UNIVERSITY OF SOUTH CAROLINA

PROJECT MILESTONE #1

UNIVERSITY OF SOUTH CAROLINA

COMPUTER SCIENCE AND ENGINEERING

CSCE-585: Machine Learning Systems
Project Proposal: Visual M.L. System for Identifying and Minimizing Civilian Fatality in Urban War Zones

---

Authors:

Logan Nall (github.com/Clnall),
Walter Pach (github.com/waltster),
Vera Svensson (github.com/Lux-Vera)

October 4, 2022

1 Project Repository

The repository for our team can be found at https://github.com/csce585-mlsystems/MilCiv. This repository is used for collaborating, issue tracking, and updating/version control. This milestone can be identified by the GitHub tag: **Milestone1**.

2 Introduction

As A.I. systems are increasingly deployed in battlefield environments, there is a need for efficient models that distinguish between civilian and military targets. Our project aims to create a M.L. system that is able to provide this functionality.

3 Problem Statement

Our goal is to develop an A.I. model that is enabled to differentiate between civilian and military vehicles. The problem presented includes both the issue of accuracy of characterization and the ability to audit the success of the system with a clearly defined confidence metric.

4 Technical Approach

- The data set was pre-labelled and we developed a Jupyter notebook to convert the CSV and images into TensorFlow Record format.
- We created a Jupyter Notebook to download and configure the dependencies for the project and format their configuration appropriately.
- The model's environment is configured from the Common Objects in Context model with TensorFlow object detection.
- The model is trained for fifty-five thousand steps uniformly using the `helper.sh` script on the data set and monitored using TensorBoard.
- The model is evaluated using the `helper.sh` script with the same data set but different images.
- Batch Size: 4
- Base Learning Rate: .0014
- Warm-Up Learning Rate: 0.0013333

To lint the code we have created a workflow in Github Actions. This makes the code easier to work with and help us find errors. The workflow is activated on every push to the repository to make sure unstructured code and possible errors are eliminated as early as possible.

We are using Super-Linter in our workflow, which is a combination of linters that has support for Python, among other languages [1]. However, it does not have support for ipynb files. Thus those files will have to be converted to Python before running the linter [2]. This is also be done by the workflow.

5 Intermediate/Preliminary Approach and Results

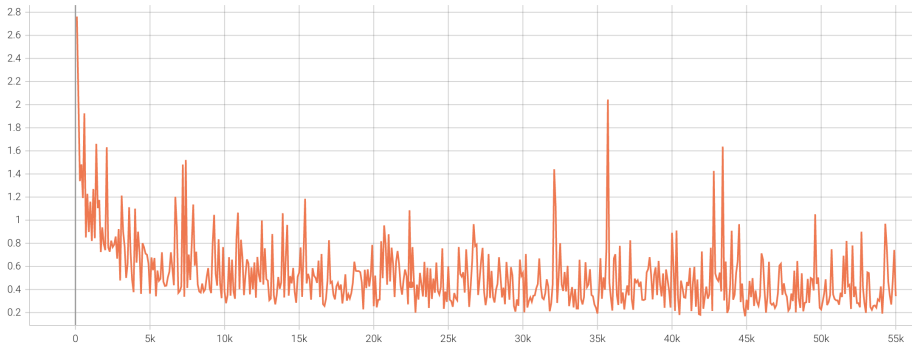

Figure 1: Classification Loss

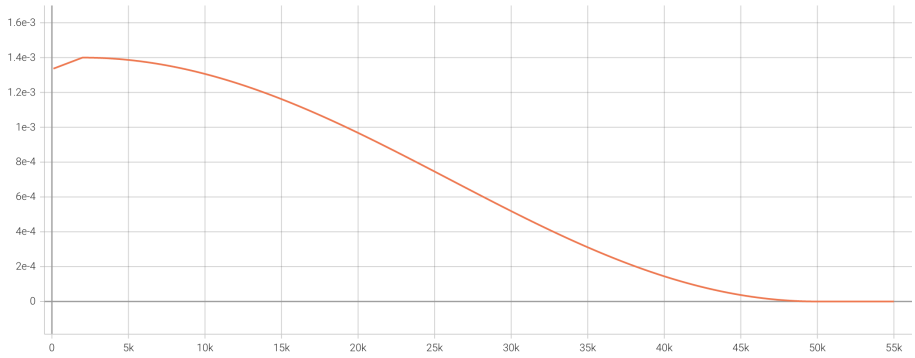

Figure 2: Learning Rate

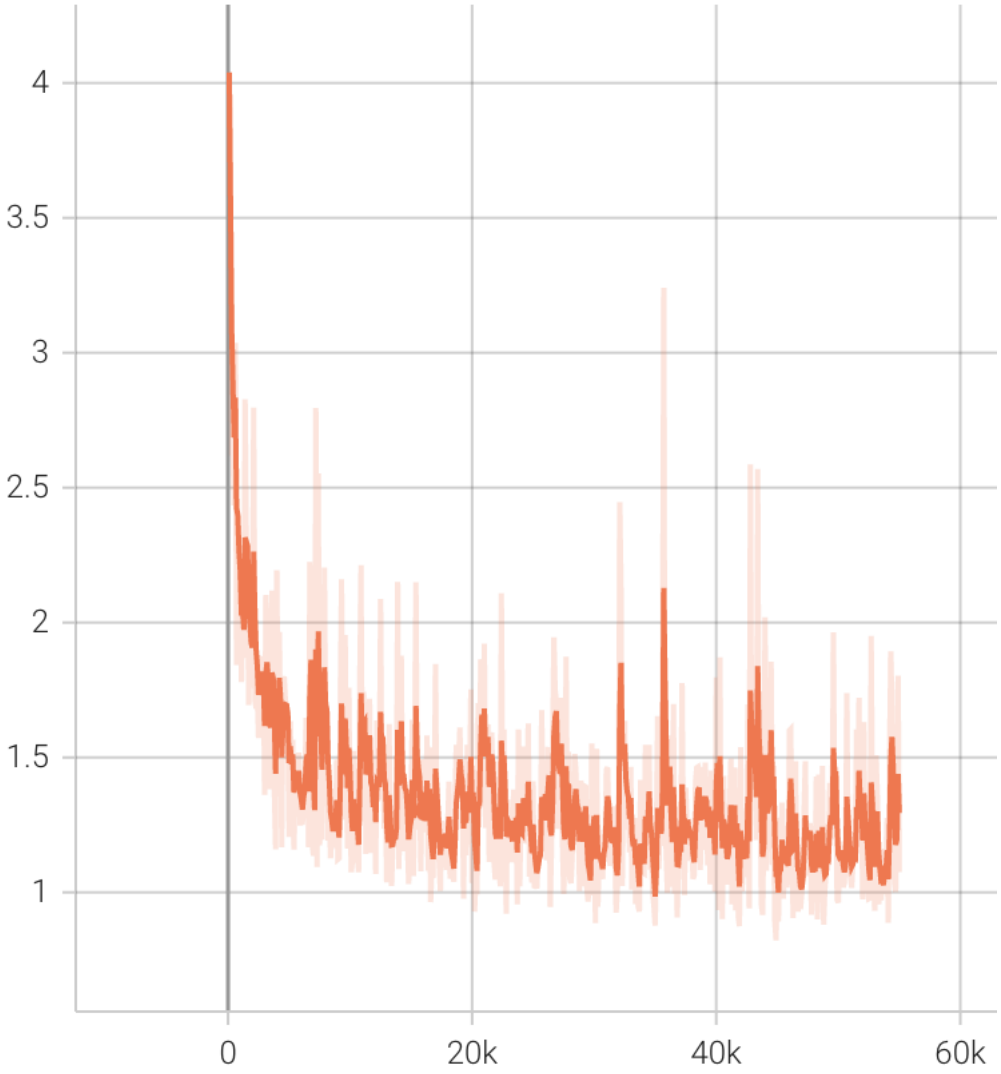

Figure 3: Total Loss Smoothed

It is clear from Figure 1 that the model began to improve tremendously from the origin to five-thousand steps. However, there is a large degree of variance following this, suggesting additional training is required. The spike at approximately step thirty-six thousand indicates that there was an image that was invalid/impossible to parse that disrupted the model's training.

From 2 it is clear that the learning rate smoothly approached zero as the training progressed, suggesting a natural tuning of the model. Previous training cycles with larger learning rates to begin with quickly grew exponentially.

The results so far are promising but fall a little short of the criteria performance for our project. The model can do solid detection on some images, however looking at Figure 1 for the Classification Loss, the data approaches about 0.4 where 0.0 would be perfect.

References

[1] GitHub. Super-Linter. GitHub;. [cited 2 October 2022]. Available at https://github.com/github/super-linter#supported-linters. pages 3

[2] notebooks with GitHub Actions L. Donagh Horgan. Donagh Horgan; 2021. [cited 2 October 2022]. Available at https://donagh.io/2021/05/10/linting-notebooks-with-github-actions.html. pages 3