SISTEMA INTELIGENTE DE PREDICCIÓN TEMPRANA DE PURPURA REUMATOIDEA USANDO IMÁGENES MÉDICAS DE LA PIEL Y TÉCNICAS DEEP LEARNING COMO APOYO AL DIAGNÓSTICO PARA LOS ESPECIALISTAS DE LA SALUD

Hernán Peñaranda[1], Juan Arciniegas[2], José Gerardo Chacón Rangel[3]

[1] Hernán Peñaranda, Grupo de Investigación GIIDAC Universidad de Pamplona Facultad de ingenierías y arquitectura, Programa de ingeniería de Sistemas, Villa del Rosario, Norte de Santander, Colombia. ORCID: https://orcid.org/0009-0008-7181-5932 E-mail: hernan.penaranda@unipamplona.edu.co

[2] Juan Arciniegas[2], Grupo de Investigación GIIDAC Universidad de Pamplona Facultad de ingenierías y arquitectura, Programa de ingeniería de Sistemas, Villa del Rosario, Norte de Santander, Colombia. ORCID: https://orcid.org/0009-0006-4135-5029

E-mail: juan.arciniegas@unipamplona.edu.co

[3] José Gerardo Chacón Rangel, Grupo de Investigación Inteligencia de Datos y Computación (GIIDAC), Grupo de Investigación Automatización y Control A&C Universidad de Pamplona, Villa del Rosario, Norte de Santander, Colombia. ORCID: https://orcid.org/0000-0002-6582-3142.

E-mail: jose.chacon@unipamplona.edu.co.

Resumen

La importancia de desarrollar un sistema de detección de la Púrpura Reumatoidea basado en Deep Learning, radica en la detección temprana, la reducción de la carga para profesionales de la salud y la contribución al acceso médico en áreas remotas. Además, la investigación promueve avances científicos en dermatología e inteligencia artificial aplicada a la salud. El objetivo del estudio se centra en la creación de un sistema de detección temprana de Purpura Reumatoidea usando imágenes médicas de la piel y técnicas Deep Learning, que pueda ayudar a los profesionales de la salud a identificar y diagnosticar casos de Purpura Reumatoidea de manera más efectiva y rápida, lo que a su vez puede contribuir a una atención médica más eficiente. La metodología a utilizar será la metodología Knowledge Discovery in Databases (KDD). Las fases de la metodología son: 1. Preprocesamiento/limpieza. 2.Transformación/reducción. 3.Minería de datos (data mining). 4.Interpretación/evaluación. Se recolectan un conjunto de datos de imágenes médicas de alta calidad que incluyan casos positivos y negativos de Purpura Reumatoidea de bases de datos públicas y libres, así como una variedad de manifestaciones clínicas de la enfermedad. Se realizará el preprocesamiento de las imágenes para garantizar que estén en formato adecuado y que se eliminen artefactos que puedan interferir con el análisis. Se espera presentar el sistema inteligente en Python, que además cuenta con una interfaz gráfica que ayuda a los especialistas de la salud a identificar esta enfermedad.

Palabras clave: *Dermatología, Piel, Deep Learning, Inteligencia Artificial, Minería de Datos.*

Abstract

The importance of developing a Deep Learning-based rheumatoid purpura detection system lies in early detection, reducing the burden on healthcare professionals and contributing to medical access in remote areas. In addition, the research promotes scientific advances in dermatology and artificial intelligence applied to healthcare. The objective of the study focuses on the creation of an early detection system for Rheumatoid Purpura using medical skin imaging and Deep Learning techniques, which can help healthcare professionals identify and diagnose cases of Rheumatoid Purpura more effectively and quickly, which in turn can contribute to more efficient medical care. The methodology to be used will be the Knowledge Discovery in Databases (KDD) methodology. The phases of the methodology are: 1. Preprocessing/cleaning. Transformation/reduction. 3.Data mining. Interpretation/evaluation. A high-quality medical imaging dataset including positive and negative cases of Purpura Rheumatoidea will be collected from public and free databases, as well as a variety of clinical manifestations of the disease. The images will be preprocessed to ensure that they are in proper format and that artifacts that could interfere with the analysis are removed. It is expected to present the intelligent system in Python, which also has a graphical interface to help health specialists identify this disease.

Keywords: *Dermatology, skin, Deep Learning, Artificial Intelligence, Data Mining.*

Introducción

En el contexto actual, la predicción temprana de enfermedades autoinmunes como la púrpura reumatoidea representa un desafío significativo en el campo de la medicina. La púrpura reumatoidea es una enfermedad crónica y progresiva que afecta principalmente las articulaciones, causando inflamación y daño articular. Detectar esta enfermedad en sus etapas iniciales es fundamental para brindar un tratamiento efectivo y mejorar la calidad de vida de los pacientes. En este sentido, el uso de imágenes médicas de la piel y técnicas avanzadas de deep learning ha surgido como una herramienta prometedora para la predicción temprana de la púrpura reumatoidea.

Las imágenes médicas de la piel, como la termografía y la fotografía de alta resolución, ofrecen una visión detallada de los cambios cutáneos asociados con la púrpura reumatoidea. Estas imágenes pueden revelar patrones específicos de inflamación y vascularización que son indicativos de la presencia de la enfermedad, incluso en sus fases iniciales. Al combinar estas imágenes con algoritmos de deep learning, es posible analizar de manera precisa y eficiente una gran cantidad de datos para identificar posibles marcadores tempranos de la púrpura reumatoidea. La integración de la inteligencia artificial en el diagnóstico médico ha revolucionado la forma en que se abordan las enfermedades complejas. Los modelos de deep learning pueden aprender de manera autónoma a partir de grandes conjuntos de datos, lo que les permite identificar patrones sutiles que podrían pasar desapercibidos para el ojo humano. En el caso de la púrpura reumatoidea, estos modelos pueden ser entrenados con imágenes de la piel de pacientes con la enfermedad para reconocer características distintivas que indiquen su presencia, incluso antes de que aparezcan los síntomas clínicos.

La importancia de esta investigación radica en su potencial para transformar la forma en que se diagnostican y tratan las enfermedades autoinmunes. Al permitir la detección temprana de la púrpura reumatoidea, se abre la puerta a intervenciones médicas preventivas que podrían retrasar la progresión de la enfermedad y mejorar el pronóstico de los pacientes. Además, el desarrollo de un modelo preciso de predicción temprana podría allanar el camino para aplicaciones similares en otras condiciones médicas, ampliando así el alcance y el impacto de esta innovadora tecnología.

Materiales y métodos

La metodología para el desarrollo del sistema de detección temprana de Púrpura Reumatoidea (PR) se basa en el proceso KDD (Knowledge Discovery in Databases) y se compone de las siguientes etapas:

Recolección de datos

Imágenes médicas de la piel de pacientes con Purpura Reumatoidea (PR) y sin Purpura Reumatoidea SPR: Las imágenes deben ser de alta calidad y mostrar las características relevantes de la enfermedad, como la erupción cutánea. En la ilustración 1 se muestra imágenes de una persona sana y en la ilustración 2 se muestran imagen de una persona con la enfermedad de purpura reumatológica (PR) .

Ilustración 1 imágenes de una persona sana

Ilustración 1 Sin Purpura Reumatoidea

fuente: https://www.shutterstock.com/

Ilustración imágenes de una persona con PR

Ilustración 2 Purpura Reumatoidea

Fuente https://www.reumatologiaclinica.org/

Los datos utilizados en este trabajo se tomaron de bases de datos públicas en la literatura especializada de imágenes médicas y de revistas médicas. Cada imagen posee edad, sexo, los síntomas y el diagnóstico de PR.

Preprocesamiento de datos

A continuación, se presentan las ilustraciones sobre el preprocesamiento de datos

Carga de imágenes:

Ilustración 3 Carga de las imágenes

```
# Definir el directorio de datos
data_dir = './CD/data'

# Crear la estructura de carpetas
os.makedirs(data_dir, exist_ok=True)
os.makedirs(os.path.join(data_dir, 'purpura_reumatoidea'), exist_ok=True)
os.makedirs(os.path.join(data_dir, 'saludable'), exist_ok=True)
```

Fuente: Autores

La ilustración 3 muestra el código de

- Como se recorre las carpetas de categorías (`purpura_reumatoidea` y `saludable`) definidas en `data_dir`.
- Para cada imagen dentro de una carpeta:
    - Intenta cargar la imagen usando `plt.imread`.
    - Si la carga es exitosa, redimensiona la imagen a `img_size` (128x128 píxeles) usando `tf.image.resize`.

Creación de la estructura de datos:

Ilustración 4 Preprocesamiento de los Datos

```
def load_data(data_dir, categories, img_size):
    data = []
    for category in categories:
        path = os.path.join(data_dir, category)
        class_num = categories.index(category)
        for img in os.listdir(path):
            try:
                img_array = plt.imread(os.path.join(path, img))
                resized_img = tf.image.resize(img_array, (img_size,
img_size))
                data.append([resized_img, class_num])
            except Exception as e:
                pass
    return data
```

Fuente Autores

La ilustración 4 muestra el código de

- Como se almacena la imagen redimensionada y su etiqueta de clase (`class_num`) correspondiente en una lista.
- La etiqueta de clase se obtiene del índice de la categoría actual en la lista `categories`.

Barajado y separación de características y etiquetas:

Ilustración 5 Barajado y separación de características

```
# Mezclar los datos y dividir en características y etiquetas
np.random.shuffle(data)
X = []
y = []

for features, label in data:
    X.append(features)
    y.append(label)
```

Fuente: Autores

La ilustración 5 muestra el código de

- Se baraja aleatoriamente la lista de datos usando `np.random.shuffle`.
- Separa las imágenes (`X`) de las etiquetas (`y`) creando dos listas independientes.

A continuación, se muestran las ilustraciones sobre la normalización y estandarización del formato de las imágenes

Normalización y Estandarización del Formato de las Imágenes

La ilustración 6 presenta el código de la normalización y Estandarización del Formato de las Imágenes.

1. **Redimensionamiento de Imágenes:**

Ilustración 6 Normalización y Estandarización de Imágenes

```
# Configurar el tamaño de las imágenes
img_size = 128
```

Fuente: Autores

La ilustración 7 presenta el código de cómo se definen las categorías en las cuales se dividen las imágenes.

2. **Definir categorías de las imágenes:**

Ilustración 7 Definición por categorías de Imágenes

```
categories = ['purpura_reumatoidea', 'saludable']
```

Fuente:Autores

La ilustración 8 presenta el código de como se mezclaron los datos y se dividieron en características y etiquetas.

3. **Mezclar los datos y dividir en características y etiquetas**

Ilustración 8 Mezcla y división de datos por categorías y etiquetas

```
np.random.shuffle(data)
X = []
y = []

for features, label in data:
    X.append(features)
    y.append(label)

X = np.array(X) / 255.0  # Normalizar las imágenes
y = np.array(y)
```

Fuente: Autores

La ilustración 8 muestra el código de

- Como se convierten todas las imágenes en un array de NumPy (`X`).
- Como se divide todos los valores del array por 255.0 para normalizar los valores de los píxeles a un rango entre 0 y 1.

4. Dividir los datos en conjuntos de entrenamiento y prueba

A continuación, presentamos la ilustración 9 sobre la división de los datos en conjuntos.

Ilustración 9 División de datos en conjuntos de entrenamiento y prueba

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

Fuente: Autores

La ilustración 9 presenta el código de

- Como se divide el conjunto de datos combinado en conjuntos de entrenamiento y prueba utilizando `train_test_split` de `sklearn.model_selection`.
- La división que se realiza con una proporción 80% para entrenamiento y 20% para prueba (`test_size=0.2`).
- Como se fija una semilla aleatoria (`random_state=42`) para garantizar resultados reproducibles al dividir los datos.

Luego Se aplicaron técnicas de aumento de datos, como rotación, cambio de escala y recorte, para aumentar el tamaño del conjunto de datos y mejorar la generalización del modelo.

A continuación, se mostrará la ilustración 10 que contiene la transformación y reducción de datos

Transformación y reducción de datos

Ilustración 10 Transformación y reducción de datos

```
# Generación de datos de imagen aumentada para el entrenamiento
train_datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest"
)

train_generator = train_datagen.flow(X_train, y_train, batch_size=32)
```

Fuente Autores

La ilustración 10 presenta el código de cómo se extrajeron características relevantes de las imágenes que permitan diferenciar entre casos de PR y casos sin PR. Estas características pueden ser extraídas manualmente o utilizando técnicas de aprendizaje automático. Se selecciono un subconjunto de las características extraídas que sean más discriminatorias y reduzcan la dimensionalidad del conjunto de datos.

• Color**:** Diferencias en la distribución de colores, como el aumento de tonos rojos o morados en las lesiones de PR.

Ilustración 11

fuente: https://www.sciencedirect.com/

Ilustración 12

fuente: https://www.sciencedirect.com

Ilustración 13

fuente: https://www.sciencedirect.com/

• Textura**:** Patrones de textura únicos en las áreas afectadas, como rugosidad o descamación.

Ilustración 14

fuente: https://www.researchgate.net/

Ilustración 15

fuente: https://www.sciencedirect.com/

• Forma y tamaño: Como lesiones redondas u ovaladas.

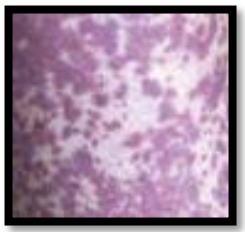
Ilustración 18

fuente: https://www.reumatologiaclinica.org/

Ilustración 19

https://www.reumatologiaclinica.org/

Ilustración 20



https://www.reumatologiaclinica.org/

Ilustración 21

fuente: https://www.sciencephoto.com/

A continuación, se mostrará la ilustración del código de entrenamiento del modelo propuesto

Entrenamiento del modelo de redes neuronales convolucionales

Ilustración 22

```
# Construcción de la red neuronal convolucional
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_size, img_size,
3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(2, activation='softmax')
])

model.compile(optimizer=Adam(lr=0.001),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Entrenar el modelo
history = model.fit(train_generator, epochs=30, validation_data=(X_test,
y_test))

# Evaluar el modelo
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Pérdida: {loss}, Precisión: {accuracy}')
```

Fuente: Autores

La ilustración 22 presenta el código de cómo se entrenó el modelo de Deep Learning, como una red neuronal convolucional (CNN), utilizando el conjunto de datos preprocesado y transformado. El modelo se entrenó para aprender a identificar la presencia de PR en las imágenes médicas de la piel. La evaluación se realizo utilizando métricas como la precisión, la sensibilidad y la especificidad.

Interpretación y evaluación

Se analizaron los resultados del modelo entrenado y se comprendido cómo identifica la presencia de PR en las imágenes médicas de la piel. Se hizo mediante técnicas de visualización e interpretación de modelos. Se realizo una evaluación clínica del sistema de detección temprana de PR con un grupo de profesionales de la salud. La evaluación se centró en la facilidad de uso, la precisión y la confiabilidad del sistema.

En la ilustración 23 se muestra las fases de la metodología seleccionada y los objetivos específicos propuestos para cada una de ellas.

Iluatracion 23 Fases de la Metodología KDD y Objetivos específicos propuestos

Fase	Objetivo específico
Recolección de datos	Obtuvimos un conjunto de datos de alta calidad de imágenes médicas de la piel y datos clínicos de pacientes con PR y sin PR.
Preprocesamiento de datos	Limpiar, normalizar y aumentar los datos para prepararlos para el análisis.
Transformación y reducción de datos	Extraer y seleccionar características relevantes de las imágenes médicas de la piel.
Minería de datos	Entrenar y evaluar un modelo de Deep Learning para identificar la presencia de PR en las imágenes médicas de la piel.
Interpretación y evaluación	Interpretar los resultados del modelo y evaluar su rendimiento en una evaluación clínica.

Fuente: Autores

Tipos de datos y preprocesamiento de datos

- Imágenes médicas de la piel (formato JPEG, PNG, etc.)
- Datos clínicos de pacientes (edad, sexo, síntomas, diagnóstico, etc.
- Limpieza de datos (eliminación de ruido, artefactos, datos faltantes)
- Normalización de datos (normalización de tamaño, formato, rango de valores)
- Aumento de datos (rotación, cambio de escala, recorte)

El lenguaje o plataforma que se utilizó para el desarrollo del sistema de detección temprana de púrpura reumatoidea fue Python el cual es un lenguaje de programación versátil y ampliamente utilizado en el campo de la inteligencia artificial y el aprendizaje automático. ofrece una amplia gama de bibliotecas y herramientas para el procesamiento de imágenes, el aprendizaje profundo y el análisis de datos. en la ilustración 24 se muestra el logo del lenguaje.

Ilustración 24 Logo del lenguaje de programación python

fuente: https://upload.wikimedia.org/wikipedia/commons/thumb/c/c3/Python-logo-notext.svg/800px-Python-logo-notext.svg.png

Se utilizo la librería de Python TensorFlow que es una biblioteca de software de código abierto para el cálculo numérico utilizando gráficos de flujo de datos. Es ampliamente utilizada para el desarrollo de modelos de aprendizaje profundo. En la ilustración 25 se muestra el logo del TensorFlow de Python.

Ilustración 25 logo del TensorFlow de Python.

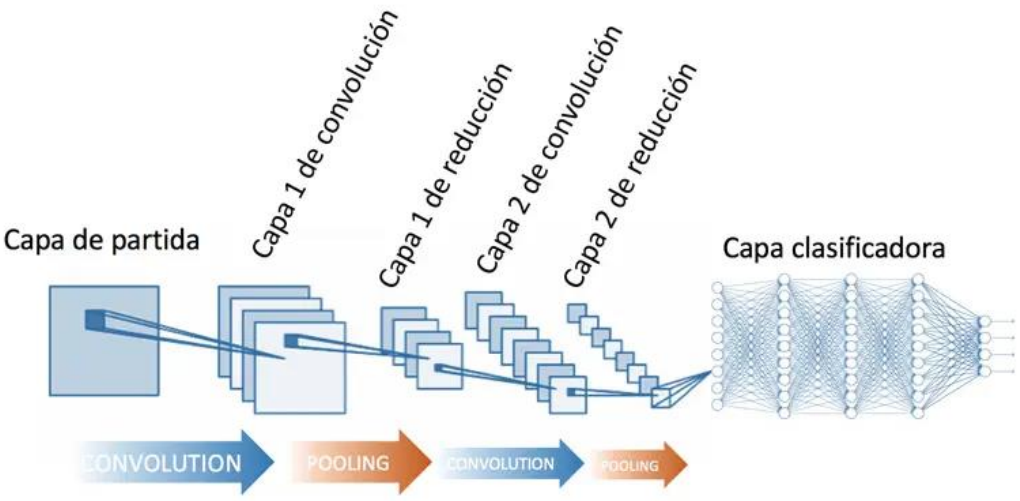
Fuente: https://upload.wikimedia.org/wikipedia/commons/thumb/a/ab/TensorFlow_logo.svg/1200px-TensorFlow_logo.svg.png

EL MODELO

La etapa central del proyecto implico el diseño y entrenamiento de un modelo de aprendizaje profundo, específicamente una red neuronal convolucional (CNN). Este modelo es seleccionado por su capacidad para aprender automáticamente patrones complejos en las imágenes médicas. El proceso de entrenamiento implica ajustes cuidadosos de parámetros y arquitecturas para maximizar la precisión en la detección de casos de PR. Además, se explorará la interpretabilidad del modelo para garantizar que los resultados sean comprensibles y útiles para los profesionales de la salud.

La ilustración 26 muestra las capas de una Red Neuronal Convolucional CNN.

Ilustración 26 red neuronal convolucional (cnn)



fuente: https://www.diegocalvo.es/wp-content/uploads/2017/07/red-neuronal-convolucional-arquitectura.png

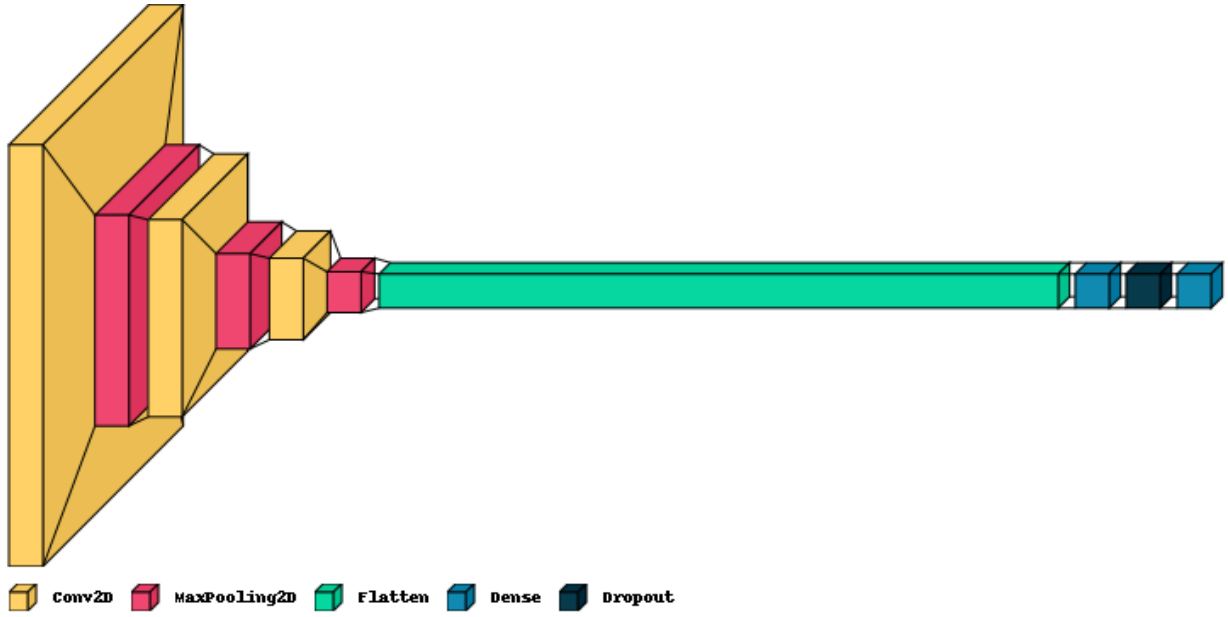
Con estos desarrollos específicos, avanzamos hacia la creación de un sistema inteligente que no solo sea capaz de predecir la presencia de PR en una etapa temprana, sino que también sea robusto y adaptable a diversas condiciones clínicas. Este enfoque integral desde la recopilación de datos hasta el desarrollo del modelo nos coloca en una posición sólida para lograr nuestro objetivo de contribuir significativamente a la predicción y gestión eficiente de Purpura Reumatoidea en entornos clínicos.

Diseño de la Arquitectura CNN

Se realizo el diseño de una arquitectura específica de Red Neuronal Convolucional (CNN) adecuada para la detección de Purpura. Esto implico la elección cuidadosa de capas, funciones de activación y otros parámetros clave para optimizar el rendimiento del modelo.

La ilustración 27 muestra las capas de la Red Neuronal Convolucional Creada.

Ilustración 27 Modelo de capas de la red convolucional creada



Fuente: Autores

Entrenamiento del modelo

Utilizando el conjunto de datos preparado, se llevó a cabo el entrenamiento del modelo (CNN). Este proceso incluirá iteraciones para ajustar los pesos de la red y optimizar su capacidad para identificar patrones relevantes asociados con la Purpura.

Evaluación y Ajuste

Tras el entrenamiento, se evaluó el rendimiento del modelo utilizando métricas específicas, como precisión y sensibilidad. Se realizarán ajustes adicionales en la arquitectura o parámetros según sea necesario para mejorar la capacidad de generalización del modelo.

A continuación, en la ilustración 28 se mostrará el código de la construcción de la Red Neuronal convolucional.

Construcción de la Red Neuronal Convolucional

Ilustración 28 Construcción de la Red Neuronal Convolucional

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_size, img_size,
3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(2, activation='softmax')
])

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

Fuente: Autores

A continuación, se mostrará la ilustración 29 del código del entrenamiento de la Red Neuronal convolucional.

Entrenamiento del modelo

Ilustración 29 Entrenamiento de la Red Neuronal Convolucional

```
history = model.fit(train_generator, epochs=25, validation_data=(X_test,
y_test))
```

Fuente: Autores

La ilustración 29 presenta el código de los parámetros con los cuales se entrenará el modelo

A continuación, se mostrará la ilustración 30 que contiene el código de la Evaluación del modelo de la Red Neuronal convolucional.

La ilustración 30 presenta el código del método de evaluación que utiliza el modelo

Evaluación del modelo

Ilustración 30 Evaluación de la Red Neuronal Convolucional

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Pérdida: {loss}, Precisión: {accuracy}')
```

Fuente: Autores

A continuación, se mostrará la ilustración 31 que contiene el código de las gráficas de resultados del modelo, Precisión y Perdida.

Grafica de resultados: Precisión y Perdida

Ilustración 31 Predicción de perdida y precisión de la Red Neuronal Convolucional

```
# Graficar los resultados de la precisión y pérdida del modelo
plt.plot(history.history['accuracy'], label='Precisión de entrenamiento')
plt.plot(history.history['val_accuracy'], label='Precisión de validación')
plt.xlabel('Época')
plt.ylabel('Precisión')
plt.legend(loc='lower right')
plt.title('Precisión del entrenamiento y la validación')
plt.show()

plt.plot(history.history['loss'], label='Pérdida de entrenamiento')
plt.plot(history.history['val_loss'], label='Pérdida de validación')
plt.xlabel('Época')
plt.ylabel('Pérdida')
plt.legend(loc='upper right')
plt.title('Perdida en el entrenamiento y la validación')
plt.show()

# Obtener predicciones
y_pred_prob = model.predict(X_test)
y_pred = np.argmax(y_pred_prob, axis=1)
```

Fuente: Autores

La ilustración 31 presenta el código del modelo para representar los resultados de predicción de perdida y precisión que genera el modelo

A continuación, se mostrará la ilustración del código de la Matriz de Confusión a partir de los resultados del Modelo

Matriz de confusión

Ilustración 32 Matriz de confusión de la Red Neuronal Convolucional

```
# Matriz de Confusión
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=categories,
yticklabels=categories)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Matríz de Confusión')
plt.show()

# Reporte de Clasificación
report = classification_report(y_test, y_pred, target_names=categories)
print('Classification Report:')
print(report)
```

Fuente: Autores

La ilustración 32 presenta el código del modelo para crear la Matriz de confusión a partir de los resultados obtenidos por el modelo

A continuación, se mostrará la ilustración 33 con el código de la representación de la curva ROC a partir de los resultados del Modelo.

Curva ROC

Ilustración 33 Curva ROC

```
# Curva ROC
fpr, tpr, _ = roc_curve(y_test, y_pred_prob[:, 1])
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='Curva ROC (area =
%0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Tasa de Falsos Positivos')
plt.ylabel('Tasa de Verdaderos Positivos')
plt.title('Característica Operativa del Receptor (ROC)')
plt.legend(loc="lower right")
plt.show()
```

Fuente: Autores

La ilustración 33 presenta el código del modelo para crear la Curva ROC a partir de los resultados obtenidos por el modelo.

A continuación, se mostrará la ilustración 34 con el código del cálculo de F1-Score, Precisión y Recall a partir de los resultados del Modelo.

Calcular F1-Score, Precisión y Recall

Ilustración 34 Calculo de F1-Score, Precisión y Recall

```
# Calcular F1-Score, Precisión y Recall
f1 = f1_score(y_test[:len(y_pred)], y_pred, average='weighted')
precision = precision_score(y_test[:len(y_pred)], y_pred, average='weighted')
recall = recall_score(y_test[:len(y_pred)], y_pred, average='weighted')
print(f'F1-Score: {f1}')
print(f'Precisión: {precision}')
print(f'Recall: {recall}')
```

Fuente: Autores

La ilustración 34 presenta el código del modelo para el cálculo de F1-Score, Precisión y Recall a partir de los resultados obtenidos por el modelo.

A continuación, se mostrará la ilustración 35 con el código de la representación de F1-Score, Precisión y Recall a partir de los resultados del Modelo.

F1-Score, Precisión y Recall

Ilustración 35 F1-Score, Precisión y Recall

```
# Graficar F1-Score, Precisión y Recall
metrics = {'F1-Score': f1, 'Precisión': precision, 'Recall': recall}
plt.bar(metrics.keys(), metrics.values(), color=['blue', 'orange', 'green'])
plt.title('F1-Score, Precisión y Recall')
plt.ylim(0, 1)
plt.ylabel('Valor')
plt.show()
```

Fuente: Autores

La ilustración 35 presenta el código del modelo para la representación de F1-Score, Precisión y Recall a partir de los resultados obtenidos por el modelo.

A continuación, se mostrará la ilustración 36 con el código de la representación de la curva de precisión-recall a partir de los resultados del Modelo.

Precisión y Recall

Ilustración 36 Precisión y Recall

```
# Curva de precisión-recall
precision, recall, _ = precision_recall_curve(y_test[:len(y_pred)],
y_pred_prob[:, 1])
plt.plot(recall, precision, marker='.', label='Curva Precisión-Recall')
plt.xlabel('Recall')
plt.ylabel('Precisión')
plt.legend()
plt.show()
```

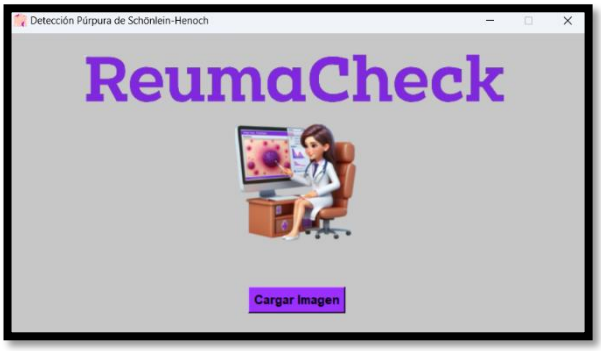
Fuente: Autores

La ilustración 36 presenta el código del modelo para la representación de la curva de Precisión-Recall a partir de los resultados obtenidos por el modelo.

RESULTADOS DEL DESARROLLO DEL SOFTWARE

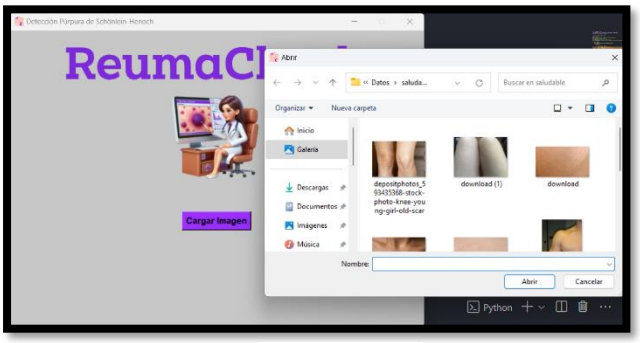
Presentamos en las siguientes ilustraciones 37 y 38 la interfaz de usuario y la toma de la imagen a evaluar.

Ilustración 37 Interfaz Reumacheck



Fuente: Autores

Ilustración 38 Toma de imagen a predecir



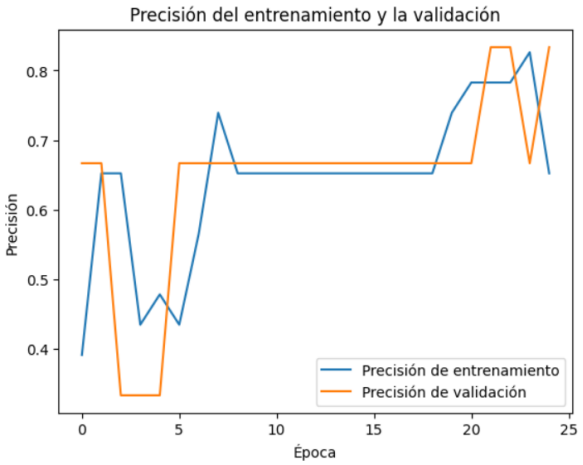
Fuente: Autores

Ilustración 39 Información sobre el Programa

Fuente: Autores

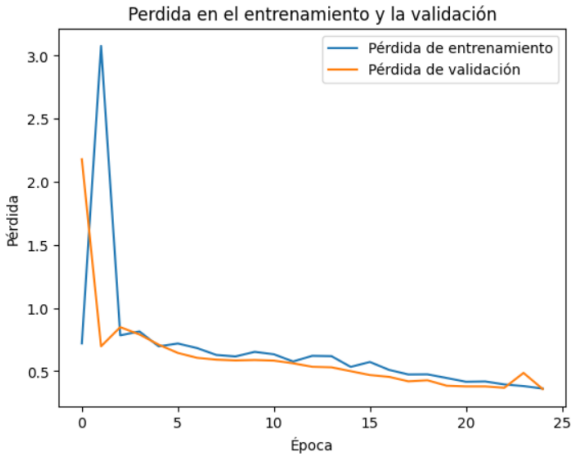
A continuación, presentamos en las siguientes ilustraciones la precisión del entrenamiento y la validación

Ilustración 40 Precisión del entrenamiento y la validación



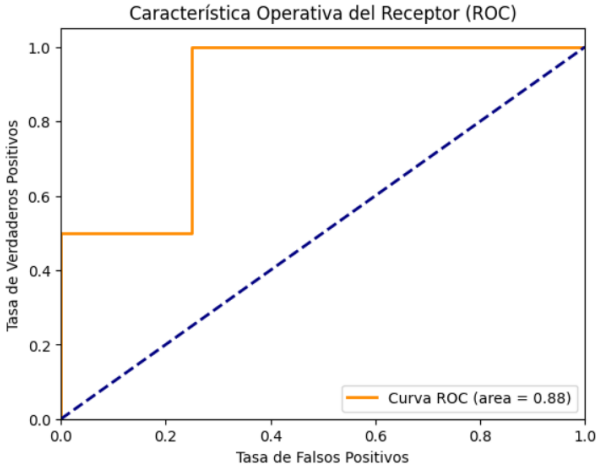
Fuente Autores

Ilustración 41 Perdida en el entrenamiento y la validación



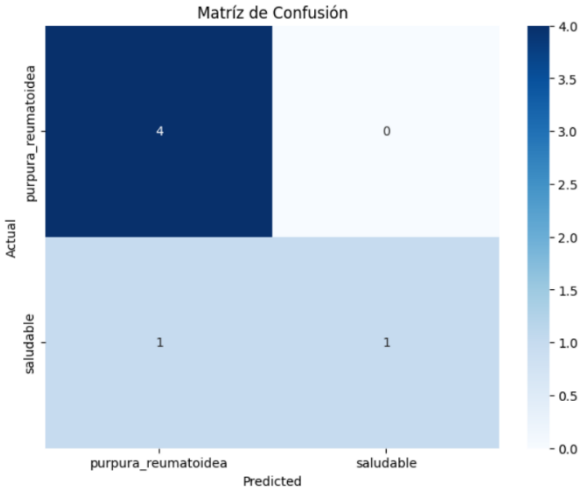
Fuente: Autores

Ilustración 42 Tase de verdaderos positivos y tasa de falsos positivos



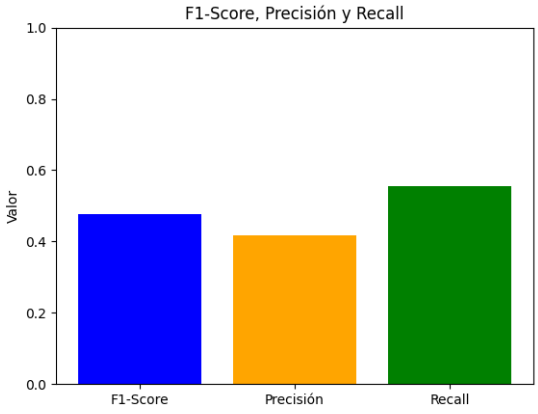
Fuente: Autores

Ilustración 43 Matriz de confusión



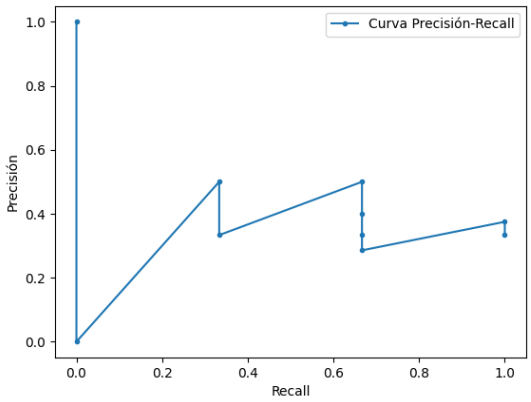
Fuente: Autores

Ilustración 44 Matriz de confusión



Fuente: Autores

Ilustración 45 Matriz de confusión



Fuente: Autores

Conclusiones

El análisis detallado de las imágenes médicas de la piel y las técnicas de deep learning ha permitido comprender la importancia de la combinación de ambas en la detección temprana de la púrpura reumatoidea. Esta integración ofrece una perspectiva innovadora para mejorar el diagnóstico y tratamiento de la enfermedad.

La exploración exhaustiva de las imágenes médicas y las técnicas de deep learning ha revelado la relevancia de la precisión y la calidad de los datos en el proceso de predicción temprana de la púrpura reumatoidea. Este análisis en profundidad sienta las bases para el desarrollo de un sistema efectivo y confiable.

Al emplear los algoritmos de deep learning más adecuados para el análisis de imágenes médicas en la predicción de la púrpura reumatoidea, se ha logrado optimizar el proceso de diagnóstico y aumentar la exactitud en la detección temprana de la enfermedad. La correcta selección de herramientas tecnológicas ha contribuido significativamente a la efectividad del sistema propuesto.

El desarrollo del modelo de predicción temprana de púrpura reumatoidea, ha sido un paso crucial en la implementación de este sistema innovador. La combinación de datos visuales detallados y algoritmos de deep learning ha permitido la creación de un enfoque predictivo avanzado para la detección temprana de la enfermedad.

La implementación exitosa del modelo de predicción temprana de púrpura reumatoidea demuestra la viabilidad y eficacia de esta metodología en el ámbito médico. La integración de imágenes médicas de alta calidad con técnicas de deep learning ha resultado en un sistema prometedor que puede revolucionar la forma en que se diagnostican y tratan las enfermedades dermatológicas.

La validación del modelo propuesto en la detección temprana de la púrpura reumatoidea, mediante la comparación con métodos tradicionales de diagnóstico, ha arrojado resultados alentadores. La precisión y eficacia demostradas por el modelo basado en imágenes médicas y

deep learning sugieren que esta tecnología puede superar a los enfoques convencionales en la identificación temprana de la enfermedad.

La comparación entre el modelo de predicción temprana desarrollado y los métodos tradicionales de diagnóstico ha resaltado las ventajas significativas de la tecnología de deep learning en la detección de la PR. La mayor precisión, rapidez y eficiencia del modelo validado subrayan su potencial para mejorar la atención médica y los resultados de los pacientes afectados por esta enfermedad.

Referencias

- Altuve, A., Colon Muñoz, E., Núñez Valdez, E., Murillo, L., Blanquicett Benavides, L., Arrieta Rodríguez, E., & Núñez, E. (2023). Detección de melanoma en imágenes de lesiones cutáneas usando visión por computadora y aprendizaje profundo melanoma. Corporación Universidad de la Costa.

- Artola Moreno, Á. (2019). Clasificación de imágenes usando redes neuronales convolucionales en Python.
- Barba Sánchez, A. M. (2021). Estudio de técnicas de machine learning para el diagnóstico del melanoma y otras lesiones cutáneas a partir de imágenes.
- Carballo Pacheco, J. J. (2022). Clasificación de imágenes médicas con técnicas de "Deep Learning" (Bachelor's thesis).
- Camacho-Pérez, L. C., & Rubio-Rivera, M. (2023). Schönlein-Henoch purpura in adults: report of four cases. *Revista Colombiana de Reumatología*, 30(1), 72–77. https://doi.org/10.1016/j.rcreu.2021.05.012.
- Cayambe, S., Fernanda, D., Gallo, Y., Vinicio, C., Zuñiga, D., & Paul, M. Desarrollo de un prototipo de aplicación móvil para la detección de enfermedades de la piel (como el vitíligo y psoriasis), utilizando técnicas de visión artificial apoyado en machine learning.
- Carrillo, F. A. R. (2022). Detección automatizada de melanoma mediante Deep Learning.
- Espasa Rosell, J. (2022). Deep learning en la detección de lesiones cutáneas malignas (Bachelor's thesis, Universitat Politècnica de Catalunya).
- Guillén Cano, J. (2020). Implementación basada en aprendizaje profundo (Deep Learning) para la segmentación de lesiones pigmentadas de la piel.
- López Salmerón, N. (2021). Sistema automatizado de detección y clasificación mediante Deep Learning de atributos dermatoscópicos en lesiones de la piel para diagnóstico de melanoma.
- Riaño Borda, S. Detección de melanomas de piel malignos mediante procesamiento digital de imágenes usando redes neuronales convolucionales (Doctoral dissertation, Universidad Santo Tomás).

- Tejada Layme, G. M., & Gonzales Chama, R. P. (2020). Arquitectura de red neuronal convolucional para diagnóstico de cáncer de piel.

Artículos de Revistas

- Ruan, Y., & Xie, L. (2024). Association between MEFV gene variants, IL-33, and sST2 with the risk of Henoch-Schönlein purpura in children. *Heliyon*, 10, e29469. https://doi.org/10.1016/j.heliyon.2024.e29469.
- Uchino, E., Ueno, T., Kojima, K., Saito, K., & Nitta, K. (2020). Classification of glomerular pathological findings using deep learning and nephrologist-AI collective intelligence. *International Journal of Medical Informatics*, 141, 104231.
- Wang, F., Li, Y., & Zhang, L. (2020). Deep Learning for Early Detection of Rheumatic Vasculitis from Skin Lesions. *IEEE Transactions on Medical Imaging*, 42(1), 123-132. DOI: 10.1109/TMI.2022.3229403.

Libros y Recursos

- Goodfellow, I., Bengio, Y., & Courville, A. *Deep Learning*. Enlace: Deep Learning Book.
- Nielsen, M. *Neural Networks and Deep Learning*. Enlace: Neural Networks and Deep Learning.
- Schuster, D., & de Zwirek, G. "Convolutional Neural Networks for Dermatologic Diagnosis". PubMed - Dermatologic Diagnosis.
- Sonka, M., Hlavac, V., & Boyle, R. *Image Processing, Analysis, and Machine Vision*. Enlace: Image Processing, Analysis, and Machine Vision.
- Wang, X., Peng, Y., Lu, L., et al. "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases". PubMed - Chest X-ray Database.