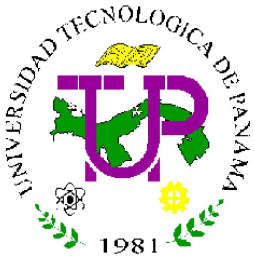
Universidad Tecnológica De Panamá
Facultad De Ingeniería De Sistemas Computacionales
Lic. en Ingeniería de Sistemas y Computación



Memoria de Trabajo para la Asignatura:
Tópicos I – Visión Artificial

Laboratorio 7
Aprendizaje Automático en Visión Artificial

Estudiante:
Charles Chuez

Grupo:
1IL141

Profesor:
José Carlos Rangel Ortiz

II Semestre, 2024

Introducción

Siguiendo con el módulo de Machine Learning, dentro de este podemos encontrar el Deep Learning, el cual se fundamenta en la utilización de Redes Neuronales Artificiales con una gran cantidad de capas para el procesamiento y aprendizaje de la información que se suministre.

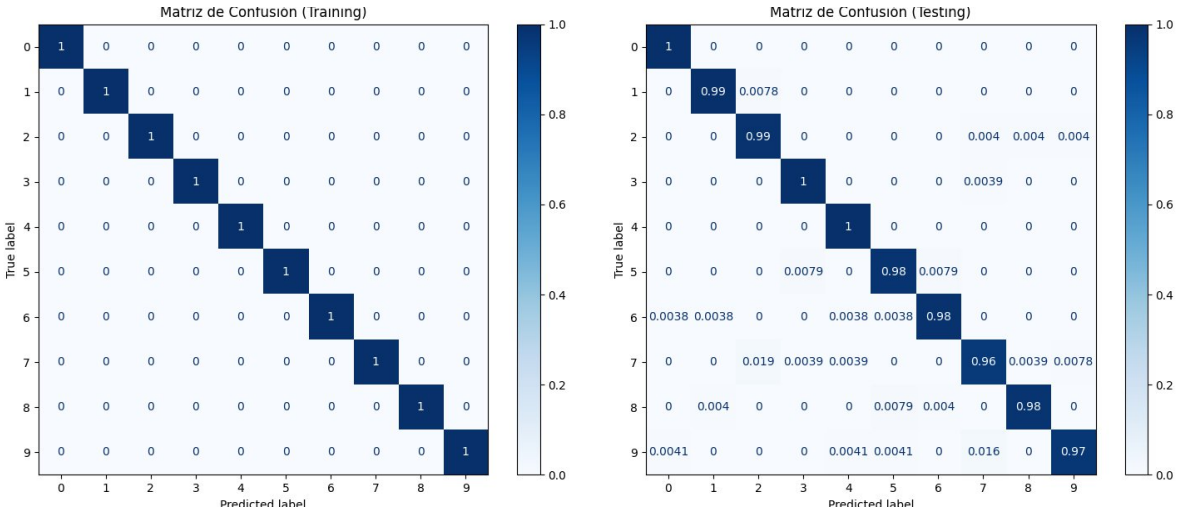
En este laboratorio se presentarán formas de construir modelos de clasificación utilizando una varios algoritmos de ML. Durante esta parte del laboratorio el modelo será entrenado utilizando datos (imágenes) que se encuentran localmente en nuestra computadora, o dicho de otra manera, un dataset local. El laboratorio muestra todo el procedimiento y permite también evaluar el modelo generado utilizando una imagen externa a las disponibles en el dataset.

El laboratorio puede ser desarrollado construyendo el script completamente y ejecutando desde consola o mediante jupyter lab realizando una ejecución por celdas.

Resultados

Parte 1

1. Genere una matriz de confusión para la secuencia de train y de test creada en el código, utilizando la función ConfusionMatrixDisplay.from predictions() de SKLearn.

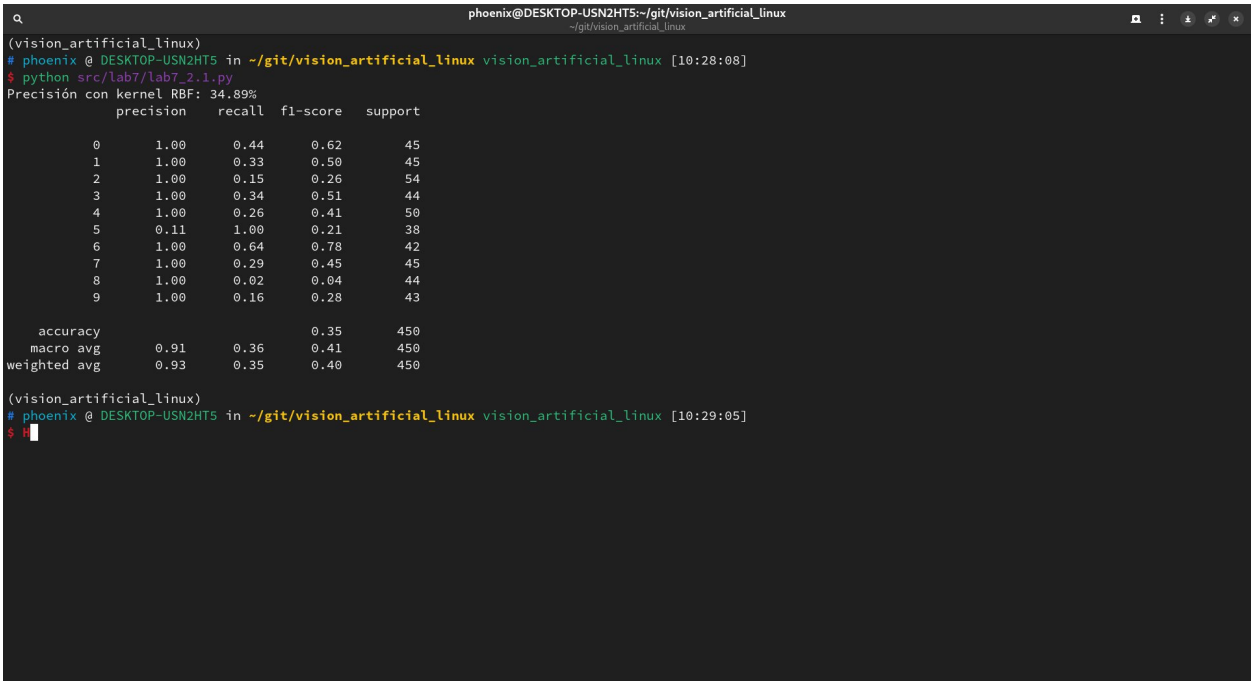


2. Coloque adecuadamente el titulo a cada matriz y presente la matriz y una captura del código utilizado para generar la matriz.

```python
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Get predictions
y_train_pred = model.predict(hog_descriptors_train)[1].ravel()
y_test_pred = model.predict(hog_descriptors_test)[1].ravel()

# Create figure with two subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

# Plot training confusion matrix
ConfusionMatrixDisplay.from_predictions(
    labels_train,
    y_train_pred,
    ax=ax1,
    cmap='Blues',
    normalize='true',
    display_labels=range(10)
)
ax1.set_title('Matriz de Confusión (Training)')

# Plot testing confusion matrix
ConfusionMatrixDisplay.from_predictions(
    labels_test,
    y_test_pred,
    ax=ax2,
    cmap='Blues',
    normalize='true',
    display_labels=range(10)
)
ax2.set_title('Matriz de Confusión (Testing)')

plt.tight_layout()
plt.show()
```

Parte 2

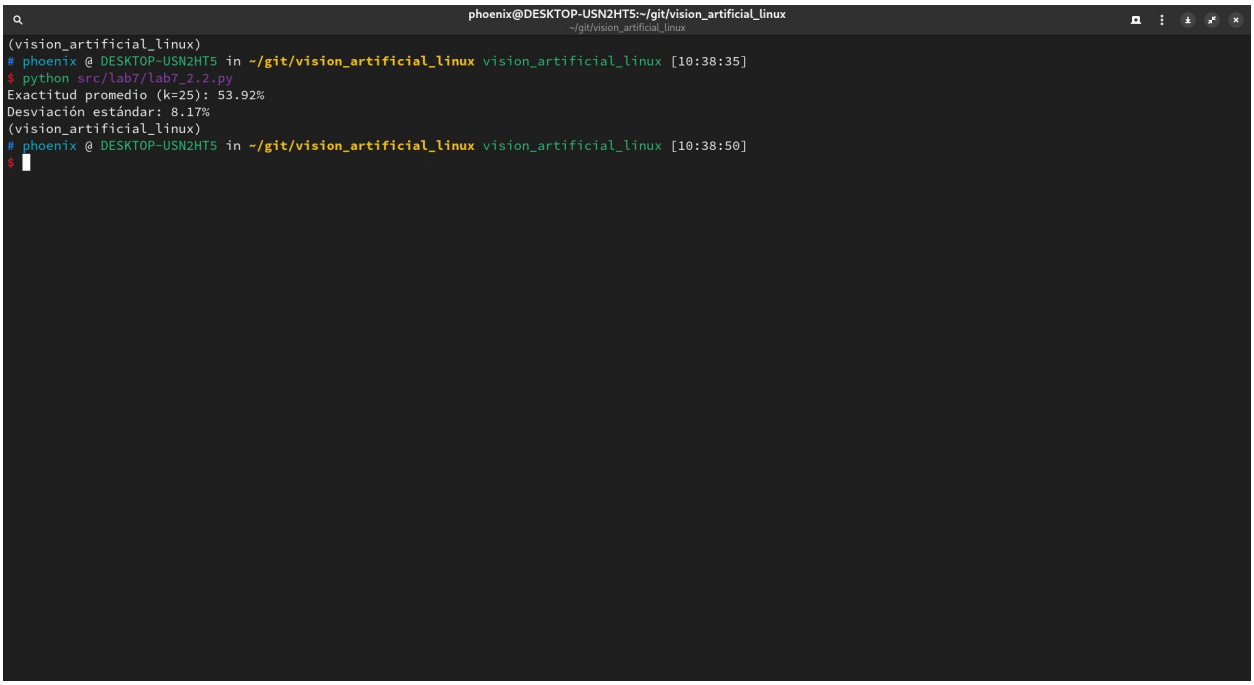
1. Defina el kernel del modelo a un tipo rbf, ejecute el sistema e indique los resultados.



2. Teniendo en cuenta la matriz de confusión generada con el programa original, indique que podemos deducir de esta, si nos enfocamos en el número 1.

    - Algunos 1's están siendo confundidos con otros números
    - Hay una tendencia a confundir los 1's con otros dígitos cercanos visualmente (como 7 o 4)
    - La matriz de confusión nos permite ver la precisión específica para el número 1 y sus "falsos positivos"

3. Aplique un k-fold con k = 25 y reporte cual es la exactitud promedio obtenida por el modelo con RBF.

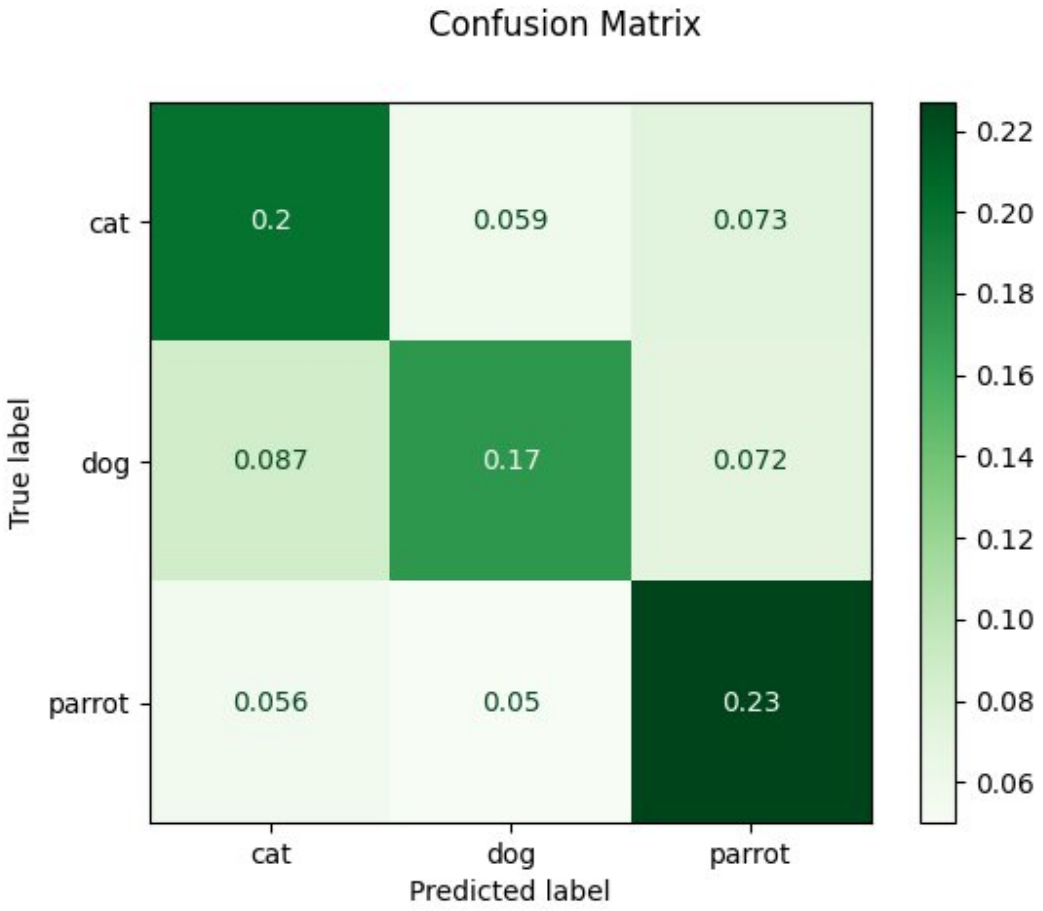


Exactitud promedio con 25-fold cross-validation.

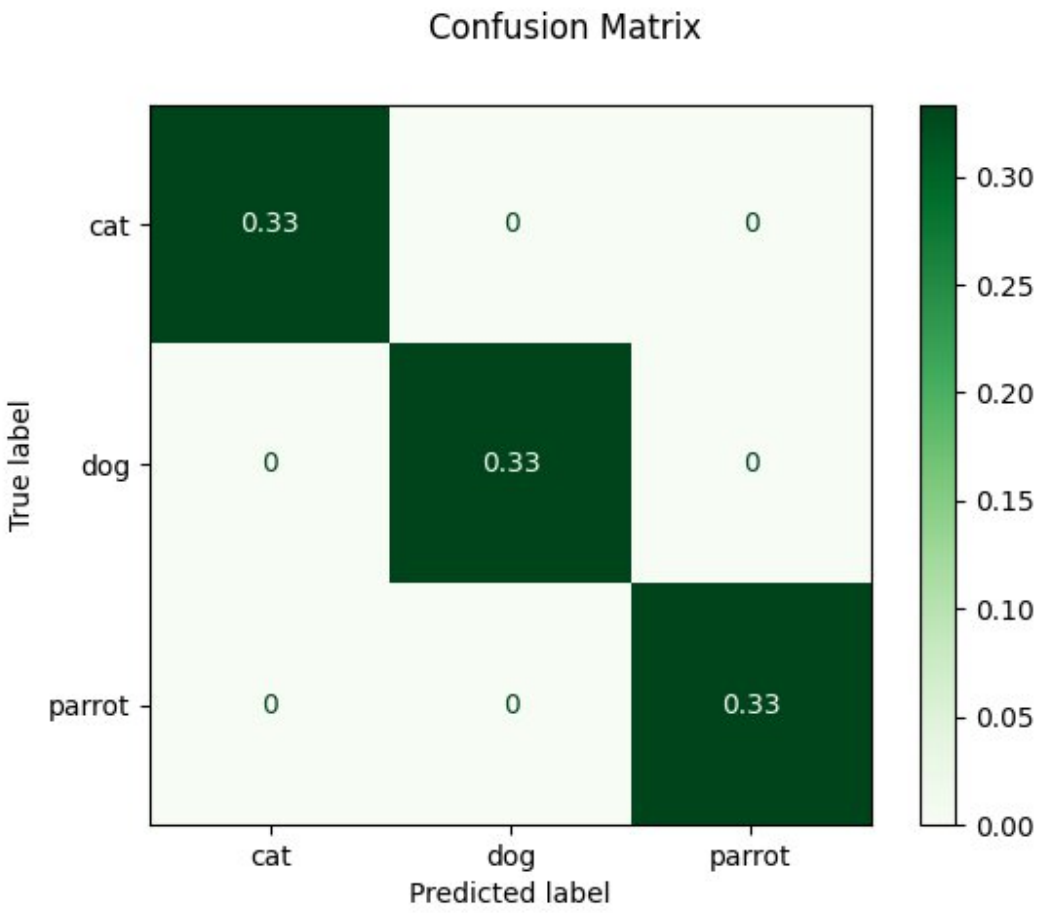
Parte 3

1. Prepare los datos de la secuencia de validación disponibles en el dataset.

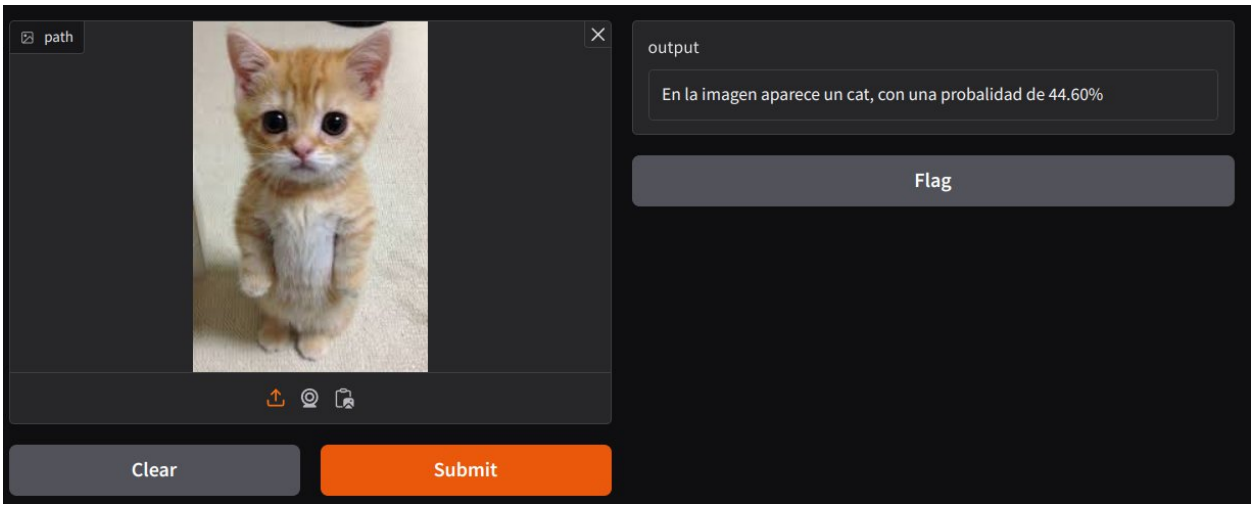
2. Pruebe su modelo (Calcular Score) de Random Forest utilizando la secuencia de Validación y presente la Matriz de Confusión Normalizada obtenida para esta secuencia.



3. Pruebe su modelo (Calcular Score) de Random Forest utilizando la secuencia de Train y presente la Matriz de Confusión Normalizada obtenida para esta secuencia.



4. Presente una captura de pantalla de una prueba de este clasificador utilizando Gradio.



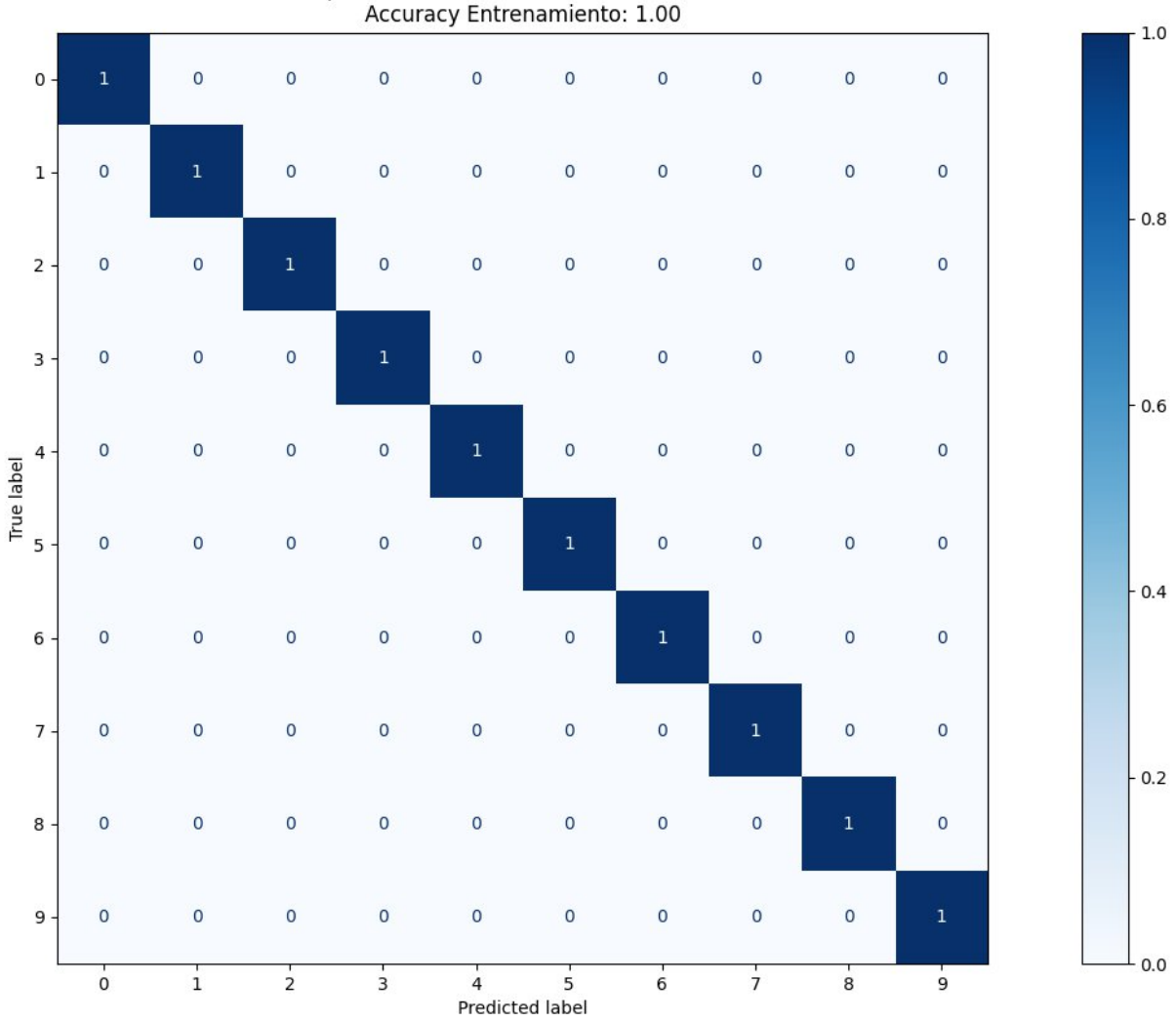
Parte 4

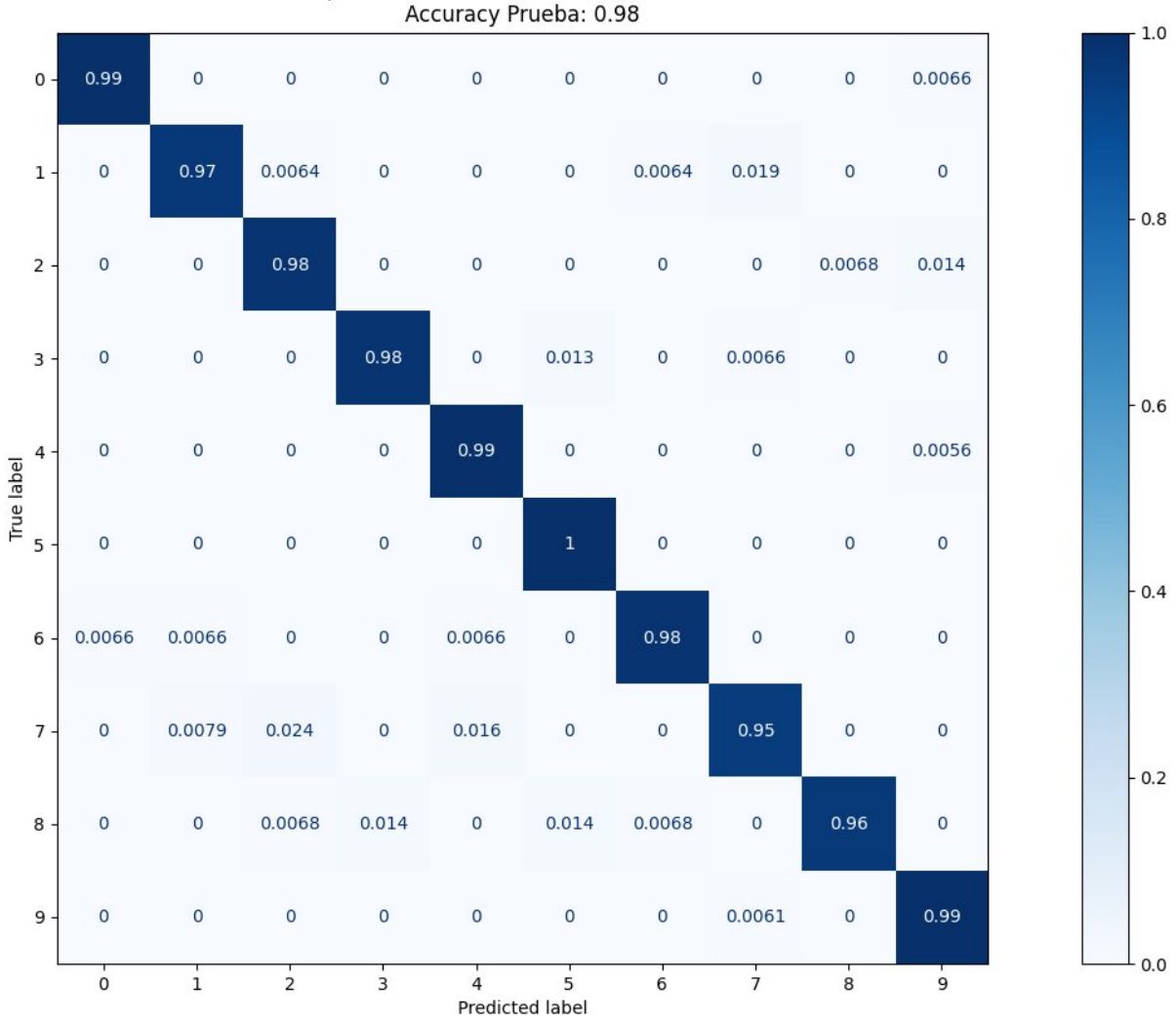
1. Realice 2 modificaciones las capas definidas del MLP y genere para cada modificación la matriz de confusión de Train y Test. Puede modificar tamaños de capa y cantidad de capas o ambas. Puede indagar otros parámetros de esta función y mortificarlos para evaluar su efecto en el resultado final.

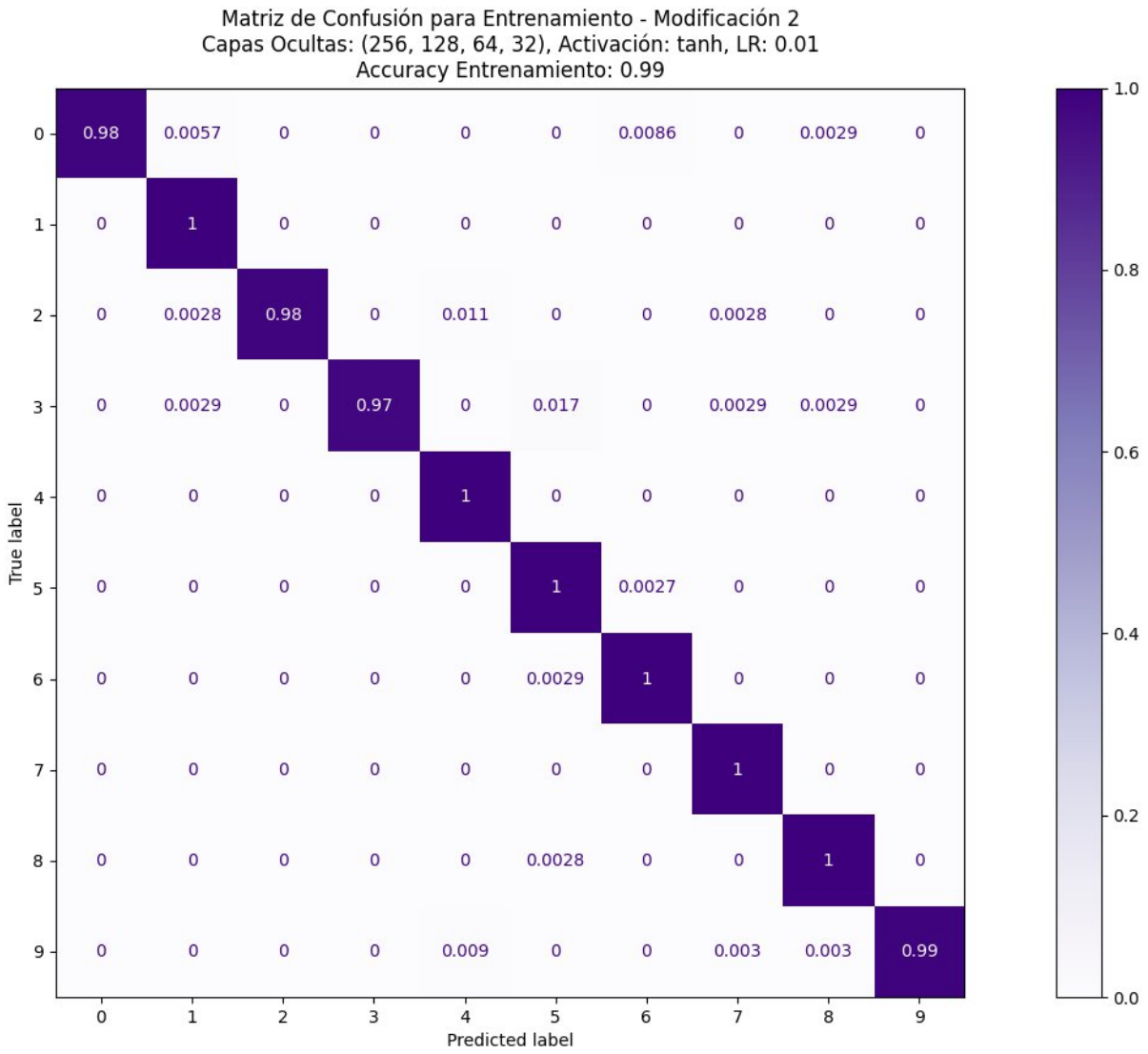
```
# Modificación 1: Cambiar capas ocultas
clf_mod1 = MLPClassifier(hidden_layer_sizes=(512, 256), activation="relu", random_state=25)
clf_mod1.fit(hog_descriptors_train, labels_train)

# Accuracy de entrenamiento y prueba
accuracy_train_mod1 = clf_mod1.score(hog_descriptors_train, labels_train)
accuracy_test_mod1 = clf_mod1.score(hog_descriptors_test, labels_test)

# Matriz de confusión para el conjunto de entrenamiento
ConfusionMatrixDisplay.from_estimator(clf_mod1, hog_descriptors_train, labels_train, normalize='true', cmap='Blues')
plt.title("Matriz de Confusión para Entrenamiento - Modificación 1\nCapas Ocultas: (512, 128), Activación: relu\nAccuracy Entrenamiento: {:.2f}".format(accuracy_train_mod1))
plt.savefig("src/lab7/out/lab7_4.1.png")
plt.show()

# Matriz de confusión para el conjunto de prueba
ConfusionMatrixDisplay.from_estimator(clf_mod1, hog_descriptors_test, labels_test, normalize='true', cmap='Blues')
plt.title("Matriz de Confusión para Prueba - Modificación 1\nCapas Ocultas: (512, 128), Activación: relu\nAccuracy Prueba: {:.2f}".format(accuracy_test_mod1))
plt.savefig("src/lab7/out/lab7_4.2.png")
plt.show()
```

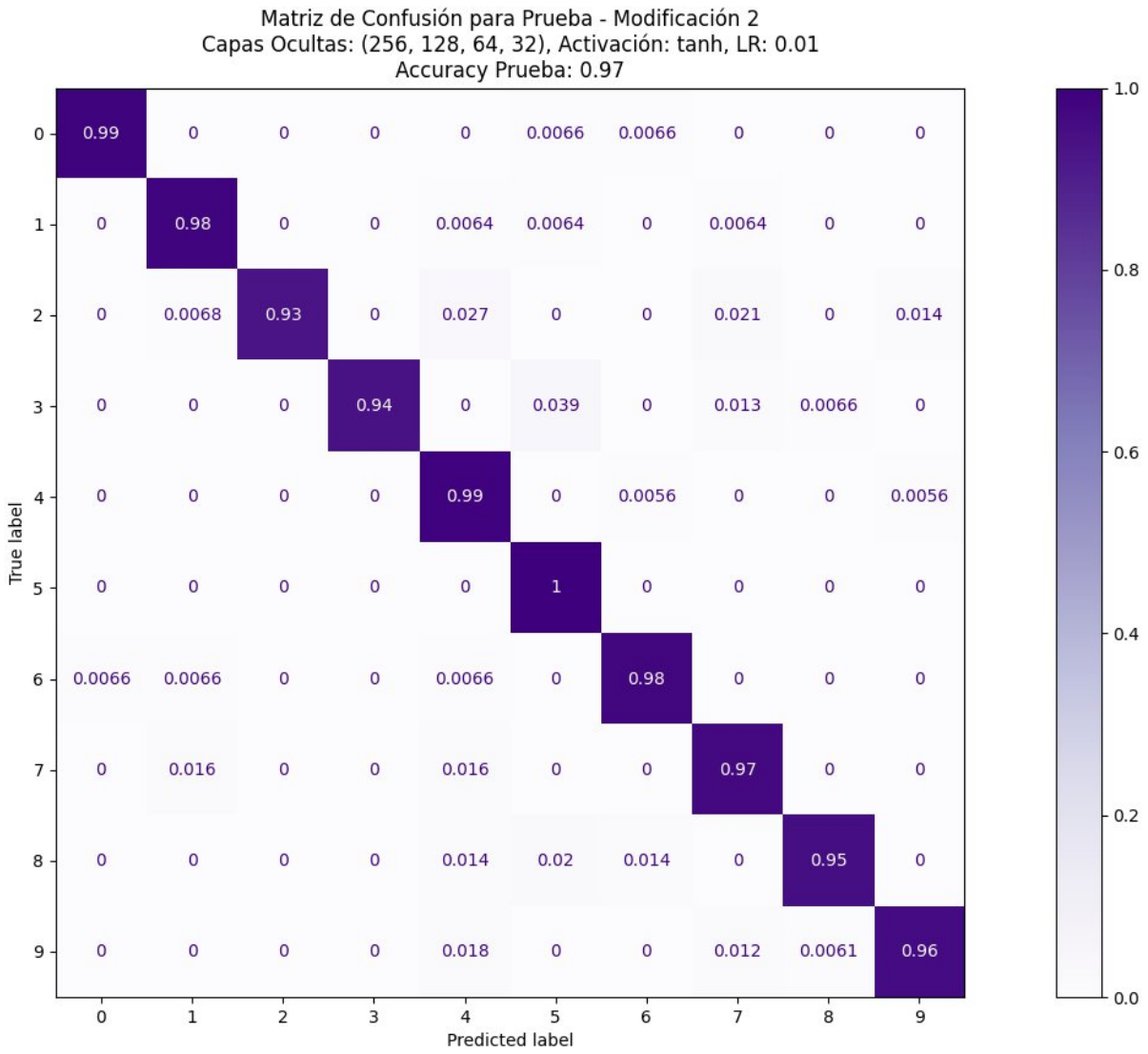
```
# Modificación 2: Cambiar activación y tasa de aprendizaje
clf_mod2 = MLPClassifier(hidden_layer_sizes=(256, 128, 64, 32), activation="tanh", learning_rate_init=0.01, random_state=25)
clf_mod2.fit(hog_descriptors_train, labels_train)

# Accuracy de entrenamiento y prueba
accuracy_train_mod2 = clf_mod2.score(hog_descriptors_train, labels_train)
accuracy_test_mod2 = clf_mod2.score(hog_descriptors_test, labels_test)

# Matriz de confusión para el conjunto de entrenamiento
ConfusionMatrixDisplay.from_estimator(clf_mod2, hog_descriptors_train, labels_train, normalize='true', cmap='Purples')
plt.title("Matriz de Confusión para Entrenamiento - Modificación 2\nCapas Ocultas: (256, 128, 64, 32), Activación: tanh, LR: 0.01\nAccuracy Entrenamiento: {:.
2f}".format(accuracy_train_mod2))
plt.savefig("src/lab7/out/lab7_4.3.png")
plt.show()

# Matriz de confusión para el conjunto de prueba
ConfusionMatrixDisplay.from_estimator(clf_mod2, hog_descriptors_test, labels_test, normalize='true', cmap='Purples')
plt.title("Matriz de Confusión para Prueba - Modificación 2\nCapas Ocultas: (256, 128, 64, 32), Activación: tanh, LR: 0.01\nAccuracy Prueba: {:.2f}".format(accuracy_test_mod2))
plt.savefig("src/lab7/out/lab7_4.4.png")
plt.show()
```

2. Coloque adecuadamente el titulo a cada matriz y presente la matriz en su informe de laboratorio, indique para cada una la cantidad de capas ocultas y el tamaño de cada una, así como también el accuracy obtenido.


Matriz de Confusión para Entrenamiento - Modificación 1
Capas Ocultas: (512, 128), Activación: relu
Accuracy Entrenamiento: 1.00
True label
Predicted label
0 1 2 3 4 5 6 7 8 9
1.0 0.8 0.6 0.4 0.2 0.0

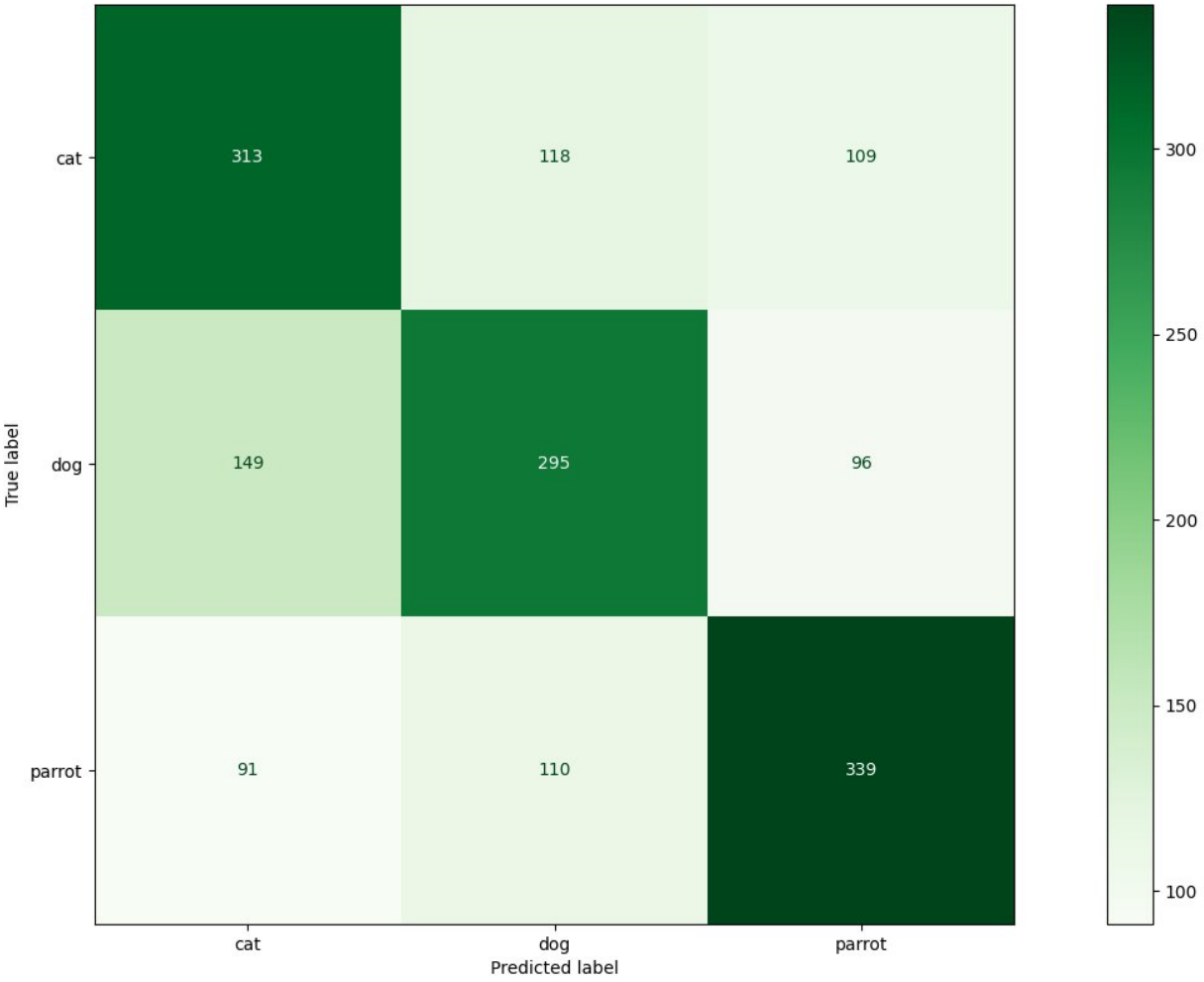

Matriz de Confusión para Prueba - Modificación 1
Capas Ocultas: (512, 128), Activación: relu
Accuracy Prueba: 0.98
True label
Predicted label
0 0.99 0 0 0 0 0 0 0 0 0.0066
1 0 0.97 0.0064 0 0 0 0.0064 0.019 0 0
2 0 0 0.98 0 0 0 0 0 0.0068 0.014
3 0 0 0 0.98 0 0.013 0 0.0066 0 0
4 0 0 0 0 0.99 0 0 0 0 0.0056
5 0 0 0 0 0 1 0 0 0 0
6 0.0066 0.0066 0 0 0.0066 0 0.98 0 0 0
7 0 0.0079 0.024 0 0.016 0 0 0.95 0 0
8 0 0 0.0068 0.014 0 0.014 0.0068 0 0.96 0
9 0 0 0 0 0 0 0 0.0061 0 0.99
0 1 2 3 4 5 6 7 8 9
1.0 0.8 0.6 0.4 0.2 0.0

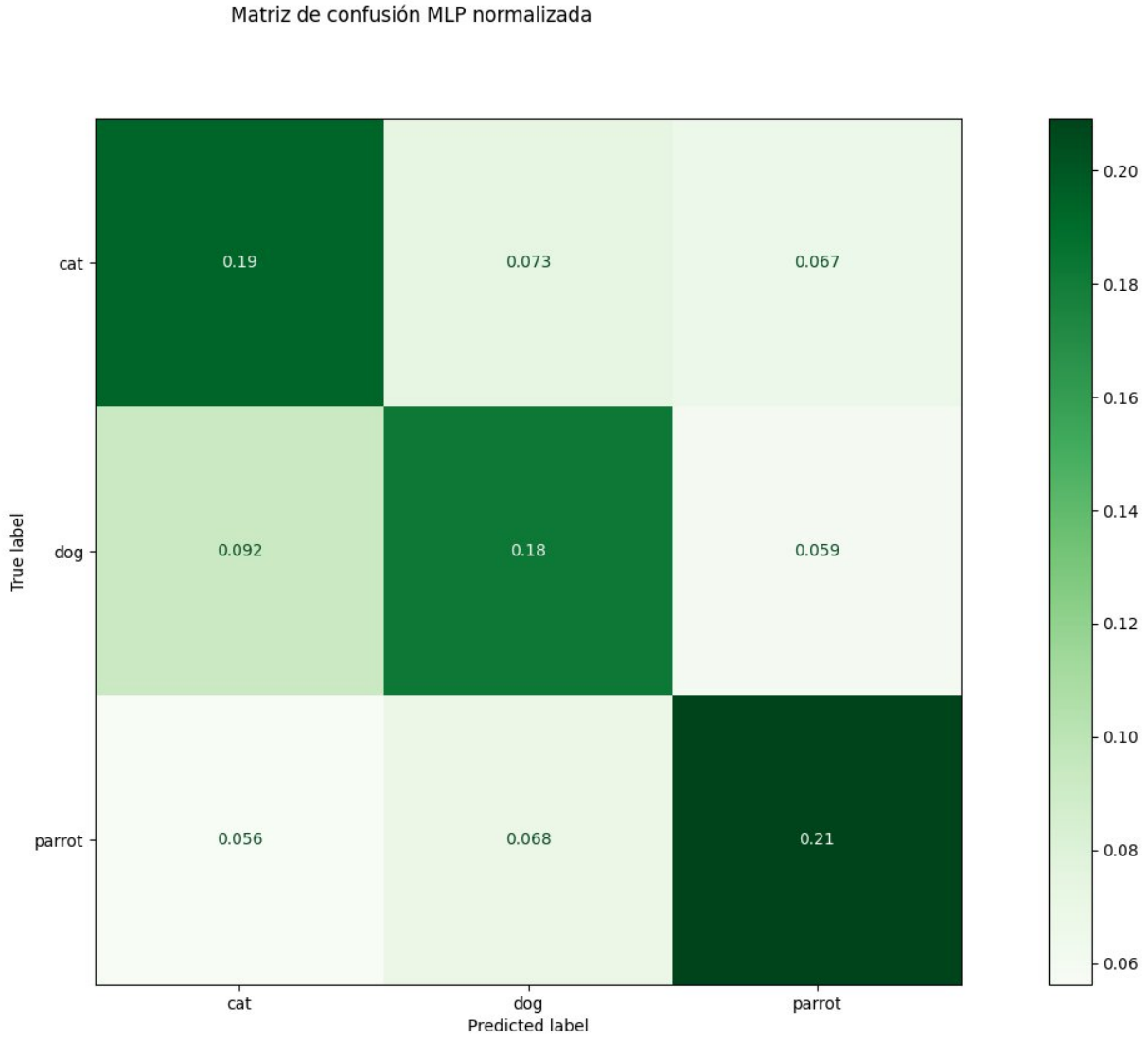

Matriz de Confusión para Entrenamiento - Modificación 2
Capas Ocultas: (256, 128, 64, 32), Activación: tanh, LR: 0.01
Accuracy Entrenamiento: 0.99
True label
Predicted label
0 0.98 0.0057 0 0 0 0 0.0086 0 0.0029 0
1 0 1 0 0 0 0 0 0 0 0
2 0 0.0028 0.98 0 0.011 0 0 0.0028 0 0
3 0 0.0029 0 0.97 0 0.017 0 0.0029 0.0029 0
4 0 0 0 0 1 0 0 0 0 0
5 0 0 0 0 0 1 0.0027 0 0 0
6 0 0 0 0 0 0.0029 1 0 0 0
7 0 0 0 0 0 0 0 1 0 0
8 0 0 0 0 0 0.0028 0 0 1 0
9 0 0 0 0 0.009 0 0 0.003 0.003 0.99
0 1 2 3 4 5 6 7 8 9
1.0 0.8 0.6 0.4 0.2 0.0

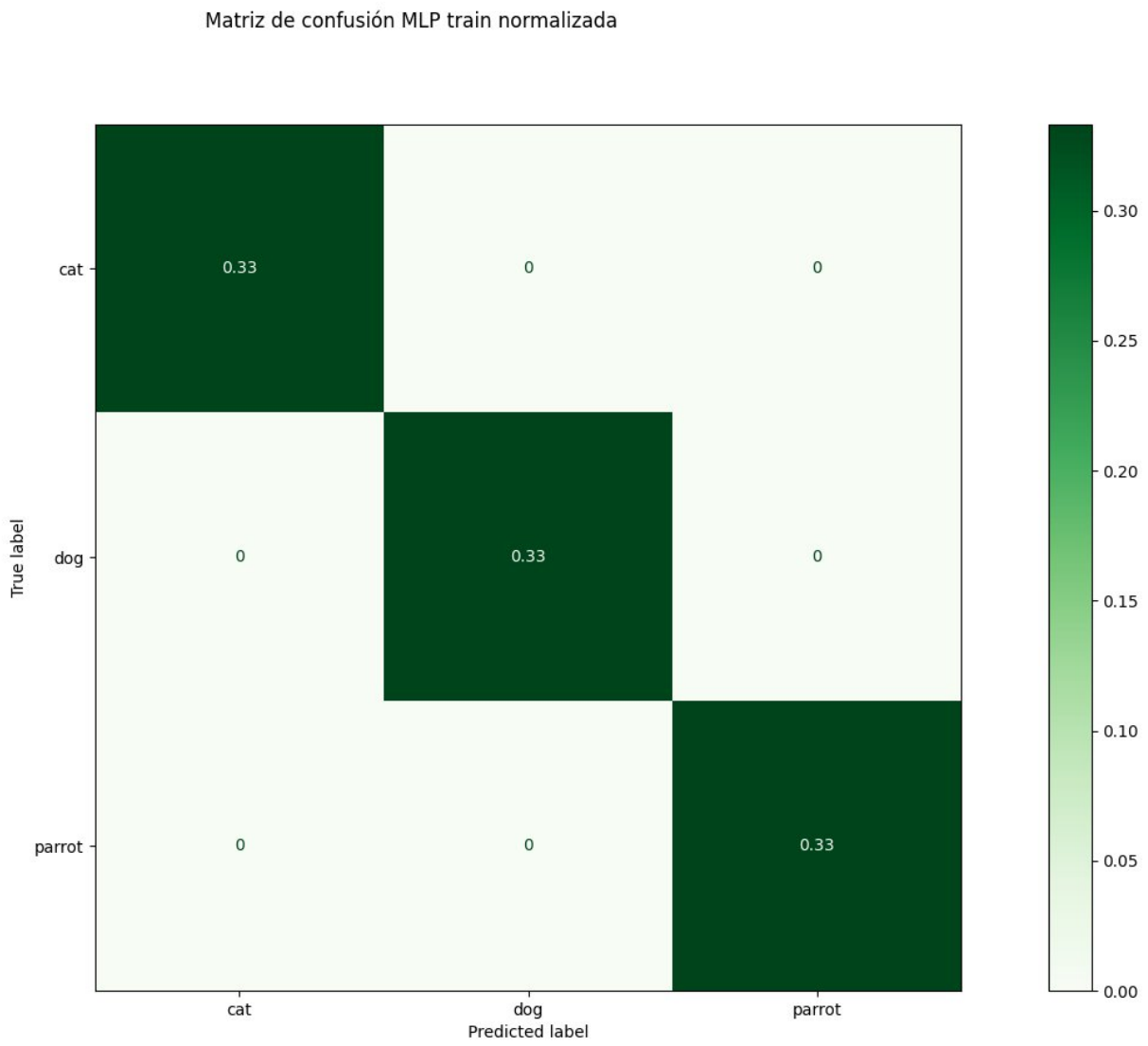


3. Utilizando el código y secuencias de train y test de la Sección 3 reemplace el clasificador de Random Forest por un MLP y presente la matriz de confusión para este nuevo modelo entrenado.

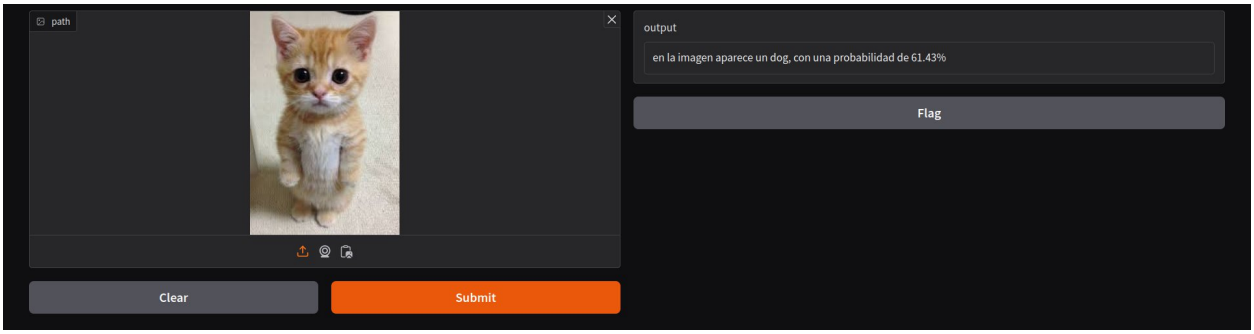
Matriz de confusión MLP no normalizada




Matriz de confusión MLP normalizada
cat
dog
parrot
True label
0.19
0.073
0.067
0.092
0.18
0.059
0.056
0.068
0.21
cat
dog
parrot
Predicted label
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06



4. Presente una captura de pantalla de una prueba de este clasificador utilizando Gradio.



Conclusión y Comentarios de los Estudiantes

Muy bonito el laboratorio pero tengo sueño 😴. Aunque sí está bueno lo de neuronas artificiales junto con la visión. Es emocionante ver el futuro.

Bibliografía y Referencias

[1] A. Fernández Villán, Mastering OpenCV 4 with Python: a practical guide covering topics from image processing, augmented reality to deep learning with OpenCV 4 and Python 3.7. Mastering Open Source Computer Vision four with Python. Birmingham: Packt Publishing, 2019. [Online]. Available: https://cds.cern.ch/record/2674578

[2] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), vol. 1, 2005, pp. 886–893 vol. 1.

[3] P. Deitel and H. Deitel, Python for Programmers. Pearson Education, 2019. [Online]. Available: https://books.google.com.pa/books?id=LauMDwAAQBAJ