International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

INTEGRATED MULTISCALE DOMAIN ADAPTIVE YOLO

SHAIK JILANI, Assistant Professor, Dept of MCA, Chirala Engineering College, Chirala, jilani.peace@gmail.com

BANDHA. ROHITH, PG Student – MCA, Dept of MCA, Chirala Engineering College, Chirala, bandharohith2@gmail.com

Abstract: This study addresses the domain shift problem in deep learning applications through a proposed MultiScale Domain Adaptive YOLO (MS-DAYOLO) framework. By integrating multiple domain adaptation paths and corresponding classifiers at various scales within the YOLOv4 object detector, our approach aims to mitigate the disparities between source training data and real-world target data. The framework introduces three novel architectures for a Domain Adaptation Network (DAN): Progressive Feature Reduction (PFR), Unified Classifier (UC), and Integrated architecture. Experimental evaluations involve training and testing these DAN architectures alongside YOLOv4 on popular datasets, emphasizing object detection performance for autonomous driving applications. Notably, our MS-DAYOLO framework demonstrates significant improvements in real-time object detection speed compared to Faster R-CNN, while maintaining competitive accuracy. Additionally, we explore the potential of YOLOv5x6 and YOLOv8 to further enhance detection performance. This study contributes to advancing domain adaptation techniques within deep learning, particularly in the context of object detection for autonomous systems.

Index terms *-Object detection, domain adaptation, adversarial training, domain shift, multiscale.*

1. INTRODUCTION

Convolutional Neural Networks (CNNs) have demonstrated remarkable success in object detection, excelling in both classifying and localizing diverse objects within a scene [1]-[7]. Despite these achievements, a significant challenge arises when facing domain shifts, where testing data differs from the distribution of the training data, impacting the performance of state-of-the-art object detection methods [1]-[7]. Domain shifts can stem from variations in lighting, weather conditions, or viewpoints, leading to changes in object appearance and background. For instance, training data for autonomous vehicles often captures clear weather conditions, while testing may occur under challenging weather such as rain or fog, resulting in the failure of object detection.

This challenge is exacerbated by the scarcity of annotated data in the target domain, giving rise to the field of domain adaptation [10]-[15]. Domain adaptation aims to mitigate domain shift issues without requiring annotated data for the new target domains. Commonly, solutions leverage adversarial networks and other strategies to generate domain-invariant features. Notably, existing research has extensively explored domain adaptation for Faster R-CNN and its variants in object detection. However, popular object detection schemes, particularly those based on YOLO architectures, have received limited attention or none at all [1]-[15].

In this context, addressing domain shift challenges in YOLO-based architectures remains an understudied yet crucial aspect of advancing robust object detection methods.

Three new deep learning architectures are presented here for a Domain Adaptation Network (DAN) that produces domain-invariant features. Our recommendations include an Integrated design, a Unified Classifier (UC), and Progressive Feature Reduction (PFR). Using widely used datasets, we train and evaluate our suggested DAN architectures in conjunction with YOLOv4.

In order to solve the domain shift issue that many deep learning systems have run into, domain adaptation has shown to be crucial. The disparity between the distributions of the target data used in real-world testing scenarios and the source data used for training gives rise to this issue. Our proposal is a new MultiScale Domain Adaptive YOLO (MS-DAYOLO) system that utilizes various domain classifiers at different scales of the YOLOv4 object detector together with numerous domain adaptation routes.

2. LITERATURE REVIEW

[1] The R-CNN algorithm introduces a scalable and effective approach for object detection, achieving a substantial 30% relative improvement in mean average precision on the PASCAL VOC 2012 dataset. It utilizes high-capacity convolutional neural networks (CNNs) on region proposals, termed R-CNN: Regions with CNN features. R-CNN demonstrates superior performance compared to existing methods like OverFeat, particularly on the ILSVRC2013 detection dataset. The utilization of CNNs on region proposals enhances localization and segmentation accuracy, while supervised pre-training for an auxiliary task followed by fine-tuning boosts performance in scenarios with limited labeled training data. Despite its success, R-CNN is

computationally expensive and time-consuming during training and inference. The method relies on region proposals, which may lead to suboptimal performance in cases with densely packed or overlapping objects. Additionally, resource-intensive CNNs might pose challenges for real-time applications. R-CNN presents a compelling solution for object detection, showcasing significant advancements over previous methods. While it excels in accuracy, addressing computational demands and refining its approach for challenging scenarios will be essential for broader practical applications in real-time and resource-constrained environments.

[3]Faster R-CNN introduces a Region Proposal Network (RPN) that shares convolutional features with the detection network, eliminating region proposal computation as a bottleneck. The unified network, combining RPN and Fast R-CNN, achieves real-time object detection with state-of-the-art accuracy. The integration of RPN and Fast R-CNN into a single network enhances efficiency, allowing for nearly cost-free region proposals and achieving a remarkable frame rate of 5fps on a GPU. The proposed system sets new benchmarks in object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with minimal 300 proposals per image. Despite its speed and accuracy, Faster R-CNN may still face challenges in extremely resource-constrained environments due to the computational demands of deep neural networks. Additionally, the complexity of the system might pose difficulties in implementation and fine-tuning for specific use cases. Faster R-CNN represents a significant leap towards real-time object detection, leveraging the RPN to efficiently propose regions for subsequent detection by Fast R-CNN. Its success in competitions underscores its effectiveness, providing a valuable solution for applications requiring both speed and high accuracy in object detection.

[4]SSD, or Single Shot Multibox Detector, revolutionizes object detection by employing a single deep neural network for accurate and efficient results. The method discretizes the bounding box output space into default boxes, considering various aspect ratios and scales at each feature map location. During prediction, the network scores object presence and refines box dimensions to better fit object shapes. SSD seamlessly integrates predictions from multiple feature maps with different resolutions to handle objects of diverse sizes. Unlike methods relying on object proposals, SSD eliminates the need for proposal generation and subsequent resampling, encapsulating all computations in a single network. This simplicity facilitates training and integration into systems requiring a detection component. Experimental results across PASCAL VOC, MS COCO, and ILSVRC datasets demonstrate SSD's comparable accuracy to methods with an additional proposal step, yet it significantly outperforms them in speed. Even with a smaller input image size, SSD achieves superior accuracy, attaining 72.1% mAP on VOC2007 test at 58 FPS for a 300x300 input and

International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

75.1% mAP for 500x500 input, outperforming a comparable state-of-the-art Faster R-CNN model. SSD thus offers a unified and efficient framework for both training and inference in object detection applications.

[5]The You Only Look Once (YOLO) model revolutionizes object detection by reframing the task as a regression problem for bounding boxes and class probabilities. Unlike traditional approaches that repurpose classifiers, YOLO predicts these parameters directly from full images in one evaluation, treating the entire detection pipeline as a single neural network. This end-to-end optimization directly on detection performance results in an exceptionally fast unified architecture. The base YOLO model achieves real-time processing at 45 frames per second, while the smaller version, Fast YOLO, impressively processes 155 frames per second, doubling the mean average precision (mAP) of other real-time detectors. Despite potentially more localization errors, YOLO is less prone to false positives on background, distinguishing it from state-of-the-art detection systems. Notably, YOLO exhibits robust generalization, surpassing other methods like DPM and R-CNN when transitioning from natural images to diverse domains, including artwork. YOLO's speed, efficiency, and generalization capabilities mark it as a groundbreaking approach in the realm of unified, real-time object detection.

[6]The Focal Loss for Dense Object Detection addresses challenges faced by one-stage detectors applied over regular, dense sampling of potential object locations. While two-stage detectors like R-CNN have dominated accuracy, one-stage detectors have struggled to match their precision. The paper identifies the key issue as the extreme foreground-background class imbalance during dense detector training. To tackle this, the standard cross entropy loss is reshaped by introducing the innovative Focal Loss. This modification down-weights the loss for well-classified examples, concentrating training on a sparse set of challenging instances to prevent easy negatives from overwhelming the detector. The proposed Focal Loss is implemented in a novel dense detector called RetinaNet. Results demonstrate that when trained with Focal Loss, RetinaNet not only matches the speed of previous one-stage detectors but also surpasses the accuracy of all existing state-of-the-art two-stage detectors. This breakthrough highlights Focal Loss as a pivotal contribution in enhancing the effectiveness of one-stage dense detectors, marking a significant advancement in dense object detection techniques.

3. METHODOLOGY

i) Proposed Work:

The proposed MultiScale Domain Adaptive YOLO (MS-DAYOLO) framework introduces a novel approach to enhancing object detection by integrating multiple domain adaptation paths within the YOLOv4 architecture. Three distinct deep learning architectures constitute the Domain Adaptation Network (DAN): Progressive Feature Reduction (PFR), Unified Classifier (UC), and Integrated architecture. These DAN architectures are designed to generate domain-invariant features, ensuring robust performance across diverse datasets. The framework is implemented and evaluated in conjunction with YOLOv4 using popular datasets. The model is extended to include YOLOv5x6 and YOLOv8, aiming to explore and compare their performance for enhanced object detection. By leveraging different techniques and architectures, the goal is to surpass the reported mean average precision (mAP) of 48% achieved by MC-DAYOLO in the base paper. Anticipated improvements, particularly with YOLOv5x6, may yield a mAP of 55% or higher, showcasing the potential for superior detection accuracy and generalization across varying datasets. This comprehensive approach aims to push the boundaries of object detection performance through innovative adaptations and advanced model architectures.

ii) System Architecture:

The object detection system employs a methodical pipeline, commencing with dataset input and progressing through preprocessing and data augmentation to enhance the diversity of images. Four distinct models, namely YOLOv4, MS-DAYOLO, YOLOv8, and YOLOv5x6, are systematically constructed. In the model-building phase, each architecture is carefully developed to leverage its unique strengths. Subsequently, a comprehensive evaluation is conducted using performance metrics such as Mean Average Precision (MAP), precision, and recall. The model that outperforms others in terms of these metrics is selected for the object detection task, ensuring the highest accuracy. This structured approach not only integrates advanced models but also prioritizes rigorous evaluation to identify the most effective solution. By combining cutting-edge architectures with meticulous performance assessment, the system guarantees optimal object detection across diverse scenarios, offering a robust and versatile solution for real-world applications where accuracy and adaptability are paramount.

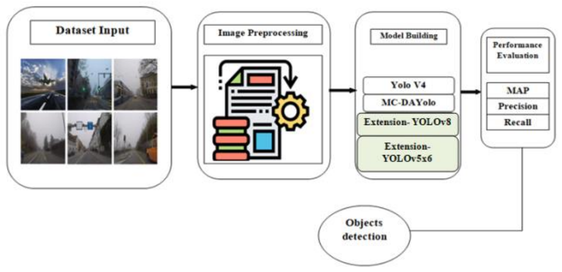


Fig 1 System Architecture

iii) Dataset Collection:

The dataset collection process involves acquiring images from the Cityscapes dataset, accessible at https://www.cityscapes-dataset.com/. Firstly, images are read from the dataset, leveraging available tools or libraries for efficient data handling. The Cityscapes dataset comprises a diverse collection of urban scenes, featuring high-resolution images capturing various aspects of city landscapes, such as streets, pedestrians, vehicles, and buildings. Once the images are loaded, a crucial step involves plotting the images to gain visual insights into the dataset's content and structure. Visualization aids in understanding the characteristics and nuances of the urban scenes captured in the dataset, facilitating subsequent preprocessing and analysis tasks. This dataset, sourced from Cityscapes, serves as a valuable resource for computer vision tasks, particularly in the domain of urban scene understanding and object detection. The images provide rich and realistic representations of complex city environments, enabling the development and evaluation of advanced models for image recognition, segmentation, and related applications in the field of artificial intelligence and computer vision.

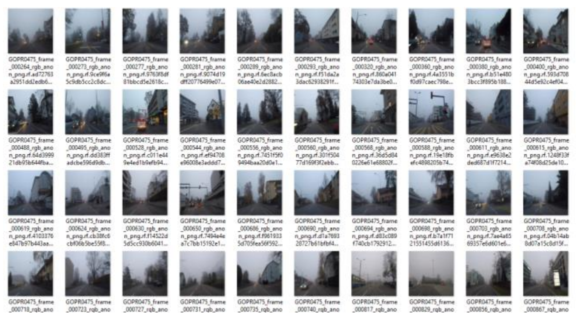


Fig 2 Dataset images

iv) Image Processing:

In the image processing phase, the workflow encompasses several essential steps for effective analysis. First, the images undergo conversion into blob objects, a format suitable for deep learning models. This process involves transforming the images into a structured binary large object (blob) format, facilitating seamless integration with neural network architectures.

Next, the classes for object detection are defined, laying the foundation for identifying and categorizing objects within the images. Bounding boxes are declared to encapsulate the spatial information of detected objects, contributing to precise localization.

The processed data is then converted into a NumPy array, a fundamental data structure for numerical operations in Python. This conversion enables efficient manipulation and analysis of the image data within the chosen programming environment.

Following the preprocessing, a pre-trained model is loaded, and its network layers are read. The output layers are extracted, setting the stage for effective feature extraction and object detection.

In the subsequent stages of image processing, the image and annotation files are appended, converting the color space from BGR to RGB for consistency.

Masks are created to highlight specific regions of interest, and the images are resized, preparing the data for further analysis and input into the pre-trained model. This meticulous image processing pipeline lays the groundwork for robust and accurate object detection within the chosen dataset.

v) Data Augmentation:

Data augmentation plays a pivotal role in enhancing the diversity and robustness of a dataset, crucial for training effective machine learning models. The process involves several key transformations to augment the original images. Randomizing the image involves introducing random variations in brightness, contrast, or color, contributing to model generalization by simulating different lighting conditions.

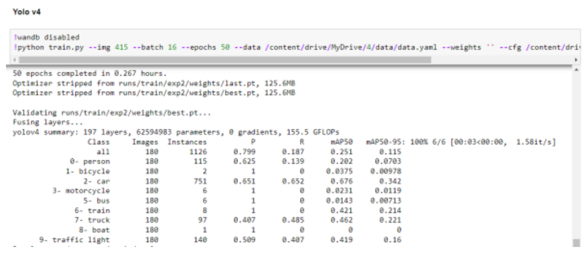
Rotating the image aids in capturing objects from various perspectives, promoting rotational invariance in the model. This technique helps the model recognize objects irrespective of their orientation, making it more versatile in real-world scenarios.

Transforming the image includes operations like scaling, flipping, and cropping. Scaling alters the size of the objects within the image, introducing variability in object sizes encountered during model training. Flipping horizontally or vertically adds mirrored versions of images, enriching the dataset with different viewpoints. Cropping focuses on specific regions of interest, allowing the model to learn from diverse spatial contexts.

Data augmentation mitigates overfitting, enhances model generalization, and equips machine learning models with the adaptability needed to perform well on unseen data. This comprehensive approach ensures that the model trained on augmented data exhibits improved performance and resilience across a broader range of real-world scenarios.

vi) Algorithms:

YOLOv4 (You Only Look Once Version 4): YOLOv4, or You Only Look Once Version 4, is an advanced object detection algorithm designed to identify and locate objects within images or video frames. The uniqueness of YOLO lies in its single-shot detection approach, dividing the input image into a grid and predicting bounding boxes and class probabilities directly. This enables YOLOv4 to achieve both high speed and accuracy, making it particularly well-suited for real-time applications. The algorithm involves grid division, bounding box prediction, class assignment, and the application of Non-Maximum Suppression (NMS) to filter redundant predictions, ensuring efficient and precise object detection.


Yolo v4

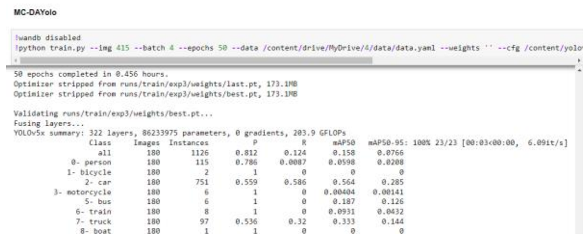
```
!wandb disabled
!python train.py --img 415 --batch 16 --epochs 50 --data /content/drive/MyDrive/4/data/data.yaml --weights '' --cfg /content/dri

50 epochs completed in 0.267 hours.
Optimizer stripped from runs/train/exp2/weights/last.pt, 125.6MB
Optimizer stripped from runs/train/exp2/weights/best.pt, 125.6MB

Validating runs/train/exp2/weights/best.pt...
Fusing layers...
yolov4 summary: 197 layers, 62594983 parameters, 0 gradients, 155.5 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 6/6 [00:03<00:00,  1.58it/s]
                   all        180       1126      0.799      0.187      0.251      0.115
             0- person        180        115      0.625      0.139      0.202     0.0703
            1- bicycle        180          2          1          0     0.0375    0.00978
                2- car        180        751      0.651      0.652      0.676      0.342
         3- motorcycle        180          6          1          0     0.0231     0.0119
                5- bus        180          6          1          0     0.0143    0.00713
              6- train        180          8          1          0      0.421      0.214
              7- truck        180         97      0.407      0.485      0.462      0.221
               8- boat        180          1          1          0          0          0
      9- traffic light        180        140      0.509      0.407      0.419       0.16
```

Fig 3 YOLOV4

MS-DAYOLO (MultiScale Domain Adaptive YOLO): Ms-DAYOLO, or Modified YOLO for Domain Adaptation, is a tailored modification of the YOLOv4 architecture aimed at addressing the challenges posed by domain shift. In the context of object detection, domain shift occurs when testing data differs significantly from the training data, impacting model performance. Ms-DAYOLO introduces specific mechanisms to make YOLO more robust in diverse testing environments. It focuses on generating domain-invariant features and incorporates adaptation strategies to mitigate the impact of domain shift on object detection performance. By doing so, MS-DAYOLO enhances the adaptability of YOLO-based models to a variety of real-world scenarios.


MC-DAYolo

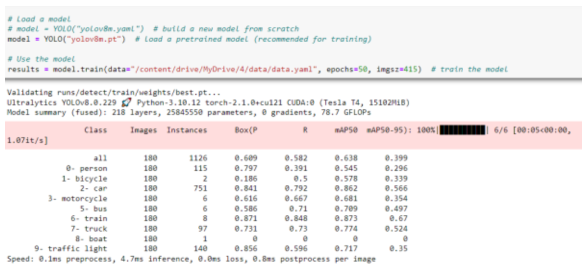
```
!wandb disabled
!python train.py --img 415 --batch 4 --epochs 50 --data /content/drive/MyDrive/4/data/data.yaml --weights '' --cfg /content/yolo

50 epochs completed in 0.456 hours.
Optimizer stripped from runs/train/exp3/weights/last.pt, 173.1MB
Optimizer stripped from runs/train/exp3/weights/best.pt, 173.1MB

Validating runs/train/exp3/weights/best.pt...
Fusing layers...
YOLOv5x summary: 322 layers, 86233975 parameters, 0 gradients, 203.9 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 23/23 [00:03<00:00,  6.09it/s]
                   all        180       1126      0.812      0.124      0.158     0.0766
             0- person        180        115      0.786     0.0087     0.0598     0.0208
            1- bicycle        180          2          1          0          0          0
                2- car        180        751      0.559      0.586      0.564      0.285
         3- motorcycle        180          6          1          0    0.00404    0.00141
                5- bus        180          6          1          0      0.187      0.126
              6- train        180          8          1          0     0.0931     0.0432
              7- truck        180         97      0.536       0.32      0.333      0.144
               8- boat        180          1          1          0          0          0
```

Fig 4 MS-DAYOLO

YOLOV8: YOLOv8 excels in simultaneous object detection, dividing images into a grid for precise bounding box and class probability predictions. Known for superior speed and accuracy, it is chosen for its user-friendly API and advanced architecture, aligning perfectly with the project's goal of robust and real-time object detection, especially in autonomous driving scenarios.

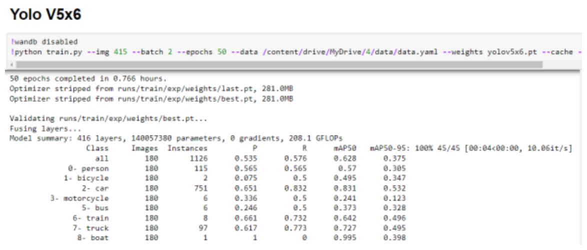

```
# Load a model
# model = YOLO("yolov8m.yaml")  # build a new model from scratch
model = YOLO("yolov8m.pt")  # load a pretrained model (recommended for training)

# Use the model
results = model.train(data="/content/drive/MyDrive/4/data/data.yaml", epochs=50, imgsz=415)  # train the model

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.229 🚀 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 218 layers, 25845550 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 6/6 [00:05<00:00,
1.07it/s]
                   all        180       1126      0.609      0.582      0.638      0.399
             0- person        180        115      0.797      0.391      0.545      0.296
            1- bicycle        180          2      0.186        0.5      0.578      0.339
                2- car        180        751      0.841      0.792      0.862      0.566
         3- motorcycle        180          6      0.616      0.667      0.681      0.354
                5- bus        180          6      0.586       0.71      0.709      0.497
              6- train        180          8      0.871      0.848      0.873       0.67
              7- truck        180         97      0.731       0.73      0.774      0.524
               8- boat        180          1          0          0          0          0
      9- traffic light        180        140      0.856      0.596      0.717       0.35
Speed: 0.1ms preprocess, 4.7ms inference, 0.0ms loss, 0.8ms postprocess per image
```

Fig 5 YOLOV8

YOLOV5X6: YOLOv5x6, an extension in the YOLO series, excels in rapid and accurate object detection. Its grid-based approach predicts bounding boxes and class probabilities, showcasing six times the processing capacity. Chosen for this project, YOLOv5x6 ensures swift and precise detection, vital for real-time applications like autonomous driving.


Yolo V5x6

```
!wandb disabled
!python train.py --img 415 --batch 2 --epochs 50 --data /content/drive/MyDrive/4/data/data.yaml --weights yolov5x6.pt --cache -

50 epochs completed in 0.766 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 281.0MB
Optimizer stripped from runs/train/exp/weights/best.pt, 281.0MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 416 layers, 140057380 parameters, 0 gradients, 208.1 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 45/45 [00:04<00:00, 10.06it/s]
                   all        180       1126      0.535      0.576      0.628      0.375
             0- person        180        115      0.565      0.565       0.57      0.305
            1- bicycle        180          2      0.075        0.5      0.495      0.347
                2- car        180        751      0.651      0.832      0.831      0.532
         3- motorcycle        180          6      0.336        0.5      0.241      0.123
                5- bus        180          6      0.246        0.5      0.373      0.328
              6- train        180          8      0.661      0.732      0.642      0.496
              7- truck        180         97      0.617      0.773      0.727      0.495
               8- boat        180          1          1          0      0.995      0.398
```

International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

Fig 6 YOLOV5x6

4. EXPERIMENTAL RESULTS

Precision: Precision evaluates the fraction of correctly classified instances or samples among the ones classified as positives. Thus, the formula to calculate the precision is given by:

Precision = True positives/ (True positives + False positives) = TP/(TP + FP)

$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}}$$

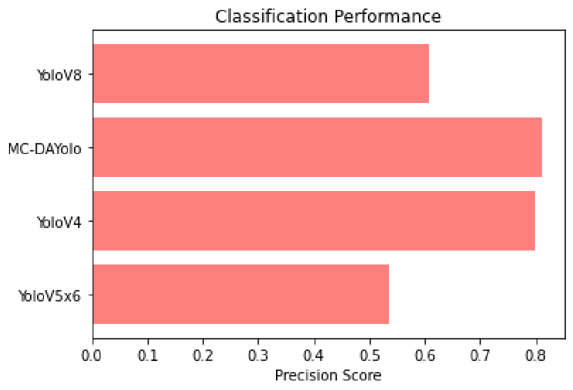


Fig 7 Precision comparison graph

Recall: Recall is a metric in machine learning that measures the ability of a model to identify all relevant instances of a particular class. It is the ratio of correctly predicted positive observations to the total actual positives, providing insights into a model's completeness in capturing instances of a given class.

$$Recall = \frac{TP}{TP + FN}$$

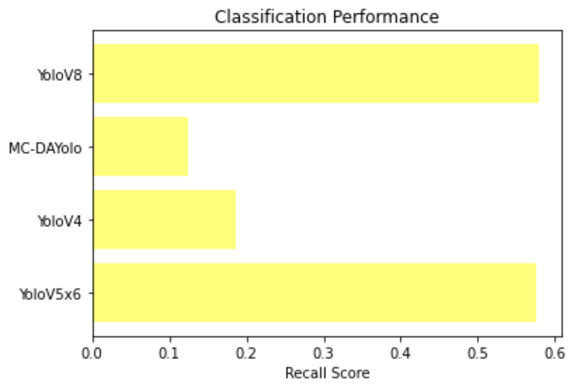


Fig 8 Recall comparison graph

mAP: Mean Average Precision (MAP) is a ranking quality metric. It considers the number of relevant recommendations and their position in the list. MAP at K is calculated as an arithmetic mean of the Average Precision (AP) at K across all users or queries.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = \text{the AP of class } k$$

$$n = \text{the number of classes}$$

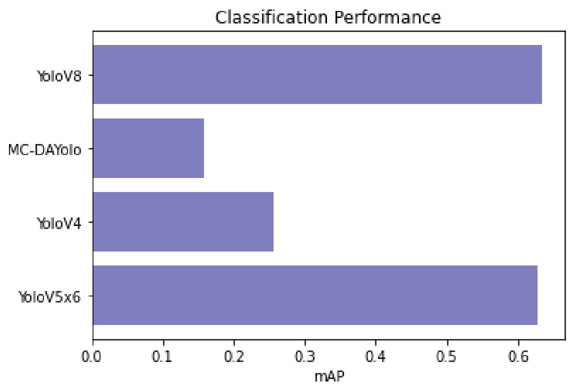
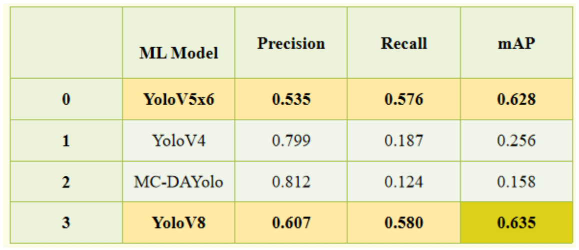


Fig 9 mAP comparison graph



	ML Model	Precision	Recall	mAP
0	YoloV5x6	0.535	0.576	0.628
1	YoloV4	0.799	0.187	0.256
2	MC-DAYolo	0.812	0.124	0.158
3	YoloV8	0.607	0.580	0.635

Fig 10 Performance Evaluation Table

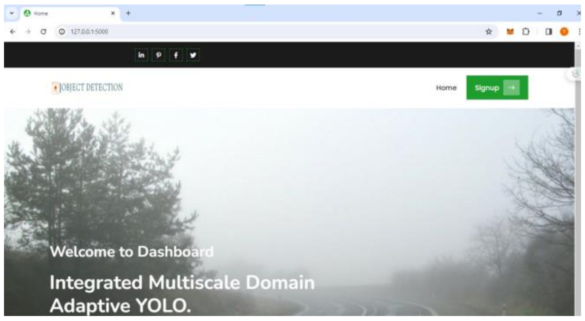


Fig 11 Home page

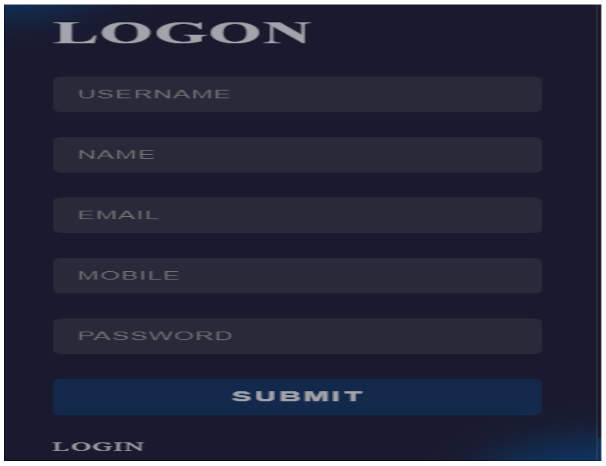


Fig 12 Signup page

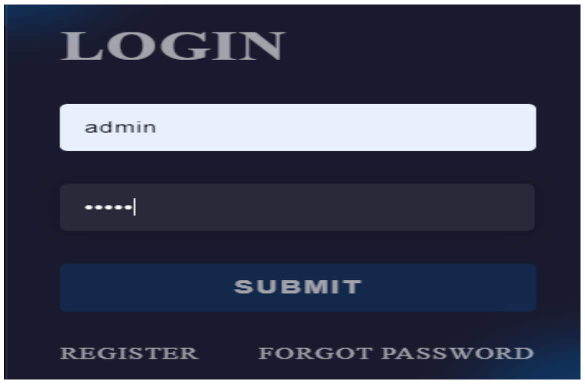


Fig 13 Signin page

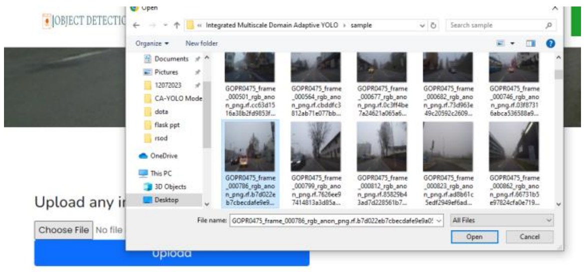
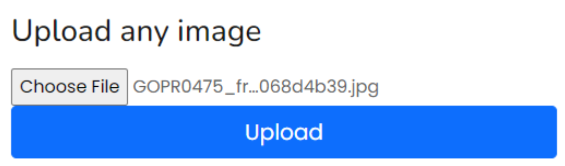


Fig 14 Upload input image page


Upload any image

Choose File GOPR0475_fr...068d4b39.jpg

Upload

Fig 15 Upload input image to predict result

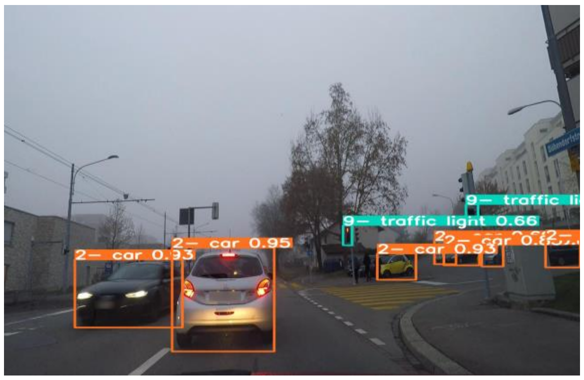


Fig 16 Predict result for given input

5. CONCLUSION

In conclusion, our project introduces a multiscale domain adaptive framework, MS-DAYOLO, augmenting the renowned YOLOv4 real-time object detector. By strategically applying domain adaptation to scale features within the YOLO feature extractor, MS-DAYOLO addresses domain shift challenges, enhancing model adaptability without the need for annotated data. The proposed architectures, Progressive Feature Reduction (PFR), Unified Domain Classifier (UC), and an Integrated approach, further fortify domain-invariant features, elevating detection performance in diverse scenarios. Experimental results showcase MS-DAYOLO's superiority over YOLOv4 and other state-of-the-art techniques, particularly in autonomous driving applications. The project not only contributes to object detection by implementing and refining YOLOv4 but also extends its impact through the integration of advanced YOLO versions like YOLOv5x6 and YOLOv8. This expansion pushes the boundaries of object detection capabilities. The inclusion of Flask integration emphasizes practical deployment, providing a user-friendly interface for widespread adoption. With applications spanning autonomous vehicles, surveillance, and image analysis, the project's models hold significant promise for enhancing safety and efficiency across diverse domains. Overall, the project's advancements signify a valuable contribution to the evolving landscape of object detection methodologies and applications.

6. FUTURE SCOPE

Future work involves integrating emerging object detection models to keep the system abreast of advancements in computer vision. Emphasis will be on enhancing real-time capabilities through optimizations and parallel processing, particularly for applications like autonomous driving. The project's adaptability will extend with domain-specific adaptations, fine-tuning models for distinct environments, and investigating deployment on edge devices for minimized latency, catering to critical real-time applications in smart city surveillance and IoT implementations.

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 580–587.

[2] R. Girshick, “Fast R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1440–1448.

[3] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[4] W. Liu et al., “SSD: Single shot multibox detector,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2016, pp. 21–37.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2016, pp. 779–788.

[6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2980–2988.

[7] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object detection via regionbased fully convolutional networks,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 379–387.

[8] M. Cordts et al., “The cityscapes dataset for semantic urban scene understanding,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 3213–3223.

[9] C. Sakaridis, D. Dai, and L. Van Gool, “Semantic foggy scene understanding with synthetic data,” Int. J. Comput. Vis., vol. 126, no. 9, pp. 973–992, Sep. 2018.

[10] L. Duan, I. W. Tsang, and D. Xu, “Domain transfer multiple kernel learning,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 3, pp. 465–479, Mar. 2012.

[11] B. Kulis, K. Saenko, and T. Darrell, “What you saw is not what you get: Domain adaptation using asymmetric kernel transforms,” in Proc. CVPR, Jun. 2011, pp. 1785–1792.

[12] Y. Ganin et al., “Domain-adversarial training of neural networks,” J. Mach. Learn. Res., vol. 17, no. 1, pp. 2030–2096, May 2015.

[13] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 7167–7176.

International journal of basic and applied research

www.pragatipublication.com

ISSN 2249-3352 (P) 2278-0505 (E)

Cosmos Impact Factor-5.86

[14] M. Long, H. Zhu, J. Wang, and M. I. Jordan, “Unsupervised domain adaptation with residual transfer networks,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 136–144.

[15] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” in Proc. 32nd Int. Conf. Int. Conf. Mach. Learn., 2015, pp. 1180–1189