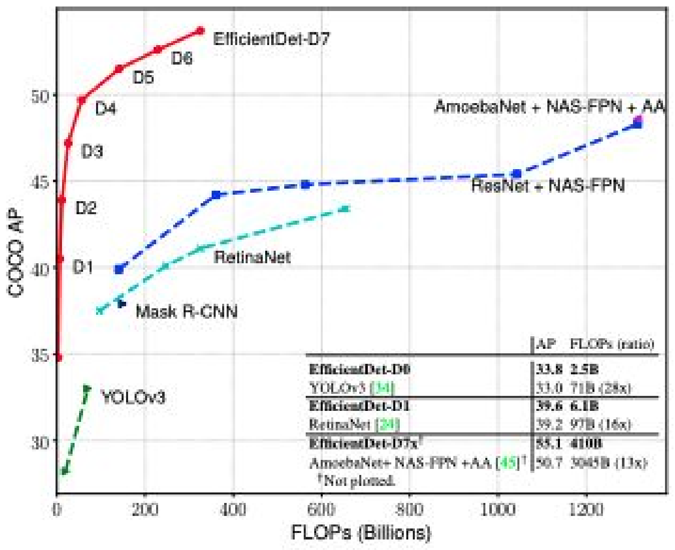
Hai Victor Habi 30006417
Sagi Timinsky 312739485

Buses Detection Project

We were given images of buses, their location and color. Our goal was to detect the positions of buses and their color on a never seen before image.

To achieve this goal we based our solution on this paper:
EfficientDet: Scalable and Efficient Object Detection and this Git repository.
You can find our code here.
Some images are taken from the said paper.

EfficientNets scales up convolutional networks (ConvNets) that can achieve better accuracy and efficiency, when comparing YOLO architecture.



	AP	FLOPs (ratio)
EfficientDet-D0	**33.8**	**2.5B**
YOLOv3 [34]	33.0	71B (28x)
EfficientDet-D1	**39.6**	**6.1B**
RetinaNet [24]	39.2	97B (16x)
EfficientDet-D7x†	**55.1**	**410B**
AmoebaNet+ NAS-FPN +AA [45]†	50.7	3045B (13x)

†Not plotted.

Figure 1 - Model FLOPs vs. COCO accuracy – All numbers are for single-model single-scale. Our EfficientDet achieves new state-of-the-art 55.1% COCO AP with much fewer parameters and FLOPs than previous detectors.

Improving performance of ConvNets can be done by increasing one of the following:

- Network depth
- Width
- Image resolution

In an earlier article describing EfficientNets, the authors indicate that it is important to balance all three dimensions of the network: width/depth/resolution. This means scaling each dimension by a constant ratio.

They discovered the performance improvement from model scaling is heavily dependent on the baseline network. Therefore, the authors use an algorithm called neural architecture search or

Hai Victor Habi 30006417
Sagi Timinsky 312739485

NAS, to find an optimum baseline network. The NAS algorithm uses reinforcement learning to determine optimum structures, given an objective function.

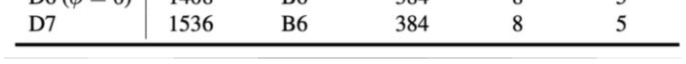
The bidirectional feature pyramid network (BiFPN) in this network serves as a feature network and it aims to aggregate features at different resolutions.

BiFPN is a result of optimization done on a traditional FPN which integrates features at different scales. The optimization included: bidirectional information flow (both top-down and bottom-up) and adding an extra edge from the input node to the output node only at the same level.

As can be seen in figure 1, it takes the features from the levels 3 - 7 from the backbone network and repeatedly applies the BiFPN. The fused features are fed into a class and box network to predict the object class and bounding box.

Since these features are at different resolutions, they usually contribute to the output feature unequally. To get around this an additional weight for each input feature is calculated to allow the network to learn the importance of each feature.

Inspired by the compound scaling introduced in EfficientNets, a new compound scaling method was proposed for object detection. This method uses a coefficient (Φ) to jointly scale-up all dimensions of the backbone network, BiFPN network, class/box network and resolution. This coefficient altering from one to seven creates a family of networks: EfficientDet-D0 (Φ = 0) to D7 (Φ = 7).



	Input size R_{input}	Backbone Network	BiFPN #channels W_{bifpn}	BiFPN #layers D_{bifpn}	Box/class #layers D_{class}
D0 ($\phi = 0$)	512	B0	64	2	3
D1 ($\phi = 1$)	640	B1	88	3	3
D2 ($\phi = 2$)	768	B2	112	4	3
D3 ($\phi = 3$)	896	B3	160	5	4
D4 ($\phi = 4$)	1024	B4	224	6	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1408	B6	384	8	5
D7	1536	B6	384	8	5

The reason we chose to work with EfficientDet models is that its results compared to other models showed both better accuracy and efficiency.

After testing several models, we ended up using D1.

Hai Victor Habi 30006417
Sagi Timinsky   312739485

Mosaic

The idea is to mix different images together, to make the model perform better in different scenarios. The different images were brought together to a single image, thus forcing models to detect the buses using diverse key features.
We can also claim that the model had slightly different context than around the buses, contributing to its robustness.

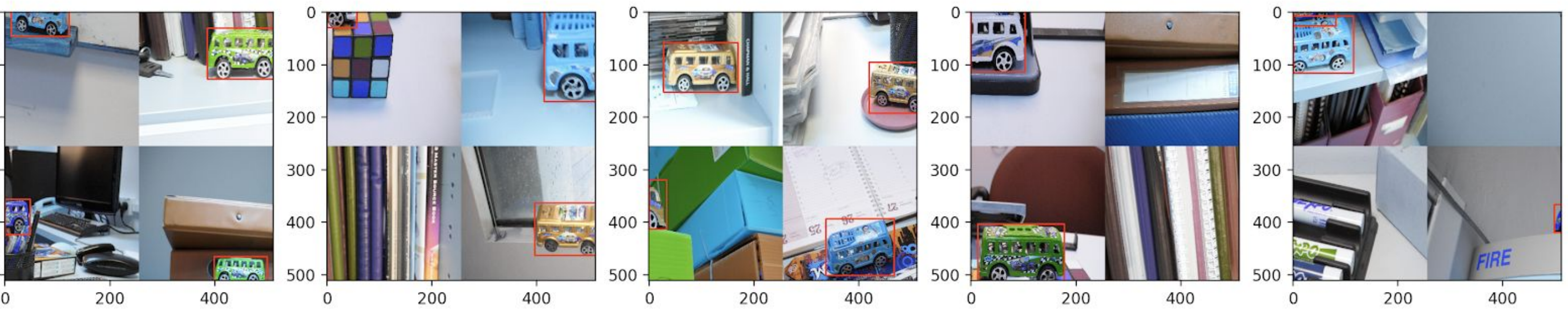


Figure 3 - Examples of mosaic images used

Mosaic solves two problems:

- We have a relatively small dataset available for training. And by applying mosaic we generated more data.
- This method gives more robustness to the model in detecting key features.