sensors

MDPI

Article

Detecting Human Actions in Drone Images Using YoloV5 and Stochastic Gradient Boosting

Tasweer Ahmad [1,*], Marc Cavazza [2], Yutaka Matsuo [3] and Helmut Prendinger [2]

[1] Department of Electrical and Computer Engineering, COMSATS University Islamabad, Islamabad 45550, Pakistan
[2] National Institute of Informatics, Tokyo 101-8430, Japan
[3] Department of Engineering, The University of Tokyo, Tokyo 113-8654, Japan
* Correspondence: tasveerahmad@gmail.com; Tel.: +92-333-677-2430

Abstract: Human action recognition and detection from unmanned aerial vehicles (UAVs), or drones, has emerged as a popular technical challenge in recent years, since it is related to many use case scenarios from environmental monitoring to search and rescue. It faces a number of difficulties mainly due to image acquisition and contents, and processing constraints. Since drones' flying conditions constrain image acquisition, human subjects may appear in images at variable scales, orientations, and occlusion, which makes action recognition more difficult. We explore low-resource methods for ML (machine learning)-based action recognition using a previously collected real-world dataset (the "Okutama-Action" dataset). This dataset contains representative situations for action recognition, yet is controlled for image acquisition parameters such as camera angle or flight altitude. We investigate a combination of object recognition and classifier techniques to support single-image action identification. Our architecture integrates YoloV5 with a gradient boosting classifier; the rationale is to use a scalable and efficient object recognition system coupled with a classifier that is able to incorporate samples of variable difficulty. In an ablation study, we test different architectures of YoloV5 and evaluate the performance of our method on Okutama-Action dataset. Our approach outperformed previous architectures applied to the Okutama dataset, which differed by their object identification and classification pipeline: we hypothesize that this is a consequence of both Yolov5 performance and the overall adequacy of our pipeline to the specificities of the Okutama dataset in terms of bias–variance tradeoff.

Keywords: action detection; YoloV5; gradient boosting classifier

Citation: Ahmad, T.; Cavazza, M.; Matsuo, Y.; Prendinger, H. Detecting Human Actions in Drone Images Using YoloV5 and Stochastic Gradient Boosting. *Sensors* **2022**, *22*, 7020. https://doi.org/10.3390/s22187020

Academic Editor: Benoit Vozel

Received: 7 August 2022
Accepted: 14 September 2022
Published: 16 September 2022

1. Introduction

In recent years, drones and unmanned aerial vehicles (UAVs) have found numerous applications in urban surveillance, search and rescue, and situational awareness applications. Several of these applications require the ability to recognize actions from UAV cameras, either through video or single-image analysis. Human action recognition is considered to be a challenging task, which has been fairly addressed over the last decade [1]. However, extending this task to drone-captured images and videos is an emerging topic. Human action recognition is a well-studied problem which is categorized into (i) pose-based [2–4], (ii) single-image-based [5–7], and (iii) video-based action recognition [8–10]. However, detecting actions in single images is a less explored area because it faces the problem of the unavailability of annotated temporal data for action detection [11–13]. This task requires the integration of components for entity detection, and classification that can be adapted to the distribution of target situations, as well as practical deployment constraints. With the availability of popular and efficient object detection methods such as Yolo, it becomes possible to envision solutions that incorporate a detection module on top of object recognition. Eweiwi et al. [2] built an efficient pose-based action recognition using

histograms of relative location, velocity, etc., and thus learned a compact and discriminative representation. Wang et al. [3] recognized human actions by obtaining K-best estimations and additional segmentation cues. An encoder-based approach was proposed by [4], which efficiently encodes 3D skeleton information using pose descriptor and uses extreme learning machine for classification. A single-image-based action recognition was devised by [5], which segments out the human body into five parts: head, torso, arms, hands, and legs. A semi-FCN network detects each of these five body parts and then Action ResNet predicts the action for each body parts. Finally, an SVM fuses these five body part actions and thus recognizes the entire body action. Sreela et al. [6] evolved the action recognition model using residual neural network as feature extractor and support vector machine as the classifier. For single-image action recognition, Liu et al. [7] suppressed misleading context of person bounding boxes using guided-loss activation with ResNet-50 deep learning architecture. Several video-based action recognition techniques were summarized by [8] for three types of benchmark datasets, i.e., single-viewpoint, multi-viewpoint, and RGB-depth video. Pareek et al. [9] briefly covered the human action recognition techniques using machine learning and deep learning methods for the years 2011–2019. Pham et al. [10] presented most important deep learning models for action recognition, analyzed their performance, and then discussed future prospects and challenges for recognizing actions in realistic videos. Rohrbach et al. [11] prepared a dataset for kitchen activities. Singh et al. [12] used LSTM for building a multi-stream bidirectional network for action detection. Reinforcement learning was used by [13] for detecting different actions in videos.

In previous work on pedestrian detection, Zhang et al. [14] showed the effectiveness of gradient boosting. Such a technique can also be applied to action detection and recognition from single images in UAV data. Therefore, we adopted gradient boosting on top of Yolo-based pedestrian detection [14] as an action classifier.

Human action detection in aerial images is closely related to pedestrian detection and tracking in UAV videos [14,15]. Liu et al. [16] applied the Yolo architecture to small object detection in UAV image data. Mittal et al. [17] highlighted various small object detection techniques for UAV video data. Shinde et al. [18] employed a vanilla Yolo architecture for detection and localization of human activities in the Liris dataset [19]. Very recently, Jung et al. [20] improved the YoloV5 architecture by modifying convolution layer architecture and activation function for efficient object and person detection in aerial images of the VisDrone dataset. Caputo et al. [21] examined lightweight versions such as YoloV5s and YoloV5m for speedy search and rescue of humans in dangerous situations. The authors evaluated their models on the two newly collected HERIDAL and SAR datasets.

The Yolo series [22–25] has played an important role as a single-stage detector in object detection and action detection tasks. In this paper, we specifically investigate YoloV5 with gradient boosting for solving the problem of human action recognition and localization in drone videos. Our proposed method is able to address the challenges of drone camera motion, smaller actor size, and the limited size of aerial action datasets for training the model.

The main contribution of this paper is to suggest an action recognition pipeline compatible with use in UAVs on real-world datasets, such as the Okutama-Action dataset, based on state-of-the-art object recognition techniques. As a secondary contribution, we also experiment with different variants of YoloV5 and propose a compromise between model performance, size, and computational complexity.

In the remainder of this paper, Section 2 summarizes the related work. We explain our proposed methodology in Section 3, while experimental details and results are discussed in Section 4. Finally, we conclude our paper in Section 5.

2. Related Work

Over the last decade, recognizing human actions in videos has been challenging and an active area of research among the computer vision community [26–34]. However,

the recognition and detection of actions in aerial images is a less developed area, and differs from previous work that simply adopts the perspective of pedestrians in the scene [31,35].

Over the course of time, several aerial action datasets have been collected. The UCF-ARG dataset [36] contains 10 realistic human actions in the three settings of ground, rooftop, and aerial triplets. This is considered to be a challenging dataset as the dataset contains various instances of camera motion and humans tend to occupy only a few pixels within images. Perera et al. [37] recorded a slow and low-altitude (about 10 ft) UAV video dataset for detecting 13 different gestures in aerial videos. These gestures are mainly related to UAV navigation and aircraft handling. The authors investigated a pose-based convolutional neural network (CNN) for this work. Ding et al. [38] overcame the challenging problem of heavy computations by devising a lightweight model for real-world drone action recognition. The backbone architecture for this method contains a temporal segment network with MobileNetV3, where temporal structures are responsible for capturing self-attention and the focal loss emphasizes misclassified samples. Geraldes et al. [39] proposed a UAV-based situational awareness system called Person-Action-Locator (PAL). The PAL system is robust enough to automatically detect people and then recognize their actions in near-real-time.

Mliki et al. [40] introduced a two-stage methodology for recognizing human activities in UAV-captured videos. The first stage is responsible for generating human/non-human entities and human activity models using CNN. The second inference phase employs CNN-based human activity modeling to recognize human activities by using majority voting for the whole video sequence. Choi et al. [41] investigated the emerging problem of action recognition in drone videos using unsupervised and semi-supervised domain adaptation. The proposed method transfers the knowledge from source to target domain using video and an instance-based adaptation methodology. The authors also created a dataset of 5250 videos for evaluating their proposed method.

Barekatain et al. [42] presented the Okutama-Action dataset as a concurrent aerial view dataset for human action recognition and detection. This dataset contains 43 min of video with 12 different action categories. The Okutama-Action dataset poses a generic challenge due to the realistic condition of video acquisition that results in dynamic transition of actions, and challenges specific to single-image action detection, such as significant changes in scale and aspect ratio of the subject, abrupt camera movement, and side and top views of the subjects, as well as multi-labeled actors.

If we could devise a working pipeline for such a dataset, it will increase its suitability to process real-world situations.

3. Methodology

The offline pipeline of our proposed method for action detection and recognition in drone images is illustrated in Figure 1. In the first stage, a camera mounted either at 45° or 90° on a drone captures the outdoor scene. In the second stage, these drone-captured images are input into an anchor-free single-staged YoloV5 detector. A gradient boosting classifier accepts the output of the Yolo detector and detects and recognizes different actions. The final stage draws the bounding boxes and confidence score for each prediction. Moreover, we present a detailed diagram of YoloV5 architecture and gradient boosting classifier in Figure 2.

3.1. YoloV5 for Drone Action Detection

Two-stage object detectors have been a popular choice among the research community and include R-CNN series [43–45]. As compared to two-stage detectors, single-stage detectors are faster because they can simultaneously predict the bounding box and the class of objects. However, there is a compromise for the slight drop of accuracy for single-stage object detectors. The prominent single-stage detector may contain Yolo series [22–25], SSD [46], and RetinaNet [47].

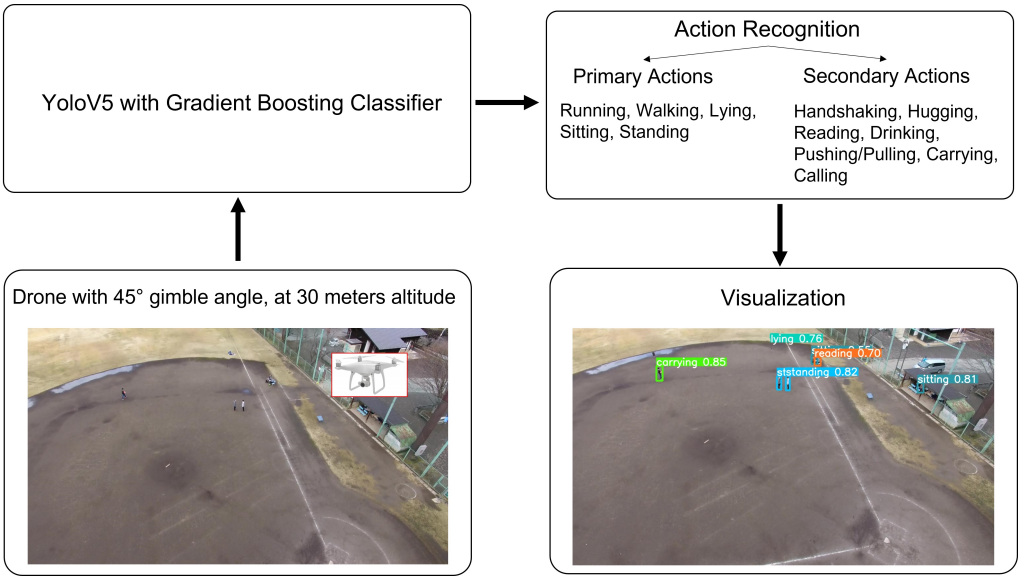

Figure 1. Overview of pipeline of work. First, the flying drone with a camera captures the images of the scene, which are input into the YoloV5 detector in an offline process. The gradient boosting classifier makes weight adjustments for difficult samples and re-trains the model for final action detection.

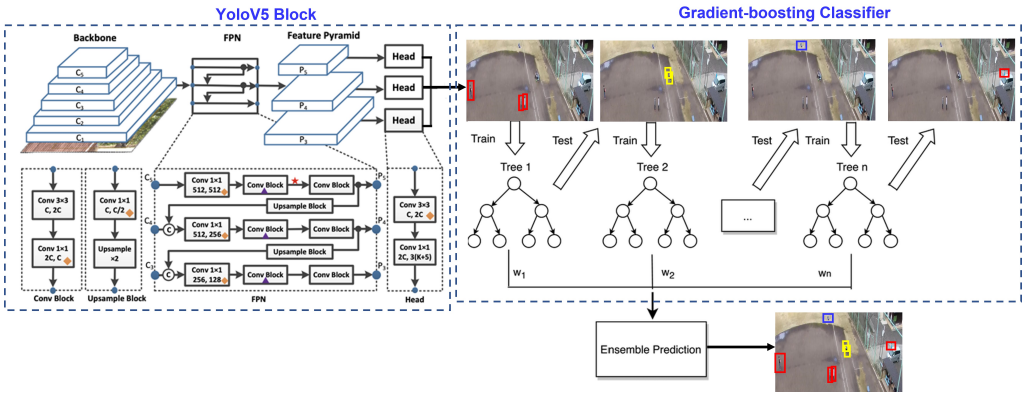

Figure 2. The left side of this figure explains the anchor-box free single-stage YoloV5 detector. YoloV5 architecture comprises of backbone architecture, neck layer as feature pyramid network (FPN), and prediction heads as bottleneck layers. The right side of this figure illustrates gradient boosting, which ensembles the predictions of the Yolo classifier for final action detection.

YoloV5 is one of the most famous detection algorithms due to its fast speed and high accuracy. YoloV5 divides the images into a grid system, where each cell in the grid is responsible for detecting objects within itself. This approach provides a specific advantage when multiple objects are involved, which is of particular importance to multi-person action recognition [48]. YoloV5 is the most recent model of the Yolo detection family and it contains the best architectures among the Yolo family. We interchangeably use the object detection and action detection terminology.

In a broader sense, YoloV5 comprises three main architectures: (i) backbone architecture, (ii) neck layer of feature pyramids, and (iii) bottleneck layer with prediction heads for action detection. The backbone architecture generally contains VGG [49], ResNet [50], DenseNet [51], MobileNet [52], EfficientNet [53], and CSPDarknet53. PANet aggregates features in the neck layer which includes several bottom-up paths and several top-down paths for upsampling and downsampling. The path-aggregation block generally consists of FPN [54], PANet [55], NAS-FPN [56], Bi-FPN [57], ASFF [58], and SFAM [59]. In the pipeline of work, the bottleneck layer of prediction heads may comprise the one-stage or two-stage detector.

There are five different models for YoloV5: YoloV5s, YoloV5m, YoloV5l, YoloV5x, and YoloV5n, which offer various options adapted to different computational and deployment constraints. We choose our baseline as YoloV5 which includes CSPDarknet53 as backbone, PANet as neck, and Yolo detection as head layer for a single-stage detector [22]. While training, we noticed that YoloV5x outperforms other models, i.e., YoloV5s, YoloV5m, YoloV5l, and YoloV5n. One clear disadvantage of YoloV5x is longer training time and larger model size. YoloV5n is the tiny version of YoloV5, which reduces one-third of the depth of YoloV5s and, therefore, results in 75% reduction in model parameters (7.5 M to 1.9 M), which make it an ideal choice for deploying on mobile devices and CPU-only machines. In YoloV5 architecture, there is other recent advancement, such as YoloV5-P5 and YoloV5-P6. YoloV5-P5 models have three output layers, P3, P4, and P5, with stride sizes of 8, 16, and 32 at an image size of 640 × 640. On the other hand, YOLOv5-P6 models have four output layers, P3, P4, P5, and P6, having stride sizes of 8, 16, 32, and 64, which were trained for image size of 1280 × 1280. YoloV5-P6 with stride size of 64 works well for detecting larger objects in high-resolution training images. By the time of study, YoloV7 was not released; however, YoloV5 as a single-stage detector is an appropriate choice for single-image action detection, since it brings different variants of YoloV5, thus offering a compromise between usability and performance for a UAV scenario.

3.2. Gradient Boosting Classifier

Previous work on the Okutama dataset explored various architectures based on a pipeline of entity recognition and temporal feature recognition using specialized variants of a classical CNN–LSTM approach. Barekatain et al. [42], more specifically, used SSD-classifer and ensembled two-streams as RGB and optical flow for action detection where both streams work in a complimentary fashion. Geraldes et al. [39] also ensembled the output of two architectures, POINet and ActivityNet. POINet detects the bounding box for each person in the scene using CNN; meanwhile, ActivityNet employs LSTM and computes the temporal features and action labels for each person.

The new architecture we introduce here adopts a similar philosophy, but aims at upgrading individual components for object identification and action classification, while also substituting boosting into the RNN component. While boosting is equally able to deal with temporal information, this might be a smaller issue with image-based action recognition. In addition, it offers more flexibility in terms of learning behavior, and is gaining popularity for action recognition tasks.

It should be noted that some previous work has explored a tight integration between CNN and boosting for vision tasks. However, by incorporating boosting weights into deep learning architecture [60], our use of boosting follows the previous pipeline processing philosophy with independent processing stages.

We have the same rationale for our work by using gradient boosting, which ensembles the output of different classifiers working in a complementary fashion. The Okutama-Action dataset contains images of camera angles with 45°and 90°, at altitude of 30 m and varying the distances between subjects and camera. The dataset images are also self-occluded and occluded by different objects in the scene. All these factors contribute to high variance for the Okutama-Action dataset.

Gradient boosting (GB) classifiers are good at mitigating high variance and high bias, which may cause overfitting and underfitting problems, respectively. Gradient boosting significantly reduces the high variance problem by decreasing the learning rate, because a higher learning rate more aggressively captures the variation among training samples [61]. On the other hand, gradient boosting controls high bias by increasing the boosting rounds, in which each round corresponds to the addition of a new decision tree [62,63]. The bias term consistently decreases as the number of boosting rounds is increased. Gradient boosting builds a mechanism for reducing the bias and variance in expected prediction error. Using gradient boosting, when a model is trained with low learning rate and higher number of boosting rounds, results in low bias and variance and correspondingly improves

the model performance. This is another strong motivation for using gradient boosting as a powerful general-purpose learning algorithm in our work.

Gradient boosting is conceived to be better than other machine learning algorithms, such as bagging and random forest decision trees, because it involves weight adjustment using decision tree predictions. GB also involves cross-validation, efficient handling of missing data, regularization to avoid overfitting, tree pruning, and paralyzed tree building, [64]. Gradient boosting fits the nonlinear (piecewise linear) decision boundary, while SVM always fits the linear boundaries even if the dataset is not linearly separable; therefore, GB brings more flexibility and, therefore, performs better than polynomial-based SVM approaches [65].

GB assigns different weights to different samples in such a manner that difficult-to-classify samples are weighted more whilst easily-classified samples receive less weight. Using gradient boosting, weak learners are sequentially added up to better classify the difficult samples. We employ log likelihood as loss function for the gradient boosting classifier. The gradient boosting is explained in Algorithm 1. The same concept of gradient boosting is also explained in Figure 3. During experimentation, we used 100 trees (boosting rounds) of maximum depth of 3 and a learning rate of 0.001. In addition, we used the Scikit-learn package for implementation of the gradient boosting algorithm [66].

Algorithm 1 Gradient Boosting Algorithm

Inputs: (i) $\{x_i, y_i\}$, (ii) loss function, $\mathcal{L}(y, F(x))$, (iii) No. of trees M
Procedure:
(1) Initialize Model with Constant Value

$$F_0(x) = argmin_\gamma(\sum_{i=1}^{n} \mathcal{L}(y, \gamma)) \tag{1}$$

(2) Iterate $m = 1$ to M
(i) Compute Pseudo-residuals

$$\gamma_{im} = \left[\frac{\partial \mathcal{L}(y_i F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}^{i=1,\ldots,n} \tag{2}$$

(ii) Fit a Base-Learner $h_m(x)$, input $\{(x_i, \gamma_{im})\}$
(iii)

$$\gamma_m = argmin_\gamma \sum_{i=1}^{n} \mathcal{L}(y_i, F_{m-1}(x_i) + fl) \tag{3}$$

(iv) Update Model,

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{4}$$

3.3. Description of Architecture

In our work, gradient boosting is implemented by increasing the frequency of difficult samples in order to better learn from these samples. Gradient boosting adjusts the weights based on the previous decision tree predictions. The residual error is computed and added to the initial values and then fine-tuned, so that the final prediction approaches closer to the ground-truth values.

The main challenges that occur during action detection in the Okutama-Action dataset are (i) human subjects are not well-exposed to the camera, as are subjects in images taken on the ground, (ii) subjects are of very small size, as compared to the whole image, and (iii) subjects occur from different camera viewpoints.

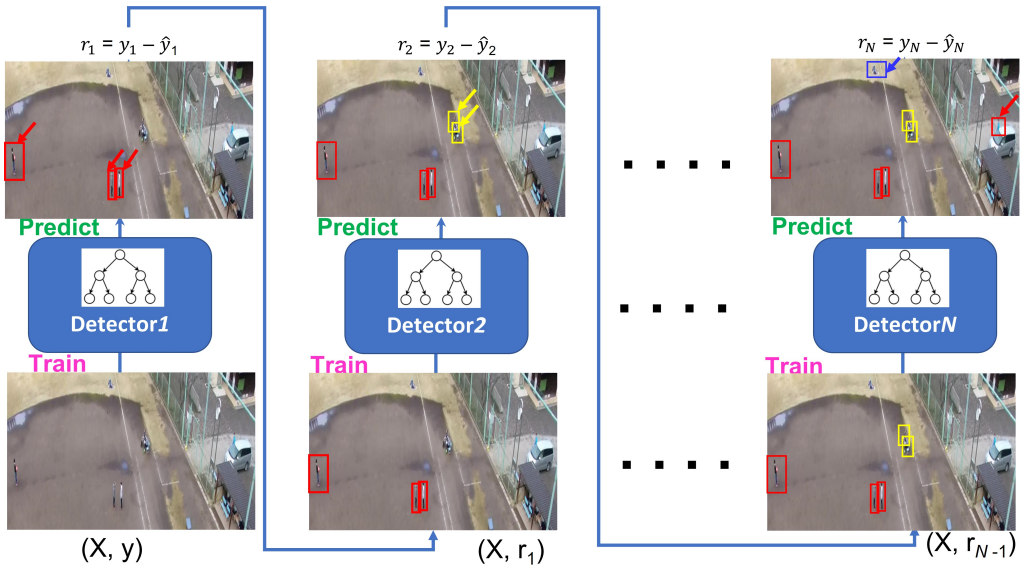

Figure 3. This diagram explains the gradient boosting classifier for learning and retraining on difficult instances. The boosting technique involves weight adjustment based on the previous decision tree prediction. The output features of YoloDector are encoded as feature vectors for input into the boosting classifier. The regularization is introduced for this model using shallow boosting trees.

3.3.1. Input

Following the common practice of object detection, the input image is expected to have a size of $H \times W \times C$, where H and W correspond to image height and width, whereas C represents the number of channels. In our case, the number of channels is set to three (RGB).

3.3.2. Backbone

Multi-scale features are extracted using either ResNet [50], ResNeXt [67], or DenseNet [51] as encoder. These features are made compatible and input into a feature pyramid network (FPN). Feature map extraction at different stages $1 \sim N$ is represented correspondingly as $C_1 \sim C_N$.

3.3.3. Feature Pyramid Network (FPN)

The feature pyramid network helps to detect objects at different scales. The lower-level layers in FPN have higher resolution with fewer semantic details, whilst higher-level layers have lower resolution with stronger semantic meaning. The residual connections fuse the features between different layers and thus facilitate smaller objects' detection.

3.3.4. Detection Heads

The prediction heads carry out per-pixel prediction, where a prediction is output from three heads of similar architectures, i.e., a 2D convolution $\rightarrow$ a group normalization $\rightarrow$ a rectified linear unit (ReLU). The three outputs of these heads are centerness head, class prediction head, and box-regression head, as shown in Figure 2.

3.3.5. Network Settings

In our experiments, we used CSPDarknet53 and ResNet-50 as backbone architectures. We empirically choose all the network parameters, e.g., initial base learning rate was set to 0.001 with a weight decay of 0.0001 and a momentum of 0.9. The image mosaic parameter was set to high to take advantage of data augmentation. We ran our experiments for 200 epochs with batch size of 32. We implemented our method in the Pytorch platform to run these experiments [68], while the Scikit-learn library was used for implementing gradient boosting algorithm with 100 trees and depth-level of 3. Tesla P100-PCIE with cuda-10.2 was the hardware machine for this implementation.

4. Experiments

4.1. Dataset Description

The Okutama-Action dataset [42] consists of a 43-minute-long annotated video sequence of 12 different outdoor action categories with about 77,000 image frames. This is a challenging dataset because it includes abrupt camera motion, dynamic transition of actions, scale variation due to near–far movement of drone, and variation in aspect ratio of actor while performing different actions. Some example images of the Okutama-Action dataset are shown in Figure 4. Okutama-Action videos were captured using a DJI Phantom 4 UAV at a baseball field in Okutama, Japan. These twelve actions of the Okutama-Action dataset are grouped into three types:

- Human-to-human interaction (handshaking, hugging).
- Human-to-object interaction (reading, drinking, pushing/pulling, carrying, calling).
- No interaction (running, walking, lying, sitting, standing).

For capturing the action videos, the UAV was operated at altitude range of 30 m to 45 m and the gimbal angle was set as either 45°or 90°.

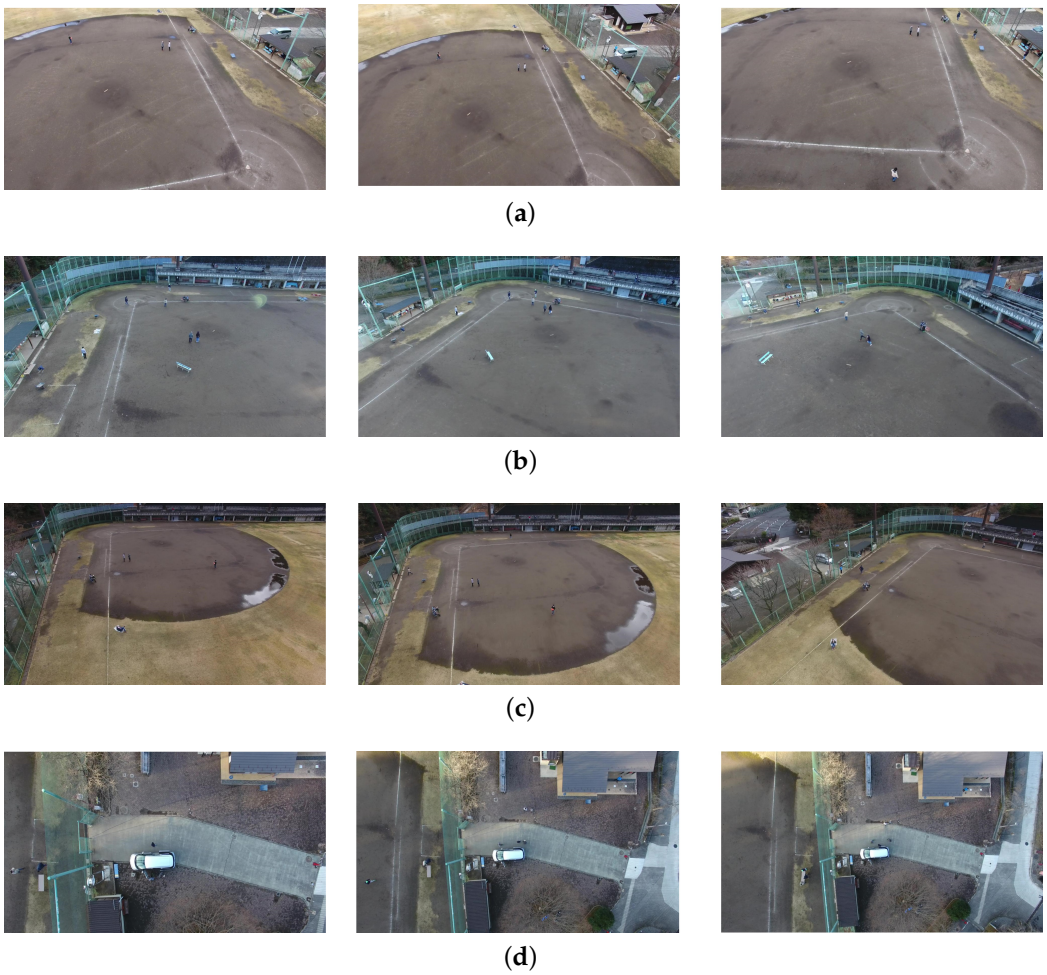

Figure 4. Some sample images from the Okutama-Action dataset. (**a**) Drone1, morning. Frame 1, 225, 2220 in video 1.1.1; (**b**) Drone1, noon. Frame 2, 270, 685 in video 1.2.2; (**c**) Drone2, morning. Frame 1, 200, 870 in video 2.1.1; (**d**) Drone2, noon. Frame 175, 325, 700 in video 2.2.5. In this setting, camera tilt is near to 90 degrees.

The Okutama-Action dataset covers two different scenarios for data collection in the morning and noon settings for incorporating different lighting conditions (sunny and cloudy). Additionally, this dataset was captured using two different drones operated

by two different pilots with different speeds and maneuvers. For some of the videos, metadata was provided for altitude, speed, and gimbal angle. During data collection, a 4K high-resolution UAV-mounted camera was operating at 30 FPS.

4.2. Ablation Study

We evaluate the performance of our proposed models using precision, mAP@50%IoU, recall, and F1-score.

4.2.1. Baseline Model

We define YoloV5s and YoloV5s6 as our baseline models. YoloV5s is termed as YoloV5-P5 and has three output layers with stride sizes of 8, 16, and 32 for image size 640 × 640, whilst YoloV5s6 is termed as YoloV5-P6, having four output layers of stride 8, 16, 32, and 64 with image size of 1280 × 1280. We report the performance using YoloV5s and YoloV5s6 in Table 1, whilst a performance chart for each category of actions, primary and secondary actions, is also reported in Table 2. Primary actions are performed by the single subject without any interaction with other subject or object (e.g., run, walk, lying, sit, stand). Meanwhile, secondary actions may involve interaction with some object (e.g., read, call, drink, actions require book, phone, bottle) or interaction with some other subject (e.g., handshake, hug).

Table 1. Comparison among YoloV5s-P5 and YoloV5s6-P6 as baseline architectures. mAP is computed in (%) at IoU = 0.5:0.05:0.95, while mAP@0.5 is computed at 50% IoU.

Method	mAP	mAP@0.5	Recall	F-Score
YoloV5s	64.6	62.9	65.8	65.2
YoloV5s6	68.7	63.4	61.4	64.8

Table 2. Performance evaluation for each individual action category for YoloV5s6 architecture.

	Method	mAP	mAP@0.5	Recall	F-Score
Primary Actions	run	50.9	46.6	49.4	50.1
	walk	67.8	69.8	70.5	69.1
	lying	69.8	83.1	79.7	74.4
	sit	70.2	66.4	67.4	68.8
	stand	67.8	67.6	69.4	68.6
	Avg. Metric	65.3	66.7	67.3	66.3
Secondary Actions	handshake	58.1	56.8	55.6	56.8
	hug	66.8	59.4	51.8	58.3
	read	70.7	55.2	50.6	59.0
	drink	69.5	47.7	37.8	49.0
	push-pull	74.0	65.5	63.1	68.1
	carry	76.5	78.0	78.3	77.4
	call	77.3	60.6	59.3	67.1
	Avg. Metric	70.4	60.4	56.6	62.7

4.2.2. Comparison among YoloV5 Architectures

We conducted our experiment with different YoloV5 architectures and report our research findings in Table 3. We exercised different architectures of YoloV5-P5, such as YoloV5s, YoloV5n, YoloV5m, YoloV5l, and YoloV5x, and YoloV5-P6, such as YoloV5s6, YoloV5n6, YoloV5m6, YoloV5l6, and YoloV5x6. YoloV5n and YoloV5n6 are the smallest in size with a slight compromise in accuracy which makes them an ideal choice for deploying in drone applications where computational resources are always a constraint. Mainly, this experiment works offline but devises small sizes of Yolo architectures to work online, as shown in Table 3. The YoloV5-P6/64 output layer performs well for detecting larger

objects in high-resolution images. YoloV5x and YoloV5x6 architectures are the largest in size and performed better than other architectures.

Table 3. Comparison among different YoloV5 architectures. The first main comparison is between YoloV5-P5 and YoloV5-P6. The parameters are measured in millions (M), average precision is measured in %, training time is measured in hours, and model size is measured in MB. In the names of YoloV5, the subscripts "s", "n", "m", "l", and "x" refer to small, nano, medium, large, and extra-large network architectures.

Method	Layers	Parameters	GFlops	mAP	mAP@0.5	Recall	F-Score	Time	Size
YoloV5s	213	7.04	15.9	64.6	62.9	65.8	65.2	7.1	14.4
YoloV5x	444	86.2	204.2	74.5	49.7	51.9	61.2	32.8	173.2
YoloV5n	213	1.77	4.2	65.2	59.0	59.8	62.4	5.6	3.9
YoloV5m	290	20.9	48.1	72.3	67.8	67.3	69.7	12.7	71.2
YoloV5l	367	46.2	108	74.2	68.6	67.0	70.4	17.2	92.9
YoloV5s6	280	12.3	16.3	68.7	63.4	61.4	64.8	7.3	25.1
YoloV5x6	574	140.1	208.3	**75.4**	68.9	67.4	71.0	35.4	281.1
YoloV5n6	280	3.11	4.3	61.5	56.7	55.0	58.1	5.7	6.6
YoloV5m6	378	35.3	49.1	70.5	64.7	63.3	66.7	12.2	71.2
YoloV5l6	476	76.2	110.2	73.5	68.0	65.8	69.4	18	153.2

4.3. Comparison with State-of-the-Art Methods

We present a comparison of our proposed method with other state-of-the-art methods in Table 4. Barekatain et al. [42] used SSD for detecting actions in Okutama-Action dataset with image size of 512 × 512. The authors ran their experiments on RGB and optical flow streams and then combined the results of both streams for better accuracy. During experimentation, it was realized that the SSD model performed best when the camera angle was 45 degrees. It was also noticed that strongly related temporal actions, e.g., the *running* action, resulted in lower recognition accuracy because SSD sequentially performed detection in a frame-by-frame manner. Soleimani et al. [69] proposed a two-stage architecture for identifying the action categories in the Okutama-Action dataset. The first stage relies on SSD for finding objects of interest, whereas the second stage uses another CNN to learn the latent sub-space for associating aerial imagery and action labels. The main difference between SSD and Yolo architectures lies in the handling of bounding boxes, as SSD treats each bounding box prediction as a regression problem. Yolo architecture, on the other hand, computes non-maximum suppression and thus retains the final bounding box. Yolo methods are slightly better than SSD in detecting smaller objects. This is exactly the situation in our scenario due to the distance between flying drones and human subjects on the ground. This proposed architecture resulted in significant increase of 28.3% in mAP for 50% IoU. Geraldes et al. [39] devised POINet (Position and Orientation Invariant Neural Networks) by employing MobilenetV2 [70] and reported the action detection performance separately for primary and secondary actions. The performance of our proposed method in terms of primary and secondary actions is also better than [39]. In Table 4, we explicitly mentioned the type of classifier for each methodology and realized that the gradient boosting classifier with Yolo action detector shows better performance than the fully-connected feed-forward classifier with detectors. The qualitative detection results of our proposed methodology are presented in Figure 5.

Table 4. Comparison of our proposed method with other state-of-the-art methods for the Okutama-Action dataset. We report the YoloV5x6 as the best performing Yolo architecture. The performance is measured in mean average precision (%) for 50% IoU. PA stands for primary actions, while SA stands for secondary actions. FFNN represents fully-connected feed-forward neural networks, while GB denotes gradient boosting.

Method	Image/Video	Architecture	$mAP_{0.5}$	PA	SA
SSD512 [42]	Image	FFNN	15.4	-	-
SSD960 [42]	Image	FFNN	18.8	-	-
SSD+CNN [69]	Image	FFNN	28.3	-	-
POINet-Benchmark [39]	Video	LSTM	-	38.8	44.6
POINet-SFT [39]	Video	CNN–LSTM	-	26.5	17.9
Proposed Method	Image	Yolov5+GB	**75.4**	**71.8**	**77.9**

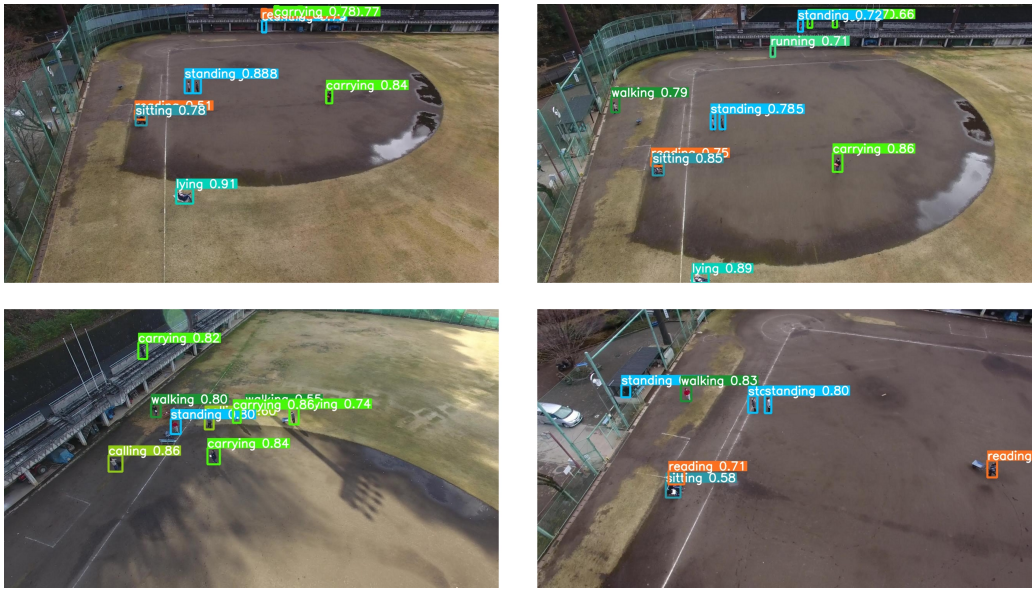

Figure 5. Inference results for action detection in drone images. Each color bounding box corresponds to a separate action category.

5. Conclusions

Deep learning is a method of choice for drone action recognition and detection due to its high performance and easy deployment. In this paper, we propose a Yolo-based framework with gradient boosting for detecting and recognizing twelve different actions. We investigated the performance of our model on the Okutama-Action dataset, where it outperformed other methods for individual actions, primary and secondary actions. We evaluated the performance of our method for different settings, such as morning or noon, and varying gimbal angle, of 45 degree or 90 degree, for an altitude of 30 m. We also evaluated the performance of different variants of YoloV5. Since YoloV5 is efficient for detecting multiple objects in an image, and the Okutama-Action dataset contains concurrent actions in a frame, our proposed method achieved better results for single-image action detection than other methods in the literature that may include explicit temporal information, e.g., LSTM or 3D-CNN.

The main limitation of our work is that the performance of our algorithm may degrade if the gimbal angle becomes closer to 90°or if the drone flies above 30 m. Moreover, the speed of the drone may also negatively affect the performance of our algorithm due to motion blurring. In addition, low lighting conditions of early morning–evening and extreme weather conditions of raining, clouds, or snowing may severely degrade the performance of our method.

YoloV7 shows good results for pose estimation and is beneficial for single-image object detection; therefore, in future work, it is possible that YoloV7 can be a good choice for action detection. As an alternative to YoloV7, self-attention transformers and their variants, e.g., ViT or ScaleViT, are good candidates that could replace the pipeline of YoloV5 and gradient boosting for action detection.

Author Contributions: H.P. and Y.M. conceptualized the idea of action recognition and detection in drone images for the Okutama-Action dataset. T.A. made an early investigation of Yolo models for the Okutama dataset and devised a methodology in collaboration with H.P., M.C., Y.M. and T.A. carried out experiments and validated the results. All authors contributed to the preparation of the manuscript. Finally, the funding for this project was acquired by Y.M. and H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by a donation from Matsuo Institute, Japan.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Girish, D.; Singh, V.; Ralescu, A. Understanding action recognition in still images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020.
2. Eweiwi, A.; Cheema, M.S.; Bauckhage, C.; Gall, J. Efficient pose-based action recognition. In *Asian Conference on Computer Vision*; Springer: Cham, Switzerland, 2014; pp. 428–443.
3. Wang, C.; Wang, Y.; Yuille, A.L. An approach to pose-based action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 915–922.
4. Agahian, S.; Negin, F.; Köse, C. An efficient human action recognition framework with pose-based spatiotemporal features. *Eng. Sci. Technol. Int. J.* **2020**, *23*, 196–203. [CrossRef]
5. Zhao, Z.; Ma, H.; You, S. Single image action recognition using semantic body part actions. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 3391–3399.
6. Sreela, S.R.; Idicula, S.M. Action recognition in still images using residual neural network features. *Procedia Comput. Sci.* **2018**, *143*, 563–569. [CrossRef]
7. Liu, L.; Tan, R.T.; You, S. Loss guided activation for action recognition in still images. In *Asian Conference on Computer Vision*; Springer: Cham, Switzerland, 2018; pp. 152–167.
8. Wu, D.; Sharma, N.; Blumenstein, M. Recent advances in video-based human action recognition using deep learning: A review. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 2865–2872.
9. Pareek, P.; Thakkar, A. A survey on video-based human action recognition: Recent updates, datasets, challenges, and applications. *Artif. Intell. Rev.* **2021**, *54*, 2259–2322. [CrossRef]
10. Pham, H.H.; Khoudour, L.; Crouzil, A.; Zegers, P.; Velastin, S.A. Video-based human action recognition using deep learning: A review. *arXiv* **2022**, arXiv:2208.03775.
11. Rohrbach, M.; Amin, S.; Andriluka, M.; Schiele, B. A database for fine grained activity detection of cooking activities. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012.
12. Singh, B.; Marks, T.K.; Jones, M.; Tuzel, O.; Shao, M. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
13. Yeung, S.; Russakovsky, O.; Mori, G.; Fei-Fei, L. End-to-end learning of action detection from frame glimpses in videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016.
14. Zhang, D., Sr.; Shao, Y.; Mei, Y.; Chu, H.; Zhang, X.; Zhan, H.; Rao, Y. Using YOLO-based pedestrian detection for monitoring UAV. In Proceedings of the Tenth International Conference on Graphics and Image Processing, Chengdu, China, 12–14 December 2018.
15. Yang, Z.; Huang, Z.; Yang, Y.; Yang, F.; Yin, Z. Accurate specified-pedestrian tracking from unmanned aerial vehicles. In Proceeding of the IEEE 18th International Conference on Communication Technology, Chongqing, China, 8–11 October 2018.
16. Liu, M.; Wang, X.; Zhou, A.; Fu, X.; Ma, Y.; Piao, C. Uav-yolo: Small object detection on unmanned aerial vehicle perspective. *Sensors* **2020**, *20*, 2238. [CrossRef]

17. Mittal, P.; Singh, R.; Sharma, A. Deep learning-based object detection in low-altitude UAV datasets: A survey. *Image Vis. Comput.* **2020**, *104*, 1040–1046. [CrossRef]
18. Shinde, S.; Kothari, A.; Gupta, V. YOLO based human action recognition and localization. *Procedia Comput. Sci.* **2018**, *133*, 831–838. [CrossRef]
19. Wolf, C.; Lombardi, E.; Mille, J.; Celiktutan, O.; Jiu, M.; Dogan, E.; Eren, G.; Baccouche, M.; Dellandréa, E.; Sankur, B. Evaluation of video activity localizations integrating quality and quantity measurements. *Comput. Vis. Image Underst.* **2014**, *127*, 14–30. [CrossRef]
20. Jung, H.K.; Choi, G.S. Improved YoloV5: Efficient Object Detection Using Drone Images under Various Conditions. *Appl. Sci.* **2022**, *12*, 7255. [CrossRef]
21. Caputo, S.; Castellano, G.; Greco, F.; Mencar, C.; Petti, N.; Vessio, G. Human Detection in Drone Images Using YOLO for Search-and-Rescue Operations. In *International Conference of the Italian Association for Artificial Intelligence*; Springer: Cham, Switzerland, 2022; pp. 326–337.
22. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
23. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
24. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
25. Bochkovskiy, A.; Wang, C.; Liao, H.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
26. Ali, S.; Shah, M. Human action recognition in videos using kinematic features and multiple instance learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *32*, 288–303. [CrossRef] [PubMed]
27. Cao, L.; Liu, Z.; Huang, T.S. Cross-dataset action detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010.
28. Wang, H.; Schmid, C. Action recognition with improved trajectories. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, NSW, Australia, 1–8 December 2013.
29. Sultani, W.; Saleemi, I. Human action recognition across datasets by foreground-weighted histogram decomposition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014.
30. Carreira, J.; Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
31. Zhou, X.; Liu, S.; Pavlakos, G.; Kumar, V.; Daniilidis, K. Human motion capture using a drone. In Proceedings of the IEEE International Conference on Robotics and Automation, Brisbane, QLD, Australia, 21–25 May 2018.
32. Ahmad, T.; Jin, L.; Feng, J.; Tang, G. Human action recognition in unconstrained trimmed videos using residual attention network and joints path signature. *J. IEEE Access* **2019**, *7*, 121212–121222. [CrossRef]
33. Ahmad, T.; Jin, L.; Lin, L.; Tang, G. Skeleton-based action recognition using sparse spatio-temporal GCN with edge effective resistance. *Neurocomputing* **2021**, *423*, 389–398. [CrossRef]
34. Ahmad, T.; Jin, L.; Zhang, X.; Lai, S.; Tang, G.; Lin, L. Graph Convolutional Neural Network for Human Action Recognition: A Comprehensive Survey. *IEEE Trans. Artif. Intell.* **2021**, *2*, 128–145. [CrossRef]
35. Sultani, W.; Shah, M. Human action recognition in drone videos using a few aerial training examples. *Comput. Vis. Image Underst.* **2021**, *206*, 103186. [CrossRef]
36. Ucf-Arg Data Set. Available online: Https://www.crcv.ucf.edu/data/UCF-ARG.php (accessed on 5 May 2022).
37. Perera, A.; Wei L.; Yee; Chahl, J. UAV-GESTURE: A dataset for UAV control and gesture recognition. In Proceedings of the European Conference on Computer Vision Workshops, Munich, Germany, 8–14 September 2018.
38. Ding, M.; Li, N.; Song, Z.; Zhang, R.; Zhang, X.; Zhou, H. A Lightweight Action Recognition Method for Unmanned-Aerial-Vehicle Video. In Proceedings of the IEEE 3rd International Conference on Electronics and Communication Engineering, Xi'an, China, 14-16 December 2020.
39. Geraldes, R.; Goncalves, A.; Lai, T.; Villerabel, M.; Deng, W.; Salta, A.; Nakayama, K.; Matsuo, Y.; Prendinger, H. UAV-based situational awareness system using deep learning. *J. IEEE Access* **2019**, *7*, 122583–122594. [CrossRef]
40. Mliki, H.; Bouhlel, F.; Hammami, M. Human activity recognition from UAV-captured video sequences. *Pattern Recognit.* **2020**, *100*, 107140. [CrossRef]
41. Choi, J.; Sharma, G.; Chandraker, M.; Huang, J. Unsupervised and semi-supervised domain adaptation for action recognition from drones. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision, Snowmass, CO, USA, 1–5 March 2020.
42. Barekatain, M.; Martí, M.; Shih, H.; Murray, S.; Nakayama, K.; Matsuo, Y.; Prendinger, H. Okutama-Action: An aerial view video dataset for concurrent human action detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Honolulu, HI, USA, 21–26 July 2017.
43. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014.
44. Girshick, R. Fast r-cnn. *arXiv* **2015**, arXiv:1504.08083.
45. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Conference Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015.

46. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016.
47. Lin, T.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.
48. YoloV5 Documentation. Available online: Https://docs.ultralytics.com/ (accessed on 5 June 2022).
49. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
50. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
51. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
52. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
53. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019.
54. Lin, T.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
55. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
56. Ghiasi, G.; Lin, T.; Le, Q.V. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2019.
57. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020.
58. Liu, S.; Huang, D.; Wang, Y. Learning spatial fusion for single-shot object detection. *arXiv* **2019**, arXiv:1911.09516.
59. Zhao, Q.; Sheng, T.; Wang, Y.; Tang, Z.; Chen, Y.; Cai, L.; Ling, H. M2det: A single-shot object detector based on multi-level feature pyramid network. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.
60. Moghimi, M.; Belongie, S.J.; Saberian, M.J.; Yang, J.; Vasconcelos, N.; Li, L.J. Boosted convolutional neural networks. In Proceedings of British Machine Vision Conference, York, UK, 19–22 September 2016.
61. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
62. Friedman, J.H. Stochastic gradient boosting. *Comput. Stat. Data Anal.* **2002**, *38*, 367–378. [CrossRef]
63. Chen, T.; Guestrin, C. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016.
64. Dave, P.; Chandarana, A.; Goel, P.; Ganatra, A. An amalgamation of YoloV4 and XGBoost for next-gen smart traffic management system. *PeerJ Comput. Sci.* **2021**, *7*, e586. [CrossRef]
65. Caruana, R.; Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 161–168.
66. sklearn.ensemble.GradientBoostingClassifier. Available online: Https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html (accessed on 15 March 2022).
67. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
68. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic Differentiation in Pytorch. Available online: Https://pytorch.org/ (accessed on 10 January 2022).
69. Soleimani, A.; Nasrabadi, N.M. Convolutional neural networks for aerial multi-label pedestrian detection. In Proceedings of the IEEE 21st International Conference on Information Fusion, Cambridge, UK, 10–13 July 2018.
70. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.