**Deep Learning Final Report: Object Detection of People, Animals, Daleks, and Lightsabers**

**Author:** Robert Stevens  
**Course:** ECEN 5060: Deep Learning  
**Date:** April 30, 2025

**1. Introduction**

This project focuses on the application of deep learning to a multi-class object detection task involving real and fictional objects: people, dogs, cats, Daleks, Sith lightsabers, and other lightsabers. Leveraging a pretrained YOLOv8 object detection model, the project implements custom training and inference pipelines, dataset preparation, evaluation metrics, and GitHub-integrated CI/CD for validation. The primary goal is to build a deployable system with robust detection performance and detailed documentation of its training and testing behavior.

**2. Dataset Overview**

The training dataset was constructed from a blend of the COCO subset (for people, cats, and dogs) and custom image datasets for Daleks and lightsabers. The dataset was annotated in YOLO format using both manual and automated tools. The final dataset comprised six classes:

* 0: person
* 1: dog
* 2: cat
* 3: dalek
* 4: other\_lightsaber
* 5: sith\_lightsaber

Dataset directories were structured for use with YOLOv8, including subfolders for images/train, images/val, and images/test, and corresponding labels/ directories.

**3. Model Architecture**

The project uses the YOLOv8 object detection framework provided by Ultralytics, a powerful real-time object detection system known for its speed and accuracy. YOLOv8 ("You Only Look Once") is built on a convolutional neural network backbone, paired with feature pyramid networks (FPN) and path aggregation networks (PANet) to better capture spatial hierarchies at different scales. The detection heads simultaneously predict bounding boxes, objectness scores, and class probabilities, making it a fast and elegant solution for multi-object detection.

A pretrained YOLOv8n model served as the starting point. It was fine-tuned using our combined dataset, which included both real-world and synthetic classes. YOLOv8's modular configuration allowed rapid experimentation, and its robust training routines handled class imbalance and label noise gracefully.

This phase of the project was particularly enjoyable — watching bounding boxes snap into place during live inference felt like digital magic. Seeing a Dalek identified correctly for the first time was genuinely exciting.

Model configuration details:

* **Base Model:** YOLOv8n pretrained weights
* **Input Resolution:** 640x640 pixels
* **Training Epochs:** 50 (early stopping used to prevent overfitting)
* **Optimizer:** Stochastic Gradient Descent (SGD) with Nesterov momentum
* **Loss Function:** Combination of bounding box regression loss, class probability loss, and Distribution Focal Loss (DFL), which improves localization accuracy for small objects

Overall, the architecture provided a solid foundation that made object detection both effective and fun to build.

**4. Training Diagnostics**

Training was conducted locally on an NVIDIA RTX 3060 GPU. The training process included on-the-fly data augmentation and label smoothing.

**Training vs Validation Loss Curve:**

The training and validation loss consistently declined, indicating effective learning and no overfitting. Additional training epochs could yield further improvements.

**5. Evaluation Metrics**

Validation was performed on a held-out test set containing 707 objects across all six classes. The following results were obtained:

| **Class** | **Precision** | **Recall** | **mAP@0.5** |
| --- | --- | --- | --- |
| person | 0.683 | 0.658 | 0.704 |
| dog | 0.521 | 0.600 | 0.604 |
| cat | 0.562 | 0.523 | 0.579 |
| dalek | 0.816 | 0.918 | 0.960 |
| other\_lightsaber | 0.324 | 0.860 | 0.311 |
| Sith\_lightsaber | 0.446 | 0.806 | 0.429 |

High recall was achieved across most classes. Dalek detection was particularly strong, surpassing the 75% accuracy threshold. Lightsaber types presented lower precision due to dataset imbalance and visual similarity.

**6. Code Infrastructure and Automation**

All training and testing scripts were modularized for clarity. A GitHub Actions CI/CD pipeline was implemented to validate train\_combined.py and predict\_combined.py via dry-run tests.

**CI Features:**

* Auto installs dependencies
* Verifies scripts start up correctly
* Skips gracefully if model or data are unavailable

**Documentation:**

* Pydoc HTML generated for all Python scripts
* docs/index.html provides an easy navigation portal for code documentation

**7. Innovation and Technical Challenges**

* Integrated real-time video inference with playback control (pause, speed, jump)
* Implemented conditional label logic (e.g., show only one lightsaber class per frame)
* Addressed annotation edge cases and path portability for local and CI use
* Enabled automated test result logging and evaluation artifact saving

**8. Conclusion**

The project demonstrates a complete object detection pipeline using YOLOv8 for custom and real-world classes. Performance metrics validate the model’s effectiveness, especially for Daleks and people. The addition of CI/CD, documentation, and reproducibility measures further reinforce this project's engineering rigor.

Future work could involve expanding the dataset, refining class definitions (especially for overlapping objects), and deploying the model to edge devices for real-time field use.

**9. Acknowledgments**

This project used the Ultralytics YOLOv8 framework and Microsoft COCO dataset. Assistance from OpenAI's ChatGPT was used for code refactoring, documentation generation, and debugging support.

**10. References**

1. Ultralytics YOLOv8 Documentation. [https://docs.ultralytics.com](https://docs.ultralytics.com/)
2. COCO Dataset. [https://cocodataset.org](https://cocodataset.org/)
3. TorchVision. <https://pytorch.org/vision/>