Introducing **EfficientDet** before **FasterRCNN**

**AIM:**

To compare these models in terms of their feature learning strategy , object detection strategies and other markable differences and to ascertain the reason why one is good before the other on same dataset and same dependencies.

**Brief Comparison**:

Both the models are benchmark object detection model for object detection.  
FasterRCNN is discovered by FAIR (Facebook Research ) while EfficientDet is been proposed by Google Brain in year 2016 and mid of 2020 respectively.

Here we will try to point out the key difference in the working style between these networks.

**Note** : Since the latter model is so novice, there is no proper documentation available and also the use case for this model is not yet been tested rigorously on different dataset containing different variation of image, but the former case it is opposite till today it is the first choice.

Hence, we will try to present our comparison with the benchmark git repository of both the models and their publication paper

Links:

* FasterRcnn
* [Publication Paper](https://arxiv.org/pdf/1506.01497.pdf)
* [Git Repo](https://github.com/facebookresearch/maskrcnn-benchmark)
* EfficientDet
  + - [Publication Paper](https://arxiv.org/pdf/1911.09070.pdf)
    - [Git Repo](https://github.com/google/automl/tree/master/efficientdet)
* COMPARING BACKBONE NETWORK

Both the models offers a wide range of flexibility to opt one among the many backbones.

Backbone Networks are the **feature extractor** which select/learn the best feature from the input image and on top of that classifier and box regressor works.

Both model has a long list of backbone network. We have used FPN-ResNet101 in case of FasterRCNN and tried with EfficientNet0 and EfficientNet1 both for EfficientDet.  
  
Lets summarize the differences in the way the features are being extracted in the above defined models

|  |  |
| --- | --- |
| FasterRCNN | EfficientDet |
| 1. Backbone Structure | Backbone Structure |

If we look a closely, the depth of the feature map at the intermediate step of FasterRCNN is getting changed at different layer at a very orderly fashion , but for EfficientNET it is very zig zag.

**Example** :

* + - Lets see the res3 block of FRCNN backbone that is ResNET101.

It receives a input of 256 layers and produce a output of 512 layers

At each Bottleneck layer it takes the input from its previous layer and transform the feature to its bottleneck shape 128 and iterate 2 times at the same shape and finally outputs its corresponding layer output shape

This actually , certainly happening at all the bottleneck and if the output shape is large, more deep bottleneck layer is used , res3 has 5 bottleneck while res4 has 23 bottleneck. This model tries to finetune the feature extracted from the input.

* + - Looking the backbone of EfficientDet, wiz efficientNet, a very random and zig zag ordering for the depth of the feature map at different layer is found.  
      One place in the network the input depth of 96 is dropped to 4 channel feature also 4th MBConvBlock a feature map of depth 240 is being learned from a feature map of depth 10 that too there is no such kind of repetition which we can see in Resnet in terms of bottleneck.

**Inference** : No doubt, ResNet101 (nearly 110 Conv including FPN and Stem) is far deeper than EfficientNet (80 Conv), we noted that depth of the feature map are getting from 10 to 240 and 20 to 480 and vice versa at one shot without any repetition, we can easily see that this model is actually wanting to take care of very important feature of the input down the model. Which is not the case with Resnet it gives importance to feature learning uniformly and the depth of the feature map increases down the model, so learning is purely full fledged.  
🡪 This quality of EfficientNet suffice with the case when we have a very heavily featured input, where the large amount of information inside image is present with high and divers texture which help the model to pick these variation as the best feature as here the learning is with filters with adhoc feature depth.

**ResNet**🡪 This model tries to finetune the feature extracted from the input, uniform and accout for the best representation of the input

**EfficienNet** 🡪 This model tries to account for more dynamic feature of the input, non uniform and accounts for the very important feature of the input

Coming to our dataset , our dataset has not got the abundance in terms of texture, pattern when compared to COCO or PASCAL dataset. We have black and white images . Among them, diverse variation is near to the *stamp* and *signature* and proportionally towards the *logo* as they are pretty dynamic in shape and marginally dynamic in terms of intensity and location, which matters a lot for feature learning. Coming to table the features are more or less not dynamic and can be considered to be redundant due to abrupt feature learning pattern. Due to such abrupt zig zag learning of EfficientNet they are not giving glance to *tables* as to keep in the important and fine-grained feature they are picking features from *logo* *stamp* and *signature* where images is more dynamic and textureful.

The backbone of EfficientDet is not that perfect for feature-less input (black and white) that too the pattern of the texture in the majority of the part of the image is similar which might be getting dropped in zig zag convolution in order to bring out the most important information of the input.

* COMPARING FPN [Feature Pyramid Network]  
  FPN is basically used to extract the feature at different resolutions , hence the name is pyramid

|  |  |
| --- | --- |
| FasterRCNN | EfficientDet |
| 1. Backbone Itself is FPN | They use separate FPN and backbone |

As we move ahead in Backbone the spatial dimension of the feature is getting reduced.

Feature Pyramid Network allows us to consider the feature learned during the backpone propagation at different stages to be used as pyramids so that objects of different scales can be detected effectively.

* **FasterRCNN** backbone network uses FPN to get the features of the input at different spatial resolution. It takes the intermediate output of the feature map at different stage from ResNet and perform 1\*1 convolution to reduce the depth then apply 3\*3 convolution to remove aliasing and get final feature map over which Region Proposal Network Start.

P2,P3…P5 are the different feature map of different spatial resolution. All pyramid feature map has 256 -d channels.  
As Depicted in Fig-1 M4 is a element wise sum of C4 after conv(1,1) is applied and its previous pyramid layer up sampled 2 times spatially.

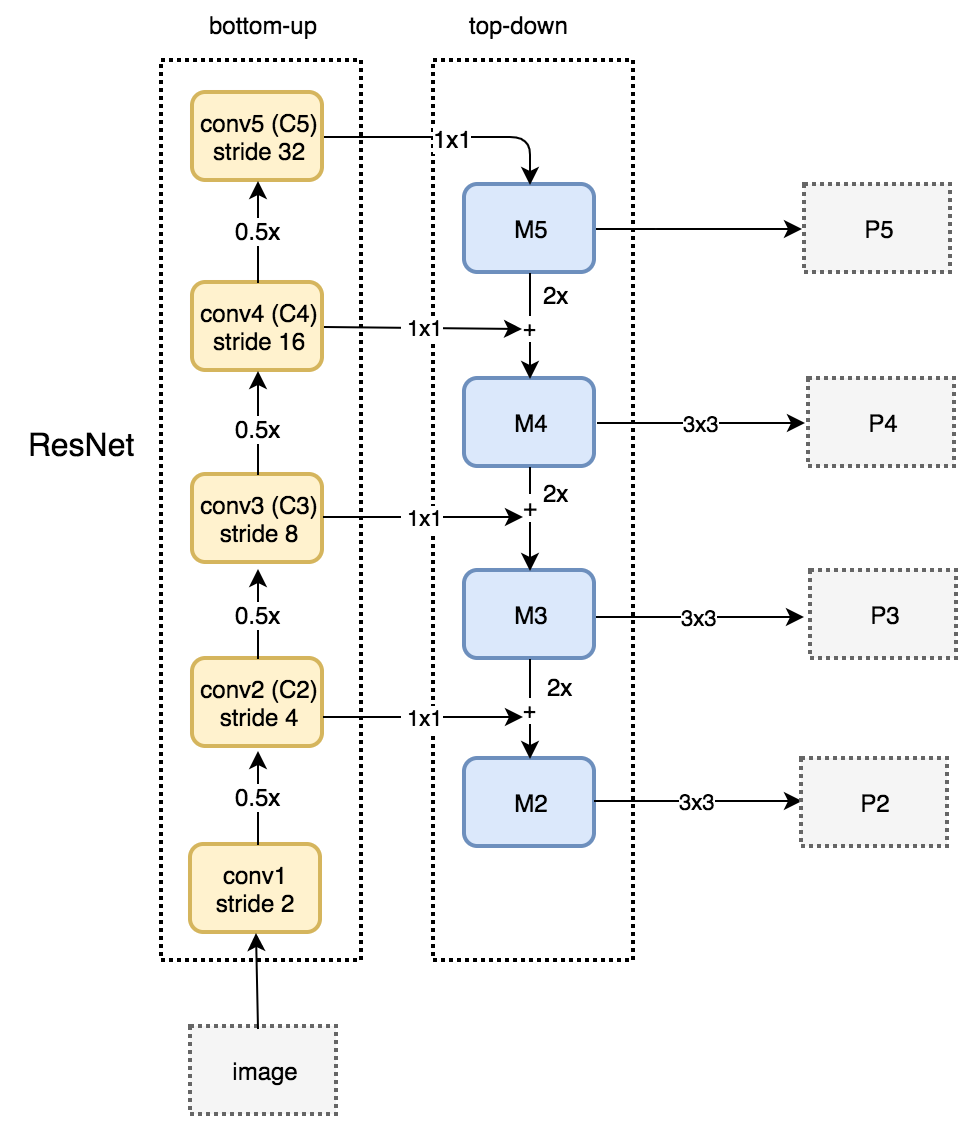


Fig 🡪 1 FPN FasterRCNN

* **EfficientDet** have a separate backbone network. Their backbone is bidirectional. They also take the feature map from backbone network at different scale but it utilizes a dedicated full-fledged network for this.

self.bifpn = nn.Sequential(\*[BiFPN(self.num\_channels) for \_ in range(min(2 + self.compound\_coef, 8))])

Note : self.compund.coef is used to define the number of dedicated bifpn layer. When self.compund.coef is 0 we get a sequential block of 2 bifpn layer as shown in the fig2, i.e. the input of the green block will become the input of the blue block in the next successive block. The arrowed lines defined the participation of one feature set to define the other feature set.

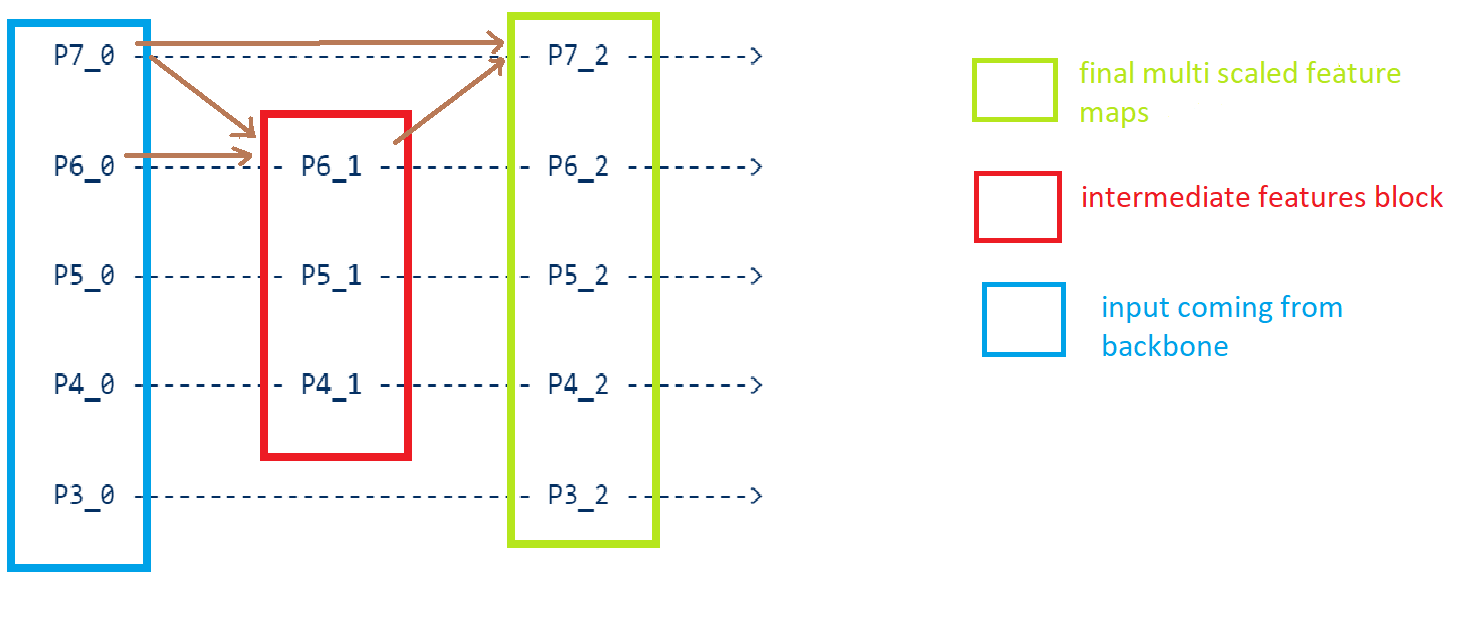


Fig 2 🡪 FPN of EfficientDet

**Inference:** Both the network is absolutely different when it comes to FPN. FasterRCNN uses backbone as a building block for FPN while EfficientDet have a dedicated sequential block with number of subblock as a parameter.

This behaviour of the FPN block in the latter case again makes more use of convolutional blocks, which certainly going to select more tuned features among the earlier tuned from backbone ,of the input and makes it unfit for the images which are tempered quality and not textural in nature .

Coming to our dataset which is far noisy and texture less when compared to COCO and Pascal , in context of EfficientDet, we can assume that since the feature itself coming out from the backbone is monotonic to high quality fine-grained and very important features, which is very tough in invoice dataset due to less textures, again doing a rapid pyramidal search on a parametric basis making the ultimate feature to be input to the later classification and regression model , makes them more biased toward fine-grained input feature learning . Again as mentioned earlier , the feature variation is much toward signature stamp and logo hence these bidirectional weighted of FPN making its way to get tuned toward these classes as compared to other class where the features are not prominently varying.

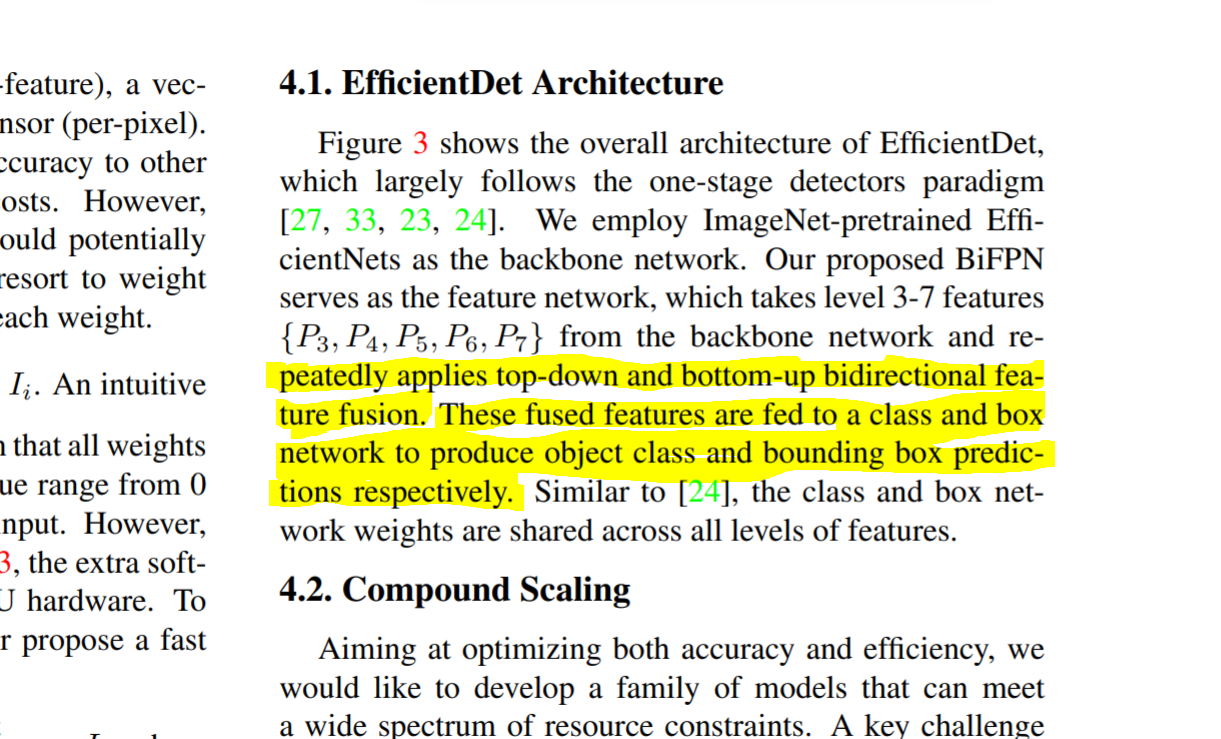
* COMPARING RPN (Region Proposal Network)

RPN are to generate proposals for the region where the object lies, a small **network**is slide over a convolutional feature map that is the output FPN. RPN has a classifier and a regressor. Classifier determines the probability of a proposal having the target object and regression regresses the coordinates of the proposal.   
RPN is a translation invariant algorithm.

|  |  |
| --- | --- |
| FasterRCNN | EfficientDet |
| Use RPN | No RPN, (publication there is no mention of RPN or anything ) |

**FasteRCNN** uses RPN, they take the features map from the FPN layers at different scales , proceed with 3\*3 convolution and lastly performs 1\*1 convolution(also called sibling) on the output 2 times first for objectness and another for bounding box. This convolutions are called RPN Head. After RPN head they perform ROI pooling. Its purpose is to perform max pooling on inputs of nonuniform sizes to obtain fixed-size feature maps.

**EfficientDet** does not utilize any dedicated region based learning fashion like FasterRCNN in model side. It tries to handle this taste of rpn while computing the loss when regression for the bounding box annotation is already taken place and classification is already taken place. Also EfficientDet is missing the last FC layers.



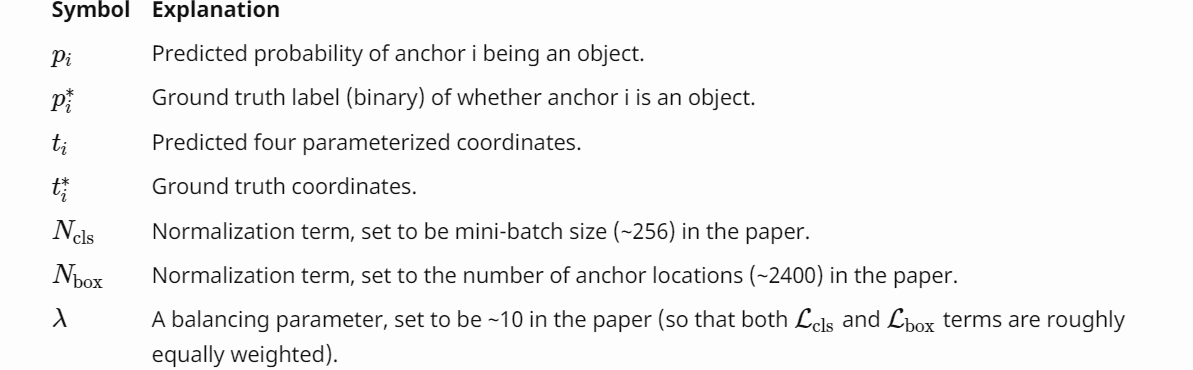
**Inference :** FasterRCNN has much smart way of looking for object finding by looking for regions in FPN output, in the same time EfficientDet is lacking this quality they goes directly to bounding box regression and classification without region oriented view.

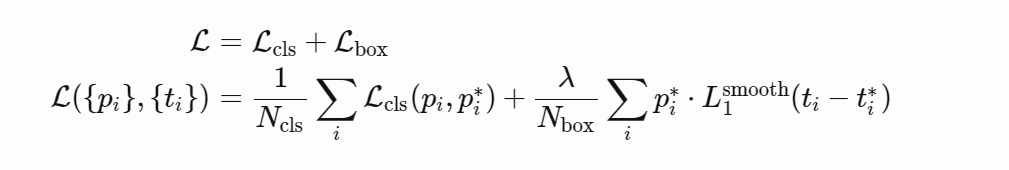
From this it could be concluded again, FasterRCNN gives a second chance for the object to be picked if the backbone network is not that fully tuned toward finding the object efficiently, but this second chance is not present in EfficientDet, it relies solely on its backbone network feature extraction. The feature selected/learned have to have fully oriented to the elements present in the input for better result.

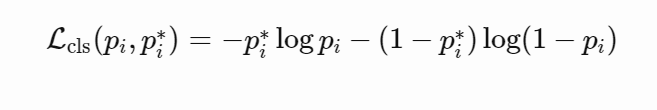
Our case, the backbone of EfficientDet is not that perfect for feature-less input (black and white) that too feature in the majority of the part of the image is similar in terms of the image information when viewed at pixel level in majority of data in our dataset which might be getting dropped in zig zag convolution of backbone network.

* COMPARING Loss Function

Both the model uses same loss function. The value for Lambda is different for EfficientDet in different implementation while in FasterRCNN it is mentioned in paper







Smooth L1 loss:  
f(x) = 0.5 \* (sigma \* x)^2 if |x| < 1 / sigma / sigma

|x| - 0.5 / sigma / sigma otherwise

value of sigma for EfficientDet and FasterRCNN is taken as constant by some heuristics.

**Conclusion:**

After a short and brief view of these models components, we are near to point out the elusive findings in a bit concrete format:

1. Backbone network of FasterRCNN rely to find the best representation of input and EfficientDet rely to find the important feature from the input. If there is no novice feature EfficientDet is getting rid of this by means of zig zag convolution.
2. The main motive of EfficientDet from the paper I can see is they want to tweak the working style of other detection model by means of proportionally varying the model shape in relation to input and to compensate the feature tuning by means of BIFPN.1 But FasterRCNN is a region-based network.
3. Pyramidal network in EfficientDet is parameterized, the bigger the value, the more sort selection of features. This property makes this model really unfit for featureless images and good for images small objects. FasterRCNN keep FPN simple and no rigorous feature selection after the backbone.
4. EfficientDet is not region based network, they rely solely on dedicated classifier and regression box for the final prediction. This makes the working of FPN very sensitive for EfficientDet. But For FasterRCNN the FPN are getting finetuned from the mistakes of RPN.
5. Both models in their benchmarks are using same number of *anchors* and loss function, but the loss function parameter are different for these models.