Comparative Study of Activation Functions and Their Impact on the YOLOv5 Object Detection Model

John Doherty[1](✉), Bryan Gardiner[1], Emmett Kerr[1], Nazmul Siddique[1], and Sunilkumar S. Manvi[2]

[1] Intelligent Systems Research Centre, Ulster University, Northland Road, Derry BT48 7JL, United Kingdom
{doherty-j92,b.gardiner,ep.kerr,nh.siddique}@ulster.ac.uk

[2] School of Computer Science and Engineering, REVA University, Karnataka 560064, India
Ssmanvi@reva.edu.in

Abstract. Object detection is an important aspect of computer vision research, involving determining the location and class of objects within a scene. For an object detection system to run in real-time, it is vital to minimise the computational costs while maintaining an acceptably high accuracy. In a Convolutional Neural Network (CNN) there is a direct correlation between the accuracy and the computational cost incurred by increasing the number of layers. Activation functions play a key role in a CNN to utilise nonlinearity to help balance the computational cost and accuracy. In this paper, a series of improvements are proposed to the state-of-the-art one-stage real-time object detection model, YOLOv5, providing the capability to enhance the overall performance. The validity of replacing the current activation function in YOLOv5, Swish, with a variety of alternative activation functions was investigated to aid in improving the accuracy and lowering the computational costs associated with visual object detection. This research demonstrates the various improvements in accuracy and performance that are achievable by appropriately selecting a suitable activation function to use in YOLOv5, including ACON, FReLU and Hardswish. The improved YOLOv5 model was verified utilising transfer learning on the German Traffic Sign Detection Benchmark (GTSDB) achieving state-of-the-art performance.

Keywords: Activation function · Deep learning · Object detection · YOLO

1 Introduction

Since the launch of the Microsoft Common Objects in Context (MSCOCO) dataset in 2015, many research teams have been striving to achieve state-of-the-art object detection performance on this dataset. Many researchers have used Convolutional Neural Network (CNN) based object detectors. These detectors can be separated into one-stage and two-stage detectors. Two-stage visual object detection models have seen a vast improvement in accuracy, however, much of this comes at the cost of increased inference time. For a system to be capable of running in real-time, it is vital that computational costs are minimised while ensuring that the accuracy is not severely impacted. To enable this, a recent rise has been seen in the development and usage of CNN-based one-stage object detection models [1]. These one-stage object detection models can achieve impressive accuracy, rivalling that of some two-stage object detectors, while running with substantially lower computational costs. This allows for these one-stage object detectors to run at incredible speeds on standard computer hardware or embedded systems [2]. While developing these one-stage object detection models, an extremely wide variety of activation functions have been utilised [3–5].

Activation functions are utilised in deep learning networks to enable the transformation of a weighted input signal to an output signal, which in turn enables the transfer of this output signal to the next layer in the network. The prediction accuracy of a deep learning network relies heavily on a variety of factors, two of the most important are the number of layers within the network, and the activation function used [6]. The ideal number of layers that should be utilised as well as the specific activation function that should be chosen varies based on the application of the network. Adding additional layers to a deep learning network may positively impact its performance, but it will do so at great cost to computational time [7]. To offset the increase in computational time, it is an option to alter the activation function to achieve improved performance.

An activation function creates a differentiator between a deep learning network and a linear regression model. If a deep learning network has no activation function, the predicted output will be proportional to the provided input [8]; similarly, if a linear activation function is utilised, the output will be similar to the input, with the addition of some small error. A non-linear activation function enables a deep network to take full advantage of its ability to learn from noisy data, such as the errors present in real-world input data [9]; in a CNN, this can be seen as a non-linear activation function being applied after each convolutional layer, with the activation function converting the output from the convolutional layer into a suitable input for the next layer in a non-linear manner. This change is one of the key reasons deep learning networks have seen such widespread adoption in object detection, as the use of non-linear activation functions can allow for quicker interpretation of complex data without the need of adding additional hidden layers, helping to keep computational costs as low as possible.

In 2016, Redmon et al. [10] proposed the YOLO algorithm. YOLO, or *You Only Look Once*, has went through many iterations, through to YOLOv5 in 2020 [11]. This paper modifies the most modern, accurate and extremely efficient YOLOv5 providing an improved model which will enhance accuracy and reduce inference time and will be made available for use in many real-time object detection tasks. This is achieved by introducing a variety of state-of-the-art activation functions which are applied to a create a novel version of YOLOv5. The improved YOLOv5 models are then trained and tested on MSCOCO proposing a selection of models which outperform the standard implementation. The top performing models are then trained and tested on the German Traffic Sign Detection Benchmark (GTSDB) dataset achieving state-of-the-art performance for a real-time object detection method.

The remainder of this paper is organised as follows. Section 2 includes a brief overview of CNN architectures for general object detection followed by traffic sign object detection, while Sect. 3 discusses the architecture of YOLOv5 in detail, while also outlining the various activation functions that have been used throughout the experiments. Section 4 features details of the experimental work, alongside the analysis of the experimental results, while Sect. 5 concludes the paper and discusses the direction of future work.

2 Related Work

As previously discussed, CNN-based object detection models can be divided into a variety of categories, including one-stage, two-stage, anchor-based and anchor-free detectors. For the purpose of this research, focus will be placed on one-stage detectors as they achieve a good balance of inference time and computational overhead.

Feng et al. [12] proposed a task-aligned one-stage object detection model (TOOD) which utilises a task-aligned head and learning mechanism, aiming to provide balance between the classification and localisation tasks within object detection - TOOD achieves a Mean Average Precision (mAP) at 0.5 of 60.0 on MSCOCO.

Fully Convolutional One-Stage (FCOS) is a one-stage object detector which solves object detection in a per-pixel prediction fashion [13]. This network eliminates the need of anchor boxes, vastly reducing computational costs. FCOS achieved a peak mAP at 0.5 of 64.1 on MSCOCO.

Chen et al. [14] proposed a Location-Aware Multi-Dilation (LAMD) module, which embeds spatial information from the head into the classifier, which improves robustness to the shifting of bounding boxes. They successfully implemented LAMD into state-of-the-art one-stage object detectors and improved the performance of ResNet-50 mAP at 0.5 on MSCOCO from 55.0 to 59.7.

You Only Look Once (YOLO) is a family of object detectors which have pushed the envelope for performance of one-stage object detectors. YOLO revolutionized the one-stage object detection method, by only giving an image one full forward pass through a single neural network prior to classification [10]. YOLO splits the input image into a $s \times s$ grid pattern, with each grid cell being responsible for predicting bounding boxes and confidence scores for objects. YOLO features a variety of advantages when compared to other one-stage object detectors, including the utilisation of global context, as the network evaluates the entire image at test time, along with speed that is an order of magnitude faster than other comparable one-stage object detectors [15].

CNN-based object detection models have been used for a wide variety of tasks, but an increasingly popular task is real-time traffic sign detection due to the increase into research surrounding autonomous driving.

Zhu et al. [16] proposed a novel labelled traffic sign detection dataset consisting of over 100,000 images with 30,000 traffic-sign instances gathered from Tencent Street Views, called the Tsinghua-Tencent 100k (TT100K). They also designed an end-to-end CNN-based object detector which achieved 88% accuracy in traffic sign detection in TT100K, compared to Fast R-CNN which achieved 50% accuracy. The TT100K dataset features a few problems, including extreme class imbalance, along with 100,000 images, with only 10,000 of those containing a traffic sign, meaning 90% of the dataset are background images.

Zhang et al. [17] utilised a YOLOv3 based approach to achieve real-time detection on small traffic signs. Their model utilised data augmentation, mainly through image mixup. Alongside this, they also used a multi-scale spatial pyramid pooling block in Darknet53 to learn object features more comprehensively. While training the model on the TT100K dataset. They achieved a mAP of 86% in detection while running at 23.81 FPS.

Liang et al. [18] developed a two-stage network which utilises a deep feature pyramid architecture with lateral connections, along with a densely connected CNN to strengthen the feature transmission, leading to more accurate classification with less parameters. They achieved 95% Area Under Precision-Recall Curve (AUC) on traffic sign detection from the GTSDB, and 82%–100% Accuracy across multiple TT100K classes, approximately a 10%–15% improvement from Faster R-CNN in the same task.

Wang et al. [19] proposed a traffic sign detection method which utilises histogram of oriented gradient and a "coarse-to-fine" sliding window scheme to achieve high recall and precision ratios, while also being robust to adverse situations including back lighting, occlusion and low quality.

3 Methodology

3.1 YOLOv5 Network Architecture

As YOLOv5 is yet to have a published peer-reviewed scientific paper from the creator, the YOLOv5 GitHub page has served as the main source of information to date regarding the model [11]. The network structure of YOLOv5, presented in Fig. 1, can be described as follows:

- The backbone is a modified version of the original DarkNet [10] architecture used across the family of YOLO models. The version in YOLOv5 has been written in Python and modified to utilise Cross-Stage-Partial-Connections (CSP). This is based on the utilisation of CSPs in YOLOv4 [20]. The backbone receives input images with a $640 \times 640 \times 3$ resolution. This input is transformed into a $320 \times 320 \times 12$ feature map, followed by a convolutional operation of 32 kernels, becoming a $320 \times 320 \times 32$ feature map.
- The neck is a modified Path Aggregation Network (PANet). The PANet is utilised to generate feature pyramids at multiple scales to enhance the model's ability to detect and recognise objects of varying sizes [21].
- The head is a standardised YOLO head. It takes inputs at 3 scales (8, 16, 32), which enables it to detect small, medium, and large objects respectively. Previous single scale heads have been tested on YOLOv5 and perform worse than the current implementation.

There are a few features specific to YOLOv5 which help improve its performance over previous iterations, including the use of the BottleneckCSP module which performs feature extractions on the feature map, as well as the use of mosaic data augmentation and auto learning bounding box anchors. When compared with other large-scale CNNs, this module can help reduce gradient information duplication in the optimisation process

[22]. YOLOv5 has been noted to achieve similar or even higher accuracy at a lower computational cost when compared to YOLOv4 [23], while YOLOv4 achieved a ~10% higher mAP than YOLOv3 [24].

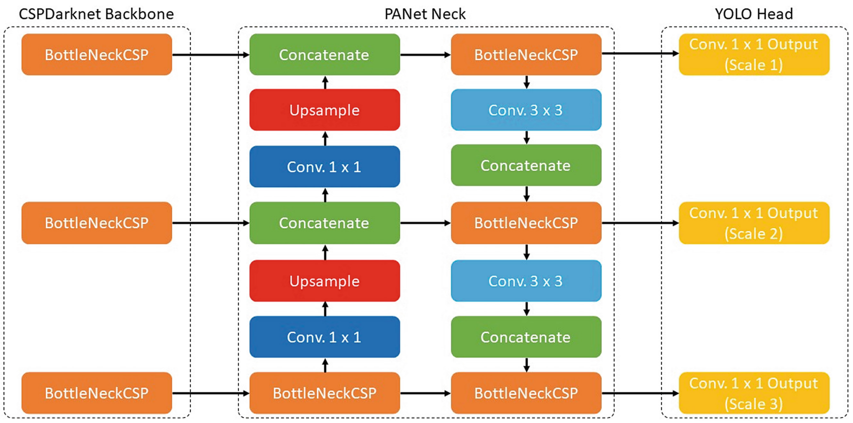

Fig. 1. YOLOv5 network structure

3.2 Activation Functions

There are a wide range of potential activation functions to choose from when attempting to create the most optimal object detector. As YOLOv5 is a one-stage detector with a large focus placed on computational efficiency, it is believed that the best activation functions to evaluate are those which will have a minimal impact on computational cost while simultaneously increasing accuracy.

In its default configuration, YOLOv5 uses the Swish activation function [25]. Swish can be best described as an activation function which nonlinearly interpolates between a standard linear function and the ReLU activation function [26]. Swish is generally viewed as a best-of-both-worlds scenario between a linear and a more complex non-linear activation function. It has been noted that Swish improves performance on ImageNet when compared to the ReLU and Sigmoid functions [27].

A Sigmoid activation function, sometimes referred to as a squashing function is much more complex than a linear activation function due to its use of an exponential term which causes increased computational cost, while also being much better at learning nonlinearity in patterns in wide datasets. A Sigmoid function has some major drawbacks, including sharp damp gradients during backpropagation from deeper hidden layers, along with gradient saturation and slow convergence [28].

The LeakyReLU activation function is an improved version of the ReLU [29] activation function. Whereas ReLU is a piecewise linear function that outputs the input directly if positive, otherwise, outputting zero, LeakyReLU provides a small positive slope for negative values [30]. This alleviates the dying ReLU problem, where nodes within a deep network are inactive and only output 0 for any input. LeakyReLU often performs on par or slightly better than ReLU [31].

The Flexible Rectified Linear Units (FReLU) activation function is a modified version of the ReLU activation function, where the rectified point of ReLU is redesigned as a learnable parameter [32]. FReLU tends to converge on a negative value, improving the expressiveness and performance while being no more computationally expensive than the original ReLU or LeakyReLU.

Hardswish is a modified version of the Swish activation function. Hardswish replaces the more computationally expensive sigmoid section of the Swish activation function with a piecewise linear analogue section [33], making Hardswish less computationally expensive than Swish while maintaining a similar accuracy.

Activate or Not (ACON) modifies the Maxout [34] activation function in a similar manner as to how Swish modifies ReLU. ACON has a dynamic non-linear degree with a switching factor that decays to zero as the non-linear function becomes linear. ACON has been noted to improve performance in ImageNet when compared to Swish and ReLU [35].

Mish is inspired by Swish, and uses the Self-Gating property where the non-modulated input it multiplied by the output of the non-linear function of the input [36]. Mish has shown improvements in both final accuracy and stability when compared with ReLU and Swish [37].

The mathematical functions and ranges of each activation function are represented in Table 1.

Table 1. Further activation function information

Activation function	Mathematical definition	Range (x)
Swish	$f(x) = x * \left(1 + \mathrm{e}^{-x}\right)^{-1}$	$-0.28, +\infty$
Sigmoid	$f(x) = \frac{1}{1+\mathrm{e}^{-x}}$	$0, +1$
LeakyReLU	$f(x) = \begin{cases} 0.01x & if\,(x < 0) \\ x & if\,(x \geq 0) \end{cases}$	$-\infty, +\infty$
FreLU	$f(x) = \begin{cases} x + b_L & if\,(x > 0) \\ b_L & if\,(x \leq 0) \end{cases}$	$-\infty, +\infty$
Hardswish	$f(x) = \begin{cases} 0 & if\,(x \leq -3) \\ x & if\,(x \geq +3) \\ x \cdot \frac{x+3}{6} & otherwise \end{cases}$	$-1, +\infty$
ACON	$f(x) = (P_1 - P_2)x \cdot \sigma(\beta(P_1 - P_2)x) + P_2 x$	$-\infty, +\infty$
Mish	$f(x) = \frac{\mathrm{e}^x \omega}{\delta^2}$	$-0.31, +\infty$

4 Experiments

4.1 Datasets

The experiments are conducted using two separate datasets. Firstly, the models are trained from scratch on the MSCOCO dataset. MSCOCO features over 328,000 images across 80 classes with an extremely wide range of images, alongside annotations for object detection including bounding boxes and per-instance segmentation masks.

The second dataset is the German Traffic Sign object detection dataset (GTSDB) containing 900 images across four classes; mandatory, danger, prohibitory and other. Each image in the dataset is fully annotated with bounding boxes and per-instance segmentation masks and is originally split into 600 training images and 300 evaluation images. As GTSDB is not natively supported by YOLOv5 due to the annotation type used, a pre-processing step of converting its annotations to a format which YOLOv5 requires was completed. During this, the dataset split was also changes to 80% training, 10% testing and 10% validation.

4.2 YOLOv5 Model Selection

YOLOv5 features four models: YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x. These models all include the core features of YOLOv5 but feature an increasing number of BottleneckCSP modules and hyperparameters, increasing both accuracy and inference time. The advertised inference time and mAP on the COCO test-dev dataset can be seen in Table 2.

Table 2. Comparison of YOLO models [11]

Model	Input size	mAP 0.5:0.95	mAP 0.5	Speed (ms)
YOLOv5s	640 × 640	36.7	55.4	2.0
YOLOv5m	640 × 640	44.5	63.1	2.7
YOLOv5l	640 × 640	48.2	66.9	3.8
YOLOv5s	640 × 640	50.4	68.8	6.1

Based on this, it was decided to utilise the YOLOv5l model as the basis for the object detection model as it is believed it has the best balance of accuracy to inference time.

4.3 Experimental Setup

The experiments feature a two-pronged approach. Initially, MSCOCO is utilised as the training dataset and the various YOLOv5 models are trained from scratch with modified activation functions. An activation function comparative analysis is conducted, comparing vital factors in performance including inference and training time, along with

Precision, Recall, also known as True Positive Rate (TPR) and equivalent to the inverse of the False Positive Rate (FPR), and mAP. Following this, the top performing models are used to complete a series of transfer learning experiments to determine which achieves the best performance on the GTSDB dataset.

4.4 Activation Function Comparative Analysis

A number of experiments were conducted using the YOLOv5 models with modified activation functions. Firstly, the models were trained on the MSCOCO dataset for a set number of 1000 epochs; this training occurred with default YOLOv5 dataset augmentation, with the resolution of each training image augmented from the original 640 × 480 pixels to 640 × 640 pixels. This is required as the default configuration of YOLOv5 expects a 640 × 640 input to map correctly to the three scaled outputs. Each experiment was run on the Kelvin2 Cluster, which is a part of the Northern Ireland High Performance Computing (NI-HPC) cluster [38]. The experiments were carried out on an Nvidia Tesla V100 graphics card with 32 GB of VRAM, along with 4 cores of an AMD EPYC 7702 64-Core Processor, acting as 8 data loaders, and a batch size of 32 (with the exception of ACON, requiring a batch size of 16 to fit into the 32 GB VRAM limit). It was found that the peak performance of each activation function occurred at different points in the training cycle, so the results for the peak performance have been shown, and the epoch this performance was achieved at has been noted; along with the time taken per epoch (in minutes). The results from these various experiments can be seen in Table 3.

Table 3. Experimental results from training from scratch on MSCOCO with modified activation functions on YOLOv5l

Activation function	Epoch	Precision	Recall	mAP 0.5	mAP 0.5:0.95	Inference speed (ms)	Time per epoch (min)	Total training time (hours)
Swish	274	0.706	0.602	0.648	0.452	6.8	35	159.8
ACON	272	0.694	0.611	0.650	0.456	10.5	110	498.7
FReLU	437	0.729	0.604	0.658	0.459	9.0	45	327.8
Hardswish	465	0.712	0.603	0.649	0.456	6.5	22	170.5
LeakyReLU	472	0.726	0.591	0.646	0.450	6.7	25	197.7
Mish	341	0.658	0.544	0.578	0.393	6.9	10	56.8
Sigmoid	799	0.724	0.583	0.633	0.439	6.7	24	319.6

Upon analysis of the obtained results, both Hardswish and LeakyReLU either match or exceed the default Swish activation function (in terms of mAP) while maintaining a similar inference time and reduced training time while some others, namely FReLU and ACON, achieve an extremely higher mAP at the cost of some computational overhead during both inference and training.

From these results it has been noted that FReLU is the best performer, achieving the highest Precision at 0.729, mAP 0.5 at 0.658 and mAP 0.5:0.95 at 0.459, while losing to ACON in Recall which scored 0.610. A close runner up to FReLU and ACON is Hardswish, achieving a Precision of 0.712, Recall of 0.602, mAP 0.5 of 0.649 and mAP 0.5:0.95 of 0.459. This is a drop of only 0.017, 0.001, 0.009 and 0.003 respectively, achieving the same training and inference time as the default Swish activation function.

While these results demonstrate improvement on the current state-of-the-art, it is important to note that FReLU took 45 min per epoch, almost double that of Swish, while also running 2.2 ms slower per image, equating to a real-world performance drop of 36 FPS. It is also important to note that each activation function performed similarly from a computational standpoint with the exception of FReLU and ACON. This is to be expected as these are much more complex activation functions which require greater computational overhead to utilise. Surprisingly, Mish completed each epoch in only 10 min, while still taking 6.9 ms per image.

It is vital to consider the combination of both the number of epochs required for an activation function to achieve its best performance, along with the time taken for each epoch to be completed. ACON achieves peak performance at only 272 epochs, but with each epoch taking over 110 min this equates to a real-world training time of almost 500 h, while FReLU requires 437 epochs at 45 min each, which is over 327 h of training. Hardswish achieves peak performance at 465 epochs, with each epoch only taking 25 min, meaning this is only 194 h of training, under half of the training time required for ACON and around 2/3 of the time required for FReLU.

For this reason, it is recommended that when aiming for the best balance of training time and performance, the Hardswish activation function should be selected to replace the standard Swish activation function in the YOLOv5l model, as Hardswish achieves better precision, recall, mAP 0.5 and mAP 0.5:0.95 for a marginal training time increase and slightly lower computational cost at run-time.

If a peak in accuracy is required while still being able to run in real-time, the recommendation is to select FReLU as the desired activation function. FReLU achieves the highest precision, mAP 0.5 and mAP 0.5:0.95 out of all tested activation functions, while still running at over 110 FPS.

4.5 Model Verification Using Transfer Learning

Following from the previous experiments and to verify the proposed models, further experiments were completed to investigate which YOLOv5 model will achieve the best performance on the GTSDB dataset. Transfer learning was utilised as GTSDB is a small dataset which is not well suited to full training from scratch. These experiments were completed on the same hardware and settings as previously used.

The fully trained Swish, FReLU and Hardswish models from the previous experiments were used as the basis for transfer learning. Using the weights generated from the previous experiment, two independent training sessions for each set of weights were created. Firstly, each model was trained with the first 24 layers frozen, leaving only the final fully connected layer unfrozen. This should provide the quickest training time, at the cost of some accuracy. Following this, the same initial weights were reused, and the experiment was repeated, but only the first 10 layers representing the backbone were frozen; this should slightly increase the training time but provide substantially higher accuracy. The results from the transfer learning experiment are presented in Table 4.

Table 4. Experimental results from transfer learning on GTSDB with modified activation functions on YOLOv5l

Activation function	Precision	Recall	mAP 0.5	mAP 0.5:0.95	Speed (ms)
10 layers frozen					
Swish	0.956	0.894	0.937	0.773	9.6
FReLU	0.950	0.915	0.944	0.774	13.0
Hardswish	0.972	0.937	0.960	0.801	9.4
24 layers frozen					
Swish	0.641	0.391	0.478	0.308	9.6
FReLU	0.600	0.555	0.579	0.406	12.8
Hardswish	0.650	0.346	0.455	0.266	9.9

The results from the transfer learning experiments show that impressive results can be achieved with minimal training. Each model took around 50–200 epochs to achieve their top performance, with each epoch taking between 3 and 5 s. The exact time per epoch or number of epochs for each result was not noted as they are within the margin of error, and peak performance can be achieved in a matter of minutes.

Some key observations can be made from the analysis of these results; firstly, the performance from freezing only the initial 10 layers is vastly superior to the performance of freezing all layers other than the final layer. This is to be expected, as the GTSDB dataset is quite small at only 900 images, and extremely different than the original MSCOCO dataset the network is pretrained on.

Secondly, similar comparisons to the original experiment on MSCOCO can be drawn. Both FReLU and Hardswish outperform Swish in mAP 0.5 and 0.5:0.95, while Hardswish has a marginally lower inference time, and FReLU has a significantly higher inference time. In contrast to the original experiment, Hardswish performs higher than FReLU across the board while running over 3 ms quicker per image.

The results from Hardswish with the first 10 layers frozen are state-of-the-art for the GTSDB dataset, outperforming previous examples such as Rajendran et al. [39] achieving a 0.922 mAP 0.5 at 100 ms per image on a modified YOLOv3 network.

The results presented are key in showing the improvements in accuracy that can be gained by improving the existing YOLOv5 model through changing the activation function with a choice of suggested activation functions. The improved YOLOv5 model achieves a higher accuracy than current state-of-the-art GTSDB models outlined in Sect. 2.

5 Conclusion

In this paper an in-depth analysis of the effects of changing the existing Activation Function in the YOLOv5 object detection model was completed. The experimental results show that a significant improvement in mAP can be found by replacing the standard Swish activation function with a variety of others, including FReLU, ACON and Hardswish. This work was expanded, showing the potential application of the improved YOLOv5l models on the task of real-time traffic sign object detection. This paper demonstrates the improvements in performance than be achieved by using a synergistic strategy focusing on activation function and transfer learning balance. Future work will focus on further improvements to YOLOv5, including modifying model structure, potentially replacing the PANet neck, as well as improving the top performing YOLOv5l models through further hyperparameter tuning utilising a genetic algorithm.

References

1. Sultana, F., Sufian, A., Dutta, P.: A review of object detection models based on convolutional neural network. In: Advances in Intelligent Systems and Computing, pp. 1–16 (2020)
2. Soviany, P., Ionescu, R.T.: Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction. In: 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC (2018)
3. Hou, Q., Xing, J.: KSSD: single-stage multi-object detection algorithm with higher accuracy. IET Image Process. **14**(15), 3651–3661 (2020). https://doi.org/10.1049/iet-ipr.2020.0077
4. Kim, S., Kim, H.: Zero-centered fixed-point quantization with iterative retraining for deep convolutional neural network-based object detectors. IEEE Access **9**, 20828–20839 (2021). https://doi.org/10.1109/ACCESS.2021.3054879
5. Shakarami, A., Menhaj, M.B., Mahdavi-Hormat, A., Tarrah, H.: A fast and yet efficient YOLOv3 for blood cell detection. Biomed. Sig. Process. Control **66**, 102495 (2021). https://doi.org/10.1016/j.bspc.2021.102495
6. Goyal, M., Goyal, R., Reddy, P.V., Lall, B.: Activation functions. In: Pedrycz, W., Chen, S.-M. (eds.) Deep Learning: Algorithms and Applications. SCI, vol. 865, pp. 1–30. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-31760-7_1
7. Patel, S., Patel, A.: Object detection with convolutional neural networks. In: Joshi, A., Khosravy, M., Gupta, N. (eds.) Machine Learning for Predictive Analysis. LNNS, vol. 141, pp. 529–539. Springer, Singapore (2021). https://doi.org/10.1007/978-981-15-7106-0_52
8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
9. Li, S., Chen, S., Liu, B.: Accelerating a recurrent neural network to finite-time convergence for solving time-varying Sylvester equation by using a sign-bi-power activation function. Neural Process. Lett. **37**, 189–205 (2013). https://doi.org/10.1007/s11063-012-9241-1
10. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788. IEEE (2016)
11. Jocher, G.: YOLOv5 Github. https://github.com/ultralytics/yolov5
12. Feng, C., Zhong, Y., Gao, Y., Scott, M., Huang, W.: TOOD: task-aligned one-stage object detection. In: IEEE/CVF International Conference on Computer Vision, pp. 3510–3519 (2021)
13. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: fully convolutional one-stage object detection. In: IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9626–9635. IEEE (2019)

14. Chen, Q., Wang, P., Cheng, A., Wang, W., Zhang, Y., Cheng, J.: Robust one-stage object detection with location-aware classifiers. Pattern Recogn. **105**, 107334 (2020). https://doi.org/10.1016/j.patcog.2020.107334
15. Hui, J.: Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3. https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088
16. Zhu, Z., Liang, D., Zhang, S., Huang, X., Li, B., Hu, S.: Traffic-sign detection and classification in the wild. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2110–2118 (2016)
17. Zhang, H., et al.: Real-time detection method for small traffic signs based on Yolov3. IEEE Access **8**, 64145–64156 (2020). https://doi.org/10.1109/ACCESS.2020.2984554
18. Liang, Z., Shao, J., Zhang, D., Gao, L.: Traffic sign detection and recognition based on pyramidal convolutional networks. Neural Comput. Appl. **32**(11), 6533–6543 (2019). https://doi.org/10.1007/s00521-019-04086-z
19. Wang, G., Ren, G., Wu, Z., Zhao, Y., Jiang, L.: A robust, coarse-to-fine traffic sign detection method. In: International Joint Conference on Neural Networks (IJCNN), pp. 1–5. IEEE (2013)
20. Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M.: YOLOv4: optimal speed and accuracy of object detection. arXiv (2020)
21. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8759–8768. IEEE (2018)
22. Tan, S., Lu, G., Jiang, Z., Huang, L.: Improved YOLOv5 network model and application in safety helmet detection. In: IEEE International Conference on Intelligence and Safety for Robotics (ISR), pp. 330–333. (2021)
23. Nelson, J.: YOLOv5 is here: state-of-the-art object detection at 140 FPS. https://blog.roboflow.com/yolov5-is-here/. Accessed 11 Dec 2021
24. Ampadu, H.: Yolov3 and Yolov4 in object detection. https://ai-pool.com/a/s/yolov3-and-yolov4-in-object-detection
25. Solawetz, J., Nelson, J.: YOLOv5 improvement strategy. https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/. Accessed 10 Dec 2021
26. Ramachandran, P., Zoph, N., Le, Q. V.: Searching for activation functions. In: 6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings (2018)
27. Ye, A.: Swish: booting ReLU from the activation function throne. https://towardsdatascience.com/swish-booting-relu-from-the-activation-function-throne-78f87e5ab6eb. Accessed 13 Dec 2021
28. Nwankpa, C., Ijomah, W., Gachagan, A., Marshall, S.: Activation Functions: Comparison of trends in Practice and Research for Deep Learning (2018)
29. Shen, F., Gan, R., Zeng, G.: Weighted residuals for very deep networks. In: 3rd International Conference on Systems and Informatics (ICSAI), pp. 936–941 (2016)
30. Xu, J., Li, Z., Du, B., Zhang, M., Liu, J.: Reluplex made more practical: leaky ReLU. In: IEEE Symposium on Computers and Communications (ISCC), pp. 1–7 (2020)
31. Khalid, M., Baber, J., Kasi, M.K., Bakhtyar, M., Devi, V., Sheikh, N.: Empirical evaluation of activation functions in deep convolution neural network for facial expression recognition. In: 43rd International Conference on Telecommunications and Signal Processing (TSP), pp. 204–207 (2020)
32. Qiu, S., Xu, X., Cai, B.: FReLU: flexible rectified linear units for improving convolutional neural networks. In: 24th International Conference on Pattern Recognition (ICPR), pp. 1223–1228 (2018)
33. Howard, A., et al.: Searching for MobileNetV3. In: IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1314–1324. IEEE (2019)

34. Castaneda, G., Morris, P., Khoshgoftaar, T.M.: Evaluation of maxout activations in deep learning across several big data domains. J. Big Data **6**(1), 1–35 (2019). https://doi.org/10.1186/s40537-019-0233-0
35. Ma, N., Zhang, X., Liu, M., Sun, J.: Activate or not: learning customized activation (2020)
36. Misra, D.: Mish: a self regularized non-monotonic activation function (2019)
37. Wright, L.: Meet Mish—new state of the art AI activation function. The successor to ReLU? https://lessw.medium.com/meet-mish-new-state-of-the-art-ai-activation-function-the-successor-to-relu-846a6d93471f. Accessed 09 December 2021
38. Northern Ireland High Performance Computing. https://www.ni-hpc.ac.uk/about/
39. Rajendran, S.P., Shine, L., Pradeep, R., Vijayaraghavan, S.: Real-time traffic sign recognition using YOLOv3 based detector. In: International Conference on Computing, Communication and Networking Technologies, ICCCNT (2019)