OBJECT DETECTION ALGORITHM USING YOLOv3-PRUNING (TRANSFER) BASED ON TRANSFER LEARNING

[1]G. RAJESH BABU, [2]T. LAVANYA, [3]SK.MD. SHARMILA, [4]S.MANOJ, [5]N.D.NAGA RAJU
[1]Assistant Professor, Dept. of ECE, Usha Rama College of Engineering and Technology, Telaprolu, Krishna, Andhra Pradesh, India.
[2,3,4,5]B. Tech Students, Dept. of ECE, Usha Rama College of Engineering and Technology,Telaprolu, Krishna , Andhra Pradesh, India.

Abstract-

Object detection algorithms have gained great success in the field of machine vision in recent years. pursuing the discovery The model's accuracy, as well as the network's scale, are constantly growing, resulting in a continuous rise in computational cost and a high memory requirement. Because of the bigger network scale, their execution takes longer, requiring a trade-off between detection accuracy and execution speed. As a result, the created algorithm is incompatible with real-time applications. We suggest a novel method, the real-time object detection algorithm based on transfer learning, to improve the detection performance of small targets. Pruning is used to reduce the model's scale based on the baseline Yolov3 model, and migration learning is used to guarantee the model's detection accuracy. The object recognition technique based on transfer learning achieves a good balance between detection accuracy and inference speed, making it more suitable for real-time image processing. The experimental results from the evaluation of the dataset voc2007 + 2012 show that the parameters of the Yolov3-Pruning (transfer): model are reduced by 3X compared to the baseline Yolov3 model, and the detection accuracy is improved, achieving real-time processing and improving detection accuracy.

Keywords- Object detection, Transfer learning, Pruning, Detection accuracy

Introduction-

Deep neural networks have made substantial progress in the global wave of deep learning, solving problems ranging from image classification to reinforcement learning, and the field of computer vision has garnered unprecedented attention. Object detection, as a

difficult issue in this field, has also become one of the most popular research subjects among academics. Object detection tasks are more difficult to perform than deep learning programmes such as picture classification. Not only is it necessary to accurately locate the position of the object in various scenes, but it is also necessary to determine the category to which the object belongs. There may be more than one or two things in a complex scene. Multiple object detection in the same image will be more difficult to handle. As a result, object detection algorithms are bound to encounter issues such as a large quantity of computation and inference delay and will be unable to achieve real-time processing. With the rapid development of farming intensification, scale, and intelligence, sheep farming management quality and welfare healthier farming requirements are growing, and sheep identification is becoming increasingly essential to avoid diseases and improve sheep growth. The question of how to swiftly and efficiently complete the individual identification of sheep has become urgent. People's demand for mutton and goat milk products is

growing as their living standards improve, while quality standards are becoming increasingly stringent. Large-scale farming to improve the productivity and production level of the meat and dairy industries is thus an effective way to increase farmers' income, ensure food safety, improve the ability to prevent and control epidemics, and achieve coordinated development of animal husbandry and the environment.

Sheep identification is the foundation of intelligent farm management, and presently available methods for individual identification include hot iron tagging, freeze tagging, and ear notching. These methods have the potential to cause severe physical harm to animals. In the field of individual livestock identification, electronic identification methods such as Radio Frequency Identification (RFID) tags are extensively used. This technique, however, is costly and prone to issues such as unreadability and fraud. To address the aforementioned issues, experts have begun developing novel approaches based on deep learning methods. Deep learning algorithms are increasingly being used in the animal industry for a variety of uses. Biometric identification, activity tracking, body condition scoring, and behaviour analysis are examples of these uses. Because of the development of Convolutional Neural Networks (CNNs), using biometric traits instead of traditional techniques for identifying individual animals has recently gained focus. To apply deep learning techniques to biometric recognition, image data of particular body parts must be collected. Currently, a sheep retina, a bovine body pattern, and a pig's face are used. Obtaining clear data on the retina or other body parts, on the other hand, is extremely challenging. As a result, based on the improved YOLOv3, this paper suggests a contactless sheep face recognition technique. Uniqueness was one of the contactless identification method-based biometric characteristics. This technique cannot be readily forged or lost, making contactless identification methods more convenient, reliable, and accurate. Nasal print, iris recognition, and other non-contact animal identification techniques are commonly used. In comparison to contact identity methods (such as ear incision, hot iron branding, freezing hit number, and so on), the application range is broad, the identifying distance requirement is low, and the operation is straightforward. Animal image data can be acquired using portable photo devices such as cell phones and cameras, and the acquisition cost is low, resulting in significant savings in human resources. The non-contact identification technique does not require human-animal contact, which minimises animal harm and promotes animal growth and development.Biometric-based identification methods are mostly used in the identification of animals.

The feature extraction method using Gabor filter has been attempted by Tharawat et al. Support vector machine classifiers with different kernels (Gaussian, polynomial, linear) were compared. The results show that the classifier-based on Gaussian kernel is able to achieve 99.5% accuracy in the problem of nose-print recognition. However, the nasal pattern acquisition process is more difficult and the captured images contain much redundant information, so the method is not suitable for large-scale animal identification Tharawat et al. have tried the feature extraction technique using the Gabor filter. Support vector machine classifiers with various kernels (Gaussian, polynomial, linear) were compared. The findings demonstrate that the classifier based on the Gaussian kernel can

recognise nose prints with a 99.5% success rate. However, because acquiring the nasal pattern is more challenging and the images captured contain a lot of redundant information, the technique is not appropriate for extensive animal identification.One of the key characteristics for identifying an individual mammal is its iris, which contains spots, filaments, crowns, bands, and other shape features that do not change after birth and are distinctive. Atwo-dimensional complex wavelet transform feature method has been used in several studies to examine the iris features.

Literature review-

DETECTION OF OBJECTS:

The area of computer vision known as object detection is concerned with the identification, localization, and classification of things present in images and videos.

(Or)

Simply stated, drawing bounding boxes around things that are detected allows us to locate them in a scene. This is how object detection works. (or how they move through it).

MODELS FOR OBJECT DETECTION

Let's examine some well-known object detection models now that we are clear on what object detection is.

Faster R-CNN, Mask R-CNN, and R-CNN

The family of regionally based CNN models includes the most well-known object detection algorithms. The way Object Detection used to operate has been completely altered by this paradigm. They have improved in the last few years in both accuracy and efficiency. Many object detection algorithms, such as Faster R-CNN, continually strive to strike a balance between model accuracy and inference speed. Several object detection methods, including Yolov3, Yolov-tiny, and others.The object detection issue is used as the classification problem by R-CNN, which then extracts and categorises the features using the CNN model before identifying the particular content using RNN. As the name implies, faster R-CNN seeks to reduce latency by sharing the convolutional layer through the region proposal and then minimising numerous computations.

The fundamental idea behind Yolo is to approach the issue of object detection as a regression one. It is an end-to-end model that accepts the input picture at one end, processes it through the network framework, and outputs the desired boundary box coordinates and various category probabilities. Yolov3 employs the residual structure on the network, using the first 52 layers of convolution of darknet53 as the backbone network, which increases the model's scale while improving model detection accuracy. While small-scale target detection algorithms are more likely to find semantic information, large-scale target detection algorithms are more likely to find comprehensive information. Mask optimisation was used by Mask-Refned

R-CNN (MR R-CNN), which took this issue into account. Combining the global and detailed information element maps enhances the accuracy of image segmentation obviously. Chu and co.

YOLO and SSD

The single shot detector family includes a plethora of versions that were released in 2016.

SSDs outperform CNN models in terms of speed, but their precision rate is much lower. You only glance once, or YOLO, is very dissimilar from region-based algorithms. Yolo is faster than R-CNNs but falls short due to poor accuracy, just like SDDs. SDDs are the ideal option for mobile or embedded devices. One detection can lead to the forecast outcome. Examples of common algorithms are Yolo and SSD. Additionally, the term "two-stage" alludes to classifying and regressing data in two steps, first generating the candidate

area and then classifying the candidate area, which will require more time.

CenterNet

These object detection algorithms have become more and more well-liked in recent years. For object identification, CentreNet uses a crucial point-based methodology. This model shows to be both more precise and more efficient when compared to SSD or R-CNN methods. This method's slow training procedure is its only flaw.

Proposed method- YOLOV3-PRUNING

Based on Yolov3, Yolov3-Pruning has changed a few things. In the trimmed model is displayed. Yolov3-Pruning reduces the amount of network parameters by scaling down the model and pruning the network model's layer structure. We evaluated the performance of each layer structure, chose the one that had the least effect on the model's detection accuracy, and pruned it before removing the layer structure.

MODEL PRUNING ALGORITHM

Convolutional neural networks are becoming larger, so the model needs to do a lot of calculations and use a lot of memory. Even the reasoning time is too lengthy to achieve real-time performance. Therefore, based on the Yolov3 as a starting point, we made improvements to the object detection algorithm to increase application. We develop a pruning technique to shrink the size of the model and speed up inference. Several different tests were performed on the baseline Yolov3 for the model pruning portion, and various performances were analysed. . We examined every step of the Yolov3 object recognition algorithm, from the backbone network's feature value extraction to the prediction outcome. For 13 13, 26 26, and 52 52 feature layers, we computed the sum of three values for delay t, parameter m, and accuracy.

Algorithm 1 :Model pruning in Yolov3.

Input:

H_{input}:height of feature layer input
W_{input}:width of feature layer input
H_{output}:height of feature layer output
W_{output}:width of feature layer output
t_i:delay of feature layer
m_i:parameter quantity of feature layer
c_i:accuracy of feature layer
$channels_i$:the layers of the teacher network

Output:

$Total_i$: total evaluation index of feature layer

Procedure
//Obtain network models with pruned 13× 13, 26× 26 and 52× 52 feature layers
Execute training on each pruned model and obtain the detection accuracy c_i of each model.
Execute the test against the baseline Yolov3
if $H_{input} == 13$ **and** $W_{input} == 13$ **then**
    Initialize t_1,m_1
else if $H_{input} == 26$ **and** $W_{input} == 26$ **then**
    Initialize t_2,m_2
else if $H_{input} == 52$ **and** $W_{input} == 52$ **then**
    Initialize t_3,m_3
end if
if $H_{output} == 13$ **and** $W_{output} == 13$ **and** $channels_i == 75$ **then**
    $Total_1 = t_1 + m_1 + a_1$
else if $H_{output} == 26$ **and** $W_{output} == 26$ **and** $channels_i == 75$ **then**
    $Total_2 = t_2 + m_2 + a_2$
else if $H_{output} == 52$ **and** $W_{output} == 52$ **and** $channels_i == 75$ **then**
    $Total_3 = t_3 + m_3 + a_3$
end if
return argmax($Total_i$)

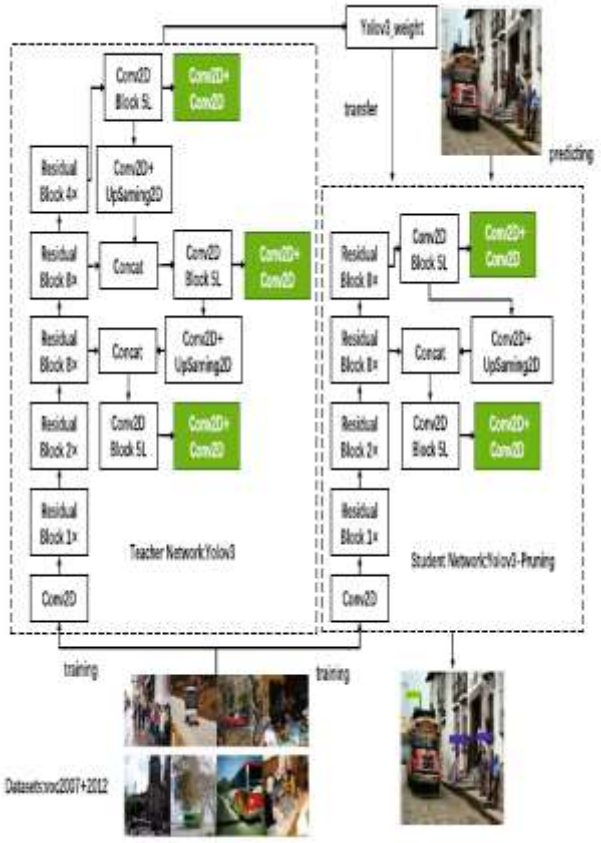


Fig: The overall framework structure of the object detection algorithm is based on transfer learning.

The basal Yolov3 is on the left, and the pruned Yolov3-Pruning is on the right. The accuracy of the small model increases as it picks up on the big model's capacity for generalisation.

The network models with pruned 13 13, 26 26, and 52 52 feature layers are first obtained for the model compression procedure. The trained pruned models are then used to determine the detection efficiency. The basic model is then put to the test to determine how long it takes the three feature layers to execute and how many parameters are employed. To choose which feature layer to use, the highest value of the sum of the delay, parameter amount, and detection accuracy is used.

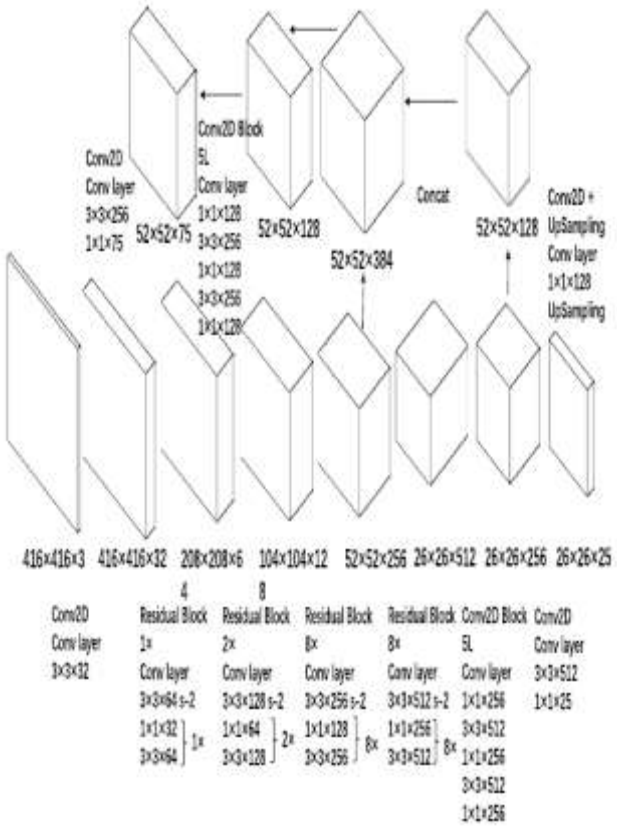


Fig: Yolov3-Pruning network structure

The design removes 9 layers of convolution computation from the baseline's baseline's 43 layers of convolution and 13 13 feature layers.We decide to prune the 13 x 13 feature layer using the compression pruning method mentioned above. The model's decreased parameter amount—from 235.37 to 76.13 MB—is 3X slower than the baseline model. Finally, we'll take additional steps to raise the model's detection precision.

Design of transfer learning:

The model's recognition accuracy must be decreased after the Yolov3-Pruning object detection algorithm prunes a portion of the object. After pruning, we use the transfer learning technique as a reference to increase

detection accuracy. The concept is distinct from conventional transfer learning, though.

To effectively transfer knowledge, it is essential to identify similarities between the new issue and the original problem. The previously learned model can be used in new areas with similar data or tasks. And to assist the pruned model increase detection accuracy, we draw on the original large-scale model's strong performance and generalizability.

Algorithm 2 :Transfer learning in teacher-student network.

Input:

D_{train}:the input training datasets of model
$weight_T$:the weight of the teacher network
$weight_S$:the weight of the student network
$bias_T$:the bias of the teacher network
$bias_S$:the bias of the student network
len_T:convolution layers in teacher network
len_S:convolution layers in student network
$layer_T$:the layers of the teacher network
$layer_S$:the layers of the student network

Output:

S_{pre}: pre-training parameters of the student network model

Procedure
//Obtain the network model parameters saved after the teacher network training
Execute training for D_{train} and obtain the $weight_T$ and $bias_T$ of each layer
for $i = 1; i < len_T; i++$ **do**
  for $j = 1; j < len_S; j++$ **do**
    if $layer_S == layer_T$ **then**
      $bias_S = weight_T$
      $bias_S = weight_T$
    end if
  end for
end for
Execute the student network to get the predicted result S_{pre}
return S_{pre}

The migration procedure of our object detection algorithm is summarised in Algorithm 2. The pupil network is initialised during the migration procedure. The pupil network is subjected to the weights and biases of a few layers in the teacher network, which are then used as pre-trained model parameters in the following formulas:

Yolo:

A real-time object recognition algorithm called YOLOv3 (You Only Look Once, Version 3) recognises particular items in videos, live feeds, or still images. To find an object, the YOLO machine learning algorithm uses features that a deep convolutional neural network has acquired. The YOLO machine learning algorithm has three versions, with the third version being a more accurate variant of the first ML algorithm. Versions 1-3 of YOLO were developed by Joseph Redmon and Ali Farhadi. Version 1 of YOLO was made in 2016, and version 3, which is the one that is heavily discussed in this piece, was made in 2018. An enhanced variant of YOLO and YOLOv2, YOLOv3 is available. The Keras or OpenCV deep learning libraries are used to create YOLO.

YOLOv4 and the recently published YOLOv7 (2022) are the recognised successors to YOLOv3, which represents the state-of-the-art. Artificial intelligence (AI) programmes use object classification systems to identify particular items in a class as interesting objects. The systems categorise objects in images into groups, placing objects with comparable traits together while ignoring others unless specifically instructed to do so. Using an end-to-end neural network to forecast bounding boxes and class probabilities simultaneously is what You Only Look Once (YOLO) suggests doing. It is distinct from the strategy used by earlier object recognition algorithms, which used classifiers as detectors. With a completely different approach to object detection, YOLO outperformed other real-time object detection algorithms and produced cutting-edge results. While Faster RCNN and similar algorithms use the Region Proposal Network to identify potential regions of interest before conducting recognition on each of those regions individually, YOLO performs all of its predictions using a single fully connected layer. Since the initial release of YOLO in 2015, several new versions of the same model have been suggested, each building on and improving its predecessor. Methods that use region proposal networks perform multiple iterations for the same

picture, whereas YOLO gets away with a single iteration.

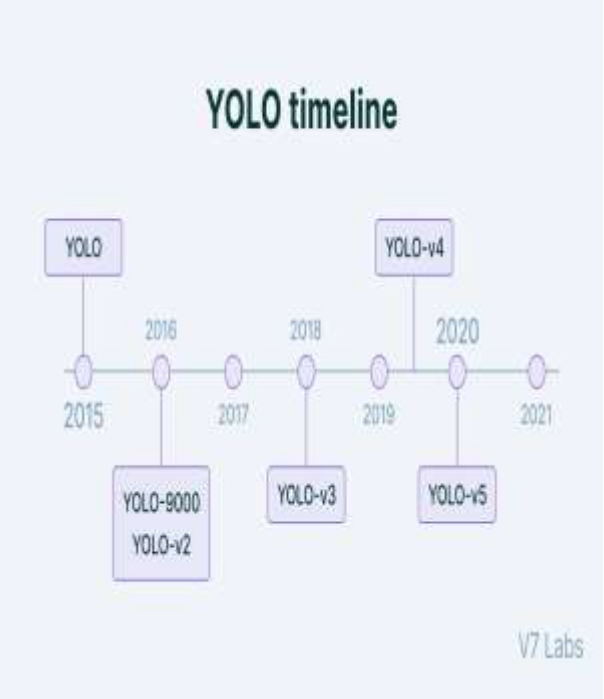


Fig: YOLO time line

How does YOLO work? YOLO Architecture-

The YOLO algorithm employs a straightforward deep convolutional neural network to identify objects in an input image. The CNN model's design, which serves as the foundation for YOLO, is displayed below.

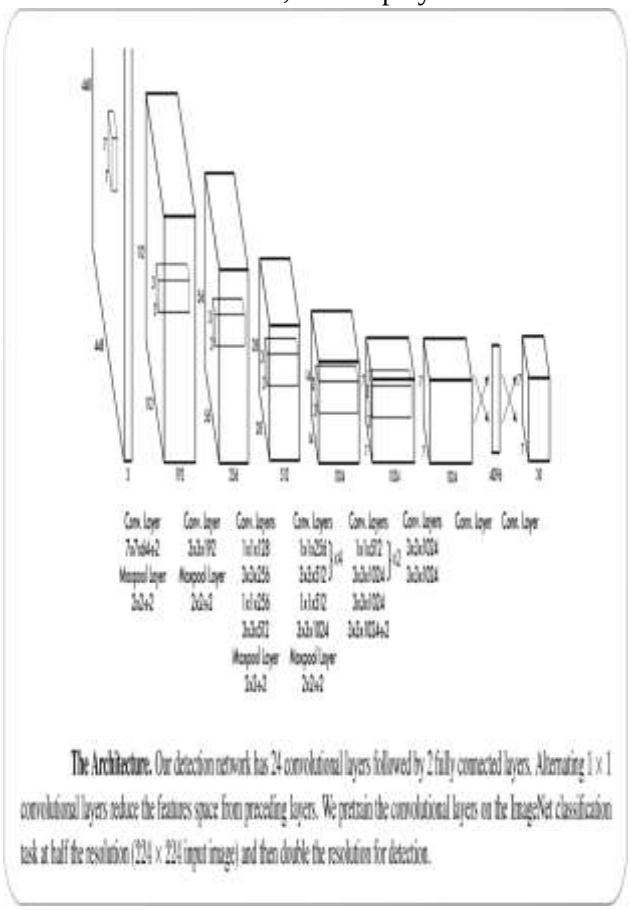


Fig: YOLO Architecture

By adding a temporary average pooling and completely connected layer, the model's first 20 convolution layers are pre-trained using ImageNet. The pre-trained model is then changed to conduct detection because prior studies have shown that a pre-trained network performs better when convolution and connected layers are added. The final completely connected layer of YOLO forecasts bounding box coordinates as well as class probabilities.

An input picture is divided into a S S grid by YOLO. A grid cell is in charge of detecting an object if its centre lies within that grid cell. The B bounding

boxes and confidence ratings for each box are predicted in each grid cell. These confidence scores demonstrate the model's level of assurance that the box holds an object and its estimation of the accuracy of the object.

YOLO v3-

The third iteration of the YOLO object recognition algorithm is known as YOLO v3. It was released in 2018 as an upgrade to YOLO v2 with the goal of enhancing the algorithm's precision and quickness.

The implementation of a new CNN architecture dubbed Darknet-53 is one of YOLO v3's key advancements. The ResNet architecture's Darknet-53 variant was created especially for object detection tasks. On a variety of object detection benchmarks, it can produce state-of-the-art outcomes thanks to its 53 convolutional layers. Anchor boxes with various sizes and aspect ratios are an addition to YOLO v3. Because the anchor boxes in YOLO v2 were all the same size, the algorithm had trouble detecting items of various sizes and shapes. Additionally, the idea of "feature pyramid networks" is introduced in YOLO v3. (FPN). A CNN architecture called FPNs is used to find things at various scales. They build a pyramid of feature maps, and they use each level of the pyramid to find items at various scales. Due to the model's ability to view the objects at different scales, this serves to enhance the detection performance for small objects.

YOLO v3 is also capable of handling a broader range of object sizes and aspect ratios. Additionally, compared to earlier YOLO versions, it is more reliable and precise.

Interpreting the output:

The convolutional layers' features are typically passed on to a classifier/regressor, which makes the detection forecast, as is the case for all object detectors. (coordinates of the bounding boxes, the class label... etc.). In YOLO, a convolutional layer that employs 1 x 1 convolutions is used to perform the forecast. First of all, you should note that our result is a feature map. The size of the prediction map is precisely the same as the size of the feature map before it because we used 1 x 1 convolutions.

Anchor Boxes:

Predicting the bounding box's breadth and height might seem logical, but in reality, doing so results in unstable gradients when training. Instead, the majority of contemporary object detectors forecast log-space transforms or merely offsets from anchors, which are predefined preset bounding boxes.

The forecast is then obtained by applying these transforms to the anchor boxes. Three anchors in YOLO v3 lead to the forecast of three bounding boxes for each cell. Returning to our original query, the bounding box whose anchor has the greatest IOU with the ground truth box will be the one that detects the dog.

Making Predictions:

The network output is transformed to produce bounding box forecasts using the following \formulae.

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

bx, *by*, *bw*, *bh* are the x,y center co-ordinates, width and height of our prediction. *tx*, *ty*, *tw*, *th* is what the network outputs. *cx* and *cy* are the top-left co-ordinates of the

grid. *pw* and *ph* are anchors dimensions for the box.

Center Coordinates:

Take note of the fact that we are applying a sigmoid function to our centre coordinates forecast. This requires the output number to fall between 0 and 1. What makes this the case? Adapt to me. The exact coordinates of the centre of the bounding box are typically not predicted by YOLO. It foresees the following offsets:

- Relative to the top left corner of the grid cell which is predicting the object.
- Normalised by the dimensions of the cell from the feature map, which is, 1.

Think about our dog picture as an example. If the centre is predicted to be at (0.4, 0.7), the centre is located at (6.4, 6.7) on the 13 x 13 feature grid. Given that the crimson cell's top-left coordinates are (6,6).

However, what if, say, the predicted x, y coords are larger than one? (1.2, 0.7). Consequently, the centre is at (7.2, 6.7). The centre is now in the cell directly across from our red cell, or the eighth cell in the seventh row. This contradicts the YOLO theory because, if the red box were to be the source of the dog's prediction, the dog's centre would have to be there rather than in the cell next to it. In order to solve this issue, the output is passed through a sigmoid function, which essentially keeps the centre of the predicting grid by squashing it into a range from 0 to 1. Bounding Box Dimensions

By applying a log-space transform to the output and multiplying by an anchor, the bounding box's dimensions are anticipated.

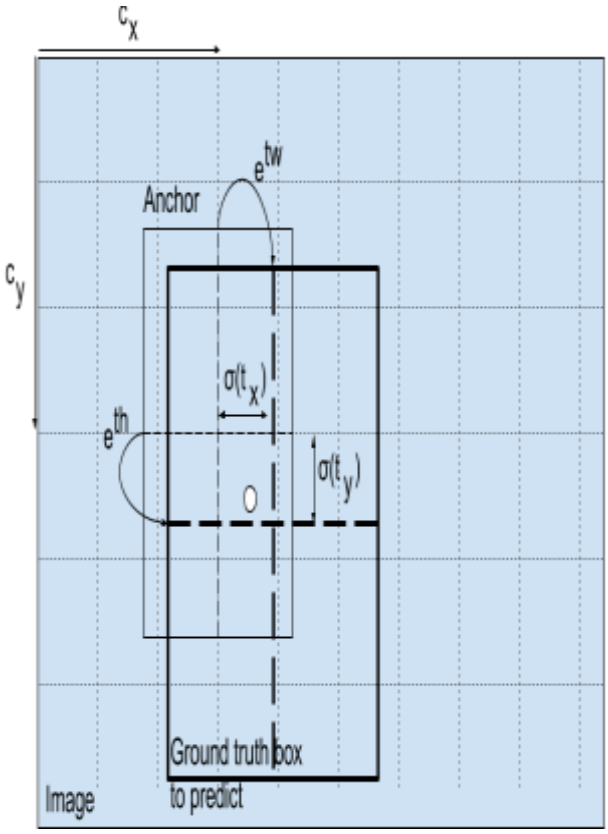


Fig: Dimensions of anchor box

The resultant predictions, *bw* and *bh*, are normalised by the height and width of the image. (Training labels are chosen this way). So, if the predictions *bx* and *by* for the box containing the dog are (0.3, 0.8), then the actual width and height on 13 x 13 feature map is (13 x 0.3, 13 x 0.8).

Objectness Score-

The likelihood that an item is contained inside a bounding box is indicated by the object score. For the red and surrounding grids, it should be close to 1, whereas for, say, the grid at the sides, it should be close to 0. The objectness value, which should be interpreted as a probability, is also put through a sigmoid.

Class Confidences-

The probabilities that the detected item belongs to a specific class are represented by

class confidences. (Dog, cat, banana, car etc.). YOLO used to soft max the class results prior to version 3. In contrast, that design decision has been abandoned in version 3, and writers have chosen to use sigmoid. The assumption that the classes are mutually exclusive when class scores are soft maxed is the cause. Simply put, it is guaranteed that an object cannot adhere to more than one class if it is a member of one. This is valid for the COCO database, on which our detection will be based.

When we have courses like Women and Person, though, these presumptions might not be accurate. Because of this, writers have refrained from using a Soft max activation.

CHANNEL PRUNING –

Despite being coarse-grained, channel trimming is a powerful technique. More significantly, the pruned model does not require specialised hardware or software to be implemented. Deep networks' channel trimming is primarily driven by the similarity between feature channels and convolutional kernels. To accomplish model compression, remove a feature channel and its corresponding convolution kernel.

Suppose the i-th convolutional layer of the network is x_i, and the dimension of the input feature map of this convolutional layer is (hi,wi,ni), where ni is the number of input channels, and hi,wi denote the height and width of the input feature map respectively The convolution layer transforms the map $x_i \in R^{ni+hi+wi}$ into the feature $x_{i+1} \in R^{ni+1 \times hi+1 \times wi+1}$ by ni+1 3D filters $F_{i,j} \in R^{ni \times k \times k}$ on ni channels, and is used as the input feature map for the next convolutional layer. Each of these filters consists of ni 2D convolution kernels $K \in R^{k \times k}$, and all filters together form the matrix $F_i \in R^{ni \times ni+1 \times k \times k}$. As shown in below figure, suppose the current convolutional layer is computed as $n_{i+1} n_i k^2 h_{i+1} w_{i+1}$. When the filter $F_{i,j}$ is pruned, its corresponding feature map $x_{i+1,j}$ will be deleted. This reduces $n_i k^2 h_{i+1} w_{i+1}$ times computation. At the same time, the filter corresponding to $x_{i+1,j}$ in the next layer will also be pruned, which again additionally reduces the computation of $n_{i+1} k^2 h_{i+2} w_{i+2}$. That is, the m filters in layer i are pruned, and the computational cost of layers i and j can be reduced at the same time, which finally achieves the purpose of compressing the deep convolutional network.

YOLO v3-

The third iteration of the YOLO object recognition algorithm is known as YOLOv3. It was released in 2018 as an upgrade to YOLO v2 with the goal of enhancing the algorithm's precision and quickness. The implementation of a new CNN architecture dubbed Darknet-53 is one of YOLO v3's key advancements. The ResNet architecture's Darknet-53 variant was created especially for object detection tasks. On a variety of object detection benchmarks, it can produce state-of-the-art outcomes thanks to its 53 convolutional layers. Anchor boxes with various sizes and aspect ratios are an addition to YOLO v3. Because the anchor boxes in YOLO v2 were all the same size, the algorithm had trouble detecting items of various sizes and shapes. To better match the size and shape of the items being detected, YOLO v3 changes the aspect ratios and scales the anchor boxes. Additionally, the idea of "feature pyramid networks" is introduced in YOLO v3. (FPN). A CNN architecture called FPNs is used to find things at various scales.

They build a pyramid of feature maps, and they use each level of the pyramid to find items at various scales.

Due to the model's ability to view the objects at different scales, this serves to enhance the detection performance for small objects.YOLO v3 is also capable of handling a broader range of object sizes and aspect ratios. Additionally, compared to earlier YOLO versions, it is more reliable and precise.

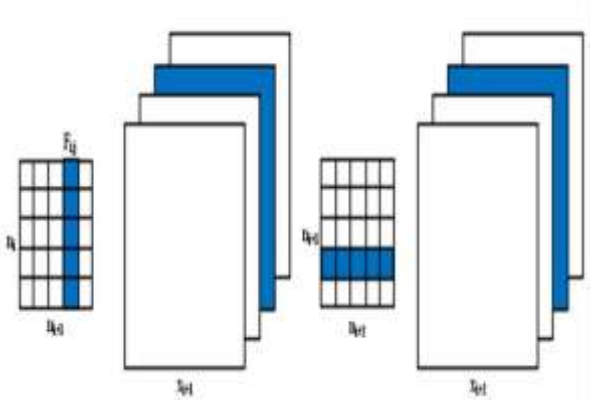


Fig.: Channel pruning principle

Principle of channel reduction. The network's i-th convolutional layer is identified by the letters xi; the input feature map's height and breadth are indicated by hi and wi, respectively; and the convolutional layer's number of input channels is indicated by ni.

CONVOLUTIONAL NEURAL NETWORKS-

brain systems with convolutions. These networks may sound like an odd amalgam of mathematics, biology, and computer science with a dash of CS, but they have been some of the most important advancements in the area of computer vision. According to Alex Krizhevsky, 2012 was the year that neural networks really started to gain traction.

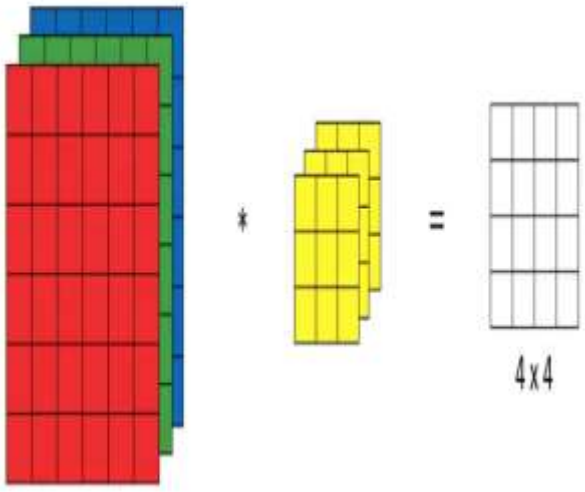
used them to win that year's ImageNet contest (basically, the yearly Olympics of computer vision), dropping the classification error record from 26% to 15%, which was a remarkable increase at the time. Since then, deep learning has become a key component of many businesses' services. For their automatic tagging algorithms, Facebook uses neural networks, while Google, Amazon, Pinterest, and Instagram are used for their search infrastructure and product suggestions, respectively.

Convolutions over Volume-

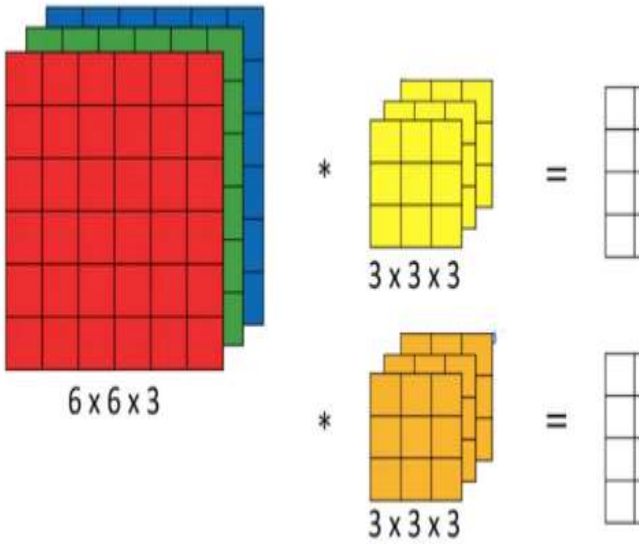
Suppose, instead of a 2-D image, we have a 3-D input image of shape 6 X 6 X 3. How will we apply convolution on this image? We will use a 3 X 3 X 3 filter instead of a 3 X 3 filter. Let's look at an example:

- **Input:** 6 X 6 X 3
- **Filter:** 3 X 3 X 3

The dimensions above represent the height, width and channels in the input and filter. ***Keep in mind that the number of channels in the input and filter should be same.*** This will result in an output of 4 X 4. Let's understand it visually:



The filter will logically have three channels since there are three channels in the input. The output form is a 4 X 4 matrix following convolution. As a result, the first element of the output is the total of the element-wise products of the 27 values from the filter and the first 27 values from the input (nine from each channel). After that, the complete image is convolved. We have the option of employing numerous filters as well as just one. How do we go about doing that? Let's assume that the first filter will look for vertical edges in the picture, and the second filter will look for horizontal edges. Multiple filters will alter the output dimension. Therefore, if we used two filters, we would have a 4 X 4 X 2 output rather than the 4 X 4 output shown in the case above:



Generalized dimensions can be given as:

- **Input:** n X n X n_c
- **Filter:** f X f X n_c
- **Padding:** p
- **Stride:** s
- **Output:** [(n+2p-f)/s+1] X [(n+2p-f)/s+1] X n_c'

Here, n_c is the number of channels in the input and filter, while n_c' is the number of filters.

One Layer of a Convolutional Network-

Once we get an output after convolving over the entire image using a filter, we add a bias term to those outputs and finally apply an activation function to generate activations. *This is one layer of a convolutional network*. Recall that the equation for one forward pass is given by:

In our case, input (6 X 6 X 3) is $a^{[0]}$and filters (3 X 3 X 3) are the weights $w^{[1]}$. These activations from layer 1 act as the input for layer 2, and so on. Clearly, the number of parameters in case of convolutional neural networks is independent of the size of the image. It essentially depends on the filter size. Suppose we have 10 filters, each of shape 3 X 3 X 3. What will be the number of parameters in that layer? Let's try to solve this:

- Number of parameters for each filter = 3*3*3 = 27
- There will be a bias term for each filter, so total parameters per filter = 28
- As there are 10 filters, the total parameters for that layer = 28*10 = 280

No matter how big the image is, the parameters only depend on the filter size. Awesome, isn't it? Let's have a look at the summary of notations for a convolution layer:

- $f^{[l]}$ = filter size
- $p^{[l]}$ = padding
- $s^{[l]}$ = stride
- $n_{[c]}^{[l]}$ = number of filters

Let's combine all the concepts we have learned so far and look at a convolutional network example.

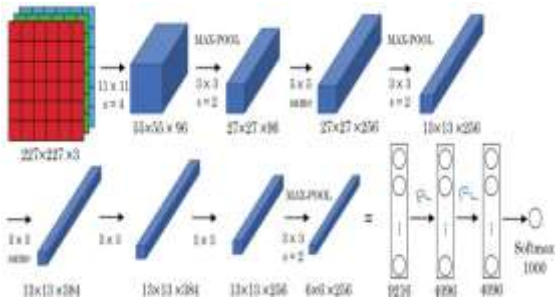
LeNet-5

It takes a grayscale image as input. Once we pass it through a combination of convolution and pooling layers, the output will be passed through fully connected layers and classified into corresponding classes. The total number of parameters in LeNet-5 are:

- **Parameters:** 60k
- **Layers flow:** Conv -> Pool -> Conv -> Pool -> FC -> FC -> Output
- **Activation functions:** Sigmoid/tanh and ReLu

AlexNet:

An illustrated summary of AlexNet isgiven below:

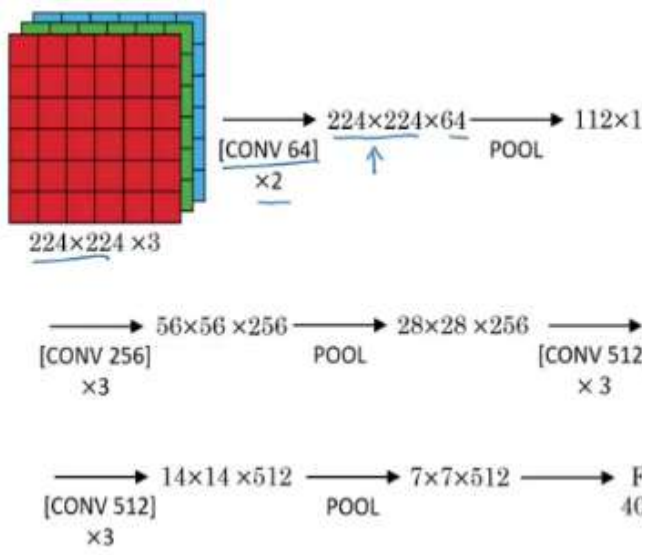


This network is similar to LeNet-5 with just more convolution and pooling layers:

- **Parameters:** 60 million
- **Activation function:** ReLu

VGG-16:

The underlying idea behind VGG-16 was to use a much simpler network where the focus is on having convolution layers that have 3 X 3 filters with a stride of 1 (and always using the same padding). The max pool layer is used after each convolution layer with a filter size of 2 and a stride of 2. Let's look at the architecture of VGG-16:



As it is a bigger network, the number of parameters are also more.

- **Parameters:** 138 million

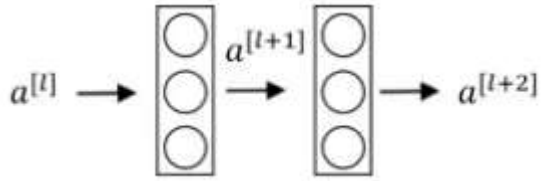
These are three classic architectures. Next, we'll look at more advanced architecture starting with ResNet.

ResNet

Training very deep networks can lead to problems like vanishing and exploding gradients. How do we deal with these issues? We can use skip connections where we take activations from one layer and feed it to another layer that is even more deeper in the network. There are residual blocks in ResNet which help in training deeper networks.

Residual Blocks:

The general flow to calculate activations[128] from different layers can be given as:



$$z^{[l+1]} = W^{[l+1]}\, a^{[l]} + b^{[l+1]}$$

$$a^{[l+1]} = g(z^{[l+1]})$$

$$z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$$

$$a^{[l+2]} = g(z^{[l+2]})$$

This is how we calculate the activations $a^{[l+2]}$ using the activations $a^{[l]}$ and then $a^{[l+1]}$. $a^{[l]}$ needs to go through all these steps to generate $a^{[l+2]}$:

In a residual network, we make a change in this path. We take the activations $a^{[l]}$ and pass them directly to the second layer:

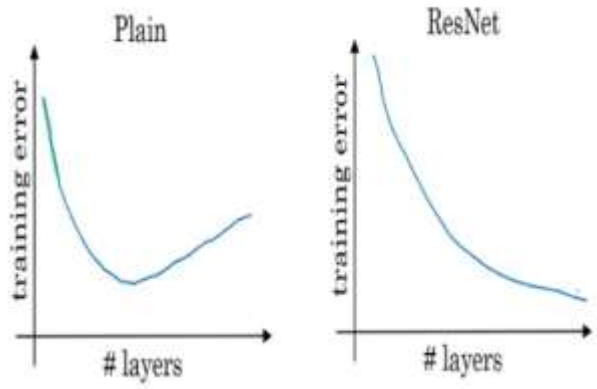
So, the activations $a^{[l+2]}$ will be:

$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$

The residual network can be shown as:

The benefit of training a residual network is that even if we train deeper networks, the training error does not increase. Whereas in case of a plain

network, the training error first decreases as we train a deeper network and then starts to rapidly increase:



We now have an overview of how ResNet works. But why does it perform so well? Let's find out

Fig: ResNet Architecture

SIMULATION RESULTS

YOLOv3-pruning algorithm perfomance:

The detection accuracy of the Yolov3-Pruning model based on transfer learning is as high as which is higher than that of the Yolov3-Pruning (unused) without transfer learning and is comparable to the baseline Yolov3 detection accuracy. From the visualization process of the three detection algorithms, baseline: Yolov3 and Yolov3-Pruning(transfer) is better than Yolov3-Pruning(unused), as shown in below

Fig: Visualization of the prediction process for baseline Yolov3 andYolov3-Pruning (transfer),Yolov3- Pruning(unused)

a) Baselne:YOLOv3

b)YOLOv3-pruning(transfer)

(c)YOLOv3-pruning(unused)

Recall and precision rate curve:

In below Figures show the recall and precision curves of Yolov3-Pruning and baseline Yolov3 on the voc2007 test dataset, respectively. It can be seen from the fgure that the recall rate of Yolov3-Pruning (transfer) is between the baseline Yolov3 and Yolov3-Pruning (unused), and it is very close to Yolov3 in multiple categories such as bottle, bus, and car; the degree is slightly higher than Yolov3

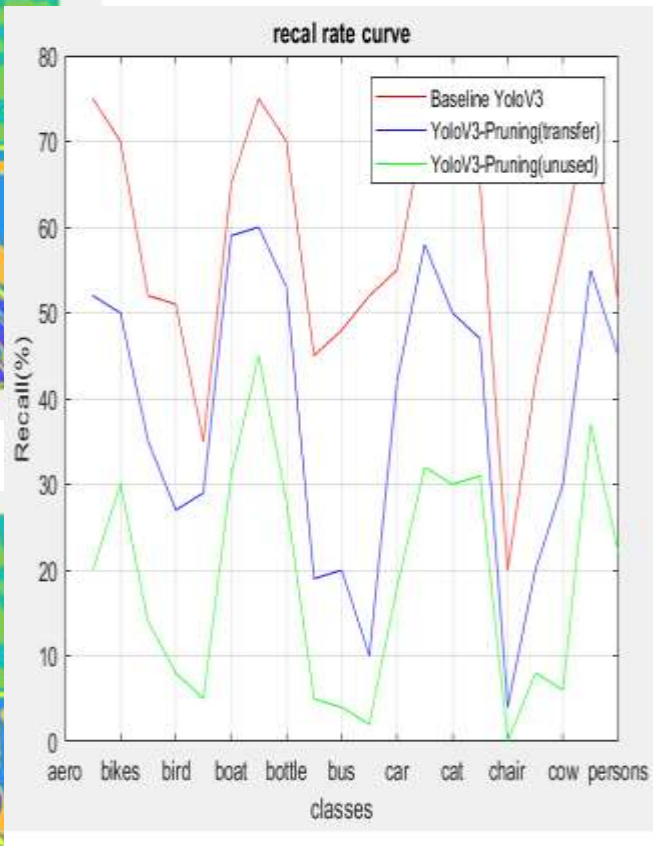


Fig: Recall rate curve

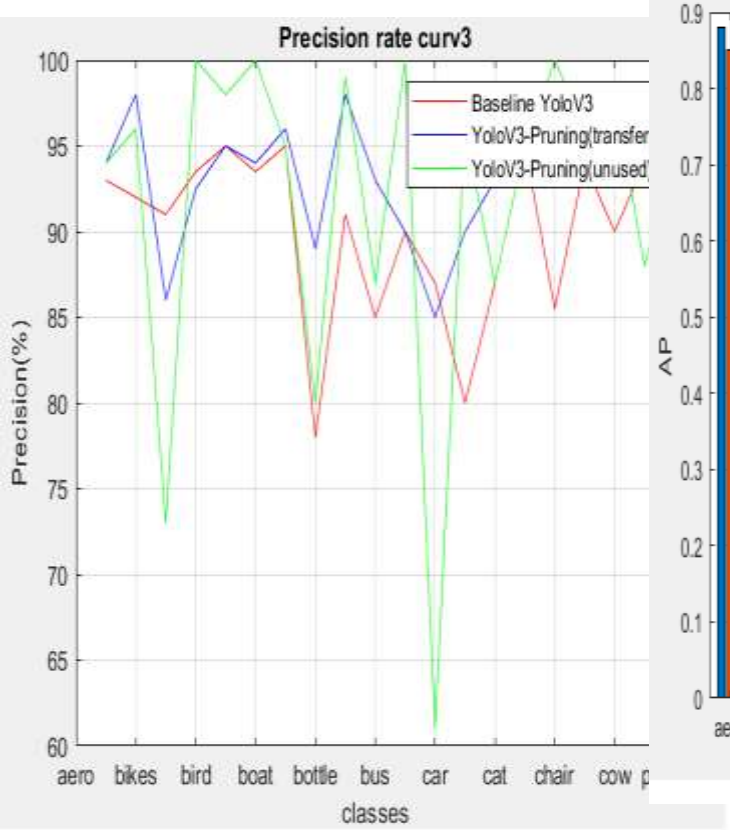


Fig: Precision rate curve

Yolov3-pruning algorithm performance:

In below Figure shows the AP values of Yolov3-Pruning and the baseline Yolov3 in 10 categories (in the coordinate system, the recall rate is the horizontal axis, the precision is the vertical axis to get the PR curve, and the enclosed area is the AP). The MAP in the fgure is the average value of various AP. It can be seen from the fgure that the performance of the object detection algorithm that uses transfer learning is much higher than that of the object detection algorithm

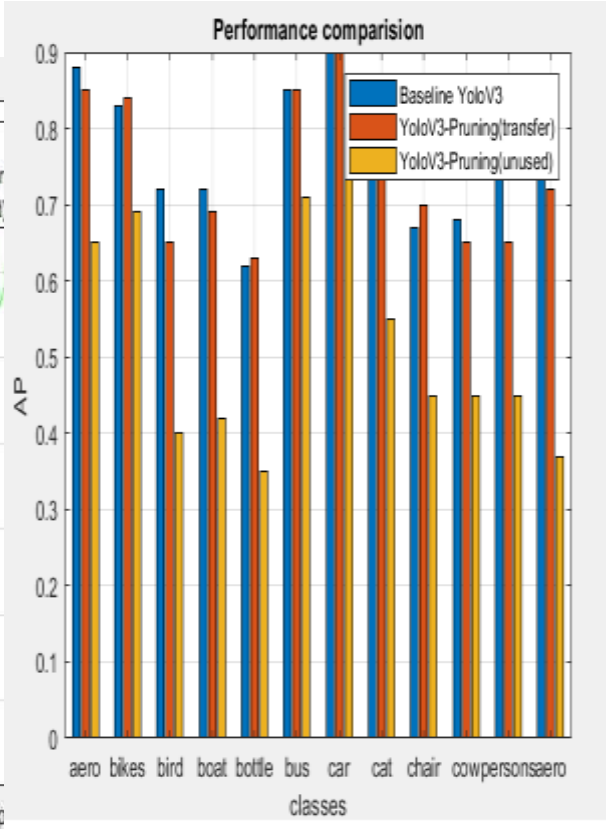


Fig: comparison of baseline yolov3 and yolov3-pruning (transfer), yolov3-pruining(unused)

Image real time processing comparison:

In below Figure compares the real-time performance of three object detection algorithms. FLOPS stands for foatingpoint operations per second. It can be seen from the fgure that baseline Yolov3 has the largest fops, indicating that it requires more data calculations and is more complex. Latency is the total time it takes for an image to be detected from the moment it is detected to the time it recognizes an object. The MAP of Yolov3-Pruning(transfer) is very close to the baseline Yolov3, which is much higher than that of Yolov3-Pruning(unused).

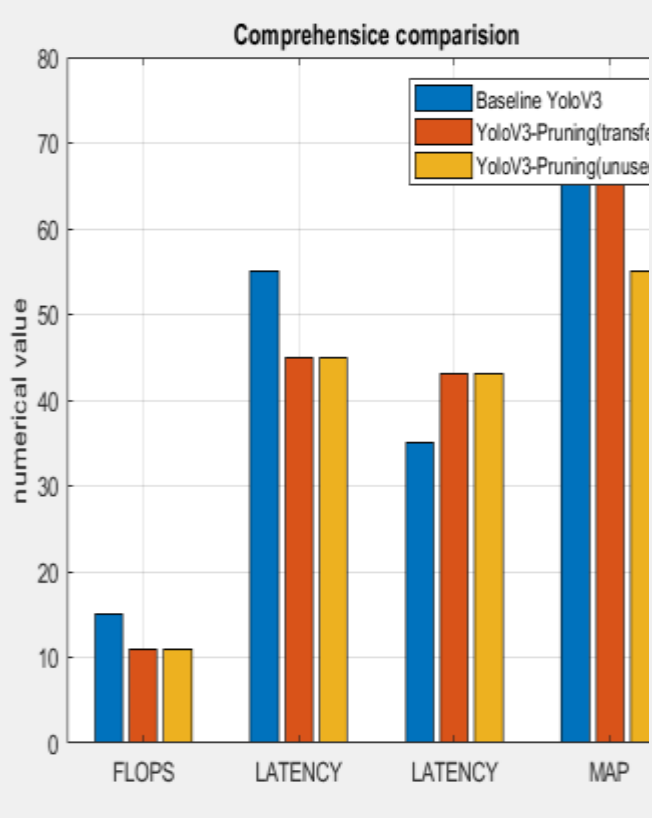


Fig: comprehensive comparison of algorithms in real -time image processing, including flops ,latency ,flops and map

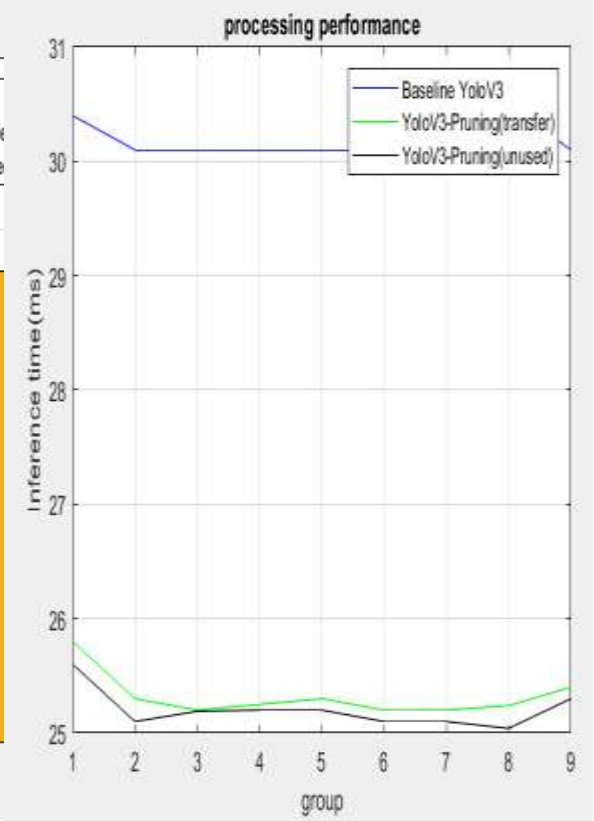


Fig: Comparison of real-time processing performance of baseline Yolov3, Yolov3-Pruning(transfer), and Yolov3-Pruning(unused) algorithms

Image real time processing comparison experiment:

In below Figure 5.6 shows the inference time of each object detection algorithm. When the model executes, the shorter the inference time it takes, the faster it infers, and the better the real-time performance

The comparitive experiment of different object detection algorithms:

Indicating that our model has a better target detection ability. It can be seen from the figures. The YOLOv3-pruning (transfer) can still maintain a high detection level after models basline-YOLOv3 and YOLOv3-pruning(unused). In the below figures shows a), original image of input data b)

converted noiseless image & figure shows the objects are detected by using YOLOv3-pruning(transfer) algorithm.

Fig: Input image

Fig: Convertred noiseless image

Fig: Objects are detected by using YOLOv3-pruning (transfer)

Conclusion-

This study implements the transfer learning-based object detection algorithm. Yolov3-Pruning is an upgrade to the original Yolov3. To reduce the size of the model, greatly speed up image detection inference, and support real-time image processing, the baseline Yolov3 structure is trimmed. The detection accuracy diminishes with decreasing model size. To solve this issue, we employ the transfer learning technique to guarantee the model's detection precision.

The YOLOv3-Pruning algorithm's network structure is simpler, easier to set up, and has fewer parameters than baseline Yolov3, which enables the object recognition algorithm to operate in real-time.

FUTURE SCOPE-

In the future, we will further study the algorithm to improve the performance of the model. We can choose several of the model compression algorithms and combine them. Improve the accuracy of the model from different aspects, analyse the performance of the algorithm from different perspectives to achieve better real-time processing. Based on reducing the size of the model, we will make it applicable to practical scenarios, such as remote sensing image detection, and target detection and tracking of UAVs.

REFERENCES

1. Hong, D., Gao, L., Yao, J., et al.: "Graph convolutional networks for hyperspectral image classification". IEEE Trans. Geosci. Remote Sens. 59(7), 5966–5978 (2020)

2. Chen, X., Yu, K.: "Hybridizing cuckoo search algorithm with biogeography based optimization for estimating photovoltaic model parameters". Sol. Energy 180, 192–206 (2019)

3. MIN QIAO , GANG ZHOU, QIU LING LIU, AND LI ZHANG,." Salient Object Detection: An Accurate and Efficient Method for Complex Shape Objects",. Date of current version December 31, 2021. Digital Object Identifier 10.1109/ACCESS.2021.3138782

4. Weiwei Fang , Feng Xue, Yi Ding , NaixueXiong , Senior Member, IEEE, and Victor C. M. Leung,"EdgeKE: An On-Demand Deep Learning IoT System for Cognitive Big Data on Industrial Edge Devices",. , IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 17, NO. 9, SEPTEMBER 2021

5. GIHA YOON, GEUN-YONG KIM, HARK YOO , SUNG CHANG KIM, AND RYANGSOO KIM," Implementing Practical DNN-Based Object Detection Offloading Decision for Maximizing Detection Performance of Mobile Edge Devices", Digital Object Identifier 10.1109/ACCESS.2021.3118731

6. Yuanyi Zhong∗1 ,Jianfeng Wang2 , Jian Peng1 , and Lei Zhang2 1University of Illinois at Urbana-Champaign,." Anchor Box Optimization for Object Detection",. IEEE Microsoft June 07,2020.

7. SajidJaved, ArifMahmood, Somaya Al-Maadeed, Thierry Bouwmans, and Soon Ki Jung,." Moving Object Detection in Complex Scene Using Spatiotemporal Structured-Sparse RPCA",.IEEE Transactions on Image Processing

8. YE Tao1,* , ZHAO Zongyang1 , ZHANG Jun1 , CHAI Xinghua2 , and ZHOU Fuqiang3,."Low-altitude small-sized object detection using lightweight feature-enhanced convolutional neural network", IEEE, Journal of Systems Engineering and Electronics Vol. 32, No. 4, August 2021, pp.841 – 853.

9. BaoGuanjun,CaiShibo,QiLiyong,XunYi,ZhangLibin,YangQinghua,." Multi-template matching algorithm for cucumber recognition in natural environment", .IEEE,journal 2016