Refurbishing_Project_doc

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 La Inspiración del Proyecto

LEGO ha sido una de mis pasiones desde la infancia. Me fascinaba cómo piezas aparentemente desordenadas podían transformarse en estructuras organizadas y funcionales. Este proyecto busca replicar esa habilidad humana, enseñando a una máquina a identificar, clasificar y organizar piezas individuales en imágenes desordenadas. Más allá de ser un simple experimento técnico, esta iniciativa refleja mi interés por explorar cómo la inteligencia artificial puede emular procesos humanos de reconocimiento visual.

1.1.2 1.2 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.

Este proyecto también busca ser una base para aplicaciones futuras, como sistemas de clasificación robótica o asistentes visuales en tiempo real, mostrando la capacidad de escalar esta solución inicial hacia un impacto más amplio.

1.2 2. Creación del Dataset

El éxito de este proyecto dependió en gran medida de un dataset bien estructurado y diverso que capturara la naturaleza de las piezas de LEGO. El proceso de creación del dataset fue dividido en dos fases principales: una inicial centrada en los ladrillos básicos y otra posterior dedicada a los studs.

1.2.1 2.1 Recopilación de Datos

Para la primera fase, seleccioné ladrillos básicos de mi colección, excluyendo tiles y piezas transparentes, y los clasifiqué por dimensiones. En un espacio bien iluminado, dispuse los ladrillos con suficiente separación para que fueran fácilmente distinguibles. Usando la cámara de mi móvil configurada en modo de baja calidad y captura automática, generé más de 2000 imágenes mientras caminaba lentamente por la habitación, enfocándome en ángulos típicos desde los que un usuario de LEGO observaría las piezas.

En la segunda fase, monté un set fotográfico casero utilizando una plataforma giratoria hecha con piezas de LEGO. Coloqué nuevos conjuntos de ladrillos sobre esta plataforma y capturé imágenes desde un trípode con el móvil, utilizando captura automatizada nuevamente.

1.2.2 2.2 Anotación de los Datos

En la primera fase, utilicé LabelMe para anotar las piezas con bounding boxes en las más de 2000 imágenes, eliminando aquellas que estaban borrosas. Posteriormente, desarrollé scripts para convertir estas anotaciones al formato compatible con YOLO y organizar las carpetas de acuerdo con los requerimientos del modelo.

En la segunda fase, utilicé un modelo entrenado inicialmente para detectar ladrillos y generé anotaciones de puntos (point annotations) dentro de las bounding boxes recortadas. Un script adicional transformó estos puntos en nuevas bounding boxes dinámicas, adaptándose a las dimensiones y posiciones de los studs detectados.

1.2.3 2.3 Limpieza y Preparación del Dataset

Las imágenes borrosas o redundantes fueron eliminadas. Para mitigar la falta de calidad en algunas capturas iniciales, implementé técnicas de aumento de datos (data augmentation) directamente en el alimentador del modelo durante las sesiones de entrenamiento. Esto incluyó rotaciones, variaciones de iluminación y espejado para incrementar la diversidad de las muestras.

1.2.4 2.4 Insights para Iteraciones Futuras

- **Automatización en la anotación inicial**: Considera herramientas como Supervisely o plataformas que integren anotación semiautomática para acelerar procesos iniciales de bounding boxes.
- **Mejorar la calidad inicial**: Aunque la baja calidad fue útil para la primera fase, para futuros proyectos se recomienda una resolución estándar para maximizar el detalle en las imágenes.
- **Balance de clases**: Implementar análisis de distribución más temprano en el proceso puede garantizar un balance adecuado en el dataset y evitar clases subrepresentadas.
- **Data augmentation externo**: Herramientas como Albumentations o Fastai podrían ofrecer técnicas más avanzadas de aumento, como distorsiones geométricas y filtros, fuera del alimentador del modelo.
- **Pruebas preliminares del modelo**: Entrenar modelos pequeños desde el inicio puede revelar rápidamente problemas en el dataset, como errores de anotación o distribuciones desbalanceadas, y ahorrar tiempo en iteraciones posteriores.

1.3 3. Selección del Modelo

1.3.1 3.1 Elección del Modelo

El desafío central del proyecto era diseñar un modelo que pudiera identificar piezas individuales de LEGO en imágenes desordenadas y con condiciones variables de iluminación y fondo. Para abordar este problema, era crucial elegir un modelo que equilibrara precisión, rapidez y facilidad de implementación. Tras investigar varias opciones, seleccioné YOLO (You Only Look Once), específicamente la versión YOLOv8n, por las siguientes razones:

- **Simplicidad y Configuración Inicial**: YOLO ofrecía una configuración amigable para mi hardware con capacidades de GPU limitadas. Su implementación directa me permitió centrarme más en la preparación del dataset y en los ajustes iterativos.
- **Velocidad en Detección en Tiempo Real**: YOLO es conocido por su capacidad de realizar detecciones en una sola pasada de red neuronal, optimizando tanto el tiempo como el uso de recursos computacionales. Esto resultó ideal para entrenar y validar rápidamente el modelo con un dataset en constante refinamiento.
- **Robustez en Ambientes Complejos**: Su arquitectura permite manejar imágenes con múltiples objetos, incluso en escenarios donde las piezas están parcialmente ocluidas, lo que alineaba perfectamente con las condiciones del proyecto.

1.3.2 3.2 Estrategia de Entrenamiento y Adaptación

El proceso de entrenamiento incluyó los siguientes pasos:

- **Creación y Preparación del Dataset**: Se definió un pipeline para convertir anotaciones en formato LabelMe a un formato compatible con YOLO. Esto incluyó la limpieza de datos, aumento (augmentation) durante el entrenamiento y un split 80-20 para entrenamiento y validación.
- **Hiperparámetros**: Ajusté los siguientes parámetros clave para optimizar el rendimiento:
  - Tamaño de imágenes: Utilicé imágenes redimensionadas para maximizar la compatibilidad con el modelo.
  - Épocas: Comencé con 50 épocas para pruebas rápidas, aumentando gradualmente según los resultados.
  - Tasa de aprendizaje inicial: 0.001, con un decaimiento cíclico (cos_lr).
- **Uso de Scripts Personalizados**: Implementé scripts para:
  - Recortar piezas detectadas en imágenes, simplificando tareas posteriores de clasificación.
  - Transformar anotaciones de puntos clave en bounding boxes adaptativas según los studs detectados y dimensiones de entrada.
- **Iteraciones y Validación**: Cada iteración del modelo fue validada visualizando predicciones sobre nuevas imágenes del dataset, ajustando hiperparámetros según las métricas de precisión y recall.

1.3.3 3.3 Ejemplos y Visualizaciones

Para esta sección se incluirán:

- Imágenes con Bounding Boxes de ladrillos detectados: Visualizaciones de ejemplos representativos de detecciones exitosas, resaltando piezas con múltiples colores y tamaños.
- Comparación antes y después del modelo: Mostrar ejemplos de imágenes con y sin anotaciones generadas automáticamente por el modelo.

- Visualización de transformación de keypoints a bounding boxes: Ejemplos ilustrativos de cómo las anotaciones de studs se convirtieron en cajas 2D.

1.4 4. Próximos Pasos

El desarrollo iterativo del modelo ha proporcionado un camino claro para expandir el alcance del proyecto. A medida que se continúe mejorando, estas iteraciones ayudarán a explorar nuevas áreas de aplicación y optimización en el reconocimiento visual de LEGO.

1.5 Introduction

This notebook narrates the development of a project that started as a personal challenge and became a demonstration of creative problem-solving and a deep dive into computer vision. The project explores how computer vision techniques can be used to identify and analyze LEGO bricks.

1.5.1 Motivation

As a lifelong learner and a problem-solver, I found this project to be a perfect opportunity to apply my curiosity and drive to a practical challenge. It also reflects a significant moment in my life: transitioning from my work in the hospitality industry to pursuing a career in technology, where my skills and passions can have a greater impact.

1.6 Problem Statement

LEGO bricks come in many shapes and sizes, often with subtle differences. The challenge was to create a system capable of identifying individual bricks and determining their dimensions using computer vision techniques. While the initial scope was limited to 26 distinct brick types, the problem presented several complexities:

- Variability in brick dimensions and shapes.
- The need for accurate annotations to train detection models.
- Limited data availability, requiring adaptive solutions.

1.7 Methodology

1.7.1 1. Initial Approach: Direct Brick Classification

The project began with the idea of classifying LEGO bricks directly using object detection models like YOLO. However, early experimentation revealed significant challenges:

- **Dataset limitations**: The dataset was small, unbalanced, and contained subtle differences between classes.
- **Model performance**: Direct classification yielded low accuracy due to the aforementioned challenges.

1.7.2 2. Adaptation: Focusing on Stud Detection

To overcome these hurdles, the approach shifted towards detecting studs (the small bumps on LEGO bricks) and using their positions to infer brick dimensions. This adjustment allowed for:

- Simplified classification logic.
- Better utilization of the dataset by focusing on a common feature across all bricks.

The process involved: - Annotating images with stud positions using LabelMe. - Training a YOLOv8 model to detect studs. - Developing algorithms to compute brick dimensions from the detected stud positions.

1.7.3 3. Experimentation and Iteration

The development process was highly iterative, involving: - Multiple rounds of dataset refinement and augmentation. - Experimenting with model hyperparameters to improve detection accuracy. - Validating the dimension-calculation algorithms with test images.

1.8 Results

The project achieved: - **Accurate stud detection**: The trained YOLO model demonstrated high precision and recall in detecting studs. - **Dimension calculation**: Algorithms successfully inferred the dimensions of various bricks using detected stud positions.

1.8.1 Visualizations

- Examples of stud detection with bounding boxes.
- Graphs showing model performance metrics (precision, recall, F1 score).

1.9 Reflection and Learnings

This project exemplified the power of curiosity and adaptability in tackling challenges. Key takeaways include:

1. The importance of flexibility: Pivoting to a simpler approach yielded better results and deeper insights.
2. Learning through experimentation: Iterative testing refined the solution and improved my understanding of the problem.
3. A new direction: The project reaffirmed my passion for learning and solving complex problems, motivating me to transition into a technology-focused career.

1.10 Future Directions

This project is a starting point. Potential next steps include: - Expanding the dataset to include more brick types. - Refining the detection and dimension-calculation algorithms for production-level accuracy. - Scaling the solution for real-world applications, such as automated sorting or inventory management.

1.11 Code and Implementation

1.11.1 1. Data Preparation

[ ]:
```python
# Example code for data preprocessing
import os
import cv2
```

```
# Load and preprocess images
image_dir = 'data/raw'
processed_dir = 'data/processed'
os.makedirs(processed_dir, exist_ok=True)

for image_file in os.listdir(image_dir):
    image_path = os.path.join(image_dir, image_file)
    image = cv2.imread(image_path)
    # Resize, normalize, and save
    processed_image = cv2.resize(image, (640, 640))
    cv2.imwrite(os.path.join(processed_dir, image_file), processed_image)
```

1.11.2 2. Model Training

[ ]:
```python
# Example code for training the YOLO model
from ultralytics import YOLO

# Load the YOLO model
model = YOLO('yolov8n.pt')

# Train the model
model.train(data='data.yaml', epochs=50, imgsz=640)
```

1.11.3 3. Dimension Calculation

[ ]:
```python
# Algorithm to calculate brick dimensions from stud positions
def calculate_dimensions(stud_positions):
    rows = len(set([pos[1] for pos in stud_positions]))  # Unique Y-coordinates
    cols = len(set([pos[0] for pos in stud_positions]))  # Unique X-coordinates
    return rows, cols

# Example usage
stud_positions = [(10, 20), (10, 40), (30, 20), (30, 40)]
dimensions = calculate_dimensions(stud_positions)
print(f"Brick dimensions: {dimensions[0]} rows x {dimensions[1]} columns")
```

1.12 Closing Remarks

This notebook not only showcases a technical solution but also tells the story of how a love for learning and persistence can turn a simple idea into a meaningful project. By sharing this journey, I aim to inspire others and demonstrate my readiness for new challenges in the field of technology.