AWS Machine Learning Engineer Nanodegree

Capstone Proposal

Airplane Object Detection Project

Author:
Tomás Baidal

Proposal

Domain Background

The detection of airplanes has become a pivotal concern in the spheres of airport security and air traffic control strategies. Historically, detection and monitoring have mainly depended on human surveillance and traditional tracking systems. However, the advent of extensive data availability and progress in machine learning have enabled the development of computer vision models as potent tools for identifying airplanes in still images.

The motivation for this project could emanate from the growing need for more efficient air traffic control mechanisms, especially in regions with high flight density or near busy airports. Automated detection of airplanes in images can support airport security personnel and air traffic controllers by offering real-time alerts, reducing human error, and enhancing efficiency and safety in air operations.

Problem Statement

The paramount challenge to tackle is the identification and detection of airplanes in various contexts, particularly focusing on areas like busy airports and dense air traffic regions. Conventional methods, such as human surveillance and standard air traffic control systems, have inherent limitations, including human errors, which can potentially result in serious safety concerns.

This project endeavors to fulfill the need for a more reliable, adept, and precise system for detecting airplanes in static images. The system aims to provide instantaneous, accurate predictions through an AWS Gateway endpoint, enabling users to upload images and receive immediate predictions. The primary aspects of the problem this project seeks to address are:

- **Accuracy & Efficiency:** Utilizing machine learning and computer vision to detect airplanes with optimal precision and speed.

- **Real-time Response Enhancement:** Elevating the real-time response capabilities of air traffic control personnel in varied environments through automated alerts and detection, empowering them to make immediate and informed decisions.

Solution Statement

This project presents a comprehensive solution grounded in machine learning and cloud-based technologies. Our approach merges traditional object detection methodologies with state-of-the-art machine learning practices to offer a solution. By utilizing the ResNet50 model, fine-tuned on our dataset, and benefiting from the power of transfer learning from ImageNet, we aim to harness patterns from extensive and diverse datasets. This contrasts with the pre-trained TensorFlow model, educated on the COCO 2017 dataset, which already possesses an inherent ability to detect airplanes that we will use as a benchmark model.

However, the broader scope of this project is not just about model performance. It's about the exploration and comparison of different strategies to understand object detection deeply. Such insights will drive future iterations, leading us towards an optimal solution for airplane detection.

To architect this solution:

- **AWS Sagemaker:** Serves as the backbone, streamlining the processes of building, training, and deploying machine learning models.

- **Google's Open Images Dataset V7:** Provides a diverse foundation for training scenarios. Transfer Learning with ResNet-50: Alongside hyperparameter tuning, this approach will be employed to enhance the model's performance.

- **Evaluation Metric:** Models will be evaluated based on the mean average precision (mAP) score and IoU.

Post-development infrastructure:

- **AWS Lambda Functions:** Building the pipeline, automating tasks and API requests.

- **AWS API Gateway:** Bridges user requests to the model, ensuring interactions.

- **Python Dash App:** A user-friendly interface for direct model interaction, allowing image uploads for inference.

In essence, the culmination of this project will be a scalable, secure, and sophisticated object detection system, adeptly identifying airplanes in varied conditions and settings, and encapsulating the best of machine learning and cloud technologies.

Datasets and Inputs

We will be using datasets extracted from Google's Open Images Dataset V7, systematically arranged into distinct training and validation sections, both in folders and dataframes.

Dataset are composed by:

- **Training set:**
    - Train Image files: 773 Images
    - Train dataset: 1352 rows

- **Test set:**
    - Test Image files: 260 Images
    - Test dataset: 338 rows

- **Validation set:**
    - Validation Image files: 250 Images
    - Validation dataset: 325 rows

Datasets are composed by the following columns:

- **ClassName:** name of our class. In this case we will only use "Airplane" class
- **ImageID:** Id of the image. This is a reference to the image located in the folder.
- **XMin, YMin, XMax, YMax:** coordinates of the bounding boxes.

Eventually we have more rows than images. The reason is that some images have more than one bounding box, so ImageID values can be duplicated in the dataset, meaning that the image has more than one airplane.

With our datasets we will plan to perform:

- Ensure the datasets are optimized and pertinent to the project's primary objective of airplane detection.
- EDA and data preprocessing.
- Feature Engineering and Data Augmentation.
- Data preparation for MXNet algorithm generating *RecordIO* files.
- Inference in our models and compare results.

Benchmark Model

We will use a pre-trained model from TensorFlow, specifically the SSD ResNet50 V1 FPN 640x640, often referred to as RetinaNet50. This is an object detection model optimized for both speed and accuracy. It combines the power of the ResNet50 architecture, known for its deep residual layers and robust feature extraction, with the Single Shot MultiBox Detector (SSD) methodology for rapid object detection. The model is pre-trained on the COCO 2017 dataset. Given its capability to detect 80 classes it is particularly useful for our project since it has been trained to recognize various objects, including airplanes. Thus, we will not need to fine-tune this model to use it as a benchmark.

Regardless of which model ultimately performs better, the main purpose of our project is to learn, experiment, and adapt. The absolute performance of the models is only a part of the entire project; the methodology, experimental design, and interpretation of the results are equally vital. Furthermore, by considering and contrasting different models and approaches, we're showcasing a comprehensive strategy for the airplane detection task. Our choice of the RetinaNet50 benchmark model aligns with this philosophy, providing us with a solid baseline against which we can measure and understand our experimental model.

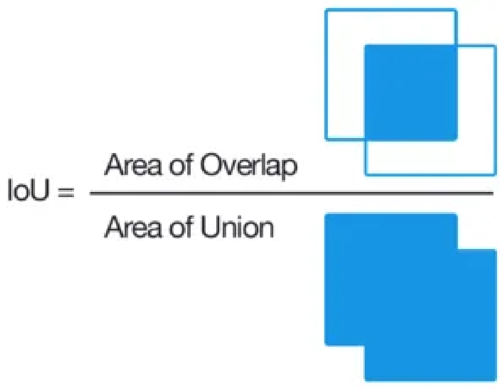
Evaluation Metrics

For this project, we will be focusing on three primary evaluation metrics: Intersection over Union (IoU) and Mean Average Precision (mAP)

- **Intersection over Union (IoU):**

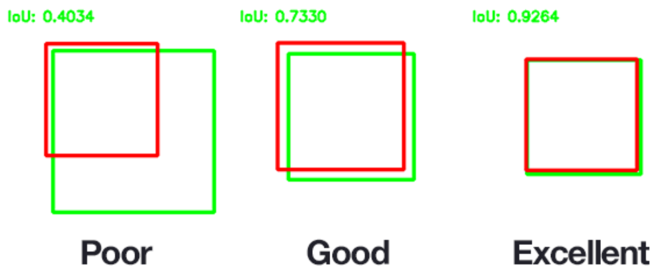
IoU measures the overlap between two regions, typically the predicted bounding box area from the model and the actual bounding box (ground truth).

It is calculated by dividing the area of Intersection (the overlapping region between the predicted and actual bounding box) by the area of Union (the combined area of both the predicted and actual bounding box):



Source: Wikipedia

In our project, after our model makes predictions on the validation dataset, we will compute the IoU for each prediction compared to its ground truth.



Source: Wikipedia

A higher IoU value (close to 1) suggests a more accurate prediction, whereas a lower IoU (close to 0) indicates a less precise prediction. This metric will provide insight into the spatial accuracy of our object detection model and compare it with the benchmark model.

- **Mean Average Precision (mAP):**

mAP is a comprehensive metric used to evaluate the accuracy of an object detection model across various confidence thresholds. It computes the average precision values for each object class, taking into account both false positives and false negatives.

- **Precision:** Of all detections claimed positive by the model, how many are actually positive.

- **Recall:** Out of all possible true positives, how many were detected by the model.

In our project, for our benchmark model, we will refer to the documented mAP value, which gives an overall performance metric based on prior evaluations. For the model we develop, we will compute the mAP using predictions on our validation dataset. This will allow us to compare the two models' effectiveness in object detection across various confidence thresholds and object classes.

Project Design

1 - Data Preprocess and Feature Engineering

In the project's initial stages, we plan to carry out extensive EDA and visualization to understand our 'weapon' class images, utilizing Python, pandas, and Matplotlib for insightful dataset overviews. Following this, we'll engage in meticulous data pre-processing to filter, clean, and split our data, ensuring the formation of coherent and consistent training and test sets. Further, feature engineering will see the preparation of our datasets to meet the MXNet framework's requirements, with careful formatting and restructuring to ensure seamless integration. We will then augment our dataset through horizontal flipping, enhancing the model's learning capability by expanding the dataset. Lastly, the creation of MXNet .rec files will optimize data handling and processing for effective model training, focusing on the conversion efficiency to ensure a user-friendly experience in future modeling phases:

- **EDA and Visualization:** Initial exploration and visualization of 'weapon' class images using Python, pandas, and Matplotlib to gain insights.

- **Data Pre-processing:** Filtering, cleaning, and splitting datasets to form consistent training and test sets.

- **Feature Engineering:** Preparing datasets for the MXNet framework, involving careful restructuring and formatting.

- **Data Augmentation:** Expanding the dataset through horizontal flipping to enhance the model's learning capability.

- **RecordIO files generation:** Converting datasets into MXNet's RecordIO format to optimize data handling and streamline the model training process.

2 - Hyperparameter Tuning and Model Deployment

We will perform an extensive hyperparameter tuning to optimize the object detection model's performance. Hyperparameter tuning involves experimenting with multiple values and combinations to identify the most effective set, improving model accuracy and efficiency. We will focus on optimizing critical hyperparameters, such as learning rate, mini-batch size, and optimizer type, to enhance model convergence and learning capabilities. The process will be iterative, employing techniques like grid search or random search, and we will utilize SageMaker's Hyperparameter Tuner to automate and streamline this essential optimization phase. This tuning will be crucial for achieving high mAP (Mean Average Precision) scores and ensuring the model's robustness and reliability in detecting weapons in images.

After the hyperparameter tuning, we will perform inference in our model using the validation dataset and compare it with the results of the benchmarking model using the metrics we selected. After that, we will deploy one of the models.

3 - AWS Lambda, API Gateway and Dash App implementation

After deploying the endpoint, a crucial step is the creation of an AWS Lambda function, which serves as an automation engine to run the backend logic. The Lambda function will be invoked through a REST API created using AWS API Gateway. This API will act as the conduit between the user interface and the machine learning model, allowing secure and scalable interactions.

Lambda Function: The AWS Lambda function will be designed to receive image inputs, preprocess the image data, and interact with the deployed model, managing the execution of predictions.

API Gateway: The API Gateway will serve as a secure entry point, exposing the necessary functionalities to interact with the backend services. It will handle the HTTP request and response transformations and routes the requests to the appropriate Lambda function.

Python Dash App: The web app will allow users to interact with the machine learning model and enable users to upload images and receive predictions from the model.

References

Evaluation Metrics:

- https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173
- https://yanfengliux.medium.com/the-confusing-metrics-of-ap-and-map-for-object-detection-3113ba0386ef
- https://towardsdatascience.com/introduction-to-object-detection-model-evaluation-3a789220a9bf

Object Detection:

- https://medium.com/visionwizard/object-detection-4bf3edadf07f
- https://pub.towardsai.net/maximizing-the-impact-of-data-augmentation-effective-techniques-and-best-practices-c4cad9cd16e4
- https://docs.aws.amazon.com/sagemaker/latest/dg/object-detection-tensorflow.html
- https://github.com/aws/amazon-sagemaker-examples/blob/main/introduction_to_amazon_algorithms/object_detection_tensorflow/Amazon_Tensorflow_Object_Detection.ipynb