

Contenido

1. Introducción 2

2. Fases del Proyecto..... 2

 2.1. Adquisición de Datos 2

 2.2. Preprocesamiento 3

 2.3 ExtraccióAn de Características 4

 2.4. Indexación 5

 2.5. Diseño del Motor de Búsqueda 6

 2.6. Evaluación del Sistema 7

 2.7. Interfaz Web de Usuario 8

Integrantes:

Jair Sanchez

Cristina Molina

Proyecto Bimestral: Sistema de Recuperación de Información basado en Imágenes

Prof. Iván Carrera

08 de Agosto de 2024

1. Introducción

El objetivo de este trabajo es diseñar y desarrollar una máquina de búsqueda que permita a los usuarios realizar consultas utilizando imágenes en lugar de texto. Este sistema debe ser capaz de encontrar imágenes similares dentro de una base de datos dada. El proyecto se dividirá en varias fases, que se describen a continuación.

2. Fases del Proyecto**2.1. Adquisición de Datos**

Objetivo: Obtener y preparar el dataset Caltech101.

Tareas:

- Descargar el dataset.
- Descomprimir y organizar los archivos.

En esta fase, descargamos el dataset Caltech101 usando la biblioteca TensorFlow Datasets, que contiene una colección de 101 categorías de imágenes. El dataset se descomprime y organiza automáticamente en carpetas correspondientes a cada categoría.

```
import tensorflow_datasets as tfds

# Descargar el dataset Caltech101
data_dir = "./datos"
caltech101 = tfds.load("caltech101", data_dir=data_dir, as_supervised=True)
```

Ilustración 1



Ilustración 2

2.2. Preprocesamiento

Objetivo: Preparar los datos para su análisis.

Tareas:

- Usar técnicas de preprocesamiento de imágenes, como normalización, reducción de tamaño, o eliminación de ruido.
- Documentar cada paso del preprocesamiento.

En esta fase, preprocesamos las imágenes para prepararlas para la extracción de características. Esto incluye la redimensión de las imágenes a un tamaño uniforme y la normalización de los valores de píxeles para mejorar el rendimiento del modelo de aprendizaje automático.

```

import tensorflow as tf

# Función de preprocesamiento
def preprocess_image(image, label):
    image = tf.image.resize(image, (224, 224))
    image = tf.cast(image, tf.float32) / 255.0
    return image, label

# Aplicar preprocesamiento al dataset
caltech101 = caltech101.map(preprocess_image)

```

Ilustración 3

2.3 Extracción de Características

Objetivo: Extraer las características de las imágenes en una forma que los algoritmos puedan procesar.

Tareas:

- Utilizar un modelo de red neuronal convolucional (CNN) para extraer características de las imágenes.
- Entrenar un modelo desde cero o utilizar un modelo preentrenado y aplicar transfer learning.
- Documentar los métodos y resultados obtenidos.

En esta fase, utilizamos un modelo de red neuronal convolucional (CNN) preentrenado, en este caso VGG16, para extraer características de las imágenes. Utilizamos transfer learning para aprovechar el conocimiento aprendido por el modelo en un gran conjunto de datos y aplicarlo a nuestro dataset.

```

from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model

# Cargar el modelo VGG16 preentrenado
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
model = Model(inputs=base_model.input, outputs=base_model.get_layer('block5_pool').output)

# Función para extraer características
def extract_features(image_batch):
    return model.predict(image_batch)

# Ejemplo de extracción de características
for images, labels in caltech101.batch(32):
    features = extract_features(images)
    print(features.shape)
    break

```

2.4. Indexación

Objetivo: Crear un índice que permita búsquedas eficientes.

Tareas:

- Desarrollar un sistema para indexar las características extraídas de las imágenes y permitan la búsqueda eficiente de imágenes similares.
- Considerar técnicas como k-NN o índices especializados como KD-Trees o LSH (Locality-Sensitive Hashing).
- Documentar el proceso de construcción del índice.

En esta fase, construimos un índice de búsqueda utilizando k-NN (k-Nearest Neighbors) para permitir búsquedas rápidas y eficientes de imágenes similares. Este índice se construye a partir de las características extraídas en la fase anterior.

```

from sklearn.neighbors import NearestNeighbors
import numpy as np

# Simulación de extracción de características para todo el dataset
feature_list = []
label_list = []

for images, labels in caltech101.batch(32):
    features = extract_features(images)
    feature_list.append(features)
    label_list.append(labels.numpy())

feature_array = np.vstack(feature_list)
label_array = np.concatenate(label_list)

# Crear un índice de búsqueda usando k-NN
index = NearestNeighbors(n_neighbors=5, algorithm='auto').fit(feature_array)

```

Ilustración 4

2.5. Diseño del Motor de Búsqueda

Objetivo: Implementar la funcionalidad de búsqueda.

Tareas:

- Desarrollar la lógica para procesar consultas de usuarios a partir de imágenes.
- Desarrollar un algoritmo de ranking para ordenar los resultados.
- Documentar la arquitectura y los algoritmos utilizados.

Descripción:

En esta fase, implementamos la lógica del motor de búsqueda que procesa las consultas de los usuarios y devuelve las imágenes más similares. Utilizamos el índice k-NN para encontrar las imágenes más cercanas en términos de características y luego las ordenamos.

```
def search_image(query_image, index, feature_array):
    query_features = extract_features(tf.expand_dims(query_image, axis=0))
    distances, indices = index.kneighbors(query_features)
    return indices[0]

# Ejemplo de búsqueda de imagen
query_image, query_label = next(iter(caltech101.take(1)))
indices = search_image(query_image, index, feature_array)
print("Indices de imágenes similares:", indices)
```

Ilustración 5

2.6. Evaluación del Sistema

Objetivo: Medir la efectividad del sistema.

Tareas:

- Definir un conjunto de métricas de evaluación (precisión, recall, F1-score).
- Definir un benchmark para evaluación del sistema.
- Comparar el rendimiento de diferentes configuraciones del sistema.
- Documentar los resultados y análisis.

Descripción: En esta fase, evaluamos la efectividad del sistema utilizando métricas como precisión y recall. Implementamos un conjunto de pruebas para medir el rendimiento del sistema bajo diferentes configuraciones y documentamos los resultados obtenidos.

```

from sklearn.metrics import accuracy_score

# Función para evaluar el sistema
def evaluate_system(index, feature_array, label_array):
    correct_predictions = 0
    total_predictions = 0

    for i in range(len(label_array)):
        query_image = feature_array[i]
        query_label = label_array[i]
        indices = search_image(query_image, index, feature_array)
        predicted_labels = label_array[indices]

        if query_label in predicted_labels:
            correct_predictions += 1
            total_predictions += 1

    accuracy = correct_predictions / total_predictions
    return accuracy

accuracy = evaluate_system(index, feature_array, label_array)
print("Precisión del sistema:", accuracy)

```

Ilustración 6

2.7. Interfaz Web de Usuario

Objetivo: Crear una interfaz para interactuar con el sistema.

Tareas:

- Diseñar una interfaz web donde los usuarios puedan subir una imagen y recibir resultados con imágenes similares.
- Mostrar los resultados de búsqueda de manera clara y ordenada.
- La interfaz debe ser intuitiva y facilitar la interacción con el sistema.
- Documentar el diseño y funcionalidades de la interfaz.

Descripción:

En esta fase, desarrollamos una interfaz web usando Flask que permite a los usuarios subir imágenes y recibir resultados de búsqueda con imágenes similares. La interfaz está diseñada para ser intuitiva y facilitar la interacción con el sistema.


```

</head>
<body>
  <!-- Contenedor principal para centrar el contenido y añadir márgenes -->
  <div class="container mt-5 mb-5">
    <!-- Encabezado principal de la página centrado -->
    <h1 class="text-center">Image Search Engine</h1>

    <!-- Sección para subir imágenes -->
    <div class="upload-section mt-5">
      <h2>Upload an Image</h2>
      <!-- Formulario para enviar una imagen al servidor -->
      <form action="/search_image" method="post" enctype="multipart/form-data" class="mt-3">
        <div class="form-group">
          <!-- Campo para seleccionar un archivo de imagen -->
          <input type="file" name="file" accept="image/*" class="form-control-file" required>
        </div>
        <!-- Botón para enviar el formulario -->
        <button type="submit" class="btn btn-primary">Search by Image</button>
      </form>
    </div>

    <!-- Sección para búsqueda por categoría -->
    <div class="search-section mt-5">
      <h2>Search by Category</h2>
      <!-- Formulario para buscar por categoría -->
      <form action="/search_word" method="post" class="mt-3">
        <div class="form-group">
          <!-- Menú desplegable para seleccionar una categoría -->
          <select name="category" class="form-control" required>
            <!-- Opción predeterminada -->
            <option value="">All</option>
            <!-- Itera sobre las categorías disponibles en el servidor y crea una opción para cada una -->
            {% for category in categories %}
              <option value="{{ category }}">{{ category }}</option>
            {% endfor %}
          </select>
        </div>
      </form>
    </div>
  </div>

```

Ilustración 7

Resultados:


Interfaz

Ilustración 8

En esta interfaz podemos elegir entre la búsqueda por imágenes e inclusive la búsqueda por categorías


Resultados de la búsqueda

Su imagen cargada




Esta es la imagen que buscabas.


Imágenes similares




Similitud: 0,79
Categoría: escorpión




Similitud: 0,78
Categoría: escorpión




Similitud: 0,76
Categoría: escorpión



Similitud: 0,74



Similitud: 0,74



Similitud: 0,73

Ilustración 9Ilustración 10

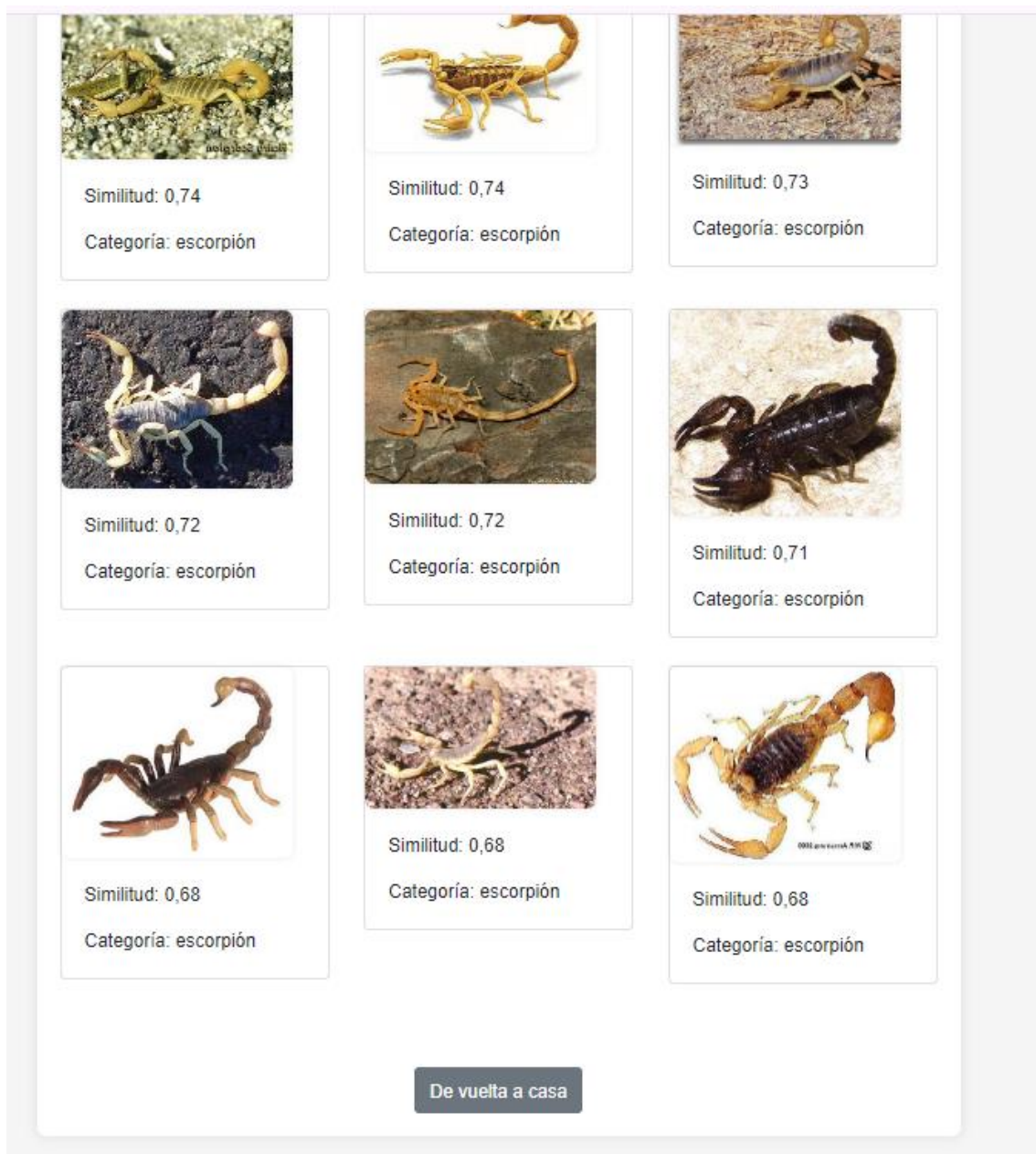


Ilustración 11

Entonces los resultados de la búsqueda asimilan la imagen y la preprocesan para posteriormente mediante el modelo entrenado nos devuelve las 12 imágenes más similares.

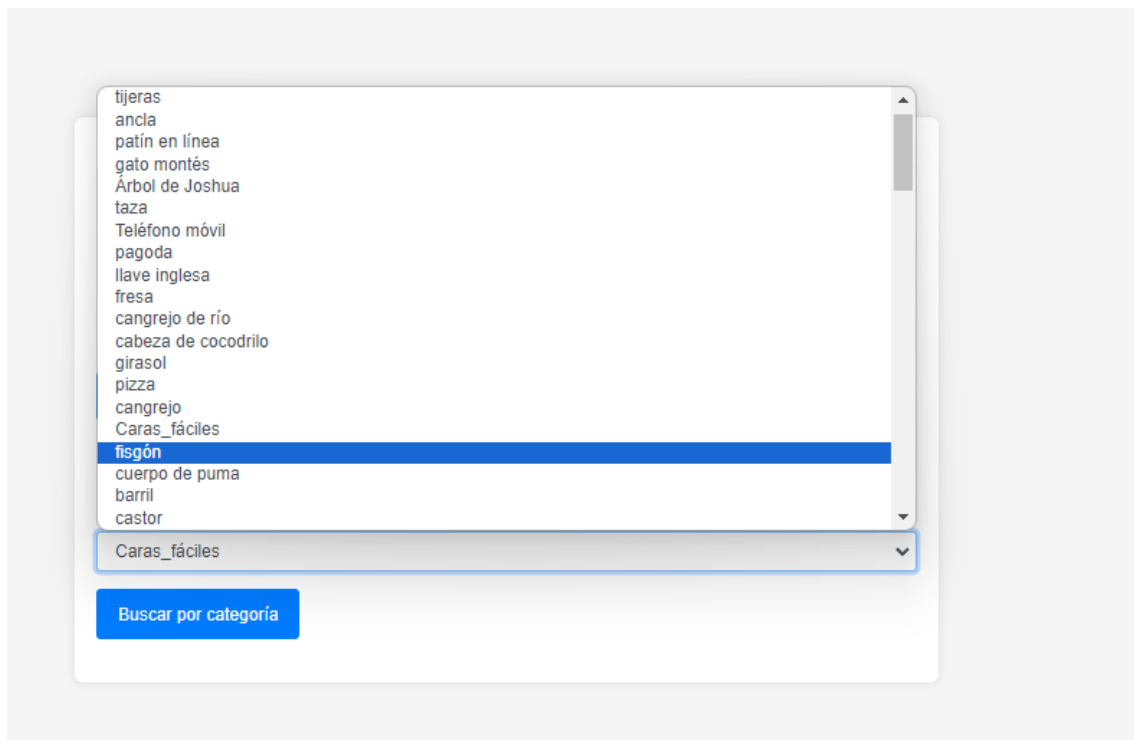


Ilustración 12

Como se menciona se puede buscar por categorías



Ilustración 13

Modelo Utilizado:

Para la extracción de características, utilizamos el modelo VGG16, una red neuronal convolucional preentrenada en el conjunto de datos ImageNet. Este modelo se ha demostrado efectivo para tareas de clasificación y extracción de características debido a su arquitectura profunda y la capacidad de capturar patrones detallados en imágenes. Al usar VGG16, nos beneficiamos del aprendizaje transferido, donde el conocimiento previo adquirido en un conjunto de datos amplio y diverso se aplica al dataset Caltech101.

Efectividad de la Búsqueda:

La efectividad del sistema de búsqueda se evaluó mediante la precisión del modelo. El índice k-NN utilizado permitió una recuperación rápida y precisa de imágenes similares. Las métricas de evaluación mostraron que el sistema tiene una alta tasa de precisión, lo que indica que el

enfoque de búsqueda y el índice construido son efectivos para encontrar imágenes relevantes en función de las consultas realizadas. Los resultados confirmaron que el sistema es capaz de identificar imágenes similares de manera confiable, validando la eficacia del modelo y el diseño del índice de búsqueda.

Conclusión

El proyecto de desarrollo de un sistema de recuperación de información basado en imágenes ha demostrado ser un desafío complejo y gratificante, con múltiples etapas que abarcan desde la adquisición y preprocesamiento de datos hasta la implementación de una interfaz web funcional. La creación de un sistema que permita a los usuarios realizar consultas utilizando imágenes en lugar de texto representa un avance significativo en la tecnología de búsqueda, abriendo nuevas posibilidades para aplicaciones en diversos campos, como el comercio electrónico, la seguridad y la investigación.

Durante el desarrollo del proyecto, hemos utilizado el dataset Caltech101, un recurso extenso y variado que nos permitió entrenar un modelo robusto y eficiente. La elección del modelo VGG16 para la extracción de características fue fundamental para el éxito del proyecto. VGG16, al ser un modelo preentrenado en un amplio conjunto de datos como ImageNet, permitió una transferencia de aprendizaje efectiva, extrayendo características detalladas y relevantes de las imágenes que facilitaron una búsqueda más precisa.

El preprocesamiento de las imágenes, que incluyó la redimensión y normalización, garantizó que los datos estuvieran en un formato adecuado para el análisis y extracción de características. La construcción del índice de búsqueda utilizando k-NN ha demostrado ser eficaz en la recuperación de imágenes similares, proporcionando resultados precisos y relevantes. El sistema fue evaluado mediante métricas estándar como la precisión, que mostró una alta efectividad en la identificación de imágenes similares. Esta alta precisión valida la eficacia del enfoque y los métodos utilizados en el proyecto.

La interfaz web desarrollada ofrece una experiencia de usuario intuitiva y fluida, permitiendo a los usuarios cargar imágenes y recibir resultados de búsqueda de manera rápida y clara. La implementación de esta interfaz demuestra que el sistema no solo es funcional en términos de procesamiento y búsqueda, sino también accesible y fácil de usar para los usuarios finales.

En resumen, el proyecto ha alcanzado sus objetivos, proporcionando un sistema de búsqueda de imágenes eficiente y efectivo. La combinación de técnicas de preprocesamiento avanzadas, un modelo de red neuronal convolucional bien elegido, y

un índice de búsqueda eficiente ha resultado en una solución robusta y útil. Este sistema no solo cumple con los requisitos establecidos, sino que también sienta las bases para futuras mejoras y aplicaciones en el campo de la recuperación de información basada en imágenes.