An Energy Conscious Review of Current State-of-the-Art Object Detection Algorithms*

Gareth Farrell
School of Computing
Dublin City University
Dublin, Ireland
gareth.farrell34@mail.dcu.ie

***Abstract*—Energy consumption is becoming an increasingly important aspect of modern society where anyone from companies to countries are increasingly conscious of their energy use due to the impacts of climate change, the demands to address carbon emissions, and the cost of energy itself. In the world of deep learning, the aspect of energy consumption is largely overlooked, with performance almost the exclusive metric to which models are measured by. As a result, deep learning models have made significant performance improvements in recent years, but in many cases the computer power used to achieve these improvements has increased also. This paper explores the relationship between the leading Object Detection algorithms performance and the energy consumption used to train the respective models.**

***Index Terms*—object detection, energy, YOLO, R-CNN, DETR**

I. Introduction

Object detection algorithms have seen great advancements in recent years with the YOLO series of models especially achieving significantly faster inference speeds [6] [39] with comparable performance to the other types of leading models in this space. The rise of autonomous vehicle technology in particular has brought greater demand for research into more effective models due to the high degree of accuracy and reliability required for such a safety sensitive environment. In addition to being faster and more accurate, models must also be as energy efficient as possible in order to minimise their drain on the vehicle's battery in an environment where every extra kilometer of range counts.

The improvements in Object Detection algorithms have coincided with similar advancements in graphical processing technology, enabling far more computing power to be directed at this problem. The question therefore is have the algorithms improved significantly compared to what was previously developed, or has the technology available to run the models provided the biggest difference, and as such, has energy use increased with performance. This question will be examined through the lens of power efficiency, where the leading models from the past and present in Object Detection will be assessed for their performance measured in Average Precision (AP) and Frames per Seconds (FPS), contrasted against the energy consumption (Watts) required for training for each respective model. This will provide a clear picture of what models perform best in both absolute terms and with respect to the energy cost of the model.

As object detection models become more and more applicable to real-world problems and move from the research and development space into real-world application, the training and use of such models will consume more and more energy, at a time when society needs to reduce its energy needs. The suggestion of this work is not to stop progress of object detection models in the name of climate change, but to highlight the often unmentioned energy costs associated with these models, so that this is at the very least a consideration in future developments into object detection and deep learning as a whole. Particularly when one of the leading use cases for object detection is autonomous driving, where many of the companies that either plan to or are currently employing a version of this technology repeatedly promote their focus on clean energy, the energy costs of such software should be of specific concern.

II. Object Detection

To best help computers understand the visual world as humans do, many different techniques have been proposed to aid artificial intelligence systems in recognition and interpretation of objects that appear in images. These challenges are known as the fundamental problems of computer vision, one of which is object detection [40].

Object detection is the task of precisely estimating the classification and location of an object in a given image [11] and can be seen as the foundation on which higher level systems process this information to create more complex semantic understandings. For example, an object detection systems could detect a *person* and a *crosswalk* beside each other. Higher level systems could then infer that when these two objects are present and close, right-of-way should be given to the person.

Effective and efficient object detection is crucial to autonomous driving systems where real time detection and classification of objects in a vehicle's surroundings is vital to accurately and reliably avoid collisions and to intelligently understand it's environment [6]. Object detection has many other applications such as tracking of objects in a warehouse setting [4], but due to the highly energy sensitive nature of autonomous driving, the focus of this paper is on this particular use case.

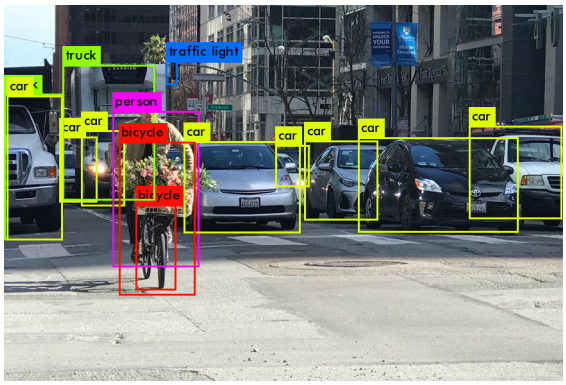

Fig. 1. Object classification with bounding boxes [33].

Modern object detection models are usually deep learning based and can primarily be separated into two categories, one-stage detectors and two-stage detectors [8]. The most common models in one stage detectors are variations of the You Only Look Once (YOLO) series of models, whereas Region-based Convolutional Neural Networks (R-CNNs) are the most common variant of two-stage detectors. The main difference between these two categories is the concept of the region-proposal, the first stage of a two stage model where the most relevant regions of an image are first identified, then these regions are then passed to the CNN in the second stage where the object detection is performed [16]. One stage detectors skip this region proposal stage and go straight to detection [9]. Due to the more careful assessment of regions in the two-stage models, these types of algorithms are usually more accurate. However, the one-stage models are generally significantly faster and as such are much more suitable for real-time applications.

In order to compare these models, a consistent benchmark across all models needs to be applied. This is commonly one of either Microsoft COCO (MS-COCO) or PASCAL Visual Object Classes (VOC), though many other datasets can be seen. The metrics used to assess a models performance on these datsets are mean average precision (mAP) to calculate accuracy, though this is frequently used interchangeably with average precision (AP), and frames per second (FPS) to calculate inference speed.

As mean average precision is the average precision of the models accuracy across all classes, this number may be misleading when assessed for reliability in specific contexts, such as autonomous driving. For example, if a model is 90% accurate on the *apple* class, but 50% accurate on the *cars* class, the mAP of these two would be 70% but quite obviously this would not be a very good model for autonomous vehicles. For real-time applications, it is generally accepted that at least 30 FPS is required for effective use, and ideally higher [29].

III. Background

The energy efficiency of object detection models is largely ignored in the main space of assessing a model's performance with the metrics of accuracy and inference speed by far the most popular. However, there has been some work on assessing the energy cost associated with these models. Analysis has been performed on optimisation techniques to improve CNN performance on mobile hardware [18] with a view to being used for self-driving vehicles. The focus is on model run time performance contrasted with energy use as opposed to training costs as well, with a similar approach taken by Wang et al. [37]. Collange et al. [7] examines the power usage of a GPU during the low-level computation performed on the device in a CUDA environment, but in more of a general sense and without any relation to object detection or deep learning. An interesting investigation into the carbon footprint of algorithms and subsequent tool for estimating cost is proposed by Lannelongue, Grealey, and Inouye [21], though again with no particular focus on object detection.

A. Model Architectures

This section is meant to provide a guide to the milestones of object detection development between two-stage and one-stage models over the past decade. An overview of the general architecture of some of the most commonly adopted models in object detection, as well as a brief description of the advancements in newer versions of each model type is discussed. This section is not meant to provide a detailed examination on the exact design and workings of any feature of any respective model, though the high-level idea and result of the model's main features is explored.

After the rise of popularity of deep learning in computer vision, object detection algorithms have seen huge developments, which can be largely attributed to the research into CNN based models that started decades ago but gained significant traction since 2012 after the resurgence of "AlexNet" [35].

B. Two-Stage Models

The first CNN based two stage object detection model [35], region proposal with convolutional neural network (R-CNN), was published in 2014. The architecture of R-CNN contains three modules in two stages. Stage one is composed of selective search used to generate region proposals [15]. Stage two has two modules which are the CNN that extracts a fixed-length feature vector from each region, and a set of class-specific linear support vector machines (SVMs) which classify each feature vector [15]. The accuracy of this model was vastly superior to anything that had previously been developed, with improvements in mAP of more than 30% relative to the previous best result on the VOC 2012 dataset [15]. The disadvantages however were the inference speed, as selective search is CPU compute and so could not take advantage of the speed improvements of GPU compute like the CNN could, and the training speed due to the inefficient Region of Interests (RoI) pooling method [35].

Fast R-CNN (2015) sought to address the main training and inference speed problems of R-CNN by introducing single-stage training and changing the RoI pooling layer [35]. The RoI pooling change was inspired by the Spatial Pyramid

Pooling Network (SSPNet) model, but further improved upon this to perform max pooling on all RoIs into only one layer instead of a pyramid of layers [14]. The result of the changes is a far faster model for both training and inference, and also more accurate [35] [14].

Faster R-CNN (2017) provides the next evolution of two-stage models by changing how the region proposal stage works. The evolution over Fast R-CNN is that the region proposal stage is itself a deep fully convolutional network that sends the proposed regions to the Fast R-CNN detector stage. By taking advantage of GPU calculation with a CNN instead of CPU on a selective search method, this greatly improves the inference speed whilst maintaining state-of-the-art accuracy on the MS-COCO and PASCAL VOC 2007, 2012 datasets at the time of publication in 2017 [32].

C. One-Stage Models

The first one stage model to use a CNN was You Only Look Once (YOLO) published in 2015. Unlike two-stage detectors, there is no region proposal stage and instead region detection and classification are combined into one stage, giving the model a significant speed advantage. The construction of the model as a unified CNN allows all computation to take place on the GPU [29], eliminating the CPU based region proposal bottleneck of R-CNN models at the time. Less than 100 bounding boxes are predicted per image in YOLO compared to 2000 region proposals in Fast R-CNN which additionally uses selective search to generate them [20]. This was the first model to offer truly real-time detection [20] while also offering 58% mAP on VOC 2012 [29].

Single shot multibox detector (SSD) can be seen as a mix between Faster R-CNN and YOLO. SSD is a one stage model but is even faster than standard YOLO [25], while also being more accurate than Faster R-CNN on the VOC 2012 test dataset [25]. The algorithm uses a multi-reference technique to create anchor boxes of different sizes and apsect ratios at different points across an image which are used to predict the bounding box of objects [26], similar to how Fast R-CNN approaches this problem. Further, a multi-resolution technique helps to predict objects at different scales through different layers of the network [26]. At the time only Fast YOLO, a less precise but far faster version of YOLO could provide faster inference times than SSD, but not nearly as accurately [25].

YOLOv2 or YOLO9000: Better, Faster, Stronger as it is technically named, is as the name suggests better, faster and stronger in some sense than YOLO [30]. This model was released in 2016 and by having a higher resolution classifier, batch normalization and a switch to anchor boxes similar to Faster R-CNN, the model is more accurate overall with better recall [19] [30]. Speed increases come from a new classification model called Darknet-19 that has 19 convolutional layers and 5 maxpooling layers as opposed to the 24 convolutional layers of YOLO [30].

YOLOv3 (2018) as the title of it's paper suggests, is an incremental improvement over YOLOv2 [31]. Anchors boxes

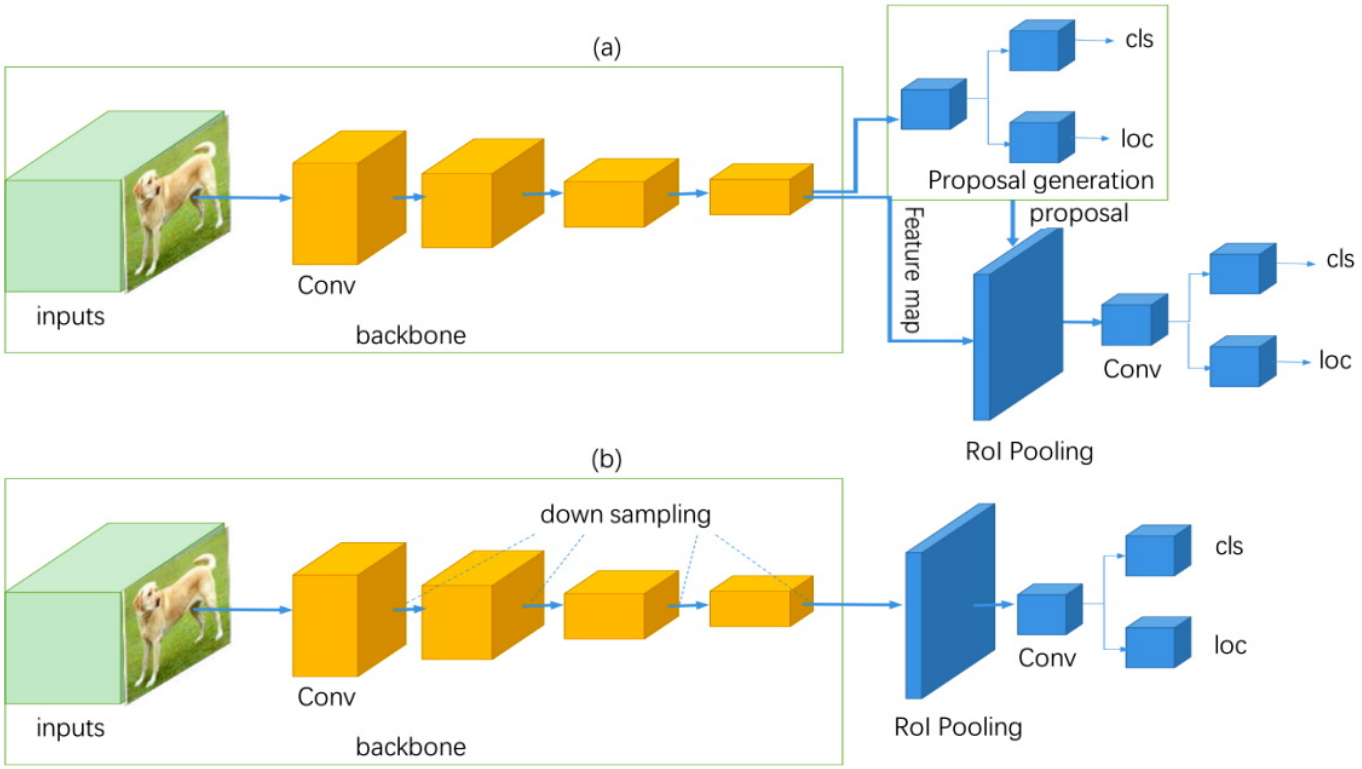

Fig. 2. (Adapted from Jiao, L et. al [2019]) (a) displays the basic architecture of a two-stage detectors where the region proposal network feeds into the classifier and regressor. (b) displays the basic architecture of a one-stage detector where bounding boxes are predicted from input images directly.

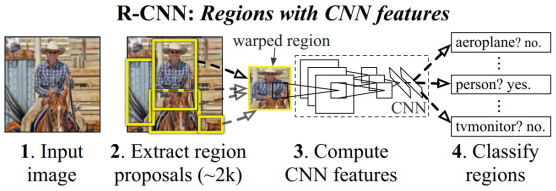

Fig. 4. (Adapted from Girshick, R et. al [2014]) R-CNN overview where around 2000 region proposals are extracted from an image, each of which are computed for features by the CNN which are then classified.

are used again but an objectness score or confidence value is assigned for each bounding box prediction, where 3 bounding boxes are predicted for each cell as opposed to 2 in YOLOv2. Multilabel class prediction allows for an object to be predicted as multiple classes at the same time, for example, "man" and "person" [20]. The model predicts objects at 3 different scales to improve detection on smaller objects, though at a cost of longer detection times [1]. Finally the model is larger by replacing Darknet-19 with Darknet-53 which as the name implies has 53 convolutional layers for feature extraction [31].

YOLOv4 (2020) is the first YOLO series model by different authors and at the time claimed to be "faster and more accurate than all available alternative detectors" (Bochkovskiy, Wang and Liao, 2020). A new backbone is introduced via a modified version of Darknet-53 in which Cross-Stage-Partial connections (CSP) split the current layer into two where one part follows the Darknet-53 structure as normal and the other part skips this to be finally merged together at the end [17]. The resulting CSPDarknet-53 can improve learning while reducing calculation and memory cost [17]. Several other features are added to aggregate information and improve accuracy such as the "bag of freebies" which is a collection of techniques to improve accuracy through data augmentation [2] at no additional inference time cost [28], and the "bag of specials" technique that does have a small inference time cost but much more significant accuracy improvements than the bag of freebies [2].

YOLOv5 was released just two months later in 2020 than YOLOv4, but by different authors. YOLOv5 is less of a major update and more of a slight alteration to YOLOv4, atleast initially, with comparisons between YOLOv4 and YOLOv5 often producing conflicting results in terms of which model was faster and/or more accurate [27]. Additionally, this model was the first in the YOLO series to be released without an accompanying research paper. One of the main changes in this version was the rewriting of the model from the Darknet framework written in C, to the PyTorch framework written in Python [1]. This greatly increased the accessibility of the model and was far easier to bring the model to production [19].

YOLOv6 was released just a few months ago in June 2022, and as such there is little-to-no peer reviewed papers about this model. Similar to YOLOv5 there is also no peer reviewed research paper published alongside the model, though there is a technical report from the authors Meituan technical team [10]. The focus of the model is industrial applications with support for hardware a key point, alongside faster and more accurate object detection than YOLOv5, exemplified by the YOLOv6-nano model achieving 35.0% AP accuracy on MS COCO at 1,242 FPS [10]. These improvements come from changes such as improved backbone networks EfficientRep Backbone and Rep-PAN Neck, and an anchor-free approach for training [10].

YOLOv7 was published just a month later in July 2022, and is only a few weeks old at the time of writing this paper. However, unlike YOLOv6 there is an accompanying research paper and is from the same authors of YOLOv4 [2]. This model claims state-of-the-art performance in speed and

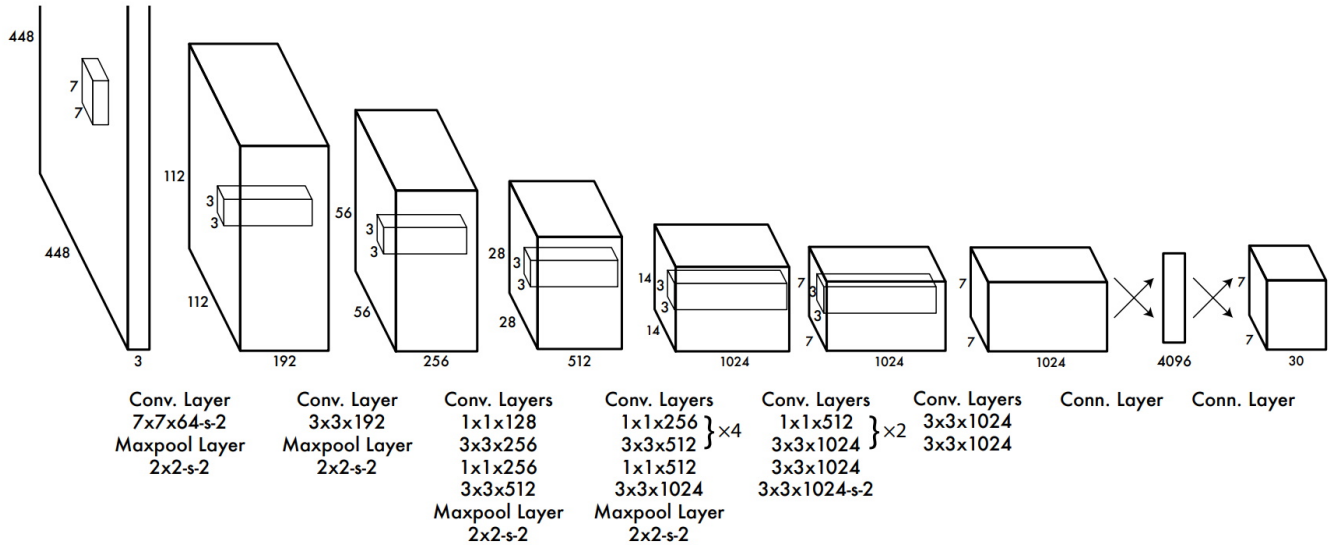

Fig. 3. (Adapted from Redmom, J et. al [2016]) YOLOv1 architecture of 24 convolutional folowed by 2 fully connected layers.

accuracy in the range of 5 FPS to 160 FPS and the highest AP of 56.8% of real-time detectors above 30 FPS [36]. A new final layer aggregation called Extended-ELAN (E-ELAN) shortens the distance the gradient takes to back-propogate through the layers, while model re-parameterization, model scaling and an auxiallary head features all improve accuracy and inference times while being practical for different computing devices [34].

DEtection TRansformer (DETR) was published by Facebook Research in May 2020 as a novel framework that uses the increasingly popular transformer technology that has been shown to be revolutionary in natural language processing [3]. Instead of a traditional one-stage or two-stage model, DETR can be described as an end-to-end detector [5]. The architecture is based on the standard transformer encoder-decoder architecture [38] where image features are encoded with positional context, then passed to a 6 layer encoder [38]. Similarly, the output of the encoder along with a small number (100) learned positional encodings known as *object queries* are passed to the 6 layer decoder [5] [38]. An advantage of this approach, particularly not using any post processing steps such as non-max suppression in YOLO [31], is that duplicate predictions are removed [24] and competitive results relative to Faster R-CNN are achieved on COCO 2017 [5]. However, training time is slow with around 300 epochs needed to achieve the published results [5].

DETR with Improved deNoising anchOr boxes (DINO) was published in July 2022 and is the state-of-the-art object detector holding first place in terms of box AP on COCO test and minival 2017 (Papers with Code - COCO test-dev Benchmark (Object Detection), 2022) at the time of writing. To achieve these record results, dynamic anchor boxes are introduced and refined across decoder layers as well as ground truth labels and boxes with noise [39]. The noise technique in particular is based off DeNoising DETR (DN-DETR) which only a few months prior, first introduced the concept of a denoising training method to increase DETR training speeds [22]. DINO significantly improves training speed with 49.4 AP achieved in 12 epochs and 51.3 AP in 24 epochs [39].

IV. Methodology

A. Configuration

All testing took place on a local machine, running Nvidia RTX 3050 8GB GPU, AMD Ryzen 5 2600 Six-Core Processor, 24GB DDR4 RAM, and a Corsair CV450 power supply. No overclocking is used for the GPU with power limit defaulted to 100%. Each algorithm is installed to it's own virtual environment to ensure there were no compatibility issues between versions of libraries and each algorithm. Where possible, the number of epochs each algorithm is trained for in their respective published paper or GitHub repository is used in the total estimated training time and consequent energy consumption. This number varies significantly from model to model, such the recommended training duration of YOLOv4 being 300 epochs, whereas the recommended duration for Detectron2 implementation of Faster R-CNN is only 37 epochs. All models are trained for a minimum of 2 hours.

The configuration of each model was set to the default where possible, and where not possible, the configuration was adjusted to maximize the performance of the RTX 3050, which resulted in "effective" batch sizes between 1-8 depending on the model. The use of the term "effective" is due to the subdivision process of YOLOv4 where the batch size is split into mini-batches during an iteration. This results in YOLOv4 running far less iterations but being comparatively far slower per iteration than other models. In the case of YOLOv4, the actual batch-size was 64, however, due to the subdivision also being 64, the "effective" batch-size was 1.

To ensure the best comparison between results, MS COCO 2017 was used for the training of all algorithms. The training of each algorithm lasted a minimum of 2 hours, where power consumption readings were measured in 0.1 seconds intervals to best capture the energy usage fluctuations. These results were then averaged across the training time, minus a small window in the beginning of the training to allow for "warm-up" of the GPU and ensure the device was under the full stress of the main training operation.

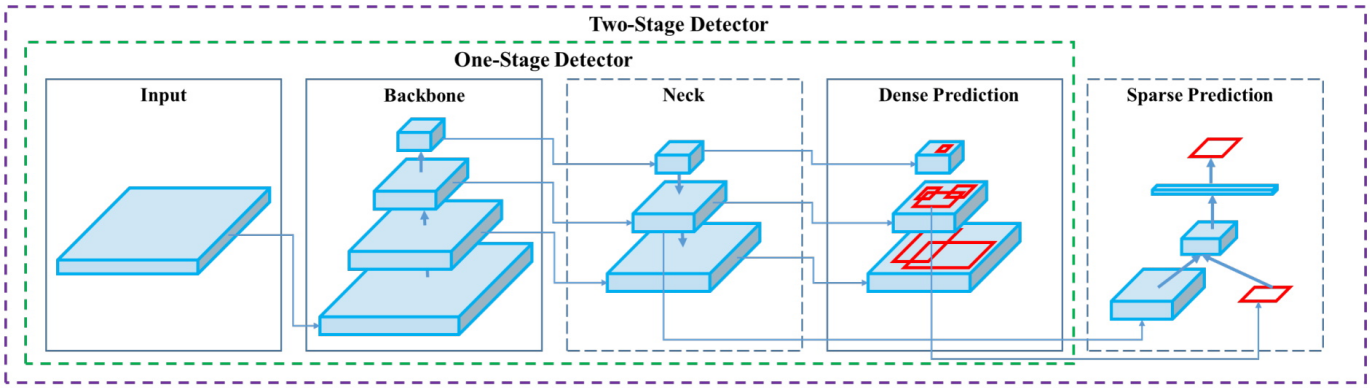

Fig. 5. (Adapted from Bochkovskiy, Wang and Liao [2020]) General object detector architecture.

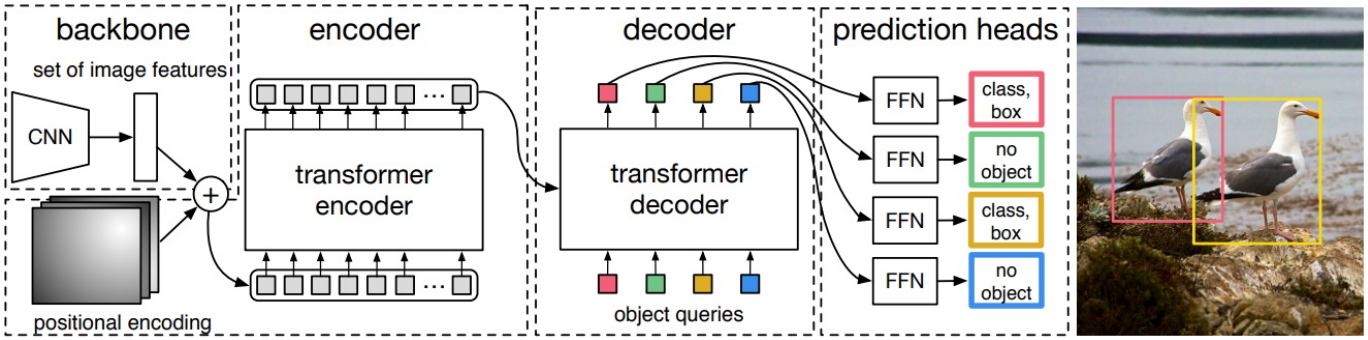

Fig. 6. (Adapted from Carion, et al [2022]) DETR architecture using a conventional CNN backbone which represent input image as 2D, flattens it, takes into account positional encoding then passes it to a transformer encoder. A transformer decoder takes some learned positional embeddings called *object queries* and the encoder output. Finally each output embedding of the decoder is passed to a shared feed forward network (FFN) that predicts a class and bounding box or "no object".

The implementation of each model may be different to the exact version when the algorithm was initially released, such as Faster R-CNN which has undergone many improvements and alterations since it's first publication, so the results can be seen to the reflect this papers particular implementation of the models at present. Additionally, the particular architecture of the GPU, such as Kepler, Maxwell and Pascal can affect the performance of object detection algorithms [12] [2]. For the experiments in this paper, the RTX 3050 uses the "Ampere" architecture.

To estimate the final power consumption of the full training time, the average power usage across the actual training time was extrapolated for the full epoch count, to estimate how much power the full training run would have consumed. It is accepted that this number is not exact and that events such as optimisation to the dataset after a certain number of epochs could affect GPU power usage among a number of possible factors. During testing, occasionally the GPU core clock and consequent energy use would not reach maximum output when model training was initiated, resulting in slower training times and reduced energy consumption. The exact reasons for this are unknown, so to account for this issue, measurements were not tracked until several trial runs of the model training were conducted along with the clearing of the GPU cache, to best allow the maximum performance to be measured.

B. Power Measurement

In order to test the energy consumption of the GPU, hardware or software is needed to measure the power usage. Hardware options will provide more accurate results but have drawbacks such as higher costs and more complications when measuring. Such complications could be ensuring to take into account the total energy usage and fluctuation of the computer system as a whole and isolating only the GPU's affect on the changes in power use when using the power "draw from wall" approach. Other hardware methods are available that can estimate GPU power usage far more accurately but with far greater hardware costs, such as the use of oscilloscopes [8].

Software approaches can measure the power usage of the GPU by measuring the values of built-in sensors in the GPU itself [13]. These sensors are vulnerable to a small degree of error, but this margin is acceptable when the average power usage is taken across a large enough period of time, particularly when the actual power consumption fluctuates considerably during high stress usage regardless. Several software applications enable real-time measurement of GPU power usage, and for the experiments in this paper two were chosen to help validate each others results. The final results for every measurement are an average of the two, though the difference is often effectively nil between each application. The two software applications used are "TechPowerUp GPU-Z" and "HWiNFO", which allow in-depth measurements of various GPU sensors, though in this case the only GPU board power draw is tracked.

V. Results

TABLE I
COMPARISON OF OBJECT DETECTORS PERFORMANCE

Model	AP	AP_{50}	AP_{75}	FPS	Est. kWh
YOLOv7	51.4	69.7	55.9	161	82.8
YOLOv4	43.5	65.7	47.3	65	104.6
DINO-4scale	49.0	66.6	53.5	24	23.3
Faster R-CNN (R101-C4)	41.1	61.4	44.0	7	53.8

Faster R-CNN is implemented using Detectron2 library.

The various AP and FPS values for each model are directly from each model's published results, as in order to find results from testing, the full training cycle would need to be completed, which due to the required training times as referenced in Table 2, would be unfeasible. The values in Table 1 should be seen as the optimal performance of each model and the respective energy consumption used during training to achieve these results.

From this, the results of testing showed DINO using the 4scale model version to be the most power efficient in terms of training of the officially recommend number of epochs on the MS COCO 2017 training set, while also offering very competitive accuracy. The least power efficient is YOLOv4,

TABLE II
COMPARISON OF OBJECT DETECTORS TRAINING PERFORMANCE AND CONFIGURATION

Model	Batch Size	Speed (sec/img)	Training set	Epochs	Est. Time (hrs)	Avg. Power (W)	Est. Total Power (kWh)
YOLOv7	8	0.08	COCO 2017	300	819	101	82.8
YOLOv4	1*	0.16	COCO 2014	300	1078	97	104.6
DINO-4scale	1	0.63	COCO 2017	12	248	94	23.3
Faster R-CNN (R101-C4)	2	0.62	COCO 2014	37	512	105	53.8

* Batch size for YOLOv4 is configured as 64 but due to subdivision of 64, effective batch size is 1. Faster R-CNN is implemented using Detectron2 library. COCO 2014 training set is comprised of ∼83,000 images whereas COCO 2017 is ∼118,000.

largely due to the high number of epochs (300) required for training and relative slower training speed compared to YOLOv7. Faster R-CNN performs the worst in terms of accuracy and inference speed, however the training time and subsequent energy required to complete training is almost half of that of YOLOv4, which offers similar levels of accuracy but with almost 10 times faster inference speeds. YOLOv7 is the most accurate and faster model, however, the training time and energy cost is almost 4 times greater than that of the comparably accurate DINO, but similarly to YOLOv4 and Faster R-CNN, YOLOv7 is nearly 7 times faster and inference.

The results in Table 2 show very clearly that training any of the four tested object detection models fully requires significant GPU compute power. Training these models on a single RTX 3050 would take far too long to be practical with the shortest training time of DINO-4scale lasting an estimated 248 hours. Additionally, the due to the 8GB memory limit of the RTX 3050, batch size must be greatly reduced from the default recommendation for each model's training, likely leading to small decreases in the final AP even if the model was training to completion.

The average power consumption of each model during training is quite similar, with a ranging from 94W for DINO to 105W for Faster R-CNN (R101-C4). The difference in power usage despite all models being training in order to maximise the performance on the RTX 3050 is likely due to differences in model size, complexity and efficiency.

VI. CONCLUSION

State of the art object detection models have improved in both performance and energy efficiency, with YOLOv7 being significantly more accurate and faster than the once state of the art YOLOv4, and also being more energy efficient to train. DINO is similarly far more accurate and faster than Faster R-CNN while being over twice as energy efficient to train, showing the evolution of design in terms of performance and energy consumption in modern state of the art object detection models. Although more computing power is available to modern models that certainly benefit from this advantage, the use of this power is not strictly necessary, though it has made object detection training and deployment much more practical for many users. However, the compute requirements to effectively train object detection models in a realistic time frame has not significantly reduced and high performing, high cost GPU devices are still essential for pragmatic use. Cloud computing solutions offer potential solutions in terms of accessibility but still incur the same energy costs overall.

VII. FUTURE WORK

Future work into this area could examine a wider range of models and attempt to pinpoint the highest energy cost components of these models. Access to similar GPU hardware that the original models are trained on would allow for full training and complete power usage measurements instead of estimations, particularly as some models stop training after a certain number of epochs to change configuration in order to start training again to refine for a small number of epochs. Hardware tools would allow for more precise energy usage measurements of the GPU instead of purely software sensor readings. Establishing a clear guide for training time vs performance for all leading models and their derivatives could be beneficial for many users where training time could be a crucial consideration. This information is largely not officially available for older models though more recent algorithms such as YOLOv7 [36] and DINO [39] do provide good examples of how long their models were trained for. Another consideration for training energy costs is the pre-training phase for the network backbone that is often training on the ImageNet dataset of over 1 million images [23].

REFERENCES

[1] Sahla Muhammed Ali. Comparative Analysis of YOLOv3, YOLOv4 and YOLOv5 for Sign Language Detection. *Internation Journal Of Advance Research And Innovative Ideas In Education*, 7(4):2393–2398, 2021. Publisher: IJARIIE.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[3] Raphael Brüngel and Christoph M. Friedrich. Detr and yolov5: Exploring performance and self-training for diabetic foot ulcer detection. In *2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 148–153, 2021.

[4] Yuanqiang Cai, Longyin Wen, Libo Zhang, Dawei Du, and Weiqiang Wang. Rethinking Object Detection in Retail Stores. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(2):947–954, May 2021.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

[6] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2722–2730, 2015.

[7] Sylvain Collange, David Defour, and Arnaud Tisserand. Power Consumption of GPUs from a Software Perspective. volume 5544, May 2009.

[8] Sylvain Collange, David Defour, and Arnaud Tisserand. Power consumption of gpus from a software perspective. volume 5544, 05 2009.

[9] Lixuan Du, Rongyu Zhang, and Xiaotian Wang. Overview of two-stage object detection algorithms. *Journal of Physics: Conference Series*, 1544(1):012033, May 2020. Publisher: IOP Publishing.

[10] Chu Yikaiheng et al. YOLOv6: A fast and accurate target detection framework is open source, 2022. [Online; accessed 29-July-2022].

[11] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[12] Fernando Fernandes dos Santos, Lucas Draghetti, Lucas Weigel, Luigi Carro, Philippe Navaux, and Paolo Rech. Evaluation and Mitigation of Soft-Errors in Neural Network-Based Object Detection in Three GPU Architectures. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 169–176, 2017.

[13] Mariza Ferro, André Yokoyama, Gabrieli Silva, Ricardo Braganca, Rodrigo Gandra, André Bulcão, and Bruno Schulze. Analysis of gpu power consumption using internal sensors. 01 2017.

[14] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, 2016.

[17] Bo Gong, Daji Ergu, Ying Cai, and Bo Ma. A Method for Wheat Head Detection Based on Yolov4. 2020.

[18] J. Yu, K. Guo, Y. Hu, X. Ning, J. Qiu, H. Mao, S. Yao, T. Tang, B. Li, Y. Wang, and H. Yang. Real-time object detection towards high power efficiency. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 704–708, March 2018. Journal Abbreviation: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE).

[19] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. A Review of Yolo Algorithm Developments. *Procedia Computer Science*, 199:1066–1073, 2022.

[20] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A Survey of Deep Learning-Based Object Detection. *IEEE Access*, 7:128837–128868, 2019.

[21] Loïc Lannelongue, Jason Grealey, and Michael Inouye. Green Algorithms: Quantifying the Carbon Footprint of Computation. *Advanced Science*, 8(12):2100707, June 2021. Publisher: John Wiley & Sons, Ltd.

[22] Feng Li, Hao Zhang, Shilong Liu, Jian Guo, Lionel M. Ni, and Lei Zhang. Dn-detr: Accelerate detr training by introducing query denoising, 2022.

[23] Hengduo Li, Bharat Singh, Mahyar Najibi, Zuxuan Wu, and Larry S. Davis. An analysis of pre-training on object detection, 2019.

[24] Matthieu Lin, Chuming Li, Xingyuan Bu, Ming Sun, Chen Lin, Junjie Yan, Wanli Ouyang, and Zhidong Deng. Detr for crowd pedestrian detection, 2020.

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016.

[26] Ángel Morera, Ángel Sánchez, A. Belén Moreno, Ángel D. Sappa, and José F. Vélez. SSD vs. YOLO for Detection of Outdoor Urban Advertising Panels under Multiple Variabilities. *Sensors*, 20(16), 2020.

[27] Upesh Nepal and Hossein Eslamiat. Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. *Sensors*, 22(2), 2022.

[28] Praagna Prasad, Akhil Chawla, and Mohana. Facemask Detection to Prevent COVID-19 Using YOLOv4 Deep Learning Model. In *2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, pages 382–388, 2022.

[29] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. June 2016. Pages: 788.

[30] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

[31] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, June 2017.

[33] Manasvi Sagarkar, Shreyas More, Akash Singh, Prithwish Jana, and Bithika Pal. Perception and planning in autonomous car, 12 2020.

[34] Jacob Solawetz. YOLOv7 Breakdown, 2022. [Online; accessed 29-July-2022].

[35] Farhana Sultana, Abu Sufian, and Paramartha Dutta. A review of object detection models based on convolutional neural network. *CoRR*, abs/1905.01614, 2019.

[36] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.

[37] Ying Wang, Zhenyu Quan, Jiajun Li, Yinhe Han, Huawei Li, and Xiaowei Li. A retrospective evaluation of energy-efficient object detection solutions on embedded devices. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 709–714, 2018.

[38] Zhuyu Yao, Jiangbo Ai, Boxun Li, and Chi Zhang. Efficient detr: Improving end-to-end object detector with dense prior, 2021.

[39] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection, 2022.

[40] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.