LEGO_Bricks_Identification_Project_Technical_Report

January 19, 2025

1 LEGO Bricks Identification Project: A Journey of Learning and Adaptation

1.1 1. Introduction and Motivation

Este proyecto surge como una oportunidad para aplicar y demostrar mis habilidades técnicas en el contexto de un cambio profesional hacia la ciencia de datos e inteligencia artificial. Durante esta transición, decidí combinar mi pasión por los LEGO con mi interés por la visión por computadora, dando vida a una idea que conecta creatividad y tecnología.

1.1.1 1.1 Propósito y Objetivos

El propósito principal del proyecto es desarrollar una solución práctica y escalable que demuestre mis habilidades en ciencia de datos y visión por computadora. Los objetivos específicos incluyen:

1. Diseñar un pipeline eficiente para la detección y clasificación de piezas de LEGO.
2. Documentar el proceso de desarrollo, desde la creación del dataset hasta la implementación del modelo.
3. Generar resultados replicables y visualizaciones claras que destaquen el impacto de mi enfoque técnico.

1.2 2. Creación del Dataset

La base de este proyecto es un dataset bien estructurado y diverso. El proceso de creación incluyó la recolección, anotación y preprocesamiento de datos.

1.2.1 2.1 Recolección de Datos

Para capturar imágenes representativas de piezas LEGO, utilicé diferentes configuraciones de iluminación y disposición, generando un dataset inicial de más de 2000 imágenes.

1.2.2 2.2 Anotación de los Datos

Se utilizó LabelMe para anotar las imágenes con bounding boxes y puntos clave, generando archivos JSON que posteriormente se convirtieron al formato YOLO.

1.2.3 2.3 Preprocesamiento y Normalización

Código para Preprocesamiento y Descarga de Datos:

```
from scripts.pipeline import download_dataset_from_kaggle, preprocess_images

# Descargar el dataset desde Kaggle
kaggle_dataset = "migueldilalla/spiled-lego-bricks"
output_dir = "datasets"
download_dataset_from_kaggle(kaggle_dataset, output_dir)

# Preprocesar imágenes
preprocess_images(f"{output_dir}/raw", f"{output_dir}/processed")
```

1.3 3. Selección e Implementación del Modelo

1.3.1 3.1 Configuración del Pipeline de Entrenamiento

El pipeline implementa un enfoque modular para la preparación, entrenamiento y validación del modelo YOLOv8n.

Entrenamiento del Modelo:

```
from scripts.pipeline import train_yolo_pipeline

# Configurar y entrenar YOLO
train_yolo_pipeline(
    dataset_path="datasets",
    annotations_format="YOLO",
    epochs=50,
    img_size=256
)
```

1.3.2 3.2 Conversión de Anotaciones

Código para Convertir Anotaciones de LabelMe a YOLO:

```
from scripts.pipeline import labelme_to_yolo

# Convertir anotaciones al formato YOLO
labelme_to_yolo(
    input_folder="datasets/processed",
    output_folder="datasets/annotations"
)
```

1.4 4. Validación y Resultados

1.4.1 4.1 Pruebas en Imágenes Reales

Código para Probar el Modelo en Imágenes Reales:

```
from scripts.pipeline import test_model_on_real_images
```

```
# Evaluar modelo YOLO entrenado
model_path = "YOLO_Lego_Detection/best.pt"
test_images_dir = "test_images"
output_dir = "results"

test_model_on_real_images(model_path, test_images_dir, output_dir)
```

1.4.2 4.2 Visualización de Resultados

Visualización de Imágenes Anotadas:

```
from scripts.pipeline import visualize_results

visualize_results("datasets")
```

1.5 5. Reflexión y Futuro

1.5.1 5.1 Mejoras Futuras

1. **Optimización del Pipeline:** Explorar estrategias de fine-tuning para mejorar el rendimiento del modelo.
2. **Ampliación del Dataset:** Incorporar nuevos tipos de piezas y condiciones de captura.
3. **Automatización Completa:** Integrar herramientas avanzadas para anotaciones automáticas y expansión de datos.

Con este enfoque iterativo y modular, el proyecto está diseñado para evolucionar continuamente y adaptarse a nuevas aplicaciones y desafíos.