modelo en el conjunto de prueba
test_loss3, test_acc3 = model3.evaluate(test_images, test_labels)
print('Precisión en el conjunto de prueba:', test_acc3)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0943 - accur
Precisión en el conjunto de prueba: 0.9721999764442444
```

```
history_dict3 = history3.history
loss_values3 = history_dict3['loss']
val_loss_values3 = history_dict3['val_loss']

fig = plt.figure(figsize=(10,10))
epoch = range(1,len(loss_values3)+1)
plt.plot(epoch,loss_values3, 'o',label='training')
plt.plot(epoch,val_loss_values3, '--',label='val')
plt.legend()
plt.grid()
plt.show()
```





En esta red seleccioné un activación 'relu' para las capas ocultas y una softmax para la última capa. Redujé el número de epocas a 30 y trabajé con una $\eta = 0.0001$ y Modifiqué el número de neuronas utilizadas de ([512,30,10]) por ([784,24,10]). También modifiqué los $\beta' s$ para una aprendizaje más lento.

Los resultados no mostraron sobreajuste, pero la tasa de aprendizaje fue más lenta y obtuvo un menor accurancy, 0.9721 respecto a un 0.9797 de la anterior configuración.

```
!git status
!git add .
```

```
!git commit -m "Cambio de parámetros y número de neuronas. No hay sobreajuste pero
!git push
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Tarea3-Redes.ipynb

no changes added to commit (use "git add" and/or "git commit -a")
[main 813279b] Cambio de parámetros y número de neuronas. No hay sobreajuste
 1 file changed, 1 insertion(+), 1 deletion(-)
 rewrite Digitos/src/Tarea3-Redes.ipynb (97%)
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 24.29 KiB | 2.70 MiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Jorge-1501/Redes-Neuronales
   f050d97..813279b  main -> main
```

Tercera modificación

```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam


# Cargar el conjunto de datos MNIST
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()


# Reestructuración los datos
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)


# Creación del modelo
```

```
model4 = models.Sequential()
model4.add(layers.Dense(784, activation='relu', input_shape=(28 * 28,)))
#model3.add(layers.Dense(128, activation='relu', input_shape=(28 * 28,)))
model4.add(layers.Dense(24, activation='relu', input_shape=(28 * 28,)))
model4.add(layers.Dense(10, activation='softmax'))


# Compilar el modelo
custom_optimizer4 = Adam(learning_rate=0.000005, beta_1=0.97, beta_2=0.999, epsilo

model4.compile(optimizer = custom_optimizer4,
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'])


# Entrenar el modelo
history4 = model4.fit(train_images, train_labels, epochs=50, batch_size=10, valida
```

```
Epoch 1/50
4800/4800 [==============================] - 20s 4ms/step - loss: 1.3797 - ac
Epoch 2/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.6556 - ac
Epoch 3/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.4726 - ac
Epoch 4/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.3921 - ac
Epoch 5/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.3448 - ac
Epoch 6/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.3122 - ac
Epoch 7/50
4800/4800 [==============================] - 21s 4ms/step - loss: 0.2878 - ac
Epoch 8/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.2683 - ac
Epoch 9/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.2520 - ac
Epoch 10/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.2382 - ac
Epoch 11/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.2265 - ac
Epoch 12/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.2159 - ac
Epoch 13/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.2066 - ac
Epoch 14/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.1980 - ac
Epoch 15/50
4800/4800 [==============================] - 21s 4ms/step - loss: 0.1903 - ac
Epoch 16/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1834 - ac
Epoch 17/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1767 - ac
Epoch 18/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.1708 - ac
```

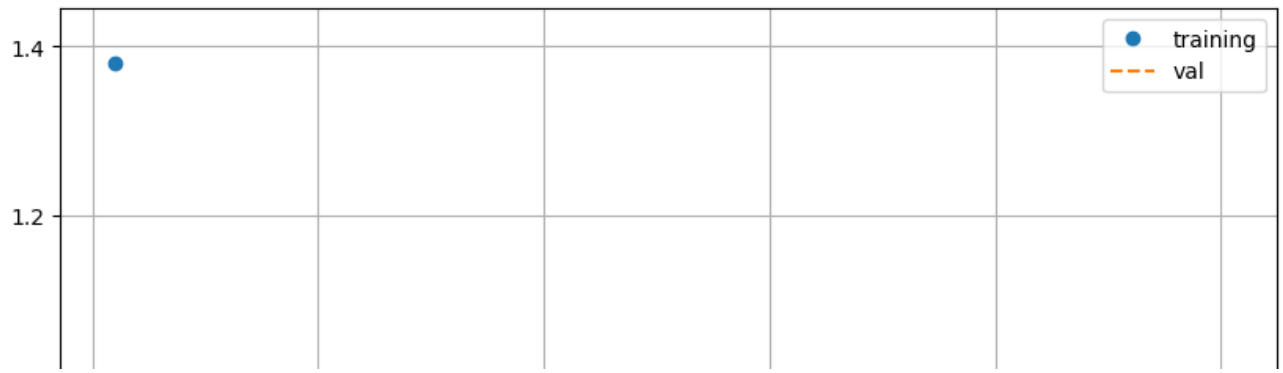
```
Epoch 19/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.1651 - ac
Epoch 20/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1599 - ac
Epoch 21/50
4800/4800 [==============================] - 20s 4ms/step - loss: 0.1550 - ac
Epoch 22/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.1503 - ac
Epoch 23/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.1461 - ac
Epoch 24/50
4800/4800 [==============================] - 20s 4ms/step - loss: 0.1419 - ac
Epoch 25/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.1381 - ac
Epoch 26/50
4800/4800 [==============================] - 19s 4ms/step - loss: 0.1343 - ac
Epoch 27/50
4800/4800 [==============================] - 20s 4ms/step - loss: 0.1308 - ac
Epoch 28/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1274 - ac
Epoch 29/50
4800/4800 [==============================] - 18s 4ms/step - loss: 0.1243 - ac
```

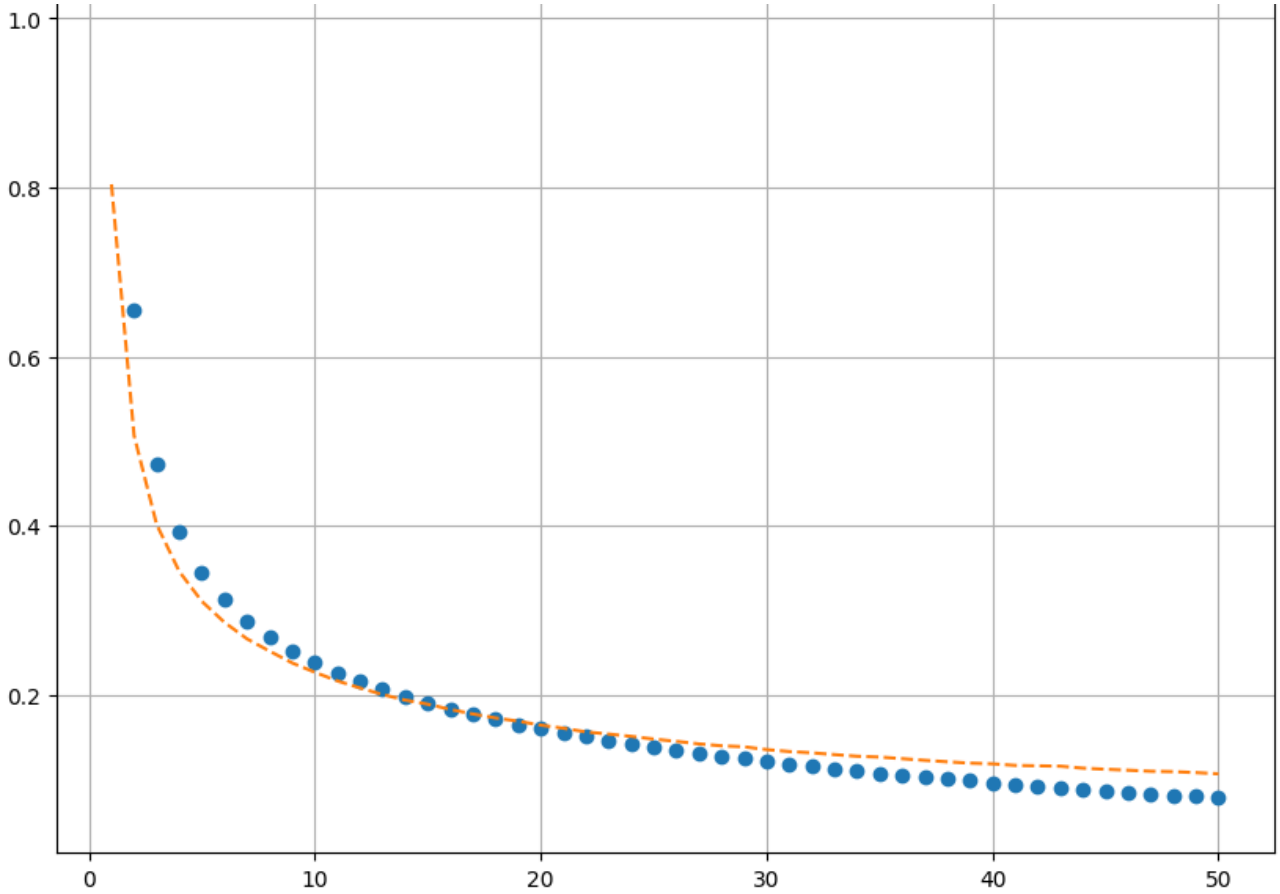
```
# Evaluación el modelo en el conjunto de prueba
test_loss4, test_acc4 = model4.evaluate(test_images, test_labels)
print('Precisión en el conjunto de prueba:', test_acc4)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.1016 - accur
Precisión en el conjunto de prueba: 0.9697999954223633
```

```
history_dict4 = history4.history
loss_values4 = history_dict4['loss']
val_loss_values4 = history_dict4['val_loss']

fig = plt.figure(figsize=(10,10))
epoch = range(1,len(loss_values4)+1)
plt.plot(epoch,loss_values4, 'o',label='training')
plt.plot(epoch,val_loss_values4, '--',label='val')
plt.legend()
plt.grid()
plt.show()
```





Modificación del modelo 3. aumenté el número de epocas a 50 y trabajé con una $\eta = 0.00005$. También modifiqué los $\beta's$ para una aprendizaje más lento.

Los resultados no mostraron sobreajuste, pero la tasa de aprendizaje fue más lenta y obtuvo un menor accurancy, 0.9697 respecto a un 0.9721 de la anterior configuración.

```
!git status
!git add .
!git commit -m "Modelo 4: cambio de parámetros del modelo 3 e incremento de épocas
!git push
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
```

```
	(use "git add <file>..." to update what will be committed)
	(use "git restore <file>..." to discard changes in working directory)
	      modified:   Tarea3-Redes.ipynb

no changes added to commit (use "git add" and/or "git commit -a")
[main 180feef] Modelo 4: cambio de parámetros del modelo 3 e incremento de ép
 1 file changed, 1 insertion(+), 1 deletion(-)
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 27.00 KiB | 3.00 MiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Jorge-1501/Redes-Neuronales
   813279b..180feef  main -> main
```

Guardado de modelos

```
# Guardar el primer modelo
model3.save("modelo3.h5")

# Guardar el segundo modelo con un nombre diferente
model4.save("modelo4.h5")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3000: Us
  saving_api.save_model(
```

```
!git status
!git add .
!git commit -m "Guardado de los 2 mejores modelos sin sobreajuste"
!git push
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
	(use "git add <file>..." to update what will be committed)
	(use "git restore <file>..." to discard changes in working directory)
	      modified:   Tarea3-Redes.ipynb

Untracked files:
	(use "git add <file>..." to include in what will be committed)
	      modelo3.h5
	      modelo4.h5

no changes added to commit (use "git add" and/or "git commit -a")
[main d72a956] Guardado de los 2 mejores modelos sin sobreajuste
 3 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 Digitos/src/modelo3.h5
 create mode 100644 Digitos/src/modelo4.h5
```

```
 create mode 100644 Digitos/src/modelo1.h5
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 2 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 8.55 MiB | 6.90 MiB/s, done.
Total 7 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Jorge-1501/Redes-Neuronales
   180feef..d72a956  main -> main
```

Parte 3

Implimentación de regularizadores

Entrenamiento hasta sobreajuste

```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam


# Cargar el conjunto de datos MNIST
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()


# Reestructuración los datos
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)


from tensorflow.keras.callbacks import ModelCheckpoint

# Cargar el modelo preentrenado
modelo_preentrenado = tf.keras.models.load_model("modelo3.h5")

# Se define el callback ModelCheckpoint para guardar el mejor modelo en un archivo
checkpoint_mejor_modelo = ModelCheckpoint("mejor_modelo_segundo_entrenamiento.h5",

# Compilación del modelo preentrenado (utilizando el mismo optimizador y función d
```

```
custom_optimizer3 = Adam(learning_rate=0.00001, beta_1=0.95, beta_2=0.999, epsilon

modelo_preentrenado.compile(optimizer=custom_optimizer3, loss="categorical_crossen

# Entrena el modelo preentrenado con el callback para guardar el mejor modelo
history_train2 = modelo_preentrenado.fit(train_images, train_labels, epochs=60, ba
                                         callbacks=[checkpoint_mejor_modelo],
                                         validation_split=0.2)

# Cargar el mejor modelo guardado durante el segundo entrenamiento
mejor_modelo_segundo_entrenamiento = tf.keras.models.load_model("mejor_modelo_segu
```

```
Epoch 1/60
4800/4800 [==============================] - 20s 4ms/step - loss: 0.0630 - ac
Epoch 2/60
  27/4800 [..............................] - ETA: 19s - loss: 0.0693 - accura
  saving_api.save_model(
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0603 - ac
Epoch 3/60
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0581 - ac
Epoch 4/60
4800/4800 [==============================] - 22s 4ms/step - loss: 0.0557 - ac
Epoch 5/60
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0539 - ac
Epoch 6/60
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0517 - ac
Epoch 7/60
4800/4800 [==============================] - 20s 4ms/step - loss: 0.0499 - ac
Epoch 8/60
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0480 - ac
Epoch 9/60
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0461 - ac
Epoch 10/60
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0443 - ac
Epoch 11/60
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0426 - ac
Epoch 12/60
4800/4800 [==============================] - 22s 5ms/step - loss: 0.0411 - ac
Epoch 13/60
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0397 - ac
Epoch 14/60
4800/4800 [==============================] - 21s 4ms/step - loss: 0.0380 - ac
Epoch 15/60
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0367 - ac
Epoch 16/60
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0355 - ac
Epoch 17/60
4800/4800 [==============================] - 20s 4ms/step - loss: 0.0339 - ac
Epoch 18/60
4800/4800 [==============================] - 20s 4ms/step - loss: 0.0326 - ac
Epoch 19/60
4800/4800 [==============================] - 22s 5ms/step - loss: 0.0316 - ac
Epoch 20/60
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0303 - ac
Epoch 21/60
```

```
Epoch 21/60
4800/4800 [==============================] - 22s 5ms/step - loss: 0.0292 - ac
Epoch 22/60
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0280 - ac
Epoch 23/60
4800/4800 [==============================] - 22s 5ms/step - loss: 0.0270 - ac
Epoch 24/60
4800/4800 [==============================] - 19s 4ms/step - loss: 0.0260 - ac
Epoch 25/60
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0249 - ac
Epoch 26/60
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0240 - ac
Epoch 27/60
4800/4800 [==============================] - 17s 4ms/step - loss: 0.0230 - ac
Epoch 28/60
4800/4800 [==============================] - 18s 4ms/step - loss: 0.0222 - ac
```

```
# Evaluación el modelo en el conjunto de prueba
test_lossR3, test_accR3 = mejor_modelo_segundo_entrenamiento.evaluate(test_images,
print('Precisión en el conjunto de prueba:', test_accR3)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0696 - accur
Precisión en el conjunto de prueba: 0.9789999723434448
```

```
history_dictR3 = history_train2.history
loss_valuesR3 = history_dictR3['loss']
val_loss_valuesR3 = history_dictR3['val_loss']

fig = plt.figure(figsize=(10,10))
epoch = range(1,len(loss_valuesR3)+1)
plt.plot(epoch,loss_valuesR3, 'o',label='training')
plt.plot(epoch,val_loss_valuesR3, '--',label='val')
plt.legend()
plt.grid()
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-e42fb1ff7f82> in <cell line: 1>()
----> 1 history_dictR3 = history_train2.history
      2 loss_valuesR3 = history_dictR3['loss']
      3 val_loss_valuesR3 = history_dictR3['val_loss']
      4
      5 fig = plt.figure(figsize=(10,10))

NameError: name 'history_train2' is not defined
```

BUSCAR EN STACK OVERFLOW

```
!git status
!git add .
```

```
!git commit -m "Mejor modelo de 2do entrenamiento"
!git push
```

```
Refresh index: 100% (98/98), done.
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Tarea3-Redes.ipynb

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        mejor_modelo_segundo_entrenamiento.h5

no changes added to commit (use "git add" and/or "git commit -a")
[main 4df50aa] Mejor modelo de 2do entrenamiento
 2 files changed, 1 insertion(+), 1 deletion(-)
 rewrite Digitos/src/Tarea3-Redes.ipynb (91%)
 create mode 100644 Digitos/src/mejor_modelo_segundo_entrenamiento.h5
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 6.29 MiB | 1.19 MiB/s, done.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Jorge-1501/Redes-Neuronales
   d72a956..4df50aa  main -> main
```

L1

```
from tensorflow.keras.layers import Dropout
from tensorflow.keras.regularizers import l1, l2, l1_l2
from tensorflow.keras.callbacks import ModelCheckpoint


# Cargar el modelo preentrenado
modelo_preentrenado_l1 = tf.keras.models.load_model("mejor_modelo_segundo_entrenam

# Agregar capas de dropout y regularización L1
modelo_preentrenado_l1.add(layers.Dense(10, activation='relu', kernel_regularizer=

# Compilar el modelo con las nuevas configuraciones
custom_optimizer3 = Adam(learning_rate=0.00001, beta_1=0.95, beta_2=0.999, epsilon
modelo_preentrenado_l1.compile(optimizer=custom_optimizer3, loss="categorical_cros

# Define el callback ModelCheckpoint para guardar el mejor modelo
checkpoint_mejor_modelo_l1 = ModelCheckpoint("mejor_modelo_l1.h5", monitor="val_lo
```

```
# Entrena el modelo preentrenado con el callback para guardar el mejor modelo
history_train_l1 = modelo_preentrenado_l1.fit(train_images, train_labels, epochs=1
                                              callbacks=[checkpoint_mejor_modelo_l1],
                                              validation_split=0.2)

# Cargar el mejor modelo guardado durante el segundo entrenamiento
mejor_modelo_l1 = tf.keras.models.load_model("mejor_modelo_l1.h5")
```

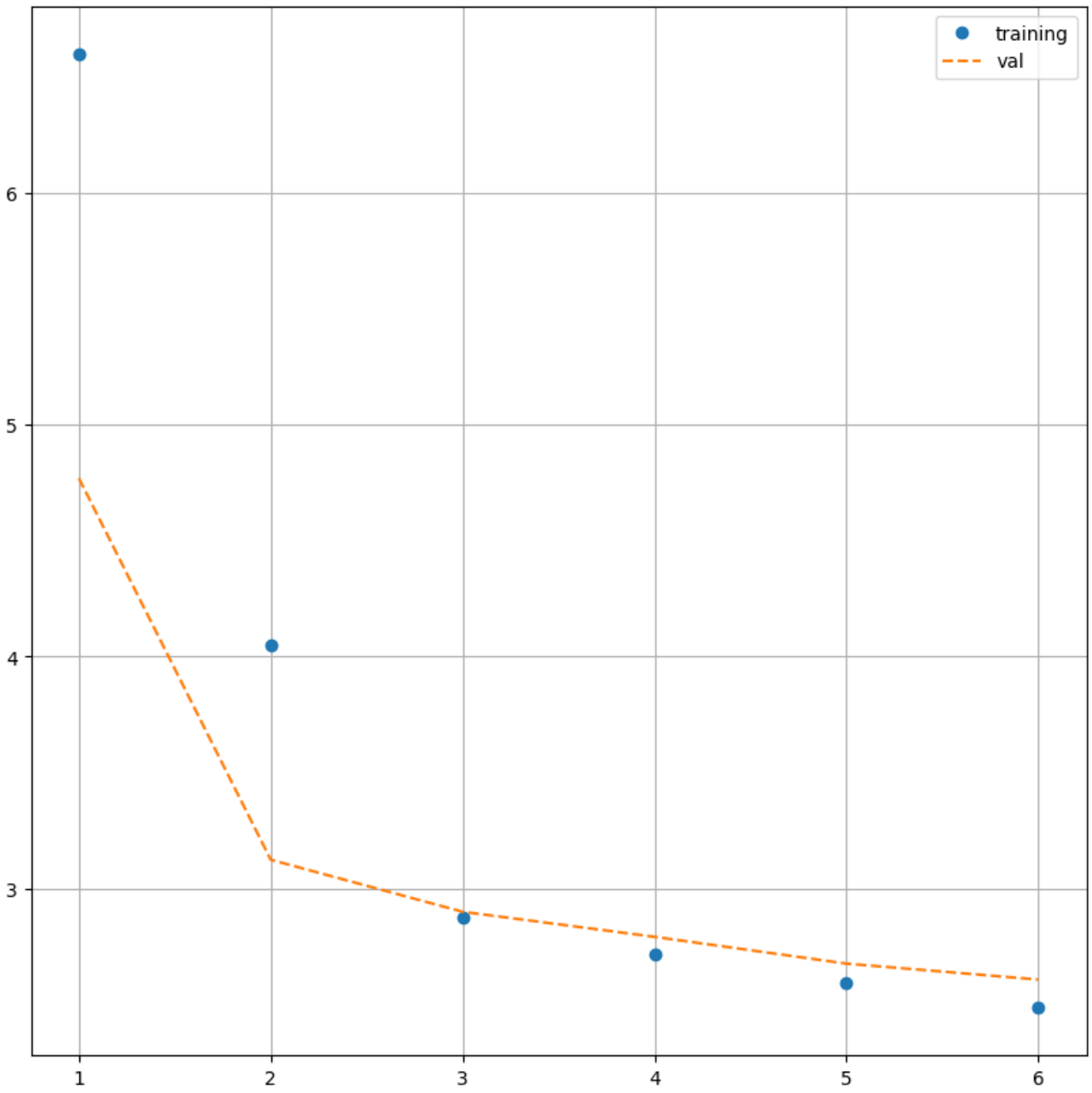
```
Epoch 1/15
4783/4800 [============================>.] - ETA: 0s - loss: 6.6032 - accurac
  saving_api.save_model(
4800/4800 [==============================] - 23s 4ms/step - loss: 6.5968 - ac
Epoch 2/15
4800/4800 [==============================] - 21s 4ms/step - loss: 4.0507 - ac
Epoch 3/15
4800/4800 [==============================] - 19s 4ms/step - loss: 2.8745 - ac
Epoch 4/15
4800/4800 [==============================] - 21s 4ms/step - loss: 2.7147 - ac
Epoch 5/15
4800/4800 [==============================] - 20s 4ms/step - loss: 2.5941 - ac
Epoch 6/15
4800/4800 [==============================] - 20s 4ms/step - loss: 2.4860 - ac
Epoch 7/15
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 8/15
4800/4800 [==============================] - 19s 4ms/step - loss: nan - accur
Epoch 9/15
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 10/15
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 11/15
4800/4800 [==============================] - 19s 4ms/step - loss: nan - accur
Epoch 12/15
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 13/15
4800/4800 [==============================] - 19s 4ms/step - loss: nan - accur
Epoch 14/15
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 15/15
4800/4800 [==============================] - 19s 4ms/step - loss: nan - accur
```

```
# Evaluación el modelo en el conjunto de prueba
test_loss_l1, test_acc_l1 = mejor_modelo_l1.evaluate(test_images, test_labels)
print('Precisión en el conjunto de prueba:', test_acc_l1)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 2.5517 - accur
Precisión en el conjunto de prueba: 0.5515999794006348
```

```
history_dict_l1 = history_train_l1.history
loss_values_l1 = history_dict_l1['loss']
val_loss_values_l1 = history_dict_l1['val_loss']
```

```
fig = plt.figure(figsize=(10,10))
epoch = range(1,len(loss_values_l1)+1)
plt.plot(epoch,loss_values_l1, 'o',label='training')
plt.plot(epoch,val_loss_values_l1, '--',label='val')
plt.legend()
plt.grid()
plt.show()
```



El modelo con la regularización L1 pierde muy rápido los valores de entrenamiento.

```
!git status
!git add .
!git commit -m "Corrección de gráfica e implementación L1. Pérdida de valores dema
!git push
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Tarea3-Redes.ipynb

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        mejor_modelo_l1.h5
        mejor_modelo_l2.h5

no changes added to commit (use "git add" and/or "git commit -a")
[main a8a0e5c] Corrección de gráfica e implementación L1. Pérdida de valores
 3 files changed, 1 insertion(+), 1 deletion(-)
 rewrite Digitos/src/Tarea3-Redes.ipynb (90%)
 create mode 100644 Digitos/src/mejor_modelo_l1.h5
 create mode 100644 Digitos/src/mejor_modelo_l2.h5
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 2 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 12.72 MiB | 10.61 MiB/s, done.
Total 7 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Jorge-1501/Redes-Neuronales
   4df50aa..a8a0e5c  main -> main
```

L2

```
# Cargar el modelo preentrenado
modelo_preentrenado_l2 = tf.keras.models.load_model("mejor_modelo_segundo_entrenam

# Agregar capas de dropout y regularización L2
modelo_preentrenado_l2.add(layers.Dense(10, activation='relu', kernel_regularizer=

# Compilar el modelo con las nuevas configuraciones
custom_optimizer3 = Adam(learning_rate=0.00001, beta_1=0.95, beta_2=0.999, epsilon
modelo_preentrenado_l2.compile(optimizer=custom_optimizer3, loss="categorical_cros

# Define el callback ModelCheckpoint para guardar el mejor modelo
```

```
# Define el callback ModelCheckpoint para guardar el mejor modelo
checkpoint_mejor_modelo_l2 = ModelCheckpoint("mejor_modelo_l2.h5", monitor="val_lo

# Entrena el modelo preentrenado con el callback para guardar el mejor modelo
history_train_l2 = modelo_preentrenado_l2.fit(train_images, train_labels, epochs=1
                                              callbacks=[checkpoint_mejor_modelo_l2],
                                              validation_split=0.2)

# Cargar el mejor modelo guardado durante el segundo entrenamiento
mejor_modelo_l2 = tf.keras.models.load_model("mejor_modelo_l2.h5")
```

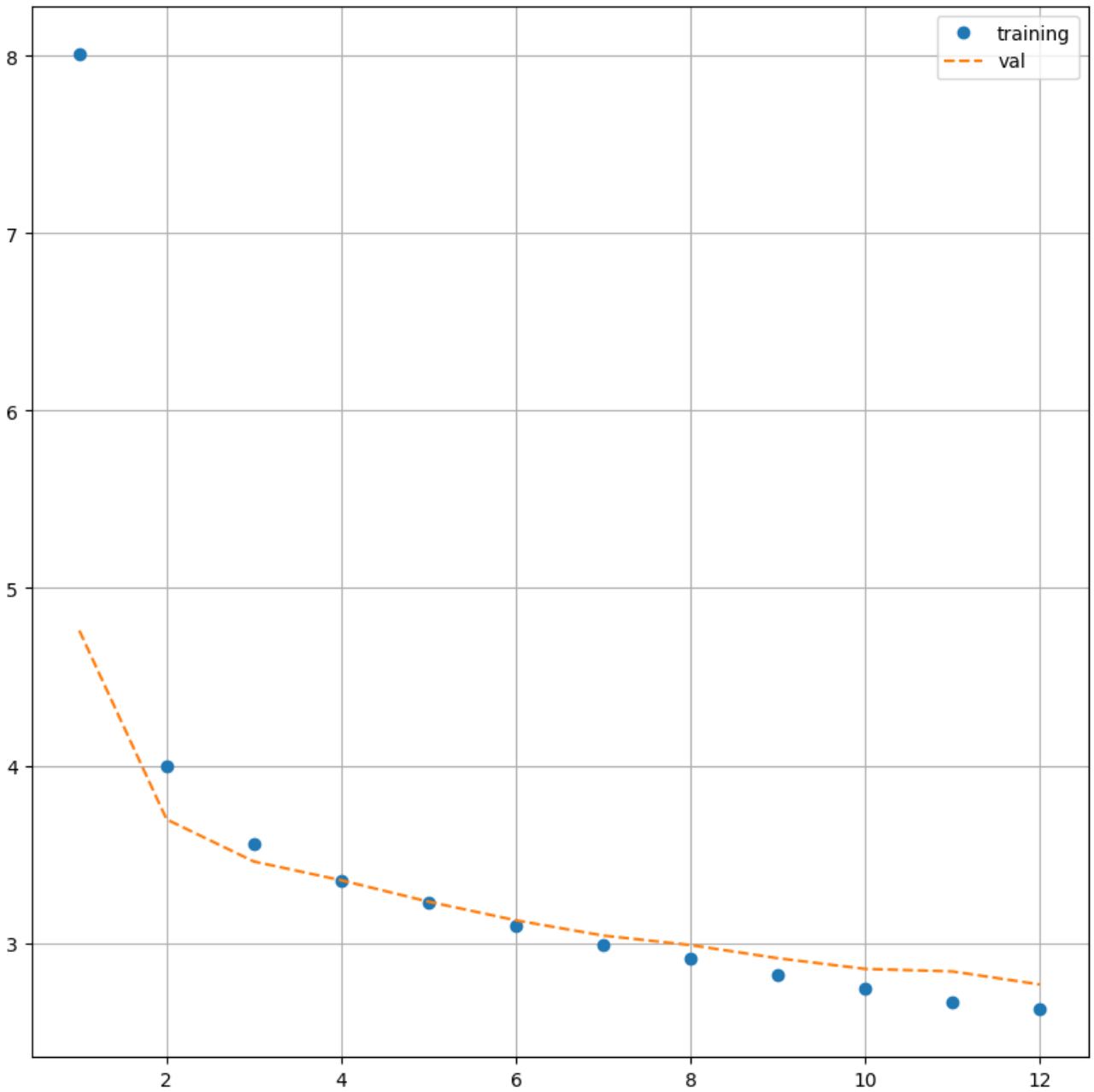
```
Epoch 1/15
4792/4800 [============================>.] - ETA: 0s - loss: 8.0105 - accurac
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/jax/_src/lib/__init__.py", li
    def _xla_gc_callback(*args):
KeyboardInterrupt:
4800/4800 [==============================] - 72s 15ms/step - loss: 8.0081 - a
Epoch 2/15
4800/4800 [==============================] - 19s 4ms/step - loss: 3.9984 - ac
Epoch 3/15
4800/4800 [==============================] - 21s 4ms/step - loss: 3.5573 - ac
Epoch 4/15
4800/4800 [==============================] - 20s 4ms/step - loss: 3.3512 - ac
Epoch 5/15
4800/4800 [==============================] - 22s 5ms/step - loss: 3.2269 - ac
Epoch 6/15
4800/4800 [==============================] - 20s 4ms/step - loss: 3.0984 - ac
Epoch 7/15
4800/4800 [==============================] - 20s 4ms/step - loss: 2.9865 - ac
Epoch 8/15
4800/4800 [==============================] - 20s 4ms/step - loss: 2.9132 - ac
Epoch 9/15
4800/4800 [==============================] - 23s 5ms/step - loss: 2.8234 - ac
Epoch 10/15
4800/4800 [==============================] - 19s 4ms/step - loss: 2.7398 - ac
Epoch 11/15
4800/4800 [==============================] - 23s 5ms/step - loss: 2.6657 - ac
Epoch 12/15
4800/4800 [==============================] - 20s 4ms/step - loss: 2.6254 - ac
Epoch 13/15
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 14/15
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 15/15
4800/4800 [==============================] - 21s 4ms/step - loss: nan - accur
```

```
# Evaluación el modelo en el conjunto de prueba
test_loss_l2, test_acc_l2 = mejor_modelo_l2.evaluate(test_images, test_labels)
print('Precisión en el conjunto de prueba:', test_acc_l2)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 2.6965 - accur
Precisión en el conjunto de prueba: 0.4569000005722046
```

```
history_dict_l2 = history_train_l2.history
loss_values_l2 = history_dict_l2['loss']
val_loss_values_l2 = history_dict_l2['val_loss']

fig = plt.figure(figsize=(10,10))
epoch = range(1,len(loss_values_l2)+1)
plt.plot(epoch,loss_values_l2, 'o',label='training')
plt.plot(epoch,val_loss_values_l2, '--',label='val')
plt.legend()
plt.grid()
plt.show()
```



```
!git status
!git add .
!git commit -m "Implementación L2. Mejor control que L1 pero termina perdiendo los
!git push
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Tarea3-Redes.ipynb

no changes added to commit (use "git add" and/or "git commit -a")
[main 1bf7635] Implementación L2. Mejor control que L1 pero termina perdiendo
 1 file changed, 1 insertion(+), 1 deletion(-)
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 872 bytes | 290.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Jorge-1501/Redes-Neuronales
   a8a0e5c..1bf7635  main -> main
```

L1-L2

```
# Cargar el modelo preentrenado
modelo_preentrenado_l1_l2 = tf.keras.models.load_model("mejor_modelo_segundo_entre

# Agregar capas de dropout y regularización L2
modelo_preentrenado_l1_l2.add(layers.Dense(10, activation='relu',
                                           kernel_regularizer=l1_l2(l1=0.00001, l2=0.
                                           name="mi_capa_l1_l2"))

# Compilar el modelo con las nuevas configuraciones
custom_optimizer3 = Adam(learning_rate=0.00001, beta_1=0.95, beta_2=0.999, epsilon
modelo_preentrenado_l1_l2.compile(optimizer=custom_optimizer3,
                                  loss="categorical_crossentropy",
                                  metrics=["accuracy"])

# Define el callback ModelCheckpoint para guardar el mejor modelo
```

```
checkpoint_mejor_modelo_l1_l2 = ModelCheckpoint("mejor_modelo_l1_l2.h5", monitor="

# Entrena el modelo preentrenado con el callback para guardar el mejor modelo
history_train_l1_l2 = modelo_preentrenado_l1_l2.fit(train_images, train_labels, ep
                                                    callbacks=[checkpoint_mejor_modelo_l1_l2]
                                                    validation_split=0.2)

# Cargar el mejor modelo guardado durante el segundo entrenamiento
mejor_modelo_l1_l2 = tf.keras.models.load_model("mejor_modelo_l1_l2.h5")
```

```
Epoch 1/20
4800/4800 [==============================] - 23s 4ms/step - loss: 8.8039 - ac
Epoch 2/20
4800/4800 [==============================] - 20s 4ms/step - loss: 6.2218 - ac
Epoch 3/20
4800/4800 [==============================] - 20s 4ms/step - loss: 4.4351 - ac
Epoch 4/20
4800/4800 [==============================] - 21s 4ms/step - loss: 4.0030 - ac
Epoch 5/20
4800/4800 [==============================] - 20s 4ms/step - loss: 3.6427 - ac
Epoch 6/20
4800/4800 [==============================] - 22s 5ms/step - loss: 3.2242 - ac
Epoch 7/20
4800/4800 [==============================] - 23s 5ms/step - loss: 2.8841 - ac
Epoch 8/20
4800/4800 [==============================] - 20s 4ms/step - loss: 2.7037 - ac
Epoch 9/20
4800/4800 [==============================] - 23s 5ms/step - loss: 2.6047 - ac
Epoch 10/20
4800/4800 [==============================] - 20s 4ms/step - loss: 2.5243 - ac
Epoch 11/20
4800/4800 [==============================] - 21s 4ms/step - loss: 2.4657 - ac
Epoch 12/20
4800/4800 [==============================] - 20s 4ms/step - loss: 2.3921 - ac
Epoch 13/20
4800/4800 [==============================] - 21s 4ms/step - loss: nan - accur
Epoch 14/20
4800/4800 [==============================] - 19s 4ms/step - loss: nan - accur
Epoch 15/20
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 16/20
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 17/20
4800/4800 [==============================] - 21s 4ms/step - loss: nan - accur
Epoch 18/20
4800/4800 [==============================] - 23s 5ms/step - loss: nan - accur
Epoch 19/20
4800/4800 [==============================] - 26s 5ms/step - loss: nan - accur
Epoch 20/20
4800/4800 [==============================] - 25s 5ms/step - loss: nan - accur
```

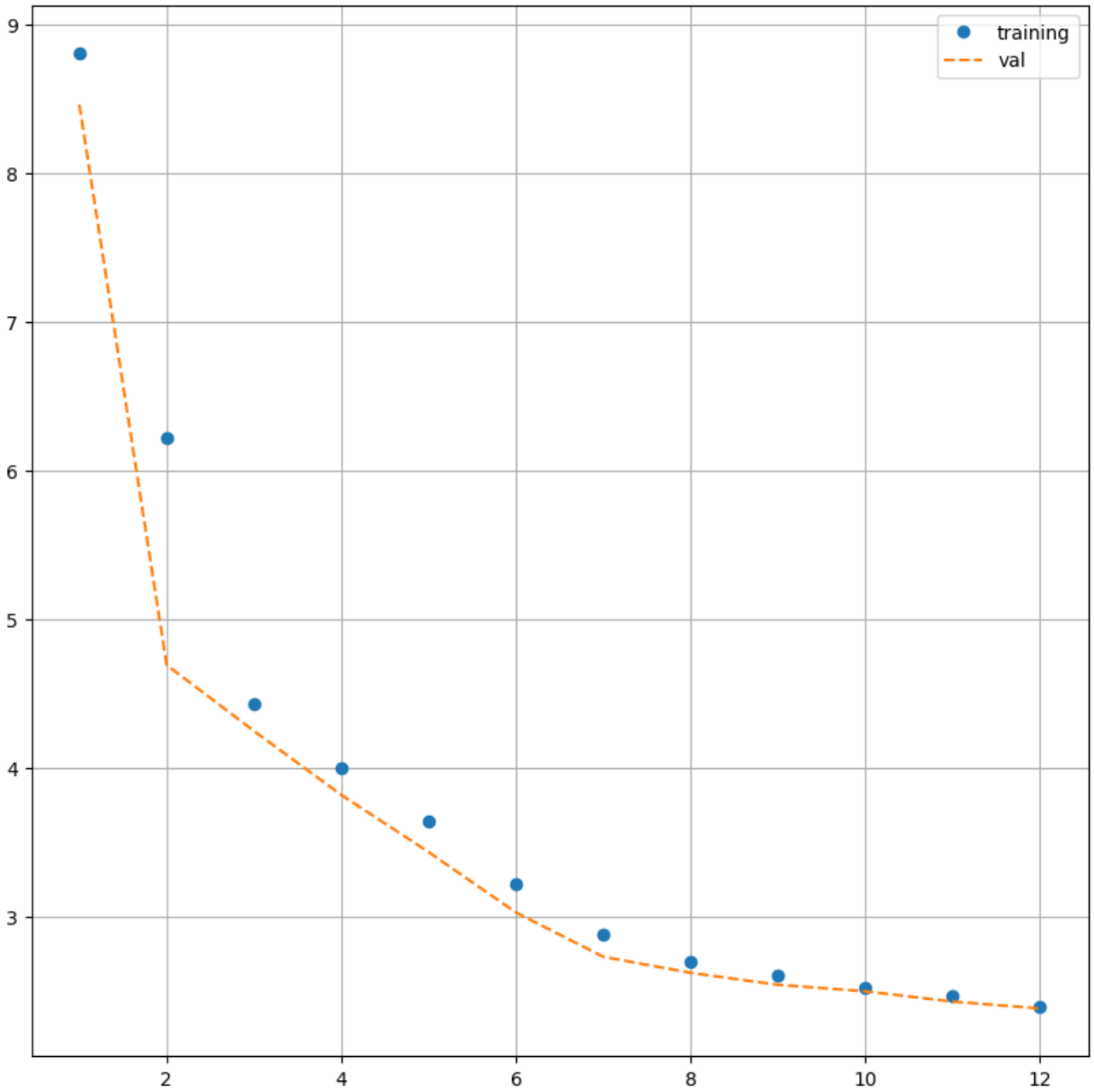
```
# Evaluación el modelo en el conjunto de prueba
test_loss_l1_l2, test_acc_l1_l2 = mejor_modelo_l1_l2.evaluate(test_images, test_la
print('Precisión en el conjunto de prueba:', test_acc_l1_l2)
```

```
print('Precisión en el conjunto de prueba:', test_acc_l1_l2)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 2.3866 - accur
Precisión en el conjunto de prueba: 0.6866000294685364
```

```
history_dict_l1_l2 = history_train_l1_l2.history
loss_values_l1_l2 = history_dict_l1_l2['loss']
val_loss_values_l1_l2 = history_dict_l1_l2['val_loss']

fig = plt.figure(figsize=(10,10))
epoch = range(1,len(loss_values_l1_l2)+1)
plt.plot(epoch,loss_values_l1_l2, 'o',label='training')
plt.plot(epoch,val_loss_values_l1_l2, '--',label='val')
plt.legend()
plt.grid()
plt.show()
```



Con la combinación de los reguladores L1-L2 se ha logrado llegar a un mejor acurrancy sin sobreajuste, sin embargo sigue siendo bajo con el detalle adicional que en pocas épocas se pierden los valores.

```
!git status
!git add .
!git commit -m "Implementación L1-L2. Mejor control y accurancy pero continua perd
!git push
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Tarea3-Redes.ipynb

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        mejor_modelo_l1_l2.h5

no changes added to commit (use "git add" and/or "git commit -a")
[main 121caea] Implementación L1-L2. Mejor control y accurancy pero continua
 2 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 Digitos/src/mejor_modelo_l1_l2.h5
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 6.24 MiB | 7.20 MiB/s, done.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Jorge-1501/Redes-Neuronales
   1bf7635..121caea  main -> main
```

Dropout y L1-L2

```
# Cargar el modelo preentrenado
modelo_preentrenado_Dropout = tf.keras.models.load_model("mejor_modelo_segundo_ent
```

```
modelo_preentrenado_Dropout = tf.keras.models.load_model("mejor_modelo_segundo_ent

# Agregar capas de dropout y regularización L2
modelo_preentrenado_Dropout.add(layers.Dense(10, activation='relu',
                                        kernel_regularizer=l1_l2(l1=0.00001, l2=0.
                                        name="mi_capa_Dropout"))
modelo_preentrenado_Dropout.add(Dropout(0.2))
modelo_preentrenado_Dropout.add(layers.Dense(10, activation='relu',
                                        kernel_regularizer=l1_l2(l1=0.00001, l2=0.
                                        name="mi_capa_Dropout2"))

# Compilar el modelo con las nuevas configuraciones
custom_optimizer3 = Adam(learning_rate=0.00001, beta_1=0.95, beta_2=0.999, epsilon
modelo_preentrenado_Dropout.compile(optimizer=custom_optimizer3,
                                    loss="categorical_crossentropy",
                                    metrics=["accuracy"])

# Define el callback ModelCheckpoint para guardar el mejor modelo
checkpoint_mejor_modelo_Dropout = ModelCheckpoint("mejor_modelo_Dropout.h5",
                                                  monitor="val_loss", save_best_on

# Entrena el modelo preentrenado con el callback para guardar el mejor modelo
history_train_Dropout = modelo_preentrenado_Dropout.fit(train_images, train_labels
                                         callbacks=[checkpoint_mejor_modelo_Dropou
                                         validation_split=0.2)

# Cargar el mejor modelo guardado durante el segundo entrenamiento
mejor_modelo_Dropout = tf.keras.models.load_model("mejor_modelo_Dropout.h5")
```

```
Epoch 1/25
4800/4800 [==============================] - 23s 4ms/step - loss: 4.9259 - ac
Epoch 2/25
4800/4800 [==============================] - 21s 4ms/step - loss: 3.7363 - ac
Epoch 3/25
4800/4800 [==============================] - 20s 4ms/step - loss: 3.0498 - ac
Epoch 4/25
4800/4800 [==============================] - 21s 4ms/step - loss: 2.6583 - ac
Epoch 5/25
4800/4800 [==============================] - 23s 5ms/step - loss: 2.3709 - ac
Epoch 6/25
4800/4800 [==============================] - 22s 5ms/step - loss: 2.1080 - ac
Epoch 7/25
4800/4800 [==============================] - 22s 5ms/step - loss: 1.9043 - ac
Epoch 8/25
4800/4800 [==============================] - 21s 4ms/step - loss: 1.7332 - ac
Epoch 9/25
4800/4800 [==============================] - 21s 4ms/step - loss: 1.5685 - ac
Epoch 10/25
4800/4800 [==============================] - 20s 4ms/step - loss: 1.4240 - ac
Epoch 11/25
4800/4800 [==============================] - 21s 4ms/step - loss: 1.3184 - ac
Epoch 12/25
4800/4800 [==============================] - 20s 4ms/step - loss: 1.1317 - ac
```

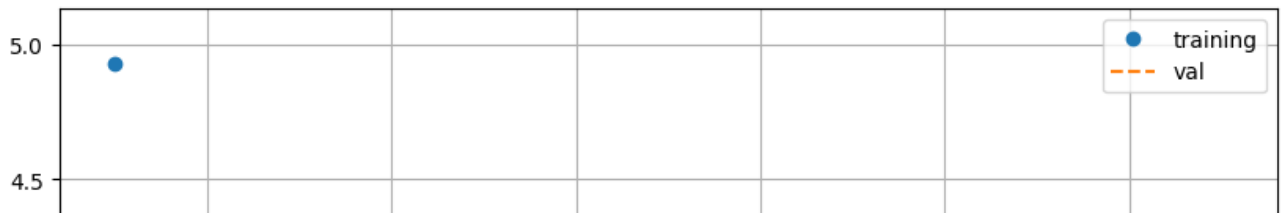
```
Epoch 13/25
4800/4800 [==============================] - 21s 4ms/step - loss: 1.0772 - ac
Epoch 14/25
4800/4800 [==============================] - 21s 4ms/step - loss: nan - accur
Epoch 15/25
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 16/25
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 17/25
4800/4800 [==============================] - 21s 4ms/step - loss: nan - accur
Epoch 18/25
4800/4800 [==============================] - 21s 4ms/step - loss: nan - accur
Epoch 19/25
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 20/25
4800/4800 [==============================] - 21s 4ms/step - loss: nan - accur
Epoch 21/25
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 22/25
4800/4800 [==============================] - 21s 4ms/step - loss: nan - accur
Epoch 23/25
4800/4800 [==============================] - 20s 4ms/step - loss: nan - accur
Epoch 24/25
4800/4800 [==============================] - 21s 4ms/step - loss: nan - accur
Epoch 25/25
4800/4800 [==============================] - 21s 4ms/step - loss: nan - accur
```

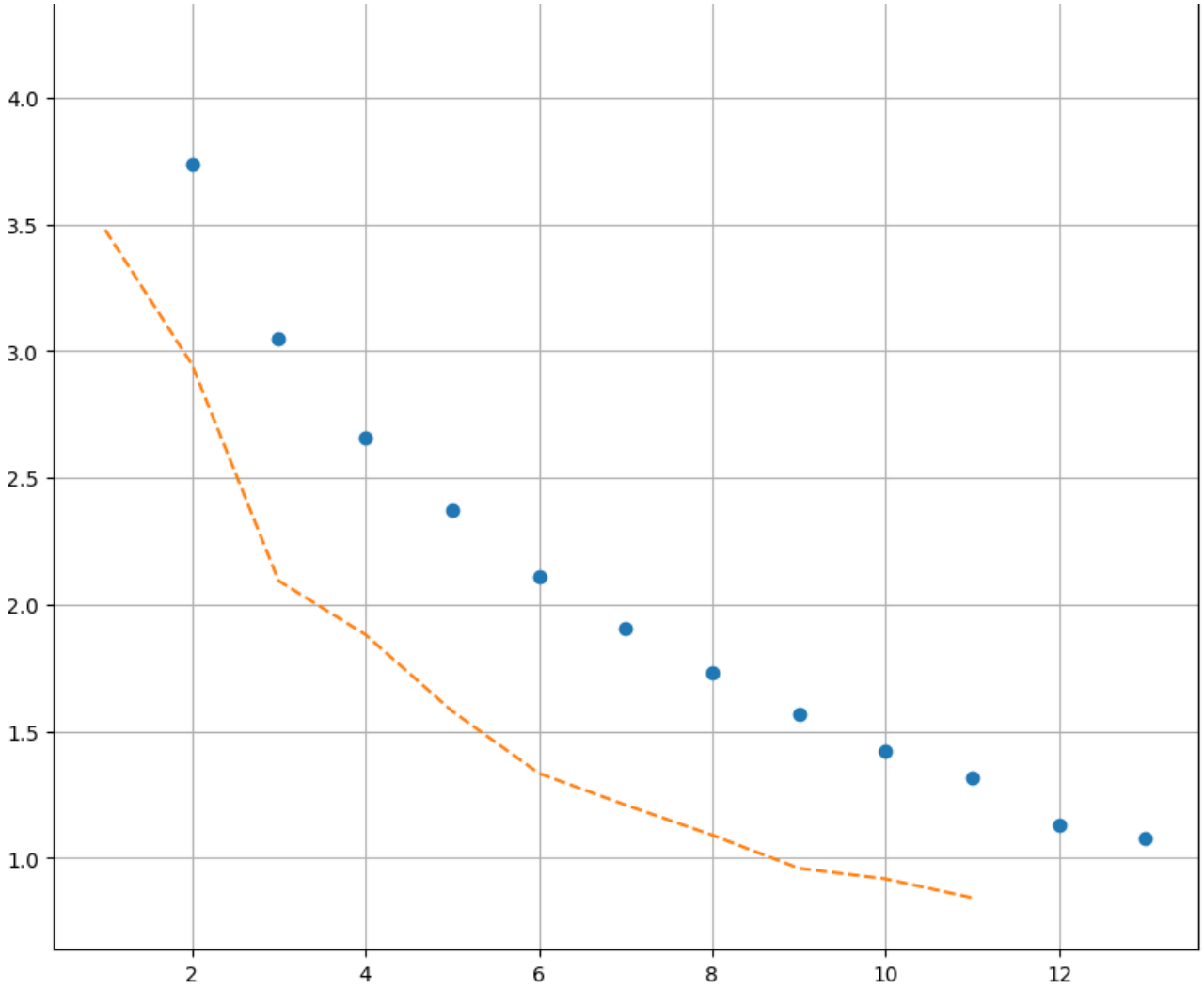
```
# Evaluación el modelo en el conjunto de prueba
test_loss_Dropout, test_acc_Dropout = mejor_modelo_Dropout.evaluate(test_images, t
print('Precisión en el conjunto de prueba:', test_acc_Dropout)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.8788 - accur
Precisión en el conjunto de prueba: 0.7721999883651733
```

```
history_dict_Dropout = history_train_Dropout.history
loss_values_Dropout = history_dict_Dropout['loss']
val_loss_values_Dropout = history_dict_Dropout['val_loss']

fig = plt.figure(figsize=(10,10))
epoch = range(1,len(loss_values_Dropout)+1)
plt.plot(epoch,loss_values_Dropout, 'o',label='training')
plt.plot(epoch,val_loss_values_Dropout, '--',label='val')
plt.legend()
plt.grid()
plt.show()
```





La implementación de Dropout logró ralentizar el momento donde los valores se pierden, gracias a eso se logró llegar a un mejor accurancy de los resultados además de no caer en un sobreajuste. Accurancy: 0.77

```
!git status
!git add .
!git commit -m "Implementación Dropout. Mejor accurancy pero aún se pierden los da
!git push
```

Reporte de resultados

Logré obtener mejores resultados con un rango de aprendizaje lento y muchas epocas que volviendo a entrenar una red e implementando algún regularizador ya que los valores se perdían algo rápido con los regularizadores.

La implementación de Dropout junto con los regularizadores logró obtener mejores resultados que todo el uso único de regularizadores, sin embargo aun se encontraron problemas de pérdida de información.

```
!git status
!git add .
!git commit -m "Resumen de cambios"
!git push
```

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Tarea3-Redes.ipynb

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        mejor_modelo_Dropout.h5

no changes added to commit (use "git add" and/or "git commit -a")
[main 76ea93f] Resumen de cambios
 2 files changed, 1 insertion(+), 1 deletion(-)
 rewrite Digitos/src/Tarea3-Redes.ipynb (94%)
 create mode 100644 Digitos/src/mejor_modelo_Dropout.h5
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 6.35 MiB | 8.83 MiB/s, done.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Jorge-1501/Redes-Neuronales
   121caea..76ea93f  main -> main
```