Benchmarking Object Detection Models with Mummy Nuts Datasets*

Darren Ng[1], Colin Schmierer[1], Andrew Lin[1], Zeyu Liu[2], Falin Yu[3], Shawn Newsam[1], Reza Ehsani[1], and Xiaoyi Lu[1]

[1] University of California Merced, Merced, USA
{dng350,cschmierer,alin85,snewsam,rehsani,xiaoyi.lu}@ucmerced.edu
[2] Valley Christian High School, San Jose, USA
bennyzeyuliu@gmail.com
[3] Santa Margarita Catholic High School, Rancho Santa Margarita, USA
falin.yu@smhsstudents.org

Abstract. Agriculture presents challenges in automation, especially so in vision systems. Varying lighting conditions, sporadic diversity, and large amounts of noise create difficulty in detecting target objects. Our Mummy Nuts datasets present these challenges in tiny scale, camouflaged, dark, or even hidden target objects. However, the most recent advancements in Convolutional Neural Networks (CNN) in the object detection task have become increasingly accurate and robust. As there are many different CNNs, selecting which CNN will perform the best may become challenging. This paper proposes a two-dimensional benchmarking methodology to evaluate five popular CNN models (YOLOv3, YOLOv5, CenterNet, Faster R-CNN, and MobileNet SSD) on two NVIDIA GPUs (Tesla T4 and A100). Our benchmarking methodology evaluates accuracy across all models and performance among models on each GPU. Our results show the benefits of selecting models using our Augmented dataset over the Original dataset. CNN Models overall see an increase in recall values during inference by an average of 2.77X (with the highest increase as YOLOv3 by 6.5X). For performance, over both Original and Augmented datasets, the model training time reduces by an average of 4.45X when using A100 over Tesla T4.

Keywords: Benchmarking · Object Detection · Mummy Nuts

1 Introduction

Object detection using Convolutional Neural Networks (CNN) has become increasingly popular. CNNs appear in an ever-growing field of applications and are crucial in precision agriculture automation tasks such as yield estimation, disease detection, and robotic harvesting. Agricultural object detection proves to be a complex engineering problem due to many unseen variables that come as a trait of agriculture.

* This work was supported in part by the NSF research grants CCF #2132049, EEC #1941529, and a COR grant from University of California, Merced.

Pest control is a challenging problem in agriculture and, if not performed correctly, will damage farmer crop yield significantly. Almond growers are a prime target as orchards may take permanent damage from pests known as Navel Orangeworm (NOW), which nestle and feed on off-season almond nuts (Mummy Nuts). Growers must adequately monitor and track NOW disease to prevent spreading [1]. Furthermore, Mummy Nuts have diverse appearances and prove difficult for manual inspection. Due to this, Mummy Nut object detection has been an under-researched topic in precision agriculture.

CNNs can be applied to an environment that may contain erratic behaviors. For Mummy Nut detections, this is in the endless variations of the appearance in the target object. Furthermore, agriculture relies heavily on the season and restricts necessary data collection to a particular time frame each year. In the case of Mummy Nuts, image data can only be taken during the winter. This window would limit the data available to train CNNs.

In CNN training, the most critical component would be the dataset used. A robust CNN model must be trained with a carefully curated dataset. High-quality datasets should include thousands of images, with each class containing images of similar features, respectively. For example, the Microsoft Common Objects in COntext (MS COCO) dataset contains 2,500,000 labeled instances in 328,000 images standing among the richest datasets [2]. Other popular, large-scale datasets include ImageNet [3], Pascal VOC [4], SUN Database [5], and Pedestrian Database [6]. Due to the necessity of large datasets, there is little work on training CNNs with insufficient data.

For Mummy Nut detection problems, we face numerous issues with dataset curation. Most existing datasets are easy to annotate and can be considered a simple task for human workflow. MS COCO deployed an annotation pipeline to richly annotate each image using Amazon Mechanical Turk workers [2]. This workflow assumes each image contains objects easily identifiable by non-expert annotators. However, our proposed datasets are difficult to annotate, showing varying results in recall values between each annotator. Multiple expert annotators reviewed each image to ensure the highest annotation recall and precision. A large amount of noise contributes to the complication of annotation and detection of target objects within an image, though it allows for model robustness. Lastly, there is a significant underrepresentation of the variety of nuts. Each nut classifies for a different difficulty class (e.g., Noisy, Dark, Tiny, etc.) we assigned that will tell us how complex a particular detection may be. Difficulty classes are not equally represented in the training set, which may decrease the recall value per underrepresented class.

Another factor that predominantly affects CNN performance is the type of object detection model used. Multiple popular CNN models are in use, and each has a very different structure in training and inference computations. Lately, quick one-stage detectors have been utilized in systems and show high accuracy, such as You Only Look Once (YOLO) [7], Regions with CNN features (R-CNN) [8], Fast R-CNN [9], and Faster R-CNN [10]. These models create spaced boxes across the input image called anchor boxes, each individually responsible

for determining if a target object is within that box. Models like CenterNet [11] take a different approach, using heatmaps to determine peak points where a target object may appear. MobileNet [12] prioritizes a smaller network with significantly lower parameters to run well on mobile devices.

With the Mummy Nuts dataset, deciding which CNN model may provide the best results for each possible vision solution can become challenging. Only a few benchmark studies in the community can aid the selection of CNN models for the Mummy Nuts problem. Each model provides varying results depending on the proposed solution. We design tools to richly annotate and enhance our dataset to tackle its challenges. Our Data Augmentation tool expands the original dataset by performing image transformations on each annotation, artificially generating more diversity in our dataset. Our Difficulty Classification Annotator (DCA) tool is a notation tool that marks each annotation in our dataset with flags. These flags provide much more informative annotations denoting what difficulty class each annotation falls within.

To help guide the community in selecting the proper CNN model, this paper proposes a two-dimensional benchmarking methodology (e.g., accuracy and performance) on different CNN models. Each image is large in resolution and complexity, so performance latency during training and inference is evaluated on different hardware accelerations.

We deploy five different CNN models (e.g., YOLOv3, YOLOv5, CenterNet, Faster R-CNN, and MobileNet SSD) on two NVIDIA GPUs (e.g., A100 and Tesla T4). Throughout our experiments, our significant observations include: 1) All models saw an increase in recall value by an average of 2.77X (@IoU50) when using the Augmented dataset compared to the Original dataset; 2) The recall value of YOLOv3 increases by 6.5X (@IoU50) when using the Augmented dataset compared to the Original dataset; 3) All models except for MobileNet SSD suffer localization precision issues when using the Original dataset; 4) CenterNet, Faster R-CNN, and MobileNet SSD all decrease precision by an average of 10.53X (@IoU50) using Augmented over Original; 5) For performance, over both Original and Augmented datasets, the model training time reduces by an average of 4.45X when using A100 over Tesla T4; 6) Faster R-CNN sees a considerable speed up when running computations on A100 over Tesla T4 by about 5.76X; and 7) YOLO models have the fastest overall inference speed.

This paper makes the following contributions: 1) We create real-world datasets for the Mummy Nuts problem, which can help the community to perform in-depth interdisciplinary research between computer science and precision agriculture areas; 2) We design easy-to-use benchmarking tools (e.g., Data Augmentation Tool, DCA Tool, etc.) and integrate representative deep learning models into tools for agriculture scientists to investigate the Mummy Nuts problems conveniently; and 3) Through our benchmarking methodology and results, we provide guidance on which models may be ideal for different proposed solutions.

2 Background and Motivation

Mummy Nuts and NOW Disease: Leaving Mummy Nuts on trees can attract Navel Orangeworm pests (NOW) that will feed off of the nutmeat, leaving behind aflatoxins, a food safety contaminate linked to cancers [1]. Due to this, managing the spread of NOW is crucial and requires several steps. Firstly, monitoring the number of NOW pests and their mating growth is vital for planning the timing of pesticide applications. NOW capture and mating disruption methods also effectively reduce the population growth throughout NOW generations. Lastly, winter sanitation by removing remaining Mummy Nuts from orchards helps to remove the attractive food source for NOW pests. We tackle an early experience in automating the monitoring process of the spread. By detecting Mummy Nuts, we can monitor the food supply of NOW pests, produce a concentration heatmap of Mummy Nuts, and target and sanitize areas with high concentrations. We observe the task of object detection to implement this monitoring system. However, we notice that CNN model selection is essential as not all models can provide similar results. The dataset provided to each model also must be rich in features and carefully curated.

Object Detection and Inference: Object detection is an essential application for CNNs, and many models can perform this differently. It takes both localization and classification tasks into account by creating a spatially aware bounding box labeled with the name of the class detected. Training a CNN requires millions of parameter updates, equating to a large number of computations. Inference is the compression of those millions of parameters into a model that can quickly run and make detections. Thus CNN training requires high throughput while inference requires low latency [13]. As newer versions of object detection systems have been built, each has continued to speed up the inference process using different detection methods. There are multiple methods currently in use to perform detections. One popular method is employing anchor boxes and region proposals. This method creates anchor boxes spaced throughout input images in which each anchor box is responsible for making a detection. Other models eliminate the need for anchor boxes by converting the input image into a heatmap where the maxima are assumed to be a detection. We will go into more detail about each model in Section 3.

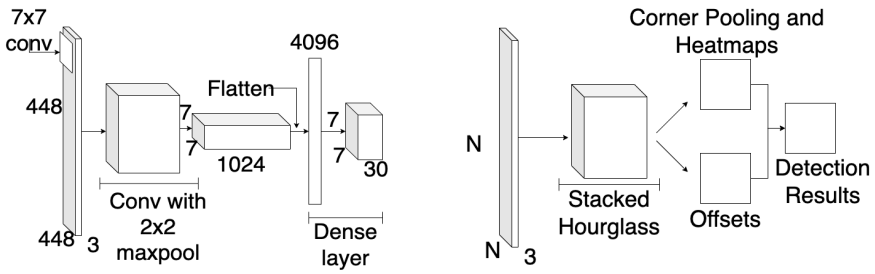
3 Requirements of Detecting Mummy Nuts with CNNs

This section presents an overview of selected object detection CNNs. We then review the requirements of detecting Mummy Nuts on our datasets with CNNs.

3.1 Overview of Object Detection Models

You Only Look Once (YOLO) is based on spatially set anchor boxes on input images that can observe if an object classification is within the boxes [7]. These anchor boxes will then produce a large number of bounding boxes around

each instance of the object that is detected. The algorithm uses Non-Maximum Suppression (NMS) to reduce bounding boxes within the Intersection over Union (IoU) of another box with a higher confidence rate than the other boxes. In Figure 1a, NMS is performed during the dense layer after making detections.



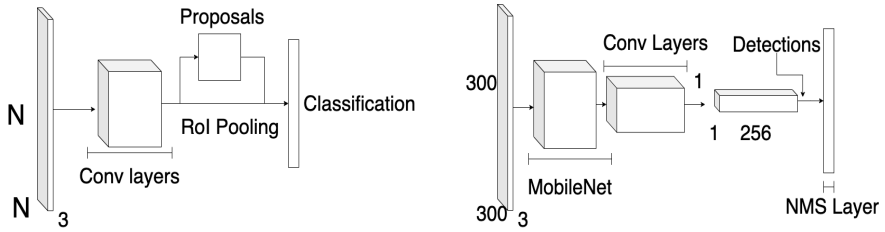
(a) YOLO (b) CenterNet

Fig. 1: (a) shows the design of YOLO, which contains 24 convolutional layers with two fully connected layers at the end. (b) shows the design of CenterNet, which uses an hourglass backbone where the convolutional layers are structured with input and output layers at the largest sizes and the middle at the smallest, mimicking an hourglass shape.

CenterNet focuses on the center points of each box detection rather than the box dimensions. A heatmap of the detection is created, and the maxima of the heatmap produce the detected center points [11]. CenterNet was proposed to eliminate the need for anchor-based detectors. Removing the chances that an anchor box would not be in range to make a detection. In Figure 1b, we see that CenterNet uses the Stacked Hourglass network as its backbone, which is 104 layers deep [14].

Faster R-CNN is a two-stage detector with a region proposal stage and Region of Interest (RoI) pooling stage [10]. Its post-processing stage, which includes the RoI pooling, puts this model behind YOLO and other single-stage detectors in speed. However, R-CNN has been a good baseline in previous years to compare to other speed-centered models in precision. The newer versions (e.g., Fast R-CNN and Faster R-CNN) also have increased in speed and now compete with YOLO models.

Faster R-CNN can run with multiple backbone networks (e.g., VGG [15], ResNet [16]) which in our results we use ResNet-101. In Figure 2a, the Conv layers denote the section where different backbone networks may be used. The original network that Faster R-CNN runs on (VGG-16) has high memory usage compared to other networks. VGG-16 has between 95-125 million operations compared to ResNet-101 at 35-65 million [17]. Though ResNet uses less memory, it is also a significantly deeper network. ResNet is 101 layers deep which is 8X deeper than VGG [16].



(a) Faster R-CNN (b) MobileNet SSD

Fig. 2: (a) shows the network of Faster R-CNN. We can observe the region proposal stage (Proposals) and the Region of Interest (RoI) pooling as the first detection stage. (b) shows the network of MobileNet SSD. It uses MobileNet as its backbone before entering convolutional layers.

MobileNet Single Shot Multibox Detector (MobileNet SSD) is a single stage detector. It provides competitive results in accuracy and computation speeds compared to other single-stage detectors like YOLO. MobileNet SSD makes more detections per class than YOLO, 8732 to 98 detections, respectively [18]. In our experiments, we use MobileNet SSD to observe a model that can operate on a mobile device. In Figure 2b, we can see MobileNet SSD uses the MobileNet backbone. Should edge computing be necessary, this model can provide satisfactory results alongside the small network size of MobileNet. The MobileNet backbone significantly decreases the number of parameters in the network. MobileNet lowers the number of parameters on COCO object detection results from 138 million (VGG) to 4.2 million [12], making it ideal for mobile computations.

3.2 Requirements of Benchmarking Models

To properly benchmark our models, each must tackle a different challenge in our Mummy Nut datasets. YOLO outperforms R-CNN when trained on artwork and natural images. This high generalizability makes YOLO less likely to break down when applied to new domains or unexpected inputs [7]. Given our dataset's vast appearance diversity, high generalizability may increase recall value. Faster R-CNN may also create more detection errors on background content compared to YOLO [19]. Considering the amount of noise in our Original dataset, Faster R-CNN may be more suitable for our Augmented dataset, presenting less noise within images.

In our experiments, we use modern versions of YOLO, including YOLOV3 [19] and V5 [20]. YOLOv3 runs on the DarkNet-53 backbone network, which is 53 convolutional layers deep [21]. This is nearly half the size of the ResNet-101 network that Faster R-CNN uses. Considering the tiny objects in our dataset, the chances of missing a target object with anchor-based detectors (e.g., YOLO, Faster R-CNN, or MobileNet SSD) are relatively high. Thus CenterNet provides

a different approach for problems where typical anchor boxes may fail. The backbone CenterNet uses is created for the task of human pose estimation and is built to capture information at every scale [14]. With this structure, CenterNet can tackle the problem of a large variety of Mummy Nut sizes. Considering the need for edge computation in the agricultural setting, MobileNet SSD provides a very small network to fulfill that requirement.

4 Benchmarking Methodology

Our benchmarking methodology is as follows: we collect real datasets on Mummy Nuts, analyze object detection results with popular CNN models, propose our Augmented dataset, and redo our object detection analysis until we achieve adequate numbers with our chosen metrics.

4.1 Proposed Datasets: Original and Augmented

A carefully curated dataset must be used during training to build a robust model. Underrepresentation in different values within the dataset should be avoided whenever possible. However, when data can only be collected during a specific time of the year, it would significantly limit the amount of data and improvements to the data (e.g., better lighting, more angles, etc.). In collaboration with agriculture scientists, we have curated two datasets (Original and Augmented) for the task of Mummy Nut object detection. In our Original dataset, we observe a large amount of underrepresentation in specific shapes and sizes of the nuts. Due to the sporadic diversity in agriculture, a rich dataset is challenging to produce. Hence, we propose a new Augmented dataset that derives from the Original and includes data augmentation methods to artificially increase the size of annotations. Our Original dataset contains 33 images (4032x3024) and about 267 annotated nuts, while our Augmented dataset contains 294 images (200x200), each containing a modified or original annotated nut.

The proposed datasets pose many challenges for deep neural networks: **Noise**: Most training datasets should include noise to a certain degree to create a more robust model. However, our original dataset contains a significantly larger object-to-noise ratio that most models can not understand well. Approximately 75% of each image include noise that severely hinders the visual features of our objects. **Small Scale**: Our dataset contains tiny annotation boxes (smallest at an area of 121 pixels), which may cause difficulty for anchor-based object detection systems. We observe which models may be affected by this difficulty. **Large Data**: To preserve as many visual features as possible, we use the full-size image (3024 by 4032). However, this requires much larger training times between models and larger memory space for computations. We experiment with different models' training and inference times. **Diversity**: There is a large variety in the appearance of our objects in shapes, sizes, and coloration. Due to unpredictable exposure to light, target objects within our dataset may lose all surface textures or appear overwhelmingly rich in features.

Fig. 3: Four sample pictures of the Mummy Nut dataset. Hanging on the branches of these trees are the target objects (Mummy Nuts).

In Figure 3, we can observe the variety of difficult detections in our dataset. These nuts account for 71.8% of all annotated nuts throughout our dataset, affecting a large portion of our accuracy. Within our difficult detections, we also classified tiny nuts as about 9% of the total dataset.

Figure 4 shows a close-up collage to display examples of difficult Mummy Nuts detections. Frames (a) and (f) contain dark nuts that lack visual features and are further surrounded by a noisy background, causing models to miss these nuts as detections. Frames (b) and (d) contain bundles of nuts that, conversely to frames (a) and (f), are rich in visual features. However, they are overly rich and feature a rare trait in coloration. Frames (c) and (e) contain camouflaged nuts, which cause trouble even for the human eye to spot.

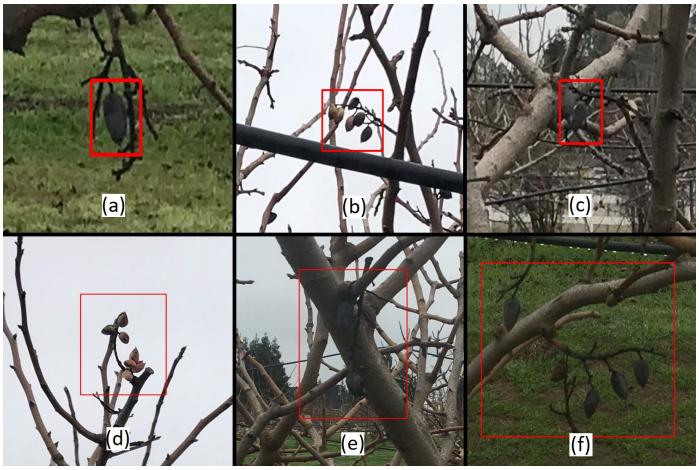


Fig. 4: Six examples of difficult detections. Each frame contains a red box that denotes where the Mummy Nuts appear.

Without tuning the anchor boxes of a CNN, the chances of missing detections increase. This raises the question of whether to resize anchors for small or large

objects. In Figure 5, most annotations are tiny, though many outliers exist. Resizing anchors for small target objects would make detecting larger target objects more difficult and vice versa. Models that remove the need for anchor boxes can altogether avoid this problem (e.g., CenterNet).

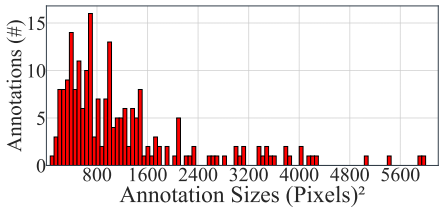


Fig. 5: The area (in pixels) of each annotation box across the entire dataset. We can see a majority of annotation boxes are smaller than 2000px in size.

We also introduce our second dataset (Augmented), which expands our Original training dataset by 8.9X. The Augmented dataset contains cropped data augmented annotations from the Original dataset. Networks that rely heavily on anchor boxes may see a change in performance from this dataset. To create this dataset, we first used the original fully annotated dataset and produced a crop window per annotation centered on each annotation. The crop window (200x200) is larger than the largest annotation in the training set. Cutoff annotations or multiple annotations within one crop were excluded. Each annotation was subject to a random number of image transformations (up to 6), including stretching and rotations. These augments are done to each cropped image, effectively enlarging our input dataset.

4.2 Metrics

In the evaluation of the accuracy of each model, we observe the following metrics: precision (Prec), recall (Rec), and average precision (AP). These are evaluated based on the true positives, true negatives, false positives, and false negatives we observe after a model has run its predictions. IoU tells us if a detection will classify as one of the four prediction categories. For performance, we evaluate latency between models and GPUs. These values give us the four evaluation metrics we will focus on, which are as follows: **Precision**: is the proportion that the target object in an image will be detected correctly over all attempts. **Recall**: tells us the proportion of target objects captured over all attempts. In other words, it represents the target objects overlooked by the detection model. **Average Precision (AP)**: is the precision with respect to recall. AP is calculated by the area under the curve (AUC) of a precision-recall graph based on each model's precision and recall values. In the MS COCO object detection challenge, a 101-point interpolation is used to evaluate models [2]. While for ImageNet, AUC method is used [3]. We used AUC rather than 11 or 101-point interpolation to evaluate lower accuracy values better, as n-point interpolation may be prone to drops in precision in between interpolations and lead to less evaluation accuracy [22]. In our datasets, we observe that 101-point interpolation leads to less sensitive accuracy, where AUC evaluation calculates the area each time there is a drop in

precision. For this reason, we evaluate AUC. The equation is provided below:

$$AP = \sum (r_{n+1} - r_n) P_{interp}(r_{n+1}) \tag{1}$$

Performance: Performance represents how long it takes for models to perform computations on our dataset. The time it takes to perform required computations will be vital in determining which model is most suitable for specific agriculture applications. We measure two different processing phases. 1) Training results observe the time it takes to train a model to the point where the loss converges. 2) Inference measures the total time (including pre-processing time of convolutional layers) to perform inference on a single image.

4.3 Proposed Tools

We propose tools to ease the workflow with our Mummy Nuts dataset. In Table 1, we list all tools we have created for the object detection task of Mummy Nuts. **Difficulty Class Annotation (DCA)** is a tool that allows the user to input an annotated dataset and receive a richer annotation set that includes one or more difficulty class flags per annotation. For example, we can create a specific set of flags (e.g., Camouflaged, Dark, Overlap, etc.) and mark each annotation with as many difficulty flags that apply. **Metric Evaluation (Metric Eval)** computes all evaluation metrics on input detection results from any of the 5 models. This tool is flexible to multiple model annotation formats. If the input contains DCA flags, the evaluation will also include results for each difficulty class. **Data Augmentation (Data Aug)** tool takes an annotated dataset, crops windows (200x200 px) centering on each annotation, and produces new images with image transformations. The complete output creates an enlarged dataset. This tool is used to create our Augmented dataset.

Annotation Plot (APlot) plots all annotations on an image, which allows us to visualize annotation box concentrations, scale sizes, and empty areas. Annotation Plot tool can also provide data on all annotation box sizes, similar to Figure 5. **Noise Isolation (Noise Iso)** helps reduce the amount of noise in a Mummy Nut tree image. Pixels that classify as the grass gets lowered, allowing focus on pixels that are the tree or Mummy Nut. This tool can aid in the annotation pipeline to increase annotation recall. **Annotation View (Anno View)** creates a temporary viewing window during the tool's runtime that shows the target nut and the annotation bounding box. The tool can cycle through all annotations in a dataset, aiding the annotation pipeline when peer-reviewing annotations. **Annotation Reformat (Anno Reformat)** allows reformatting of an annotation to a different format. This is especially useful when using multiple models requiring different annotation formats.

5 Experiments

This section provides our experimental setups and benchmarking results.

Table 1: All proposed tools and their functions. These tools require two types of inputs: Annotated Dataset (AD) and Detection Box (DT) Results.

Tool	Input(s)	Output(s)
DCA	AD	AD w/difficulty class flags per annotation
Metric Eval	AD+DT	metrics based on detection box positions w/DCA
Data Aug	AD	enlarged AD via image augments per annotation
APlot	AD	all annotation boxes plotted + data of all box sizes
Noise Iso	AD	AD w/all noise values lowered
Anno View	AD	viewing window of each annotation
Anno Reformat	AD	reformatted annotations for different model formats

5.1 Platform Selection

We use two different GPU hardware to run our computations. The first is an in-home cluster with an NVIDIA A100 40GB PCIe GPU. The NVIDIA A100 GPU is capable of 156 TFLOPS on dense 32-bit float tensors [23]. Model training benefits significantly from the GPU's high 1555 GB/s bandwidth and 40GB of GPU memory. The cluster's CPU is an Intel(R) Xeon(R) Gold 6336Y with 24 cores, 36MB of cache, and a base frequency of 2.40GHz [24]. The second GPU is a Tesla T4 which is more easily accessible via Google Colab. The Tesla T4 contains 2560 NVIDIA CUDA cores and is capable of 65 trillion mixed-precision floating point operations [25]. The Colab CPU is a virtual CPU with two cores with a clock speed of 2.2GHz. We use these two platforms to test each model's performance throughput and speed.

5.2 Results

Our results are categorized into two subsections, as shown below. **Overall Accuracy**: An overview of how each model performs on our dataset. Due to the small size of our dataset, our numbers are preliminary. However, we may still make some significant distinctions. We observe the number of predictions a model makes on background pixels, quantifying the robustness of each model to noise. Localization performance can be observed between IoU results from 0.50 and 0.10. IoU becomes very sensitive with small-scale objects and scales disproportional to normal-sized boxes [26]. For this reason, we acknowledge the results of an IoU at 0.10. Recall also provides valuable data in a complex dataset where collecting maximum true positives is essential. **Performance**: We have observed each model's performance in training and inference. We quantify differences in those numbers with our Original dataset, our Augmented dataset, and different GPU platforms. For the Original dataset, we record the time for one inference. For the Augmented dataset we split input images evenly into 64 crops to match training inputs. We then record the total time of performing 64 inferences, equivalent to one Original dataset image.

Overall Accuracy: With a very noisy dataset, models will generate excessive detections on background pixels. Models that do not have a high threshold NMS

may not properly filter out duplicate detections on the same true positive. In Table 2, MobileNet SSD suffers from this as it produces 2X the number of ground truths in the test set. We observe that this is due to the size of the MobileNet convolutional network. Since the network is much smaller than other modern networks, MobileNet SSD may need a deeper network to learn complex features properly. However, the lowest AP score comes from YOLOv3. YOLOv5's precision and recall increase by about 2X and AP by about 5X when the IoU threshold is set to 0.10. This tells us that YOLOv5 struggles with localization issues for detections on our dataset. The highest values overall are in Faster R-CNN with low localization error and high precision. The lowest amount of background detections is made from Faster R-CNN, which gives this model 100.0 precision at an IoU of 0.10. This may be due to Faster R-CNN running on the ResNet network, as it is a very deep network (101 layers), which provides higher confidence in detections and fewer false positives.

Table 2: Inference accuracy with our Original dataset (4032 by 3024 images). The predictions (Pred) show how many predictions each model has made. The true positives (TP) show the number of those predictions classified as correct according to the IoU threshold. Precision (Prec), Recall (Rec), and Average Precision (AP) are shown with corresponding IoU thresholds. The test partition has 15 ground truths.

Model	Pred	TP@.5	Prec@.5	Rec@.5	AP@.5	TP@.1	Prec@.1	Rec@.1	AP@.1
YOLOV3	11	2	18.18	13.3	2.87	3	27.2	20.0	7.33
YOLOV5	25	4	16.0	26.6	4.31	8	32.0	53.3	21.9
CenterNet	12	4	33.3	26.6	13.7	6	50.0	40.0	30.0
Faster R-CNN	9	7	**77.7**	**46.6**	**41.6**	9	**100.0**	**60.0**	**60.0**
MobileNet SSD	30	6	20.0	40.0	10.0	6	20.0	40.0	12.0

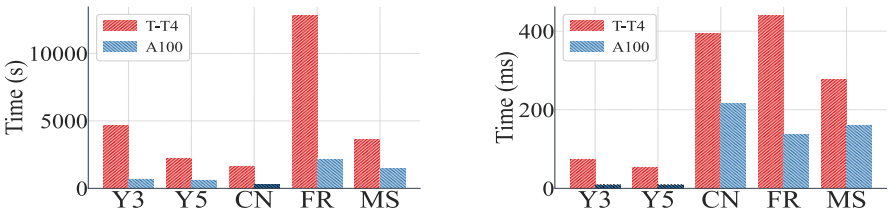
In Table 3, we notice our Augmented dataset causes each model to increase background detection errors. Since our Augmented test set contains much smaller input images, we notice higher number of detections. However, we see a significant increase in recall in the YOLO models compared to the Original dataset. For YOLOv3, We observed a 6.5X increase in the recall at IoU 0.50 from the Original dataset to the Augmented dataset. While for YOLOv5, we saw a 2.5X increase. While the YOLO models do not decrease in precision, each of the three other models does. This is due to the large number of background errors that these models now produce. CenterNet increases the most at about 29.6X more background detections. Since CenterNet uses heatmaps rather than anchor boxes, these background errors are likely due to a large amount of noise, as the maxima of the heatmaps may become lower when there are fewer distinctions in coloration. MobileNet SSD is ranked second, producing about 24X more background detections. As with the Original dataset, this is likely due to the small size of the backbone network. YOLO models are more robust to background errors, with the most significant increase in errors at 2.6X. Since they use an-

chor boxes rather than heatmaps and a larger backbone network than MobileNet SSD, they get better results using the Augmented dataset.

Table 3: Inference accuracy with our Augmented dataset (200 by 200 images).

Model	Pred	TP@.5	Prec@.5	Rec@.5	AP@.5	TP@.1	Prec@.1	Rec@.1	AP@.1
YOLOV3	37	13	35.1	**86.6**	38.5	13	35.1	**86.6**	38.5
YOLOV5	32	10	31.2	66.6	26.2	10	31.2	66.6	26.2
CenterNet	245	8	3.26	53.3	3.27	9	3.67	60.0	5.17
Faster R-CNN	30	13	**43.3**	**86.6**	**51.0**	13	**43.3**	**86.6**	**51.0**
MobileNet SSD	583	6	1.02	40.0	0.57	7	1.20	46.6	0.78

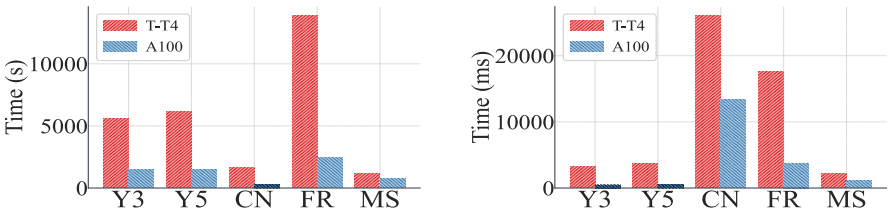
Performance: We can observe in Figure 6 that on the Tesla T4, Faster R-CNN has a significantly longer training time than the rest of the models, likely due to the large depth of the backbone network and a large number of computations. On the A100, we see a significant decrease in latency due to the larger network benefiting greatly from the considerable upgrade in bandwidth, up to a 5.85X speedup for Faster R-CNN.



(a) Training Time (b) Inference Time

Fig. 6: Computation times of each model using Original dataset on both GPUs. We compare the performance speed of our five models, YOLOv3 (Y3), YOLOv5 (Y5), CenterNet (CN), Faster R-CNN (FR), and MobileNet SSD (MS) on the Tesla T4 (T-T4) and A100 (A100).

In Figure 7, we can observe that Faster R-CNN still takes the longest to run training on the Augmented dataset on the Tesla T4 and speedup of 5.68X with A100. YOLOv5 has faster training and inference times on the Original dataset, while YOLOv3 performs slightly better on the Augmented dataset. This is likely due to optimizations for larger image sizes of successive versions of YOLO. We also notice that YOLO has the fastest overall inference time on both datasets. MobileNet SSD can perform much quicker on the Augmented dataset than the Original dataset since it is a tiny network and benefits from smaller inputs, as provided by the Augmented dataset.



(a) Training Time (b) Inference Time

Fig. 7: Computation times of each model using Augmented dataset.

5.3 Observations and Discussion

We evaluate our two-dimensional benchmarking methodology of accuracy and performance with all the results. We observe accuracy with a heavier emphasis on recall values rather than precision due to the small scale of our annotations which aids in compensating for the difficulty within our Mummy Nut dataset. For performance, we evaluate two different GPUs to observe computation latency. We collect the computation time of model training and inference.

From these results, we notice several key observations, and we make a note of their significance. Firstly, all models saw an average recall value increase of 2.77X, with YOLOv3 at the most significant increase of 6.5X, when using our Augmented dataset over the Original. This observation implies that our method can further improve the model recall. We notice that all models except MobileNet SSD face localization precision issues with the Original dataset. These localization issues tell us that most models create bounding boxes that may not enclose the Mummy Nut well, though MobileNet SSD performs more precisely. CenterNet, Faster R-CNN, and MobileNet SSD decrease precision by an average of 10.53X using the Augmented dataset over the Original. If an Augmented dataset approach is desired, YOLO models may provide better results in precision than the other three models. For performance, the overall model training time over both Original and Augmented datasets can be reduced by an average of 4.45X when using A100 over Tesla T4. Though we notice Faster R-CNN sees the most significant benefit from this speedup by 5.76X over both datasets. This would be due to larger CNNs being able to take full advantage of the increased computation capacity of A100. YOLO models have the fastest overall inference speed and prove useful for quick inference applications.

6 Related Work

A survey on popular precision agriculture datasets overviews 34 different datasets for deep learning workloads that include multimodal data [27].

Other highly rich datasets create an annotation pipeline to ensure the highest possible annotation precision and recall. MS COCO splits the pipeline into three

sections: Category Labeling, Instance Spotting, and Instance Segmentation [2], creating a workflow simple enough for inexperienced annotators to provide a rich dataset. Other large-scale datasets [3,4,5,6] outline strong annotation pipelines.

Contrast limited adaptive histogram equalization (CLAHE) is a computer vision technique that can equalize the brightness exposure between images to maintain features within an image [28]. Using a dual-modal detection of color images and thermal, both precision and recall values are much higher than using color images alone [29]. Other methods can make 3D detection using LiDAR point clouds [30], which is much more informative for a complex environment. IoU becomes very sensitive with small-scale objects and scales disproportional to normal-sized boxes, lowering model accuracy. Normalized Wasserstein Distance (NWD) creates a new metric for IoU that can be implemented into Faster R-CNN [26]. Most models also struggle to make rotation-invariant detections. Multiple different models have been proposed [31,32,33,34] which tackles the challenges in oriented target objects.

Other proposed methods tackle accelerating EdgeAI inference systems for object detection, improving performance and accuracy with modern EdgeAI platforms [35,13]. There is also a study on the performance of mobile GPUs [36]. A benchmark of multiple Deep Learning models on different edge devices evaluates the performance latency of object detection tasks [37]. Work that lists and surveys multiple popular Deep Learning benchmarks provides insightful observations on each type of benchmark [38].

Our work is different than all of these existing studies since we aim to provide valuable benchmarking datasets and tools for an engaging, meaningful, and challenging research problem in the precision agriculture area (i.e., Mummy Nuts).

7 Conclusion and Future Work

We propose a benchmarking methodology to evaluate five different CNN models using two dimensions of measurement, which include accuracy and performance. Our results show that using our Augmented dataset can drastically change the CNN model's overall performance. All models increase recall with our Augmented dataset, with YOLO models as the most significant increase. Faster R-CNN and YOLOv3 can achieve the highest recall of all models when using data from the Augmented dataset. However, if it is desired that the Original dataset is used, we find that Faster R-CNN performs very well in both precision and recall values. YOLO models do well in speed, with the overall fastest inference speed. However, the Augmented dataset is significantly more sensitive to noise for each model, though less for YOLO models and Faster R-CNN. To sum up, YOLO and MobileNet SSD models may be more suitable for an Augmented dataset method, whereas CenterNet may perform better on the Original dataset. Faster R-CNN is the most versatile and may be well applicable to both datasets. For future work, we will enrich our datasets further.

References

1. C. Almonds, "Navel Orangeworm." https://www.almonds.com/almond-industry/industry-news/mummy-nut-removal-ready-set, 2022.
2. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *European Conference on Computer Vision*, pp. 740–755, Springer, 2014.
3. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.
4. M. Everingham, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2009.
5. J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN Database: Large-Scale Scene Recognition from Abbey to Zoo," *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492, 2010.
6. P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 743–761, 2012.
7. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.
8. R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.
9. R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.
10. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Advances in Neural Information Processing Systems*, vol. 28, 2015.
11. K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint Triplets for Object Detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6569–6578, 2019.
12. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017.
13. Y. Hui, J. Lien, and X. Lu, "Early Experience in Benchmarking Edge AI Processors with Object Detection Workloads," in *International Symposium on Benchmarking, Measuring and Optimization*, pp. 32–48, Springer, 2019.
14. A. Newell, K. Yang, and J. Deng, "Stacked Hourglass Networks for Human Pose Estimation," in *European Conference on Computer Vision*, pp. 483–499, Springer, 2016.
15. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.
16. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.
17. A. Canziani, A. Paszke, and E. Culurciello, "An Analysis of Deep Neural Network Models for Practical Applications," *arXiv preprint arXiv:1605.07678*, 2016.

18. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot Multibox Detector," in *European Conference on Computer Vision*, pp. 21–37, Springer, 2016.
19. J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018.
20. G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, K. Michael, Lorna, A. V, D. Montes, J. Nadar, Laughing, tkianai, yxNONG, P. Skalski, Z. Wang, A. Hogan, C. Fati, L. Mammana, AlexWang1900, D. Patel, D. Yiwei, F. You, J. Hajek, L. Diaconu, and M. T. Minh, "ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference," Feb. 2022.
21. J. Redmon, "Darknet: Open Source Neural Networks in C." http://pjreddie.com/darknet/, 2013–2016.
22. J. Hui, "mAP (mean Average Precision) for Object Detection." https://medium.com/p/45c121a31173, 2022.
23. NVIDIA, "NVIDIA A100 Tensor Core GPU Datasheet." https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf, 2022.
24. Intel, "Intel Xeon Gold 6336Y Processor Datasheet." https://www.intel.com/content/www/us/en/products/sku/215280/intel-xeon-gold-6336y-processor-36m-cache-2-40-ghz/specifications.html, 2022.
25. NVIDIA, "NVIDIA Tesla T4 Tensor Core GPU Datasheet." https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-t4/t4-tensor-core-datasheet-951643.pdf, 2022.
26. J. Wang, C. Xu, W. Yang, and L. Yu, "A Normalized Gaussian Wasserstein Distance for Tiny Object Detection," *arXiv preprint arXiv:2110.13389*, 2021.
27. Y. Lu and S. Young, "A Survey of Public Datasets for Computer Vision Tasks in Precision Agriculture," *Computers and Electronics in Agriculture*, vol. 178, p. 105760, 2020.
28. D. Choi, W. Lee, R. Ehsani, J. Schueller, and F. Roka, "Detection of Dropped Citrus Fruit on the Ground and Evaluation of Decay Stages in Varying Illumination Conditions," *Computers and Electronics in Agriculture*, vol. 127, pp. 109–119, 2016.
29. H. Gan, W. Lee, V. Alchanatis, R. Ehsani, and J. Schueller, "Immature green citrus fruit detection using color and thermal images," *Computers and Electronics in Agriculture*, vol. 152, pp. 117–125, 2018.
30. C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
31. R. Qin, Q. Liu, G. Gao, D. Huang, and Y. Wang, "MRDET: A Multi-Head Network for Accurate Oriented Object Detection in Aerial Images," *arXiv preprint arXiv:2012.13135*, 2020.
32. J. Yi, P. Wu, B. Liu, Q. Huang, H. Qu, and D. Metaxas, "Oriented Object Detection in Aerial Images with Box Boundary-Aware Vectors," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2150–2159, 2021.
33. M. Zand, A. Etemad, and M. Greenspan, "Oriented Bounding Boxes for Small and Freely Rotated Objects," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2022.
34. J. Han, J. Ding, N. Xue, and G.-S. Xia, "ReDet: A Rotation-Equivariant Detector for Aerial Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2786–2795, 2021.

35. Y. Hui, J. Lien, and X. Lu, "Characterizing and Accelerating End-to-End EdgeAI Inference Systems for Object Detection Applications," in *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 01–12, 2021.
36. C. Gao, A. Gutierrez, M. Rajan, R. G. Dreslinski, T. Mudge, and C.-J. Wu, "A Study of Mobile Device Utilization," in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 225–234, 2015.
37. A. Allan, "Benchmarking Edge Computing." https://aallan.medium.com/benchmarking-edge-computing-ce3f13942245, 2022.
38. Q. Zhang, L. Zha, J. Lin, D. Tu, M. Li, F. Liang, R. Wu, and X. Lu, "A Survey on Deep Learning Benchmarks: Do We Still Need New Ones?," in *International Symposium on Benchmarking, Measuring and Optimization*, pp. 36–49, Springer, 2018.