KeneXoft TECHNOLOGIES

Enabling Excellence Through IT

Documentation of Landmine Detection Machine Learning Model

-Help Doc

Table of Contents

Introduction

Detecting landmines using machine learning, particularly employing Convolutional Neural Networks (CNNs) with TensorFlow. The fusion of advanced algorithms and robust data preparation techniques is instrumental in creating models capable of accurately identifying landmines amidst diverse environmental contexts. This page distils the essential methodologies involved in the process, from meticulously preparing datasets and crafting intricate model architectures to rigorously training and evaluating the efficacy of these systems.

Data Preparation:

The dataset consists of JPG images representing various scenarios for object detection, encompassing class of landmines. TensorFlow's Dataset API facilitates efficient data loading in batches. Augmentation, via the Albumentations library, includes techniques such as random cropping and flipping, enhancing dataset diversity. Visualization pre- and post-augmentation aids in understanding technique impact, crucial for robust model training. Benefits include enhanced diversity, streamlined loading, and insight into augmentation effectiveness.

Model Architecture:

Our model integrates CNNs for classification and localization tasks. Utilizing pre-trained VGG16 backbone enables effective feature extraction. Custom heads handle classification and bounding box regression, each with a global max-pooling layer and fully connected layers. The model's output comprises classification probabilities and bounding box coordinates. A visual architecture diagram illustrates the flow, ensuring simultaneous object detection and classification.

Training & Evaluation:

Training involves the augmented dataset, employing Adam optimizer with polynomial decay learning rate. Loss functions encompass binary cross-entropy for classification and custom localization loss for regression. Monitoring training with loss curves ensures convergence and tracks classification accuracy and regression loss. Evaluation on a separate test dataset measures real-world performance, with metrics like accuracy, precision, recall, and F1-score computed. Conclusion highlights model efficacy, guiding future improvements through fine-tuning, ensemble methods, or advanced architectures.

Configurations and Prerequisites

Hardware Configurations:

Note: These are just Recommended configurations and can be ignored if you are using Cloud Environment.

CPU: Dual Core or more, >1.0 GHz

RAM: 2 GB or more

Disk Space: 4 GB or more

GPU: GPU is preferred

*For faster training of Neural network.

Microsoft Visual C++ 2015-2022 Redistributable.

Prerequisites:

Must have basic understanding of Python.

And have knowledge on Machine Learning models and libraries.

Instructions

Download the dataset and source code from here.

There are 2 methods for execution:

1. Anaconda (Local Environment)
2. Colab or etc. (Cloud Environment).

Steps for using Anaconda:

Follow this for installation of Anaconda.

1. Launch Anaconda Navigator.
2. Navigate to Jupyter Notebook and launch it.
3. Open the downloaded notebook file and insert a new code cell at the beginning to enter the following command to make sure the necessary libraries are installed.

```
pip install tensorflow opencv-python matplotlib json albumentations
```

Steps for using Colab or etc.:

Follow this for getting started with Colab.

1. Enter into Colab Environment.
2. Open the downloaded notebook file and insert a new code cell at the beginning to enter the following command to install necessary libraries.

```
pip install tensorflow opencv-python matplotlib json albumentations
```

After completing either one of the methods follow below steps for executing the code to implement the model:

1. The code consists of following sections:
    a. Data Preprocessing.
    b. Data Loading.
    c. Implementation of Neural Network Architecture.
    d. Training of Model.
    e. Testing and Evaluating the Model's performance and visualizing it.
2. Run all the cells by one at a time to understand its output clearly.

Note: 1. Training part of model can be skipped instead load the saved model.
2. Read the comment lines for better understanding of the code.