Archaeological Automatic Detection in LiDAR: Architecture Proposal Document

Martin Olivier

May 11, 2020

Contents

1 Data Augmentation

1.1 Image Modification

An issue seen in LiDAR is the fact that object are often occluded, fragmented and do not appear with the same clarity as examples seen in satellite imagery. This means that we need to construct and train a network that is robust to changes in context and occlusion. Fortunately, research has been focussed in constructing better data augmentation techniques that attempt to solve those issues. Those techniques "mixes" different images from the dataset, covering parts of one another to make it harder for the network to correctly infer. CutMix[1] is one of those techniques. Mosaic, introduced by Bochkovsky, Wang and Liao in the YOLOv4 paper[2] creates a new images out of 4, by creating a "mosaic" of sort, where the 4 images can take varying portions of the new image.

1.2 Regularization and Normalization

Regularization allows to reduce the complexity of a network and prevent overfitting. This is usually done by "dropping" random connections in a network, and training without those connections, a technique known as DropOut[3]. DropBlock[4] relies on a similar method, but is more suitable for convolutional networks. DropBlock works by first choosing random seed points in a mask, and dropping a continuous region around those points. This is more effective than removing purely random activation as close activations contain closely related information.

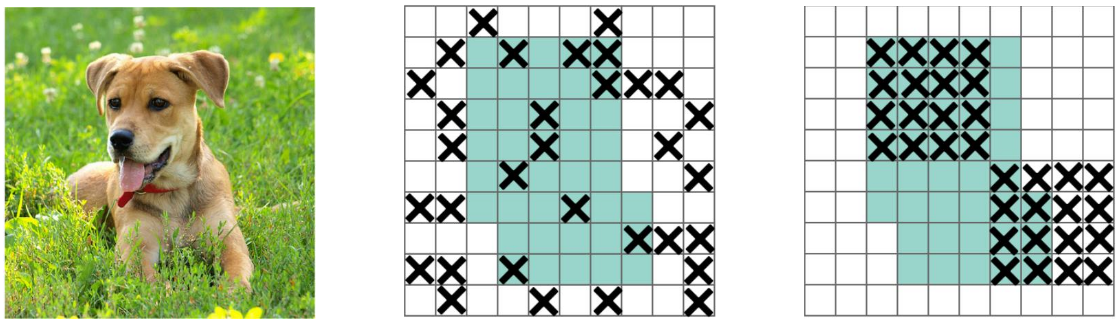

(a) Base image (b) Random Drop in activations (c) DropBlock

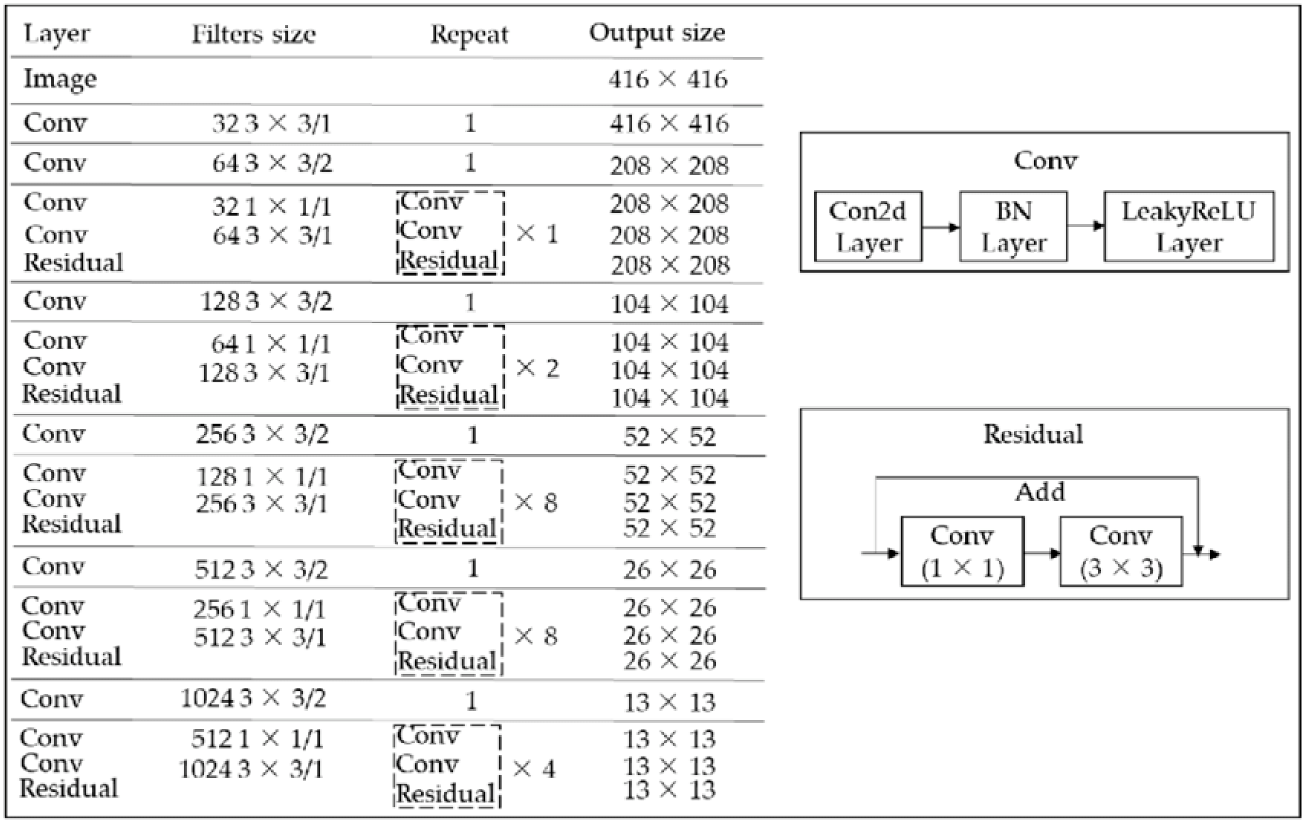
Figure 1: The blue regions represents neurons that contains semantic information on the base image. (b) shows the effect of removing activations at random, which is not effective as neurons close to each other contains closely related semantic information. (c) shows the DropBlock method, which have a better chance of entirely removing important semantic information on the base image, such as the head or the feet of the dog, forcing the remaining neurons to learn useful features

Label smoothing [5] convert hard labels, like one hot labeling into soft labels. This works by introducing a noise distribution in the labeling of the data, and converting the original label in relation to this noise distribution. **However, this label smoothing requires a rewriting of the loss system.**

2 Model Architecture

2.1 Backbone and activation

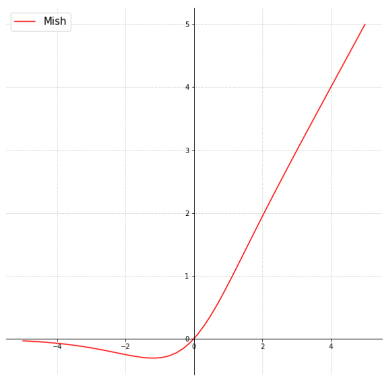
The backbone is CSPDarknet53[6], most notably used in YOLOv4[2]. This backbone obtained better result than the CSPResNeXt-50[7] in the YOLO paper, which is why it is used. Inspiration could also be taken from Zhuang *et al.*[8], with passthrough and concatenation from earlier layers.



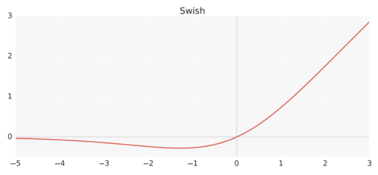
Layer	Filters	size	Repeat	Output size
Image				416 × 416
Conv	32	3 × 3/1	1	416 × 416
Conv	64	3 × 3/2	1	208 × 208
Conv	32	1 × 1/1	Conv	208 × 208
Conv	64	3 × 3/1	Conv × 1	208 × 208
Residual			Residual	208 × 208
Conv	128	3 × 3/2	1	104 × 104
Conv	64	1 × 1/1	Conv	104 × 104
Conv	128	3 × 3/1	Conv × 2	104 × 104
Residual			Residual	104 × 104
Conv	256	3 × 3/2	1	52 × 52
Conv	128	1 × 1/1	Conv	52 × 52
Conv	256	3 × 3/1	Conv × 8	52 × 52
Residual			Residual	52 × 52
Conv	512	3 × 3/2	1	26 × 26
Conv	256	1 × 1/1	Conv	26 × 26
Conv	512	3 × 3/1	Conv × 8	26 × 26
Residual			Residual	26 × 26
Conv	1024	3 × 3/2	1	13 × 13
Conv	512	1 × 1/1	Conv	13 × 13
Conv	1024	3 × 3/1	Conv × 4	13 × 13
Residual			Residual	13 × 13

Figure 2: Architecture of the darknet 53 backbone.

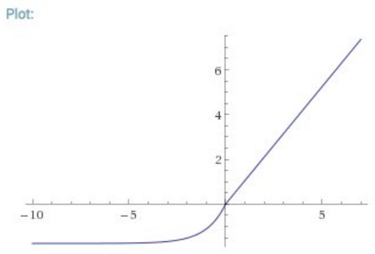
The choice of a activation function would need to be tested. Mish[9](Figure 3a) and Swish (Figure 3b) or even Scaled Exponential Linear Unit (SELU) (Figure 3c) are good candidates.



(a) Mish Activation Function



(b) Swish Activation Function



(c) Scaled Exponential Linear Unit (SELU) Activation Function

Figure 3: Activation Function Candidates

2.2 Dual Scale Detectors

An idea introduced in YOLT[10] is to use two different detectors running simultaneously: one trained on small scale images to detect small objects, the other trained on larger images to detect bigger objects. The small scale network would be fed small chips of images, while the large scale network would be fed downscaled large swathes of terrain. This meant that the large scale network would run much less often than the small scale network, and mitigate the loss of performance of running two network at the same time.

An ensemble method similar to this could be used, where two networks try to find objects at different scales.

2.3 Multi-scale Feature Fusion

To be able to detect the small objects often seen in LiDAR data, a multi-scale feature fusion module needs to be used. There are a few existing modules and techniques that can boost the detection rates of a model by aggregating low level feature maps from earlier layers with ones from higher layers. We call those modules Multi Layer Feature Fusion (MLFF).

The MLFF module from Zhuang *et al.*[8] takes 3 feature maps from different layers, upscales and concatenates them, then applies a series of convolution operations to obtain 3 predictions at different scale.

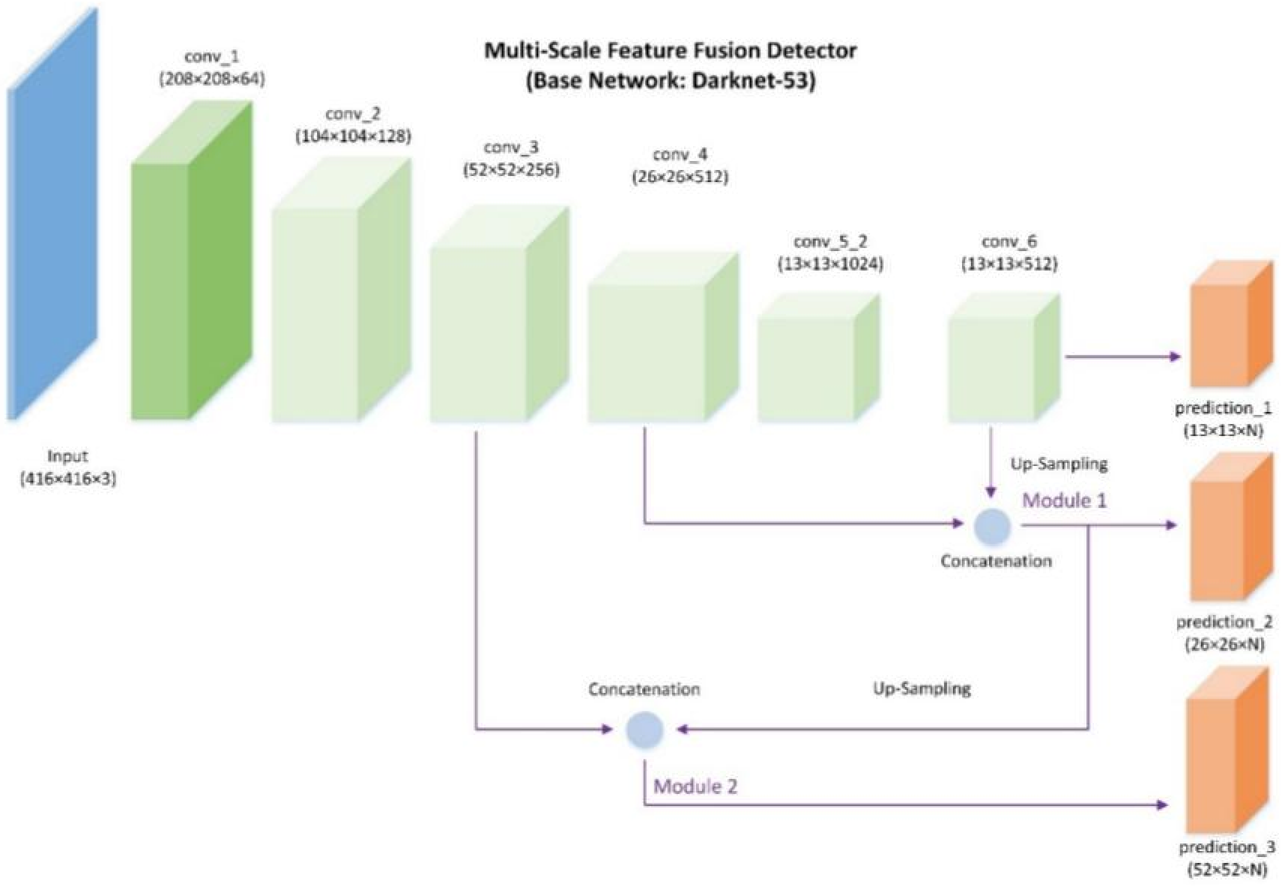


Figure 4: Feature Fusion implementation of Zhuang *et al.*. We are only interested in the bottom part of the figure, with feature maps from different levels being concatenated.

The MLFF module from Qian *et al.*[11] shown in Figure 5 takes each proposal generated by a Region Proposal Networks, and maps its position to all level of feature map generated by the Feature Proposal Networks. From there it obtains N regions of the feature maps (N being the number of levels), which it transforms into 7×7 feature maps through the RoiAlign[**maskRCNN**] operation. Finally, it concatenates these 4 regions along the channel dimensions, and applies two convolution operations along with a Fully Connected Layer for bounding box regression and classification.

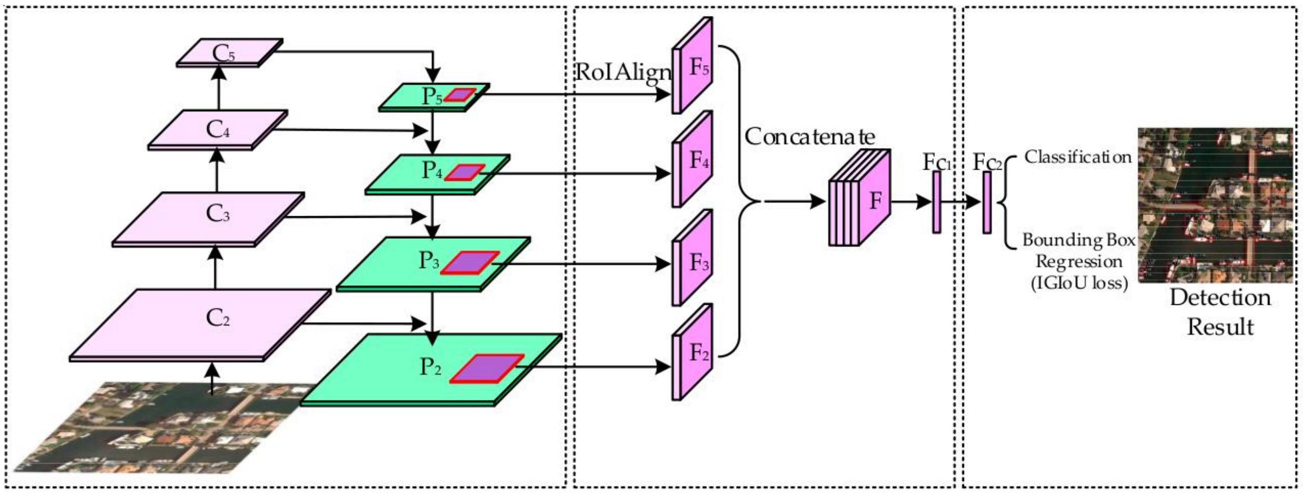
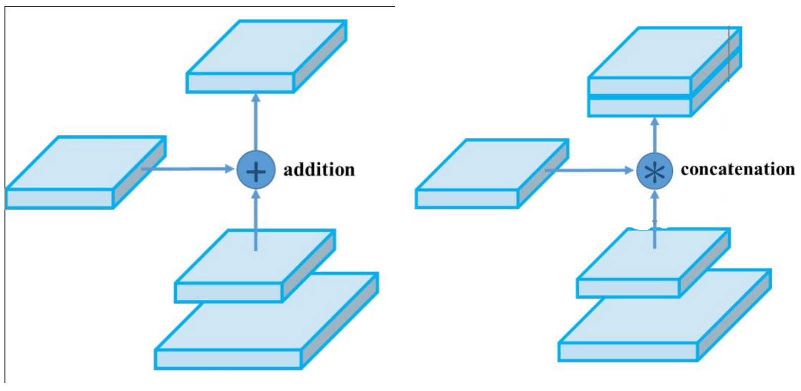


Figure 5: Architecture of the model from Qian et al. We are interested in the module shown in (b), where particular regions of different feature maps are concatenated together

The Path Aggregation Network is a sort of multi layer feature fusion module. Following the same principle as the MLFF from Qian *et al.*, PAN takes the features maps generated by a Feature Proposal Networks. It first downscales the lower level layer (denoted P_i with a 3×3 convolution operation with a stride of 2. It then adds element wise this downscaled layer with the layer P_{i+1}. This is done iteratively until the up most layer is attained. **In the YOLOv4 paper, instead of adding the layers, a concatenation is done**.



(a) PAN (b) Modified PAN

Figure 6: PAN and its YOLO modification

2.4 Receptive Field Improvements

To improve the size of the receptive field, the modified SPP module from YOLOv3[12] would be used. Experimentations could be done using dilated convolutions, as seen in Yu *et al.*[13]

and Ju *et al.*. Dilated convolution improve the receptive field by increasing the kernel size without increasing the number of parameters in the kernel. This can help reducing the number of parameters, improving performance of the network in terms of Frames Per Seconds. We can replace some of the classical convolution blocks from the CSPDarknet53 backbone with dilated convolution modules, as can be seen in Figure 7.

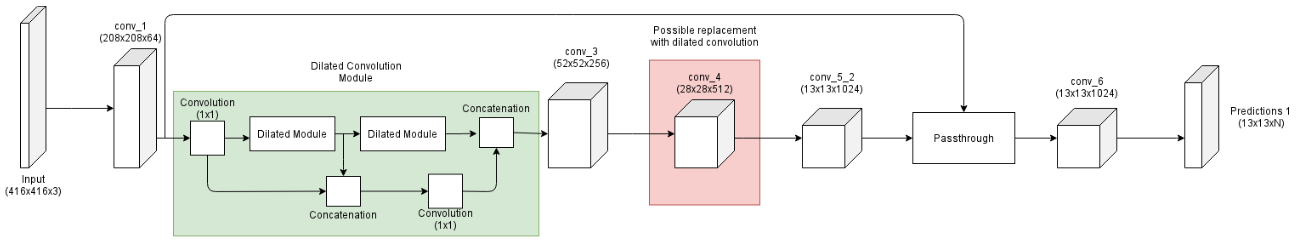
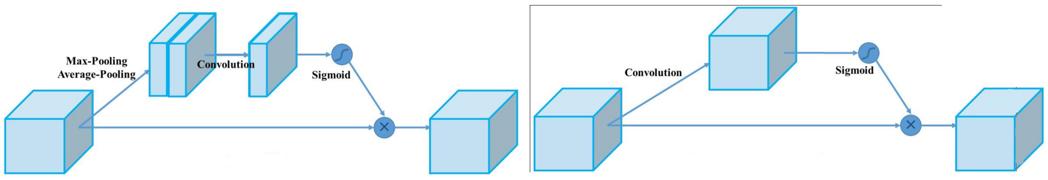


Figure 7: General Architecture of the Darknet Backbone with a dilated convolution module replacing a convolution block

2.5 Attention Modules

Attention modules are used to improve the accuracy of detections network by increasing the importance of particular regions or channels. The Spatial Attention Module[14] would be used, as it increases the accuracy without significantly impacting inference speed. Testing would have to be done to evaluate whether the SAM improvement done by Bochkovsky *et al.* in the YOLOv4 paper[2] gives a performance increase.



(a) SAM

(b) Modified SAM

Figure 8: SAM and its YOLO modification

2.6 Post-processing

Non maximal suppression needs to be used to remove the superfluous bounding boxes generated by the network. Soft Non maximum suppression, as introduced by Qian *et al.* [11] would be used to achieve such a task, while trying to not miss the smaller objects.

2.7 List of all the possible testing arrangements

We will need to test a multiple of different modifications on the base CSPDarknet-53 and test out if those modifications results in performances improvements.

- Double YOLOv4 trained on large and small scale (similar to YOLT[10])
- CSPDarknet with one or two dilated convolution modules
- MLFF module from Zhuang *et al.*: see Figure 4

- MLFF module from Qian *et al.*: see Figure 5
- Modified PAN module: see Figure 6
- Modified SAM Module: see Figure 8
- Different activations: ReLU, Leaky ReLU, SELU, Mish or Swish

3 Training and Evaluation

3.1 Loss System

The Generalized Intersection over Union, an improvement on the traditional Intersection over Union performance metric would be used to more accurately measure the accuracy of the bounding boxes. The GIoU is defined as follows:

$$GIoU = IoU + \frac{area(B_{GT} \cup B_{PT}}{area(B_{EC}} - 1 \tag{1}$$

With IoU being defined in Equation 2

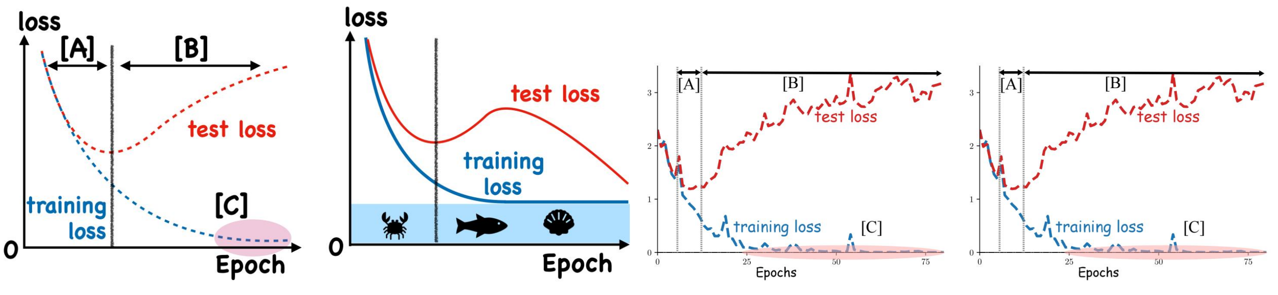
$$IoU = \frac{area(B_{GT} \cap B_{PT}}{area(B_{GT} \cup B_{PT}} \tag{2}$$

In those equations, B_{GT} represents the bounding box ground truth, B_{PT} represents the predicted bounding box and B_{EC} represents the smallest enclosing box of B_{GT} and B_{PT}.

The GIoU loss is a better measure of the quality of the predicted bounding box. Qian *et al.* also presents a new loss system that has a stronger gradient when the predicted bounding box is farther away to the truth. This loss system is shown in Equation 3

$$L_{GIoU} = 2 \times log_2 - 2 \times log(1 + GIoU) \tag{3}$$

In a recent paper by Ishida *et al.*[15], a new kind of regularization is presented, where **the model is prevented from reaching a zero training loss, even though it has zero training error**. This technique, called *flooding* intentionally prevents further reduction of he training loss when it reaches a certain low value: the *flooding level*. This approach makes the loss move around the flooding level by doing gradient descent if the loss is above the flooding level, but gradient ascent if it is below. This technique allows the network to obtain better generalization results and is very easily implementable.



(a) Without Flooding (b) With Flooding (c) CIFAR-10 without flooding (d) CIFAR-10 with flooding

Figure 9: Subfigure (a) shows the different regime during which a network first learns, then overfit. During [A], the network learns correctly, and both the training loss and the test loss decreases. During [B] the training loss continues to decreases, but the test loss increases; this is overfitting. In [C] the training loss is nearly zero. The authors show a way to avoid [C] by flooding the bottom area, forcing the loss to stay around a constant, which leads to a decreasing test loss. This is seen in (b) and (c) on the CIFAR-10 dataset

By letting the loss constant, the model hopefully drift towards a subspace where the loss landscape is flat, leading to better generalization. The authors tested this approach with various datasets, namely MNIST[16], Fashion-MNIST[17], CIFAR-10[18] and often obtained better results and in nearly every datasets, suggesting this approach is effective. Considering the ease of implementation, the lack of performance degradation and the possible improvements on performance, this technique could be very worthwhile.

Finally, in the YOLOv4[2] paper, the author mention the use of **self adversarial training**. In this training regime, the network first tries to alter the original image instead of its weights. This is an adversarial attack on itself, modifying the original image to fool itself into a wrong inference. Then, the network is trained to detect an object on this modified image in the normal way.