

DESD (Detection Errors Scanned Documents) API Rest

Menu:

- [DESD \(Detection Errors Scanned Documents\) API Rest](#)
 - [Description](#)
 - [Installation](#)
 - [Ejecución con Docker](#)
 - [Ejecución Local](#)
 - [Instalación de Dependencias](#)
 - [Inicio del Servidor Backend](#)
 - [Inicio del Servior Frontend](#)
 - [Usage](#)
 - [API Endpoints](#)
 - [Detalles de los Endpoints](#)
 - [/audit \(POST\)](#)
 - [/tilde/v1 \(POST\)](#)
 - [/rode/v1 \(POST\)](#)
 - [Models](#)
 - [TilDeV1 \(Tilted Detection Version 1\)](#)
 - [RoDeV1 \(Rotation Detection Version 1\)](#)

Description

DESD (Detection Errors Scanned Documents) es una *API Rest* que permite detectar errores en documentos escaneados, como inclinación y rotación. La API cuenta con diferentes modelos de detección de errores, los cuales pueden ser utilizados a través de endpoints específicos.

La aplicación cuenta con una interfaz web que permite interactuar con la API de forma visual. La interfaz permite subir una imagen de un documento escaneado y obtener la detección de errores en la misma.

Installation

Ejecución con Docker

Para ejecutar la aplicación utilizando Docker, necesitarás utilizar Docker Compose, una herramienta que permite definir y manejar aplicaciones multi-contenedor con Docker.

Sigue los siguientes pasos para iniciar la aplicación:

1. Levantar los servicios con Docker Compose

Ejecuta el siguiente comando para iniciar todos los servicios definidos en el archivo `docker-compose.yml`:

```
docker-compose up -d
```

La opción `-d` hace que los servicios se ejecuten en segundo plano.

2. Acceso a los servicios

Tras ejecutar el comando anterior, se expondrán dos puertos:

- **Puerto 5000:** Aquí se encuentra la API de la aplicación.
- **Puerto 80:** Aquí se encuentra la interfaz web de la aplicación.

Puedes acceder a estos servicios a través de un navegador web o cualquier cliente HTTP, utilizando `localhost` seguido del número de puerto correspondiente (por ejemplo, `http://localhost:5000` para la API).

Ejecución Local

Para ejecutar este proyecto localmente, necesitarás instalar algunas dependencias tanto para el backend como para el frontend.

Instalación de Dependencias

Primero, instala las dependencias del backend con el siguiente comando:

```
pip install -r backend/requirements.txt
```

Si también deseas ejecutar el frontend, instala sus dependencias con:

```
pip install -r frontend/requirements.txt
```

Inicio del Servidor Backend

Para iniciar el servidor backend, ejecuta el siguiente comando:

```
python backend/app.py
```

El servidor se iniciará en la dirección `http://localhost:5000/`.

Inicio del Servior Frontend

Para iniciar el frontend, ejecuta el siguiente comando:

```
streamlit run frontend/main.py --server.port=80 --server.address=0.0.0.0
```

Esto iniciará la interfaz web en el puerto 80, la cual hará peticiones a la API del backend en el puerto 5000. Para acceder a la interfaz, abre tu navegador y dirígete a `http://localhost`.

Usage

La aplicación cuenta con una interfaz web que permite interactuar con la API de forma visual. Para acceder a la interfaz, abre tu navegador y dirígete a `http://<ip>` (si estás ejecutando la aplicación localmente, utiliza `http://localhost`).


API Endpoints

La API consta de los siguientes endpoints:

Endpoint	Método	Descripción	Parámetros	Respuesta
<code>/audit</code>	POST	Utiliza todos los modelos para detectar errores en documentos escaneados.	<code>image</code> (multipart/form-data)	JSON con la detección de errores usando todos los modelos disponibles.
<code>/tilde/v1</code>	POST	Detección de inclinación en documentos escaneados.	<code>image</code> (multipart/form-data)	JSON con la detección de inclinación.
<code>/rode/v1</code>	POST	Detección de rotación en documentos escaneados.	<code>image</code> (multipart/form-data)	JSON con la detección de rotación.

Detalles de los Endpoints

`/audit` (POST)

 warning: **WARNING:** Actualmente no se puede seleccionar la versión del modelo. Por defecto, se utiliza la versión 1 de cada modelo.

Este endpoint acepta una imagen de un documento escaneado y devuelve la detección de diferentes errores usando todos los modelos disponibles en la API.

Parámetros:

- `image`: Un archivo de imagen en formato multipart/form-data.

Respuesta:

Devuelve un objeto JSON con los errores detectados en la imagen. Cada error contiene los siguientes campos:

- name: El nombre del error detectado.
- confidence: La confianza de la predicción.

Ejemplo de petición:

```
curl -X POST -F "image=@image.png" http://localhost:5000/audit
```

```
import requests

url = "http://localhost:5000/audit"
files = {"image": open("image.png", "rb")}
response = requests.post(url, files=files)

print(response.json())
```

Ejemplo de respuesta:

```
{
  "rode": {
    "confidence": 0.98,
    "name": "rotated"
  },
  "tilde": {
    "confidence": 0.57,
    "name": "no tilted"
  }
}
```

/tilde/v1 (POST)

Este endpoint acepta una imagen de un documento escaneado y devuelve la detección de inclinación.

Parámetros:

- image: Un archivo de imagen en formato multipart/form-data.

Respuesta:

Devuelve un objeto JSON con un array data que contiene objetos con los siguientes campos:

- name: El nombre de la clase detectada.
- confidence: La confianza de la predicción.

Ejemplo de petición:

```
curl -X POST -F "image=@image.png" http://localhost:5000/tilde/v1
```

```
import requests

url = "http://localhost:5000/tilde/v1"
files = {"image": open("image.png", "rb")}
response = requests.post(url, files=files)

print(response.json())
```

Ejemplo de respuesta:

```
{
  "data": [
    {
      "confidence": 0.57,
      "name": "no tilted"
    },
    {
      "confidence": 0.43,
      "name": "tilted"
    }
  ]
}
```

/rode/v1 (POST)

Este endpoint acepta una imagen de un documento escaneado y devuelve la detección de rotación.

Parámetros:

- `image`: Un archivo de imagen en formato multipart/form-data.

Respuesta:

Devuelve un objeto JSON con un array `data` que contiene objetos con los siguientes campos:

- `name`: El nombre de la clase detectada.
- `confidence`: La confianza de la predicción.

Ejemplo de petición:

```
curl -X POST -F "image=@image.jpg" http://localhost:5000/rode/v1
```

```
import requests

url = "http://localhost:5000/rode/v1"
files = {"image": open("image.jpg", "rb")}
response = requests.post(url, files=files)

print(response.json())
```

Ejemplo de respuesta:

```
{
  "data": [
    {
      "confidence": 0.57,
      "name": "no_rotated"
    },
    {
      "confidence": 0.43,
      "name": "rotated"
    }
  ]
}
```

Models

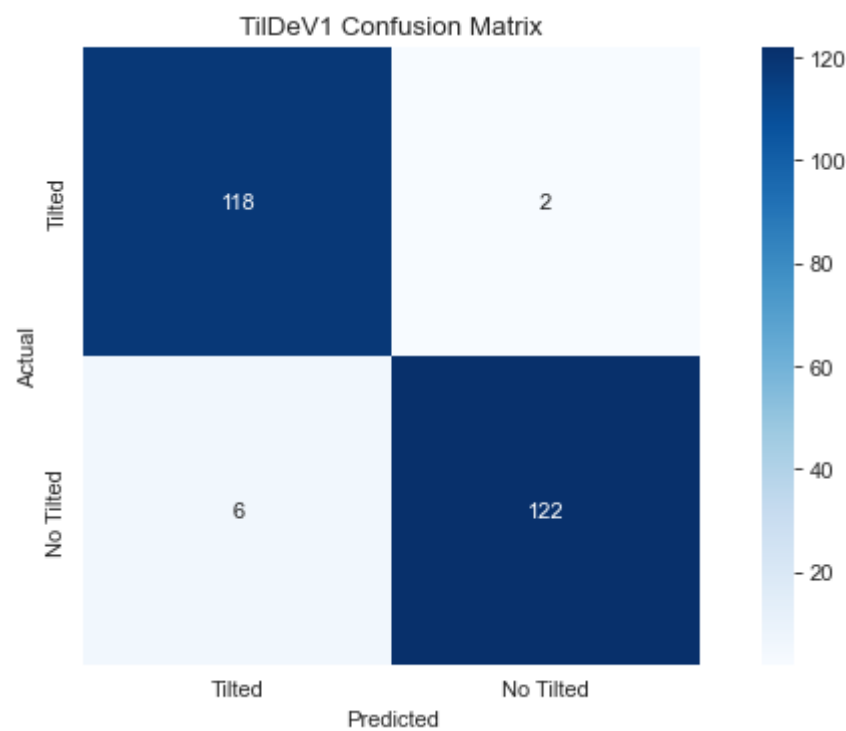
table of all models

Model	Description	Accuracy	Precision
TilDeV1	Modelo de detección de inclinación en documentos escaneados.	0.97	0.98
RoDeV1	Modelo de detección de rotación en documentos escaneados.	0.99	0.99

TilDeV1 (Tilted Detection Version 1)

Modelo de detección de inclinación de documentos escaneados. Entrenado mediante fine-tuning de un modelo pre-entrenado de detección de objetos YOLOv8n y con un dataset de 319 imágenes de documentos escaneados con y sin inclinación.

Accuracy: 0.97
Precision: 0.98



RoDeV1 (Rotation Detection Version 1)

Modelo de detección de rotación de documentos escaneados. Entrenado mediante fine-tuning de un modelo pre-entrenado de detección de objetos YOLOv8m y con un dataset de 288 imágenes de documentos escaneados con y sin rotación.

Accuracy: 0.99
Precision: 0.99

