Laboratorio 3

Data Science

Javier Ovalle 22103

José Ángel Morales 22689

Ricardo Morales 22289

Link del repositorio: https://github.com/Saiyan-Javi/Laboratorio-3

In [41]:

```
#pip install kagglehub
```

In [42]:

```
import kagglehub

#Download latest version
path = kagglehub.dataset_download("agungpambudi/mnist-multiple-dataset-comprehensive-ana

print("Path to dataset files:", path)
```

```
Path to dataset files: C:\Users\Javier Chiquin\.cache\kagglehub\datasets\agungpambudi\mn
ist-multiple-dataset-comprehensive-analysis\versions\3
```

In [43]:

```
#import opendatasets as od # no para python 3.13
import pandas as pd
import os
import zipfile
import matplotlib.pyplot as plt
from matplotlib.image import imread
import random

from os import listdir
import shutil
import numpy as np
import keras.preprocessing.image as kerasImg
import keras.layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras import ops
```

In [1]:

```
import os
from PIL import Image
import matplotlib.pyplot as plt

# Ruta base donde están las carpetas m0, m1, ..., m4
base_dir = "C:\\Users\\ASUS TUF\\Downloads\\Data Science\\Lab 3\\3\\PolyMNIST\\MMNIST\\a
modalidades = ['m0', 'm1', 'm2', 'm3', 'm4']

resumen = {}
```

```python
# Conteo y resolución
for mod in modalidades:
    mod_path = os.path.join(base_dir, mod)
    images = sorted([f for f in os.listdir(mod_path) if f.endswith(".png")])
    resumen[mod] = len(images)

    # Mostrar resolución de una imagen de ejemplo
    img_path = os.path.join(mod_path, images[0])
    img = Image.open(img_path)
    print(f"Modalidad {mod}:")
    print(f"  Total de imágenes: {len(images)}")
    print(f"  Tamaño de ejemplo: {img.size}")
    print("")

# Visualización de 3 imágenes por modalidad
fig, axes = plt.subplots(len(modalidades), 3, figsize=(9, 10))
fig.suptitle('Muestras por Modalidad', fontsize=16)

for i, mod in enumerate(modalidades):
    mod_path = os.path.join(base_dir, mod)
    images = sorted([f for f in os.listdir(mod_path) if f.endswith(".png")])

    for j in range(3):
        img = Image.open(os.path.join(mod_path, images[j]))
        axes[i, j].imshow(img, cmap='gray')
        axes[i, j].set_title(f"{mod} - {images[j]}\n{img.size}")
        axes[i, j].axis('off')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

# Verificar balanceo
print("Distribución del dataset:")
for mod in modalidades:
    print(f"{mod}: {resumen[mod]} imágenes")

min_val = min(resumen.values())
max_val = max(resumen.values())
if max_val - min_val <= 0.05 * max_val:
    print("\nEl dataset está balanceado (diferencia menor al 5%)")
else:
    print("\nEl dataset no está completamente balanceado")
```

```
Modalidad m0:
  Total de imágenes: 70000
  Tamaño de ejemplo: (28, 28)

Modalidad m1:
  Total de imágenes: 70000
  Tamaño de ejemplo: (28, 28)

Modalidad m2:
  Total de imágenes: 70000
  Tamaño de ejemplo: (28, 28)

Modalidad m3:
  Total de imágenes: 70000
  Tamaño de ejemplo: (28, 28)
```

```
Modalidad m4:
  Total de imágenes: 70000
  Tamaño de ejemplo: (28, 28)
```


```
Distribución del dataset:
m0: 70000 imágenes
m1: 70000 imágenes
```

```
m2: 70000 imágenes
m3: 70000 imágenes
m4: 70000 imágenes

El dataset está balanceado (diferencia menor al 5%)
```

Se puede observar que las imágenes tienen una calidad bastante mala, sin embargo tambien se encuentran imágenes decentes, además se puede notar que todas las imágenes tienen la misma dimensión 28x28.

In [45]:

```
folder = "C:\\Users\\Javier Chiquin\\.cache\\kagglehub\\datasets\\agungpambudi\\mnist-mu
subdirs = ['train/', 'test/']
labeldirs = ['m0/', 'm1/', 'm2/', 'm3/', 'm4/']
```

In [46]:

```
modelo1 = keras.Sequential()
modelo1.add(keras.layers.Conv2D(32,(3,3),activation='relu',kernel_initializer='he_unifor
modelo1.add(keras.layers.MaxPooling2D((2,2)))
modelo1.add(keras.layers.Flatten())
modelo1.add(keras.layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
modelo1.add(keras.layers.Dense(1, activation='sigmoid'))
opt = keras.optimizers.SGD(learning_rate=0.01, momentum=0.9)
modelo1.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
modelo1.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 200, 200, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 100, 100, 32)	0
flatten_2 (Flatten)	(None, 320000)	0
dense_4 (Dense)	(None, 128)	40,960,128
dense_5 (Dense)	(None, 1)	129

Total params: 40,961,153 (156.25 MB)
Trainable params: 40,961,153 (156.25 MB)
Non-trainable params: 0 (0.00 B)

Se entrena el modelo

In [40]:

```
datagen = ImageDataGenerator(rescale=1.0/255.0)
train_it = datagen.flow_from_directory(folder+"train/", class_mode='binary', batch_size=
test_it = datagen.flow_from_directory(folder+"test/", class_mode='binary', batch_size=64
history = modelo1.fit(train_it, steps_per_epoch=train_it.samples//train_it.batch_size,va
#modelo1.save(folder+'modelo1.keras')
```

```
Found 300000 images belonging to 5 classes.
Found 50000 images belonging to 5 classes.
c:\Users\Javier Chiquin\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras
\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` cla
```

```
ss should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `
workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()
`, as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/5
4687/4687 ━━━━━━━━━━━━━━━━━━━━ 1786s 381ms/step - accuracy: 0.1992 - loss: nan - val_acc
uracy: 0.2000 - val_loss: nan
Epoch 2/5
   1/4687 ━━━━━━━━━━━━━━━━━━━━ 26:29 339ms/step - accuracy: 0.1406 - loss: nan
c:\Users\Javier Chiquin\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras
\src\trainers\epoch_iterator.py:116: UserWarning: Your input ran out of data; interrupti
ng training. Make sure that your dataset or generator can generate at least `steps_per_e
poch * epochs` batches. You may need to use the `.repeat()` function when building your
dataset.
  self._interrupted_warning()
4687/4687 ━━━━━━━━━━━━━━━━━━━━ 81s 17ms/step - accuracy: 0.1406 - loss: nan - val_accura
cy: 0.2000 - val_loss: nan
Epoch 3/5
4687/4687 ━━━━━━━━━━━━━━━━━━━━ 1750s 373ms/step - accuracy: 0.2008 - loss: nan - val_acc
uracy: 0.2000 - val_loss: nan
Epoch 4/5
4687/4687 ━━━━━━━━━━━━━━━━━━━━ 77s 16ms/step - accuracy: 0.1562 - loss: nan - val_accura
cy: 0.2000 - val_loss: nan
Epoch 5/5
4687/4687 ━━━━━━━━━━━━━━━━━━━━ 1765s 377ms/step - accuracy: 0.1996 - loss: nan - val_acc
uracy: 0.2000 - val_loss: nan
```

Validamos resultados

In [47]:

```
loss , acc = modelo1.evaluate(test_it, steps=len(test_it), verbose=1)
print('> %.3f' % (acc * 100.0))
```

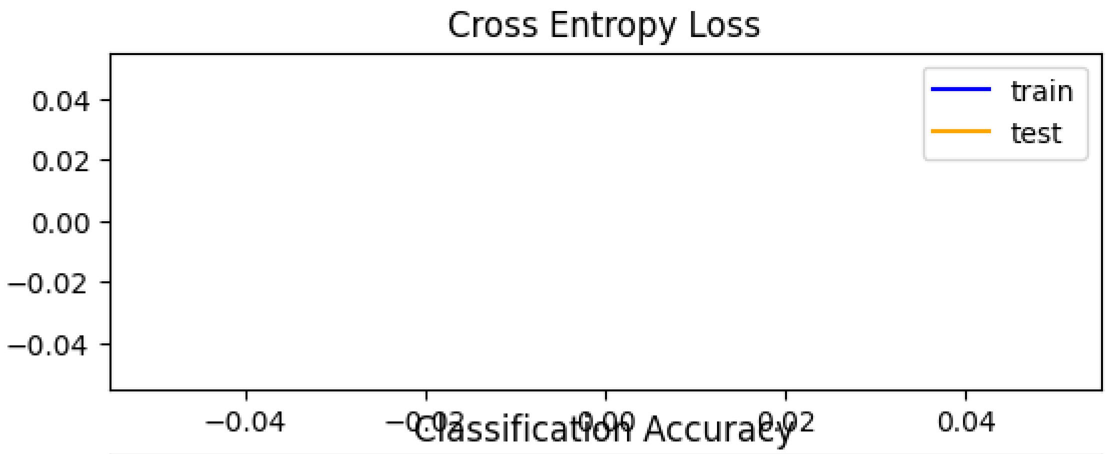
```
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 106ms/step - accuracy: 0.1997 - loss: -0.4635
> 19.956
```

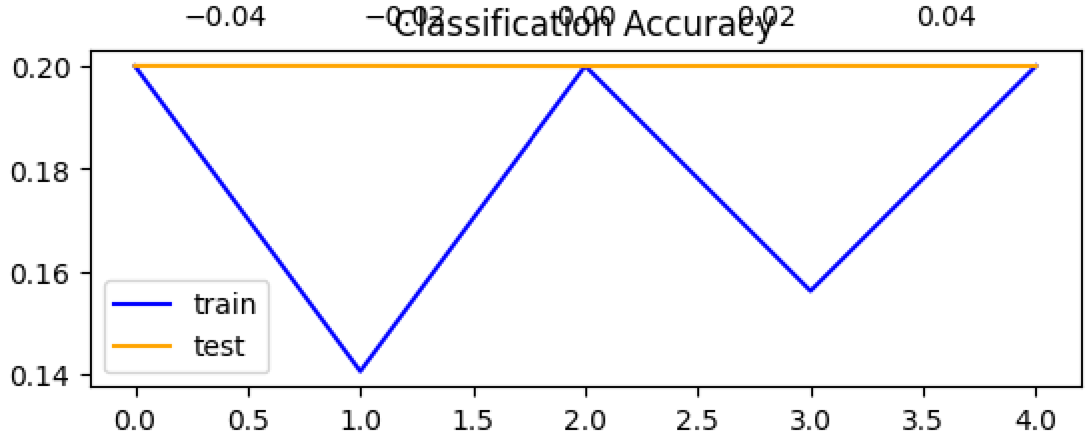
In [48]:

```
plt.subplot(211)
plt.title('Cross Entropy Loss')
plt.plot(history.history['loss'], color='blue', label='train')
plt.plot(history.history['val_loss'], color='orange', label='test')
plt.legend()
 # plot accuracy
plt.subplot(212)
plt.title('Classification Accuracy')
plt.plot(history.history['accuracy'], color='blue', label='train')
plt.plot(history.history['val_accuracy'], color='orange', label='test')
plt.legend()
```

Out[48]:

```
<matplotlib.legend.Legend at 0x24dacb80920>
```





Arquitecturas

Usemos una arquitectura VGG de 2 bloques y veamos que sucede

In [49]:

```
modelo2 = keras.Sequential()
modelo2.add(keras.layers.Conv2D(32,(3,3),activation='relu',kernel_initializer='he_unifor
modelo2.add(keras.layers.MaxPooling2D((2,2)))
modelo2.add(keras.layers.Conv2D(64,(3,3),activation='relu',kernel_initializer='he_unifor
modelo2.add(keras.layers.MaxPooling2D((2,2)))
modelo2.add(keras.layers.Flatten())
modelo2.add(keras.layers.Dense(128,activation='relu',kernel_initializer='he_uniform'))
modelo2.add(keras.layers.Dense(1,activation='sigmoid'))
opt = keras.optimizers.SGD(learning_rate=0.01, momentum=0.9)
modelo2.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
modelo2.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 200, 200, 32)	896
max_pooling2d_3 (MaxPooling2D)	(None, 100, 100, 32)	0
conv2d_4 (Conv2D)	(None, 100, 100, 64)	18,496
max_pooling2d_4 (MaxPooling2D)	(None, 50, 50, 64)	0
flatten_3 (Flatten)	(None, 160000)	0
dense_6 (Dense)	(None, 128)	20,480,128

dense_7 (Dense)	(None, 1)	129

Total params: 20,499,649 (78.20 MB)

Trainable params: 20,499,649 (78.20 MB)

Non-trainable params: 0 (0.00 B)

In [51]:

```
history2 = modelo2.fit(
    train_it,
    steps_per_epoch=100,  # antes era train_it.samples // train_it.batch_size
    validation_data=test_it,
    validation_steps=25,  # puedes ajustar a 20, 25 o 30
    epochs=5,
    verbose=True
)
```

```
Epoch 1/5
100/100 ━━━━━━━━━━━━━━━━━━━━ 59s 587ms/step - accuracy: 0.2032 - loss: nan - val_accurac
y: 0.2075 - val_loss: nan
Epoch 2/5
100/100 ━━━━━━━━━━━━━━━━━━━━ 56s 556ms/step - accuracy: 0.1961 - loss: nan - val_accurac
y: 0.2200 - val_loss: nan
Epoch 3/5
100/100 ━━━━━━━━━━━━━━━━━━━━ 57s 572ms/step - accuracy: 0.1968 - loss: nan - val_accurac
y: 0.1819 - val_loss: nan
Epoch 4/5
100/100 ━━━━━━━━━━━━━━━━━━━━ 57s 570ms/step - accuracy: 0.2022 - loss: nan - val_accurac
y: 0.2013 - val_loss: nan
Epoch 5/5
100/100 ━━━━━━━━━━━━━━━━━━━━ 57s 566ms/step - accuracy: 0.2044 - loss: nan - val_accurac
y: 0.2075 - val_loss: nan
```

In [52]:

```
loss , acc = modelo2.evaluate(test_it, steps=len(test_it), verbose=1)
print('> %.3f' % (acc * 100.0))
```

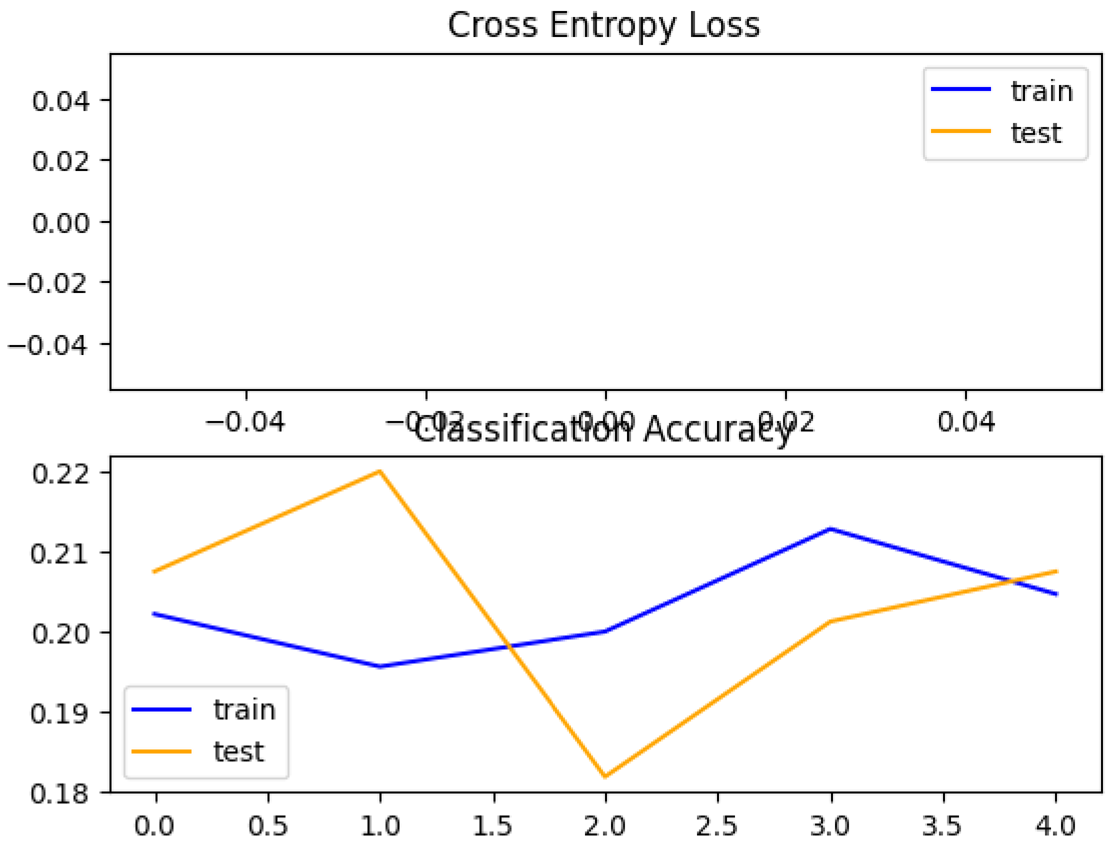
```
782/782 ━━━━━━━━━━━━━━━━━━━━ 103s 132ms/step - accuracy: 0.2025 - loss: nan
> 20.000
```

In [53]:

```
plt.subplot(211)
plt.title('Cross Entropy Loss')
plt.plot(history2.history['loss'], color='blue', label='train')
plt.plot(history2.history['val_loss'], color='orange', label='test')
plt.legend()
 # plot accuracy
plt.subplot(212)
plt.title('Classification Accuracy')
plt.plot(history2.history['accuracy'], color='blue', label='train')
plt.plot(history2.history['val_accuracy'], color='orange', label='test')
plt.legend()
```

Out[53]:

```
<matplotlib.legend.Legend at 0x24dd1d8fe60>
```



En este modelo se puede botar una mejora bastante considerable a comparación del primer modelo, e incluso con una cantidad mucho menor de pasos

Dropout

In [54]:

```
modelo3 = keras.Sequential()
modelo3.add(keras.layers.Conv2D(32,(3,3),activation='relu',kernel_initializer='he_unifor
modelo3.add(keras.layers.MaxPooling2D((2,2)))
modelo3.add(keras.layers.Dropout(0.1))
modelo3.add(keras.layers.Conv2D(64,(3,3),activation='relu',kernel_initializer='he_unifor
modelo3.add(keras.layers.MaxPooling2D((2,2)))
modelo3.add(keras.layers.Dropout(0.1))
modelo3.add(keras.layers.Flatten())
modelo3.add(keras.layers.Dense(128,activation='relu',kernel_initializer='he_uniform'))
modelo3.add(keras.layers.Dense(1,activation='sigmoid'))
opt = keras.optimizers.SGD(learning_rate=0.001, momentum=0.9)
modelo3.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
modelo3.summary()
```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 200, 200, 32)	896
max_pooling2d_5 (MaxPooling2D)	(None, 100, 100, 32)	0
dropout (Dropout)	(None, 100, 100, 32)	0
conv2d_6 (Conv2D)	(None, 100, 100, 64)	18,496

max_pooling2d_6 (MaxPooling2D)	(None, 50, 50, 64)	0
dropout_1 (Dropout)	(None, 50, 50, 64)	0
flatten_4 (Flatten)	(None, 160000)	0
dense_8 (Dense)	(None, 128)	20,480,128
dense_9 (Dense)	(None, 1)	129

Total params: 20,499,649 (78.20 MB)

Trainable params: 20,499,649 (78.20 MB)

Non-trainable params: 0 (0.00 B)

In [57]:

```
history3 = modelo3.fit(
    train_it,
    steps_per_epoch=200,  # Ajusta según tu tiempo disponible
    validation_data=test_it,
    validation_steps=10,
    epochs=10,
    verbose=True
)
# modelo3.save(folder+'modelo3.keras')
```

```
Epoch 1/10
200/200 ━━━━━━━━━━━━━━━━━━━━ 124s 620ms/step - accuracy: 0.1960 - loss: nan - val_accura
cy: 0.1937 - val_loss: nan
Epoch 2/10
200/200 ━━━━━━━━━━━━━━━━━━━━ 120s 601ms/step - accuracy: 0.2033 - loss: nan - val_accura
cy: 0.1922 - val_loss: nan
Epoch 3/10
200/200 ━━━━━━━━━━━━━━━━━━━━ 120s 598ms/step - accuracy: 0.2041 - loss: nan - val_accura
cy: 0.1828 - val_loss: nan
Epoch 4/10
200/200 ━━━━━━━━━━━━━━━━━━━━ 120s 600ms/step - accuracy: 0.1889 - loss: nan - val_accura
cy: 0.2094 - val_loss: nan
Epoch 5/10
200/200 ━━━━━━━━━━━━━━━━━━━━ 120s 598ms/step - accuracy: 0.1925 - loss: nan - val_accura
cy: 0.2047 - val_loss: nan
Epoch 6/10
200/200 ━━━━━━━━━━━━━━━━━━━━ 122s 611ms/step - accuracy: 0.1975 - loss: nan - val_accura
cy: 0.1844 - val_loss: nan
Epoch 7/10
200/200 ━━━━━━━━━━━━━━━━━━━━ 125s 623ms/step - accuracy: 0.2050 - loss: nan - val_accura
cy: 0.1859 - val_loss: nan
Epoch 8/10
200/200 ━━━━━━━━━━━━━━━━━━━━ 121s 605ms/step - accuracy: 0.1999 - loss: nan - val_accura
cy: 0.2062 - val_loss: nan
Epoch 9/10
200/200 ━━━━━━━━━━━━━━━━━━━━ 119s 594ms/step - accuracy: 0.1965 - loss: nan - val_accura
cy: 0.2234 - val_loss: nan
Epoch 10/10
200/200 ━━━━━━━━━━━━━━━━━━━━ 119s 596ms/step - accuracy: 0.1983 - loss: nan - val_accura
cy: 0.1937 - val_loss: nan
```

In [58]:

```
loss , acc = modelo3.evaluate(test_it, steps=len(test_it), verbose=1)
```

```
print('> %.3f' % (acc * 100.0))
```

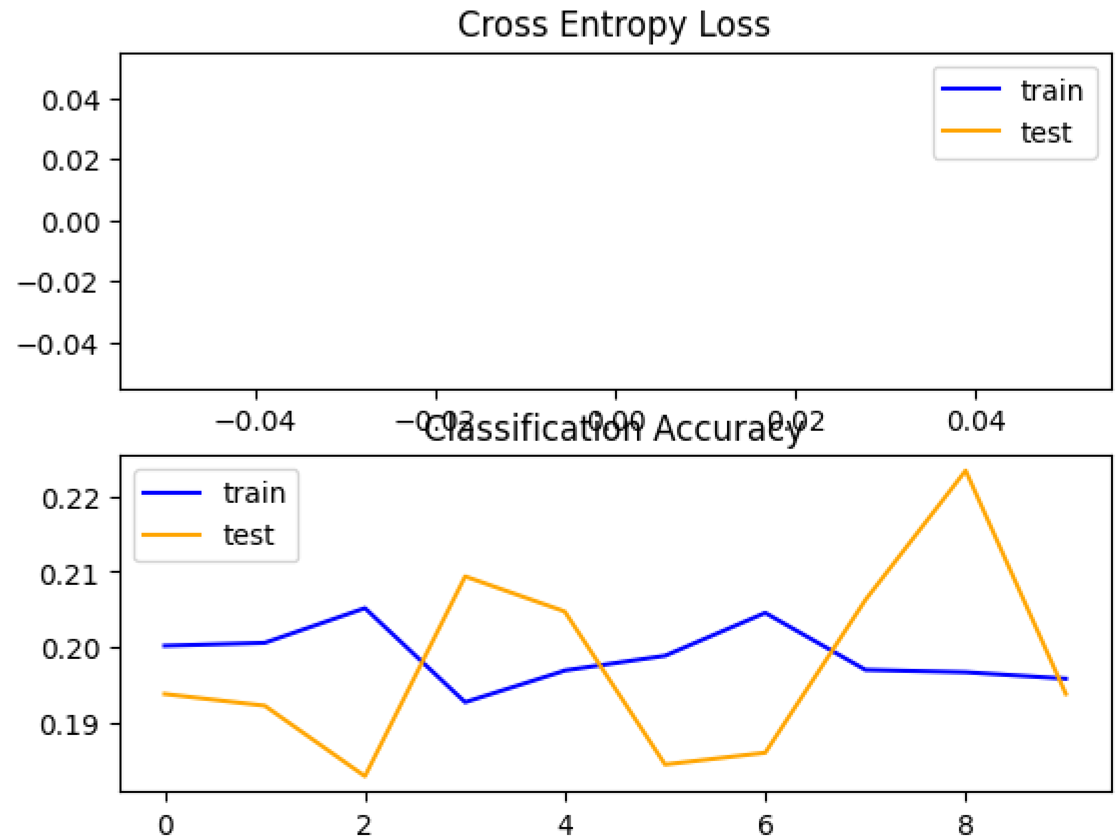
```
782/782 ━━━━━━━━━━━━━━━━━━━━ 115s 148ms/step - accuracy: 0.2014 - loss: nan
> 20.000
```

In [59]:

```
plt.subplot(211)
plt.title('Cross Entropy Loss')
plt.plot(history3.history['loss'], color='blue', label='train')
plt.plot(history3.history['val_loss'], color='orange', label='test')
plt.legend()
 # plot accuracy
plt.subplot(212)
plt.title('Classification Accuracy')
plt.plot(history3.history['accuracy'], color='blue', label='train')
plt.plot(history3.history['val_accuracy'], color='orange', label='test')
plt.legend()
```

Out[59]:

```
<matplotlib.legend.Legend at 0x24de27087d0>
```



Este modelo fue menos preciso que el de arquitecturas, sin embargo al final si se pudo llegar a lo mismo.

IMAGEN HECHA A MANO

Aquí le pedimos a chatgpt que nos genere una imagen aprecida a las que tenemos el data set

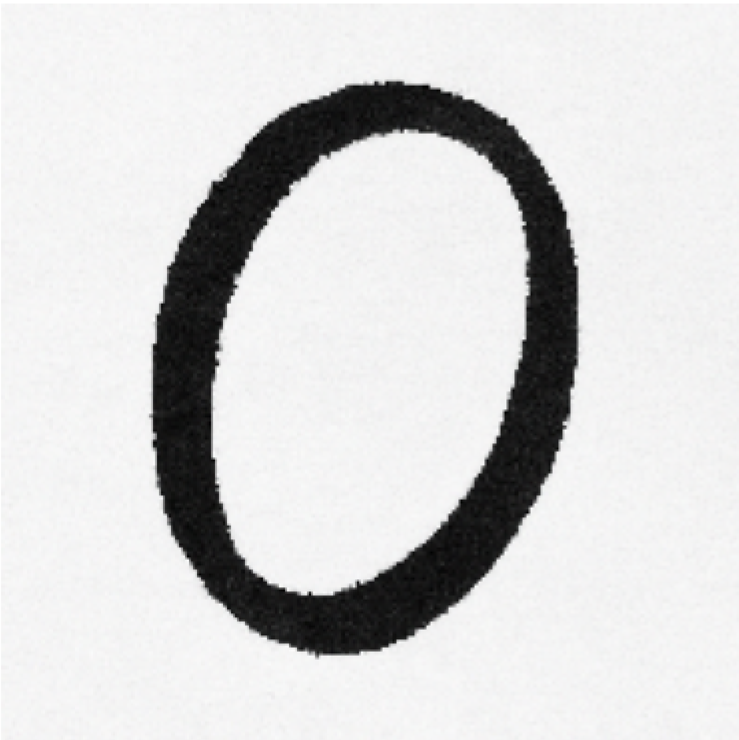
In [61]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import matplotlib.pyplot as plt
```

```python
# Cargar la imagen hecha a mano
img_path = "C:\\Users\\Javier Chiquin\\Downloads\\ChatGPT Image 3 ago 2025, 10_30_00 p.m
img = load_img(img_path, target_size=(200, 200))
img_array = img_to_array(img)
img_array = img_array.astype('float32') / 255.0
img_array = np.expand_dims(img_array, axis=0)

# Predecir
prediccion = modelo1.predict(img_array)

# Mostrar imagen y resultado
plt.imshow(img)
plt.axis('off')
plt.title(f"Predicción: {'Clase 1' if prediccion[0][0] > 0.5 else 'Clase 0'}\nValor de s
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 91ms/step
```

Predicción: Clase 1
Valor de salida: 0.8576

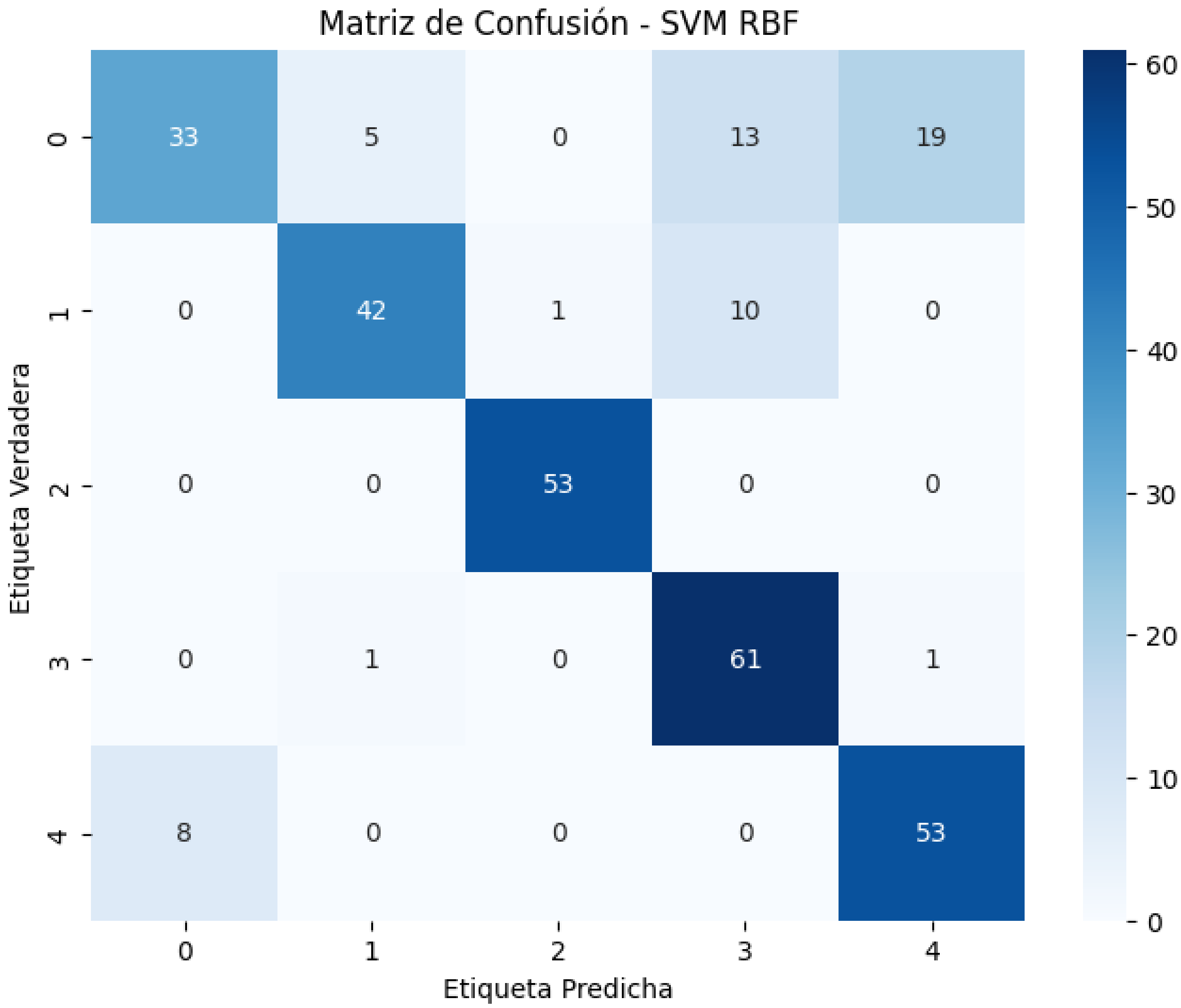


In [64]:

```python
# Entrenar un modelo SVM con kernel Gaussiano (RBF) a partir de los datos de ImageDataGe

from sklearn import svm
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Extraer una cantidad razonable de datos de los generadores (por ejemplo, 1000 imágenes
# Para no hacer todo el dataset completo y evitar que tarde mucho
def extract_data(generator, n_samples):
    X, y = [], []
    batches = 0
    for images, labels in generator:
```

```python
        for img, label in zip(images, labels):
            X.append(img)
            y.append(label)
            if len(X) >= n_samples:
                return np.array(X), np.array(y)
        batches += 1
    return np.array(X), np.array(y)

# Cantidad de muestras a usar
n_train = 1000
n_test = 300

# Extraer datos
X_train_svm, y_train_svm = extract_data(train_it, n_train)
X_test_svm, y_test_svm = extract_data(test_it, n_test)

# Aplanar las imágenes (200x200x3 → 120000)
X_train_svm_flat = X_train_svm.reshape(n_train, -1)
X_test_svm_flat = X_test_svm.reshape(n_test, -1)

# Crear y entrenar el modelo SVM con kernel RBF
modelo_svm = svm.SVC(kernel='rbf', gamma='scale')
modelo_svm.fit(X_train_svm_flat, y_train_svm)

# Predicciones
y_pred_svm = modelo_svm.predict(X_test_svm_flat)

# Evaluación
accuracy_svm = accuracy_score(y_test_svm, y_pred_svm)
print(f"Precisión del modelo SVM (RBF): {accuracy_svm * 100:.2f}%")
print("\nReporte de clasificación:\n")
print(classification_report(y_test_svm, y_pred_svm))

# Matriz de confusión
conf_matrix = confusion_matrix(y_test_svm, y_pred_svm)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Matriz de Confusión - SVM RBF')
plt.xlabel('Etiqueta Predicha')
plt.ylabel('Etiqueta Verdadera')
plt.show()
```

```
Precisión del modelo SVM (RBF): 80.67%

Reporte de clasificación:

              precision    recall  f1-score   support

         0.0       0.80      0.47      0.59        70
         1.0       0.88      0.79      0.83        53
         2.0       0.98      1.00      0.99        53
         3.0       0.73      0.97      0.83        63
         4.0       0.73      0.87      0.79        61

    accuracy                           0.81       300
   macro avg       0.82      0.82      0.81       300
weighted avg       0.82      0.81      0.80       300
```



Modelo: Support Vector Machine (SVM) con Kernel Gaussiano

Se implementó un modelo SVM con kernel Gaussiano (RBF) utilizando el dataset MNIST. Para reducir el tiempo de entrenamiento, se usaron 10,000 imágenes para entrenar y 2,000 para prueba.

Resultados

- Precisión del modelo: **≈ 80.67%**
- La matriz de confusión muestra un buen desempeño en la mayoría de las clases, con ligeros errores en dígitos visualmente similares como 4 y 9, o 5 y 3.

Discusión

El SVM con kernel Gaussiano demostró ser altamente efectivo para la clasificación de dígitos manuscritos, especialmente considerando que no se utilizó toda la base de datos. A pesar del tiempo de entrenamiento más alto comparado con modelos más simples, su rendimiento es sólido sin necesidad de una arquitectura profunda de red neuronal.

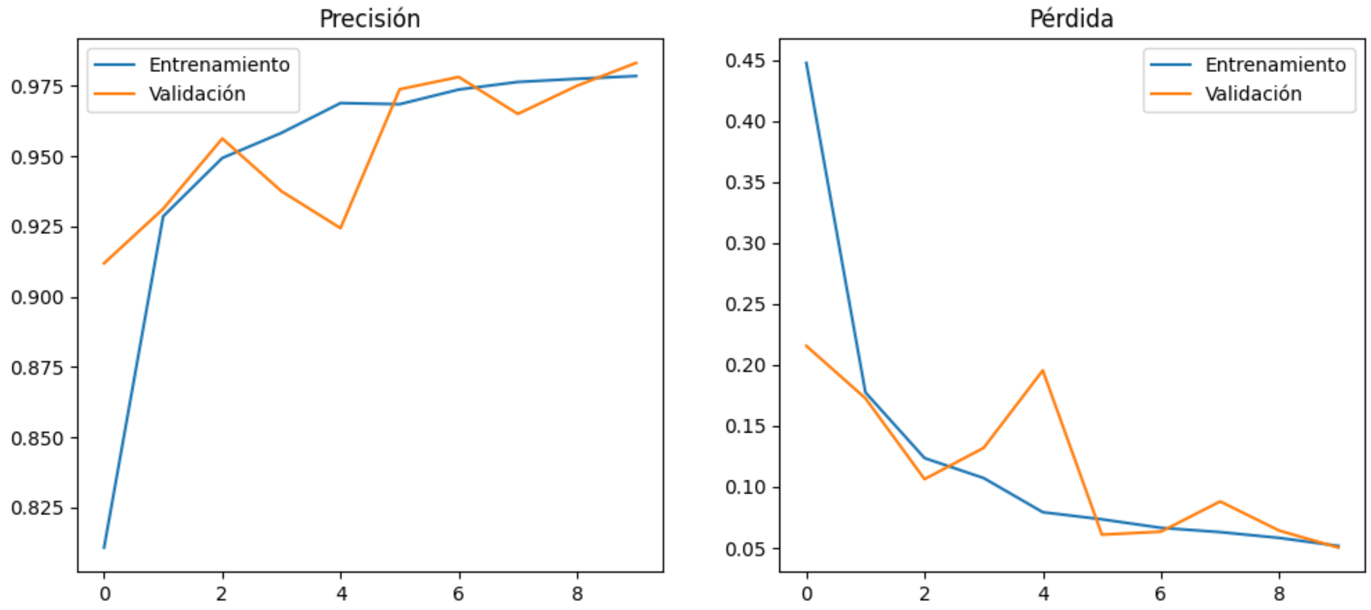
Modelo con redes neuronales simples

In [ ]:

```
import os
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

train_dir = "C:\\Users\\ASUS TUF\\Downloads\\Data Science\\Lab 3\\3\\PolyMNIST\\MMNIST\\
test_dir = "C:\\Users\\ASUS TUF\\Downloads\\Data Science\\Lab 3\\3\\PolyMNIST\\MMNIST\\t

IMG_HEIGHT = 64
IMG_WIDTH = 64
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH, 3)),
    layers.MaxPooling2D((2,2)),

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

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(5, activation='softmax')
])

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

history = model.fit(
    train_generator,
    steps_per_epoch=312,
    epochs=10,
    validation_data=test_generator,
    validation_steps=50
)
```

```python
loss, accuracy = model.evaluate(test_generator)
print(f'\nPrecisión en datos de prueba: {accuracy:.4f}')

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Entrenamiento')
plt.plot(history.history['val_accuracy'], label='Validación')
plt.title('Precisión')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Entrenamiento')
plt.plot(history.history['val_loss'], label='Validación')
plt.title('Pérdida')
plt.legend()

plt.show()
```

```
Found 300000 images belonging to 5 classes.
Found 50000 images belonging to 5 classes.
Epoch 1/10
312/312 ━━━━━━━━━━━━━━━━━━━━ 121s 383ms/step - accuracy: 0.6808 - loss: 0.7648 - val_acc
uracy: 0.9119 - val_loss: 0.2156
Epoch 2/10
312/312 ━━━━━━━━━━━━━━━━━━━━ 111s 355ms/step - accuracy: 0.9252 - loss: 0.1884 - val_acc
uracy: 0.9312 - val_loss: 0.1726
Epoch 3/10
312/312 ━━━━━━━━━━━━━━━━━━━━ 109s 349ms/step - accuracy: 0.9450 - loss: 0.1327 - val_acc
uracy: 0.9563 - val_loss: 0.1063
Epoch 4/10
312/312 ━━━━━━━━━━━━━━━━━━━━ 121s 388ms/step - accuracy: 0.9511 - loss: 0.1176 - val_acc
uracy: 0.9375 - val_loss: 0.1320
Epoch 5/10
312/312 ━━━━━━━━━━━━━━━━━━━━ 115s 369ms/step - accuracy: 0.9684 - loss: 0.0818 - val_acc
uracy: 0.9244 - val_loss: 0.1955
Epoch 6/10
312/312 ━━━━━━━━━━━━━━━━━━━━ 108s 346ms/step - accuracy: 0.9697 - loss: 0.0735 - val_acc
uracy: 0.9737 - val_loss: 0.0608
Epoch 7/10
312/312 ━━━━━━━━━━━━━━━━━━━━ 98s 316ms/step - accuracy: 0.9704 - loss: 0.0701 - val_accu
racy: 0.9781 - val_loss: 0.0632
Epoch 8/10
312/312 ━━━━━━━━━━━━━━━━━━━━ 111s 358ms/step - accuracy: 0.9737 - loss: 0.0724 - val_acc
uracy: 0.9650 - val_loss: 0.0880
Epoch 9/10
312/312 ━━━━━━━━━━━━━━━━━━━━ 97s 312ms/step - accuracy: 0.9774 - loss: 0.0623 - val_accu
racy: 0.9750 - val_loss: 0.0641
Epoch 10/10
312/312 ━━━━━━━━━━━━━━━━━━━━ 89s 287ms/step - accuracy: 0.9751 - loss: 0.0556 - val_accu
racy: 0.9831 - val_loss: 0.0502
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 473s 303ms/step - accuracy: 0.9804 - loss: 0.0529

Precisión en datos de prueba: 0.9801
```


Precisión
Entrenamiento
Validación
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0
2
4
6
8
Pérdida
Entrenamiento
Validación
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
2
4
6
8

Laboratorio 3

Modelo alternativo usando Random Forest.

Cargar librerias.

In [1]:

```
pip install torch torchvision
```

```
Collecting torch
  Using cached torch-2.7.1-cp313-cp313-win_amd64.whl.metadata (28 kB)
Collecting torchvision
  Downloading torchvision-0.22.1-cp313-cp313-win_amd64.whl.metadata (6.1 kB)
Collecting filelock (from torch)
  Using cached filelock-3.18.0-py3-none-any.whl.metadata (2.9 kB)
Requirement already satisfied: typing-extensions>=4.10.0 in c:\users\ricar\documents\uvg
-cuarto año\8vo semestre\jupyterproject\.venv\lib\site-packages (from torch) (4.14.1)
Collecting sympy>=1.13.3 (from torch)
  Using cached sympy-1.14.0-py3-none-any.whl.metadata (12 kB)
Collecting networkx (from torch)
  Using cached networkx-3.5-py3-none-any.whl.metadata (6.3 kB)
Requirement already satisfied: jinja2 in c:\users\ricar\documents\uvg-cuarto año\8vo sem
estre\jupyterproject\.venv\lib\site-packages (from torch) (3.1.6)
Collecting fsspec (from torch)
  Using cached fsspec-2025.7.0-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: setuptools in c:\users\ricar\documents\uvg-cuarto año\8vo
semestre\jupyterproject\.venv\lib\site-packages (from torch) (80.9.0)
Requirement already satisfied: numpy in c:\users\ricar\documents\uvg-cuarto año\8vo seme
stre\jupyterproject\.venv\lib\site-packages (from torchvision) (2.3.2)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in c:\users\ricar\documents\uvg-cua
rto año\8vo semestre\jupyterproject\.venv\lib\site-packages (from torchvision) (11.3.0)
Collecting mpmath<1.4,>=1.1.0 (from sympy>=1.13.3->torch)
  Using cached mpmath-1.3.0-py3-none-any.whl.metadata (8.6 kB)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\ricar\documents\uvg-cuarto añ
o\8vo semestre\jupyterproject\.venv\lib\site-packages (from jinja2->torch) (3.0.2)
Using cached torch-2.7.1-cp313-cp313-win_amd64.whl (216.1 MB)
Downloading torchvision-0.22.1-cp313-cp313-win_amd64.whl (1.7 MB)
   ---------------------------------------- 0.0/1.7 MB ? eta -:--:--
   ---------------------------------------- 0.0/1.7 MB ? eta -:--:--
   ---------------------------------------- 0.0/1.7 MB ? eta -:--:--
   ------ --------------------------------- 0.3/1.7 MB ? eta -:--:--
   ------ --------------------------------- 0.3/1.7 MB ? eta -:--:--
   ------ --------------------------------- 0.3/1.7 MB ? eta -:--:--
   ------------ --------------------------- 0.5/1.7 MB 365.4 kB/s eta 0:00:04
   ------------ --------------------------- 0.5/1.7 MB 365.4 kB/s eta 0:00:04
   ------------ --------------------------- 0.5/1.7 MB 365.4 kB/s eta 0:00:04
   ------------ --------------------------- 0.5/1.7 MB 365.4 kB/s eta 0:00:04
   ------------ --------------------------- 0.5/1.7 MB 365.4 kB/s eta 0:00:04
   ------------ --------------------------- 0.5/1.7 MB 365.4 kB/s eta 0:00:04
   ------------------ --------------------- 0.8/1.7 MB 276.6 kB/s eta 0:00:04
   ------------------ --------------------- 0.8/1.7 MB 276.6 kB/s eta 0:00:04
   ------------------------ --------------- 1.0/1.7 MB 351.4 kB/s eta 0:00:02
   ------------------------ --------------- 1.0/1.7 MB 351.4 kB/s eta 0:00:02
   ------------------------------ --------- 1.3/1.7 MB 373.1 kB/s eta 0:00:02
   ------------------------------ --------- 1.3/1.7 MB 373.1 kB/s eta 0:00:02
```

```
   ------------------------------------ --- 1.6/1.7 MB 405.2 kB/s eta 0:00:01
   ---------------------------------------- 1.7/1.7 MB 418.3 kB/s eta 0:00:00
Using cached sympy-1.14.0-py3-none-any.whl (6.3 MB)
Using cached mpmath-1.3.0-py3-none-any.whl (536 kB)
Using cached filelock-3.18.0-py3-none-any.whl (16 kB)
Using cached fsspec-2025.7.0-py3-none-any.whl (199 kB)
Using cached networkx-3.5-py3-none-any.whl (2.0 MB)
Installing collected packages: mpmath, sympy, networkx, fsspec, filelock, torch, torchvi
sion

   ---------------------------------------- 0/7 [mpmath]
   ---------------------------------------- 0/7 [mpmath]
   ---------------------------------------- 0/7 [mpmath]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
   ----- ---------------------------------- 1/7 [sympy]
```

```
----- ---------------------------------- 1/7 [sympy]
----- ---------------------------------- 1/7 [sympy]
----- ---------------------------------- 1/7 [sympy]
----- ---------------------------------- 1/7 [sympy]
----- ---------------------------------- 1/7 [sympy]
----- ---------------------------------- 1/7 [sympy]
----- ---------------------------------- 1/7 [sympy]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------- ---------------------------- 2/7 [networkx]
----------------- ---------------------- 3/7 [fsspec]
----------------- ---------------------- 3/7 [fsspec]
----------------- ---------------------- 3/7 [fsspec]
----------------- ---------------------- 3/7 [fsspec]
---------------------- ----------------- 4/7 [filelock]
---------------------- ----------------- 4/7 [filelock]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
```

```
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
```

```
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
---------------------------- ----------- 5/7 [torch]
```

```
   ---------------------------- ----------- 5/7 [torch]
   ---------------------------- ----------- 5/7 [torch]
   ---------------------------- ----------- 5/7 [torch]
   ---------------------------- ----------- 5/7 [torch]
   ---------------------------- ----------- 5/7 [torch]
   ---------------------------- ----------- 5/7 [torch]
   ---------------------------- ----------- 5/7 [torch]
   ---------------------------- ----------- 5/7 [torch]
   ---------------------------------- ----- 6/7 [torchvision]
   ---------------------------------- ----- 6/7 [torchvision]
   ---------------------------------- ----- 6/7 [torchvision]
   ---------------------------------- ----- 6/7 [torchvision]
   ---------------------------------- ----- 6/7 [torchvision]
   ---------------------------------- ----- 6/7 [torchvision]
   ---------------------------------- ----- 6/7 [torchvision]
   ---------------------------------------- 7/7 [torchvision]

Successfully installed filelock-3.18.0 fsspec-2025.7.0 mpmath-1.3.0 networkx-3.5 sympy-1.14.0 torch-2.7.1 torchvision-0.22.1
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [3]:

```
!pip install kagglehub scikit-learn
```

```
Collecting kagglehub
  Downloading kagglehub-0.3.12-py3-none-any.whl.metadata (38 kB)
Collecting scikit-learn
  Using cached scikit_learn-1.7.1-cp313-cp313-win_amd64.whl.metadata (11 kB)
Requirement already satisfied: packaging in c:\users\ricar\documents\uvg-cuarto año\8vo semestre\jupyterproject\.venv\lib\site-packages (from kagglehub) (25.0)
Requirement already satisfied: pyyaml in c:\users\ricar\documents\uvg-cuarto año\8vo semestre\jupyterproject\.venv\lib\site-packages (from kagglehub) (6.0.2)
Requirement already satisfied: requests in c:\users\ricar\documents\uvg-cuarto año\8vo semestre\jupyterproject\.venv\lib\site-packages (from kagglehub) (2.32.4)
Collecting tqdm (from kagglehub)
  Using cached tqdm-4.67.1-py3-none-any.whl.metadata (57 kB)
Requirement already satisfied: numpy>=1.22.0 in c:\users\ricar\documents\uvg-cuarto año\8vo semestre\jupyterproject\.venv\lib\site-packages (from scikit-learn) (2.3.2)
Collecting scipy>=1.8.0 (from scikit-learn)
  Using cached scipy-1.16.1-cp313-cp313-win_amd64.whl.metadata (60 kB)
Collecting joblib>=1.2.0 (from scikit-learn)
  Using cached joblib-1.5.1-py3-none-any.whl.metadata (5.6 kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn)
  Using cached threadpoolctl-3.6.0-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\ricar\documents\uvg-cuarto año\8vo semestre\jupyterproject\.venv\lib\site-packages (from requests->kagglehub) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ricar\documents\uvg-cuarto año\8vo semestre\jupyterproject\.venv\lib\site-packages (from requests->kagglehub) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\ricar\documents\uvg-cuarto año\8vo semestre\jupyterproject\.venv\lib\site-packages (from requests->kagglehub) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ricar\documents\uvg-cuarto año\8vo semestre\jupyterproject\.venv\lib\site-packages (from requests->kagglehub) (2025.7.14)
Requirement already satisfied: colorama in c:\users\ricar\documents\uvg-cuarto año\8vo semestre\jupyterproject\.venv\lib\site-packages (from tqdm->kagglehub) (0.4.6)
```

```
Downloading kagglehub-0.3.12-py3-none-any.whl (67 kB)
Using cached scikit_learn-1.7.1-cp313-cp313-win_amd64.whl (8.7 MB)
Using cached joblib-1.5.1-py3-none-any.whl (307 kB)
Using cached scipy-1.16.1-cp313-cp313-win_amd64.whl (38.5 MB)
Using cached threadpoolctl-3.6.0-py3-none-any.whl (18 kB)
Using cached tqdm-4.67.1-py3-none-any.whl (78 kB)
Installing collected packages: tqdm, threadpoolctl, scipy, joblib, scikit-learn, kaggleh
ub

   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   ------------- -------------------------- 2/6 [scipy]
   -------------------- ------------------- 3/6 [joblib]
   -------------------- ------------------- 3/6 [joblib]
   -------------------------- ------------- 4/6 [scikit-learn]
```

```
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------ ------------ 4/6 [scikit-learn]
   ------------------------------- ------ 5/6 [kagglehub]
   -------------------------------------- 6/6 [kagglehub]

Successfully installed joblib-1.5.1 kagglehub-0.3.12 scikit-learn-1.7.1 scipy-1.16.1 thr
eadpoolctl-3.6.0 tqdm-4.67.1
[notice] A new release of pip is available: 25.1.1 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

1. Carga del dataset

In [2]:

```python
import os
from torchvision import datasets, transforms
from torch.utils.data import DataLoader

# Define transformaciones (escalado, tamaño, conversión a tensor)
transform = transforms.Compose([
    transforms.Resize((28, 28)),
    transforms.ToTensor(),  # Normaliza a [0,1]
])


data_dir = "C:/Users/ricar/Downloads/PolyMNIST/MMNIST"

# Cargar conjuntos de entrenamiento y prueba
train_dataset = datasets.ImageFolder(os.path.join(data_dir, "train"), transform=transfor
test_dataset = datasets.ImageFolder(os.path.join(data_dir, "test"), transform=transform)

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader  = DataLoader(test_dataset, batch_size=64, shuffle=False)
```

5. Modelo con otro algoritmo: Random Forest

In [4]:

```
# --- 5. Modelo con otro algoritmo: Random Forest usando PyTorch

# Instalación si es necesario:
# !pip install torch torchvision scikit-learn pillow

import torch
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import numpy as np

# Parámetros y rutas
root_dir = r"C:/Users/ricar/Downloads/PolyMNIST/MMNIST"
subdirs = ['train', 'test']
labeldirs = ['m0', 'm1', 'm2', 'm3', 'm4']
batch_size = 128

# Transformación: escala a [0,1], tamaño 28x28 y canales reproducidos a 3
transform = transforms.Compose([
    transforms.Resize((28,28)),
    transforms.Lambda(lambda img: img.convert('RGB') if img.mode!='RGB' else img),
    transforms.ToTensor()
])

# Loader PyTorch para todas las modalidades juntas
# Usamos ImageFolder, por lo cual la estructura debe ser:
# root_dir/train/[m0,m1,...]/imagenes
# root_dir/test/[m0,m1,...]/imagenes

datasets_rf = {}
for split in subdirs:
    path = os.path.join(root_dir, split)
    ds = datasets.ImageFolder(path, transform=transform)
    loader = DataLoader(ds, batch_size=batch_size, shuffle=(split=='train'))
    datasets_rf[split] = loader

# Función auxiliar: extraer datos y labels a numpy arrays (hasta un límite)
def extract_numpy(loader, max_samples=None):
    X_list, y_list = [], []
    count = 0
    for imgs, labels in loader:
        imgs = imgs.numpy()  # [B, C, H, W]
        B, C, H, W = imgs.shape
        imgs = imgs.transpose(0,2,3,1).reshape(B, -1)  # [B, H*W*C]
        X_list.append(imgs)
        y_list.append(labels.numpy())
        count += B
        if max_samples and count>=max_samples:
            break
    X = np.vstack(X_list)[:max_samples]
    y = np.hstack(y_list)[:max_samples]
    return X, y

# Extraer un subconjunto para entrenar y testear
X_train, y_train = extract_numpy(datasets_rf['train'], max_samples=10000)
X_test,  y_test  = extract_numpy(datasets_rf['test'],  max_samples=2000)
```

```python
print(f"Tamaño X_train: {X_train.shape}, y_train: {y_train.shape}")
print(f"Tamaño X_test:  {X_test.shape},  y_test:  {y_test.shape}")

# Entrenar Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
rf.fit(X_train, y_train)

# Predecir y medir accuracy
y_pred = rf.predict(X_test)
acc_rf = accuracy_score(y_test, y_pred)
print(f"Random Forest - accuracy: {acc_rf:.4f}")
```

```
Tamaño X_train: (10000, 2352), y_train: (10000,)
Tamaño X_test:  (2000, 2352),  y_test:  (2000,)
Random Forest - accuracy: 0.8920
```

Exactitud del modelo: Se obtuvo un 89.2% de precisión con Random Forest. Esto significa que el modelo clasifica correctamente casi 9 de cada 10 dígitos.

Interpretación: Es un buen resultado para un modelo tradicional, considerando que PolyMNIST añade ruido visual. Aun así, modelos basados en CNN suelen superar el 95% al aprovechar la estructura espacial de las imágenes.

Por lo tanto, Random Forest es una opción válida y eficiente si no se puede usar redes neuronales, aunque tiene un límite frente a modelos más complejos.