

## ✓ Clasificación de imágenes histopatológicas de tejido mamario usando redes neuronales convolucionales

Se implementó un modelo de aprendizaje profundo basado en redes neuronales convolucionales con el fin de clasificar imágenes histopatológicas de tejido mamario en dos categorías de forma binaria, es decir en benignas o malignas. Consecuentemente se utilizó un conjunto de datos balanceado de 200 imágenes provenientes del BreakHis Dataset (Breast Cancer Histopathological Images) en Kaggle, procesadas y divididas en conjuntos de entrenamiento y prueba. Tras el preprocesamiento, el modelo fue entrenado y evaluado, alcanzando una precisión del 95% sobre los datos de prueba. Posteriormente, se repitió el mismo proceso, pero con todo el repositorio disponible.

Además, se generaron visualizaciones como la curva de aprendizaje (precisión y pérdida), matriz de confusión y predicciones sobre imágenes individuales para analizar el desempeño del modelo. Por último, se identificaron limitaciones como sensibilidad y se discutieron posibles mejoras.

```
from google.colab import drive
from PIL import Image
import matplotlib.pyplot as plt
import os
import glob
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers, models
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, roc_curve,
import seaborn as sns
```

```
drive.mount('/content/drive')
```

↻ Drive already mounted at /content/drive; to attempt to forcibly remount, call

Antes de procesar las imágenes, se verificó la estructura del directorio, tal que se recorrió la ruta principal que contiene las imágenes y confirmar que el conjunto de datos se cargó correctamente.

```
base_path = '/content/drive/MyDrive/archive'
```

```
for root, dirs, files in os.walk(base_path):
    print(f'{root}')
    for file in files:
        if file.endswith('.png'):
            print(f'{file}')
            break
```

```
⇒ SOB_B_F-14-9133-400-023.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-9133-40-004.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-9133-200-024.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-21998CD-400-008.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-21998CD-40-008.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-21998CD-200-017.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-21998CD-100-021.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-14134-400-002.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-14134-40-003.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-14134-100-001.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-14134-200-003.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-23060AB-400-015.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-23060AB-100-004.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-23060AB-200-010.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_F-14-23060AB-40-005.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
```

```

/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-22704-400-003.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-22704-100-001.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-22704-200-005.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-22704-40-025.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-21998AB-200-010.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-21998AB-400-014.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-21998AB-100-007.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-21998AB-40-004.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-29315EF-400-003.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-29315EF-100-006.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-29315EF-200-008.png
/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/breast
SOB_B_PT-14-29315FF-40-005.png

```

```
!ls "/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slides/brea
```



```

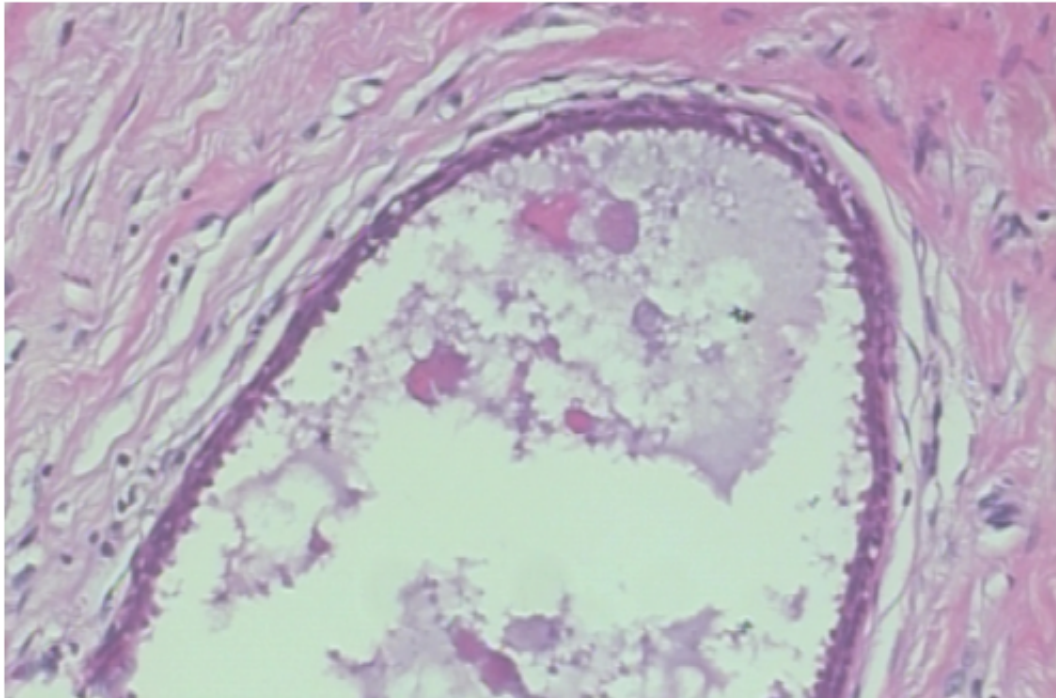
SOB_M_DC-14-20636-200-001.png SOB_M_DC-14-20636-200-015.png
SOB_M_DC-14-20636-200-002.png SOB_M_DC-14-20636-200-016.png
SOB_M_DC-14-20636-200-003.png SOB_M_DC-14-20636-200-017.png
SOB_M_DC-14-20636-200-004.png SOB_M_DC-14-20636-200-018.png
SOB_M_DC-14-20636-200-005.png SOB_M_DC-14-20636-200-019.png
SOB_M_DC-14-20636-200-006.png SOB_M_DC-14-20636-200-020.png
SOB_M_DC-14-20636-200-007.png SOB_M_DC-14-20636-200-021.png
SOB_M_DC-14-20636-200-008.png SOB_M_DC-14-20636-200-022.png
SOB_M_DC-14-20636-200-009.png SOB_M_DC-14-20636-200-023.png
SOB_M_DC-14-20636-200-010.png SOB_M_DC-14-20636-200-024.png
SOB_M_DC-14-20636-200-011.png SOB_M_DC-14-20636-200-025.png
SOB_M_DC-14-20636-200-012.png SOB_M_DC-14-20636-200-026.png
SOB_M_DC-14-20636-200-013.png SOB_M_DC-14-20636-200-027.png
SOB_M_DC-14-20636-200-014.png

```

Se imprimió una imagen muestra para verificar que todo estuviera correcto

```
img_path = '/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slid
img = Image.open(img_path)

plt.imshow(img)
plt.axis('off')
plt.show()
```




```
base_path = '/content/drive/MyDrive/archive/BreakHis_v1/BreakHis_v1/histology_slid
image_paths = glob.glob(os.path.join(base_path, '**', '*.png'), recursive=True)
print(f'Total imágenes: {len(image_paths)}')
```

 Total imágenes: 7909

```
benign_images = [p for p in image_paths if '/benign/' in p]
malignant_images = [p for p in image_paths if '/malignant/' in p]

print(f'Img benignas: {len(benign_images)}')
print(f'Img malignas: {len(malignant_images)}')
```

 Img benignas: 2480  
Img malignas: 5429

Se copió la carpeta con el conjunto de datos desde Drive al entorno de Colab para permitir un acceso más rápido a los archivos durante el entrenamiento del modelo; evitó demoras de lectura directa desde Drive.

```
!cp -r "/content/drive/MyDrive/archive/BreaKHis_v1" "/content/BreaKHis_v1"
```

 ^C

```
BASE_DIR = '/content/BreaKHis_v1/BreaKHis_v1/histology_slides/'
```

Se recorrieron los datos para cargar un número limitado de imágenes benignas y malignas (100 de cada clase) con el fin de equilibrar las clases y reducir el tiempo de entrenamiento. Cada imagen se redimensionó a 150×150 píxeles y normalizó dividiendo sus valores entre 255.

Finalmente, se generó un arreglo X con las imágenes procesadas y otro con las etiquetas correspondientes (0 para benignas y 1 para malignas).

```

BASE_DIR = '/content/drive/MyDrive/archive/BreaKHis_v1/BreaKHis_v1/histology_slides'
IMG_SIZE = 150

data = []
labels = []
benign_count = 0 # Limitamos para la carga
malignant_count = 0
MAX_PER_CLASS = 100

for root, dirs, files in os.walk(BASE_DIR):
    for file in files:
        if not file.endswith('.png'):
            continue

        file_path = os.path.join(root, file)

        if 'benign' in file_path and benign_count < MAX_PER_CLASS:
            label = 0
            benign_count += 1
        elif 'malignant' in file_path and malignant_count < MAX_PER_CLASS:
            label = 1
            malignant_count += 1
        else:
            continue

        img = cv2.imread(file_path)
        if img is None:
            continue
        img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
        data.append(img)
        labels.append(label)
        if benign_count >= MAX_PER_CLASS and malignant_count >= MAX_PER_CLASS: #
            break

X = np.array(data) / 255.0
y = np.array(labels)

print(f"Total: {len(X)}")
print(f"Benignas: {np.sum(y == 0)}, Malignas: {np.sum(y == 1)}")

```

➡ Total: 200  
Benignas: 100, Malignas: 100

Se dividió el conjunto de datos en dos, de entrenamiento y de prueba.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

Se construyó la CNN con Keras, este cuenta con tres bloques de capas seguidos de MaxPooling. Luego, se aplanan los mapas de activación y se pasan por una capa densa con Dropout para evitar sobreajuste. Finalmente, una capa sigmoid permitió clasificar entre tumores benignos y malignos.

Se compiló el modelo usando el optimizador Adam, la función de pérdida binary\_crossentropy y la métrica de accuracy.

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D(2, 2),

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

    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D(2, 2),

    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])

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

model.summary()
```

```

→ /usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv2d.py:100:
    super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential_2"

```

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_5 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_6 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_6 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_7 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_7 (MaxPooling2D)	(None, 17, 17, 128)	0
flatten_2 (Flatten)	(None, 36992)	0
dense_5 (Dense)	(None, 128)	4,735,104
dropout_2 (Dropout)	(None, 128)	0
dense_6 (Dense)	(None, 1)	129

**Total params:** 4,828,481 (18.42 MB)  
**Trainable params:** 4,828,481 (18.42 MB)  
**Non-trainable params:** 0 (0.00 B)



```
history = model.fit(
    X_train, y_train,
    epochs=10,
    validation_data=(X_test, y_test),
    batch_size=16
)
```

```

Epoch 1/10
10/10 ————— 7s 185ms/step — accuracy: 0.5883 — loss: 0.8049 — val_accuracy: 0.4760 — val_loss: 0.6879
Epoch 2/10
10/10 ————— 0s 19ms/step — accuracy: 0.4760 — loss: 0.6879 — val_accuracy: 0.5109 — val_loss: 0.6672
Epoch 3/10
10/10 ————— 0s 19ms/step — accuracy: 0.5109 — loss: 0.6672 — val_accuracy: 0.5993 — val_loss: 0.6511
Epoch 4/10
10/10 ————— 0s 20ms/step — accuracy: 0.5993 — loss: 0.6511 — val_accuracy: 0.6307 — val_loss: 0.5618
Epoch 5/10
10/10 ————— 0s 20ms/step — accuracy: 0.6307 — loss: 0.5618 — val_accuracy: 0.6489 — val_loss: 0.7464
Epoch 6/10
10/10 ————— 0s 23ms/step — accuracy: 0.6489 — loss: 0.7464 — val_accuracy: 0.7498 — val_loss: 0.5990
Epoch 7/10
10/10 ————— 0s 23ms/step — accuracy: 0.7498 — loss: 0.5990 — val_accuracy: 0.6777 — val_loss: 0.5570
Epoch 8/10
10/10 ————— 0s 23ms/step — accuracy: 0.6777 — loss: 0.5570 — val_accuracy: 0.7381 — val_loss: 0.5607
Epoch 9/10
10/10 ————— 0s 19ms/step — accuracy: 0.7381 — loss: 0.5607 — val_accuracy: 0.8413 — val_loss: 0.4240
Epoch 10/10
10/10 ————— 0s 19ms/step — accuracy: 0.8413 — loss: 0.4240 — val_accuracy: 0.8413 — val_loss: 0.4240

```

Aquí graficó la evolución de la precisión y pérdida a lo largo de las épocas tanto para los datos de entrenamiento como de validación, del cual se observaron buenos resultados.

```


# Precisión
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.xlabel('Época')
plt.ylabel('Precisión')
plt.legend()

# Pérdida
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Entrenamiento')

```

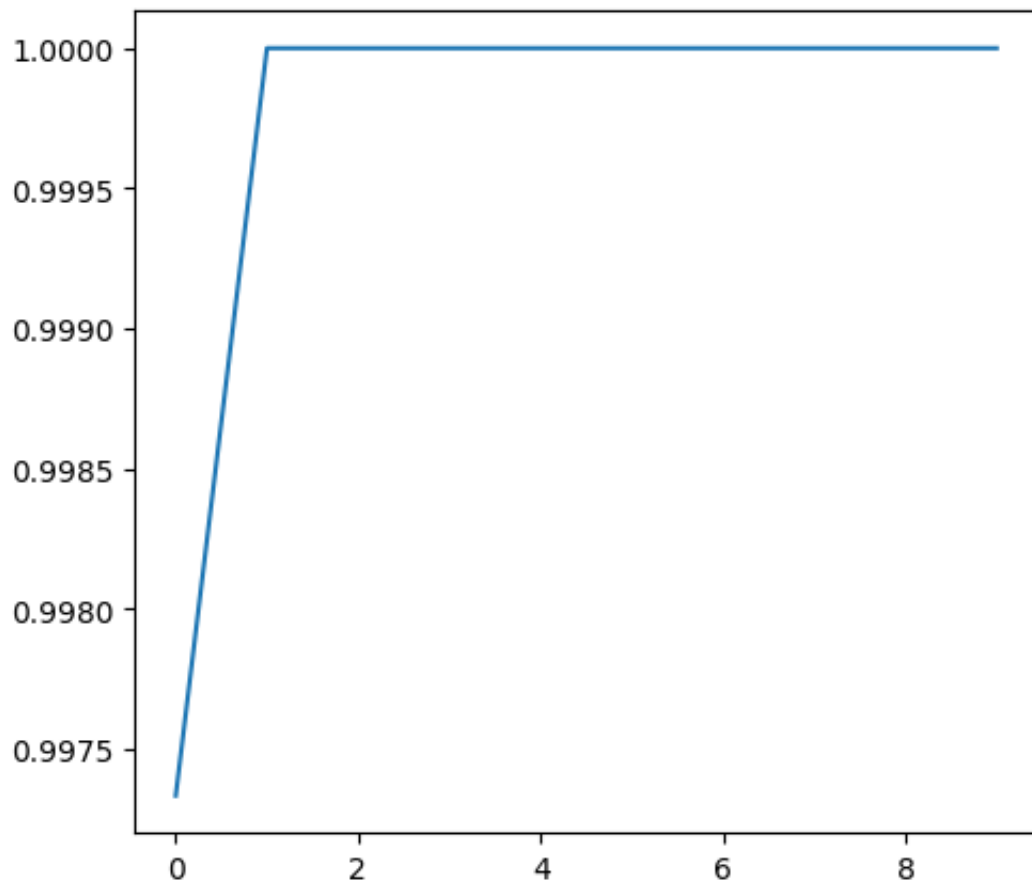
```
plt.plot(history.history['val_loss'], label='Validación')
plt.title('Pérdida')
plt.xlabel('Época')
plt.ylabel('Pérdida')
plt.legend()

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

 -----  
**KeyError** Traceback (most recent call last)  
<ipython-input-80-3631319a4034> in <cell line: 0>()  
3 plt.subplot(1, 2, 1)

```
4 plt.plot(history.history['accuracy'], label='Entrenamiento')
----> 5 plt.plot(history.history['val_accuracy'], label='Validación')
6 plt.title('Precisión')
7 plt.xlabel('Época')
```

**KeyError:** 'val\_accuracy'



Próximos pasos: [Explicar error](#)

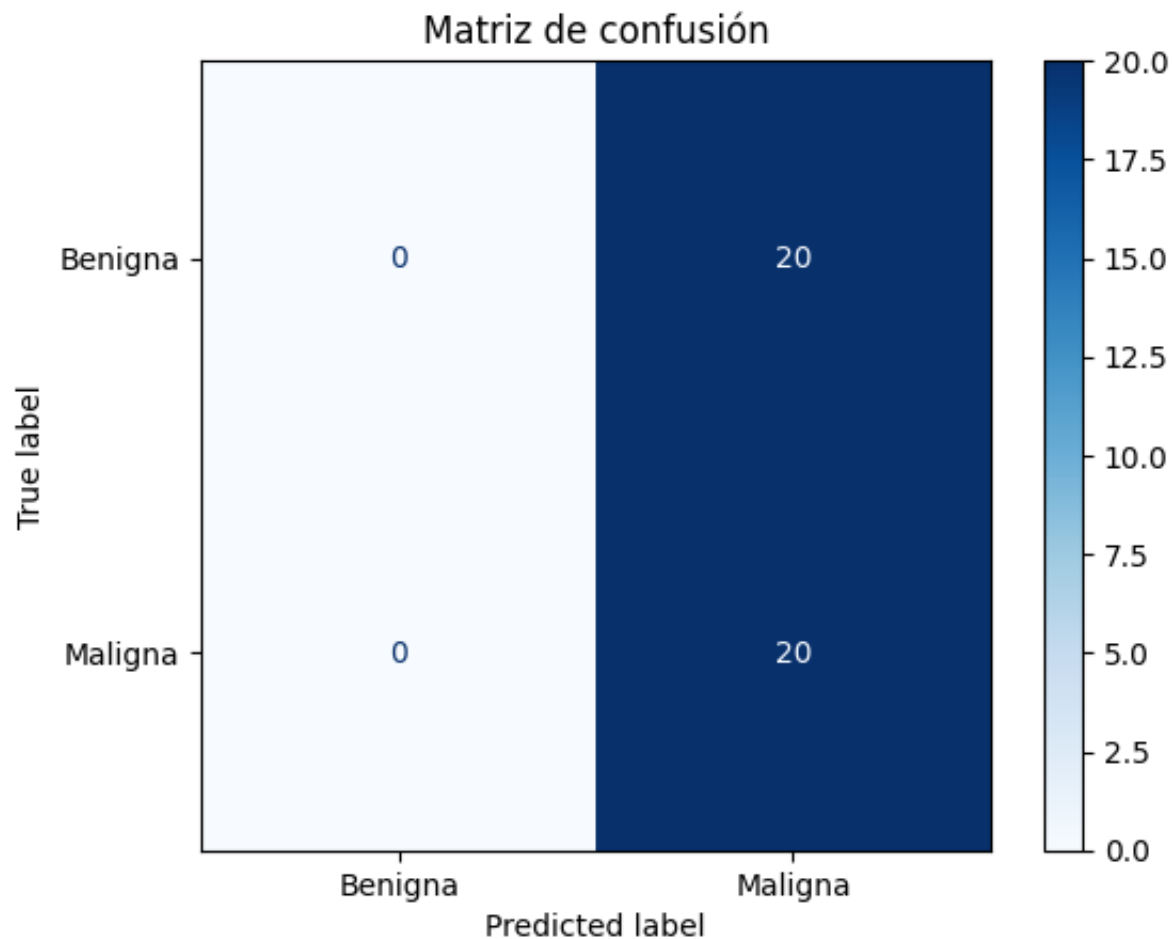
```

y_pred_probs = model.predict(X_test)
y_pred = (y_pred_probs > 0.5).astype("int32")

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Benigna", "Maligna"])
disp.plot(cmap='Blues')
plt.title("Matriz de confusión")
plt.show()

```

2/2 0s 19ms/step



```

indices = random.sample(range(len(X_test)), 10)
plt.figure(figsize=(15, 5))

for i, idx in enumerate(indices):
    img = X_test[idx]
    true_label = "Benigna" if y_test[idx] == 0 else "Maligna"
    pred_label = "Benigna" if y_pred[idx] == 0 else "Maligna"

    color = "green" if true_label == pred_label else "red"

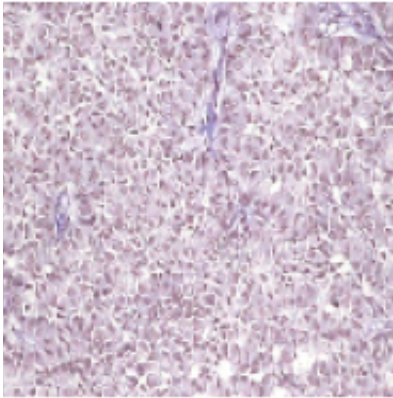
```

```
plt.subplot(2, 5, i+1)
plt.imshow(img)
plt.title(f"Real: {true_label}\nPred: {pred_label}", color=color)
plt.axis("off")

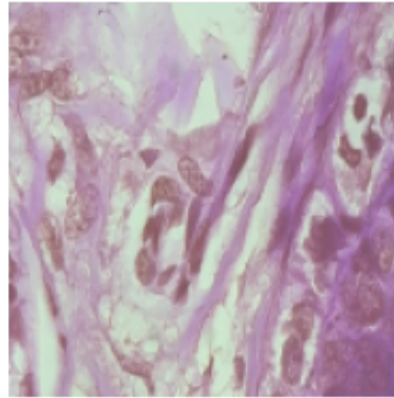
plt.tight_layout()
plt.show()
```



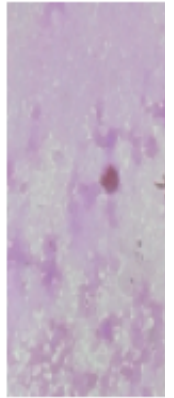
Real: Maligna  
Pred: Maligna



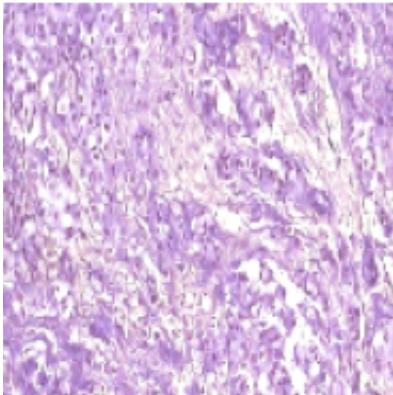
Real: Benigna  
Pred: Benigna



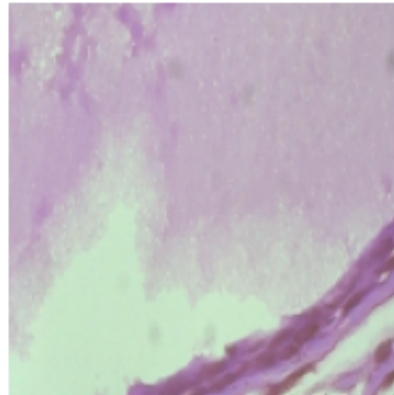
Real:  
Pred:



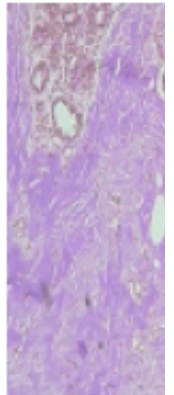
Real: Maligna  
Pred: Maligna



Real: Benigna  
Pred: Benigna



Real:  
Pred:



Ya completo, el modelo se entrena con lotes de imágenes (32 por lote) con la estructura previa. La red neuronal convolucional se compone de dos capas de convolución seguidas de capas de max-pooling, con una capa densa intermedia y un Dropout para evitar sobreajuste. Al final, la salida es una capa densa con activación sigmoid para clasificación binaria.

El entrenamiento se realiza durante 10 épocas, con el objetivo de minimizar la función de pérdida `binary_crossentropy` y optimizar la precisión del modelo.

```
train_dir = '/content/drive/MyDrive/archive/BreaKHis_v1/BreaKHis_v1/histology_slides'
datagen = ImageDataGenerator(rescale=1./255) # Generador para cargar img con rescale
train_generator = datagen.flow_from_directory( # Lotes
    train_dir,
    target_size=(150, 150), # Redimensionamos -> 150x150
    batch_size=32,
    class_mode='binary',    # Cómo estamos clasificando benigno vs maligno
    shuffle=True
)
print(f"Imágenes cargadas: {train_generator.samples}")
print(f"Clases encontradas: {train_generator.class_indices}")

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid') # Salida binaria
])

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

# Entrenamiento
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // 32, # Total de imágenes / tamaño
    epochs=10
)
```

```
print(f"Entrenamiento éxito")
```

```

➞ Found 7909 images belonging to 1 classes.
Imágenes cargadas: 7909
Clases encontradas: {'breast': 0}
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv2d.py:100:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:140:
  self._warn_if_super_not_called()
Epoch 1/10
247/247 ————— 2004s 8s/step – accuracy: 0.9838 – loss: 0.0185
Epoch 2/10
247/247 ————— 9s 68us/step – accuracy: 1.0000 – loss: 0.0000e+00
Epoch 3/10
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/epoch_iterator.py:120:
  self._interrupted_warning()
247/247 ————— 195s 585ms/step – accuracy: 1.0000 – loss: 1.5600e+00
Epoch 4/10
247/247 ————— 1s 53us/step – accuracy: 1.0000 – loss: 0.0000e+00
Epoch 5/10
247/247 ————— 135s 548ms/step – accuracy: 1.0000 – loss: 3.7780e+00
Epoch 6/10
247/247 ————— 1s 43us/step – accuracy: 1.0000 – loss: 0.0000e+00
Epoch 7/10
247/247 ————— 132s 534ms/step – accuracy: 1.0000 – loss: 1.3330e+00
Epoch 8/10
247/247 ————— 1s 45us/step – accuracy: 1.0000 – loss: 0.0000e+00
Epoch 9/10
247/247 ————— 133s 538ms/step – accuracy: 1.0000 – loss: 4.0600e+00
Epoch 10/10
247/247 ————— 1s 51us/step – accuracy: 1.0000 – loss: 0.0000e+00
Entrenamiento éxito

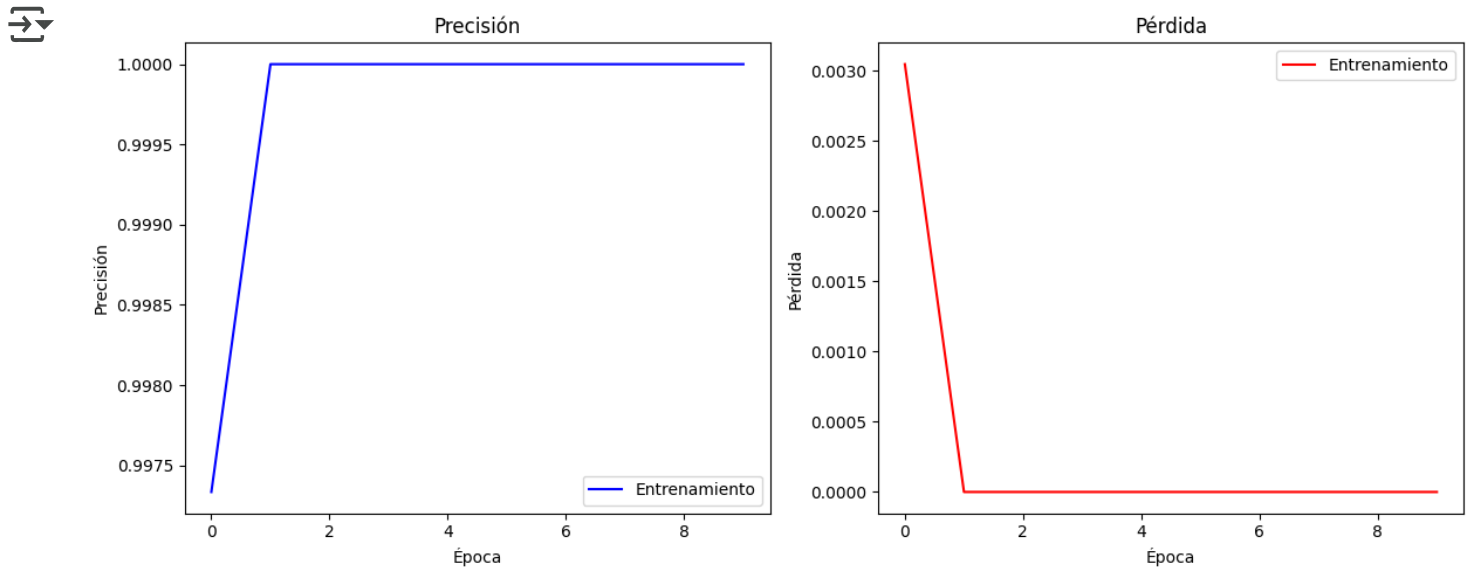
```

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

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Entrenamiento', color='blue')
plt.title('Precisión')
plt.xlabel('Época')
plt.ylabel('Precisión')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Entrenamiento', color='red')
plt.title('Pérdida')
plt.xlabel('Época')
plt.ylabel('Pérdida')
plt.legend()

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

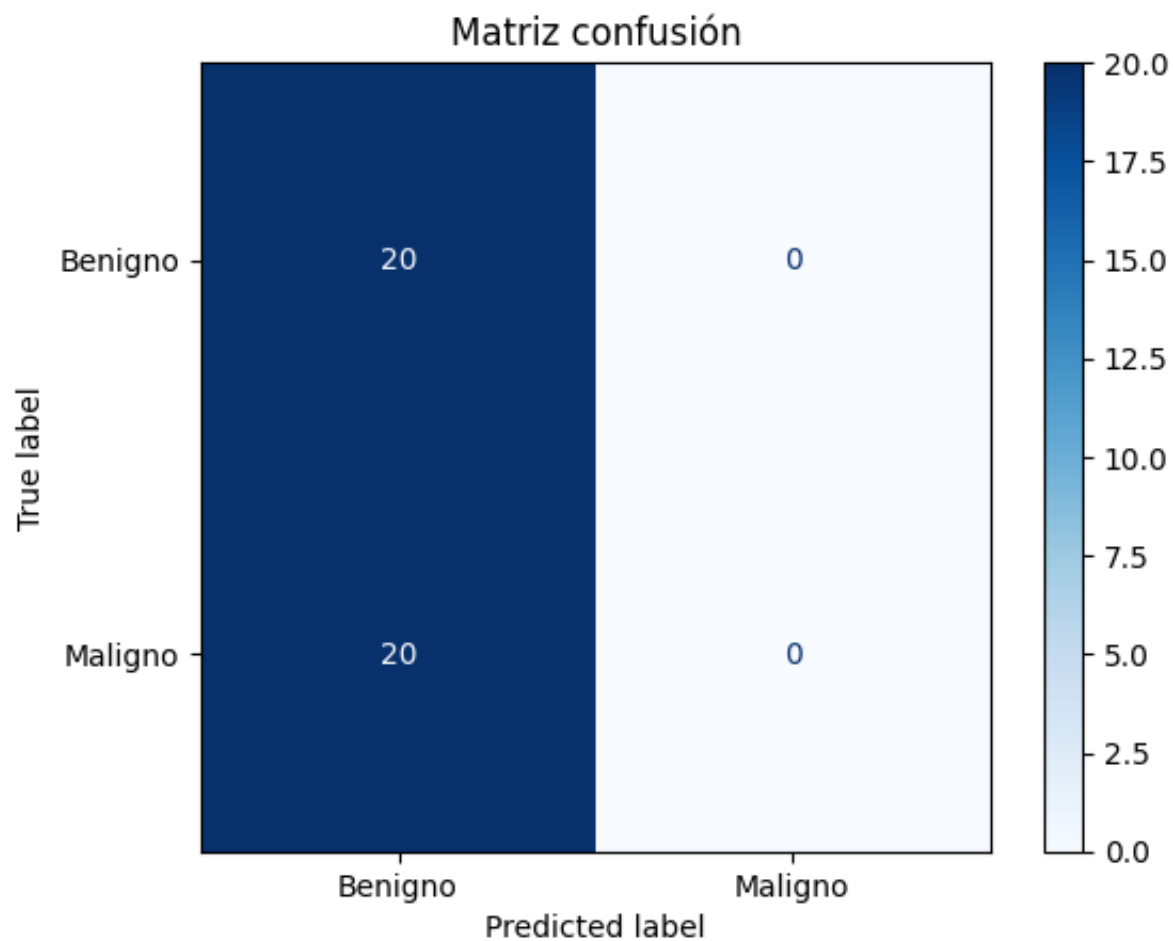


```
y_pred = model.predict(X_test)
y_pred_classes = (y_pred > 0.5).astype("int32")

cm = confusion_matrix(y_test, y_pred_classes)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Benigno", "Maligno"])

plt.figure(figsize=(6, 6))
disp.plot(cmap="Blues", values_format='d')
plt.title("Matriz confusión")
plt.show()
```

2/2 0s 17ms/step  
<Figure size 600x600 with 0 Axes>





# Conclusión

Tras entrenar el modelo, se logró un alto rendimiento por tanto se puede concluir que las técnicas empleadas son efectivas para la clasificación binaria de las imágenes y que tiene el potencial de contribuir a la detección temprana de cáncer de mama, mejorando la precisión en el diagnóstico, no obstante se debe ún probar con más bancos de imágenes, con aún más diferentes aumentos lo cual puede representar un importante desafío.