# 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 report outlines the dataset used, model architecture, training methods, results, and future directions.

## 2. Project Schedule

| **Task** | **Start Date** | **End Date** | **Assigned To** |
| --- | --- | --- | --- |
| Dataset collection & labeling | Mar 15 | Mar 25 | Robert Stevens |
| Model configuration (YOLOv8) | Mar 26 | Mar 30 | Robert Stevens |
| Training / experimentation | Apr 1 | Apr 12 | Robert Stevens |
| Evaluation & analysis | Apr 13 | Apr 22 | Robert Stevens |
| Inference enhancements | Apr 20 | Apr 24 | Robert Stevens |
| Documentation & report writing | Apr 25 | Apr 30 | Robert Stevens |

## 3. Dataset Overview

The dataset included a mix of real-world COCO subset images and custom-collected imagery for Daleks and lightsabers. Each image was labeled using YOLO format. The six class IDs were:

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

**Sample distribution:**

| **Class** | **Approx. Count** |
| --- | --- |
| Person | 2,000+ |
| Dog | ~100 |
| Cat | ~90 |
| Dalek | ~100 |
| Other Lightsaber | ~100 |
| Sith Lightsaber | ~120 |

## 4. Model Architecture & Training Algorithm

We used YOLOv8n, a state-of-the-art object detection framework with the following components:

* **Backbone:** Convolutional neural network with CSP blocks
* **Neck:** Feature Pyramid Network (FPN) + Path Aggregation Network (PANet)
* **Head:** Three detection heads at different resolutions

### Loss Functions:

* **Box loss (CIoU):** Penalizes inaccurate box predictions
* **Classification loss:** Binary Cross-Entropy for class prediction
* **Distribution Focal Loss (DFL):** Refines box regression by learning distributions of offsets

### Training Parameters:

* Input size: 640x640
* Optimizer: Stochastic Gradient Descent (SGD) with Nesterov momentum
* Learning rate: 0.01 with cosine annealing
* Epochs: 50

## 5. Experimental Setup

* **Batch Size:** Automatically selected by YOLOv8 depending on GPU memory (RTX 3060 6GB)
* **Data Augmentation:** Random flips, HSV shifts, and image scaling
* **Overfitting Prevention:** Early stopping based on validation loss, label smoothing
* **Data Loading:** YOLOv8 built-in DataLoader with prefetching, caching, and augmentation pipeline
* **Performance Metrics:** Precision, recall, mAP@0.5, and mAP@0.5–0.95

## 6. Results

**Training vs Validation Loss Curve:**

| **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 |

Dalek detection significantly outperformed others. High recall for lightsaber classes suggests strong localization but poor class separation.

## 7. Summary and Conclusions

This project successfully delivered a functional multi-object detection model using YOLOv8. The model performed well in detecting all six target classes with particular strength in identifying Daleks.

Lessons learned:

* Class imbalance and overlap affect classification confidence
* Real-time video inference is achievable with post-processing filters
* GitHub CI/CD can improve reliability of training pipeline

**Future improvements:**

* Add more balanced examples for dogs, cats, and lightsabers
* Implement bounding box tracking across frames
* Deploy to Jetson Nano or Raspberry Pi for field testing

## 8. References

1. Ultralytics YOLOv8 Documentation – [https://docs.ultralytics.com](https://docs.ultralytics.com/)
2. COCO Dataset – [https://cocodataset.org](https://cocodataset.org/)
3. PyTorch Vision – <https://pytorch.org/vision/>
4. GitHub Actions – <https://docs.github.com/en/actions>
5. YOLOv8 source repo – <https://github.com/ultralytics/ultralytics>

## 9. Appendix – Documented Code Listings

### train\_combined.py

# Sample (partial)

from ultralytics import YOLO

model = YOLO('yolov8n.pt')

model.train(data='data\_6.yaml', epochs=50, imgsz=640)

### test\_combined\_with\_speed\_control.py

# Sample (partial)

from ultralytics import YOLO

model = YOLO("Code/combined\_detector\_final\_fresh4/weights/best.pt")

results = model(frame, imgsz=640, conf=0.3, iou=0.3)

### github\_ci.yml

name: Deep Learning Dry Run

on: [push]

jobs:

dry-run:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: Setup Python

uses: actions/setup-python@v5

with:

python-version: '3.11'

- name: Install dependencies

run: |

pip install -r requirements.txt || true

pip install ultralytics

- name: Dry run training

run: |

python Code/train\_combined.py || echo "Training failed"

[Full code listings available in https://github.com/Robert-E-Stevens/Final-Project-8.git]