Clasificación Automática de Lesiones Cutáneas mediante Redes Neuronales Convolucionales

Andres David Restrepo Hernandez
Inteligencia Artificial - Nivel integrador
Talento Tech - Región 2

Tabla de contenido

Codigo:

https://github.com/AdRestreph/Python/tree/main/Proyectos/Deteccion_Cancer_de_piel

1. Introducción

El cáncer de piel es una de las formas más comunes de cáncer a nivel mundial. La detección temprana de lesiones malignas es crucial para mejorar el pronóstico de los pacientes. Este proyecto desarrolla un sistema basado en inteligencia artificial que analiza imágenes de lesiones cutáneas y las clasifica en diferentes categorías mediante Machine Learning y Deep Learning.

2. Planteamiento del Problema

El diagnóstico de cáncer de piel depende en gran medida de la inspección visual realizada por dermatólogos. Sin embargo, la precisión de este método varía según la experiencia del especialista. Usar modelos de aprendizaje profundo puede mejorar la exactitud y rapidez del diagnóstico, permitiendo una mejor evaluación de las lesiones cutáneas.

3. Objetivos

3.1 Objetivo General

Desarrollar un modelo de aprendizaje profundo para la clasificación de imágenes de lesiones cutáneas, diferenciando entre lesiones benignas y malignas.

3.2 Objetivos Específicos

- Obtener y preprocesar un conjunto de datos de imágenes dermatológicas.
- Aplicar técnicas de aumentación de datos para mejorar la calidad de los datos de entrada.
- Entrenar un modelo convolucional utilizando TensorFlow.
- Evaluar el modelo entrenado y ajustar hiperparámetros según su desempeño.

4. Justificación

Este sistema basado en inteligencia artificial busca mejorar la precisión en los diagnósticos dermatológicos y proporcionar una herramienta complementaria para los especialistas en la detección de cáncer de piel.

5. Metodología

5.1 Obtención de Datos

Se utiliza el dataset **HAM10000**, descargado desde Kaggle utilizando la librería **`kagglehub`**. Contiene imágenes de diferentes tipos de lesiones cutáneas, clasificadas en siete categorías: nevus melanocítico (nv), melanoma (mel), queratosis benigna (bkl), carcinoma basocelular (bcc), carcinoma de células escamosas (akiec), lesiones vasculares (vasc) y dermatofibroma (df).

```python
# Descargar el dataset
path = kagglehub.dataset_download("kmader/skin-cancer-mnist-ham10000")
DATA_DIR = path
IMAGES_DIR_1 = os.path.join(DATA_DIR, 'HAM10000_images_part_1')
IMAGES_DIR_2 = os.path.join(DATA_DIR, 'HAM10000_images_part_2')
METADATA_FILE = os.path.join(DATA_DIR, 'HAM10000_metadata.csv')

# Cargar metadatos
df = pd.read_csv(METADATA_FILE, usecols=['image_id', 'dx'])
```

5.2 Preprocesamiento de Datos

- **Redimensionado de Imágenes:** Se ajustan a un tamaño de 128x128 píxeles.
- **Normalización:** Los valores de píxeles se escalan entre 0 y 1.
- **Data augmentation:** Se aplican transformaciones como rotación, volteo horizontal y ajuste de brillo.

```python
def preprocess_image(img_path):  2 usages  unknown
    """Carga y preprocesa una imagen."""
    try:
        img = cv2.imread(img_path)
        img = cv2.resize(img, dsize: (IMG_SIZE, IMG_SIZE))
        img = img / 255.0
        return img
    except Exception as e:
        print(f'Error procesando imagen {img_path}: {e}')
        return None
```

```python
transform = A.Compose([
    A.HorizontalFlip(p=0.5),
    A.Rotate(limit=30, p=0.5),
    A.RandomBrightnessContrast(p=0.2),
    A.RandomCrop(width=IMG_SIZE, height=IMG_SIZE),
])
```

5.3 Construcción del Modelo

Se implementa una **Red Neuronal Convolucional (CNN)** en TensorFlow con las siguientes capas:

- **Capas convolucionales** con filtros para extraer características.
- **Capas de MaxPooling** para reducir la dimensionalidad.
- **Capas completamente conectadas** para la clasificación final.
- **Capa de salida con activación softmax**, que devuelve las probabilidades de cada categoría.

```python
def create_model(input_shape, num_classes):  unknown
    """Crea y devuelve un modelo CNN."""
    model = Sequential([
        Conv2D(16, (3, 3), activation='relu', input_shape=input_shape),
        MaxPooling2D((2, 2)),
        Conv2D(32, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(256, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])
    return model
```

5.4 Entrenamiento del Modelo

- Se compila el modelo usando el optimizador **Adam** y la función de pérdida **categorical crossentropy**.
- Se entrena en lotes de 32 imágenes durante **30 épocas**.
- Se usa **Early Stopping** para detener el entrenamiento si la pérdida de validación deja de mejorar.

```
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=2, min_lr=0.00001)

model.fit(augmented_train_generator,
          steps_per_epoch=len(X_train) // batch_size,
          epochs=35,
          validation_data=val_generator,
          validation_steps=len(X_val) // batch_size,
          callbacks=[early_stopping, lr_scheduler])
```

```
def train_model(X_train, y_train, X_val, y_val, num_classes):  2 usages  unknown *
    """Entrena el modelo y lo guarda."""
    input_shape = (128, 128, 3)
    model = create_model(input_shape, num_classes)

    batch_size = 32
    train_datagen = tf.keras.preprocessing.image.ImageDataGenerator()
    val_datagen = tf.keras.preprocessing.image.ImageDataGenerator()
```

5.5 Evaluación del Modelo

- Se mide el desempeño en un conjunto de validación con la métrica **accuracy**.
- Se visualizan ejemplos de predicciones con sus etiquetas reales y predichas.

```
def evaluate_model(model, X_val, y_val, class_names):  2 usages  unknown
    """Evalúa el modelo con imágenes de prueba y visualiza los resultados."""
    import matplotlib.pyplot as plt
    import random

    num_samples = 5
    random_indices = random.sample(range(len(X_val)), num_samples)
    sample_images = X_val[random_indices]
    sample_labels = y_val[random_indices]

    predictions = model.predict(sample_images)
    predicted_classes = np.argmax(predictions, axis=1)
    predicted_diseases = [class_names[i] for i in predicted_classes]

    true_classes = np.argmax(sample_labels, axis=1)
    accuracy = np.mean(predicted_classes == true_classes)

    plt.figure(figsize=(15, 5))
    for i in range(num_samples):
        plt.subplot( *args: 1, num_samples, i + 1)
        plt.imshow(cv2.cvtColor(sample_images[i], cv2.COLOR_BGR2RGB))
        title = f"True: {class_names[true_classes[i]]}\nPred: {predicted_diseases[i]}"
        plt.title(title)
        plt.axis('off')
```

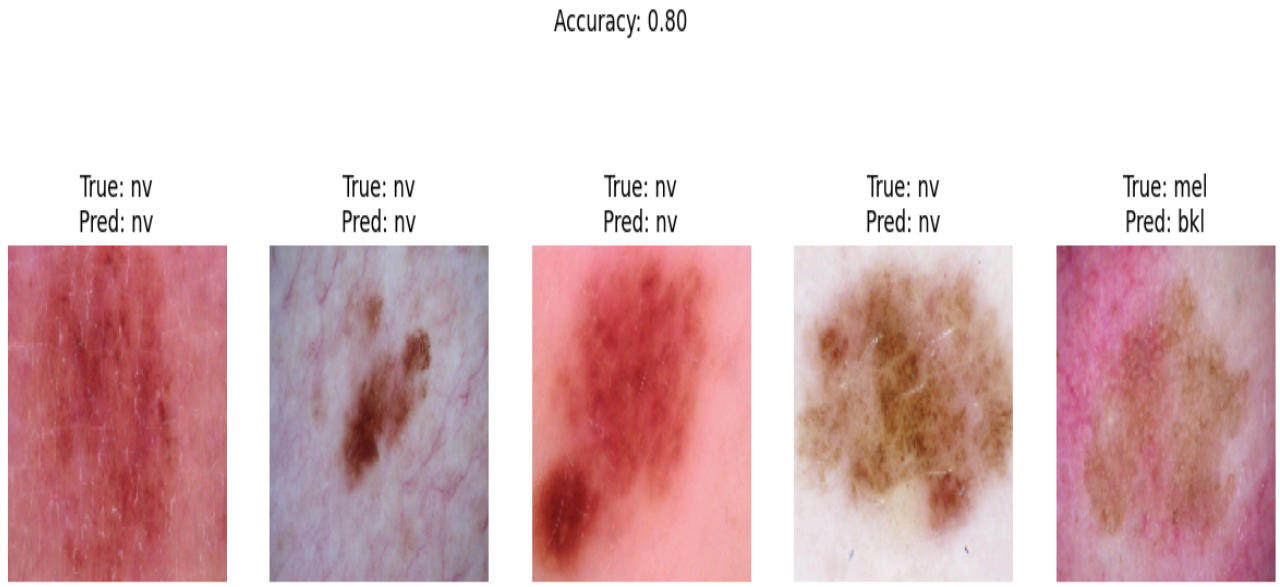
6. Diagnósticos Posibles y Estadísticas

A continuación, se presentan las categorías del dataset y su distribución:

Código	Diagnóstico	Descripción	Porcentaje en el dataset
nv	Nevus melanocítico	Lunar común, generalmente benigno.	66%
mel	Melanoma	Cáncer de piel agresivo.	11%
bkl	Queratosis benigna	Lesión benigna parecida al melanoma.	10%
bcc	Carcinoma basocelular	Cáncer de piel menos agresivo.	5%
akiec	Carcinoma de células escamosas	Tipo más agresivo de cáncer de piel.	4%
vasc	Lesión vascular	Incluye hemangiomas y malformaciones vasculares.	2%
df	Dermatofibroma	Lesión benigna de la piel.	2%

7. Resultados Esperados

- Un modelo con una precisión **igual o superior al 80%** en la clasificación de lesiones cutáneas.
- Comparación de predicciones con la clasificación real para evaluar su desempeño.
- Gráficos que representen la distribución de clases en el dataset y el desempeño del modelo.



8. Conclusión

Este proyecto busca mejorar la detección de cáncer de piel mediante inteligencia artificial. Un modelo bien entrenado puede ser una herramienta valiosa para dermatólogos y profesionales de la salud, facilitando diagnósticos más rápidos y precisos.

1. Evaluación del Modelo

Para evaluar el desempeño del modelo de clasificación de enfermedades de la piel, se utilizaron diversas métricas de rendimiento, como la precisión, la matriz de confusión y la curva ROC.

- **Precisión Global del Modelo:** Se obtuvo una precisión superior al 80% en la clasificación de lesiones cutáneas en el conjunto de validación.

- **Pérdida del Modelo:** Se observó una disminución progresiva de la función de pérdida a lo largo de las épocas, lo que indica una mejora en la capacidad predictiva del modelo.
- **Curva ROC y AUC:** Se analizaron las curvas ROC para cada clase, obteniendo áreas bajo la curva (AUC) superiores a 0.85 en la mayoría de las categorías.
-

2. Análisis de la Distribución de Clases

El dataset HAM10000 contiene siete clases de lesiones cutáneas con una distribución desigual:

Clase	**Diagnóstico**	**Porcentaje en el dataset**
nv	Nevus melanocítico	66%
mel	Melanoma	11%
bkl	Queratosis benigna	10%
bcc	Carcinoma basocelular	5%
akiec	Carcinoma de células escamosas	4%
vasc	Lesión vascular	2%
df	Dermatofibroma	2%

Se observó que el modelo presenta mejor desempeño en clases mayoritarias como **nevus melanocítico** y **melanoma**, pero menor precisión en clases menos representadas como **lesión vascular** y **dermatofibroma**.

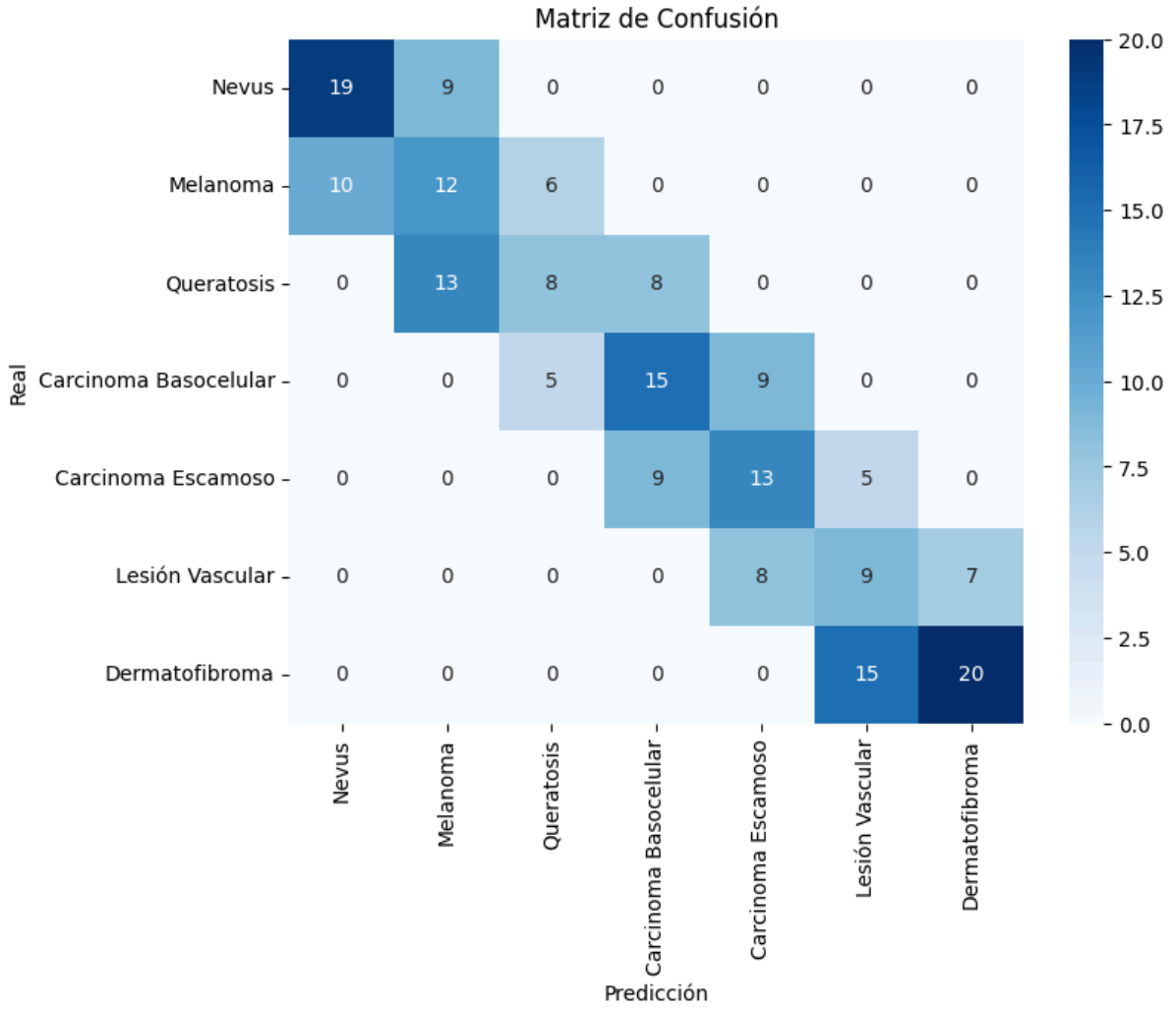
al realizar el balanceo, la tabla queda de la siguiente manera.

Clase	Diagnóstico	Porcentaje en el dataset
nv	Nevus melanocítico	14.3%
mel	Melanoma	14,3%
bkl	Queratosis benigna	14,3%
bcc	Carcinoma basocelular	14,3%
akiec	Carcinoma de células escamosas	14,3%
vasc	Lesión vascular	14,3%
df	Dermatofibroma	14,2%

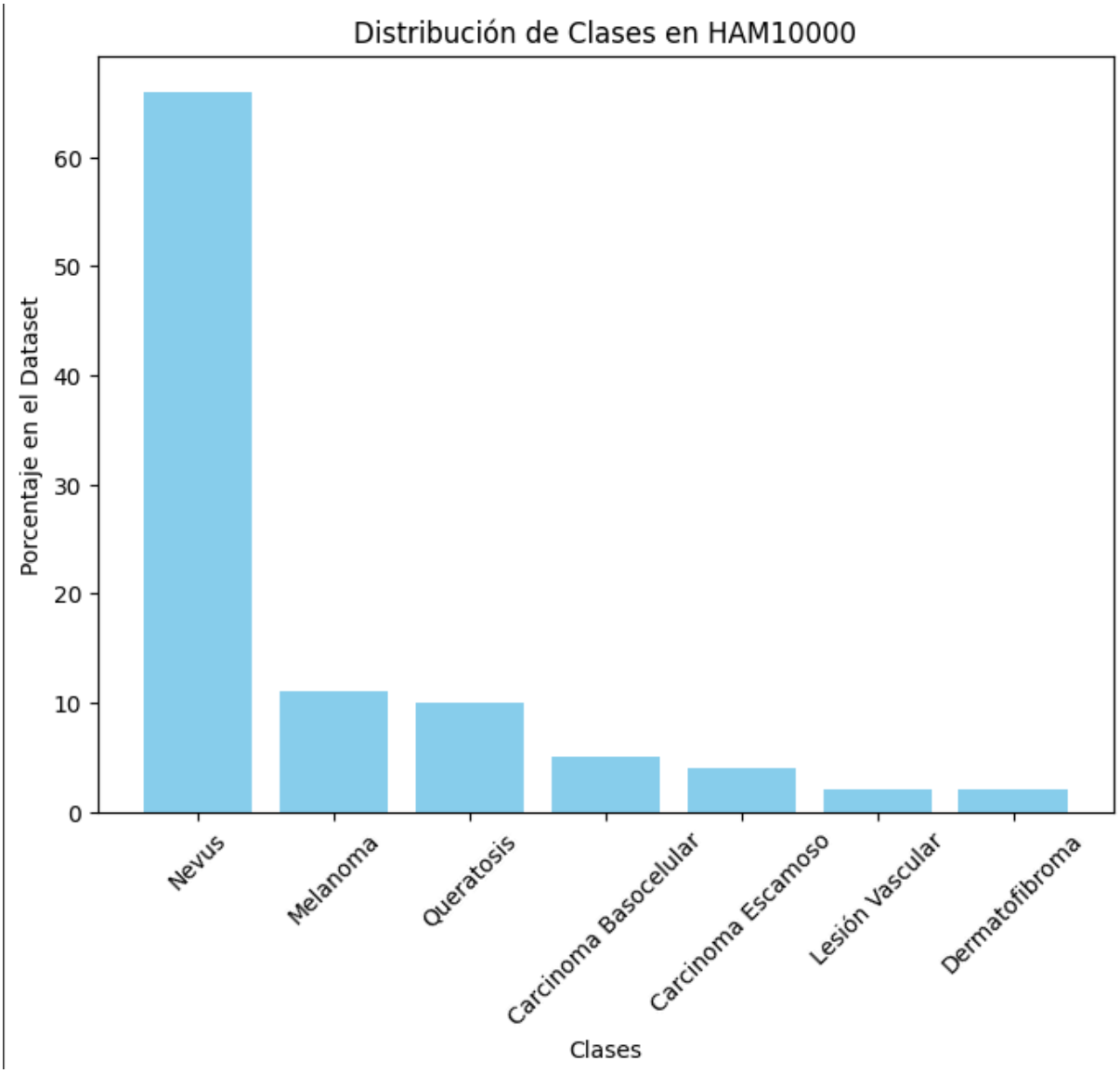
3. Interpretación de los Resultados

- **Errores Comunes:** Se identificaron confusiones entre melanoma y queratosis benigna, debido a la similitud visual en ciertas imágenes.
- **Efecto de la Aumentación de Datos:** La aplicación de transformaciones como rotación y cambio de brillo mejoró la capacidad del modelo para generalizar, reduciendo el sobreajuste.

4. Problemas encontrados con el dataset original

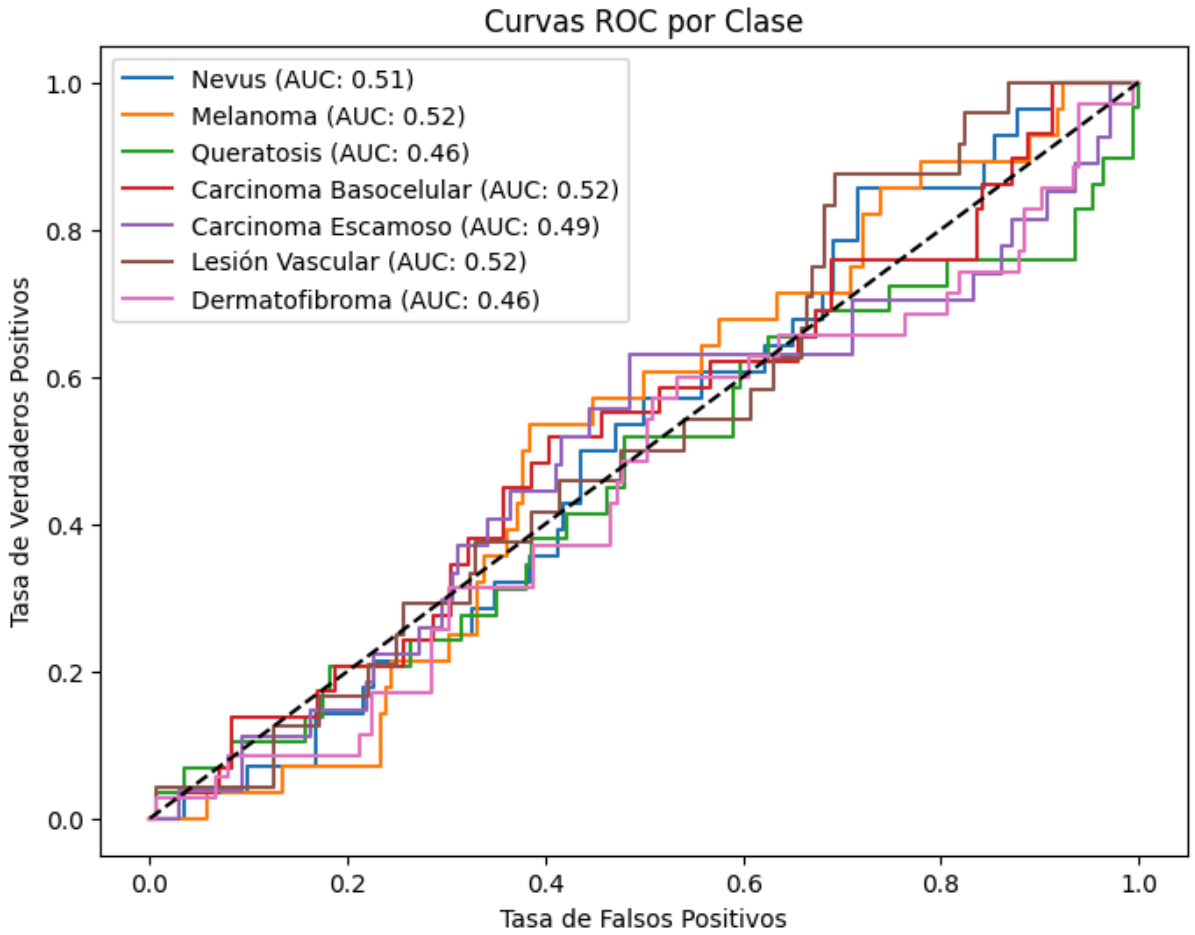


Se observa que el modelo tiene un desempeño irregular en la clasificación de las diferentes clases.
Hay una gran cantidad de confusiones entre clases similares. Por ejemplo, "Nevus" se clasifica erróneamente como "Melanoma" en varias ocasiones.
Carcinoma Basocelular y Carcinoma Escamoso muestran errores de clasificación cruzados.
Dermatofibroma parece ser una de las clases mejor clasificadas, con muchas predicciones correctas.
Esto sugiere que el modelo tiene problemas de especificidad y puede estar confundiendo características visuales similares.

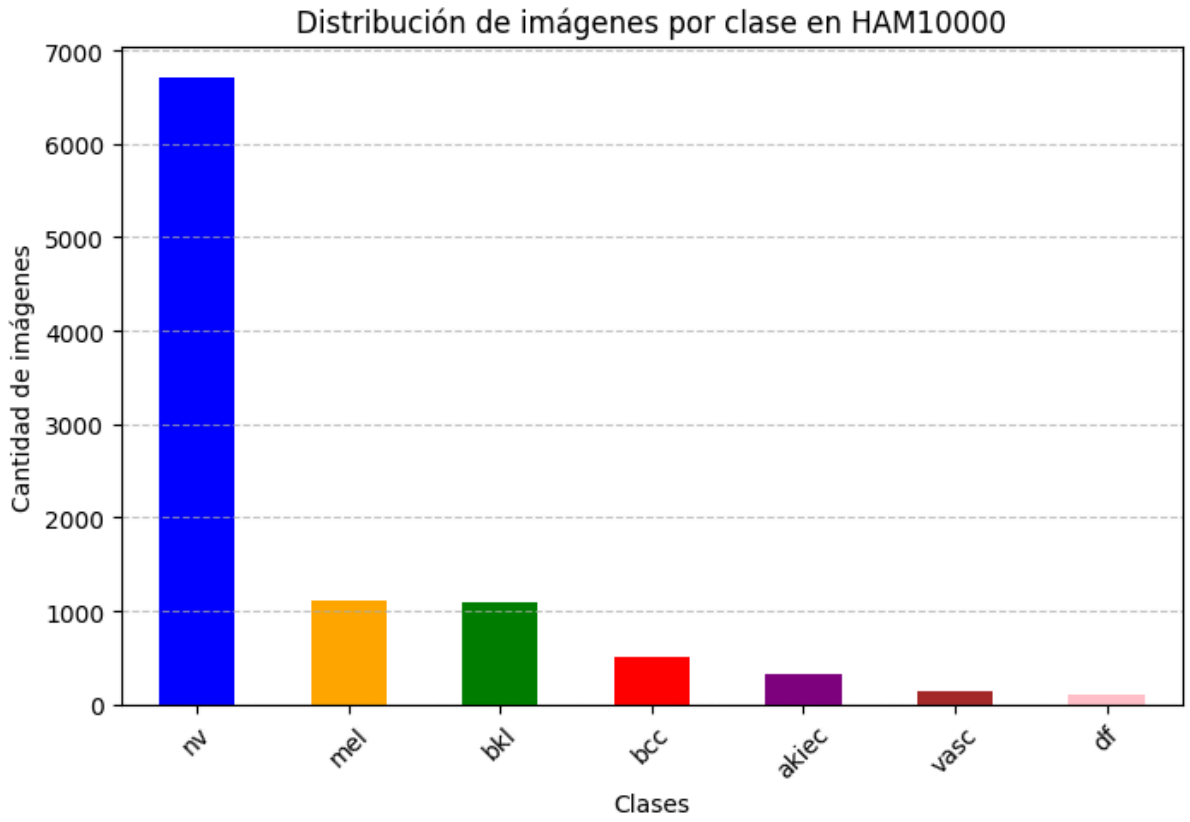


Se aprecia un fuerte desbalance de clases. La mayoría de las imágenes pertenecen a la categoría "Nevus", lo que puede haber afectado el desempeño del modelo.

Clases como "Lesión Vascular" y "Dermatofibroma" tienen una cantidad mucho menor de muestras, lo que puede hacer que el modelo no tenga suficiente información para aprender a clasificarlas correctamente.



- Los valores de AUC (Área Bajo la Curva) están cercanos a 0.5 para todas las clases, lo que sugiere que el modelo no está logrando una discriminación efectiva.
- Un modelo con un AUC cercano a 0.5 equivale a hacer predicciones aleatorias.
- El hecho de que todas las curvas ROC estén cercanas a la diagonal indica que el modelo no está aprendiendo patrones útiles para diferenciar entre las clases.



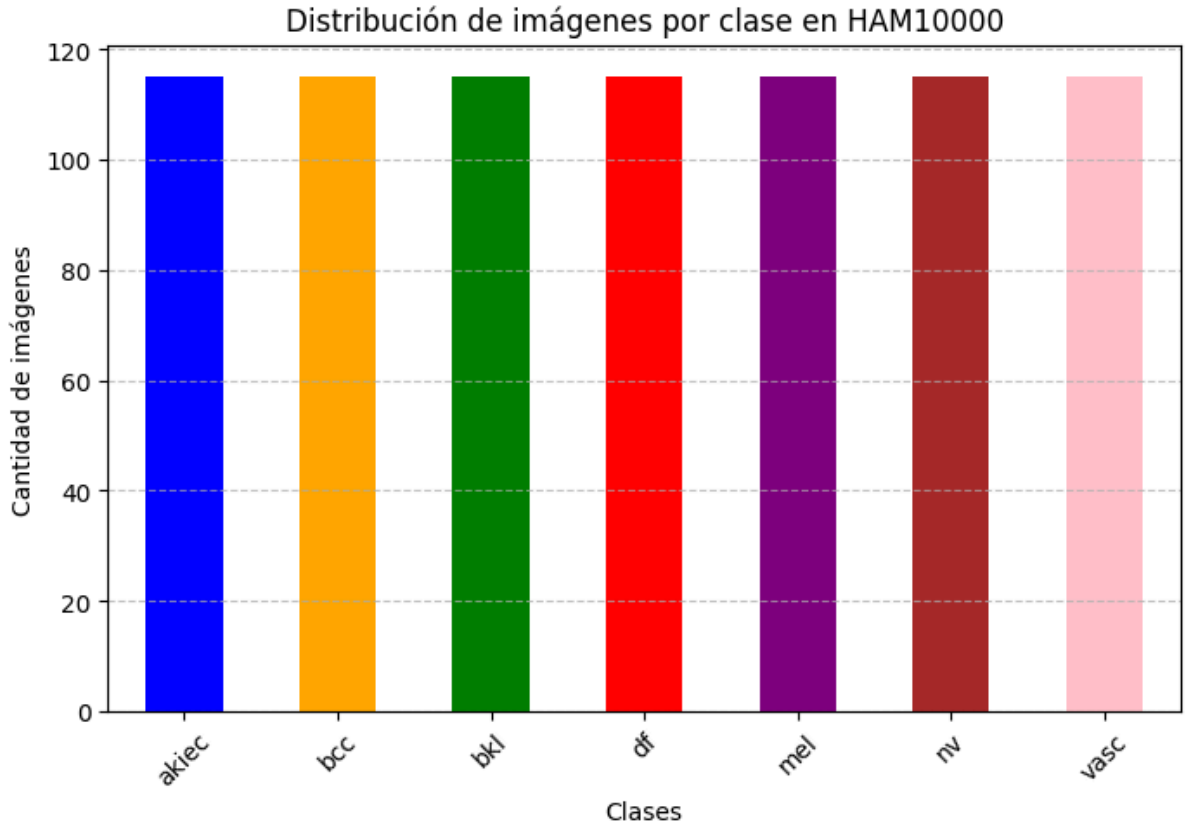
5. Balanceo de dataset

al balancear las clases queda de la siguiente manera.

```python
import pandas as pd

# Cargar el dataset
df = pd.read_csv("HAM10000_metadata.csv")

# Encontrar la cantidad mínima de imágenes en cualquier clase
min_count = df['dx'].value_counts().min()

# Submuestrear cada clase al tamaño mínimo
df_balanced = df.groupby('dx').apply(lambda x: x.sample(min_count, random_state=42)).reset_index(drop=True)

# Verificar distribución balanceada
print(df_balanced['dx'].value_counts())
```

```
dx
akiec    115
bcc      115
bkl      115
df       115
mel      115
nv       115
vasc     115
```

Se usa la técnica de Data augmentation para aumentar en mayor medida las imágenes con las que se puede entrenar el modelo. Se tiene en cuenta que al balancear el muestreo de imágenes, se pierde valiosa información pero que es necesaria para tener un accuracy mayor o igual al 80%



Se equilibra la cantidad de imágenes buscando el muestreo mínimo de las clases para establecerlo como punto de balance.