P P3. Redes Neuronales

ANGEL GAEL CALLEROS MARTÍNEZ 654375

Importacion de Datos

Los siguientes 3 modelos funcionaran con esta base de datos

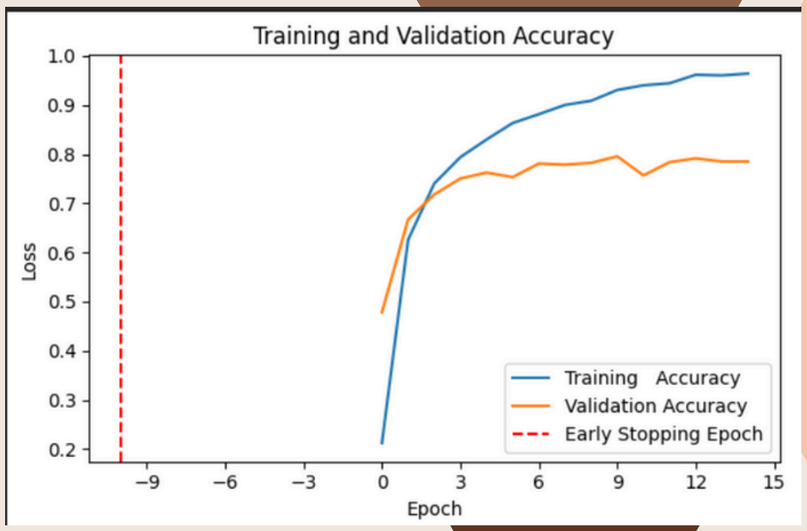
```python
from google.colab import drive
drive.mount('/content/drive')

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

TRAIN_DIR = '/content/drive/MyDrive/Numbers/Train'
TEST_DIR  = '/content/drive/MyDrive/Numbers/Test'
IMG_SIZE  = (28, 28)
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    validation_split=0.20
)

test_datagen = ImageDataGenerator(
    rescale=1.0/255.0
)
```

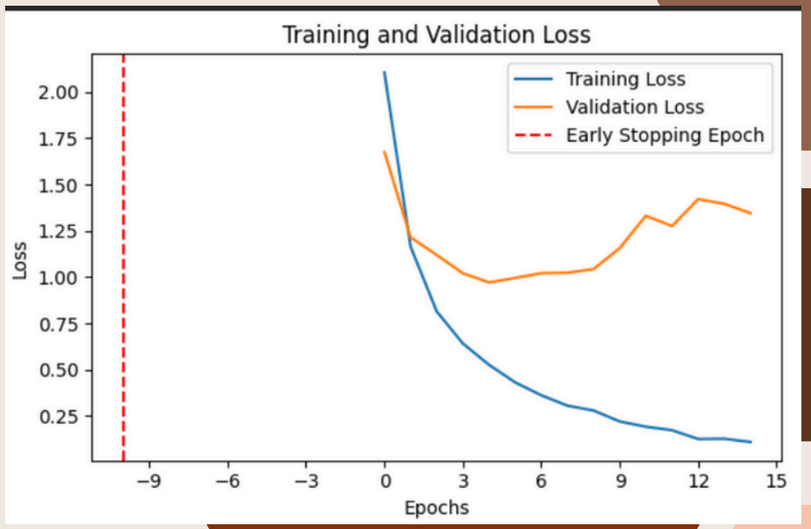
```python
train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    target_size=IMG_SIZE,
    color_mode='grayscale',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    shuffle=True,
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    target_size=IMG_SIZE,
    color_mode='grayscale',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    shuffle=True,
    subset='validation'
)

test_generator = test_datagen.flow_from_directory(
    TEST_DIR,
    target_size=IMG_SIZE,
    color_mode='grayscale',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    shuffle=False
)
```

Modelo 1

- Se creo un modelo basico con 2 redes convulucionales para imagenes de 28x28 pixeles y 4 capas densas obteniendo un accuracy de validacion de 79.85%

```
from tensorflow.keras import layers, models
model=models.Sequential()

model.add(layers.Conv2D(32,(3,3),padding='same' ,
                        activation='relu',input_shape=(28,28,1)))
model.add(layers.MaxPooling2D((2,2)))

model.add(layers.Conv2D(64,(3,3),activation='relu'))
model.add(layers.MaxPooling2D((2,2)))

model.add(layers.Flatten())
model.add(layers.Dense(128,activation='relu'))
model.add(layers.Dense(64,activation='relu'))
model.add(layers.Dense(16,activation='relu'))
model.add(layers.Dense(10,activation='softmax'))
```





Modelo 2

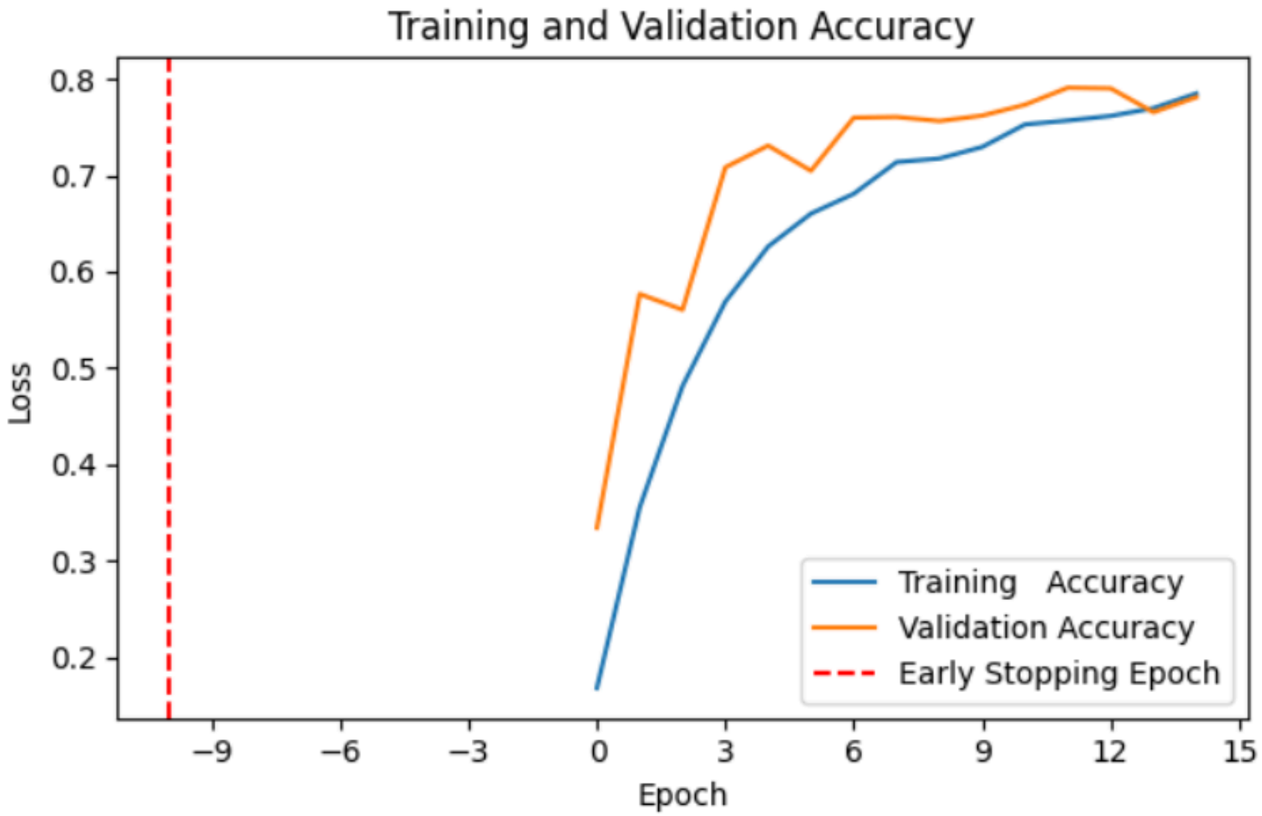
```
model2 = models.Sequential()

model2.add(layers.Conv2D(48, (5, 5), padding='same',
                         activation='relu', input_shape=(28, 28, 1)))
model2.add(layers.MaxPooling2D((2,2)))
model2.add(layers.BatchNormalization())
model2.add(layers.Activation('relu'))
model2.add(layers.MaxPooling2D((2, 2)))
model2.add(layers.Dropout(0.1))

model2.add(layers.Conv2D(64, (3, 3), padding='same', activation='relu'))
model2.add(layers.MaxPooling2D((2, 2)))
model2.add(layers.Dropout(0.3))

model2.add(layers.Conv2D(128, (3, 3), padding='same', activation='relu'))

model2.add(layers.Flatten())
model2.add(layers.Dense(64,activation='relu'))
model2.add(layers.Dropout(0.3))
model2.add(layers.Dense(16,activation='relu'))
model2.add(layers.Dropout(0.3))

model2.add(layers.Dense(10, activation='softmax'))
```

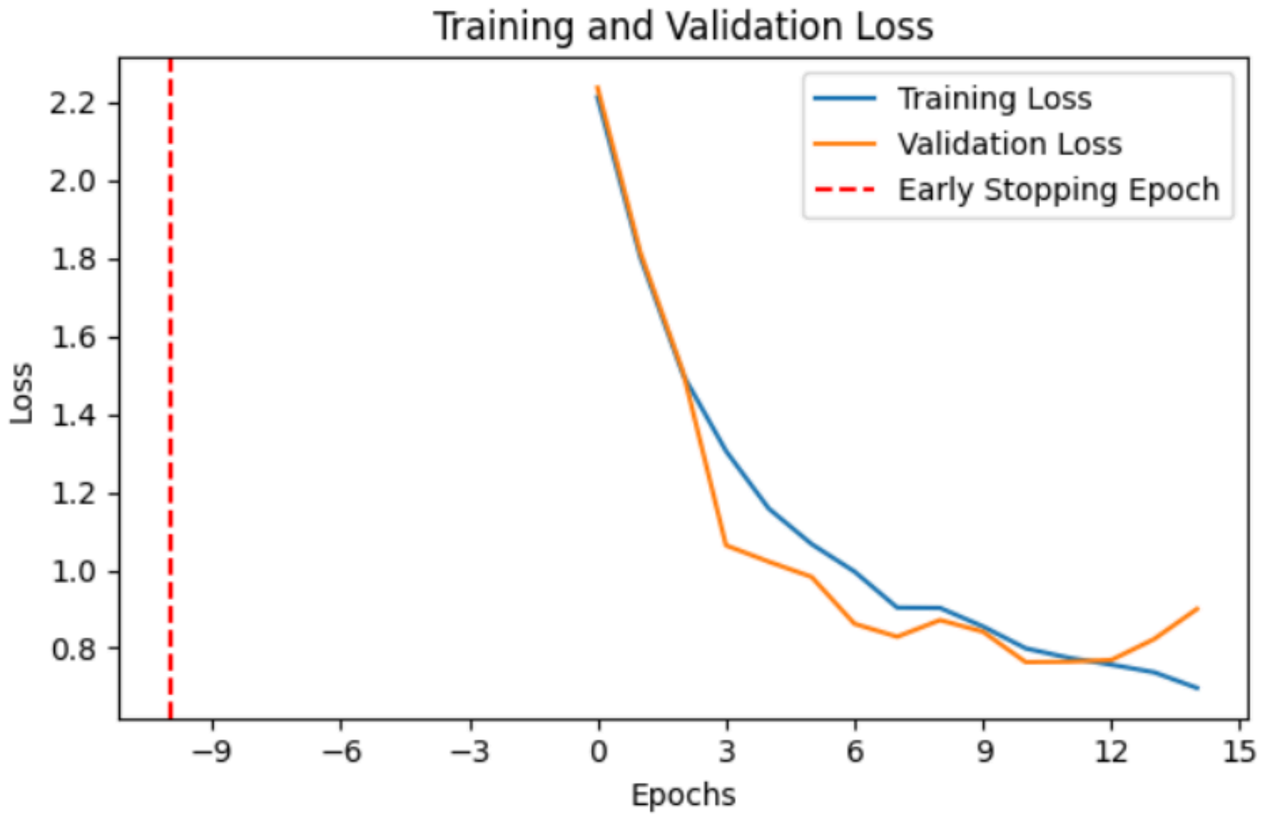
- La primera capa convulucional, de 32 a 48 y cambiaremos tambien de 3,3 a 5,5 esto para poder captar mas cosas a la hora de analizar los pixels, como curvas y bordes para mayor presición.

- Agregaremos tambien un BatchNormalization para que la red aprenda mas rapido y no se haga eterno el modelo, esto haciendo que las imagenes se normalizen. Añadiremos otra capa de puramente activación para que pueda aprender patrones nuevamente despues de activar el BatchNormalization.

- Para terminar agregaremos varios Dropout en todo el modelo para que no memorice.

Modelo 2






```
45/45 ━━━━━━━━━━━━━━━━━━━━ 8s 184ms/step - accuracy: 0.7950 - loss: 0.7052
Validation Accuracy: 79.08%
Validation Loss: 0.7649703025817871
```

Modelo 3

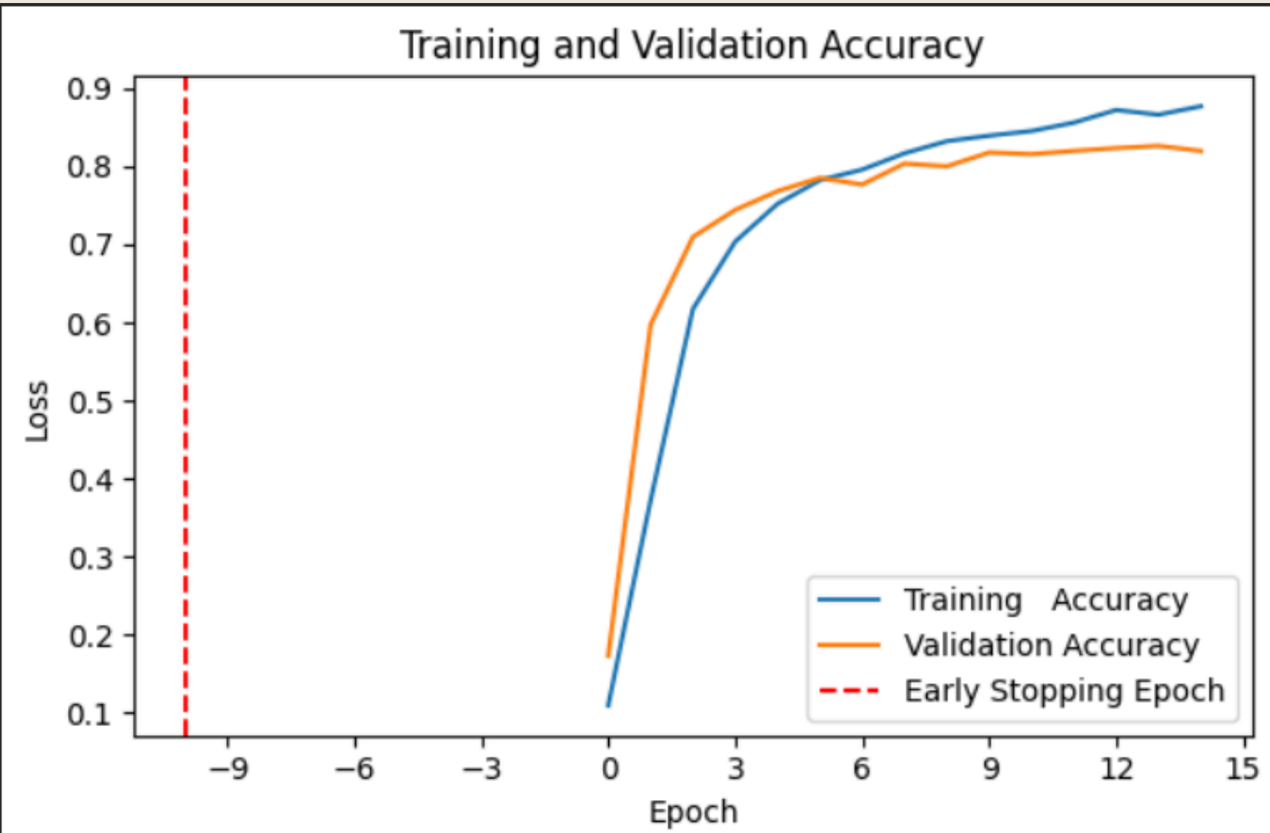
- Se quito el BatchNormalization y el Activation de todas las capas, suponiendo que era por esto que nuestro modelo no mejoro.

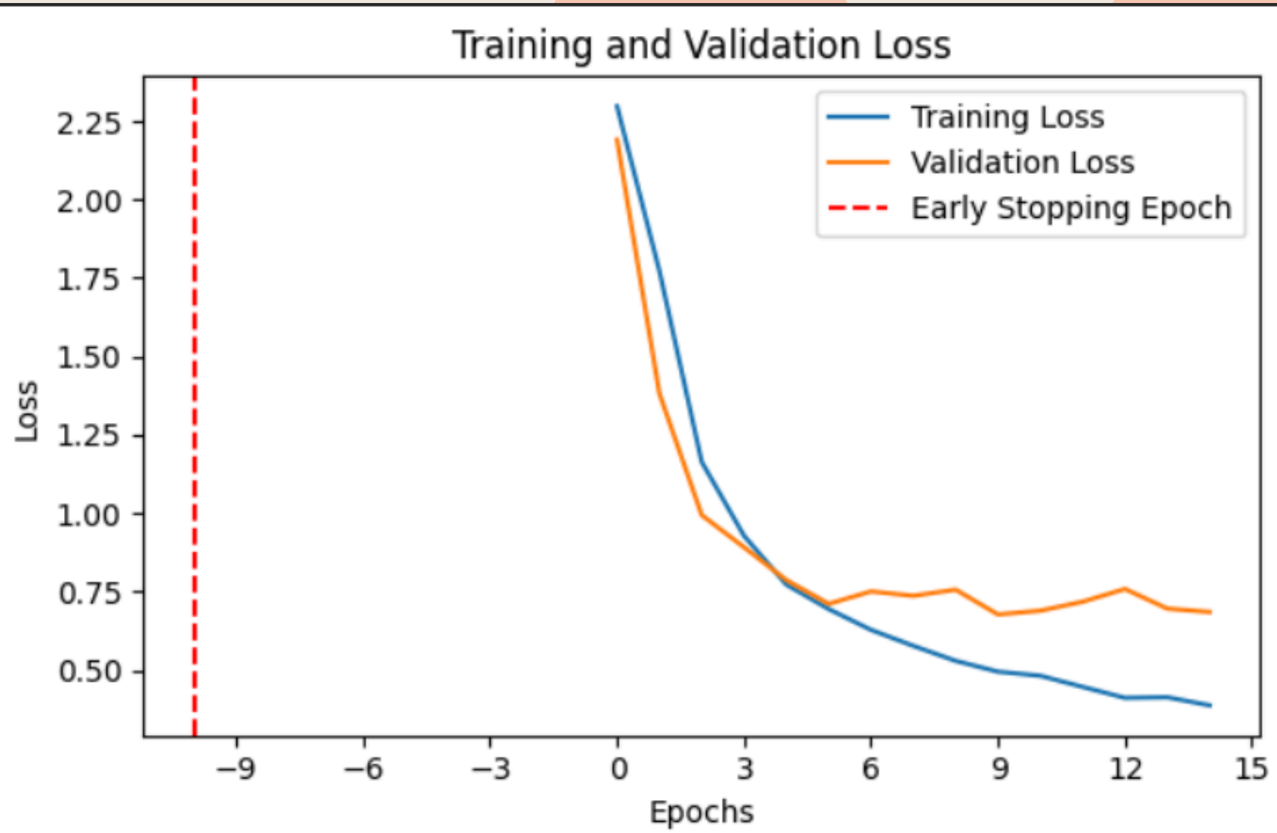
- Cambiaremos el Dropout de las convuluciones de 0.1 a 0.5 para que tenga mayor efecto.

- En la segunda y tercer capa cambiaremos los valors de 3,3 a 5,5 para a medida que la red se vaya volviendo mas compleja tome mas pixeles y pueda otorgar un mejor resultado.

```
model3 = models.Sequential()

model3.add(layers.Conv2D(48, (3, 3), padding='same',
                         activation='relu', input_shape=(28, 28, 1)))
model3.add(layers.MaxPooling2D((2,2)))
model3.add(layers.Dropout(0.5))

model3.add(layers.Conv2D(64, (5, 5), padding='same', activation='relu'))
model3.add(layers.MaxPooling2D((2, 2)))
model3.add(layers.Dropout(0.5))

model3.add(layers.Conv2D(128, (5, 5), padding='same', activation='relu'))

model3.add(layers.Flatten())
model3.add(layers.Dense(128,activation='relu'))
model3.add(layers.Dense(64,activation='relu'))
model3.add(layers.Dropout(0.3))
model3.add(layers.Dense(16,activation='relu'))
model3.add(layers.Dense(10,activation='softmax'))
```

Modelo 3






```
45/45 ━━━━━━━━━━━━━━━━━━━━ 9s 194ms/step - accuracy: 0.8393 - loss: 0.6232
Validation Accuracy: 82.61%
Validation Loss: 0.6958523988723755
```

Modelos 4 y 5

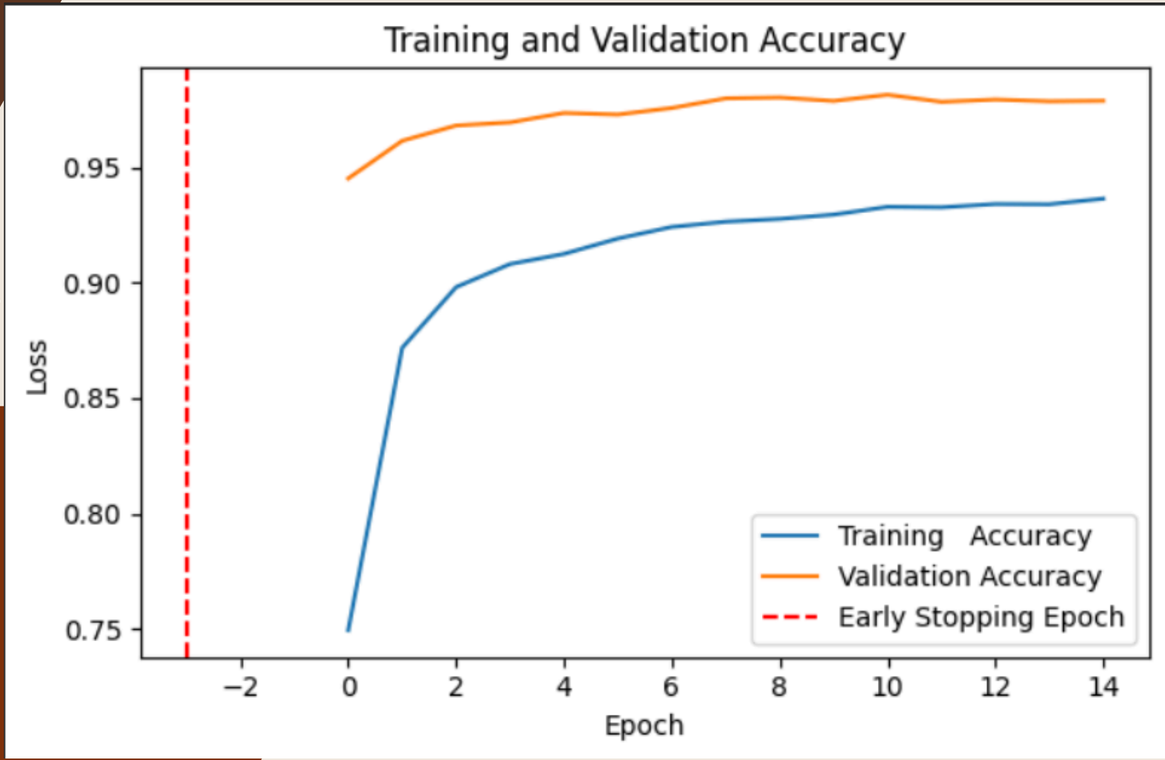
```python
import os, cv2, numpy as np
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split

def load_split(root_dir, img_size=(28,28)):
    imgs, labels = [], []
    for name in sorted(os.listdir(root_dir)):
        sub = os.path.join(root_dir, name)
        if not os.path.isdir(sub):
            continue
        try:
            label = int(name)
        except:
            continue
        for fname in os.listdir(sub):
            fpath = os.path.join(sub, fname)
            img = cv2.imread(fpath, cv2.IMREAD_GRAYSCALE)
            if img is None:
                continue
            img = cv2.resize(img, img_size, interpolation=cv2.INTER_AREA)
            imgs.append(img)
            labels.append(label)
    imgs = np.array(imgs, dtype=np.float32) / 255.0
    imgs = np.expand_dims(imgs, axis=-1)
    labels = np.array(labels, dtype=np.int64)
    return imgs, labels

train_custom, y_train_custom = load_split(TRAIN_DIR, IMG_SIZE)
test_custom,  y_test_custom  = load_split(TEST_DIR,  IMG_SIZE)

(x_train_m, y_train_m), (x_test_m, y_test_m) = mnist.load_data()
x_train_m = x_train_m.astype("float32")/255.0
x_test_m  = x_test_m.astype("float32")/255.0
x_train_m = np.expand_dims(x_train_m, -1)
x_test_m  = np.expand_dims(x_test_m,  -1)
```

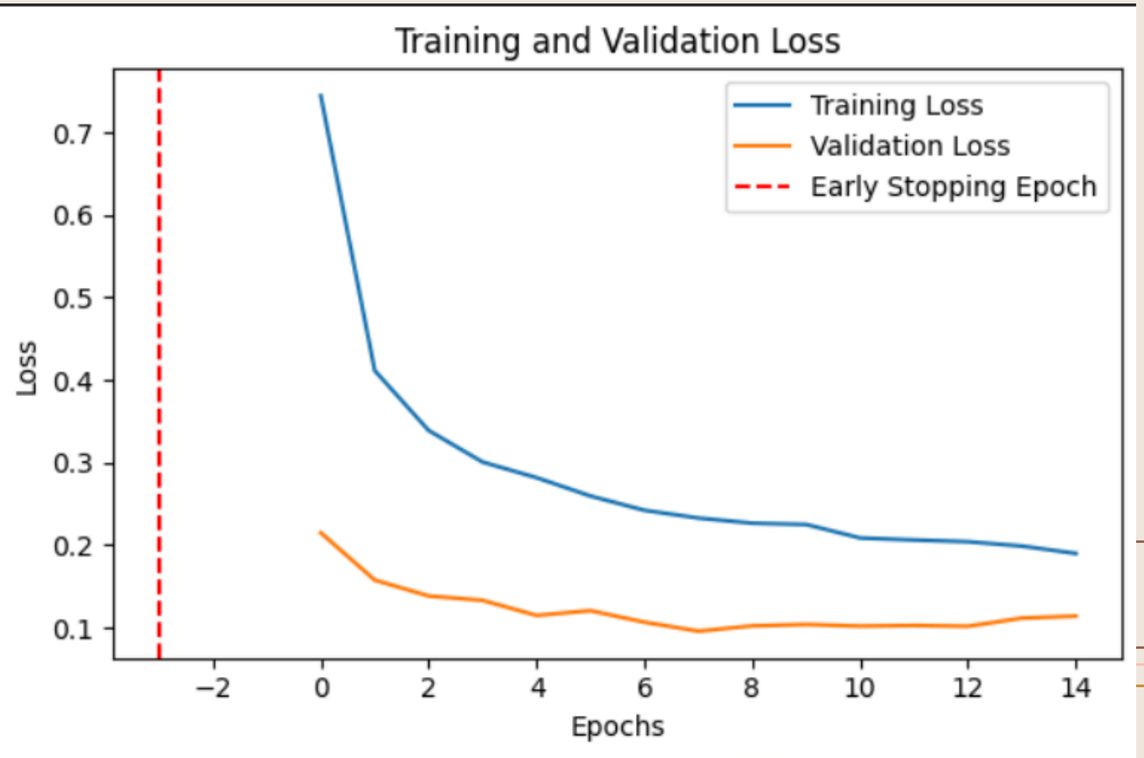
Para el modelo 4 y 5 utilizaremos la misma carpeta de drive más la base de datos de Mnist para tener mayor cantidad de datos de entrenamiento.

```python
x_train_all = np.concatenate([x_train_m, train_custom], axis=0)
y_train_all = np.concatenate([y_train_m, y_train_custom], axis=0)
x_test_all  = np.concatenate([x_test_m,  test_custom],  axis=0)
y_test_all  = np.concatenate([y_test_m,  y_test_custom], axis=0)

rng = np.random.default_rng(123)
idx_tr = rng.permutation(len(x_train_all))
idx_te = rng.permutation(len(x_test_all))

x_train_all = x_train_all[idx_tr]
y_train_all = y_train_all[idx_tr]
x_test_all  = x_test_all[idx_te]
y_test_all  = y_test_all[idx_te]

x_train, x_val, y_train, y_val = train_test_split(
    x_train_all, y_train_all,
    test_size=0.20,
    random_state=123,
    stratify=y_train_all
)

num_classes = 10
y_train_cat = to_categorical(y_train, num_classes)
y_val_cat   = to_categorical(y_val,   num_classes)
y_test_cat  = to_categorical(y_test_all, num_classes)

print("Train:", x_train.shape, y_train_cat.shape)
print("Val  :", x_val.shape,   y_val_cat.shape)
print("Test :", x_test_all.shape,  y_test_cat.shape)
```

Modelo 4

- Se uso la misma estructura convulucional que en el modelo pasada ya que es la que mejor resultados nos ha entregado.
- A su vez viendo las graficas son las primeras que spueran por mucho a las de train y esto se debe a que se obtuvo un Accuracy de 97%





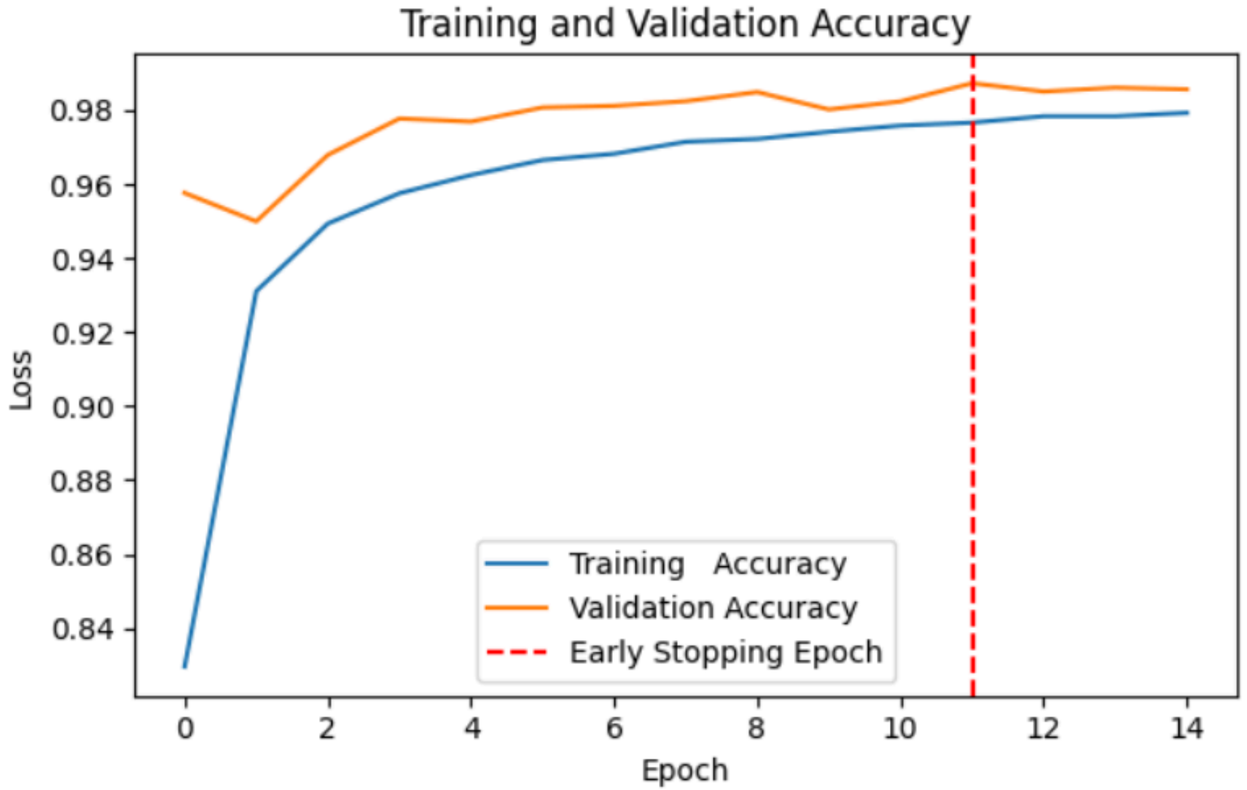
Modelo 5

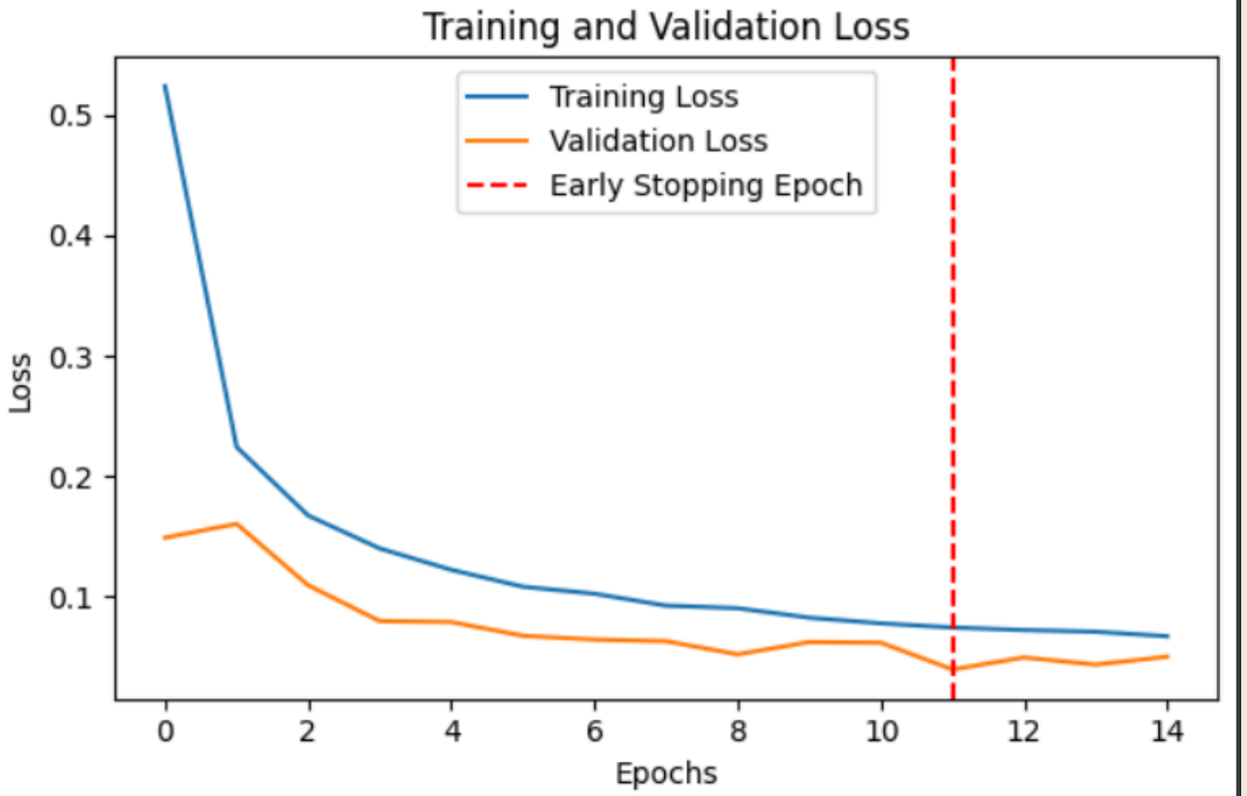
```
datagen = ImageDataGenerator(
    rotation_range=15,         # rota 15 grados
    width_shift_range=0.1,     # desplaza horizontalmente 10% del ancho
    height_shift_range=0.1,    # desplaza verticalmente 10% de la altura
    zoom_range=0.1,            # zoom in/out 10%
    shear_range=0.1,           # pequeño corte (shear)
    fill_mode='nearest'        # rellena los huecos al rotar/shift
)

datagen.fit(x_train)
```

```
model5 = models.Sequential()

model5.add(layers.Conv2D(48,(5,5),padding='same' ,
                         activation='relu',input_shape=(28,28,1)))
model5.add(layers.MaxPooling2D((2,2)))

model5.add(layers.Conv2D(64,(3,3),activation='relu'))
model5.add(layers.MaxPooling2D((2,2)))

model5.add(layers.Flatten())
model5.add(layers.Dense(128,activation='relu'))
model5.add(layers.Dense(64,activation='relu'))
model5.add(layers.Dense(16,activation='relu'))
model5.add(layers.Dense(10,activation='softmax'))
```

- Para este modelo se generaron variaciones tanto en la rotacion , traslacion, zoom, etc.
- Todo esto con el objetivo de que no caiga en el overfiting al ser entrenado por su mayoria en Mnist y que asi no quede con cosas basicas que son faciles de analizar.

Modelo 5






```
Test Accuracy: 98.25%
Test Loss    : 0.0561
```

Modelo 6

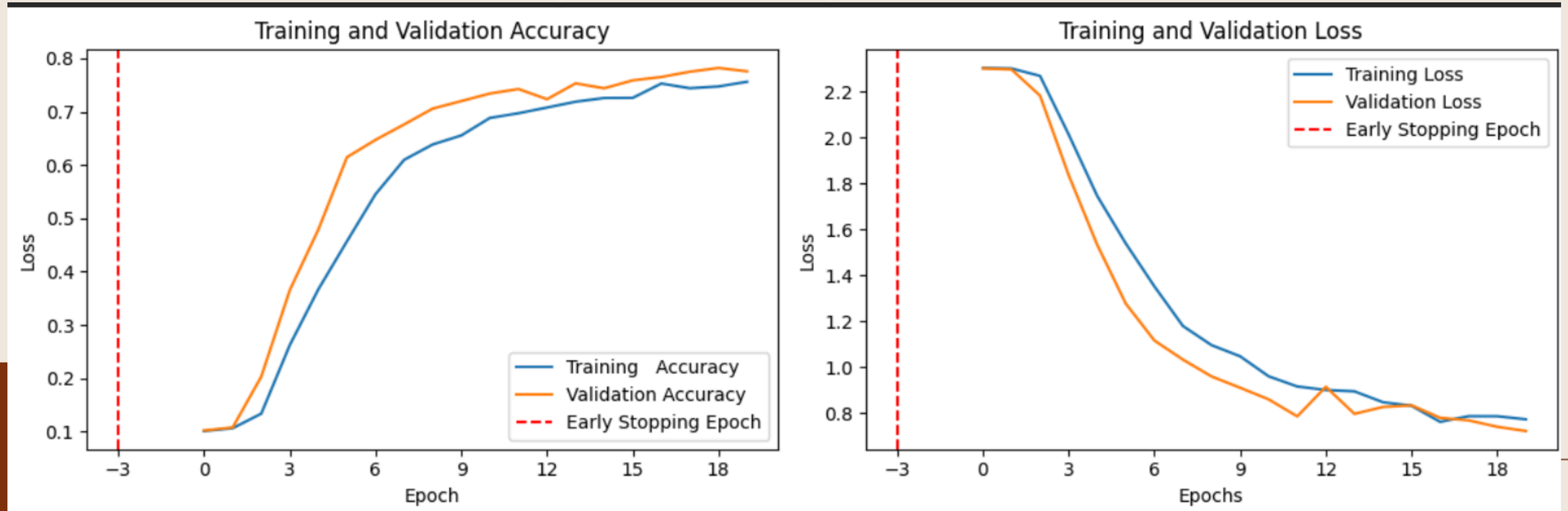
```
TRAIN_DIR = '/content/drive/MyDrive/Numbers/Train'
TEST_DIR  = '/content/drive/MyDrive/Numbers/Test'
IMG_SIZE  = (28, 28)
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(
    rescale=1.0/255.0,
    rotation_range=15,
    width_shift_range=0.10,
    height_shift_range=0.10,
    zoom_range=0.10,
    shear_range=0.10,
    fill_mode='nearest',
    validation_split=0.20
)

test_datagen = ImageDataGenerator(
    rescale=1.0/255.0
)

train_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    target_size=IMG_SIZE,
    color_mode='grayscale',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    shuffle=True,
    subset='training'
)
```

- Para este modelo se generaron variaciones tanto en la rotacion , traslacion, zoom, etc.
- Con el fin de que ahora se obtenga un accuracy mayor al 82%

```
val_generator = train_datagen.flow_from_directory(
    TRAIN_DIR,
    target_size=IMG_SIZE,
    color_mode='grayscale',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    shuffle=False,
    subset='validation'
)

test_generator = test_datagen.flow_from_directory(
    TEST_DIR,
    target_size=IMG_SIZE,
    color_mode='grayscale',
    class_mode='categorical',
    batch_size=BATCH_SIZE,
    shuffle=False
)
```

Modelo 6

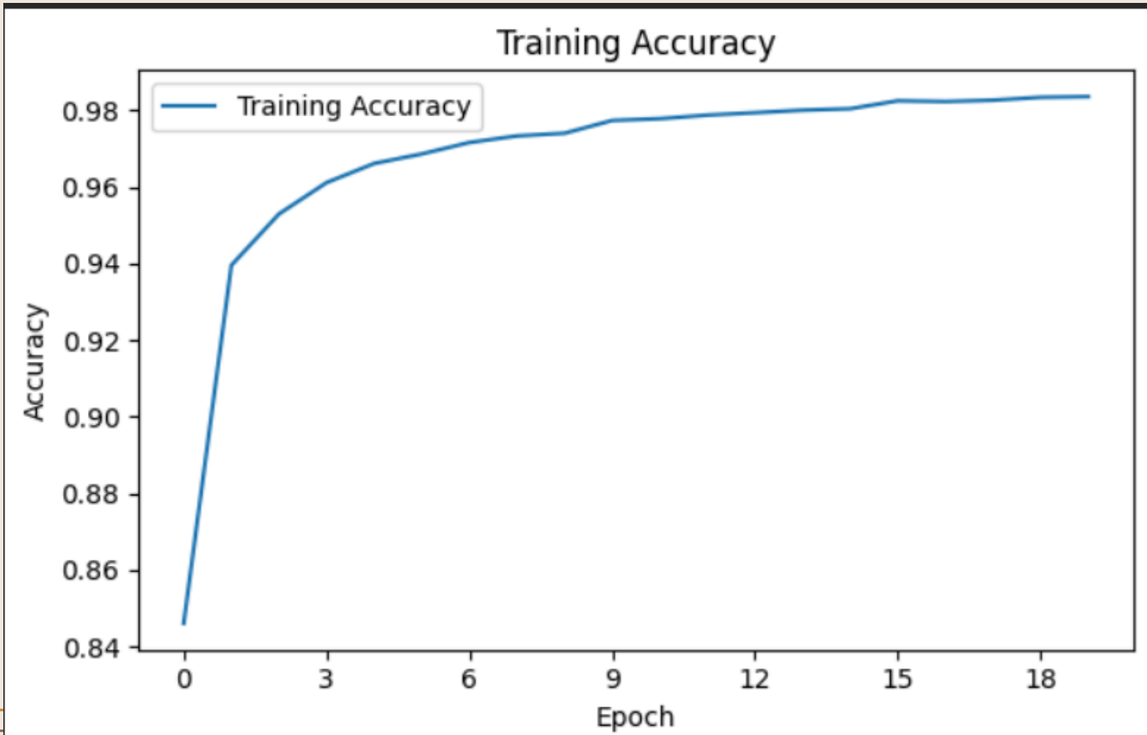
- Podemos observar que el valor es menor al 80% esperado por lo que podriamos decir que es peor a los anteriores modelos sin Mnist, sin embargo sus graficas son mucho mejores al estan parecidas al train.

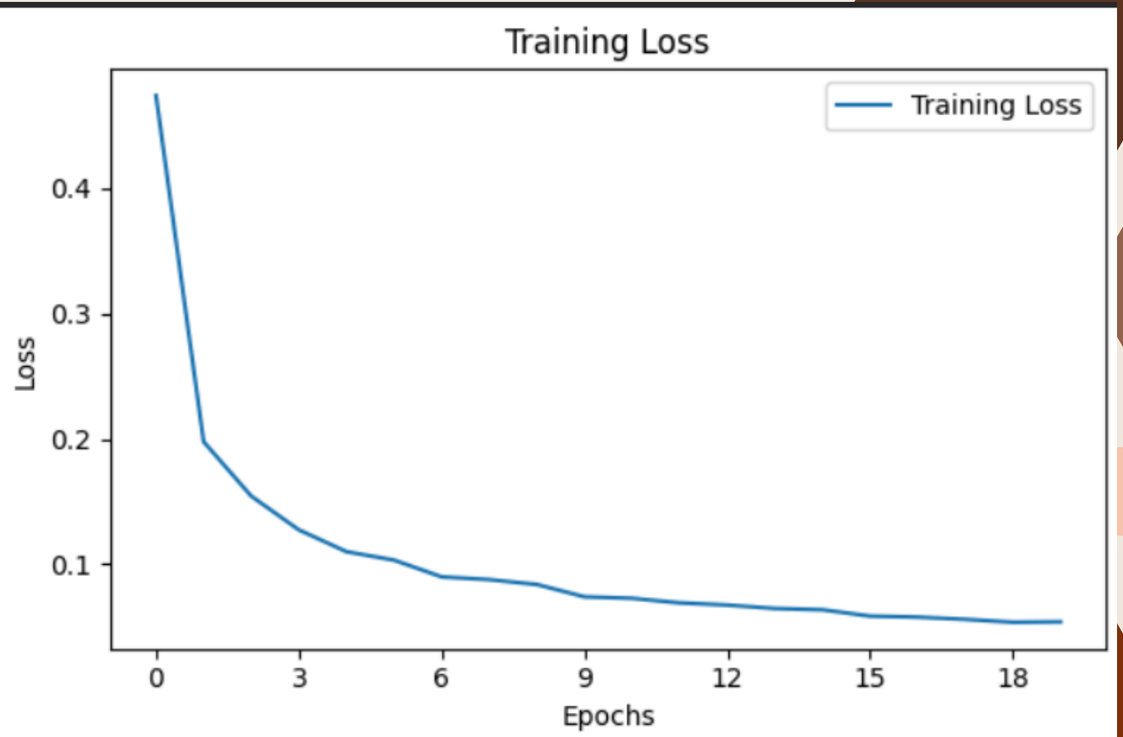



```
45/45 ━━━━━━━━━━━━━━━━━━━━ 9s 211ms/step - accuracy: 0.8245 - loss: 0.6892
Validation Accuracy: 79.86%
Validation Loss: 0.7376120686531067
```

Modelo Ganador

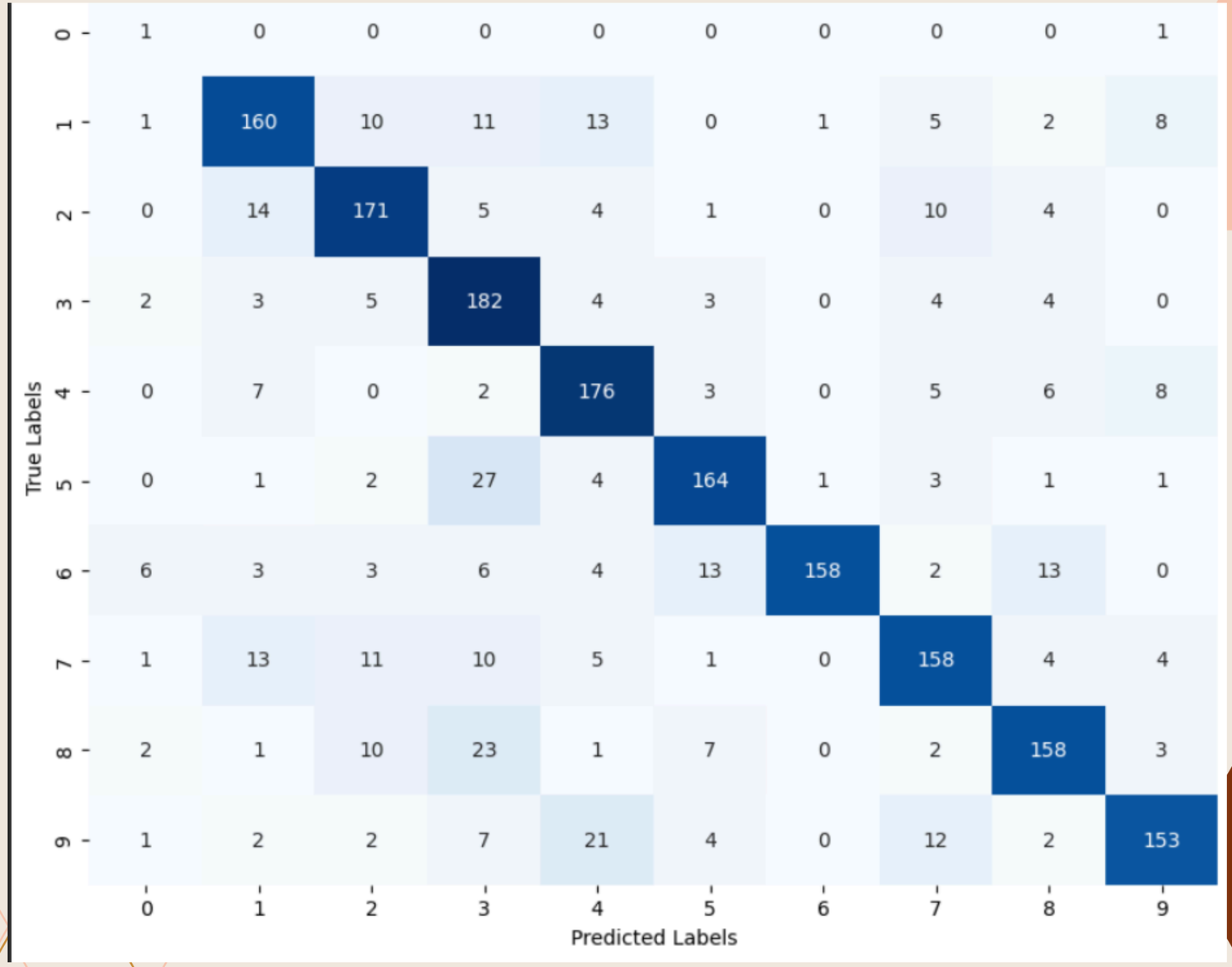
Elegiremos el modelo 5 como el ganador debido a su Accuracy y sus graficas, con este entrenaremos un nuevo modelo ahora sin separación en el Train y evaluandolo en el test



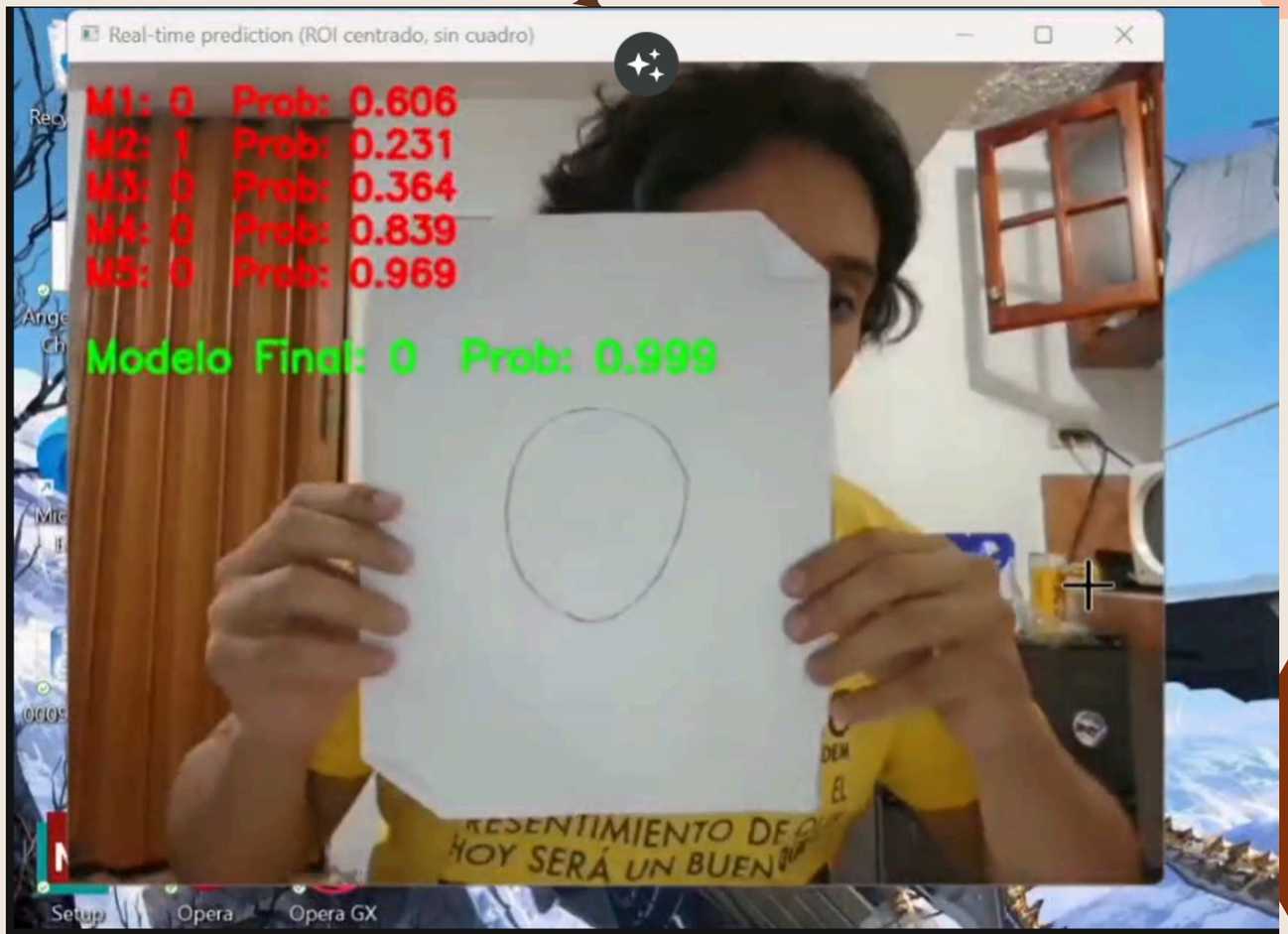



```
Test Accuracy: 93.35%
Test Loss    : 0.2483
```

Matriz de Confusión



Demostracion en tiempo real



Conclusion

Viendo la matriz, hay números que predijo mejor que otros, como 6, que tiene un 0.99 de precisión, los demas numeros tuvieron alrededor de un 0.78 a 0.86, siendo las unicas excepciones el numero 3 con 0.67 y el numero 0.

Estos numeros nos indican que claramente este modelo es muy bueno para predecir cualquier nunmero y que rara vez falla, por lo que podemos concluir que este proyecto fue un exito.

Classification Report:

	precision	recall	f1-score	support
0	0.07	0.50	0.12	2
1	0.78	0.76	0.77	211
2	0.80	0.82	0.81	209
3	0.67	0.88	0.76	207
4	0.76	0.85	0.80	207
5	0.84	0.80	0.82	204
6	0.99	0.76	0.86	208
7	0.79	0.76	0.77	207
8	0.81	0.76	0.79	207
9	0.86	0.75	0.80	204
accuracy			0.79	1866
macro avg	0.74	0.76	0.73	1866
weighted avg	0.81	0.79	0.80	1866