A Survey of Modern Deep Learning based Object Detection Models

Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam,
Nadia Kanwal, Mamoona Asghar, and Brian Lee

***Abstract*—Object Detection is the task of classification and localization of objects in an image or video. It has gained prominence in recent years due to its widespread applications. This article surveys recent developments in deep learning based object detectors. Concise overview of benchmark datasets and evaluation metrics used in detection is also provided along with some of the prominent backbone architectures used in recognition tasks. It also covers contemporary lightweight classification models used on edge devices. Lastly, we compare the performances of these architectures on multiple metrics.**

***Index Terms*—Object detection and recognition, convolutional neural networks (CNN), lightweight networks, deep learning**

I. Introduction

Object detection is a trivial task for humans. A few months old child can start recognizing common objects, however teaching it to the computer has been an uphill task until the turn of the last decade. It entails identifying and localizing all instances of an object (like cars, humans, street signs, etc.) within the field of view. Similarly, other tasks like classification, segmentation, motion estimation, scene understanding, etc, have been the fundamental problems in computer vision.

Early object detection models were built as an ensemble of hand-crafted feature extractors such as Viola-Jones detector [1], Histogram of Oriented Gradients (HOG) [2] etc. These models were slow, inaccurate and performed poorly on unfamiliar datasets. The re-introduction of convolutional neural network (CNNs) and deep learning for image classification changed the landscape of visual perception. Its use in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 challenge by AlexNet [3] inspired further research of its application in computer vision. Today, object detection finds application from self-driving cars and identity detection to security and medical uses. In recent years, it has seen exponential growth with rapid development of new tools and techniques.

This survey provides a comprehensive review of deep learning based object detectors and lightweight classification architectures. While existing reviews are quite thorough [4]–[7], most of them lack new developments in the domain. The main contributions of this paper are as follows:

S.S.A. Zaidi, N. Kanwal, M Asghar and B. Lee are with the Athlone Institute of Technology, Ireland. M.S. Ansari is with the Aligarh Muslim University, India. A. Aslam is with the Insight Center for Data Analytics, National University of Ireland, Galway. (Emails: sahilzaidi78@gmail.com, samar.ansari@zhect.ac.in, asra.aslam@insight-centre.org, nkanwal@ait.ie, masghar@ait.ie, blee@ait.ie)

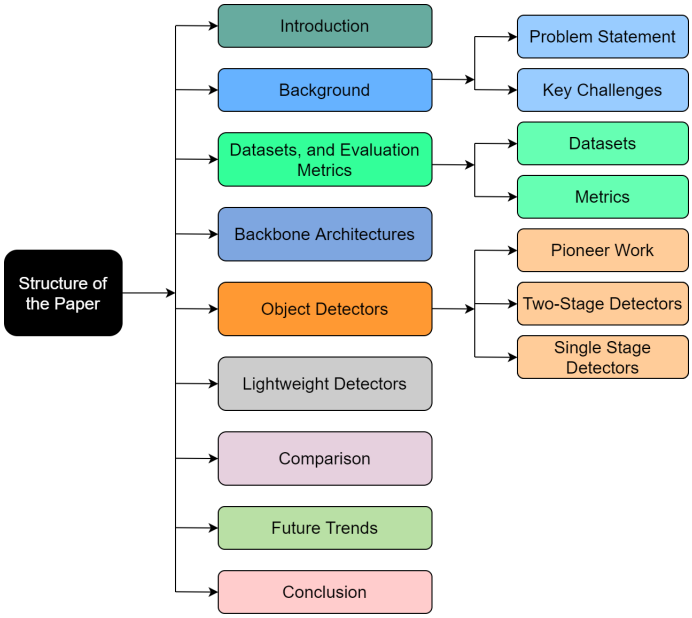

Fig. 1: Structure of the paper.

1) This paper provides an in-depth analysis of major object detectors in both categories – single and two stage detectors. Furthermore, we take historic look at the evolution of these methods.
2) We present a detailed evaluation of the landmark backbone architectures and lightweight models. We could not find any paper which provides a broad overview of both these topics.

In this paper, we have systematically reviewed various object detection architectures and its associated technologies, as illustrated in figure 1. Rest of this paper is organized as follows. In section II, the problem of object detection and its associated challenges are discussed. Various benchmark datasets and evaluation metrics are listed in Section III. In Section IV, several milestone backbone architectures used in modern object detectors are examined. Section V is divided into three major sub-section, each studying a different category of object detectors. This is followed by the analysis of a special classification of object detectors, called lightweight networks in section VI and a comparative analysis in Section VII. The future trends are mentioned in Section VIII while the paper is concluded in Section IX.

(a) PASCAL VOC 12 (b) MS-COCO (c) ILSVRC (d) OpenImage

Fig. 2: Sample images from different datasets.

II. Background

A. Problem Statement

The object detection is the natural extension of object classification, which aims only at recognizing the object in the image. The goal of the object detection is to detect all instances of the predefined classes and provide its coarse localization in the image by axis-aligned boxes. The detector should be able to identify all instances of the object classes and draw bounding box around it. It is generally seen as a supervised learning problem. Modern object detection models have access to large sets of labelled images for training and are evaluated on various canonical benchmarks.

B. Key challenges in Object Detection

Computer vision has come a long way in the past decade, however it still has some major challenges to overcome. Some of these key challenges faced by the networks in real life applications are:

- *Intra class variation* : Intra class variation between the instances of same object is relatively common in nature. This variation could be due to various reasons like occlusion, illumination, pose, viewpoint, etc. These unconstrained external can have dramatic effect of the object appearance [5]. It is expected that the objects could have non-rigid deformation or be rotated, scaled or blurry. Some objects could have inconspicuous surroundings, making the extraction difficult.
- *Number of categories*: The sheer number of object classes available to classify makes it a challenging problem to solve. It also requires more high-quality annotated data, which is hard to come by. Using fewer examples for training a detector is an open research question.
- *Efficiency*: Present day models need high computation resources to generate accurate detection results. With mobile and edge devices becoming common place, efficient object detectors are crucial for further development in the field of computer vision.

III. Datasets and Evaluation Metrics

A. Datasets

This section presents an overview of the datasets that are available, and have been most commonly used for object detection tasks.

1) PASCAL VOC 07/12: The Pascal Visual Object Classes (VOC) challenge was a multiyear effort to accelerate the development in the field of visual perception. It started in 2005 with classification and detection tasks on four object classes [8], but two versions of this challenges are mostly used as a standard benchmark. While the VOC07 challenge had 5k training images and more than 12k labelled objects [9], the VOC12 challenge increased them to 11k training images and more than 27k labelled objects [10]. Object classes was expanded to 20 categories and the tasks like segmentation and action detection were included as well. Pascal VOC introduced the mean Average Precision (*mAP*) at 0.5 IoU (Intersection over Union) to evaluate the performance of the models. Figure 3 depicts the distribution of the number of images w.r.t. to the different classes in the Pascal VOC dataset.

2) ILSVRC: The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [11] was an annual challenge running from 2010 to 2017 and became a benchmark for evaluating algorithm performance. The dataset size was scaled up to more than a million images consisting of 1000 object classification classes. 200 of these classes were hand-picked for object detection task, constitute of more than 500k images. Various sources including ImageNet [12] and Flikr, were used to construct detection dataset. ILSVRC also updated the evaluation metric by loosening the IoU threshold to help include smaller object detection. Figure 4 depicts the distribution of the number of images w.r.t. to the different classes in the ImageNet dataset.

3) MS-COCO: The Microsoft Common Objects in Context (MS-COCO) [13] is one of the most challenging datasets available. It has 91 common objects found in their natural context which a 4-year-old human can easily recognize. It was launched in 2015 and its popularity has only increased since then. It has more than two million instances and an

average of 3.5 categories per images. Furthermore, it contains 7.7 instances per image, comfortably more than other popular datasets. MS COCO comprises of images from varied viewpoints as well. It also introduced a more stringent method to measure the performance of the detector. Unlike the Pascal VOC and ILSVCR, it calculates the IoU from 0.5 to 0.95 in steps of 0.5, then using a combination of these 10 values as final metric, called Average Precision (AP). Apart from this, it also utilizes AP for small, medium and large objects separately to compare performance at different scales. Figure 5 depicts the distribution of the number of images w.r.t. to the different classes in the MS-COCO dataset.

4) Open Image: Google's Open Images [14] dataset is composed of 9.2 million images, annotated with image-level labels, object bounding boxes, and segmentation masks, among others. It was launched in 2017 and has received six updates. For object detection, Open Images has 16 million bounding boxes for 600 categories on 1.9 million images, which makes it the largest dataset of object localization. Its creators took extra care to choose interesting, complex and diverse images, having 8.3 object categories per image. Several changes were made to the AP introduced in Pascal VOC like ignoring un-annotated class, detection requirement for class and its subclass, etc. Figure 6 depicts the distribution of the number of images w.r.t. to the different classes in the Open Images dataset.

5) Issues of Data Skew/Bias: While observing Fig. 3 through Fig. 6, an alert reader would certainly notice that the number of images for difference classes vary significantly in all the datasets [15]. Three (Pascal VOC, MS-COCO, and Open Images Dataset) of the four datasets discussed above have a very significant drop in the number of images beyond the top-5 most frequent classes. As can be readily observed for Fig. 3, there are 13775 images which contain a 'person' and then 2829 images which contain a 'car'. The number of images for the remaining 18 classes in this dataset almost fall linearly to the 55 images of 'sheep'. Similarly, for the MS-COCO dataset, the class 'person' has 262465 images, and the next most-frequent class 'car' has 43867 images. The downward trend continues till there are only 198 images for the class 'hair drier'. A similar phenomenon is also observed in the Open Images Dataset, wherein the class 'Man' is the most frequent with 378077 images, and the class 'Paper Cutter' has only 3 images. This clearly represents a skew in the datasets and is bound to create a bias in the training process of any object detection model. Therefore, an object detection model trained on these skewed datasets will in all probability show better detection performance for the classes with more number of images in the training data. Although still present, this issue is slightly less pronounced in the ImageNet dataset, as can be observed from Fig. 4 from where it can be seen that the most frequent class i.e. 'koala' has 2469 images, and the least frequent class i.e. 'cart' has 624 images. However, this leads to another point of concern in the ImageNet dataset: the most frequent class is for 'koala' and the next most-appearing class is 'computer keyboard', which are clearly not the most sought after objects in a real-world object detection scenario (where person, cars, traffic signs, etc. are of higher concern).

B. Metrics

Object detectors use multiple criteria to measure the performance of the detectors viz., frames per second (FPS), precision and recall. However, mean Average Precision (*mAP*) is the most common evaluation metric. Precision is derived from Intersection over Union (IoU), which is the ratio of the area of overlap and the area of union between the ground truth and the predicted bounding box. A threshold is set to determine if the detection is correct. If the IoU is more than the threshold, it is classified as True Positive while an IoU below it is classified as False Positive. If the model fails to detect an object present in the ground truth, it is termed as False Negative. Precision measures the percentage of correct predictions while the recall measure the correct predictions with respect to the ground truth.

$$\begin{aligned} Precision &= \frac{True\ Positive}{True\ Positive + False\ Positive} \\ &= \frac{True\ Positive}{All\ Observations} \end{aligned} \tag{1}$$

$$\begin{aligned} Recall &= \frac{True\ Positive}{True\ Positive + False\ Negative} \\ &= \frac{True\ Positive}{All\ Ground\ Truth} \end{aligned} \tag{2}$$

Based on the above equation, average precision is computed separately for each class. To compare performance between the detectors, the mean of average precision of all classes, called mean average precision (*mAP*) is used, which acts as a single metric for final evaluation.

IV. Backbone architectures

Backbone architectures are one of the most important component of the object detector. These networks extract feature from the input image used by the model. Here, we have discussed some milestone backbone architectures used in modern detectors:

A. AlexNet

Krizhevsky et al. proposed AlexNet [3], a convolutional neural network based architecture for image classification, and won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012 challenge. It achieved a considerably higher accuracy (more than 26%) than the contemporary models. AlexNet is composed of eight learnable layers - five convolutional and three fully connected layers. The last layer of the fully connected layer is connected to an *N-way* (*N*: number of classes) softmax classifier. It uses multiple convolutional kernels throughout the network to obtain features from the image. It also uses dropout and ReLU for regularization and faster training convergence respectively. The convolutional neural networks were given a new life by its reintroduction in AlexNet and it soon became the go-to technique in processing imaging data.

TABLE I: Comparison of various object detection datasets.

Dataset	Classes	Train			Validation			Test
		Images	Objects	Objects/Image	Images	Objects	Objects/Image	
PASCAL VOC 12	20	5,717	13,609	2.38	5,823	13,841	2.37	10,991
MS-COCO	80	118,287	860,001	7.27	5,000	36,781	7.35	40,670
ILSVRC	200	456,567	478,807	1.05	20,121	55,501	2.76	40,152
OpenImage	600	1,743,042	14,610,229	8.38	41,620	204,621	4.92	125,436

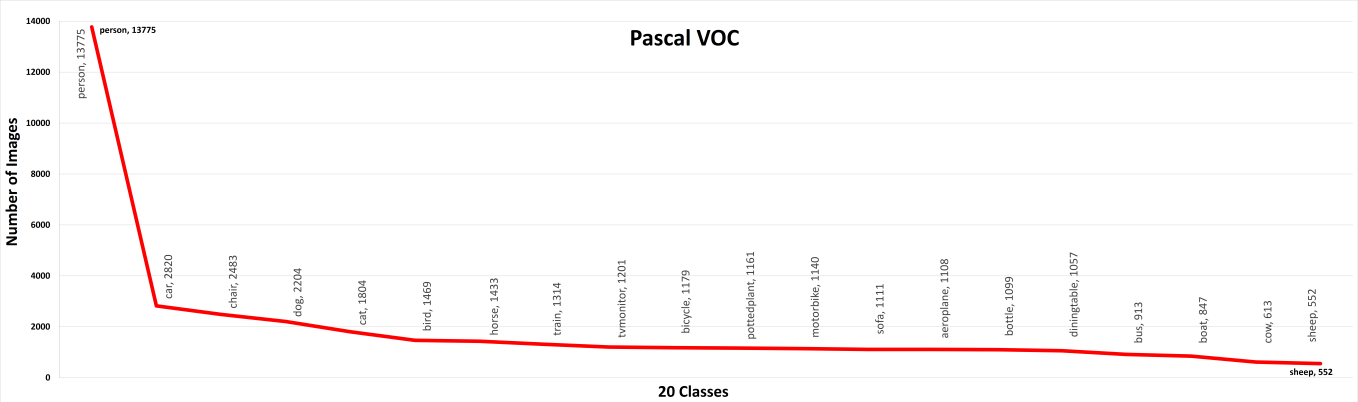

Fig. 3: (This image is best viewed in PDF form with magnification) Number of images for different classes annotated in the PascalVOC dataset [15]

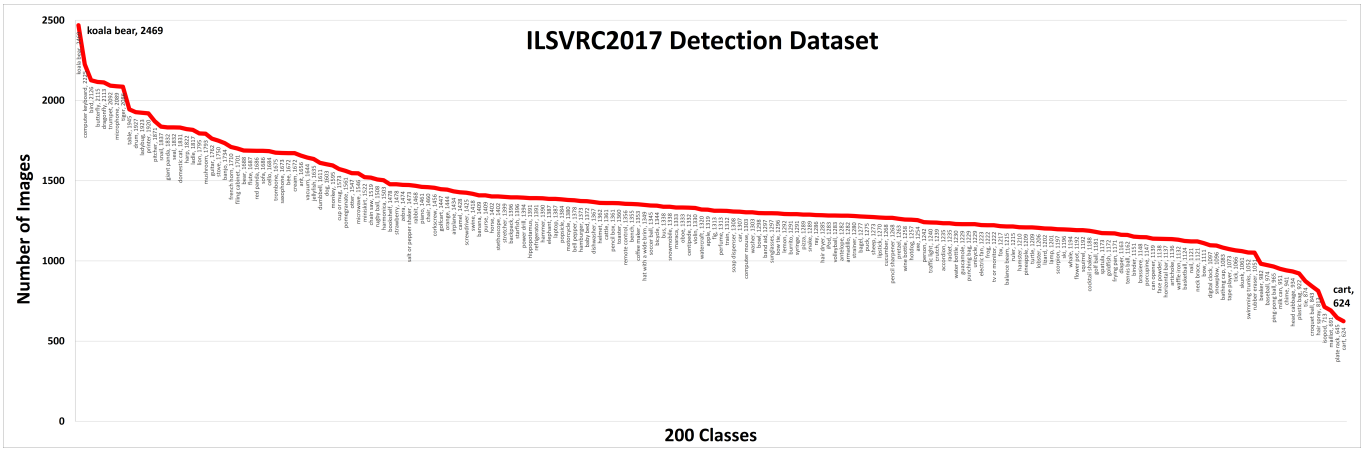

Fig. 4: (This image is best viewed in PDF form with magnification) Number of images for different classes annotated in the ImageNet dataset [15]

B. VGG

While AlexNet [3] and its successors like [16] focused on smaller receptive window size to improve accuracy, Simonyan and Zisserman investigated the effects of network depth on it. They proposed VGG [17], which used small convolution filters to construct networks of varying depths. While a larger receptive field can be captured by a set of smaller convolutional filters, it drastically reduces network parameters and converges sooner. The paper demonstrated how deep network architecture (16-19 layers) can be used to perform classification and localization with superior accuracy. VGG was created by adding a stack of convolutional layers with three fully connected layers, followed by a softmax layer. The number of convolutional layers, according to the authors, can vary from 8 to 16. VGG is trained in multiple iterations; first, the smallest 11-layer architecture is trained with random initialization whose weights are then used to train larger networks to prevent gradient instability. VGG outperformed ILSVRC 2014 winner GoogLeNet [18] in the single network performance category. It soon became one of the most used network backbones for object classification and detection models.

C. GoogLeNet/Inception

Even though classification networks were making inroads towards faster and more accurate networks, deploying them in real-world applications was still a long way off as they were resource-intensive. As networks are scaled for better performance, the computation cost increases exponentially. Szegedy et al. in [18] postulated the wastage of computations in the network as a major reason for it. Bigger models also have a large number of parameters and tend to overfit the

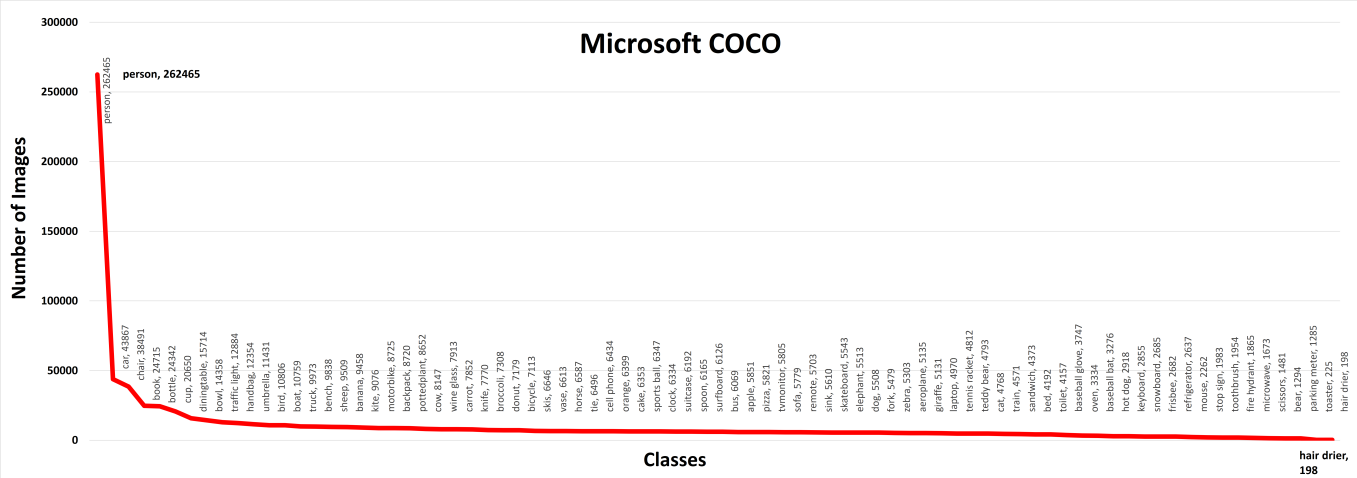

Fig. 5: (This image is best viewed in PDF form with magnification) Number of images for different classes annotated in the MS-COCO dataset [15]

data. They proposed using locally sparse connected architecture instead of a fully connected one to solve these issues. GoogLeNet is thus a 22 layer deep network, made up by stacking multiple Inception modules on top of each other. Inception modules are networks that have multiple sized filters at the same level. Input feature maps pass through these filters and are concatenated and forwarded to the next layer. The network also has auxiliary classifiers in the intermediate layers to help regularize and propagate gradient. GoogLeNet showed how efficient use of computation blocks can perform at par with other parameter-heavy networks. It achieved 93.3% top-5 accuracy on ImageNet [11] dataset without external data, while being faster than other contemporary models. Updated versions of Inception like [19], [20] were also published in the following years which further improved its performance and gave further evidence of the applications of refined sparsely connected architectures.

D. *ResNets*

As convolutional neural networks become deeper and deeper, Kaiming He et al. in [21] showed how their accuracy first saturates and then degrades rapidly. They proposed the use of residual learning to the stacked layers to mitigate the performance decay. It is realized by addition of a skip connection between the layers. This connection is an element-wise addition between input and output of the block and does not add extra parameter or computational complexity to the network. A typical 34 layer ResNet [21] is basically a large (*7x7*) convolution filter followed by 16 bottleneck modules (pair of small *3x3* filters with identity shortcut across them) and ultimately a fully connected layer. The bottleneck architecture can be adapted for deeper networks by stacking 3 convolutional layers (*1x1,3x3,1x3*) instead of 2. Kaiming He et al. also demonstrated how the 16-layer VGG net had higher complexity than their considerably deeper 101 and 152 layer ResNet architectures while having lower accuracy. In subsequent paper, the authors proposed Resnetv2 [22] which used batch normalization and ReLU layer in the blocks. It is more generalized and easier to train. ResNets are widely used in classification and detection backbones, and its core principles have inspired many networks ([20], [23], [24]).

TABLE II: Comparison of Backbone architectures.

Model	Year	Layers	Parameters (Million)	Top-1 acc%	FLOPs (Billion)
AlexNet	2012	7	62.4	63.3	1.5
VGG-16	2014	16	138.4	73	15.5
GoogLeNet	2014	22	6.7	-	1.6
ResNet-50	2015	50	25.6	76	3.8
ResNeXt-50	2016	50	25	77.8	4.2
CSPResNeXt-50	2019	59	20.5	78.2	7.9
EfficientNet-B4	2019	160	19	83	4.2

E. *ResNeXt*

The existing conventional methods of improving the accuracy of a model were by either increasing the depth or the width of the model. However, increasing any of these leads to higher model complexity and number of parameters while the gain margins diminish rapidly. Xie et al. introduced ResNeXt [24] architecture which is simpler and more efficient than other existing models. ResNeXt was inspired by the stacking of similar blocks in VGG/ResNet [3], [21] and "split-transform-merge" behavior of Inception module [18]. It is essentially a ResNet where each ResNet block is replaced by an inception-like ResNeXt module. The complicated, tailored transformation modules from the Inception is replaced by topologically same modules in the ResNeXt blocks, making the network easier to scale and generalize. Xie et al. also emphasize that the cardinality (topological paths in the ResNeXt block) can be considered as a third dimension, along with depth and width, to improve model accuracy. ResNeXt is elegant and more concise. It achieved higher accuracy while having considerably fewer hyperparameters than a similar depth ResNet architecture. It was also the first runner up to the ILSVRC 2016 challenge.

Fig. 6: (This image is best viewed in PDF form with magnification) Number of images for different classes annotated in the Open Images dataset [15]

F. CSPNet

Existing neural networks have shown incredible results in achieving high accuracy in computer vision tasks; however, they rely on excessive computational resources. Wang et al. believe that heavy inference computations can be reduced by cutting down the duplicate gradient information in the network. They proposed CSPNet [25] which creates different paths for the gradient flow within the network. CSPNet separates feature maps at the base layer into two parts. One part is passed through the partial convolution network block (e.g., Dense and Transition block in DenseNet [23] or Res(X) block in ResNeXt [24]) while the other part is combined with its outputs at a later stage. This reduces the number of parameters, increases the utilization of computation units and eases memory footprint. It is easy to implement and general enough to be applicable on other architectures like ResNet [21], ResNeXt [24], DenseNet [23], Scaled-YOLOv4 [26] etc. Applying CSPNet on these networks reduced computations from 10% to 20%, while the accuracy remained constant or improved. Memory cost and computational bottleneck is also reduced significantly with this method. It is leveraged in many state of the art detector models, while also being used for mobile and edge devices.

G. EfficientNet

Tan et al. systematically studied network scaling and its effects on the model performance. They summarized how altering network parameters like depth, width and resolution influence its accuracy. Scaling any parameter individually comes with an associated cost. Increasing depth of a network can help in capturing richer and more complex features, but they are difficult to train due to vanishing gradient problem. Similarly, scaling network width will make it easier to capture fine grained features but have difficulty in obtaining high level features. Gains from increasing the image resolution, like depth and width, saturate as model scales. In the paper [27], Tan et al. proposed the use of a compound coefficient that can uniformly scale all three dimensions. Each model parameter has an associated constant, which is found by fixing the coefficient as 1 and performing a grid search on a baseline network. The baseline architecture, inspired by their previous work [28], is developed by neural architecture search on a search target while optimizing accuracy and computations. EfficientNet is a simple and efficient architecture. It outperformed existing models in accuracy and speed while being considerably smaller. By providing a monumental increase in efficiency, it could potentially open a new era in the field of

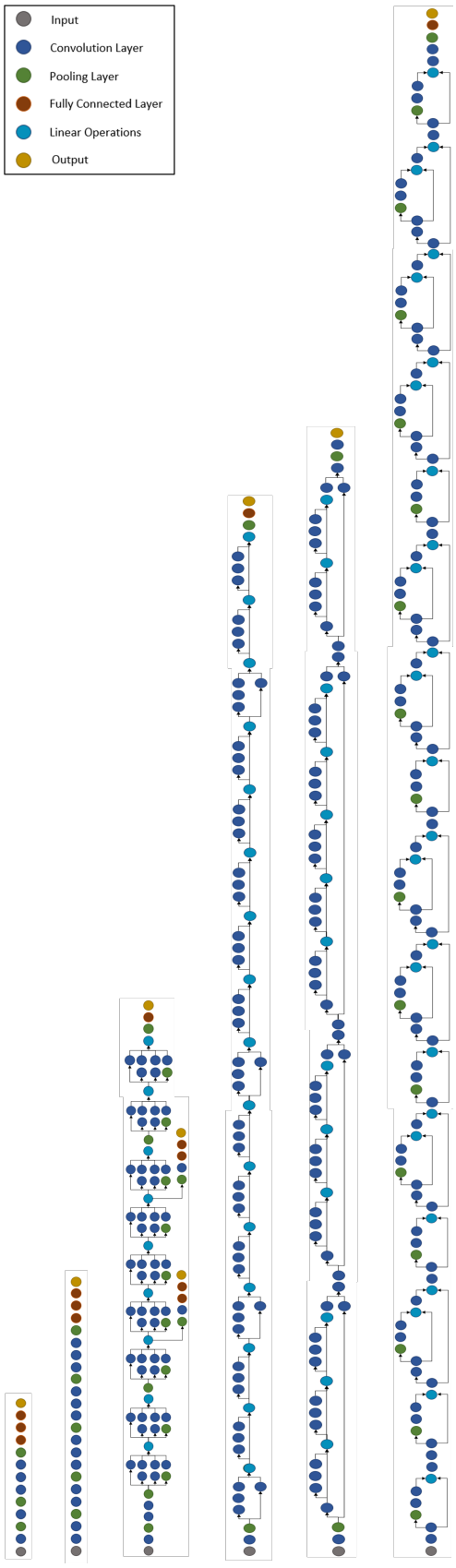

Fig. 7: Visualization of CNN Architectures[1]. Left to Right: AlexNet, VGG−16, GoogLeNet, ResNet−50, CSPResNeXt−50, EfficientNet−B4.

efficient networks.

V. Object Detectors

We have divided this review based on the two types of detectors — two-stage and single-stage detectors. However, we also discussed the pioneer work, where we briefly examine a few traditional object detectors. A network which has a separate module to generate region proposals is termed as a two-stage detector. These models try to find an arbitrary number of objects proposals in an image during the first stage and then classify and localize them in the second. As these systems have two separate steps, they generally take longer to generate proposals, have complicated architecture and lacks global context. Single-stage detectors classify and localize semantic objects in a single shot using dense sampling. They use predefined boxes/keypoints of various scale and aspect ratio to localize objects. It edges two-stage detectors in real-time performance and simpler design.

A. Pioneer Work

1) Viola-Jones: Primarily designed for face detection, Viola-Jones object detector [1], proposed in 2001, was an accurate and powerful detector. It combined multiple techniques like Haar-like features, integral image, Adaboost and cascading classifier. First step is to search for Haar-like features by sliding a window on the input image and uses integral image to calculate. It then uses a trained Adaboost to find the classifier of each haar feature and cascades them. Viola Jones algorithm is still used in small devices as it is very efficient and fast.

2) HOG Detector: In 2005, Dalal and Triggs proposed the Histogram of Oriented Gradients (HOG) [2] feature descriptor used to extract features for object detection. It was an improvement over other detectors like [29]–[32]. HOG extracts gradient and its orientation of the edges to create a feature table. The image is divided into grids and the feature table is then used to create histogram for each cell in the grid. HOG features are generated for the region of interest and fed into a linear SVM classifier for detection. The detector was proposed for pedestrian detection; however, it could be trained to detect various classes.

3) DPM: Deformable Parts Model (DPM) [33] was introduced by Felzenszwalb et al. and was the winner Pascal VOC challenge in 2009. It used individual "part" of the object for detection and achieved higher accuracy than HOG. It follows the philosophy of *divide and rule*; parts of the object are individually detected during inference time and a probable arrangement of them is marked as detection. For example, a human body can be considered as a collection of parts like head, arms, legs and torso. One model will be assigned to capture one of the parts in the whole image and the process is repeated for all such parts. A model then removes improbable configurations of the combination of these parts to produce detection. DPM based models [34], [35] were one of the most successful algorithms before the era of deep learning.

[1]Tool Used: https://netron.app/

B. Two-Stage Detectors

1) ***R-CNN:*** The Region-based Convolutional Neural Network (R-CNN) [36] was the first paper in the R-CNN family, and demonstrated how CNNs can be used to immensely improve the detection performance. R-CNN use a class agnostic region proposal module with CNNs to convert detection into classification and localization problem. A mean-subtracted input image is first passed through the region proposal module, which produces 2000 object candidates. This module find parts of the image which has a higher probability of finding an object using Selective Search [37]. These candidates are then warped and propagated through a CNN network, which extracts a 4096-dimension feature vector for each proposal. Girshick et al. used AlexNet [3] as the backbone architecture of the detector. The feature vectors are then passed to the trained, class-specific Support Vector Machines (SVMs) to obtain confidence scores. Non-maximum suppression (NMS) is later applied to the scored regions, based on its IoU and class. Once the class has been identified, the algorithm predicts its bounding box using a trained bounding-box regressor, which predicts four parameters i.e., center coordinates of box along with its width and height.

R-CNN has a complicated multistage training process. The first stage is pre-training the CNN with a large classification dataset. It is then fine-tuned for detection using domain-specific images (mean-subtracted, warped proposals) by replacing of the classification layer with a randomly initialized *N+1*-way classifier, *N* being the number of classes, using stochastic gradient descent (SGD) [38]. One liner SVM and bounding box regressor is trained for each class.

R-CNN ushered a new wave in the field of object detection, but it was slow (47 sec per image) and expensive in time and space [39]. It had complex training process and took days to train on small datasets even when some of the computations were shared.

2) ***SPP-Net:*** He et al. proposed the use of Spatial Pyramid Pooling (SPP) layer [40] to process image of arbitrary size or aspect ratio. They realized that only the fully connected part of the CNN required a fixed input. SPP-net [41] merely shifted the convolution layers of CNN before the region proposal module and added a pooling layer, thereby making the network independent of size/aspect ratio and reducing the computations. The selective search [37] algorithm is used to generate candidate windows. Feature maps are obtained by passing the input image through the convolution layers of a ZF-5 [16] network. The candidate windows are then mapped on to the feature maps, which are subsequently converted into fixed length representations by spatial bins of a pyramidal pooling layer. This vector is passed to the fully connected layer and ultimately, to SVM classifiers to predict class and score. Similar to R-CNN [36], SPP-net has as post processing layer to improve localization by bounding box regression. It also uses the same multistage training process, except that the fine tuning is done only on the fully connected layers.

SPP-Net is considerably faster than the R-CNN model with comparable accuracy. It can process images of any shape/aspect ratio and thus, avoid object deformation due to input warping. However, as its architecture is analogous to R-CNN, it shared R-CNN's disadvantages too like multistage training, computationally expensive and training time as well.

3) ***Fast R-CNN:*** One of the major issues with R-CNN/SPP-Net was the need to train multiple systems separately. Fast R-CNN [39] solved this by creating a single end-to-end trainable system. The network takes as input an image and its object proposals. The image is passed through a set of convolution layers and the object proposals are mapped to the obtained feature maps. Girshick replaced pyramidal structure of pooling layers from SPP-net [41] with a single spatial bin, called RoI pooling layer. This layer is connected to 2 fully connected layer and then branches out into a *N+1*-class SoftMax layer and a bounding box regressor layer, which has a fully connected layer as well. The model also changed the loss function of bounding box regressor from L2 to smooth L1 to better performance, while introducing a multi-task loss to train the network.

The authors used modified version of existing state-of-art pre-trained models like [3], [17] and [42] as backbone. The network was trained in a single step by stochastic gradient descent (SGD) and a mini-batch of 2 images. This helped the network converge faster as the back-propagation shared computations among the RoIs from the two images.

Fast R-CNN was introduced as an improvement in speed (146x on R-CNN) while the increase in accuracy was supplementary. It simplified training procedure, removed pyramidal pooling and introduces a new loss function. The object detector, without the region proposal network, reported near real time speed with considerable accuracy.

4) ***Faster R-CNN:*** Even though Fast R-CNN inched closer to real time object detection, its region proposal generation was still an order of magnitude slower (2 sec per image compared to 0.2 sec per image). Ren et al. suggested a fully convoluted network [43] as a region proposal network (RPN) in [44] that takes an arbitrary input image and outputs a set of candidate windows. Each such window has an associated *objectness score* which determines likelihood of an object. Unlike its predecessors like [21], [34], [39] which used image pyramids to solve size variance of objects, RPN introduces Anchor boxes. It used multiple bounding boxes of different aspect ratios and regressed over them to localize object. The input image is first passed through the CNN to obtain a set of feature maps. These are forwarded to the RPN, which produces bounding boxes and their classification. Selected proposals are then mapped back to the feature maps obtained from previous CNN layer in RoI pooling layer, and ultimately fed to fully connected layer, which is sent to classifier and bounding box regressor. Faster R-CNN is essentially Fast R-CNN with RPN as region proposal module.

Training of Faster R-CNN is more convoluted, due to the presence of shared layers between two models which perform very different tasks. Firstly, RPN is pre-trained on ImageNet dataset [12] and fine-tuned on PASCAL VOC dataset [8]. A Fast R-CNN is trained from the region proposals of RPN from first step. Till this point, the networks do not have shared convolution layer. Now, we fix the convolution layers of the detector and fine-tune the unique layers in RPN. And finally,

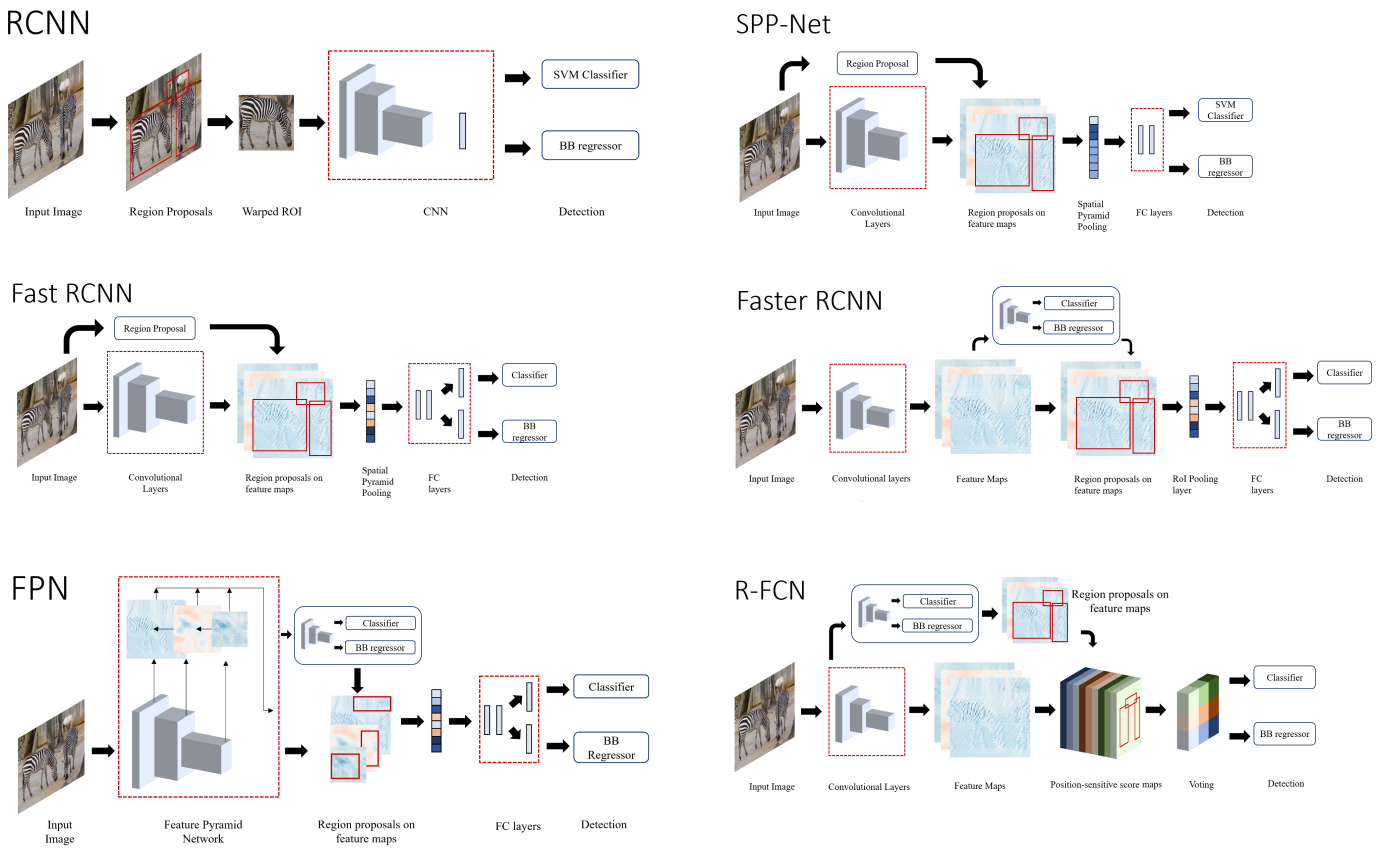

Fig. 8: Illustration of the internal architecture of different two stage object detectors[2].

Fast R-CNN is fine-tuned from the updated RPN.

Faster R-CNN improved the detection accuracy over the previous state-of-art [39] by more than 3% and decreased inference time by an order of magnitude. It fixed the bottleneck of slow region proposal and ran in near real time at 5 frames per second. Another advantage of having a CNN in region proposal was that it could learn to produce better proposals and thereby increase accuracy.

5) ***FPN****:* Use of image pyramid to obtain feature pyramid (or *featurized image pyramids*) at multiple levels is a common method to increase detection of small objects. Even though it increases Average Precision of the detector, the increase in the inference time is substantial. Lin et al. proposed the Feature Pyramid Network (FPN) [45], which has a top-down architecture with lateral connections to build high-level semantic features at different scales. The FPN has two pathways, a bottom-up pathway which is a ConvNet computing feature hierarchy at several scales and a top-down pathway which upsamples coarse feature maps from higher level into high-resolution features. These pathways are connected by lateral connection by a *1x1* convolution operation to enhance the semantic information in the features. FPN is used as a region proposal network (RPN) of a ResNet-101 [21] based Faster R-CNN here.

FPN could provide high-level semantics at all scales, which reduced the error rate in detection. It became a standard building block in future detections models and improved accuracy their accuracy across the table. It also lead to development of other improved networks like PANet [46], NAS-FPN [47] and EfficientNet [27], which is current state of art detector.

6) ***R-FCN****:* Dai et al. proposed Region-based Fully Convolutional Network (R-FCN) [48] that shared almost all computations within the network, unlike previous two stage detectors which applied resource intensive techniques on each proposal. They argued against the use of fully connected layers and instead used convolutional layers. However, deeper layers in the convolutional network are translation-invariant, making them ineffective for localization tasks. The authors proposed the use of position-sensitive score maps to remedy it. These sensitive score maps encode relative spatial information of the subject and are later pooled to identify exact localization. R-FCN does it by dividing the region of interest into k x k grid and scoring the likeliness of each cell with the detection class feature map. These scores are later averaged and used to predict the object class. R-FCN detector is a combination of four convolutional networks. The input image is first passed through the ResNet-101 [21] to get feature maps. An intermediate output (Conv4 layer) is passed to a Region Proposal Network (RPN) to identify RoI proposals while the final output is further processed through a convolutional layer and is input to classifier and regressor. The classification layer combines the generated the position-sensitive map with the RoI proposals to generate predictions while the regression network outputs the bounding box details. R-FCN is trained in a similar 4 step fashion as Faster-RCNN [44] whilst using a combined cross-entropy and box regression loss. It also adopts

online hard example mining (OHEM) [49] during the training.

Dai et al. offered a novel method to solve the problem of translation invariance in convolutional neural networks. R-FCN combines Faster R-CNN and FCN to achieve a fast, more accurate detector. Even though it did not improve accuracy by much, but it was 2.5-20 times faster than its counterpart.

*7) **Mask R-CNN**:* Mask R-CNN [50] extends on the Faster R-CNN by adding another branch in parallel for pixel-level object instance segmentation. The branch is a fully connected network applied on RoIs to classify each pixel into segments with little overall computation cost. It uses similar basic Faster R-CNN architecture for object proposal, but adds a mask head parallel to classification and bounding box regressor head. One major difference was the use of RoIAlign layer, instead of RoIPool layer, to avoid pixel level misalignment due to spatial quantization. The authors chose the ResNeXt-101 [24] as its backbone along with the feature Pyramid Network (FPN) for better accuracy and speed. The loss function of Faster R-CNN is updated with the mask loss and as in FPN, it uses 5 anchor boxes with 3 aspect ratio. Overall training of Mask R-CNN is similar to faster R-CNN.

Mask R-CNN performed better than the existing state of the art single-model architectures, added an extra functionality of instance segmentation with little overhead computations. It is simple to train, flexible and generalizes well in applications like keypoint detection, human pose estimation, etc. However, it was still below the real time performance (>30 fps).

*8) **DetectoRS**:* Many contemporary two stage detectors like [44], [51], [52] use the mechanism of looking and thinking twice i.e. calculating object proposals first and using them to extract features to detect objects. DetectoRS [53] applies this mechanism at both macro and micro level of the network. At macro level, they propose Recursive Feature Pyramid (RFP), formed by stacking multiple feature pyramid network (FPN) with extra feedback connection from the top-down level path in FPN to the bottom-up layer. The output of the FPN is processed by the Atrous Spatial Pyramid Pooling layer (ASPP) [54] before passing it to the next FPN layer. A Fusion module is used to combine FPN outputs from different modules by creating an attention map. At micro level, Qiao et al. presented the Switchable Atrous Convolution (SAC) to regulate the dilation rate of convolution. An average pooling layer with *5x5* filter and a *1x1* convolution is used as a switch function to decide the rate of atrous convolution [55], helping the backbone detect objects at various scale on the fly. They also packed the SAC in between two global context modules [56] as it helps in making more stable switching. The combination of these two techniques, Recursive Feature Pyramid and Switchable Atrous Convolution results in DetectoRS. The authors incorporated the above techniques with the Hybrid Task Cascade (HTC) [51] as the baseline model and a ResNext-101 backbone.

DetectoRS combined multiple systems to improve performance of the detector and sets the state-of-the-art for the two stage detectors. Its RFP and SAC modules are well generalized and can be used in other detection models. However, it is not suitable for real time detections as it can only process about 4 frames per second.

C. Single Stage Detectors

*1) **YOLO**:* Two stage detectors solve the object detection as a classification problem, a module presents some candidates which the network classifies as either an object or background. However, YOLO or You Only Look Once [57] reframed it as a regression problem, directly predicting the image pixels as objects and its bounding box attributes. In YOLO, the input image is divided into a *S x S* grid and the cell where the object's center falls is responsible for detecting it. A grid cell predicts multiple bounding boxes, and each prediction array consists of 5 elements: center of bounding box – x and y, dimensions of the box – w and h, and the confidence score.

YOLO was inspired from the GoogLeNet model for image classification [18], which uses cascaded modules of smaller convolution networks [58]. It is pre-trained on ImageNet data [12] till the model achieves high accuracy and then modified by adding randomly initialized convolution and fully connected layers. At training time, grid cells predict only one class as it converges better, but it is be increased during the inference time. Multitask loss, combined loss of all predicted components, is used to optimize the model. Non maximum suppression (NMS) removes class-specific multiple detections.

YOLO surpassed its contemporary single stage real time models by a huge margin in both accuracy and speed. However, it had significant shortcomings as well. Localization accuracy for small or clustered objects and limitation to number of objects per cell were its major drawbacks. These issues were fixed in later versions of YOLO [59]–[61].

*2) **SSD**:* Single Shot MultiBox Detector (SSD) [62] was the first single stage detector that matched accuracy of contemporary two stage detectors like Faster R-CNN [44], while maintaining real time speed. SSD was built on VGG-16 [17], with additional auxiliary structures to improve performance. These auxiliary convolution layers, added to the end of the model, decrease progressively in size. SSD detects smaller objects earlier in the network when the image features are not too crude, while the deeper layers were responsible for offset of the default boxes and aspect ratios [63].

During training, SSD match each ground truth box with the default boxes with the best jaccard overlap and train the network accordingly, similar to Multibox [63]. They also used hard negative mining and heavy data augmentation. Similar to DPM [33], it utilized weighted sum of the localization and confidence loss to train the model. Final output is obtained by performing non maximum suppression.

Even though SSD was significantly faster and more accurate than both state-of-art networks like YOLO and Faster R-CNN, it had difficulty in detecting small objects. This issue was later solved by using better backbone architectures like ResNet and other small fixes.

*3) **YOLOv2 and YOLO9000**:* YOLOv2 [59], an improvement on the YOLO [57], offered an easy tradeoff between speed and accuracy while the YOLO9000 model could predict 9000 object classes in real time. They replaced the backbone architecture of GoogLeNet [18] with DarkNet-19 [64]. It incorporated many impressive techniques like Batch Normalization [65] to improve convergence, joint training of classification and detection systems to increase detection

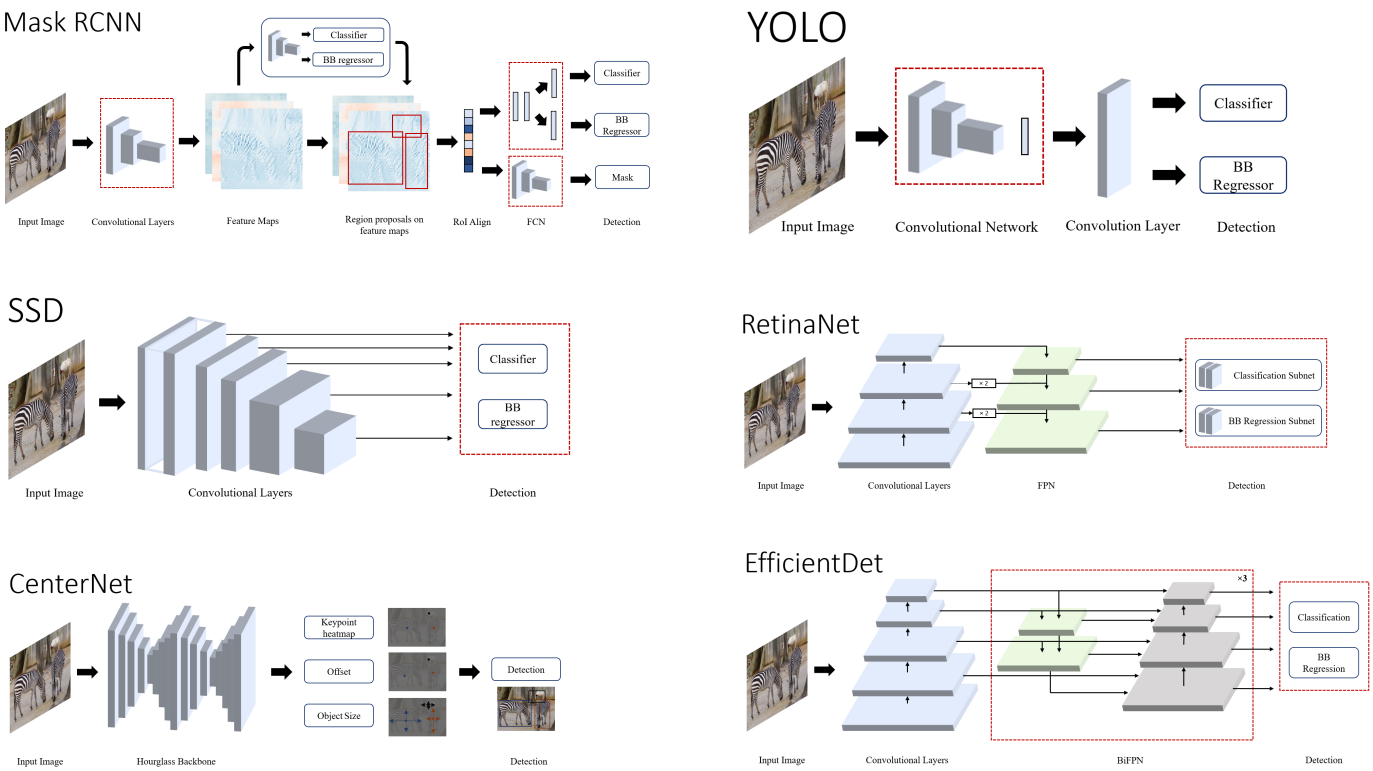

Fig. 9: Illustration of the internal architecture of different two and single stage object detectors[2].

classes, removing fully connected layers to increase speed and using learnt anchor boxes to improve recall and have better priors. Redmon et al. also combined the classification and detection datasets in hierarchical structure using WordNet [66]. This WordTree can be used to predict a higher conditional probability of hypernym, even when the hyponym is not classified correctly, thereby increasing the overall performance of the system.

YOLOv2 provided better flexibility to choose the model on speed and accuracy, and the new architecture had fewer parameters. As the title of the paper suggests, it was *"better, faster and stronger"* [59].

4) ***RetinaNet****:* Given the difference between the accuracies of single and two stage detectors, Lin et al. suggested that the reason single stage detectors lag is the "extreme foreground-background class imbalance" [67]. They proposed a reshaped cross entropy loss, called Focal loss as the means to remedy the imbalance. Focal loss parameter reduces the loss contribution from easy examples. The authors demonstrate its efficacy with the help of a simple, single stage detector, called RetinaNet [67], which predicts objects by dense sampling of the input image in location, scale and aspect ratio. It uses ResNet [21] augmented by Feature Pyramid Network (FPN) [45] as the backbone and two similar subnets - classification and bounding box regressor. Each layer from the FPN is passed to the subnets, enabling it to detect objects as various scales. The classification subnet predicts the object score for each location while the box regression subnet regresses the offset for each anchor to the ground truth. Both subnets are small FCN and share parameters across the individual networks. Unlike most previous works, the authors employ a class-agnostic bounding box regressor and found them to be equally effective.

RetinaNet is simple to train, converges faster and easy to implement. It achieved better performance in accuracy and run time than the two stage detectors. RetinaNet also pushed the envelope in advancing the ways object detectors are optimized by the introduction of a new loss function.

5) ***YOLOv3****:* YOLOv3 had "incremental improvements" from the previous YOLO versions [57], [59]. Redmon et al. replaced the feature extractor network with a larger Darknet-53 network [64]. They also incorporated various techniques like data augmentation, multi-scale training, batch normalization, among others. Softmax in classifier layer was replaced by a logistical classifier.

Even though YOLOv3 was faster than YOLOv2 [59], it lacked any ground breaking change from its predecessor. It even had lesser accuracy than an year old state-of-the-art detector [67].

6) ***CenterNet****:* Zhou et al. in [68] takes a very different approach of modelling objects as points, instead of the conventional bounding box representation. CenterNet predicts the object as a single point at the center of the bounding box. The input image is passed through the FCN that generates a heatmap, whose peaks correspond to center of detected object. It uses a ImageNet pretrained stacked Hourglass-101 [69] as the feature extractor network and has 3 heads – heatmap head to determine the object center, dimension head to estimate size of object and offset head to correct offset of object point. Multitask loss of all three heads is back propagated to feature

[2]Features created using: https://poloclub.github.io/cnn-explainer/

extractor while training. During inference, the output from offset head is used to determine the object point and finally a box is generated. As the predictions, not the result, are points and not bounding boxes, non-maximum suppression (NMS) is not required for post-processing.

CenterNet brings a fresh perspective and set aside years of progress in the field of object detection. It is more accurate and has lesser inference time than its predecessors. It has high precision for multiple tasks like 3D object detection, keypoint estimation, pose, instance segmentation, orientation detection and others. However, it requires different backbone architectures as general architectures that work well with other detectors give poor performance with it and vice-versa.

*7) **EfficientDet**:* EfficientDet [70] builds towards the idea of scalable detector with higher accuracy and efficiency. It introduces efficient multi-scale features, BiFPN and model scaling. BiFPN is bi-directional feature pyramid network with learnable weights for cross connection of input features at different scales. It improves on NAS-FPN [47], which required heavy training and had complex network, by removing one-input nodes and adding an extra lateral connection. This eliminates less efficient nodes and enhances high-level feature fusion. Unlike existing detectors which scale up with bigger, deeper backbone or stacking FPN layers, EfficientDet introduces a compounding coefficient which can be used to "jointly scale up all dimensions of backbone network, BiFPN network, class/box network and resolution" [70]. EfficientDet utilizes EfficientNet [27] as the backbone network with multiple sets of BiFPN layers stacked in series as feature extraction network. Each output from the final BiFPN layer is sent to class and box prediction network. The model is trained using SGD optimizer along with synchronized batch normalization and uses swish activation [71], instead of the standard ReLU activation, which is differentiable, more efficient and has better performance.

EfficientDet achieves better efficiency and accuracy than previous detectors while being smaller and computationally cheaper. It is easy to scale, generalizes well for other tasks and is the current state-of-the-art model for single-stage object detection.

*8) **YOLOv4**:* YOLOv4 [61] incorporated a lot of exciting ideas to design a fast and easy to train object detector that could work in existing production systems. It utilizes "bag of freebies" i.e., methods that only increase training time and do not affect the inference time. YOLOv4 utilizes data augmentation techniques, regularization methods, class label smoothing, CIoU-loss [72], Cross mini-Batch Normalization (CmBN) , Self-adversarial training, Cosine annealing scheduler [73] and other tricks to improve training. Methods that only affect the inference time, called "Bag of Specials", are also added to the network, including Mish activation [74], Cross-stage partial connections (CSP) [25], SPP-Block [41], PAN path aggregated block [46] , Multi input weighted residual connections (Mi-WRC), etc. It also used genetic algorithm for searching hyper-parameter. It has an ImageNet pre-trained CSPNetDarknet-53 backbone, SPP and PAN block neck and YOLOv3 as detection head.

Most existing detection algorithms require multiple GPUs to train model, but YOLOv4 can be easily trained on a single GPU. It is twice as fast as EfficientDet with comparable performance. It is the state-of-the-art for real time single stage detectors.

*9) **Swin Transformer**:* Transformers [75] have had a profound impact in the Natural Language Processing (NLP) domain since its inception. Its application in language models like BERT (Bidirectional Encoder Representation from Transformers) [76], GPT (Generative Pre-trained Transformer) [77], T5 (Text-To-Text Transfer Transformer) [78] etc. have pushed the state of the art in the field. Transformers [75] uses the attention model to establish dependencies among the elements of the sequence and can a attend to longer context than other sequential architectures. The success of transformers in NLP sparked interest in its application in computer vision. While CNNs have been the backbone on advancement in vision, they have some inherent shortcomings like the lack of importance of global context, fixed post-training weights [79] etc.

Swin Transformer [80] seeks to provide a transformer based backbone for computer vision tasks. It splits the input images in multiple, non-overlapping patches and converts them into embeddings. Numerous Swin Transformer blocks are then applied to the patches in 4 stages, with each successive stage reducing the number of patches to maintain hierarchical representation. The Swin Transformer block is composed of local multi-headed self-attention (MSA) modules, based on alternating shifted patch window in successive blocks. Computation complexity becomes linear with image size in local self-attention while shifted window enables cross-window connection. [80] also shows how shifted windows increase detection accuracy with little overhead.

Transformers present a paradigm shift from the CNN based neural networks. While its application in vision is still in a nascent stage, its potential to replace convolution from these tasks is very real. Swin Transformer achieved the state-of-the-art on MS COCO dataset, but utilises comparatively higher parameters than convolutional models.

VI. Lightweight Networks

A new branch of research has shaped up in recent years, aimed at designing small and efficient networks for resource constrained environments as is common in Internet of Things (IoT) deployments [81]–[84]. This trend has percolated to the design of potent object detectors too. It is seen that although a large number of object detectors achieve excellent accuracy and perform inference in real-time, a majority of these models require excessive computing resources and therefore cannot be deployed on edge devices.

Many different approaches have shown exciting results in the past. Utilization of efficient components and compression techniques like pruning ([85], [86]), quantization ([87], [88]), hashing [89], etc. have improved the efficiency of deep learning models. Use of trained large network to train smaller models, called distillation [90], has also shown interesting results. However in this section, we explore some prominent examples of efficient neural network design for achieving high performance on edge devices.

A. SqueezeNet

Recent advances in the field of CNNs had mostly focused on improving the state-of-the-art accuracy on the benchmark datasets, which led to an explosion of model size and their parameters. But in 2016, Iandola et al. proposed a smaller, smarter network called SqueezeNet [91], which reduced the parameters while maintaining the performance. They achieved it by employing three main design strategies viz. using smaller filters, decreasing the number of input channels to *3x3* filters and placing downsampling layers later in the network. The first two strategies decrease the number of parameters while attempting to preserve the accuracy and the third strategy increases the accuracy of the network. The building block of SqueezeNet is called a fire module, which consist of two layers: a squeeze layer and an expand layer, each with a ReLU activation. The squeeze layer is made up of multiple *1x1* filters while the expand layer is a mix of *1x1* and *3x3* filters, thereby limiting the number of input channels. The SqueezeNet architecture is composed of a stack of 8 Fire modules squashed in between the convolution layers. Inspired by ResNet [21], SqueezeNet with residual connections was also proposed which increased the accuracy over the vanilla model. The authors also experimented with Deep Compression [87] and achieved *510*$\times$ reduction in model size compared to AlexNet, while maintaining the baseline accuracy. SqueezeNet presented a good candidate for improving the hardware efficiency of the neural network architectures.

B. MobileNets

MobileNet [92] moved away from the conventional methods of small models like shrinking, pruning, quantization or compressing, and instead used efficient network architecture. The network used depthwise separable convolution, which factorizes a standard convolution into a depthwise convolution and a *1x1* pointwise convolution. A standard convolution uses kernels on all input channels and combines them in one step while the depthwise convolution uses different kernels for each input channel and uses pointwise convolution to combine inputs. This separation of filtering and combining of features reduces the computation cost and model size. MobileNet consists of 28 separate convolutional layers, each followed by batch normalization and ReLU activation function. Howard et al. also introduced the two model shrinking hyperparameters: width and resolution multiplier, in order to further improve speed and reduce size of the model. The width multiplier manipulates the width of the network uniformly by reducing the input and output channels while the resolution multiplier influences the size of the input image and its representations throughout the network. MobileNet achieves comparable accuracy to some full-fledged models while being a fraction of their size. Howard et al. also showed how it could generalize over various applications like face attribution, geolocalization and object detection. However, it was too simple and linear like VGG and therefore had fewer avenues for gradient flow. These were fixed in later iterations of this model [93], [94].

C. ShuffleNet

In 2017, Zhang et al. introduced ShuffleNet [95], an extremely computationally efficient neural network architecture, specifically designed for mobile devices. They recognized that many efficient networks become less effective as they scale down and purported it to be caused by expensive *1x1* convolutions. In conjunction with channel shuffle, they proposed the use of group convolution to circumvent its drawback of limited information flow. ShuffleNet consists mainly of a standard convolution followed by stacks of ShuffleNet units grouped in three stages. The ShuffleNet unit is similar to the ResNet block where they use depthwise convolution in the *3x3* layer and replace the *1x1* layer with pointwise group convolution. The depthwise convolution layer is preceded by a channel shuffle operation. The computation cost of the ShuffleNet can be administered by two hyperparameters: group number to control the connection sparsity and scaling factor to manipulate the model size. As group numbers become large, the error rate saturates as the input channels to each group decreases and therefore may reduce the representational capabilities. ShuffleNet outperformed contemporary models ([3], [18], [91], [92]) while having considerably smaller size. As the only advancement in ShuffleNet was channel shuffle, there isn't any improvement in inference speed of the model.

D. MobileNetv2

Improving on MobileNetv1 [92], Sandler et al. proposed MobileNetv2 [93] in 2018. It introduced the inverted residual with linear bottleneck, a novel layer module to reduce computation and improve accuracy. The module expands a low-dimensional representation of the input into high dimension, filters with a depthwise convolution and then projects it back to low dimension, unlike the common residual block which performs compression, convolution and then expansion operations. The MobileNetv2 contains a convolution layer followed by 19 residual bottleneck modules and subsequently two convolutional layers. The residual bottleneck module has a shortcut connection only when the stride is 1. For higher stride, the shortcut is not used because of the difference in dimensions. They also employed ReLU6 as the non-linearity function, instead of simple ReLU, to limit computations. For object detection, the authors used MobileNetv2 as the feature extractor of a computationally efficient variant of the SSD [62]. This model, called SSDLite, claimed to have 8x fewer parameters than the original SSD while achieving competitive accuracy. It generalizes well over on other datasets, is easy to implement and hence, was well-received by the community.

E. PeleeNet

Existing lightweight deep learning models like [92], [93], [95] relied heavily on depthwise separable convolution, which lacked efficient implementation. Wang et al. proposed a novel efficient architecture based on conventional convolution, named PeleeNet [96], using an assortment of computation conserving techniques. PeleeNet was centered around the DenseNet [23] but looked at many other models for inspiration. It introduced two-way dense layers, stem block, dynamic

number of channels in a bottleneck, transition layer compression and conventional post activation to reduce computation cost and increase speed. Inspired from [18], the two-way dense layer helps in getting different scales of the receptive field, making it easier to identify larger objects. To reduce information loss, a stem block was used in the same way to [20], [97]. They also parted way with the compression factor used in [23] as it hurts the feature expression and reduces accuracy. PeleeNet consists of a stem block, four stages of modified dense and transition layers, and ultimately the classification layer. The authors also proposed a real-time object detection system, called Pelee, which was based on PeleeNet and a variant of SSD [62]. Its performance against the contemporary object detectors on mobile and edge devices was incremental but showed how simple design choices can make a huge difference in overall performance.

F. ShuffleNetv2

In 2018, Ningning Ma et al. present a set of comprehensive guidelines for designing efficient network architectures in ShuffleNetv2 [98]. They argued for the use of direct metrics like speed or latency to measure computational complexity, instead of indirect metrics like FLOPs. ShuffleNetv2 is built on four guiding principles – 1) equal width for input and output channels to minimize memory access cost, 2) carefully choosing group convolution based on the target platform and task, 3) multi-path structures achieve higher accuracy at the cost of efficiency and 4) element-wise operations like add and ReLU are computationally non-negligible. Following the above principles, they designed a new building block. It split the input into two parts by a channel split layer, followed by three convolutional layers which are then concatenated with the residual connection and passed through a channel shuffle layer. For the downsampling model, channel split is removed and residual connection has depthwise separable convolution layers. An ensemble of these blocks slotted in between a couple of convolutional layers results in ShuffleNetv2. The authors also experimented with larger models (50/162 layers) and obtained superior accuracy with considerably fewer FLOPs. ShuffleNetv2 punched above its weight and outperformed other state-of-the-art models at comparable complexity.

G. MnasNet

With the increasing need for accurate, fast and low latency models for various edge devices, designing such a neural network is becoming more challenging than ever. In 2018, Tan et al. proposed Mnasnet [28] designed from an automated neural architecture search (NAS) approach. They formulate the search problem as multi-object optimization aimed at both high accuracy and low latency. It also factorized the search space by partitioning the CNN into unique blocks and subsequently searching for operations and connections in those blocks separately, thereby reducing the search space. This also allowed each block to have a distinctive design, unlike the earlier models [99]–[101] which stacked the same blocks. The authors used RNN-based reinforcement learning agent as controller along with a trainer to measure accuracy and mobile devices for latency. Each sampled model is trained on a task to get its accuracy and run on the real devices for latency. This is used to achieve a soft reward target and the controller is updated. The process is repeated until the maximum iterations or a suitable candidate is derived. It is composed of 16 diverse blocks, some with residual connections. MnasNet was almost twice as fast as MobileNetv2 while having higher accuracy. However, like other reinforcement learning based neural architecture search models, the search time of MnasNet requires astronomical computational resources.

H. MobileNetv3

At the heart of MobileNetv3 [94] is the same method used to create MnasNet [28] with some modifications. A platform aware automated neural architecture search is performed in a factorized hierarchical search space and consequently optimized by NetAdapt [102], which removes the underutilized components of the network in multiple iterations. Once an architecture proposal is obtained, it trims the channels, randomly initialize the weights and then fine-tunes it to improve the target metrics. The model was further modified to remove some expensive layer in the architecture and gain additional latency improvement. Howard et al. argued that the filters in the architecture are often mirrored images of each other, and that accuracy can be maintained even after dropping half of these filters. Using this technique reduced the computations. MobileNetv3 used a blend of ReLU and hard swish as activation filters, the latter is mostly employed towards the end of the model. Hard swish has no noticeable difference from the swish function but is computationally cheaper while retaining the accuracy. For different resource use cases, [94] introduced two models – MobileNetv3-Large and MobileNetv3-Small. MobileNetv3-Large is composed of 15 bottleneck blocks while MobileNetv3-Small has 11. It also included squeeze and excitation layer [56] on its building blocks. Similar to [93], these model act as a feature detector in SSDLite and is 35% faster than earlier iterations [28], [93], whilst achieving higher *mAP*.

I. Once-For-All (OFA)

The use of neural architecture search (NAS) for architecture design has produced state-of-the-art models in the past few years, however, they are compute expensive because of the sampled model training. Cai et al. in [103] proposed a novel method of decoupling model training stage and the neural architecture search stage. The model is trained only once and sub-networks can be distilled from it as per the requirements. Once-for-all (OFA) network provides flexibility for such sub-networks in four important dimension of a convolutional neural network – depth, width, kernel size and dimension. As they are nested within the OFA network and interfere with the training, progressive shrinking was introduced. First, the largest network is trained with all parameters set to maximum. Subsequently, network is fine-tuned by gradually reducing the parameter dimensions like kernel size, depth and width. For elastic kernel, a center of the large kernel is used as the small kernel. As the center is shared, a kernel transformation

TABLE III: Performance comparison of various object detectors on MS COCO and PASCAL VOC 2012 datasets at similar input image size.

Model	Year	Backbone	Size	$AP_{[0.5:0.95]}$	$AP_{0.5}$	FPS
R-CNN*	2014	AlexNet	224	-	58.50%	∼0.02
SPP-Net*	2015	ZF-5	Variable	-	59.20%	∼0.23
Fast R-CNN*	2015	VGG-16	Variable	-	65.70%	∼0.43
Faster R-CNN*	2016	VGG-16	600	-	67.00%	5
R-FCN	2016	ResNet-101	600	31.50%	53.20%	∼3
FPN	2017	ResNet-101	800	36.20%	59.10%	5
Mask R-CNN	2018	ResNeXt-101-FPN	800	39.80%	62.30%	5
DetectoRS	2020	ResNeXt-101	1333	53.30%	71.60%	∼4
YOLO*	2015	(Modified) GoogLeNet	448	-	57.90%	45
SSD	2016	VGG-16	300	23.20%	41.20%	46
YOLOv2	2016	DarkNet-19	352	21.60%	44.00%	81
RetinaNet	2018	ResNet-101-FPN	400	31.90%	49.50%	12
YOLOv3	2018	DarkNet-53	320	28.20%	51.50%	45
CenterNet	2019	Hourglass-104	512	42.10%	61.10%	7.8
EfficientDet-D2	2020	Efficient-B2	768	43.00%	62.30%	41.7
YOLOv4	2020	CSPDarkNet-53	512	43.00%	64.90%	31
Swin-L	2021	HTC++	-	57.70%	-	-

[a]Models marked with * are compared on PASCAL VOC 2012, while others on MS COCO.Rows colored gray are real-time detectors (>30 FPS).

TABLE IV: Comparison of Lightweight models.

Model	Year	Top-1 Acc%	Latency (ms)	Parameters (Million)	FLOPs (Million)
SqueezeNet	2016	60.5	-	3.2	833
MobileNet	2017	70.6	113	4.2	569
ShuffleNet	2017	73.3	108	5.4	524
MobileNetv2	2018	74.7	143	6.9	300
PeleeNet	2018	72.6	-	2.8	508
ShuffleNetv2	2018	75.4	178	7.4	597
MnasNet	2018	76.7	103	5.2	403
MobileNetv3	2019	75.2	58	5.4	219
OFA	2020	80.0	58	7.7	595

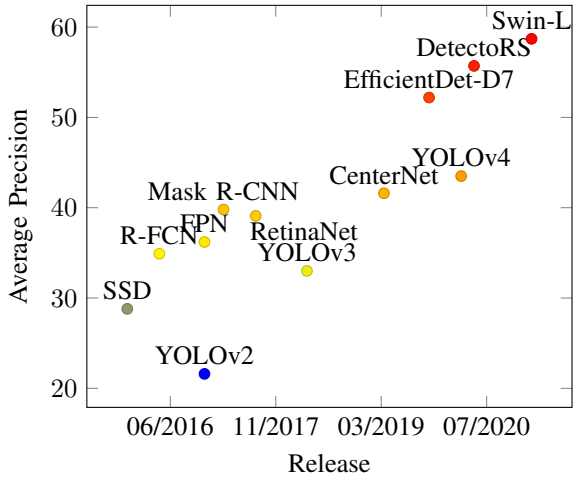

Fig. 10: Performance of Object Detectors on MS COCO dataset.

matrix is used to maintain performance. To vary depth, the first few layers are used and the rest are skipped from the large network. Elastic width employs a channel sorting operation to reorganize channels and uses the most important ones in smaller models. OFA achieved state-of-the-art of 80% in ImageNet top-1 accuracy percentage and also won the 4th Low Power Computer Vision Challenge (LPCVC) while reducing many order of magnitude of GPU training hours. It shows a new paradigm of designing lightweight models for a variety of hardware requirements.

VII. Comparative Results

We compare the performance of both single and two stage detectors on PASCAL VOC 2012 [10] and Microsoft COCO [13] datasets. Performance of object detectors is influenced by a number of factors like input image size and scale, feature extractor, GPU architecture, number of proposals, training methodology, loss function etc., which makes it difficult to compare various models without a common benchmark environment. Here in table III, we evaluate performance of models based on the results from their papers. Models are compared on average precision (AP) and processed frames per second (FPS) at inference time. $AP_{0.5}$ is the average precision of all classes when predicted bounding box has an IoU $>$ 0.5 with ground truth. COCO dataset introduced another performance metric $AP_{[0.5:0.95]}$, or simply AP, which is the average AP for IoU from 0.5 to 0.95 in step size of 0.5. We intentionally compare the performances of detectors on similarly size input image, where possible, to provide a reasonable account, as authors often introduce an array of models to provide flexibility between accuracy and inference time. In fig. 10, we use only the state-of-the-art model from the possible array of object detector family of models. Lightweight models are compared in table IV where we compare them on ImageNet Top-1 classification accuracy, latency, number of parameters and complexity in MFLOPs. Models with MFLOPs lesser than 600 are expected to perform adequately on mobile devices.

VIII. Future Trends

Object detection has seen tremendous progress in the last decade. The algorithm have almost reached human level accuracy in some narrow domains, however it still has many exciting challenges to tackle. In this section, we discuss some of the open problem in the field of object detection.

AutoML: The use of automatic neural architecture search (NAS) for determining the characteristics of object detector is already an actively growing area. We have shown some

detectors designed by NAS in earlier sections, however, it is still in its nascency. Searching for an algorithm is complex and resource intensive.

Lightweight detectors: While lightweight networks have shown great promise by matching classification errors with the full-fledged models, they still lack in detection accuracy by more than 50%. As more and more on-device machine learning applications are added to the market, need for small, efficient and equally accurate models will rise.

Weakly supervised/few shot detection: Most of the state-of-the-art object detection models are trained on millions of bounding box annotated data, which is unscalable as annotating data requires time and resources. Ability to train on weakly supervised data, i.e. image level labelled data, cold result in considerable reduction in these costs.

Domain transfer: Domain transfer refers to use of a model trained on labeled image of a particular source task on a separate, but related target task. It encourages reuse of trained model and reduces reliance on the availability of a large dataset to achieve high accuracy.

3D object detection: 3D object detection is a particularly critical problem for autonomous driving. Even though models have achieved high accuracy, deployment of anything below human level performance will bring up safety concerns.

Object detection in video: Object detectors are designed to perform on individual image which lack correlation between themselves. Using spatial and temporal relationship between the frames for object recognition is an open problem.

IX. Conclusion

Even though object detection has come a long way in the past decade, the best detectors are still far from saturation in performance. As its applications increase in real world, the need for lightweight models that can be deployed on mobile and embedded systems is going to increase exponentially. There has been a rising interest in this domain, but it is still an open challenge. In this paper, we have shown how two-stage and single stage detectors developed over their predecessors. While the two stage detectors are generally more accurate, they are slow and cannot be used for real-time applications like self-driving cars or security. However, this has changed in the last few year where one stage detectors are equally accurate and much faster than the former. As evident in Figure 10, Swin Transformer is the most accurate detector till date. With the current positive trend in the accuracy of detectors, we have high hopes for more accurate and faster detectors.

References

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1. IEEE Comput. Soc, 2001, pp. I–511–I–518. [Online]. Available: http://ieeexplore.ieee.org/document/990517/

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005-06, pp. 886–893 vol. 1, ISSN: 1063-6919.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc.

[4] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey." [Online]. Available: http://arxiv.org/abs/1905.05055

[5] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," version: 1. [Online]. Available: http://arxiv.org/abs/1809.02165

[6] K. S. Chahal and K. Dey, "A survey of modern object detection literature using deep learning." [Online]. Available: http://arxiv.org/abs/1808.07256

[7] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," vol. 7, pp. 128 837–128 868. [Online]. Available: http://arxiv.org/abs/1907.09408

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," vol. 88, no. 2, pp. 303–338. [Online]. Available: http://link.springer.com/10.1007/s11263-009-0275-4

[9] ——, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[10] M. Everingham and J. Winn, "The PASCAL visual object classes challenge 2012 (VOC2012) development kit," p. 32.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," vol. 115, no. 3, pp. 211–252. [Online]. Available: https://doi.org/10.1007/s11263-015-0816-y

[12] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, ISSN: 1063-6919.

[13] 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 *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[14] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4," vol. 128, no. 7, pp. 1956–1981. [Online]. Available: https://doi.org/10.1007/s11263-020-01316-z

[15] A. Aslam and E. Curry, "A survey on object detection for the internet of multimedia things (iomt) using deep learning and event-based middleware: Approaches, challenges, and future directions," *Image and Vision Computing*, vol. 106, p. 104095, 2021.

[16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, pp. 818–833.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." [Online]. Available: http://arxiv.org/abs/1409.1556

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions." [Online]. Available: http://arxiv.org/abs/1409.4842

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 2818–2826. [Online]. Available: http://ieeexplore.ieee.org/document/7780677/

[20] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning." [Online]. Available: http://arxiv.org/abs/1602.07261

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks." [Online]. Available: http://arxiv.org/abs/1603.05027

[23] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." [Online]. Available: http://arxiv.org/abs/1608.06993

[24] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks." [Online]. Available: http://arxiv.org/abs/1611.05431

[25] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, "CSPNet: A new backbone that can enhance learning capability of CNN," version: 1. [Online]. Available: http://arxiv.org/abs/1911.11929

[26] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network." [Online]. Available: http://arxiv.org/abs/2011.08036

[27] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks." [Online]. Available: http://arxiv.org/abs/1905.11946

[28] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile." [Online]. Available: https://arxiv.org/abs/1807.11626v3

[29] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," vol. 60, no. 2, pp. 91–110. [Online]. Available: https://doi.org/10.1023/B:VISI.0000029664.99615.94

[30] ——, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2.

[31] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components," vol. 23, no. 4, pp. 349–361. [Online]. Available: http://ieeexplore.ieee.org/document/917571/

[32] Yan Ke and R. Sukthankar, "PCA-SIFT: a more distinctive representation for local image descriptors," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. II–II, ISSN: 1063-6919.

[33] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, ISSN: 1063-6919.

[34] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," vol. 32, no. 9, pp. 1627–1645.

[35] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 2241–2248. [Online]. Available: http://ieeexplore.ieee.org/document/5539906/

[36] 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 (CVPR)*, June 2014.

[37] J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," p. 18.

[38] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," vol. 1, no. 4, pp. 541–551, publisher: MIT Press. [Online]. Available: https://doi.org/10.1162/neco.1989.1.4.541

[39] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[40] K. Grauman and T. Darrell, "The pyramid match kernel: discriminative classification with sets of image features," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2, pp. 1458–1465 Vol. 2, ISSN: 2380-7504.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," vol. 37, no. 9, pp. 1904–1916, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[42] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, ser. MM '14. Association for Computing Machinery, pp. 675–678. [Online]. Available: https://doi.org/10.1145/2647868.2654889

[43] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," p. 10.

[44] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[45] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2017.

[46] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation." [Online]. Available: http://arxiv.org/abs/1803.01534

[47] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 7029–7038. [Online]. Available: https://ieeexplore.ieee.org/document/8954436/

[48] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," 2016.

[49] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," 2016.

[50] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.

[51] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "Hybrid task cascade for instance segmentation." [Online]. Available: https://arxiv.org/abs/1901.07518v2

[52] Z. Cai and N. Vasconcelos, "Cascade r-CNN: Delving into high quality object detection." [Online]. Available: https://arxiv.org/abs/1712.00726v1

[53] S. Qiao, L.-C. Chen, and A. Yuille, "DetectoRS: Detecting objects with recursive feature pyramid and switchable atrous convolution." [Online]. Available: http://arxiv.org/abs/2006.02334

[54] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs." [Online]. Available: http://arxiv.org/abs/1606.00915

[55] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets*, ser. inverse problems and theoretical imaging, J.-M. Combes, A. Grossmann, and P. Tchamitchian, Eds. Springer, pp. 286–297.

[56] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks." [Online]. Available: http://arxiv.org/abs/1709.01507

[57] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[58] M. Lin, Q. Chen, and S. Yan, "Network in network." [Online]. Available: http://arxiv.org/abs/1312.4400

[59] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger." [Online]. Available: https://arxiv.org/abs/1612.08242v1

[60] ——, "Yolov3: An incremental improvement," 2018.

[61] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection." [Online]. Available: http://arxiv.org/abs/2004.10934

[62] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Springer International Publishing, pp. 21–37.

[63] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," 2013.

[64] J. Redmon, "Darknet: Open source neural networks in c," http://pjreddie.com/darknet/, 2013–2016.

[65] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, pp. 1026–1034. [Online]. Available: http://ieeexplore.ieee.org/document/7410480/

[66] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, "Introduction to WordNet: An on-line lexical database*," vol. 3.

[67] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[68] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points." [Online]. Available: http://arxiv.org/abs/1904.07850

[69] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Computer Vision – ECCV 2016*, ser. Lecture Notes in Computer Science, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Springer International Publishing, pp. 483–499.

[70] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 10 778–10 787. [Online]. Available: https://ieeexplore.ieee.org/document/9156454/

[71] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions." [Online]. Available: http://arxiv.org/abs/1710.05941

[72] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression." [Online]. Available: http://arxiv.org/abs/1911.08287

[73] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts." [Online]. Available: http://arxiv.org/abs/1608.03983

[74] D. Misra, "Mish: A self regularized non-monotonic activation function." [Online]. Available: http://arxiv.org/abs/1908.08681

[75] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[76] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[77] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[78] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[79] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *arXiv preprint arXiv:2101.01169*, 2021.

[80] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021.

[81] M. N. Abbas, M. S. Ansari, M. N. Asghar, N. Kanwal, T. O'Neill, and B. Lee, "Lightweight deep learning model for detection of copy-move image forgery with post-processed attacks," in *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, 2021, pp. 000 125–000 130.

[82] S. Karakanis and G. Leontidis, "Lightweight deep learning models for detecting covid-19 from chest x-ray images," *Computers in Biology and Medicine*, vol. 130, p. 104181, 2021.

[83] A. Jadon, A. Varshney, and M. S. Ansari, "Low-complexity high-performance deep learning model for real-time low-cost embedded fire detection systems," *Procedia Computer Science*, vol. 171, pp. 418–426, 2020.

[84] A. Jadon, M. Omama, A. Varshney, M. S. Ansari, and R. Sharma, "Firenet: a specialized lightweight fire & smoke detection model for real-time iot applications," *arXiv preprint arXiv:1905.11922*, 2019.

[85] Y. L. Cun, J. S. Denker, and S. A. Solla, *Optimal Brain Damage*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, p. 598–605.

[86] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *IEEE International Conference on Neural Networks*, 1993, pp. 293–299 vol.1.

[87] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." [Online]. Available: http://arxiv.org/abs/1510.00149

[88] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations." [Online]. Available: http://arxiv.org/abs/1511.00363

[89] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick." [Online]. Available: http://arxiv.org/abs/1504.04788

[90] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network." [Online]. Available: http://arxiv.org/abs/1503.02531

[91] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size." [Online]. Available: http://arxiv.org/abs/1602.07360

[92] 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." [Online]. Available: http://arxiv.org/abs/1704.04861

[93] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks." [Online]. Available: http://arxiv.org/abs/1801.04381

[94] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3." [Online]. Available: https://arxiv.org/abs/1905.02244v5

[95] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 6848–6856. [Online]. Available: https://ieeexplore.ieee.org/document/8578814/

[96] R. J. Wang, X. Li, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," p. 10.

[97] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue, "DSOD: Learning deeply supervised object detectors from scratch." [Online]. Available: http://arxiv.org/abs/1708.01241

[98] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet v2: Practical guidelines for efficient CNN architecture design." [Online]. Available: http://arxiv.org/abs/1807.11164

[99] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning." [Online]. Available: https://arxiv.org/abs/1611.01578v2

[100] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search." [Online]. Available: https://arxiv.org/abs/1712.00559v3

[101] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," vol. 33, no. 1, pp. 4780–4789, number: 01. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4405

[102] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "NetAdapt: Platform-aware neural network adaptation for mobile applications." [Online]. Available: http://arxiv.org/abs/1804.03230

[103] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," 2020.