February 2022
Master's Degree Thesis

Image Analysis based on AI application for Manufacturing Automation.

Graduate School of Chosun University

Department of Industrial Engineering

KIBET DUNCAN

Image Analysis based on AI application for Manufacturing Automation.

제조 자동화를 위한 **AI** 애플리케이션 기반 영상 분석

February 25, 2022

Graduate School of Chosun University

Department of Industrial Engineering

KIBET DUNCAN

Image Analysis based on AI application for Manufacturing Automation.

Advisor: Prof. GyuTai Kim

This thesis is submitted to Chosun University in partial fulfillment of the requirements for a Master's degree

October 2021

Graduate School of Chosun University

Department of Industrial Engineering

KIBET DUNCAN

This is to certify that the master's thesis of

Kibet Duncan,

Has been approved by the examining committee for the thesis requirement for the master's degree in Engineering.

Committee Chairperson

Prof. Jongho shin Signature:

Committee Member

Prof. GyuTai Kim Signature:

Committee Member

Prof. Seong-Joon Kim Signature:

December 2021

Graduate School of Chosun University

ACKNOWLEDGMENT

This thesis ventilates the outcome of my master's degree at the department of Industrial engineering, Chosun University. It has been a fruitful experience in the past two years, and I have had a great opportunity to gain much knowledge during this challenging period, still triumphant time. My work has been highly exciting to accord with, however the work itself was sometimes tough and demanding but due numerous people who aided me during this course it has been so inspiring and helpful.

Foremost, I would like to deliberate my out most gratitude to my advisors, Prof. Jongho Shin and Prof. GyuTai Kim for their incentive, guidance, and knowledge shared throughout the progress of this thesis. The exposure I have benefited during my research would contribute to my future endeavors.

I would also like to acknowledge and appreciate all my department professors, seniors, lab members, friends who have been of great help and family, who have withstand my partial absence but yet have been there for me.

Lastly, I would like to thank KISTI and IOsolution company collaborators for this thesis for their great input, and eagerness to share their data, and without their kind efforts would not have been possible.

Table of contents

List of figures

List of tables

Abstract

Abalone detection and counting using Yolov3 algorithm and image analysis on classification on lens using Grad-cam

Kibet Duncan
Advisor: Prof.
Department of Industrial Engineering
Graduate School of Chosun University

Abalone detection and counting systems via conveyor belts using machine learning have become a critical and most vital aspect in the fishing industry. In our paper, we present a detection and counting approach for Multi-Object Tracking (MOT) from abalone video data. Considering computational effectiveness and improved detection algorithms, Multiple Object and Tracking is one of the most sort out paradigms. You only look once (YOLOV3) with Tensor Flow algorithm built based on Deep SORT algorithm mainly implemented for MOT specification. Other methods used had low effectiveness with high operational time and wrong counting rate because some of the abalones stick to each other most of the time.

However, to elaborate the challenges posted by the video data of abalone entangling and pairing together on multiple occasions which is hard to count, we tested with the open source-based algorithm (OpenCV) which did detection but incorrect tracking and sorting. After combining Yolov3 and Tensor Flow for generating detections in each frame based on the model (Darknet-53), we use depth separable convolutions and pointwise group convolutions to reduce the parameter size of the network, the results of our experiment exhibited competitive performance in comparison with the open source-based library OpenCV. This proposed algorithm can apply in various conveyors counting controlled-based systems.

We also propose a technique for classifying normal and abnormal images from a broad class of Convolutional Neural Network (CNN) models, data classification more clear and explainable. We introduced Gradient-weighted Class Activation Mapping (Grad-CAM) that uses the gradients of any objective approach of normal or abnormal images fundamentally into the ultimate convolutional layer to yield a coarse localization map for highlighting the critical sectors in the image for predicting the approach. In contrast to previous approaches, Grad-CAM applies to a wide variety of CNN models: CNNs with fully connected layers (e.g. ResNet50 model) and CNN models used in line with various inputs (e.g. visualization) or reinforcement learning, without architectural transformation or re-training.

We incorporate Grad-CAM with current existing fine grained visualizations to generate a high-resolution class discriminative factor for visualization. Guided Grad-CAM is then applied for image classification, captioning based on pre-trained ResNet50 architectures. For image captioning, our visualizations show that even ResNet50 based models learn to localize discriminative regions of the input image. We develop identification of essential neurons via Grad-CAM and incorporate it with neuron names to provide textual explanations for model decisions. Finally, the used test data to establish appropriate trust in predictions from the model and show that Grad-CAM can give accurate predictions from deep networks and show that Grad-

CAM helps to make a more accurate prediction even close to identical images either normal or abnormal.

Keywords— Abalone detection and counting; Machine learning; Yolo-V3; TensorFlow; OpenCV; Darknet-53,Normal and Abnormal classification, Grad-CAM, ResNet50 model, Convolutional Neural Network (CNN)

1. Introduction

Abalone is a single-shelled gastropod mollusk marine snail notable not only for resembling the human ear but also one of the most valuable shellfish in the world. Its high value comes from the good tasty and beautiful shell [1]. Abalone is a high-protein low-fat food rich in minerals and vitamins. Since abalone feeds on ocean superfoods such as sea mustard and kelp, it is nutritionally almost the perfect seafood. Abalone lives in rocky natures in the ocean where it can feed on algae. Abalones' tough and strong shells creates an immensely strong attachment to the rocks in the ocean, which requires time and skill during fishing [2]. Due to the complexity of its capturing in nature, about 80% of Korean abalones are cultivated in the southern sea, where the water temperature remains the same throughout the year. The south sea has a Rias coast, so it has the optimal environment for a variety of marine life. The wide seaweed forests in these coastal waters provide an abundance of food for the abalone, while the mudflats help to purify their environment. Abalone is highly valued and sold expensive in Korean market. The culturing of abalone is composed of two steps (the first/second nursery). In the beginning, abalone are feed in indoor tanks as the first nursery and, then, the secondary nursery is carried out in the sea. The raised abalone after the first nursery that is reached about 1.5 cm or more are counted to be sold in order to be grown-out at the sea. Usually, the counting of abalone is done by human, which causes high cost. To solve this problem, some machine to count abalone juveniles and to pack it have been developed. The used algorithm for counting abalone juveniles is based on pixel sizes considering average pixel size of abalone juveniles. The problem of this method is that it cannot separate the overlapped and entangled abalone [3]. This paper proposes an application of machine learning technology based on CNN in order to count abalone in real-time.

1.1 State-of-the-art

Abalone counting based systems in small fisheries or big industrial sites predominantly uses conveyor belts. Which poses challenges such as not being able to count because abalone overlap or entangle. More so, if a camera placed in a fixed position on top of the conveyor belt to acquire video data or images then data processing performed to count the abalone in real-time can be significant for this process. A single camera with a wide range of view to oversee multiple lines in the conveyor belts.

Object detection, tracking and sorting algorithm applies in many fields such as intelligent transportation system (ITS) [4]; traffic control systems [5], pedestrian detection [6], Automated CCTV surveillance [7] and in recently emerging autonomous vehicles [8]. After abalone detection, we considered a target region of interest (ROI) [9] in the conveyor belt where we can track and count each abalone. By removing the calculation of non-interest regions, we improved the performance of YOLOV3.

Subsequently, Object tracking is inevitable to reduce the risks of abalone collisions. In contrary to tradition tracking approaches, which enforce target initialization, our undertaking uses the information from the abalone detection and distance estimation to initialize counting and to track them later on. We present an extended Kalman filter for 3D tracking and sorting abalone [10]. We first applied OpenCV library due to wide spectrum of applications. However, we found some tweaks and challenges in tracking of abalone because of low Frames per second (FPS) and accuracy in comparison to YOLOV3 which was more applicable.

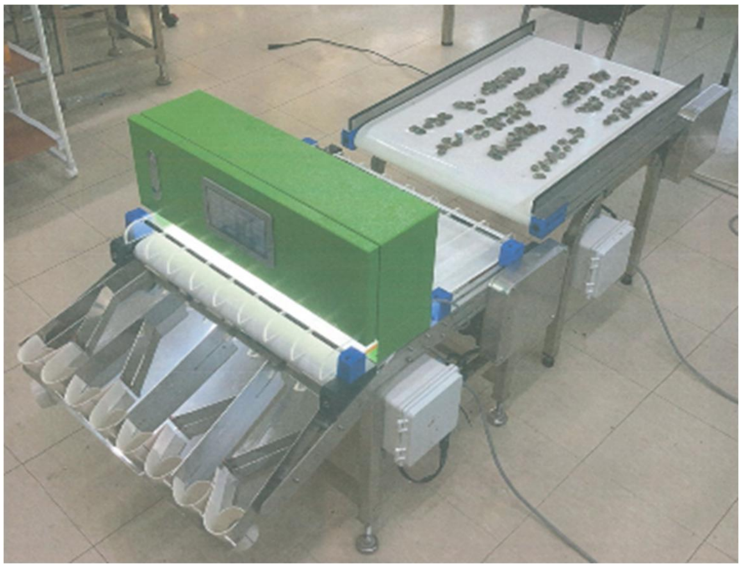
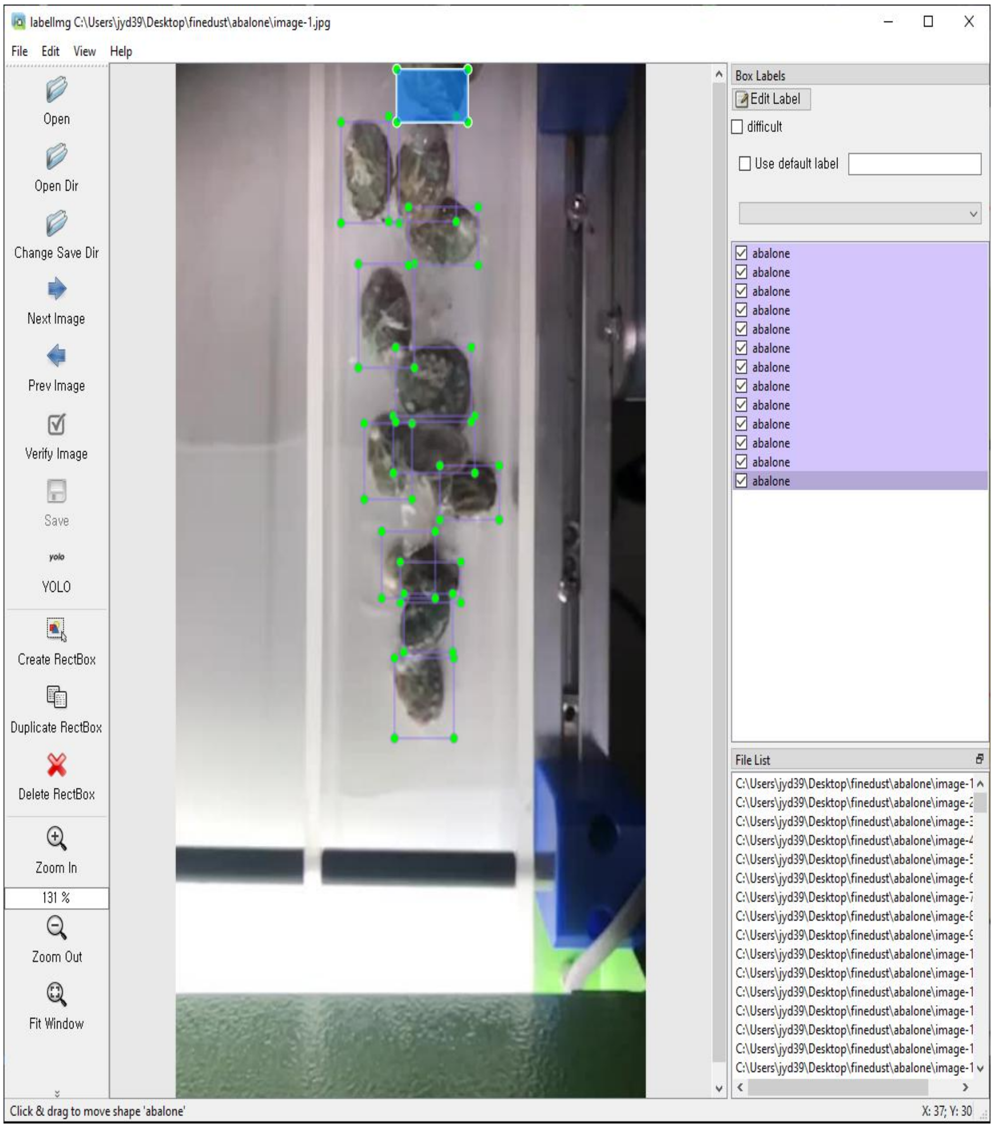


Figure 1-1. Conveyor belt for counting Abalone

1.2 Image processing

We implemented a decoding method to extract 1105 frames as images contained in our original video data by first installing FFmpeg on python environment then we used FFmpeg command framework [11]. The video sequence had a duration of 15 seconds with 30 fps which provided frames. However, due to blurriness of abalone in some of the frames, we discarded around 25% finally gathering a sum of valid frames. The task of annotating the images for the training a machine learning/deep learning model falls under the data preparation stage. An open source tool called labelImg is a tool that can be used for image annotation. This task had to be done manually to all the fames containing abalone which was is written in python and uses QT for graphical interface. Annotation which consist details about the objects in the image such as Image and label name, Image path, class of the image, co-ordinates of bounding boxes surrounding the abalones present in the image are saved as XMLfiles in PASCAL VOC format, used by ImageNet. Furthermore, it also supports YOLO and createML formats [12]. We saved the annotated images with text files in YOLO format (.jpg image- file) for which contains values such as (object-class) (x_center) (y_center) (width) and (height). The class object shows the abalone position using integer numbers.

Yolo format calculates annotation values using <center-x = x/W>, <center-y = y/H>, <width = w/W>, <height = h/>. In which the x-coordinate is the center of the bounding box, y-coordinate is also the center of the bounding box and H and W is the height and width of the entire image respectively.



labelImg C:\Users\jyd39\Desktop\finedust\abalone\image-1.jpg

abalone
abalone
abalone
abalone
abalone
abalone
abalone
abalone
abalone
abalone
abalone
abalone
abalone

C:\Users\jyd39\Desktop\finedust\abalone\image-1
C:\Users\jyd39\Desktop\finedust\abalone\image-2
C:\Users\jyd39\Desktop\finedust\abalone\image-3
C:\Users\jyd39\Desktop\finedust\abalone\image-4
C:\Users\jyd39\Desktop\finedust\abalone\image-5
C:\Users\jyd39\Desktop\finedust\abalone\image-6
C:\Users\jyd39\Desktop\finedust\abalone\image-7
C:\Users\jyd39\Desktop\finedust\abalone\image-8
C:\Users\jyd39\Desktop\finedust\abalone\image-9
C:\Users\jyd39\Desktop\finedust\abalone\image-1
C:\Users\jyd39\Desktop\finedust\abalone\image-1
C:\Users\jyd39\Desktop\finedust\abalone\image-1
C:\Users\jyd39\Desktop\finedust\abalone\image-1
C:\Users\jyd39\Desktop\finedust\abalone\image-1
C:\Users\jyd39\Desktop\finedust\abalone\image-1
C:\Users\jyd39\Desktop\finedust\abalone\image-1

Click & drag to move shape 'abalone'

X: 37; Y: 30

Figure 1-2. LabelImg annotating tool

Table 1 Saved yolo format txt file of annotated abalone (sample)

Abalone position	Height	width	x	y
15	0.504950	0.729167	0.128713	0.030556
15	0.512376	0.683333	0.123762	0.058333
15	0.538366	0.639583	0.111386	0.029167
15	0.486386	0.625000	0.106436	0.047222
15	0.632426	0.552083	0.111386	0.037500
15	0.545792	0.515972	0.136139	0.045833
15	0.454208	0.511806	0.121287	0.070833
15	0.454208	0.454167	0.163366	0.063889
15	0.440594	0.382639	0.113861	0.084722
15	0.574257	0.317361	0.163366	0.051389
15	0.527228	0.256944	0.118812	0.063889
15	0.413366	0.259028	0.103960	0.093056
15	0.537129	0.174306	0.143564	0.073611
15	0.617574	0.136111	0.126238	0.091667
15	0.495050	0.081944	0.103960	0.094444

In the first raw of figure1-3: 15 represents the value of object class abalone, 0.504950 as height, 0.729167 as height, 0.128713 as x-coordinate (center value) and (0.030556) as y-coordinate.

1.3 Contributions

The contribution of this work is to introduce abalone detection and counting algorithm with Yolov3 frame work with object tracking and deep sort. Object tracking is the process of tracking single and grouped moving or motionless objects [13]. Over the year's detection has improved because of Neural Network which uses different algorithms to distinguish patterns. Deep learning provides advanced performing which models that more data but less process to train and validate. Image classification algorithms depicts bounding boxes that represent locations of objects. R-CNN presented by Ross Girshick (regions with CNN) showing different number of regions to get 2000 regions [14]. Deep SORT introduces another association of apparent feature information in that the appearance of new detections is compared to previous tracked objects in individual tracks to aid in data association problem [15].

The obtained outcome is supportive and reliable towards real-time detection and counting of abalone.

1.4 Thesis Layout

This thesis is composed of seven chronological chapters. Following the introduction, Chapter 2 presents theory including background scoop and illustration of some of the effective stipulations for a better understanding of this work. Chapter 3 present the details of machine learning and evaluation technique in line with research. Chapter 4 provides YOLOV3 and sort for object detection and tracking abalone applying to the first section of our work. The included methods and their results of YOLOV3 accuracies are state of the art. Chapter 5 explains our methodology of Grad-CAM, particularly in the classification Normal and abnormal images. Chapter 6 we discussed about the finding of our result and chapter 7 is the conclusion the proposed working of this research.

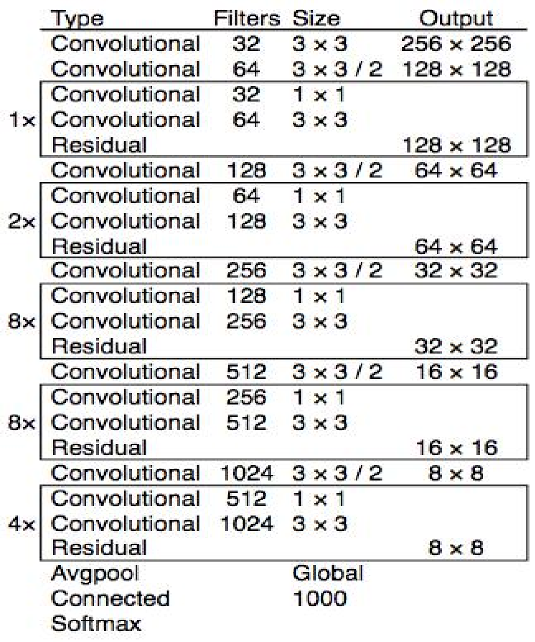
2. Theory and Background

2.1 YOLOv3 Object detector and its working

Joseph Chet Redmon created the YOLO object detection algorithm. The underlying hypothesis of how YOLO object detection can detect an object is mentioned in [16]. YOLOV3 is the third version of the original YOLO real-time object detection model. It starts predicting the same way as the YOLOV2 model, using anchor boxes that have been created by grouping the ground-truth boxes with a K-means clustering algorithm and taking the box by the IoU between all the boxes of each group [17]. Each bounding box is predicted by 4 coordinates and a confidence score that are calculated. In addition, in Figure 2-1 it can be observed how adjusting anchor boxes, a predicted bounding box is obtained. To explain the new network advances compared to previous versions, we will go from the most basic to the most complex:

2.1.1 Architecture (Feature Extractor Network)

Developers combined YOLOV2 with a Darknet-19 custom network and, in this new version, it has been replaced by a Darknet-53 custom network [18]. It is formed by 53 convolutional layers that are responsible for extracting the features along with an average pooling layer before the fully connected layer (shown in Figure 2-1). The features are extracted in the convolution layers composed by filters or kernels of dimension 3 x 3 and 1 x 1. 3 x 3 filters are used for the feature extraction, but 1 x 1 filters are mainly used to reduce the depth of the output of some convolutions. For example, in Figure 29 the first two convolutional layers are formed by a set of 3 x 3 filters, then the third convolution layer takes as input value the feature map that is the output of the second one and by applying the 1 x 1 filter it reduces the depth of the feature map. The 1 x 1 kernel converts the feature map of a convolutional layer of dimension 7 x 7 x 1024 into 7 x 7 x 125, in short, it converts an output of 1024 channels into 125. This channel reductions are very necessary to improve the speed of prediction because the YOLO (V1, V2, and V3) model is aimed at real-time detection.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2 -1. Example of Darknet-53 network architecture

The architecture was also improved by adding residual networks that are built of Shortcut Connections that are necessary for the gradient descent. In very deep neural networks formed by large layers (the case of YOLOV3), the gradient descent faces problems to propagate backwards the error in the training process. This problem is solved by adding Shortcut Connections that create "short cuts" as the name implies, for gradients to propagate backwards faster and further. In this way, they allow an efficient training for the YOLOV3 model because the first layers get the weights updated much faster than before.

2.1.2 Bounding box prediction

For each bounding box predicted it calculates object or confidence score (assurance the model has that the specific bounding box contains an object) using logistic regression [19]. This computation is done by calculating the overlap between a ground truth object and the bounding box, if the overlap is the highest one for all the bounding boxes observed before, then the score will be 1. In the other case, If the overlap of a bounding box with respect to a ground-truth box is not the best, but it is higher than a threshold = 0.5 the prediction is ignored [20]. The bounding boxes with a score of 1 totally incur in the loss function, however, the rest of the bounding boxes, do not incur to the cost function.

In Figure 2-2, we can see that the prior 1 is the bounding box that most overlaps the ground truth box for object 2, this means that has the highest IoU with respect to the rest prior boxes in the image (prior 2 and 3) [21]. Prior 3 is the predicted box that most overlaps the ground truth box for object 1 with respect to the rest prior boxes in Figure 2-2. The model associates to each ground truth box object to one prior box, so if a prior bounding box has not been associated to any ground truth object it will not affect the loss function for the localization and classification loss, only on the confidence loss. Referring to our example in Figure 2-2, prior 1 and 3 will incur in the loss function because their overlap is the highest one and they are scored with 1, however, prior 2 box is not associated to any ground truth box and will only incur to the confidence loss inside the loss function.

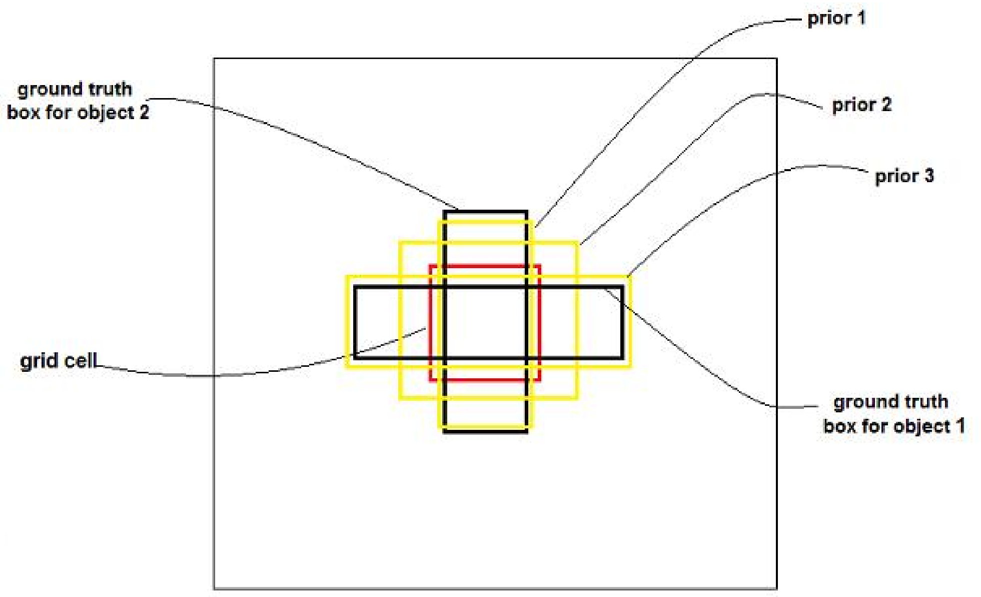


Figure 2-2. Example of prediction for a grid cell of a YOLOV3 model

2.1.3 Feature Pyramid network (FPN)

YOLOV3 makes three predictions or predicts three bounding boxes per grid cell, each prediction is composed of the coordinates of the location of the object, the confidence score that the box contains an object and 80 class scores (i.e. S x S x [3 x (4 + 1 + 80)] predictions in total). YOLOv3 makes predictions at three different scales by extracting features from those scales like in FPN (shown in Figure 2-3 how the prediction process looks like) [22]:

- First prediction in the feature map of the last convolutional layer.

• The second prediction is done by going back two layers and augments the first predicted feature map by 2, then takes the feature map that has higher resolution and merging it with the augmented feature. Apply some filters to the resulting feature map.

• Repeat the previous step until the resulting feature map has high-level structure, for example, the localization of the objects is quite clear, the object resolution is high, etc.

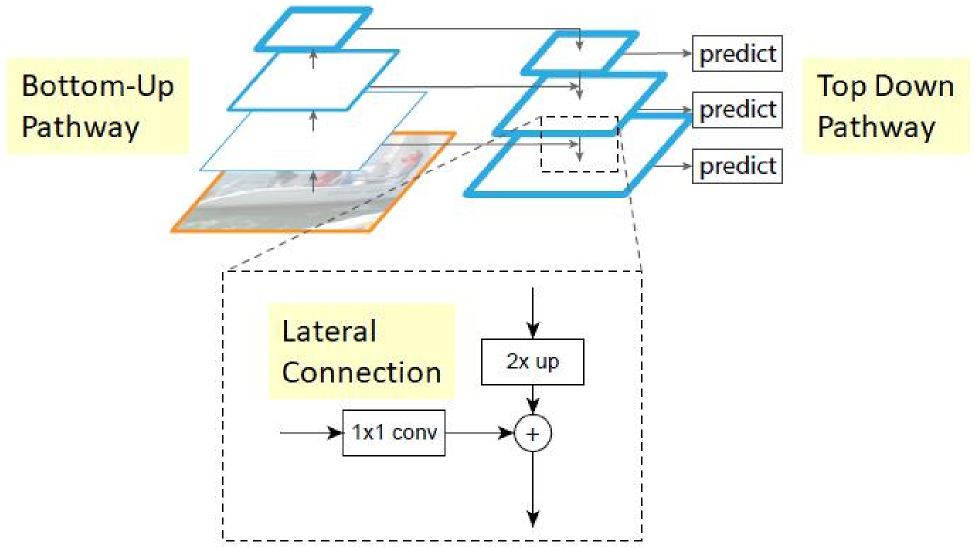


Figure 2-3. Example of predictions in an FPN

2.1.4 Class prediction

Prior in YOLO versions, researchers used soft-max method in the class values and take the resulting class with highest score to belong to the class of the object in the bounding box for which this has modified in YOLO v3 version.

Soft-maxing classes' depends on the hypothesis that classes are mutually absolute or if an object belong to a single class, then it cannot belong to another class which fine in COCO datasets [23].

However, when if we have classes like **oyster** and **abalone** in a dataset, then the above hypothesis is not applicable. This is the main reason why the research of YOLO have ceased from soft-maxing the classes. Alternatively in YOLOV3, logistic regression is used for predicting each class scores and the threshold is used to predict numerous labels for specific object. Resulting classes showing maximum score than the threshold, are assigned a bounding box.

3. Machine Learning

Machine learning is one of the prominent branch of artificial intelligence which consist algorithm design for example according to our research object detection and image classification through experience. Machine learning algorithm learns to produce intelligent ways to make decisions fully based on the capability to detect or classify, which can be relevant to recognize stock market analysis, speech recognition, handwritten pattern, image analysis or medical diagnosis. The main focus of this thesis is object detection, counting and image classification. The future prospect is that YOLOV3 can be applied to count and track abalone accurately in real-time, same to the second section in classification of normal and abnormal for prediction.

In this chapter we show the essential overview of machine learning technology that are significant with this thesis and suitable for abalone counting. Chapter 3 firstly present the specifics deep sort of technique, followed by description of estimation model of Yolov3 algorithm along with their mechanisms that explains the performance of our model.

3.1 Deep SORT

SORT (Simple Online Real-time Tracking) is an implementation of a tracking-by-detection framework that is oriented to multiple object tracking (MOT) problems in videos. This tracker by the help of an object detection model, detects objects in each frame and represents them by bounding boxes [24]. Once the detection of the objects has been done by the model, to track the objects of the current frame, the tracker uses the objects detected from the previous frame along with the current frame. The detection quality has a big impact on the real time tracking performance and that is why SORT must be very well combined with the object detection model. During this chapter the different techniques that SORT uses for real-time tracking will be explained like Kalman Filter and the Hungarian algorithm focused on the tracking components.

3.1.1 Estimation Model

SORT uses as a representation model a motion model to propagate a target into the next frame. For each object detected, it is associated to a "Target" (like the bounding box of YOLO) that stores all the necessary information about the state of the object in that particular frame. The state of each target is represented by 7 variables x = [u, v, s, r, u', v', s'] where (u, v) is the horizontal and vertical pixel location of the center of the detected object (pixel-center), the s represents the area of the target and r is the ratio of the bounding box. The coordinates (u', v', s') are the respective velocities of the center coordinates and the scale (area) of the target. Targets travel through the different frames of the video and are an essential characteristic to track objects because when the model detects an object in a frame, SORT looks back to all the previous targets and can associate a previous target to the detected object or can create a new target.

When the model makes a detection in one frame, if it is associated to a previous target, the detected bounding box coordinates are used to update the state of the target, the target that comes from the previous frame is predicted in the current frame by the bounding box of the predicted object and the velocity components are predicted by a Kalman Filter. If the detection is not associated to any previous target, its new target state is predicted by the linear velocity model that is independent from other objects. When previous frames targets have not been associated to any detection for a long time, they are deleted because this means that the objects have left the video. Also, for the first few frames, it may happen that duplicate targets are created, to prevent this, a threshold value is set, and lower targets are deleted.

At the start, the video is divided into frames and is analyzed frame per frame. In the first frames, new objects without any target are detected because they have not appeared yet, so the SORT tracker creates a target for each detected object and the state is predicted by the linear velocity model with the values of the position of the object (u, v, s, r) and their respective velocities (u', v', s'). After a few frames have been analyzed, when predicting objects in new frames, some of this objects will be associated to a previous target because they are the same object than in the previous frame and some other objects appear for the first time so again a new target will be created for them.

3.1.2 Data assocoation

To assign a detected object to a previous target, SORT predicts its new location in the frame by an estimated bounding box geometry. The detection of the object and the prediction of the previous target are independent from each other, this means that they need a metric to associate both. The IoU distance between each detection and all the predicted bounding boxes from the targets is the metric that SORT uses to quantify the association. The assignment cost matrix is another metric responsible for associating the data and it is calculated by applying the IoU (metric to quantify the association) mentioned before. Once this matrix has been computed, the Hungarian algorithm is used to solve the assignment between the predicted object and one of the previous targets.

3.1.3 How the Hungarian algorithm and Kalman filter work

The Hungarian algorithm takes as input value a cost matrix C and computes C^1 as follows [25]:

1. For each row it computes the minimum value and subtracts it to all the elements of the row.

$$C^1_{i,j} = C_{i,j} - \min C_{i,j}$$

2. For each column it computes the minimum value and subtracts it to all the elements of the column (To the matrix computed in the previous step).

$$C^1_{i,j} = C^1_{i,j} - \min C_{i,j}$$

3. If all the rows of C^1 exist a column with a cost of 0 and has not been associated to any other row, the algorithm finishes, if not it repeats step 1 and 2 again.

The Hungarian algorithm is mainly used for assignment problems, for the SORT tracking algorithm it is very useful in the data association process because it resolves the assignment cost matrix using the process described above and gives a solution to the problem.

The Kalman filter algorithm is a recursive algorithm that estimates the motion state of the target. Given an 8-dimensional state space (x,y,r,h,x′,y′,r″,h′), Where(x,y) represents the center coordinates of the detection frame, r represents the aspect ratio of the detection frame, h represents the height of the detection frame, and (x′,y′,r′,h′) is their velocity in the coordinate system [26]. Assuming that the motion of the target is uniform, using a standard Kalman filter, as well as a linear observation model, the position of the prediction frame for the next frame can be obtained (x, y, r, h). The output for predicted bounding box is particularly unlikely to be as accurate primary bounding box in reality. Therefore, we measured how accurate are the abalone in the frames by using (IOU) Intersection Over Union metric. This was a simple way to evaluate our training model and bounding boxes with its performance on testing set. Consequently, result in the previous frame, the current position of objects in the next frame is predicted by Kalman filter. Figure 3-1 show in what step is the Kalman Filter applied in a multi object tracking.

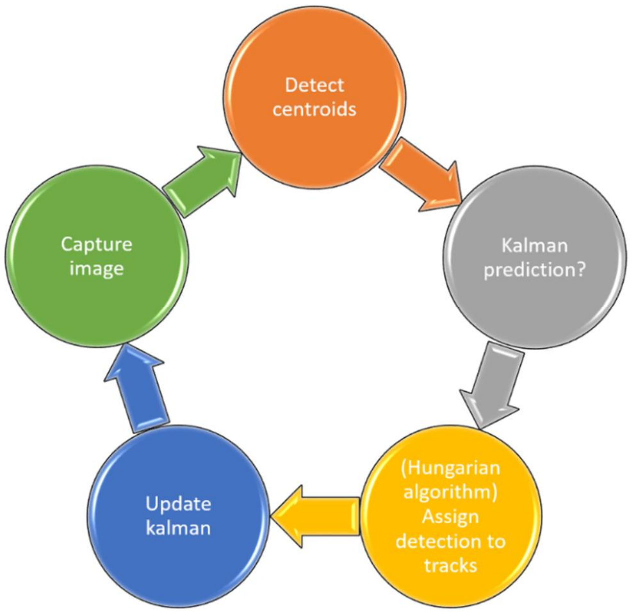


Figure 3-1. Steps of multi-object detection

3.1.4 Creation and deletion of track identities

When an object first appears in a video, this means that when the model is analyzing the video frame per frame and detects for the first time a particular object, a unique identity has to be created for that object, which will be called a tracker. The track is initialized with the coordinates of the predicted bounding box and the velocity variables of the tracker are set to zero. When the tracker of a new predicted object is created the velocity cannot be computed and therefore the covariance of the velocity is initialized with large values. After the new tracker has been created in a frame, it is necessary that in the following frames it is associated to other predicted objects in order to ensure that it has been created correctly and to prevent false positives. The tracker is deleted when it has not been associated to any detection in a threshold of frames since it was created. This threshold has been set to 1 (no_associated > 1) because if in one frame a tracker has been created and associated to a predicted object, then the next frame is studied. If in the next frame the tracker has not been associated to any object it will not be deleted yet as it may be a tracker bug, so the model will analyze the following frame and if there is also no object associated to the tracker then it will be removed. In the Figure 3.1.4 it is shown how a tracker tracks an object frame per frame.

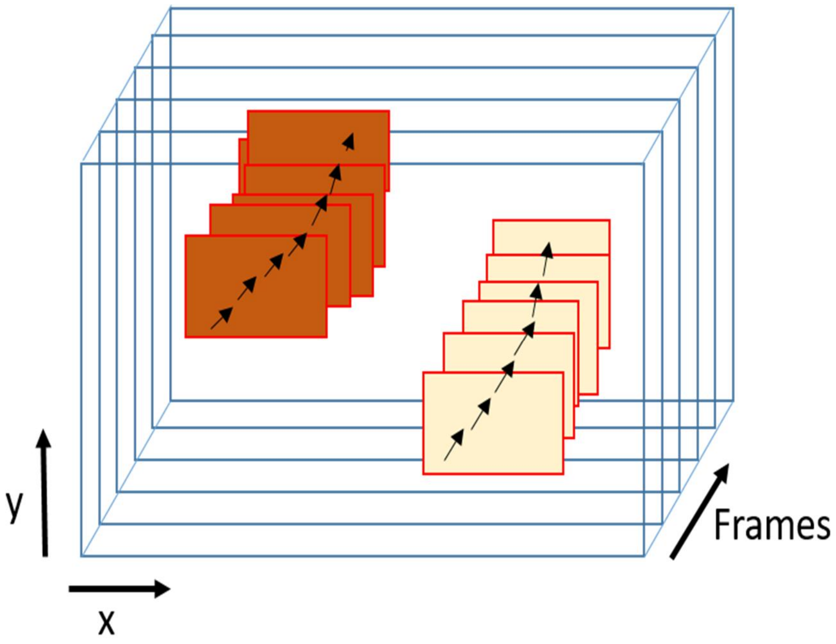


Figure 3.1.4. Example of a tracking in a video

4. Yolov3 and Sort for object detection and tracking

The aim of this project is to develop an application fast and accurate enough to accomplish the objective of tracking and detecting vehicles abalone. YOLOV3 is going to be the object detection model that is going to be trained and optimized to detect our custom objects (abalone) and the SORT algorithm will be responsible of monitoring frame per frame the abalone detected. The application that is developed in this end-of-degree work is specially designed to improve the detection of abalone occlusion and for improving the management of counting accurately. The application could be used in video vigilance cameras located in no pedestrian streets; our model will count in a video sequence how many different cars have passed. If we compare the result obtained to the average number of cars that normally pass on that street at that time, we can find out whether there is a lot of traffic or not so that we can notify a user to avoid that street. Another possible application for the model described in this work could be to fit our model in the camera of an autonomous vehicle in order to follow a predetermined vehicle that will go directly in front. By this way, our application will be capable all times to recognize the predetermined vehicle in the pursuer car's camera and thus be able to perform the same route as the vehicle in front does. In the following Chapters will be explained what steps have been followed to develop the vehicle detection and tracking model.

4.1 Train and evaluation Yolov3 to detect Abalone

The first step is to train YOLOV3 model using the dataset that has been explained before, the model will be trained with different parameters and after all the best configuration will be chosen to make the detection process. Before we start talking about the training process we have followed for the network, we must first explain what tools and frameworks have been used for this part of the work.

4.1.1 Tools and frameworks:

Training a CNN model is very hard in terms of time and complexity, if the model is trained using a CPU it may last days and even weeks. To avoid so, if the model is executed and trained supporting GPU the runtime is reduced severely and can run trainings in few hours depending on the size of the dataset. The computer that has been used during this process does not have a GPU that could support all the process of a Deep Learning model and therefore we have been forced to look for hosts on the internet that offer to run our models in powerful GPU.

• Google Collab: It is a free service where Python notebooks and Linux commands can be executed and it offers a free GPU of 12 GB that will be the one used for training, evaluating and testing our model. It has many Python libraries installed and if we need more libraries, it is possible to install them as in a Linux platform by writing installation commands in the shell. Also, it is possible to clone GitHub repositories and store them on the server giving you access to them from the platform, keep in mind that Google deletes all the stored data every 12 hours.

YOLOV3 can be implemented with many libraries that Python supports such as Keras, PyTorch (we will talk about it in the next Chapter), TensorFlow, etc., or also it can be implemented in the programming language C++. For this part of the work we will use Darknet, it is an open source neural network framework that has been written in C and CUDA and the most interesting part is that it supports GPU computation. To implement our model, we have based ourselves on the Darknet framework (https://github.com/AlexeyAB/darknet/) [27] but making numerous changes to the code in order to adapt the model to our dataset and thus be able to carry out a good training and a following evaluation. Darknet must be built to support GPU (default off), CUDNN (default off) and OpenCV (default off) in order to be run on the GPU of Google Collab.

4.2 Training and evaluaton

As we stated in image preprocessing, we image labeling tool (labelImg) to get annotated abalone annotated images with text files in Yolo format. We compiled our image and text file to configure for training. This step involves properly configuring your custom .cfg, obj.data, obj.names, train.txt and test.txt files [28]. It is important to configure all these files with extreme caution as typos or small errors can cause major problems with your custom training.

Configuration file consist of the parameters and values for training so we set our batch size to 64 and subdivision to 16 for ultimate results. Maximum batches was set to 6000, resulted by the number of classes multiply by the original 2000 but not less than 6000 training with one class. For the steps, we stet to 4800, 5400 resulted by 80% and 90% of the maximum batches consecutively and the filters to 18 because we added the number of classes to 5 and multiplies to 3 with height and width of 416.

Table 2. Values for the YOLO network parameters

Parameter	Value
Batch size	64
Subdivision = sub batches: number of mini batches is split	16
Height and width of input images	416
Max_ batches	6000
steps	4800, 5400
filters	18

Training our abalone dataset with configuration we set in the above table, YOLOV3 on the darknet framework saves the weights files. In the figure below we can see the training graph shows the average loss computes for every iteration. Note that the dataset is quite complex as it has tangled abalone in each images that are not visually easy to see, this complexity affects the object detector by obtaining a relatively low one. This happens because the model fails to detect some abalone in the validation set that it is difficult even for the human eye to detect them.

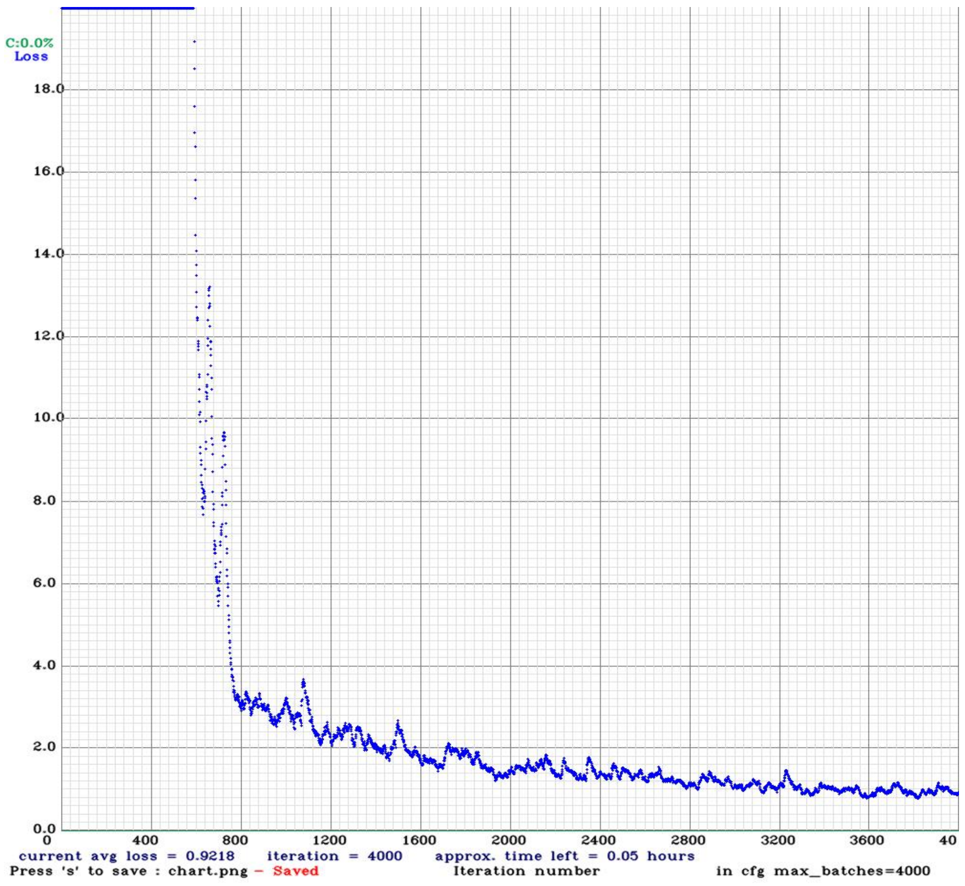


Figure 4-2. : Graph computed during the training

Weight files obtained in the Chapter of training YOLOV3, the network configuration file and the class names file in order to make the detection process. When processing the video we have made some changes: reduce the video resolution, change the frame-by-frame analysis, reduce the video shape in order to focus the analysis on the interest point, add the confidence threshold variable, add the OpenCV library (processes videos at 30 frames/sec) and implement functions to load videos and to store them in Google Collab, implement functions to show the analyzed frames in the command line, etc.

In the following pages we will discuss the different changes and tests that have been made to the parameters of the network. For example, we will see every how many frames the video must be analyzed in order to track the vehicles preserving a fast running time. Also, we will play with the video resolution as our model is able to detect abalone very well in low resolution videos and in this way the speed of detection and tracking is improved. Finally, we will focus the detection and tracking process to a specific area of the videos where the abalone pass leaving aside areas of little interest for our work.

4.3 Tests on the detection and tracking

The model will be analyzed the test video to detect and to make the tracking. The tracking must be almost perfect since it has to count the number of abalone that appear in the video (in the conveyor belt). It is very important to set a good frame threshold because OpenCV library analyses videos at 30 fps and this is too much for a real time abalone detection and tracking model.

The video resolution is 720 x 404 and has been reduced by default. We have decided to increase the confidence threshold to 0.7 in order to focus the model to detect and track abalone that are clearly visible. For this case the interest area will be reduced to a dimension of 416 width and 416 of height. The model has analyzed all frames at a speed of 0.025 s/ frame that give a total time to process the video of 1.093 seconds. The processing time takes less time than the duration of the video so we are obtaining a model that can detect and track abalone in real time. The speed of the model has been proved to be very fast and the accuracy because in this case we can observe all abalone and the model detects and tracks all abalone with different tracking numbers of abalone according to all the frames in the test video data.

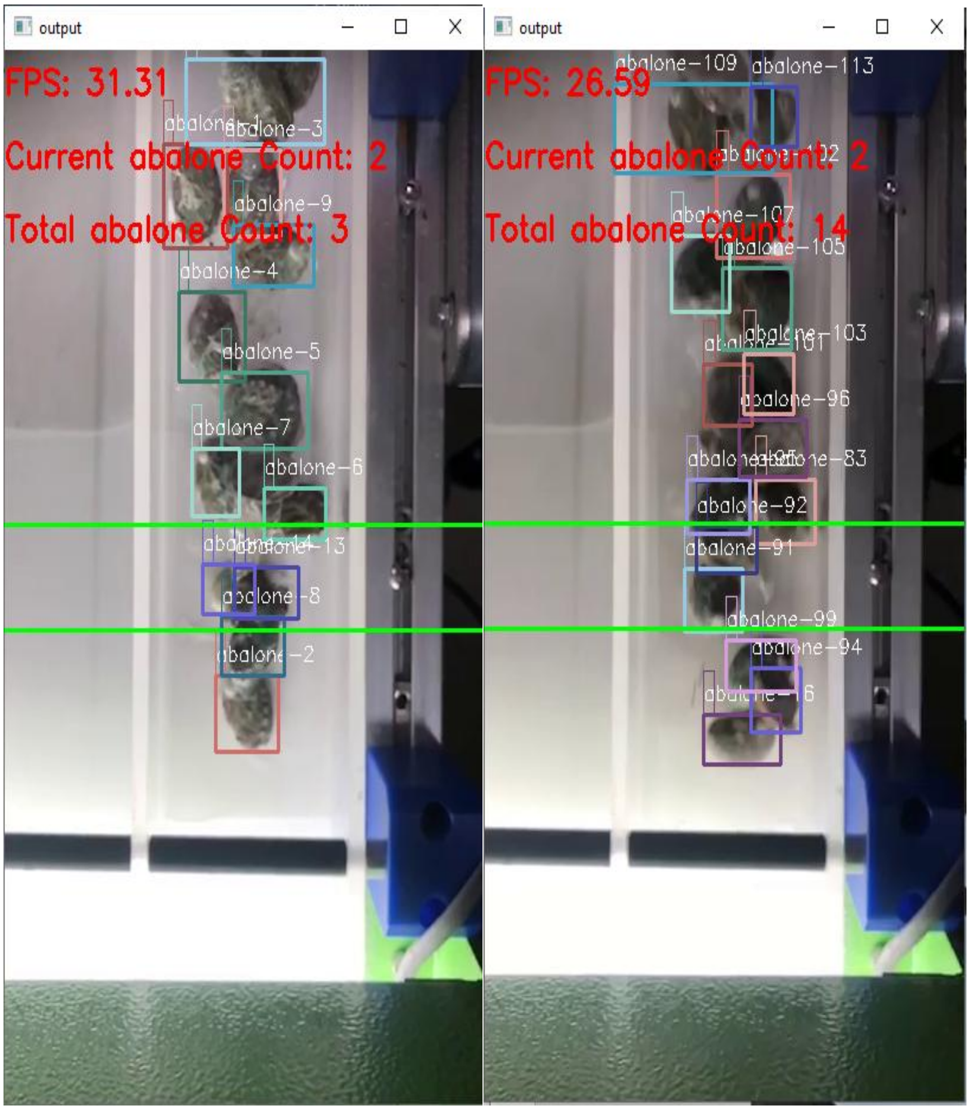


Figure 4-3 ROI with abalone count and frames per second

(Green lines in the image)

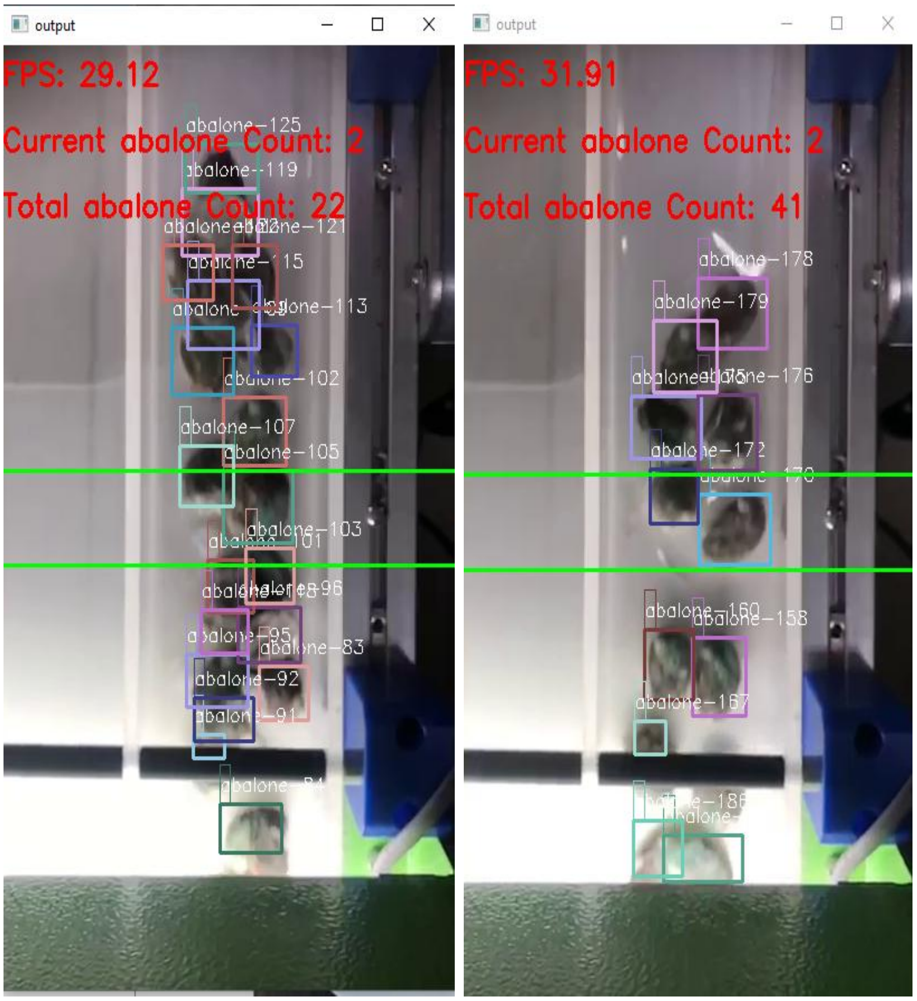


Figure 4-4 Total abalone count

The region of interest count the number of total abalone passing and the current abalone count. We can visualize the each annotated abalone tracked by our model with different colors with the number of frame associated. The total count of abalone by Yolov3 algorithm was 41 abalones in contrast to manual counting of 36 abalones.

5. Grad-CAM via Gradient-based Localization

We propose a technique for classification of lens images from a large class of Convolutional Neural Network based models, making them more clear and explainable.

We took an approach on Gradient-weighted Class Activation Mapping (Grad-CAM) [29] that uses the gradients of any target concept for example according to our data in classification network of 'normal' and 'abnormal' lens flowing into the last convolutional layer to result a distinct localization map spotting the most important regions in the image for prediction. Contrary to previous approaches, we combine Grad-CAM with existing fine-grained to create a high-resolution class-discriminative to visualize whether our lens data is normal or abnormal, and captioning then subjecting to ResNet50-based architecture [30]. With high-resolution and class-discriminative visualization, Grad-CAM provides a textual explanation for model decisions.

Convolutional Neural Networks (CNNs) have enabled extraordinary advances and improvements in a variety of computer vision tasks, including classification of images [31], recognition [32], semantic segmentation [33] to captioning of images [34], and in the recent past, pattern recognition [35] and visual question answering [36].Although these models enable more advanced performance, their lack of sustainability into individually stand-alone components makes them hard to be reliable [37]. Considering image classification a more reliable visual explanation from the model for justifying any region in an image should be class discriminative and high-resolution in capturing the fine-grained detailed in the image.

5.1.1 Data Sets

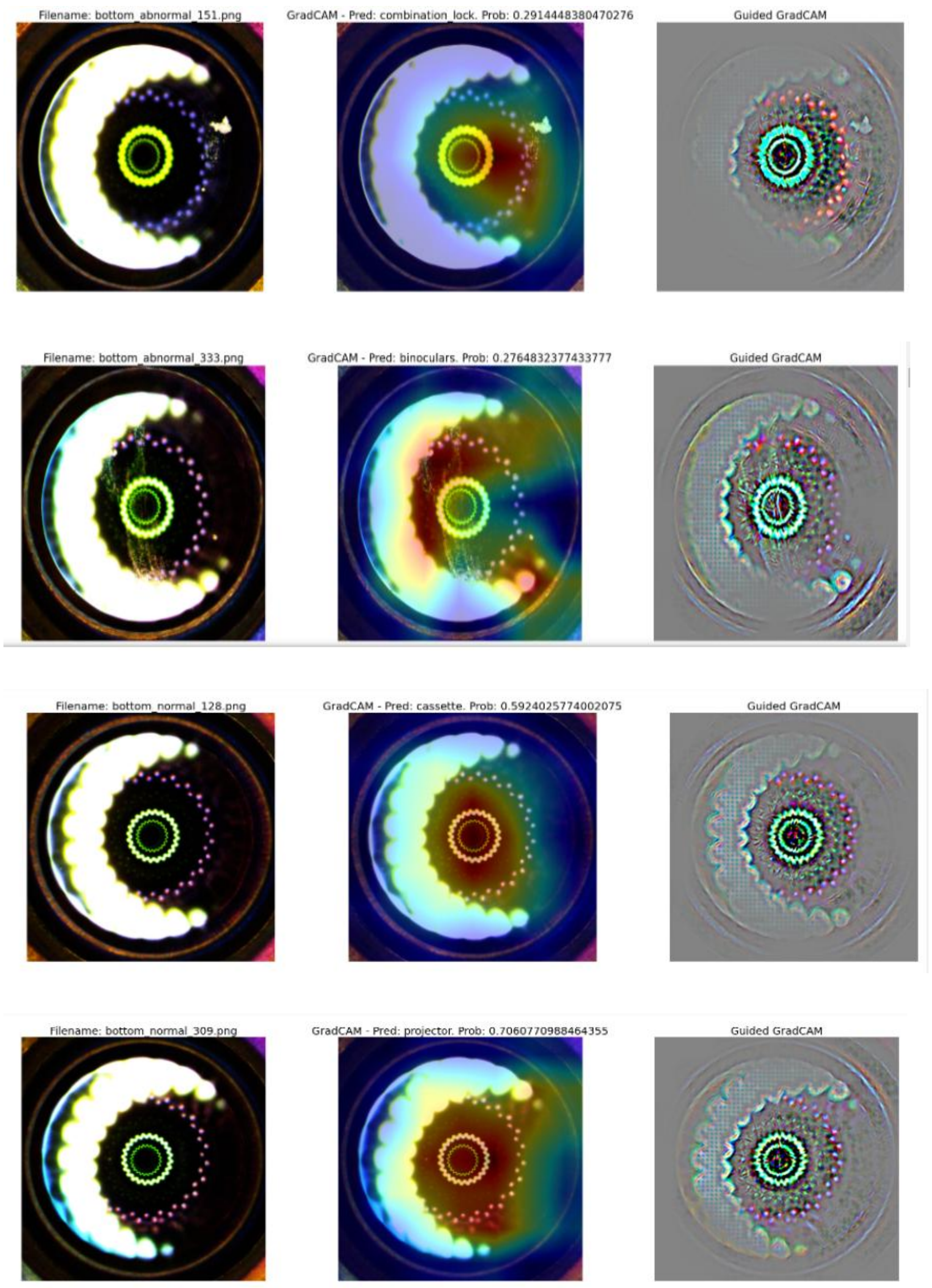
In this study, our image datasets are categorized in to two main classes, i.e. Normal and abnormal images. Our dataset consist of Abnormal and Normal lens images acquired from a company that specializes in production of lenses. There was a challenge in production because a number of lenses contained dust particles, small cracks and other defects which is hard to see with a human eye to distinguish between normal and abnormal images. We acquired a total of 1400 images (Normal images with 700 data sets and abnormal images with 700 images from the total images to be able to classify and predict the class.

5.1.2 Weakly-supervised Localization

Weakly supervised localization applies in the subject of classification of images using Convolution Neural network, where the aim is to localize regions in images using predicted holistic or universal image class labels only before data training [38]. Most relevant aspect to our technique is the Class Activation Mapping approach to localization [39]. This approach modifies image classification Convolution Neural network architectures by replacing fully connected layers with convolutional layers and global average pooling, thus achieving class-specific feature maps. A limitation of CAM is that it requires feature maps prior to soft-max layers, so it only applies to a specific Convolution Neural network architectures performing global average pooling past convolutional map directly before prediction (for example convolution feature maps to global average pooling and soft-max layer). Such architectures may achieve low accuracies compared to general networks on some tasks for example image classification. We initiate an approach of combining feature maps using the gradient signals that does not need any modification in the network architecture.

5.1.3 Visualization

Visualization predicted from the model for interpreting either target in the image should be class discriminative to localize the category in the image and high-resolution in order to capture fine-grained detail in the image. Figure 5-1 shows a number of visualization for the 'Abnormal' class and 'Normal' classes. Gradient visualizations for example Guided Grad-CAM Backpropagation [40] and Deconvolution [41] are high-resolution as far as highlighting fine-grained details in the images, but are not class-discriminative as shown in the Figure 5-1 Guided Grad-CAM Backpropagation in are very similar). In comparison, localization advances similar to CAM or our thesis method Grad-CAM, are tremendously class-discriminative the 'abnormal region' clearly highlights the 'abnormal regions' but not 'normal regions' and vice versa. In order to combine both methods, we show that it is possible to associate current pixel-space gradient visualizations with Grad-CAM to establish Guided Grad-CAM visualizations which are class discriminative and have high resolution. Subsequently, important regions in the image which correlate to any decision of interest are visualized in detail with high resolution even if the image has multiple possible regions. When visualized for 'abnormal', Guided Grad-CAM not only highlights the abnormal regions, but also highlights main region of interest where the abnormal section is, which is important for predicting both images.


Filename: bottom_abnormal_151.png
GradCAM - Pred: combination_lock. Prob: 0.2914448380470276
Guided GradCAM
Filename: bottom_abnormal_333.png
GradCAM - Pred: binoculars. Prob: 0.2764832377433777
Guided GradCAM
Filename: bottom_normal_128.png
GradCAM - Pred: cassette. Prob: 0.5924025774002075
Guided GradCAM
Filename: bottom_normal_309.png
GradCAM - Pred: projector. Prob: 0.7060770988464355
Guided GradCAM

(a) Original image (b) Resnet50 Grad-CAM (C) Guided Grad-CAM

Figure 5-1 Normal and Abnormal Visualization

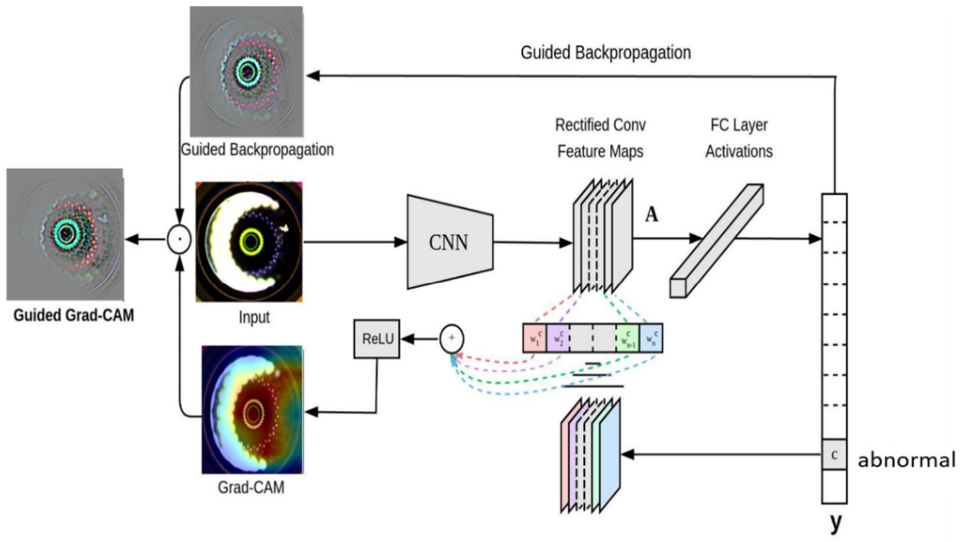


Figure 5-2 Grad-CAM overview

The basic idea behind Grad-CAM is the same as the concept behind CAM: we want to utilize the spatial information that is preserved through convolutional layers, in order to understand the basis to which regions of an input image were critical for a model prediction. Similar to CAM, Grad-CAM uses the feature maps produced by the last convolutional layer of a CNN. Researchers of Grad-CAM argue that, “We can expect the last convolutional layers to have the best compromise between high-level semantics and very detailed spatial information output.” The feature maps produced by the final convolutional layer are shown as A1, A2, and A3, the same as in the CAM.

At this juncture, for CAM we would need to compute global average pooling followed by a fully connected layer. For Grad-CAM, we can do anything – for example, multiple fully connected layers – which is shown as “any neural network layers”. The only requirement is that the layers we insert after A1, A2, and A3 have to be differentiable so that we can get a gradient. Finally, we have our classification outputs for our normal and abnormal data, etc.

The difference between CAM and Grad-CAM is in how the feature maps A1, A2, and A3 are weighted to make the final heat map. In CAM, we weight these feature maps using weights taken out of the last fully-connected layer of the network. In Grad-CAM, we weight the feature maps using "alpha values" that are calculated based on gradients. Therefore, Grad-CAM does not require a particular architecture, because we can calculate gradients through any kind of neural network layer we want. The "Grad" in Grad-CAM stands for "gradient."

As shown in Figure 5-4, to achieve the class-discriminative localization map Grad-CAM $L^c_{\text{Grad-CAM}} \in R^{u \times v}$ of width u and height v for any class c, we first compute the gradient of the score for class c and y^c is the raw output of the neural network for class *c*, before the soft-max is applied to transform the raw score into a probability, taking into account activations of the feature map A^k of a convolutional layer $\frac{\partial y^c}{\partial A^k}$. For a 2D input image, this gradient is 3D, with the same shape as the feature maps. There are *k* feature maps each of height *v* and width *u*, *i.e.* collectively the feature maps have shape [*k, v, and u*]. This means that the gradients calculated in earlier are also going to be of shape [*k, v, u*].These gradients flowing back are global-average-pooled via the height and width dimensions of i and j respectively to obtain the neuron critical weightsα^c_k:

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

During computation of α_k^c while back-propagating gradients taking into account the activations, the valid computation results to subsequent matrix products of the resulting weight and the gradient with consideration of activation functions till the last convolution layer that the gradients are being propagated to. Hence, this weight α_k^c represents a partial linearization of the deep network downstream from A, and apprehends the critical of feature map k for a target class c. We combine weight of forward activation map, moreover following by a ReLU in order to retrieve,

$$L^c_{\text{Grad-CAM}} = ReLU \underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}}$$

The results in a coarse heatmap of the equal size as the convolutional feature maps (14 × 14 like in the case of last convolutional layers of VGG [42] and AlexNet [43] networks) 3 . We use the ReLU function to the linear combination of maps considering our only interest in the features that have decisive impact on the class of 'interest,' for example pixels whose intensity should be elevated to increase y^c. Negative pixels have the probability of belonging to other categories in the image. In this case, ReLU is important because localization maps occasionally highlights more than probable class and perform poorly at localization. Figures 5-1, show Grad-CAM visualizations for 'abnormal' and 'normal' respectively. In general, y^c does not need to be the class score produced by CNN.

5.1.4 CAM vs. Grad-CAM

We took an approach on the connections between Class Activation Mapping and Grad-CAM [44], and properly demonstrate that Grad-CAM hypothesize

CAM for a broad variation of CNN architectures. Looking back that CAM produces a localization map for classification of image (CNN) with particular kind of architecture where global average pooled convolutional feature maps are fed directly into soft-max. Precisely, let the penultimate layer produce K feature maps, $A^k \in R^{u \times v}$, with each element indexed by i, j. So A^k_{ij} refers to the activation at location (i, j) of the feature map A^k. These feature maps are then spatially pooled using Global Average Pooling (GAP) and linearly transformed to produce a score Y^c for each class c,

$$Y^c = \sum_k \underbrace{w_k^c}_{\text{class feature weights}} \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{A_{ij}^k}_{\text{feature map}} \tag{1}$$

We observe that Grad-CAM maps become successively worse as we move prior convolutional layers as they have smaller receptive fields and only focus on less semantic local features. Let us define F^k to be the global average pooled output,

$$F^k = \frac{1}{Z}\sum_i \sum_j A_{ij}^k$$

CAM computes the final scores by,

$$Y^c = \sum_k w_k^c \cdot F^k$$

Where w$_k^c$ is the weight connecting the kth feature map with the cth class. Taking the gradient of the score for class c (Y^c) with respect to the feature map F^k we get,

$$\frac{\partial Y^c}{\partial F^k} = \frac{\frac{\partial Y^c}{\partial A_{ij}^k}}{\frac{\partial F^k}{\partial A_{ij}^k}}$$

(2)

Taking partial the derivative of (F^k) w.r.t.A_{ij}^k , we can see that $\frac{\partial F^K}{\alpha A_{IJ}^K}$=$\frac{1}{z}$. Substituting this in (2) computation, we get,

$$\frac{\partial Y^c}{\partial F^k} = \frac{\partial Y^c}{\partial A_{ij}^k} \cdot Z$$

From CAM computation of we get that, $\frac{\partial y^c}{\partial F^k}$ $= w_k^c$ therefore,

$$w_k^c = Z \cdot \frac{\partial Y^c}{\partial A_{ij}^k}$$

(3)

Summing both sides of computation (3) over all pixels (i, j),

$$\sum_i \sum_j w_k^c = \sum_i \sum_j Z \cdot \frac{\partial Y^c}{\partial A_{ij}^k}$$

Since Z and w_k^c do not depend on (i, j), rewriting this as

$$Z w_k^c = Z \sum_i \sum_j \frac{\partial Y^c}{\partial A_{ij}^k}$$

Note that Z is the number of pixels in the feature map (or Z = $\sum_I \sum_j$ in equation (1) of Global Average Pooling). Thus, we can re-order terms and see that

$$w_k^c = \sum_i \sum_j \frac{\partial Y^c}{\partial A_{ij}^k}$$

Up to a proportionality constant (1/Z) that gets normalized out during visualization, the expression for w_k^c is identical to α_k^c used by Grad-CAM.

Therefore, Grad-CAM is a strict generalization of CAM. This generalization allows us to generate visualization from CNN-based models that cascade convolutional layers with much more complex interactions.

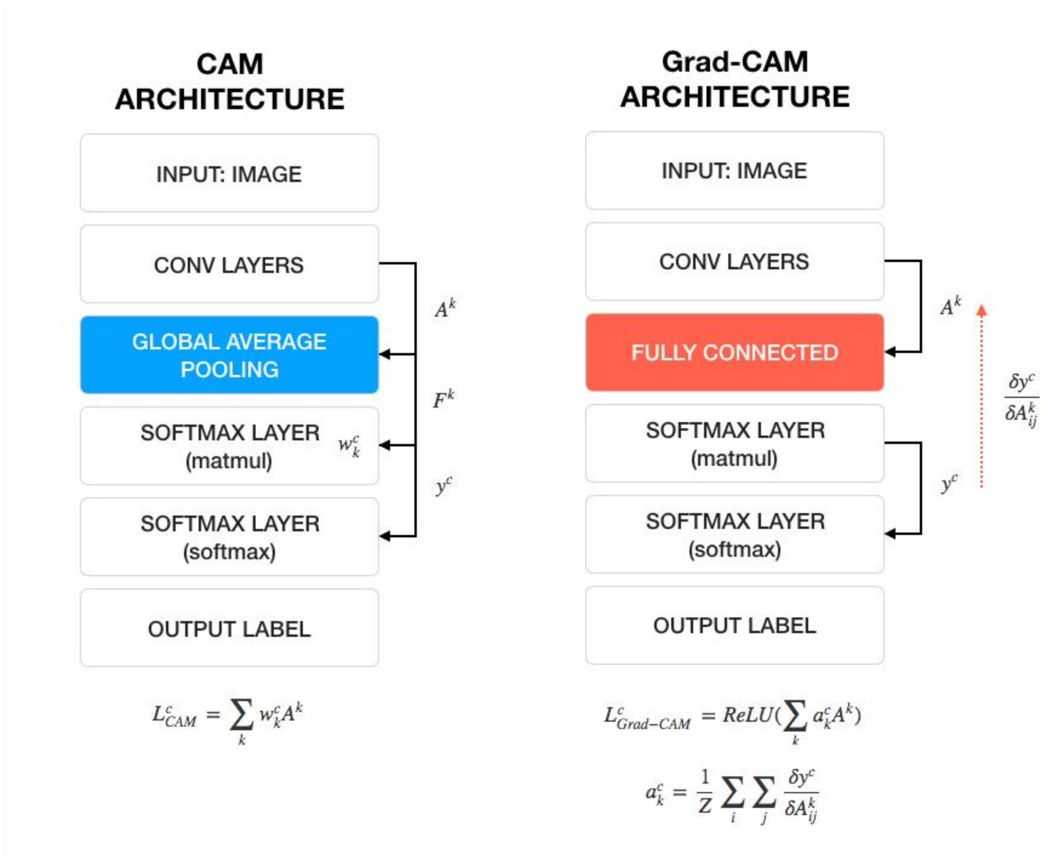


Figure 5-3 CAM vs. Grad-CAM

5.1.5 Guided Grad-CAM

Although Grad-CAM is class discriminative and localizes relevant image sections, it lacks the ability to highlight fine grained details like pixel-space gradient visualization methods like Guided Backpropagation [45], Deconvolution [46]. Guided Backpropagation focuses on visualizing gradients with aspect to the image where negative gradients are suppressed when back propagating through ReLU layers. Naturally objectively captures the pixels detected by neurons, not the ones that suppress neurons. See Figure 5-1, where Grad-CAM can easily localize the abnormal image; however, it is uncertain from the crude heat map why the network predicts this particular instance as 'binoculars'. In order to combine the best aspects of both, we merge Guided Backpropagation and Grad-CAM visualizations via element-wise multiplication ($L^c_{\text{Grad-CAM}}$ is first up sampled to the input image resolution using bilinear interpolation).Resulting to visualization that is both has high-resolution (when the class of interest is 'abnormal image', it identifies important 'abnormal image' features like scratches on the image, presence of dust particles: generally abnormality in the image) and class-discriminative (it highlights the 'abnormal as image' but not the 'normal image vice versa'). Replacing Guided Backpropagation with Deconvolution gives related results, but we established Deconvolution visualizations to have disadvantages while Guided Backpropagation to be generally less noisy.

5.1.6 Training and evaluation

Researchers have used a lot of computational power training deep neural networks that classify images into categories. Many were trained on a subset of ImageNet with an enormous database of images manually labeled with tremendous number categories as part of the ImageNet extensive Visual Recognition Challenge (ILSVRC) [47] which, since it started in 2010, has seen a lot of improvement each year since the challenge ended in 2017. Each application vary undermost architecture and performance, rather than (CNN), a type of deep learning network with outstanding performance in classifying features in images. Fortunately, these pre-trained models have been made accessible for researchers interested to experiment with, saving more time and expense of having to do the training from scratch. Among them, ResNet-50 [48] under Residual Networks defines good performance between speed and accuracy. ResNet-50 is a pre-trained convolutional neural network which utilizes more than a million images retrieved from the ImageNet dataset during training process. ResNet-50 adopts deep residual learning on 50 layers and has the ability to classify more of objects into a thousand classes like VGG-16 while also supporting an input image size of 224 x 224 height and width pixels.

In training, we used a resnet-50 model pre-trained on ImageNet weights as the CNN branch. We subjected a total of 1400 images (Normal images with 700 data sets and Abnormal images with 700 images from the total images, 70% were used for training + validation, and the remaining 30% were used for testing purposes. The training + validation dataset images was further split into a 70:30 ratio, i.e (70%) images for training purposes and (30%) images for validation purposes. Resnet50 pre-trained network followed by a dense layer with several neurons matching the number of classes by using Softmax activation. SGD optimizer is used with 0.01 learning rate. Figure 5-4 plots the training and validation accuracy and the training and validation loss during the training process.

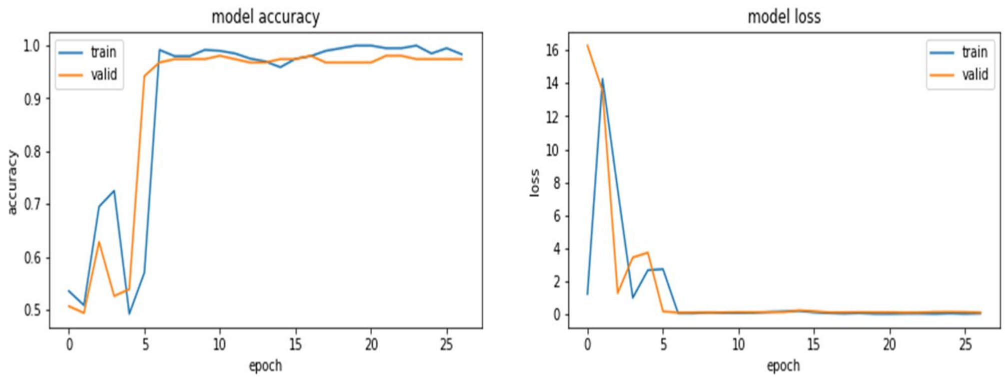


Figure 5-4 30 epochs plot ResNet-50 model

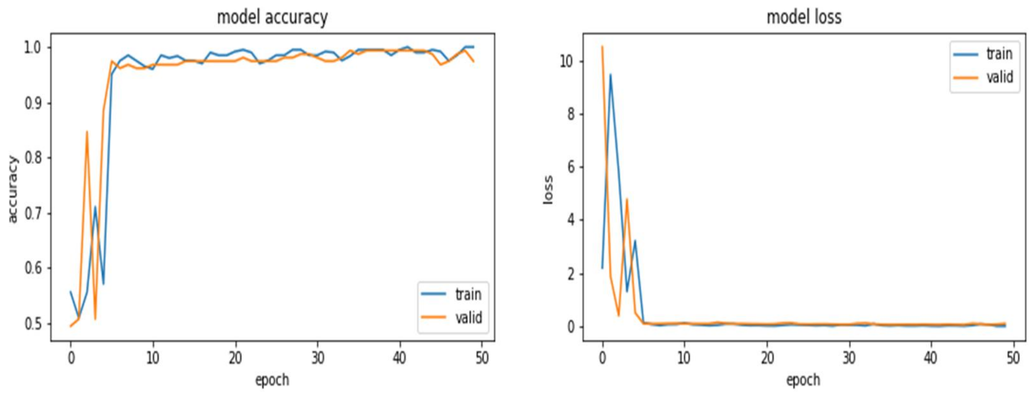


Figure 5-5 50 epochs plot ResNet-50 model

The training performance of the ResNet-50 model under our study were evaluated in terms of important parameters in training accuracy, validation accuracy, training loss, and validation loss at 30 and50 epochs. The results of these parameters can be found in Figure 5-4 and 5-5. The graphs of training loss vs. validation loss and training accuracy vs. validation accuracy of our model reported on a 70% the training dataset, whereas the validation curves are reported 30% from the validation dataset.

The model predicted some of the random normal and abnormal images displayed in figure 5-6 and 5-7 consecutively under 30 epoch.

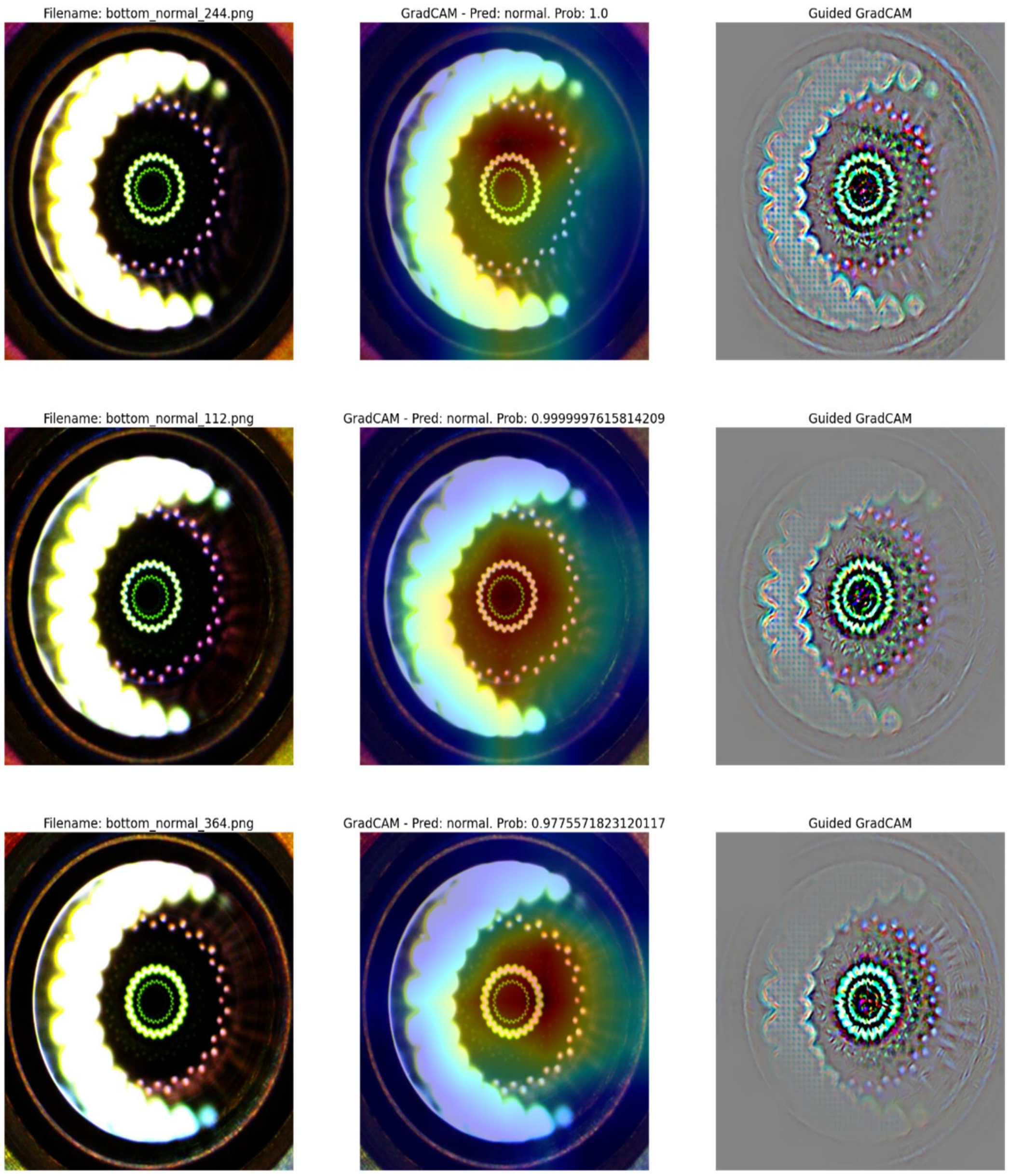

Filename: bottom_normal_244.png
GradCAM - Pred: normal. Prob: 1.0
Guided GradCAM
Filename: bottom_normal_112.png
GradCAM - Pred: normal. Prob: 0.9999997615814209
Guided GradCAM
Filename: bottom_normal_364.png
GradCAM - Pred: normal. Prob: 0.9775571823120117
Guided GradCAM

Figure 5-6 Grad-CAM prediction of normal images

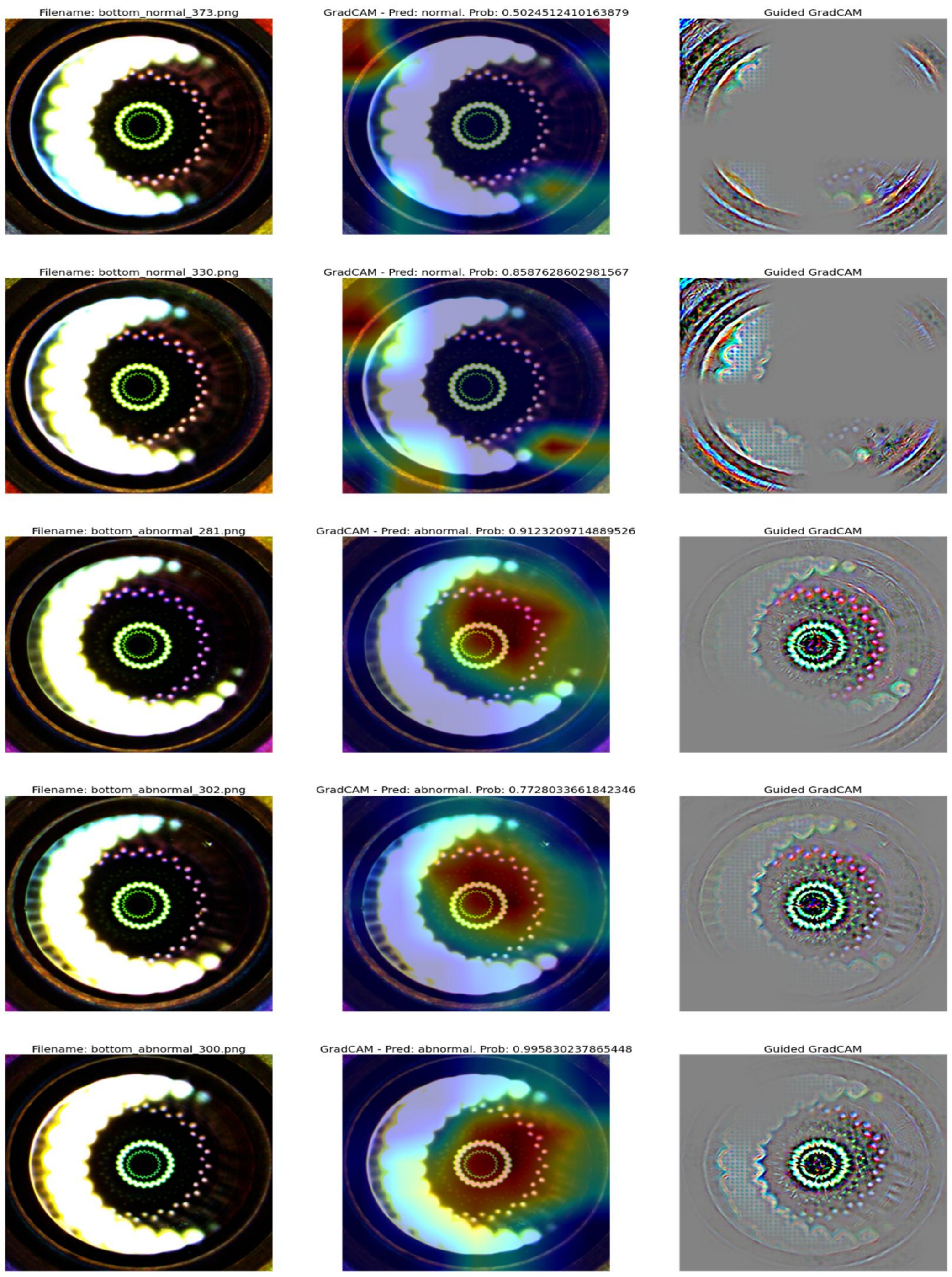

Filename: bottom_normal_373.png
GradCAM - Pred: normal. Prob: 0.5024512410163879
Guided GradCAM
Filename: bottom_normal_330.png
GradCAM - Pred: normal. Prob: 0.8587628602981567
Guided GradCAM
Filename: bottom_abnormal_281.png
GradCAM - Pred: abnormal. Prob: 0.9123209714889526
Guided GradCAM
Filename: bottom_abnormal_302.png
GradCAM - Pred: abnormal. Prob: 0.7728033661842346
Guided GradCAM
Filename: bottom_abnormal_300.png
GradCAM - Pred: abnormal. Prob: 0.995830237865448
Guided GradCAM

Figure 5-7 Grad-CAM prediction of abnormal images

Under 30 epochs the model predicted 5 images in wrong class as shown above in figure 5-7. The model also predicted some of the normal and abnormal images displayed in figure 5-8 and 5-9consecutively under 30 epoch.

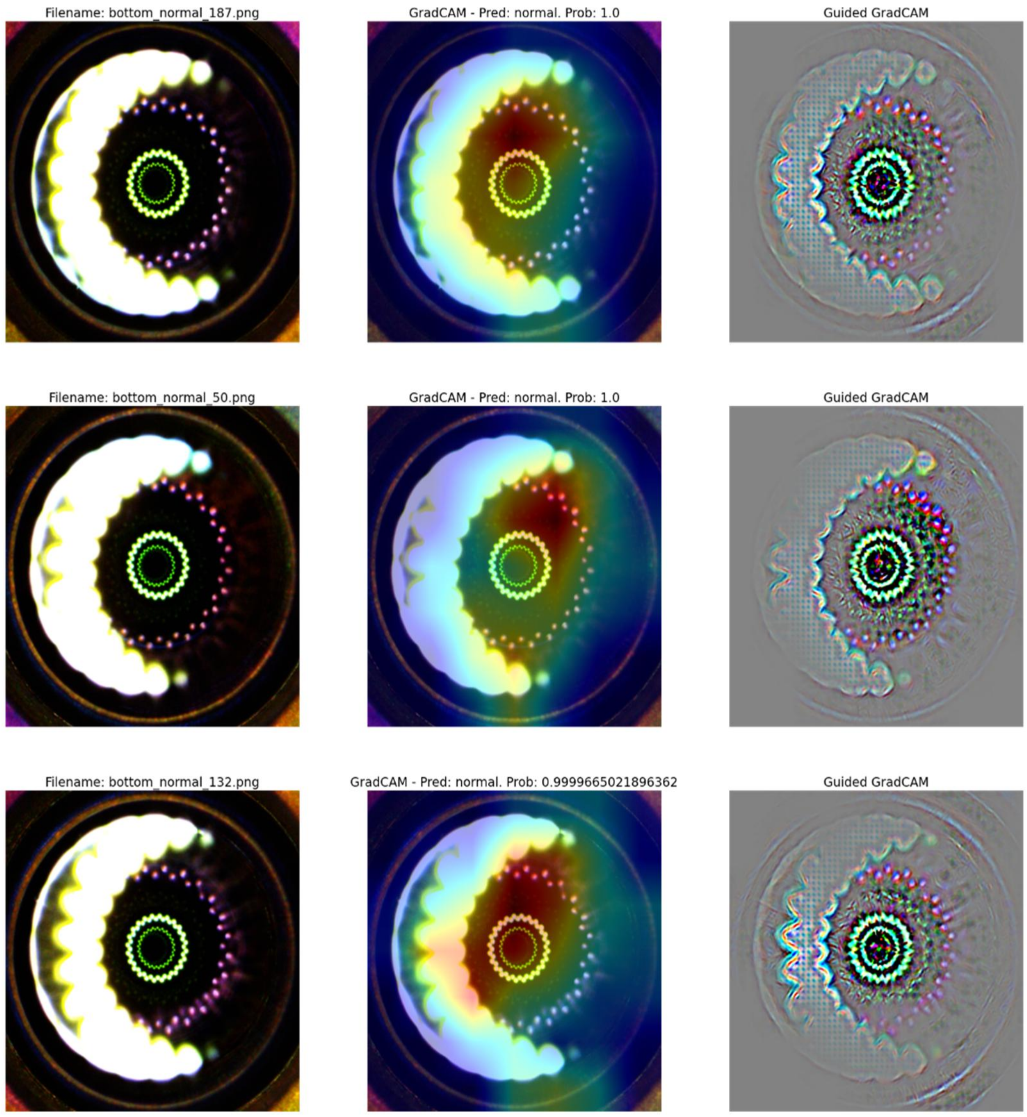

Filename: bottom_normal_187.png
GradCAM - Pred: normal. Prob: 1.0
Guided GradCAM
Filename: bottom_normal_50.png
GradCAM - Pred: normal. Prob: 1.0
Guided GradCAM
Filename: bottom_normal_132.png
GradCAM - Pred: normal. Prob: 0.9999665021896362
Guided GradCAM

Figure 5-8 Grad-CAM prediction of normal images

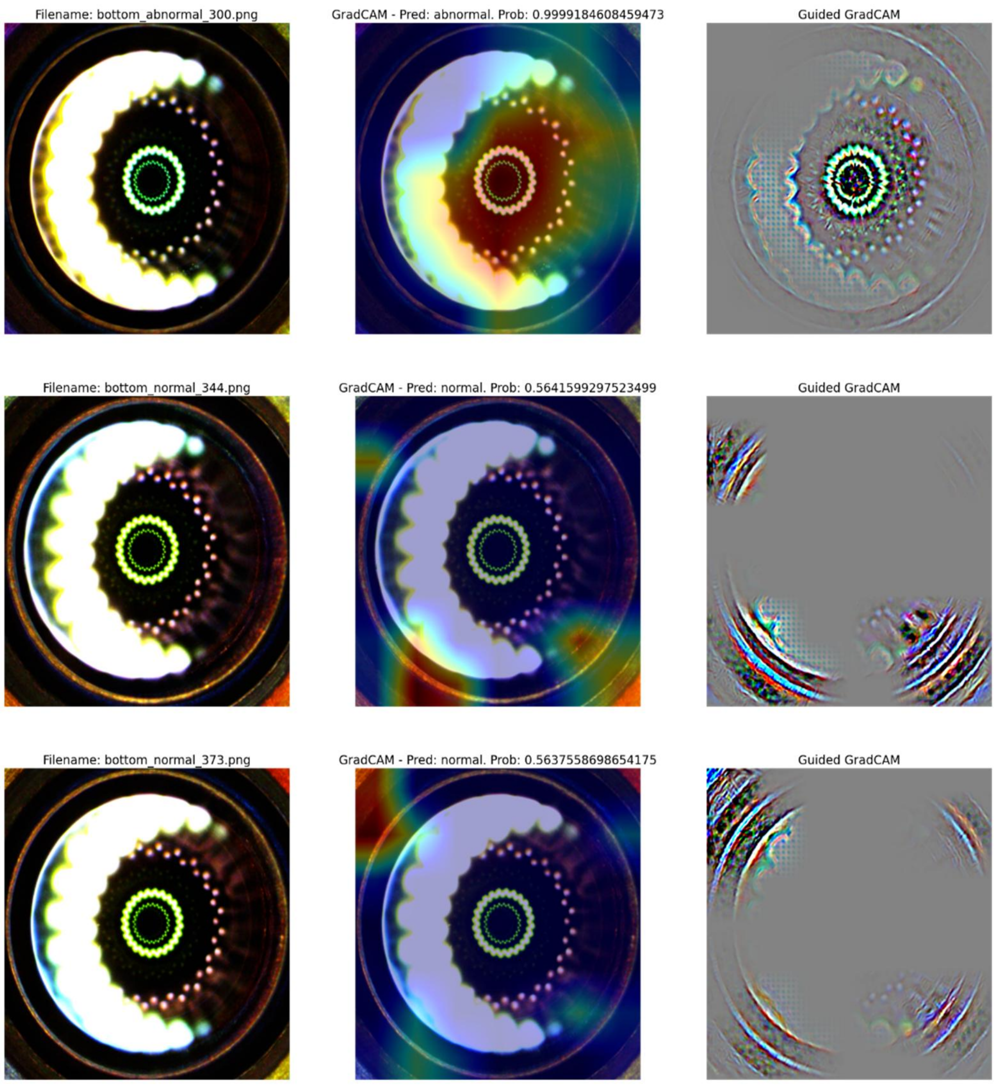

Filename: bottom_abnormal_300.png
GradCAM - Pred: abnormal. Prob: 0.9999184608459473
Guided GradCAM
Filename: bottom_normal_344.png
GradCAM - Pred: normal. Prob: 0.5641599297523499
Guided GradCAM
Filename: bottom_normal_373.png
GradCAM - Pred: normal. Prob: 0.5637558698654175
Guided GradCAM

Figure 5-9 Grad-CAM prediction of abnormal images

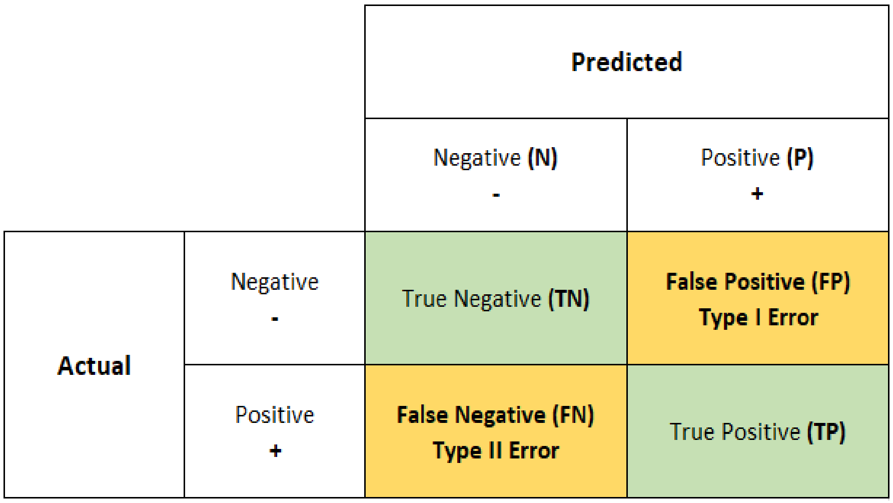
Under 50 epochs the model predicted 5 images in wrong class as shown above in figure 5-9.

5.1.7 Confusion Matrix

Classification accuracy can be disingenuous if there is an imbalanced number of results in each class or two classes in the dataset. Confusion matrix is used for assessing the performance of a classification model, for the number of target classes. The matrix analyzes the actual target values with those predicted by the machine learning model. Confusion matrix is used to analyze the performance of a classification model and can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score.

Confusion matrices are extensively used because they give a better outcome of a model's performance than classification accuracy does. For example, in classification accuracy. Imagine that your datasets have two classes where 65% of the data belongs to class X, and 35% belongs to class Y. More so, in another instance that your classification model can correctly classify all the in the datasets in class X but misclassifies the datasets of class Y. In this context, the model is 65% accurate according to the classification. The confusion matrix displays in detail the correctly and incorrectly classified instances for all the classes and will, therefore, give a better understanding into the performance of your model. As shown in the figure below:

1. A good performing model has high TP and TN rates, while low FP and FN rates.
2. In case you have an imbalanced dataset to work with, it is recommended to use confusion matrix as your evaluation criteria for your machine learning model to get a better understanding.



		Predicted	
		Negative (N) -	Positive (P) +
Actual	Negative -	True Negative (TN)	**False Positive (FP) Type I Error**
	Positive +	**False Negative (FN) Type II Error**	True Positive (TP)

Figure 5-10 Binary confusion matrix

- True Positives (TP): if the actual value is Positive and is predicted as Positive.
- True negatives (TN): if the actual value is Negative and prediction is also Negative.
- False positives (FP): When the actual is Negative, but prediction is Positive. Also known as the Type 1 error
- False negatives (FN): When the actual is Positive, but the prediction is Negative. Also known as the Type 2 error

Classification measures under confusion matrix help achieve better understanding and analysis of our model and final performance outcome:

Recall - Is a measure of actual observations which are predicted correctly, i.e., how many observations of positive class are predicted as positive. It is also known as Sensitivity. Recall is basically a legit decision of evaluation metric when acquiring plenty positives as possible.

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

The above equation shows from all the classes that have been predicted as positive. Precision should be high as possible.

Precision - is a measure of correctness that is achieved in true prediction. In the equation below it tells us how many predictions are actually positive out of all the total positive predicted.

$$\textit{Precision} = \frac{TP}{TP + FP}$$

Accuracy - measures how recurrently the classifier makes a correct prediction. It's the ratio between the number of correct predictions and the total number of predictions. The accuracy metric not preferable to imbalanced data. The downside to accuracy metric in imbalanced data is that the model prediction favors data that only belongs to the majority class label, the accuracy will be high. But the model is not accurate.

F-measure / F1-score - is a number between zero and one. F1- score has similar meaning to recall and precision because it is not sensitive to extremely large values, unlike simple averages. As shown in the equation below F1 controls equilibrium between the recall and precision for the classifier. If the precision is low, the F1 will also be low similar to when recall is low the F1 score is low. In practice, when trying to increase the precision of the model, the recall value descends and vice-versa. The F1-score appends both the trends in a single value.

$$F\text{ - }measure = \frac{2*Recall*Precision}{Recall + Precision}$$

Based on the analysis of our model the confusion matrix indicates that there is no imbalanced accuracy of both the normal and abnormal class as shown in the figure 5-11

```
[[74  5]
 [ 2 75]]

[Text(0, 0.5, 'normal'), Text(0, 1.5, 'abnormal')]
```

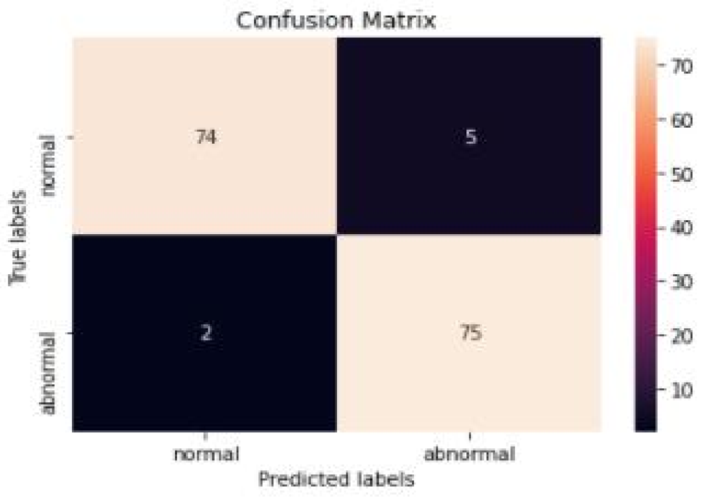


Figure 5-11 Normal and abnormal confusion matrix prediction

Our model predicted 74 normal images as true positives, 75 abnormal images as true negatives, also 2 normal image as false positive (type 1 error) and 5 abnormal images as false negative. Table 3and 4 shows the precision, recall, f1 – score, support and accuracy.

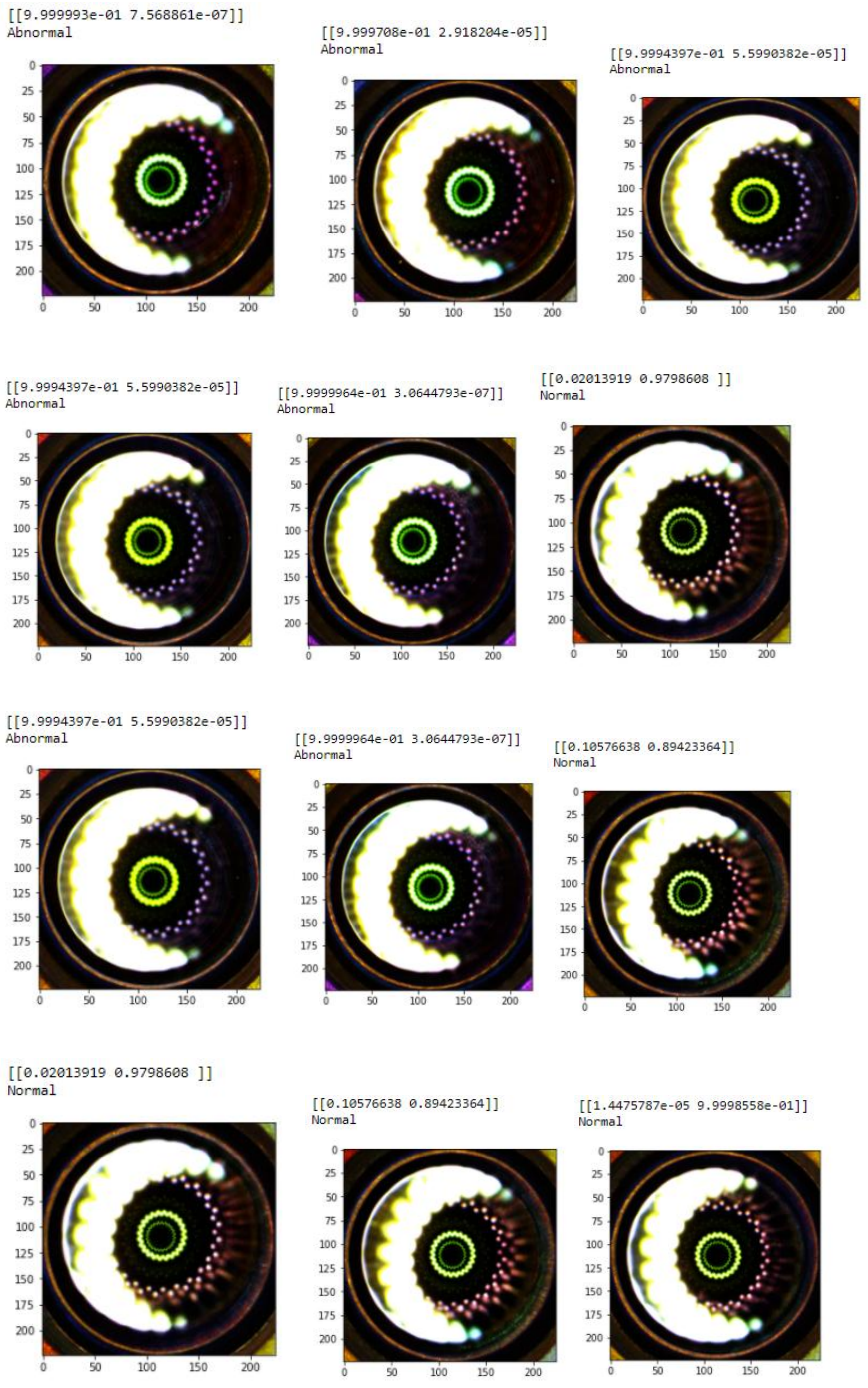
Table 3 Confusion matrix report

	precision	Recall	F1 - score	support
0	0.97	0.94	0.95	79
1	0.94	0.97	0.96	77

Table 4 confusion matrix accuracy prediction

accuracy			0.96	156
Macro avg	0.96	0.96	0.96	156
Weighted avg	0.96	0.96	0.96	156

We use Resnet-50 model for predicting normal and abnormal images. Below are normal and abnormal no label predicted images from our dataset. We can see the class and percentage prediction.



```
[[9.999993e-01 7.568861e-07]]
Abnormal

[[9.999708e-01 2.918204e-05]]
Abnormal

[[9.9994397e-01 5.5990382e-05]]
Abnormal

[[9.9994397e-01 5.5990382e-05]]
Abnormal

[[9.9999964e-01 3.0644793e-07]]
Abnormal

[[0.02013919 0.9798608 ]]
Normal

[[9.9994397e-01 5.5990382e-05]]
Abnormal

[[9.9999964e-01 3.0644793e-07]]
Abnormal

[[0.10576638 0.89423364]]
Normal

[[0.02013919 0.9798608 ]]
Normal

[[0.10576638 0.89423364]]
Normal

[[1.4475787e-05 9.9998558e-01]]
Normal
```

Figure 5-12 Random test data prediction

6. Results and Discussion

In the first section of our study under abalone counting Yolov3 algorithm trained model was tested on abalone video using the image resolution of 416 × 416 pixels set at batch size 1 in order to maintain consistency. YOLOV3 model detected abalones in the test dataset achieving good detection result demonstrating the total abalone counted as shown in Figure 4-4 passing through the region of interest (ROI) resulting to the final total number of abalone counted in contrast to the manually counted abalone.

Second section we explored of the pre-trained networks are while retraining the latter layers in order to adjust and fit our normal and abnormal classification problem. The experimental results show the resulting performances of pre-trained networks, which has been built and trained from scratch, and which augments the concept of knowledge transfer despite the big difference between our image datasets with the application of Grad-CAM visualization method on pre-trained ResNet50 to get a better understanding which features appear relevant to the CNNs in order to distinguish between the difference between the normal and abnormal image classes.

7. Conclusion

In the first research resolves the problem of abalone counting due to tangling of abalone resulting to being counted as one conveyor belt and our approach on using YOLOv3 detection and Deep Sort algorithms to detect and track abalone clearly shows improvement. The effectiveness of the proposed YOLOv3 algorithm in detection was verified in the test data application. From the final results of YOLOv3 algorithm, the analysis has proved that the improved detection algorithm has obvious improvement in detection effect and real-time performance.

For tracking of abalone the Deep Sort tracking algorithm was used. Based on analysis, we see that the Deep Sort algorithm presented greater performance advantages in complex scenes, and was robust to interference such as abalone tangling and camera movements. Hence it can be used in real time performance. However, the performance of the algorithm depends GPU with very high graphics to a large extent.

Despite the proposed algorithm in this paper obtaining sufficient results in real-time detection and tracking of abalone still needs more work in research to be carried out to improve the performance of our methods. The future aspect on data labelling needs automation for more precision during detection due to the tedious work when labelling each abalone, and the tracking of abalone needs to be analysed for more accurate in the detection of unseen abalone which is harder to detect.

Second section, Gradient-weighted Class Activation Mapping algorithm applied to any CNN based models is clearer by producing visual explanations. Furthermore, we introduced our Grad-CAM localizations with existing high resolution, resulting to class discriminative output. Our visualizations gives a better performance in contrast to other existing approaches on both aspects: interpretability and relying to original model. Lastly, we showed an extensive application of Grad-CAM to Resnet50 model available architectures for tasks including the classification of images, image captioning (highlighting the target regions in the images) and providing true visual explanations for possible model decisions. We affirm that artificial intelligent system should not only be intelligent, but also be capable of providing a reason about its actions for researchers to trust it. Future work comprise explaining the decisions made by deep neural networks in machine learning such as supervised learning, pattern recognition and video applications.

References

1. Li Chen and John Ryan, "Abalone in Diasporic Chinese Culture: The Transformation of Biocultural Traditions through Engagement with the Western Australian Environment", Heritage, vol. 1, no. 1, pp. 122-41, 2018

2. Cook, P.A., 2014. The worldwide abalone industry. Modern Economy, 5(13), p.1181.
3. Park, K., Ahn, B.-W., Park, Y.-S., & Bae, C.-O. (2017). A Study on Abalone Young Shells Counting System using Machine Vision. Journal of the Korean Society of Marine Environment and Safety, 23 (4), 415–420. https://doi.org/10.7837/kosomes.2017.23.4.415

4. Nizar, T. N., Anbarsanti, N., & Prihatmanto, A. S. (2014, November). Multi-object tracking and detection system based on feature detection of the intelligent transportation system. In 2014 IEEE 4th International Conference on System Engineering and Technology (ICSET) (Vol. 4, pp. 1-6). IEEE.
5. Fernandez-Sanjurjo, M., Bosquet, B., Mucientes, M., & Brea, V. M. (2019). Real-time visual detection and tracking system for traffic monitoring. Engineering Applications of Artificial Intelligence, 85, 410-420.
6. Khemmar, R., Gouveia, M., Decoux, B., & Ertaud, J. Y. (2019). Real time pedestrian and object detection and tracking-based deep learning. application to drone visual tracking.
7. Kakadiya, R., Lemos, R., Mangalan, S., Pillai, M., & Nikam, S. (2019, June). Ai based automatic robbery/theft detection using smart surveillance in banks. In 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 201-204). IEEE.
8. Chen, Y., Chen, S., Ren, H., Gao, Z., & Liu, Z. (2020). Path Tracking and Handling Stability Control Strategy with Collision Avoidance for the Autonomous Vehicle under Extreme Conditions. IEEE Transactions on Vehicular Technology.
9. Li, S., Tao, F., Shi, T., & Kuang, J. (2019, December). Improvement of YOLOv3 network based on ROI. In 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC) (Vol. 1, pp. 2590-2596). IEEE
10. Mehralian, M. A., & Soryani, M. (2020). EKFPnP: extended Kalman filter for camera pose estimation in a sequence of images. IET Image Processing, 14(15), 3774-3780.

11. Ffmpeg tools, 03 2020, [online] Available: https://www.ffmpeg.org/.
12. LabelImg tool, 05 2021[online] Available: https://pypi.org/project/labelImg/1.4.0/.
13. Kapania, Shivani, et al. "Multi object tracking with UAVs using deep SORT and YOLOv3 RetinaNet detection framework." Proceedings of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems. 2020.
14. Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: towards real-time object detection with region proposal networks. IEEE transactions on pattern analysis and machine intelligence, 39(6), 1137-1149.
15. Santos, Adson M., et al. "Counting Vehicle with High-Precision in Brazilian Roads Using YOLOv3 and Deep SORT." 2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). IEEE, 2020.
16. Farhadi, Ali, and Joseph Redmon. "Yolov3: An incremental improvement." Computer Vision and Pattern Recognition. Berlin/Heidelberg, Germany: Springer, 2018.
17. Dong, Enzeng, et al. "An improved convolution neural network for object detection using YOLOv2." 2018 IEEE International Conference on Mechatronics and Automation (ICMA). IEEE, 2018.
18. Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv: 1804.02767 (2018).
19. Zhao, Liquan, and Shuaiyang Li. "Object detection algorithm based on improved YOLOv3." Electronics 9.3 (2020): 537.
20. Horzyk, Adrian, and Efe Ergün. "YOLOv3 precision improvement by the weighted centers of confidence selection." 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020.
21. Zhao, Liquan, and Shuaiyang Li. "Object detection algorithm based on improved YOLOv3." Electronics 9.3 (2020): 537.
22. Shin, Seokyong, Hyunho Han, and Sang Hun Lee. "Improved YOLOv3 with duplex FPN for object detection based on deep learning." The International Journal of Electrical Engineering & Education (2021): 0020720920983524.
23. Yang, Hongbo, et al. "Research on underwater object recognition based on YOLOv3." Microsystem Technologies 27.4 (2021): 1837-1844.
24. Kapania, Shivani, et al. "Multi object tracking with UAVs using deep SORT and YOLOv3 RetinaNet detection framework." Proceedings of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems. 2020.

25. Kapania, Shivani, et al. "Multi object tracking with UAVs using deep SORT and YOLOv3 RetinaNet detection framework." Proceedings of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems. 2020.
26. Lei, Wei, Dongjun Huang, and Xiwen Cui. "Moving object tracking in video surveillance using YOLOv3 and MeanShift." Tenth International Conference on Graphics and Image Processing (ICGIP 2018). Vol. 11069. International Society for Optics and Photonics, 2019.
27. https://github.com/AlexeyAB/darknet.
28. Configuring your custom .cfg, obj.data, obj.names, train.txt and test.txt files https://github.com/theAIGuysCode/YOLOv3-Cloud-Tutorial
29. Selvaraju, Ramprasaath R., et al. "Grad-CAM: Why did you say that?." *arXiv preprint arXiv:1611.07450* (2016).
30. Sharma, Neha, Vibhor Jain, and Anju Mishra. "An analysis of convolutional neural networks for image classification." *Procedia computer science* 132 (2018): 377-384.
31. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.
32. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.
33. J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.
34. H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, et al. From Captions to Visual Concepts and Back. In CVPR, 2015.
35. A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, and D. Batra. Visual Dialog. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
36. D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. Iqa: Visual question answering in interactive environments. arXiv preprint arXiv:1712.03316, 2017
37. Z. C. Lipton. The Mythos of Model Interpretability. ArXiv e-prints, June 2016.
38. M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? – Weakly-supervised learning with convolutional neural networks. In CVPR, 2015.
39. B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. In CVPR, 2016.

40. J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for Simplicity: The All Convolutional Net. CoRR, abs/1412.6806, 2014

41. M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In ECCV, 2014.
42. K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In ICLR, 2015.
43. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.
44. J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff. Top-down Neural Attention by Excitation Backprop. In ECCV, 2016
45. J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for Simplicity: The All Convolutional Net. CoRR, abs/1412.6806, 2014.

46. M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In ECCV, 2014.
47. https://image-net.org/challenges/LSVRC/
48. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.