ISSN: 2583-9292 (ONLINE)
DOI: 10.xx/ijdsml.2024.xx

HIGH-EFFICIENCY AERIAL DETECTION WITH VISION TRANSFORMERS AND PYSPARK IN A DISTRIBUTED FRAMEWORK

Arshi Jamal and K. Ramesh
Department of Computer Science, Karnataka State Akkamahadevi Women's University, India

Abstract

Image processing tasks in Computer vision such as segmentation, object detection, and classification are foundational in geosciences, driving advancements by enabling precise analysis and interpretation of Earth's dynamic systems and landscapes. The processing of high-resolution aerial images for object detection presents significant challenges, including the need for high detection accuracy and the ability to handle vast datasets effectively. Traditional methods often struggle with the scale and complexity of such tasks, necessitating innovations that can leverage distributed computing to meet these demands. This study introduces a groundbreaking framework that integrates Vision Transformers, a cutting-edge architecture for object detection, with PySpark's distributed computing capabilities. This inference model significantly enhances batch inference processing efficiency of voluminous datasets, enabling the analysis of high-resolution aerial imagery with notable accuracy. By utilizing Resilient Distributed Datasets (RDDs), the research offers a detailed algorithmic analysis that reveals the computational advantages of this PySpark-based approach. The proposed Vision Transformer-PySpark framework is evaluated on the DOTA benchmark dataset for aerial images, demonstrating its scalability and superior performance as the amount of computing nodes rise, achieving improved scalability. The comparison of this framework against cutting edge object detection models underscores it's effectiveness and scalability, setting a new standard for efficient, large-scale aerial image analysis in distributed computing environments.

Keywords:

Object Detection, Aerial Detection, DOTA, Transformers, Distributed Computing

1. INTRODUCTION

. Aerial imagery, captured by satellites and drones, provides a crucial perspective for monitoring Earth's landscapes. These images are widely used in domains like disaster management, urban planning, environmental preservation, and ecological monitoring. However, detection of objects in aerial imagery poses complexities due to several interrelated factors. Variability in object size, depending on distance, and frequent obstruction by larger structures or terrain complicate detection efforts. Recognizing and classifying objects across varying scales in densely packed environments requires sophisticated algorithms. Additionally, high-resolution satellite images generate large file sizes, demanding significant computational power for effective processing and analysis. These challenges necessitate advanced techniques to achieve accurate results. This complexity highlights the need for advanced detection algorithms capable of handling both the scale and detail present in aerial imagery [1]. Convolutional Neural Networks (CNNs) transformed object detection by automating the process of feature extraction, eliminating the need for manual hand-crafted features [2]. Advances like Region-based CNN (R-CNN) further improved detection accuracy by introducing a two-step process of region proposals and classification [3]. However, these methods often rely on sequential processing, limiting their suitability for real-time applications.

To overcome this, the YOLO (You Only Look Once) model was developed, offering real-time object detection by processing images in a single pass. YOLO's architecture allows simultaneous prediction of bounding boxes and class probabilities, achieving impressive frame rates. Despite this, it faces contextual challenges with high-resolution satellite imagery, where varying object scales and complex scenes including occlusion impede its performance. Vision Transformers (ViTs) address some of these challenges by utilizing self-attention as well as multi-head attention mechanisms to model global relationships in images, making them more effective at handling varying object scales and densely packed environments [4]. Unlike CNNs, which focus on local pixel relationships, ViTs process images as sequences of patches, learning spatial relationships through self-attention [5]. This approach enhances their ability to capture high-level features, making them particularly well-suited for complex datasets like aerial and satellite imagery.

However, like other deep learning models, ViTs struggle with the vast scale of satellite image archives [6]. Datasets such as NASA's LANDSAT, which include over 10 million images, present significant storage and processing challenges. It becomes challenging to manage large volumes of data with traditional single-node systems. While ViTs excel in speed for standalone tasks, they fall short in batch processing large datasets. For this, big data platforms offer potential solutions, but disk-based systems like Apache Hadoop are inefficient for deep learning-based inference tasks. In contrast, Apache Spark, with its in-memory computing capabilities, provides a more scalable solution, capable of efficiently handling large- scale satellite imagery data without the limitations of disk- based processing. This research combines the advanced object detection capabilities of Vision Transformers with PySpark's distributed computing framework. By leveraging PySpark's in-memory processing and distributed data architecture, ViT-based object detection is parallelized across clusters, enabling scalable batch processing of massive aerial image datasets. This approach allows Vision Transformers to process high- resolution imagery more efficiently. Our framework demonstrates superior performance on the DOTA dataset and shows remarkable scalability, from a single node to a 50-node cluster. This integration represents a significant leap forward in aerial image analysis, providing a powerful and scalable solution for large- scale satellite data processing using Vision Transformers. The contributions of the paper are summarised as follows:

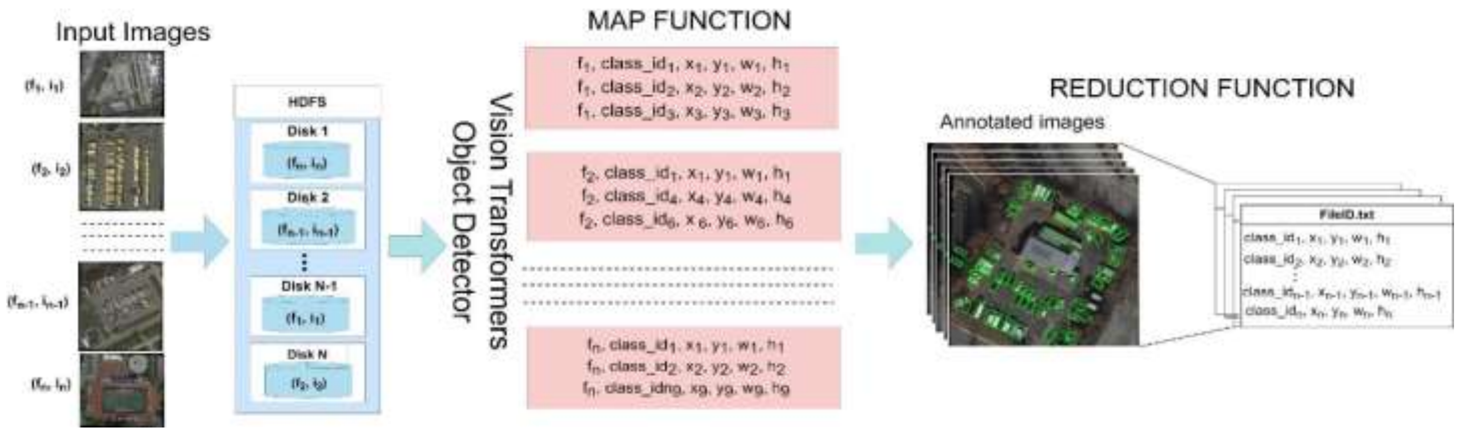

Input Images
HDFS
Disk 1
Disk 2
Disk N-1
Disk N
Vision Transformers Object Detector
MAP FUNCTION
f1, class_id1, x1, y1, w1, h1
f1, class_id2, x2, y2, w2, h2
f1, class_id3, x3, y3, w3, h3
f2, class_id1, x1, y1, w1, h1
f2, class_id4, x4, y4, w4, h4
f2, class_id6, x6, y6, w6, h6
fn, class_id1, x1, y1, w1, h1
fn, class_id2, x2, y2, w2, h2
fn, class_idng, xg, yg, wg, hg
REDUCTION FUNCTION
Annotated images
FileID.txt
class_id1, x1, y1, w1, h1
class_id2, x2, y2, w2, h2
class_idn-1, xn-1, yn-1, wn-1, hn-1
class_idn, xn, yn, wn, hn

Fig.1. Scalable Vision Transformers object detection inference module on PySpark cluster. Input images are loaded as (key, value) pairs into HDFS where key is file name and value is the image matrix

- **Advanced Object Detection**: The paper proposes employing of Vision Transformers (ViTs) for object detection in aerial imagery, addressing challenges posed by varying object scales and complex scenes with occlusion.
- **Scalable Big Data Processing**: It proposes a novel combination of ViT-based object detection with Apache Spark's distributed in-memory computing framework to process large-scale datasets like satellite imagery efficiently.
- **Enhanced Performance**: The framework demonstrates superior performance on the DOTA dataset and showcases its scalability from a single node to a 50-node cluster, offering an efficient solution for real-time and batch processing of high- resolution satellite images.

The organisation of the research paper is as follows: A comprehensive review of related work is presented in Section 2, with an outline of important developments in object detection. Section 3 delves into Vision Transformers (ViTs), discussing their architecture and relevance to the field. In Section 4, we present an enhanced object detection approach that integrates Vision Transformers with PySpark to improve scalability and performance. The details of the experimental results are discussed in Section 5, with a focus on performance metrics as well as comparative analysis. Finally, the conclusion of the paper is presented in Section 6 with a summary of the key findings of the research and proposing future research directions.

2. RELATED WORK

2.1 TRADITIONAL METHODS IN OBJECT DETECTION

Since the early 1990s, object detection has been a primary focus of computer vision research. Traditional, rule-based algorithms dominated early efforts but struggled with effective image representation. The emergence of deep learning brought a major shift, enabling more complex feature extraction and improving performance, marking a pivotal transformation in the field [7].

that highlights various algorithms that were developed with handcrafted features and were most popular in the last decade. One of them included was Deformable Parts Model (DPM) [8]. It has been among the most notable traditional algorithm for object detection and employs a graphical model to integrate carefully designed low- level features with part decompositions based on kinematic principles. However, DPM fails when it comes to computational efficiency and scalability as well as it struggles with capturing large variations in object appearance and pose, especially when objects undergo significant deformation or occlusion.

2.2 DEEP LEARNING APPROACHES

The introduction of deep learning algorithms has revolutionized object detection by enabling autonomous learning of feature representations and patterns from data and Convolutional Neural Networks (CNNs) [2] have been particularly effective in extracting meaningful features from images. A major breakthrough occurred when AlexNet [9] won the 2012 ImageNet competition, significantly surpassing previous pioneering models in prediction accuracy through the use of deep convolutional networks. Despite these advancements, CNNs still face challenges in detecting objects across varying scales and resolutions and struggle with capturing spatial dependencies or high-level features from images as convolutional filters used by CNNs majorly focus on local patches of the input image.

A significant advancement in addressing the limitations of CNNs for object detection was the introduction of Regions with CNN (RCNN) by Girshick et al. [3]. RCNN utilized selective search to generate object proposals, which were then analyzed using a pre-trained CNN. However, it suffered from slow detection speeds due to repetitive feature computation. The Spatial Pyramid Pooling Network (SPPNet) [10] improved efficiency by computing feature maps for the entire image at once, although it had limitations in its training process. Fast RCNN [11] further enhanced both speed and accuracy by enabling simultaneous training of the detector and the bounding box regressor. Building on this, Faster RCNN [12] introduced a Region Proposal Network (RPN), achieved near real-time detection while still retaining some computational redundancies in earlier stages. To enhance real- time object detection, the YOLO model was introduced, known for its compact design and fast detection capabilities. YOLO marked a notable advancement in one-shot object detection models, as it employs a single convolutional layer to forecast bounding boxes and class probabilities simultaneously. Moreover, it is optimized for speed and has low computational overhead. It is also notable that due to YOLO architecture, its implementation does not require extensive modification leading to ease of implementation. However, it faces challenges when applied to aerial imagery, where the complexities of satellite images such as variability in sizes and occlusion can hinder performance.

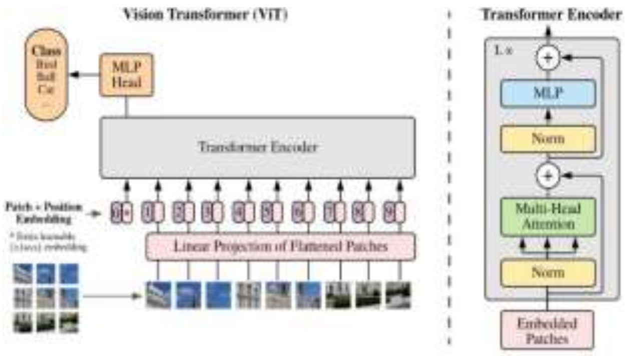


Fig.2. A basic architecture of vision transformers encoder [4]

The introduction of ViTs to computer vision was based on the success of Transformers in NLP. ViTs [4] became the first object detection models to apply transformer architectures directly to image data. The images are divided into patches to process them as a sequence of linear embeddings resembling tokens in NLP. Notably, ViTs demonstrated exceptional performance, particularly when trained on larger datasets, marking a new era in object detection.

2.3 DISTRIBUTED COMPUTING FOR OBJECT DETECTION

Although, Vision Transformers (ViTs) have demonstrated significant potential in object detection tasks, yet, they encounter challenges related to computational complexity, often prioritizing performance over speed. When applied to real-time object detection on large-scale image datasets, ViTs face scalability issues that demand substantial computational resources. To overcome these hurdles, leveraging big data platforms emerges as a promising approach to improve scalability and resource efficiency. Classification of Big Data platforms can be done into two types; vertical and horizontal scaling tools [13]. Vertical tools, such as GPUs and multi-core systems, offer significant computational power but are limited by shared memory, making them less suitable for large-scale, high-dimensional image processing [14]-[16]. In contrast, horizontal scaling systems like grid computing and Apache Hadoop provide scalable, cost-effective solutions for big data challenges [17], [18]. While grid computing struggles with data recovery and job completion, Hadoop offers distributed storage, computation, fault tolerance, and automatic data recovery, though it excels primarily in batch processing and is inefficient for real-time tasks. Apache Spark addresses this limitation by supporting both batch and real-time processing through its in-memory computation engine, advancing the efficiency of complex, multi-stage computational jobs. Studies have explored the optimization of object detection using distributed computing, with one example leveraging Hadoop and MapReduce for large-scale image processing, highlighting the potential of these systems for handling vast image datasets [19].

Our study advances the field by combining Vision Transformers with PySpark, addressing scalability and computational efficiency issues inherent in analyzing high resolution aerial imagery. We leverage a User-Defined Function (UDF) that operates on RDDs containing images to execute object detection tasks within Apache Spark's in memory computing environment.

3. EXPERIMENTAL METHOD

3.1 VISION TRANSFORMERS

Transformers are self-attention-based deep learning models originally designed for Natural Language Processing (NLP) tasks. They are typically trained through a two-step process, that is, pre-training on a large text dataset, followed by targeted fine-tuning on a task-specific dataset. This approach allows for the development of models with unprecedented sizes, exceeding 100 billion parameters, due to their computational efficiency and scalability. Inspired by these abilities, Vision Transformers (ViTs) were developed to adapt Transformer architectures for processing image data. The architecture of a Vision Transformers model involves segmentation of images into a non-overlapping sequence of patches. Each patch is subsequently reduced and linearly embedded into a vector, which is used as input for the transformer model. Moreover, positional encodings are added to the patch embeddings, helping the model to learn the relative positions of the patches within the image. The core of ViT architecture is composed of multiple layers of transformer encoders. The Fig.2 depicts a structure of the transformer encoder. It consists of the following layers:

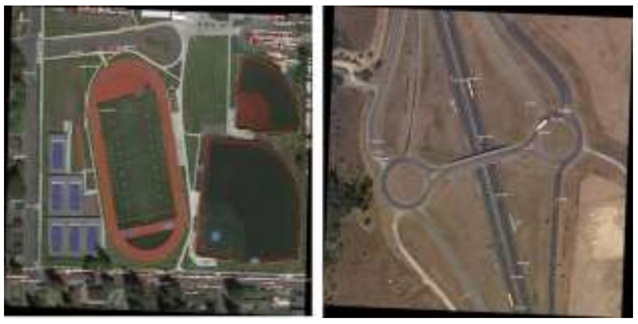


Fig.3. Detection results drawn from the final annotations generated from ViT+PySpark module for sample images. The red bounding boxes depict predicted and green are actual

- **Multi-Head Self Attention Layer:** This layer combines all the attention outputs by concatenating them into the appropriate dimensions. The multiple attention heads facilitate the learning of both local and global dependencies within an image.
- **Multi-Layer Perceptron (MLP) Layer:** This layer consists of two layers that utilize the Gaussian Error Linear Unit (GELU) activation function. The inclusion of a classification head following the MLP adds a special token with the input patch embeddings, the called classification token. The inference made by the model depends on the token output.
- **Norm Layer:** Added before each block, layer normalization does not introduce any new dependencies among the training

images. This practice improves not only the performance of the model but also the speed of training.

Furthermore, Residual connections are incorporated after each block to enable components to pass through the network directly, bypassing non-linear activations. ViTs have exhibited cutting-edge performance across various benchmarks, particularly when trained using extensive datasets, positioning them as a powerful alternative to traditional CNN-based models. In the context of object etection in aerial imagery, ViTs leverage the strengths of Transformers, offering a novel approach to image understanding that excels in capturing rich contextual information.

3.2 PYSPARK-ENHANCED OBJECT DETECTION WITH VISION TRANSFORMERS

At the heart of the Spark programming paradigm are Resilient Distributed Datasets (RDDs), which encapsulate all input data, intermediate computations, and final outputs [20]. Each transformation or reduction of an RDD results in a new, immutable RDD. Spark ensures fault tolerance through lineage graphs, which tracks the sequence of operations on RDDs and allow for the reconstruction of lost data partitions in case of runtime failures [21]. Typically, input images are stored in the Hadoop Distributed File System (HDFS). The process starts by retrieving these images from HDFS to create an RDD, which is then distributed across worker nodes for processing. Operations on RDDs are mapped to these nodes and executed on local data segments.

The Fig.1 depicts the workflow of Vision Transformers integrated with Spark RDDs. The process starts with gathering input images, where each image is represented as a (key, value) pair: the filename (f_1, f_2,…,f_n) serves as the key, and the corresponding image data matrix (f_1, f_2,…,f_n) serves as the value. Parallel processing of these images takes place after they are uploaded to HDFS and distributed across multiple disks. Further computational optimization can be achieved by dividing images into grids. In this approach, combinations of files and grids are represented as (f_1+g_1,m_1) (f_2+g_2,m_2),..., where (f_n+g_n) are the keys and $m1$and $m2$ are the values. While this method improves computational efficiency, it requires specialized algorithms to address the challenge of overlapping objects spanning multiple grids. For simplicity and to maintain detection accuracy, this discussion does not explore image slicing in detail. After storage, a map function triggers the ViT object detection algorithm on each image. The algorithm scans the images, identifying objects and assigning each a class identifier (*class id*$_1$, *class id*$_2$,…,*class id*$_3$) along with bounding box parameters (x, y, width w, height h). Once object detection is complete, a reduction function consolidates the detected data, retaining only relevant information. This process results in annotated images with objects clearly marked by bounding boxes. Additionally, a corresponding text file (FileID.txt) is generated for each image, detailing class IDs and bounding box coordinates for all detected objects. This efficient process enhances object detection scalability and performance in distributed computing environments.

3.3 DOTA DATASET

The DOTA dataset [1] was introduced to boost object detection in the field of Earth Vision. It is an extensive dataset created for detecting objects in aerial imagery. It consists of images from various satellites and sensors, gathered through crowdsourcing and featuring multiple resolutions, along with expert vetting, ensuring a collection of of high-quality images. DOTA includes annotations for 15 common object categories using oriented bounding boxes (OBBs) and horizontal bounding boxes (HBBs) which enhances the object detection accuracy, particularly for distinguishing closely positioned objects. The dataset also includes objects of varying scales, orientations, and shapes, all precisely annotated. To date, three versions of DOTA have been released, each improving on the last, with added annotations for small objects and increased category variations.

4. EXPERIMENTAL RESULTS

4.1 TRAINING VISION TRANSFORMERS

To initialize the training of ViT’s, we utilize the PyTorch framework, which facilitates model instantiation. The initialization process involves setting up various layers and parameters. We trained the ViT model for 500 epochs with patience 12 and a batch size of 128 on an NVIDIA A100 GPU, using the standard train/validation split from the DOTA dataset. Image resolution used was of 1280 × 1280. During training, we monitored loss convergence with test and value losses reaching near to 0.05. For model validation on the test set, we used mean Average Precision (mAP), which averages AP across all categories [22] and IoU thresholds, along with the area beneath the precision-recall curve. After training, the model was applied to DOTA images for object detection, generating an output array of predicted classes and bounding box coordinates. Fig.3 displays annotation results from ViTs applied on sample images. The green boxes in the images represent the ground truth annotations and red boxes indicates the model's predictions.

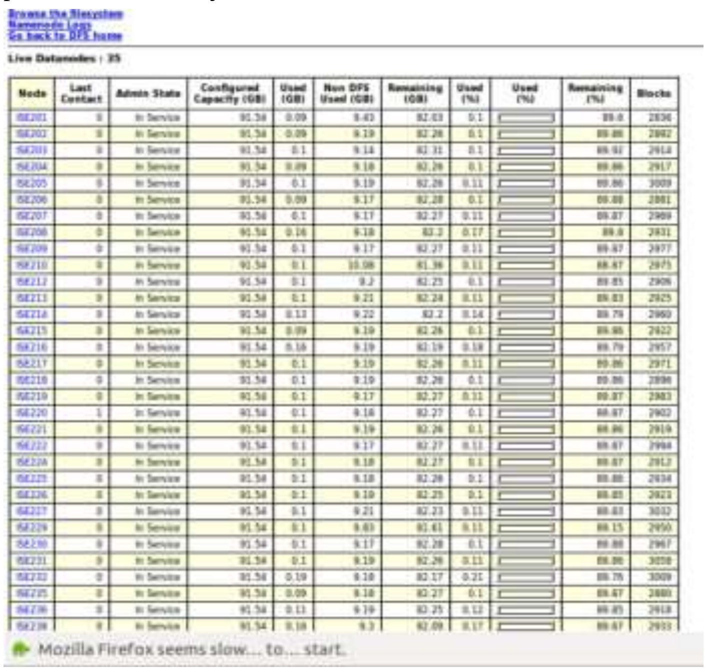
4.2 PYSPARK BASED OBJECT DETECTION

In this phase, we utilize the RDD-based algorithm for object detection, using the ViT model trained during the offline stages as described earlier. The process involves several key steps: setting up the Hadoop cluster, loading images and trained model file into HDFS, performing object detection, and saving the annotation results as text or JSON files.

These operations are carried out through a series of PySpark transformations, followed by action operations. PySpark's integration with HDFS enables efficient distributed computing on large datasets stored in the system.

4.3 EXPERIMENTAL SET

An in-house Hadoop cluster with configurations of 5, 10, 35, and 50 nodes were built at the Computer Science Lab, Akkamahadevi Women’s University Campus. Fig.4 shows a snapshot of the 35-node Hadoop cluster. It details the configuration of each node that was setup on the system depicting properties like configured capacity, capacity used, non-DFS capacity used, remaining capacity, capacity used in percentage and remaining capacity in percentage. The system was setup with Hadoop version 2.7.3 and Apache Spark version 3.1.2 to ensure compatibility and optimize the data processing capabilities of both platforms. For the development of object detection models, we used Python version 3.10 and OpenCV version 4.6.0. The experiments were conducted using Vision Transformers on the PySpark cluster with varying node counts and dataset sizes. A replication factor of 3 was set, and YARN cluster optimization parameters were adjusted.



Browse the filesystem
Namenode Logs
Go back to DFS home

Live Datanodes : 35

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)	Remaining (%)	Blocks
ISE201	0	In Service	91.54	0.09	9.43	82.03	0.1	▭	89.6	2836
ISE202	0	In Service	91.54	0.09	9.19	82.26	0.1	▭	89.88	2892
ISE203	0	In Service	91.54	0.1	9.14	82.31	0.1	▭	89.92	2914
ISE204	0	In Service	91.54	0.09	9.18	82.26	0.1	▭	89.86	2917
ISE205	0	In Service	91.54	0.1	9.19	82.26	0.11	▭	89.86	3009
ISE206	0	In Service	91.54	0.09	9.17	82.28	0.1	▭	89.88	2881
ISE207	0	In Service	91.54	0.1	9.17	82.27	0.11	▭	89.87	2989
ISE208	0	In Service	91.54	0.16	9.18	82.2	0.17	▭	89.8	2931
ISE209	0	In Service	91.54	0.1	9.17	82.27	0.11	▭	89.87	2977
ISE210	0	In Service	91.54	0.1	10.08	81.36	0.11	▭	88.87	2975
ISE212	0	In Service	91.54	0.1	9.2	82.25	0.1	▭	89.85	2906
ISE213	0	In Service	91.54	0.1	9.21	82.24	0.11	▭	89.83	2925
ISE214	0	In Service	91.54	0.13	9.22	82.2	0.14	▭	89.79	2960
ISE215	0	In Service	91.54	0.09	9.19	82.26	0.1	▭	89.86	2922
ISE216	0	In Service	91.54	0.16	9.19	82.19	0.18	▭	89.79	2957
ISE217	0	In Service	91.54	0.1	9.19	82.26	0.11	▭	89.86	2971
ISE218	0	In Service	91.54	0.1	9.19	82.26	0.1	▭	89.86	2898
ISE219	0	In Service	91.54	0.1	9.17	82.27	0.11	▭	89.87	2983
ISE220	1	In Service	91.54	0.1	9.18	82.27	0.1	▭	89.87	2902
ISE221	0	In Service	91.54	0.1	9.19	82.26	0.1	▭	89.86	2919
ISE222	0	In Service	91.54	0.1	9.17	82.27	0.11	▭	89.87	2984
ISE224	0	In Service	91.54	0.1	9.18	82.27	0.1	▭	89.87	2912
ISE225	0	In Service	91.54	0.1	9.18	82.26	0.1	▭	89.88	2834
ISE226	0	In Service	91.54	0.1	9.19	82.25	0.1	▭	89.85	2923
ISE227	0	In Service	91.54	0.1	9.21	82.23	0.11	▭	89.83	3032
ISE229	0	In Service	91.54	0.1	9.83	81.61	0.11	▭	89.15	2950
ISE230	0	In Service	91.54	0.1	9.17	82.28	0.1	▭	89.88	2967
ISE231	0	In Service	91.54	0.1	9.19	82.26	0.11	▭	89.86	3058
ISE232	0	In Service	91.54	0.19	9.18	82.17	0.21	▭	89.76	3009
ISE235	0	In Service	91.54	0.09	9.18	82.27	0.1	▭	89.87	2880
ISE236	0	In Service	91.54	0.11	9.19	82.25	0.12	▭	89.85	2918
ISE238	0	In Service	91.54	0.16	9.3	82.09	0.17	▭	89.67	2933

Mozilla Firefox seems slow... to... start.

Fig.4. Snapshot of in-house Hadoop cluster built at the university campus

4.4 PERFORMANCE EVALUATION OF PYSPARK CLUSTER

To evaluate the computational performance of Vision Transformer inference model implemented on PySpark, we conducted experiments with 10,000 DOTA images, each scaled down to a resolution of 1280 × 1280 pixels. The workflow involved initially uploading the images to the HDFS, followed by processing in the detector module for object detection tasks. We systematically assessed the detection performance across four different clusters sizes, analyzing ten different image sets. The inference time for each batch was meticulously recorded. To guarantee the reliability of our results, each experimental setup was on DOTA replicated five times, as well as the mean computational time was documented. It was noted that a rising number of nodes in a PySpark cluster leads to an approximately linear improvement in computational cost. The Fig.5 represents the computational costs involved in processing various different image batches across distinct number of nodes in a PySpark cluster. It was observed that increasing the number of nodes results in a nearly linear reduction in computational costs.

Table.1. Comparison of Performance of PySpark based ViT object detection system with varying PySpark cluster nodes

Model	Dataset	Size (px)	mAP	Speed (ms)
ViT+1-node Pyspark cluster	DOTAv2.0	1280	77.74	200
ViT+5-node Pyspark cluster	DOTAv2.0	1280	78.68	49.02
ViT+10-node Pyspark cluster	DOTAv2.0	1280	81.32	28.29
ViT+35-node Pyspark cluster	DOTAv2.0	1280	81.32	11.54
ViT+50-node Pyspark cluster	DOTAv2.0	1280	80.87	6.86

4.5 COMPARISON WITH DOTA BASELINES

In order to evaluate the performance of PySpark-based ViT object detection system, we used the DOTAv2.0 dataset. The system was tested on 10,000 images of the dataset which were downscaled to resolution of 1280 × 1280 pixels with varying numbers of PySpark cluster nodes. Moreover, all models were trained from scratch on the DOTAv2.0 dataset.

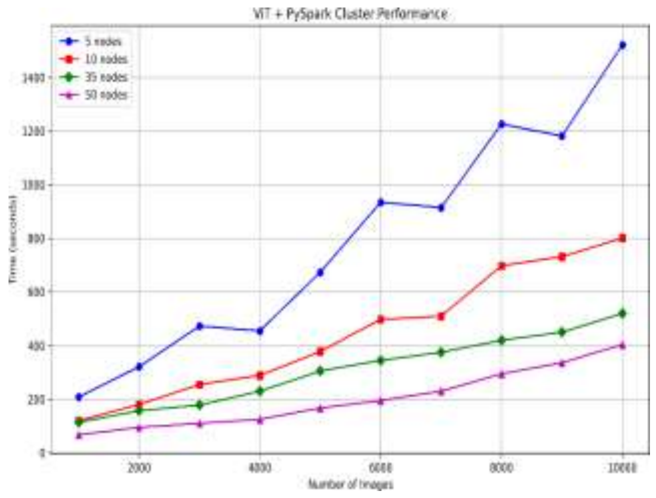


Fig.5. Computational performance analysis of PySpark clusters of size 5, 10, 35, 50. Each subplot depicts computational cost against number of images

The Table.1 presents the results of the experiments where the mean average precision (mAP) was computed followed by the speed of inference in milliseconds. As evident from Table 1, increasing the number of nodes in the cluster leads to a notable improvement in both mAP and inference speed. The mAP increases progressively from 77.74% with a 1-node cluster to a peak of 81.32% with a 10-node cluster, demonstrating the model's enhanced ability to detect objects accurately with additional computing resources. In terms of speed, the inference time per frame significantly improves as the cluster size grows, dropping from 200 ms in a 1-node cluster to just 6.86 ms in a 50-node cluster. This highlights the computational efficiency gained by leveraging a larger PySpark cluster, enabling faster processing without sacrificing much accuracy.

4.6 COMPARISON OF PYSPARK-BASED VIT AGAINST SOTA MODELS

For a more comprehensive evaluation of PySpark based Vision Transformers, it is compared with other pioneering object detection models on the DOTA dataset. The results of the comparison analysis are depicted in Table 2. For evaluation, different versions of DOTA were utilized. All images used for evaluation were downscaled to 1280 × 1280 pixels and mean average precision (mAP) was computed as the precision metrics.

All models are pre-trained using the pytorch framework It is notable that this research is primarily concentrated on exploring the computational performance incorporating Vision Transformers model into PySpark batch processing operations. Consequently, no significant efforts were made to improve the existing model architectures. In addition to the DOTA-v2.0, this setup was tested on the DOTAv1.0 and DOTAv1.5 versions and compared with baseline models including RetinaNet OBB [23], Fast R-CNN [11], Faster RCNN [12], Mask R-CNN [24] Cascade R-CNN [25], Hybrid Task Cascade [26] variants augmented with Dpool and RoI Transformer modules and YOLOv8. This comparative study focused on both OBB and HBB accuracy to ensure an equitable evaluation across all models. The analysis underscores the performance benefits of incorporating distributed computing into object detection workflows. The Table.2 depicts a comparison of object detection performance of a PySpark-based ViT model against various state-of-the-art models based on pixel size, speed in frames per second (fps), and mean Average Precision (mAP) for both Horizontal Bounding Boxes (HBB) and Oriented Bounding Boxes (OBB). The findings corroborate that the PySpark based Vision Transformer object detection stands out in terms of speed without compromising on accuracy when measured against its counterpart. Moreover, the results in Table 2 depicts that one of the most notable aspects of the ViT models is their impressive processing speed, particularly when deployed in a PySpark environment with varying cluster sizes. For instance, the 50-node configuration achieves a remarkable speed of 145.76 fps, significantly outpacing many traditional models, such as Mask R-CNN and YOLOv8n-obb, which operate at lower fps rates of 0.3 and 22 respectively.

In terms of detection accuracy, the mAP values for the ViT models exhibit competitive performance, though they generally trail behind some leading architectures like, the Faster R-CNN OBB + RoI Transformer achieves a high mAP of 74.59% on DOTA (1.0), while the ViT models in the 1-node and 5-node configurations achieve mAP scores of 65.53% and 63.17%, respectively, for the same dataset. This trend indicates that the ViT models can effectively perform object detection tasks, although there is room for improvement in their accuracy relative to the highest performing models. Another noteworthy observation is the performance trend of the ViT models as the number of nodes in the PySpark cluster increases. The mAP results reveal a slight decline in performance for HBB detection, with scores decreasing from 65.53% at 1 node to 62.16% at 35 nodes. This trend suggests that while increasing the cluster size enhances processing speed, it may also introduce complexities that impact detection accuracy.

Table.2. Vision Transformers performance evaluation against state-of-the-art methods on three different versions of DOTA dataset

Method	**Size (pixels)**	**Speed (frames per second)**	**DOTA (1.0)**		**DOTA (1.5)**		**DOTA (2.0)**	
			HBB (mAP%)	**OBB (mAP%)**	**HBB (mAP%)**	**OBB (mAP%)**	**HBB (mAP%)**	**OBB (mAP%)**
RetinaNet	1280	3.0	67.45	-	61.64	-	49.31	-
RetinaNet OBB	1280	1.2	69.05	66.28	62.49	59.16	49.26	46.68
Mask R-CNN	1280	0.3	71.61	70.71	64.54	62.67	51.16	49.47
Cascade Mask R-CNN	1280	0.8	71.61	70.71	64.54	62.67	51.16	49.47
Hybrid Task Cascade	1280	5.1	72.49	71.21	64.47	63.40	50.88	50.34
Faster R-CNN	1280	4.9	70.76	-	64.16	-	50.71	-
Faster R-CNN OBB	1280	4.5	71.93	69.40	63.82	62.01	49.36	47.31
Faster R-CNN OBB +Dpool	1280	3.2	71.84	70.15	63.65	62.22	50.49	48.77
Faster R-CNN OBB +HOB	1280	1.3	70.37	70.11	64.43	62.57	50.38	48.90
Faster R-CNN OBB + RoI Transformer	1280	1.0	74.59	73.76	66.09	65.03	53.37	52.81
YOLOv8n-obb	1280	22	64.87	56.43	54.98	48.89	39.58	45.01
ViT+ 1-node PySpark cluster	1280	05	65.53	57.53	55.43	51.74	40.69	45.61
ViT+ 5-nodes PySpark cluster	1280	20.4	63.17	58.52	57.44	50.91	41.59	45.13
ViT + 10-nodes PySpark cluster	1280	35.35	63.19	58.54	57.42	50.91	41.60	45.18
ViT + 35-nodes ySpark cluster	1280	86.67	62.16	56.55	56.40	51.93	42.58	43.10
ViT + 50-node ySpark cluster	1280	145.76	62.18	55.54	58.43	49.93	42.60	44.13

Nonetheless, the ViT model manages to maintain a reasonable level of accuracy while achieving significant speed improvements, highlighting a noteworthy trade-off between speed and precision. Overall, the results presented in the tables illustrate that the PySpark-based ViT model, despite facing competition from traditional object detection models in terms of precision, showcases significant advantages in processing speed. This balance of speed and reasonable accuracy positions the ViT model as a viable option for applications requiring rapid object detection, where quick response times are crucial.

5. CONCLUSION

This research underscores the substantial advantages of integrating Vision Transformer (ViT) models with PySpark for object detection tasks in real-time applications. Our comprehensive evaluation, utilizing the DOTA dataset, reveals that the PySpark-based ViT model achieves impressive processing speeds, particularly when deployed across varying cluster sizes. Specifically, the proposed framework achieved a notable peak performance of 145.76 frames per second (fps) with a 50-node cluster, significantly outperforming many traditional models. This enhancement in speed is particularly valuable in contexts where rapid response times are essential for improving customer experiences and operational efficiency where the ViT model demonstrates competitive mean average Precision (mAP) scores, peaking at 81.32% with a 10-node cluster, it trails behind certain state-of-the-art architectures, such as Faster R-CNN OBB + RoI Transformer. This performance gap indicates an opportunity for future enhancements, suggesting that while the model excels in speed, further optimization is necessary to improve accuracy. The results emphasize that achieving a balance between processing speed and detection precision is crucial for deploying these models in real-world scenarios. Future research should focus on bridging the accuracy gap identified in this study. Approaches such as fine-tuning pre trained models on domain-specific datasets can be explored to leverage existing knowledge for improved performance. Additionally, incorporating advanced techniques, including multi-scale feature extraction and sophisticated augmentation strategies, may yield significant improvements in detection capabilities. Moreover, investigating the integration of alternative distributed computing frameworks with the ViT model could provide deeper insights into optimizing performance and scalability. Overall, this research lays a robust foundation for advancing object detection capabilities within distributed computing environments. By addressing the identified limitations and exploring innovative approaches, future studies can further enhance the effectiveness of ViT models in various applications, paving the way for significant advancements in the field of computer vision.

REFERENCES

[1] G.S. Xia, "DOTA: A Large-Scale Dataset for Object Detection in Aerial Images", *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pp. 1-6, 2018.

[2] R.L. Galvez, A.A. Bandala, E.P. Dadios, R.R.P. Vicerra and J.M.Z. Maningo, "Object Detection using Convolutional Neural Networks," *Proceedings of International Conference on TENCON* 2, pp. 1-6, 2018.

[3] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pp. 580-587, 2014.

[4] A. Dosovitskiy, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", *Proceedings of International Conference on Learning Representations*, pp. 1-7, 2021.

[5] M.M. Naseer, K. Ranasinghe, S.H. Khan, M. Hayat, F.S. Khan and M.H. Yang, "Intriguing Properties of Vision Transformers", *Proceedings of International Conference on Neural Information Processing Systems*, pp. 23296-23308, 2021.

[6] H. Touvron, M. Cord, Alaaeldin El-Nouby, J. Verbeek and Herve Jegou, "Three Things Everyone Should Know About Vision Transformers", *Lecture Notes in Computer Science*, pp. 497-515, 2022.

[7] Z. Zou, K. Chen, Z. Shi, Y. Guo and J. Ye, "Object Detection in 20 Years: A Survey", *Proceedings of the IEEE*, Vol. 111, No. 3, pp. 1-20, 2023.

[8] P.F. Felzenszwalb, R.B. Girshick, D. McAllester and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 9, pp. 1627-1645, 2010.

[9] A. Krizhevsky, I. Sutskever and G.E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Communications of the ACM*, Vol. 60, No. 6, pp. 84-90, 2012.

[10] K. He, X. Zhang, S. Ren and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition", *Computer Vision-ECCV*, pp. 346-361, 2014.

[11] R. Girshick, "Fast R-CNN", *Proceedings of International Conference on Computer Vision*, pp. 1-6, 2015.

[12] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 6, pp. 1137-1149, 2017.

[13] D. Singh and C.K. Reddy, "A Survey on Platforms for Big Data Analytics", *Journal of Big Data*, Vol. 2, No. 1, 2015.

[14] M. Bordallo Lopez, A. Nieto, J. Boutellier, J. Hannuksela and O. Silven, "Evaluation of Real-Time LBP Computing in Multiple Architectures", *Journal of Real-Time Image Processing*, Vol. 13, No. 2, pp. 375-396, 2014.

[15] E. Christophe, J. Michel and Jordi Inglada, "Remote Sensing Processing: From Multicore to GPU", *Proceedings of the IEEE*, Vol. 4, No. 3, pp. 643-652, 2011.

[16] D. Castano-Diez, D. Moser, A. Schoenegger, S. Pruggnaller and A.S. Frangakis, "Performance Evaluation of Image Processing Algorithms on the GPU", *Journal of Structural Biology*, Vol. 164, No. 1, pp. 153-160, 2008.

[17] V. Boudet, F. Rastello and Y. Robert, "A Proposal for a Heterogeneous Cluster ScaLAPACK (dense linear solvers)", *IEEE Transactions on Computers*, Vol. 50, No. 10, pp. 1052-1070, 2001.

[18] R. Buyya and S. Venugopal, "A Gentle Introduction to Grid Computing and Technologies", *CSI Communications*, Vol. 29, pp. 9-19, 2004.

[19] E. Serrano, Javier Garcia Blas, J. Carretero, M. Abella and M. Desco, "Medical Imaging Processing on a Big Data Platform using Python: Experiences with Heterogeneous and Homogeneous Architectures", *Proceedings of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 1-7, 2017.

[20] Chaganti Sri Karthikeya Sahith, Satish Muppidi and Suneetha Merugula, "Apache Spark Big data Analysis, Performance Tuning and Spark Application Optimization", *Proceedings of International Conference on Evolutionary Algorithms and Soft Computing Techniques*, pp. 1-6, 2023.

[21] Matei Zaharia, "An Architecture for Fast and General Data Processing on Large Clusters", *Proceedings of International Conference on Association for Computing Machinery*, pp. 1-7, 2016.

[22] R. Padilla, S.L. Netto and E.A.B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms", *Proceedings of International Conference on Systems, Signals and Image Processing*, Vol. 1, No. 1, 2020.

[23] T.Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollar, "Focal Loss for Dense Object Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1-6, 2018.

[24] K. He, G. Gkioxari, P. Dollar and R. Girshick, "Mask R-CNN", *Computer Vision and Pattern Recognition*, pp. 1-7, 2018.

[25] M. Yaseen, "What is YOLOv8: An In-Depth Explorationof the Internal Features of the Next-Generation Object Detector", Available at https://arxiv.org/html/2408.15857, Accessed in 2024.

[26] P. Nawrot, J. Chorowski, A. Lancucki and E.M. Ponti, "Efficient Transformers with Dynamic Token Pooling", *Proceedings of Annual Meeting of the Association for Computational Linguistics*, Vol. 1, pp. 6403-6417, 2023.