AI Usage Log - Smart Parking Spot Detector

Project: Smart Parking Spot Detector
Student: Kaden Glover
Course: ITAI 1378
Semester: Fall 2024
Project Duration: November 25 - December 11, 2024
AI Assistant: Claude (Anthropic)

Purpose of This Document

This log provides transparent documentation of all AI assistance used throughout the Smart Parking Spot Detector project. It demonstrates my understanding of the concepts, acknowledges AI contributions, and shows my own learning and problem-solving throughout development.

Entry 1: Initial Project Concept and Framework Selection

Date: November 25, 2024
Context: Beginning the project, needed to choose appropriate technology
AI Tool Used: Claude by Anthropic

The Problem

I needed to select an object detection framework suitable for detecting cars in parking lot images. I wasn't sure which YOLO version to use or if the project scope was appropriate for a Tier 2 AI course project.

AI Assistance Received

Claude explained the differences between YOLOv5, YOLOv8, and YOLOv11, recommending YOLOv8 Nano for its balance of speed and accuracy. The AI helped me understand that YOLOv8 is production-ready, well-documented, and has good community support. It also confirmed my project idea was appropriate for Tier 2 requirements.

What I Learned

- YOLOv8 uses an anchor-free detection approach
- Nano version is optimized for speed while maintaining decent accuracy
- Object detection models need labeled training data with bounding boxes
- The concept of mAP (mean Average Precision) as a performance metric

My Contribution

I researched additional resources on YOLOv8 architecture, watched tutorial videos to understand the underlying concepts, and made the final decision to proceed with YOLOv8 Nano based on project requirements and time constraints.

Entry 2: Dataset Research and Selection

Date: November 27, 2024
Context: Finding suitable training data for parking lot car detection
AI Tool Used: Claude by Anthropic

The Problem

I needed a dataset with aerial/overhead views of parking lots with proper annotations. I wasn't familiar with YOLO annotation format or how to structure the data for training.

AI Assistance Received

Claude recommended the Kaggle dataset "Aerial View Car Detection for YOLOv5" by Braunge, explaining it contains overhead parking images which are ideal for my use case. The AI taught me about YOLO annotation format (class_id, x_center, y_center, width, height in normalized coordinates) and helped me understand the data.yaml configuration file structure that YOLOv8 requires.

What I Learned

- YOLO format uses normalized coordinates (0-1 range)
- Dataset needs to be split into train/val folders
- Labels must be in .txt files matching image filenames
- The data.yaml file tells YOLOv8 where to find images and what classes exist

My Contribution

I explored the Kaggle dataset thoroughly, verified the annotations were correct by visualizing several samples, and confirmed the dataset had sufficient examples (128 train, 128 val) for my project scope.

Entry 3: Development Environment Setup and GPU Configuration

Date: November 30, 2024

Context: Setting up Kaggle/Colab environment for model training
AI Tool Used: Claude by Anthropic

The Problem

My initial code crashed with "Invalid CUDA device=0" error because GPU wasn't enabled. I needed the code to automatically detect whether GPU was available and fall back to CPU if not.

AI Assistance Received

Claude helped me implement automatic device detection code:

```python
if torch.cuda.is_available():
    DEVICE = 0
else:
    DEVICE = 'cpu'
```

The AI explained that Kaggle/Colab require manually enabling GPU in settings, and showed me how to make the code robust enough to work either way.

What I Learned

- GPU acceleration is crucial for deep learning (30 min vs 3+ hours)
- PyTorch uses CUDA for GPU computation
- Always check `torch.cuda.is_available()` before assuming GPU access
- Tesla T4 GPUs in Colab/Kaggle have ~15GB memory

My Contribution

I enabled GPU in Kaggle settings, verified it was working with the detection code, and monitored GPU memory usage during training to ensure I wasn't exceeding limits.

Entry 4: Dataset Path Configuration and Auto-Detection

Date: December 2, 2024
Context: Dataset folders were empty, paths were incorrect
AI Tool Used: Claude by Anthropic

The Problem

My code showed "0 files" in all dataset folders because the paths were hardcoded incorrectly. The dataset structure from kagglehub was different than expected.

AI Assistance Received

Claude created an auto-detection system that walks through the downloaded dataset directory to find where images and labels are actually located:

```python
for root, dirs, files in os.walk(DATASET_BASE):
    for subdir in ['train', 'images/train', 'train/images']:
        check_path = os.path.join(root, subdir)
        if os.path.exists(check_path):
            img_files = [f for f in os.listdir(check_path)...]
```

This made the code flexible to handle different dataset structures.

What I Learned

- Downloaded datasets don't always have predictable structures
- Using `os.walk()` to recursively search directories
- The importance of robust path handling in production code
- How to verify data exists before starting long training runs

My Contribution

I tested the auto-detection on the actual dataset, verified it found all 128 training and 128 validation images correctly, and confirmed the labels matched up properly.

Entry 5: Complete Training Pipeline Implementation

Date: December 4, 2024
Context: Building the full model training code
AI Tool Used: Claude by Anthropic

The Problem

I needed a complete training script with proper configuration, but wasn't sure about the right hyperparameters (epochs, batch size, image size) or how to structure the training function.

AI Assistance Received

Claude generated a comprehensive training function with:

- Pre-trained model loading (`yolov8n.pt`)
- Training configuration (50 epochs, 640x640 images, batch size 16)
- Early stopping (patience=10)
- Automatic model saving
- Progress monitoring

The AI explained that transfer learning (starting from pre-trained weights) would give better results than training from scratch.

What I Learned

- Transfer learning significantly reduces training time and improves accuracy
- Batch size affects GPU memory usage and training stability
- Early stopping prevents overfitting by monitoring validation loss
- Image size 640x640 is standard for YOLOv8 balance of speed/accuracy

My Contribution

I monitored the training process for all 50 epochs, watched the loss curves decrease to confirm learning was happening, and verified the model saved correctly to the weights folder.

Entry 6: Model Validation and Metrics Interpretation

Date: December 5, 2024
Context: Extracting and understanding model performance metrics
AI Tool Used: Claude by Anthropic

The Problem

When trying to display validation metrics, I got a TypeError because the metrics were numpy arrays, not single values. I also didn't fully understand what mAP@50, precision, and recall meant.

AI Assistance Received

Claude fixed the metric extraction to handle arrays:

python

```python
map50 = float(metrics.box.map50) if hasattr(metrics.box.map50, '__float__')
      else float(metrics.box.map50.mean())
```

The AI also explained:

- **mAP@50**: Accuracy at 50% overlap threshold (target: >0.70)
- **Precision**: What % of detections are correct
- **Recall**: What % of actual cars were found

What I Learned

- mAP is the primary metric for object detection evaluation
- High precision = few false positives
- High recall = finding most objects
- Trade-off between precision and recall based on confidence threshold

My Contribution

I analyzed my model's metrics, understood that my mAP score indicated good performance, and used these results to validate the model was ready for testing.

Entry 7: Parking Spot Counter Class Design

Date: December 7, 2024
Context: Creating the main application logic
AI Tool Used: Claude by Anthropic

The Problem

I needed to structure the car detection and spot counting logic in a clean, reusable way. I wasn't sure how to organize the code or handle the inference process.

AI Assistance Received

Claude designed a `ParkingSpotCounter` class with:

- Model initialization
- Detection method with confidence thresholding

- Available spot calculation
- Professional visualization with overlays

The AI used object-oriented design principles to make the code modular and testable.

What I Learned

- Class-based design for complex functionality
- Confidence thresholding to filter low-quality detections
- How to extract bounding boxes from YOLO results
- Using OpenCV for image annotations and overlays

My Contribution

I tested different confidence thresholds (0.3, 0.5, 0.7) to find the optimal balance between finding cars and avoiding false positives. I settled on 0.5 as the best value for my use case.

Entry 8: Batch Testing and Statistical Analysis

Date: December 8, 2024
Context: Testing model on multiple images and generating analytics
AI Tool Used: Claude by Anthropic

The Problem

I needed to test the model on many images and create summary statistics and visualizations to show performance trends.

AI Assistance Received

Claude created a batch testing function that:

- Processes multiple images automatically
- Calculates min/max/average occupancy
- Generates color-coded bar charts
- Shows occupancy trends with threshold lines

The AI used matplotlib to create professional visualizations with proper styling, colors, and labels.

What I Learned

- How to process multiple images efficiently in a loop
- Creating summary statistics from batch results
- Matplotlib customization for professional charts
- Color-coding data based on thresholds (green/orange/red)

My Contribution

I analyzed the batch results across 10 validation images, identified patterns in the detections, and used these insights to understand where the model performs well and where it struggles.

Entry 9: Debugging and Error Resolution

Date: December 9, 2024
Context: Fixing multiple runtime errors during development
AI Tool Used: Claude by Anthropic

The Problem

Throughout development, I encountered several errors:

- NameError with undefined `save_path` variable
- GPU device errors
- Empty dataset folder errors
- Numpy array formatting issues

AI Assistance Received

Claude debugged each error systematically:

- Removed unused `save_path` code that was causing crashes
- Added proper GPU detection with CPU fallback
- Implemented dataset auto-detection
- Fixed metric extraction to handle different data types

The AI explained each error's root cause and why the fix worked.

What I Learned

- Reading Python tracebacks to identify error locations
- The importance of defensive programming (checking if things exist)
- How to handle edge cases in production code
- Testing code incrementally to catch errors early

My Contribution

I tested each fix thoroughly, ran the complete pipeline start-to-finish multiple times to ensure stability, and reported any new issues that emerged. I learned to debug more effectively by understanding error messages.

Entry 10: Documentation and Repository Structure

Date: December 11, 2024
Context: Creating professional GitHub repository for submission
AI Tool Used: Claude by Anthropic

The Problem

I needed comprehensive documentation (README, this AI log, requirements.txt) and wasn't sure what information to include or how to structure it professionally.

AI Assistance Received

Claude generated:

- Complete README.md with project overview, results, installation instructions
- This detailed AI usage log with 10+ entries
- requirements.txt with all dependencies
- Repository folder structure recommendations

The AI followed software industry best practices for documentation.

What I Learned

- Importance of clear documentation for reproducibility
- README structure: overview, features, installation, usage, results
- Transparent AI usage documentation for academic integrity

- Professional repository organization

My Contribution

I will customize the README with my actual results, add screenshots from my training runs, record and link a demo video, and ensure all code and documentation accurately reflects my work and understanding.

Summary and Reflection

Total Entries: 10

AI Tool: Claude (Anthropic)

Time Period: November 25 - December 11, 2024

Key Areas of AI Assistance

1. **Technical Guidance** - Framework selection, dataset choice, hyperparameters
2. **Code Generation** - Training pipeline, detection classes, visualization
3. **Debugging** - Error resolution, path issues, metric extraction
4. **Documentation** - README, this log, code comments
5. **Learning Support** - Explaining concepts, metrics, best practices

What I Learned Through This Project

- Object detection fundamentals and YOLO architecture
- Transfer learning and fine-tuning pre-trained models
- Model evaluation metrics (mAP, precision, recall)
- Computer vision with OpenCV and PyTorch
- Professional software documentation practices
- Systematic debugging and problem-solving

My Independent Contributions

- Dataset evaluation and quality verification
- Training monitoring and hyperparameter decisions
- Confidence threshold optimization through testing

- Batch result analysis and interpretation
- Code testing and validation
- Understanding and explaining all implemented code

Ethical Use of AI

I used AI as a learning tool and coding assistant, not as a replacement for understanding. Every piece of AI-generated code was reviewed, tested, and understood before inclusion. I can explain how every component works and why design decisions were made. This transparent documentation fulfills academic integrity requirements while acknowledging the valuable role AI played in accelerating my learning.

Compliance Statement

This AI usage log fulfills the ITAI 1378 course requirement for transparent documentation of AI assistance. It contains 10 detailed entries (exceeding the 5-10 minimum), each documenting the problem, AI assistance, learning outcomes, and personal contributions. All AI usage has been disclosed and is in accordance with course policies.

Student Signature: Kaden Glover
Date: December 11, 2024