Computer Vision for Water Detection using Deep Learning

Project Report

Supervised by

Dr.jinfeng liu

Nirav Raiyani

Student ID: 1557017

Department of Chemical and Materials Engineering

University of Alberta

Executive Summary

This work uses Mask R-CNN, a CNN architecture, for the purpose of water curtain detection and further, for controller feedback. The report provides a brief overview of some important convolution operations along with the structure of a Convolutional Neural Network and Stochastic Gradient Descent for model training. The different modules of Mask R-CNN such as ResNet, Region Proposal Network, and Fully Convolutional Network are explained in sufficient depth. As for training, transfer learning is used to reduce the computational time and the size of the training data. The trained model predicts very accurate results and provides a measure to quantify the water distribution.

Contents

Executive Summary **1**

1 Introduction **5**

2 Foundational Information **7**

2.1 Convolution Operation 7

2.2 Padding 9

2.3 Striding 10

2.4 Max - Pooling 10

2.5 CNN(Convolutional Neural Network) 11

2.6 SGD(Stochastic Gradient Descent) 13

2.7 A Simple Network 15

3 Object detection and segmentation **17**

3.1 Mask R-CNN 18

3.1.1 Object localization 18

3.1.2 Object classification 19

3.1.3 Instance Segmentation 21

3.1.4 A Unified Architecture: Mask R-CNN 21

4 Implementation and Results **24**

4.1 Preparation of Data set 24

4.2 Mask R-CNN training 26

4.3 Results 27

5 Conclusion **30**

6 Appendix **31**

List of Figures

1 Convolution Operation - Vertical edge detection [1] 7
2 Padding Operation. 9
3 Striding Operation: (3 x 3) 10
4 Max-Pooling. 11
5 One Filter/ Kernel. 12
6 One Convolution layer. 13
7 SGD - Stochastic Gradient Descent[2]. 13
8 A simple network - Network structure. 15
9 A simple network - Hidden layers. 16
10 Goals of object detection. 17
11 Region Proposal Network 19
12 Performance comparison of plain CNN and ResNet[3]. 19
13 Basic structure of simple CNN(left) and ResNet(right). 20
14 Instance Segmentation: FCN 21
15 Architecture of Mask R-CNN 22
16 ROI align. 22
17 Annotated training data-set 24
18 Masks of the training data 25
19 Bounding boxes of the training data 26
20 Test results of Mask R-CNN: i)Original Image (Top) ii) Model Output (Bottom) 27
21 The spread of water as a function of height. 28
22 The Mask R-CNN framework for instance segmentation[4]. 31

Nomenclature

ANN : Artificial Neural Network

CNN : Convolutional Neural Network

$COCO$: Common Object in Context

FCN : Fully Convolutional Network

$MaskR-CNN$: Mask Region based Convolutional Neural Network

ML : Machine Learning

P : Padding

$ReLU$: Rectified Linear Unit

$ResNet-101$: Residual Network with 101 hidden layers

$ResNet$: Residual Network

RGB : Red, Green, Blue color image

ROI : Region of Interest Align

S : Striding

SGD : Stochastic Gradient Descent

SS : Selective Search algorithm

SS : Selective Search

1 Introduction

The emergence of artificial intelligence (AI) has unfolded solutions to problems that were previously considered very difficult, or sometimes impossible. The exponential growth of AI, mostly referred to as "Machine Learning" (ML), is attributed to two main factors, namely the availability of enormous digital data, and fast computing. Almost all fields ranging from science, engineering, medicine, and finance are looking towards AI for solution to their problems.

Machine learning comprises three specific fields:

- Supervised Learning: Builds a model from training data containing the samples of input and desired output.
- Unsupervised Learning: Takes only the input data and finds the hidden structure in the data or the similarity in the data points (e.g. clustering).
- Reinforcement Learning: Uses an agent that learns from experience over a period of time to take optimal decisions for the given task.

In chemical engineering and especially in process control, Machine Learning can help facilitate many complicated task. For example, supervised learning can be used to build models of complex systems from the input and the response of the systems. It can help understand the already available and very complicated process data. Similarly, it can also be used to build vision-based sensors for fire detection, level tracking and many more applications. On the other hand, Reinforcement learning, which is an experienced-based learning, provides optimal action to reach a specified goal, and can be used directly for control. This work demonstrates the use of Computer Vision as a sensor for water curtain detection of a sprinkler.

The computer vision aims to give the computer the ability to understand rather than just displaying the content of an image or a video. From the engineering perspective, it seeks to automate the human visual system. The functioning of human vision involves taking a picture through the eyes and processing it through the brain. We have already mastered the initial task of getting high quality images through a

camera. However, the processing of that image still remains a huge challenge. Understanding an image requires the ability to process it in different sizes, shape, angles and orientation, that the human brain has learned through millions of years of evolution. In computer vision, a very complex model (such as an Artificial Neural Network (ANN)) attempts to mimic the brain to perform this highly complex task.

Another challenge in computer vision is to train such a model. Human eyes feed around 1000 images every second, along with other senses, to train the brain. Similarly, the computer vision model requires an enormously large labelled image dataset to learn different objects. The present work demonstrates some of the best techniques available in computer vision for object detection to detect the sprinkler water curtain.

This work aims to use Computer Vision to detect a water curtain of sprinkler and find a measure that can be used as a feedback for controller to control the water distribution. This case presents a very good example where there are no conventional sensors available that can measure the process variable. With this objective in mind, the present work uses Convolutional Neural Network(CNN) to train a computer vision model to locate and detect a water curtain and quantify its distribution so that it can be used as a feedback.

2 Foundational Information

The primary task to understand the Computer Vision is to first understand how images are handled by computer. Any color image can be represented by the combination of three colors Red, Green, Blue (RGB). The computer processes any color image in terms of the pixel intensity of these three colors. Hence, any picture can be seen as [3 x h x w] size matrix of pixel intensity. Where, 'h' and 'w' are the height and width of the image. The key idea behind computer vision is to develop an algorithm that learns to relate this aforementioned matrix to the real world object. The first step in that process is an operation called "Convolution". The following few sections introduces Convolution operation along with few other simple operations that are helpful for feature extraction, followed by Convolutional Neural Network.

2.1 Convolution Operation

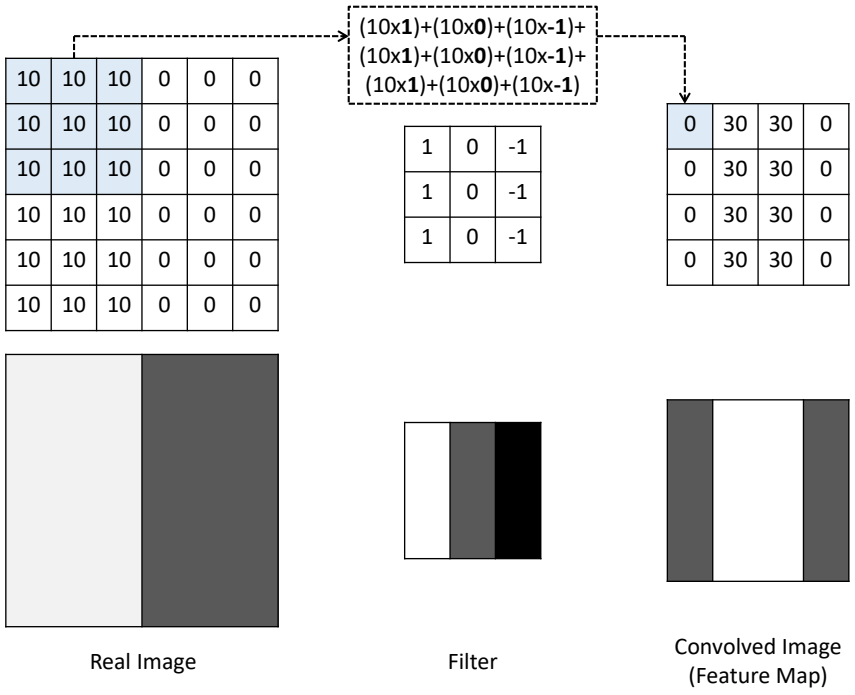

Figure 1: Convolution Operation - Vertical edge detection [1]

The convolution operation is explained here through an example of vertical edge

detection. Consider the image with a vertical edge in the center as shown in figure (1) (left). The matrix on top represents the pixel intensity of the image (0 = dark pixels and 10 = bright pixels). The aim here is to detect the vertical edge in the image and its location. Next, consider a small matrix in the middle called filter or kernel. The idea is to use this matrix to find vertical edge present in the image using convolution operation as follows:

1. Place the filter over the top left corner of the image (highlighted part).
2. Multiply each element of the filter matrix with the image matrix and add all multiples. Put the result in the top left corner of the convolved matrix.
3. Move the filter one pixel left on the image
4. Return to step 2.

After performing the above steps throughout the image the resultant matrix along with its corresponding image is shown in figure (1) (right). The resultant matrix is called Convolved Image or "Feature map". It can be seen that the feature map has a bright pixel in the middle suggesting the presence of a vertical edge and the location of the pixel indicates that it is in the middle of the image. At this point, a reasonable question would be that the width of the vertical edge (i.e. 2 pixels) in the feature map is very high compare to the original image. However, the image in the example is just 6x6, which is way small compare to a real life image (around 1080x1920). Therefore, for real image the width would still be significantly small.

The key idea to convey in the above example is that by changing the type of the filter, different edges (such as horizontal, 45°, 70°) and in turn different shapes in the image can be detected. This idea takes us to our next question of filter selection ("How do we decide which filter to use for a specific task ?"). Well, this is where "Convolutional Neural Network(CNN)" comes into the picture. In CNN, to learn the complicated images, instead of handpicking and feeding the filters to the computer, they are labelled as weights and learned by back propagation (explained in section

2.6). To put it in a nutshell, the Neural Network(NN) will learn the type of filter(Vertical, Horizontal or edges at 45 or 70°) that will best fit the desired outcome.

2.2 Padding

A closer look at the convolution operation, as shown in figure 1, reveals two shortcomings:

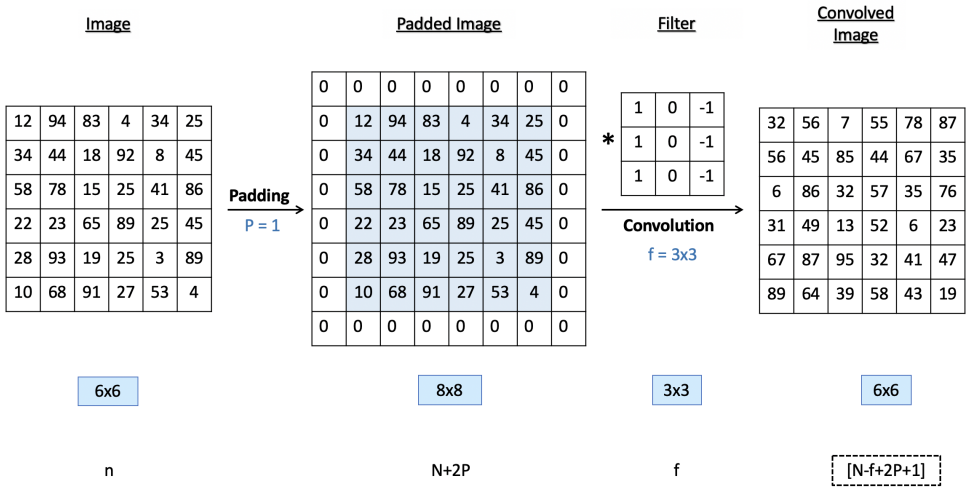

Figure 2: Padding Operation.

1. The size of the resultant image is smaller compare to the original image, by $(\frac{n-f}{2}+1)$ to be precise. Here, 'n' refers to the dimension of the original image and 'f' is the dimension of filter.

2. The pixels that are at the edge undergoes convolution operation only once while the pixels in the center are processed three times. As a result, the pixels in the center contribute more compare to the pixels that are at the edge.

Padding is a simple solution that addresses the both above mentioned problems by adding extra pixels at the border with zero values as shown in figure(2). Figure depicts that the size of the convolved image is now same as original image and the pixels at the border will now undergo two convolutions.

2.3 Striding

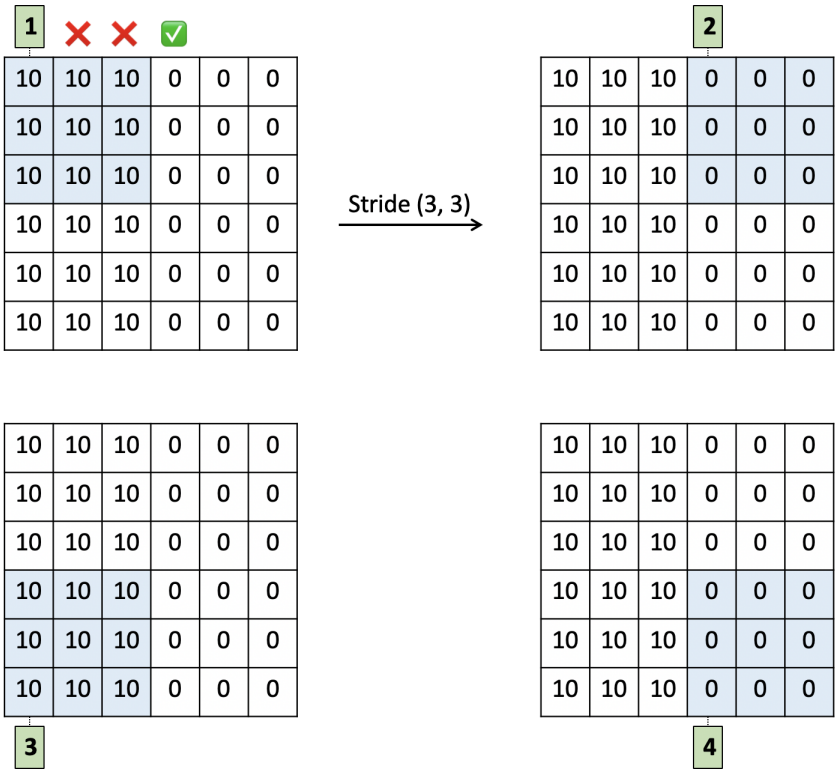

Figure 3: Striding Operation: (3 x 3)

Striding refers to a jump in number of pixel between two successive steps of convolution. Figure(3) shows the (3, 3) striding operation. It is clear from the figure that convolution operation is skipping 3 pixels in horizontal and vertical direction at every step. The default stride for normal convolution operation is (1, 1). Striding is primarily used to reduce the size of the feature map and most of the time along with Pooling (section 2.4). As we go deep in the network the reduction in the size of the feature map makes training faster as number of parameter (i.e. weights) decreases with feature maps.

2.4 Max - Pooling

The Max Polling is a widely used tool for feature extraction. As shown in figure (4)[Top], max-pool returns the maximum pixel value from the present operating region

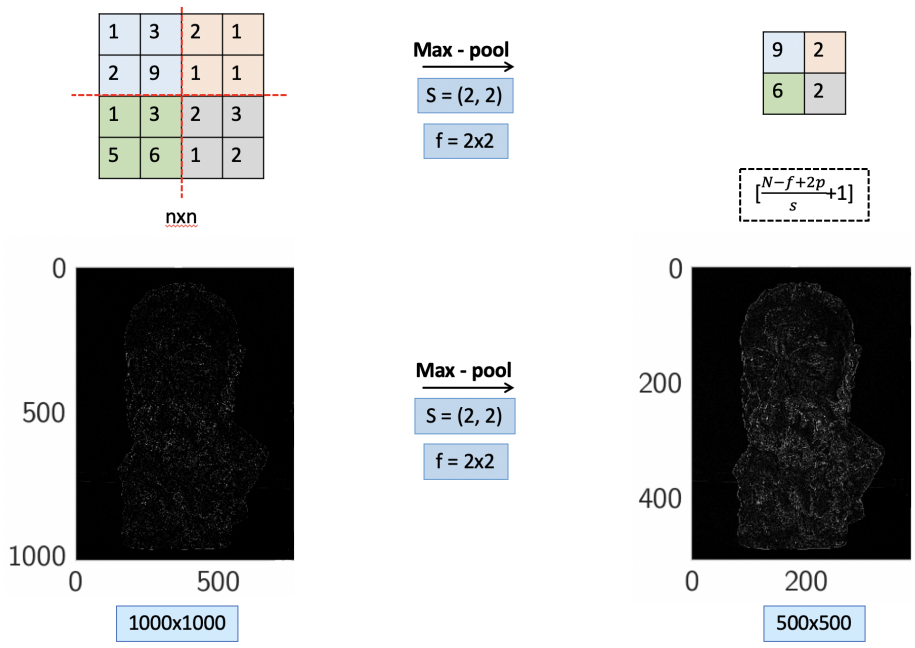

Figure 4: Max-Pooling.

of the filter. It is usually used with the stride equal to the size of the filter. In the figure, max-polling is applied to an image with filter [2 x 2] and stride (2, 2). In the CNN sense, the max-pooling when used after the convolution operation magnifies the extracted features and at the same time reduces the dimension of the feature map. For example, if the filter is trained to detect the vertical edges than max-pool would amplify the identified area and convey it to the next layer that is detecting some high level feature(such as shape, object). Figure (4)[Bottom] shows a simple example, where, the convolved image of [1000 x 1000], after max - pooling, reduces to just [500 x 500] with better visible vertical edges compare to the convolved image. In CNN, max-pool is usually used after the non-linearity (i.e. ReLU : explained in section 2.5).

2.5 CNN(Convolutional Neural Network)

The Convolutional Neural Network (CNN) is a supervised machine learning approach. When we say "Supervised", it means we provide the machine learning algorithm with the input and the outcome we expect from it. The deployed algorithm is tasked to map output from the input. As shown in figure (5), in CNN the RGB image is fed

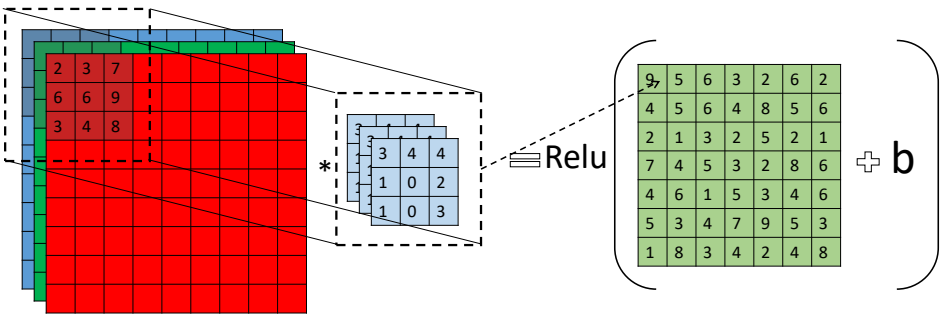

Figure 5: One Filter/ Kernel.

to the first layer of the network. This image matrix undergoes convolution operation and some additional bias (b). The final outcome passes through ReLU non-linearity and then passed over to the next layer of the network. The term ReLU stands for **Re**ctified **L**inear **U**nit (ReLU). ReLU outputs a maximum of 0(zero) and input. In other words, if it receives negative number as an input it converts it to zero otherwise passes it on as it is. The function plot of ReLU is shown in the figure (7)(Top-right). The ReLU unit in this case equips our CNN with a capability of mapping non-linear relationship between input image and target. On a side note, these features of ANN gives it an ability to represent any function[5].

The procedure mentioned above represents the process for one filter/kernal (or Neuron). Many such filters operating in parallel makes a single layer of CNN (shown in figure (6)). The initial layers learns to detect low level features such as shapes and boundaries, whereas the deeper layers combine the output of the initial layer to learn high level features such as body parts of the person. After several such layers, the network ends with the softmax layer that gives the final result (i.e. Final class: person, car, etc.). The important part, however, is to train such a model. The following section provides a quick overview about an optimization method called "Stochastic Gradient Descent" used for the training of CNN.

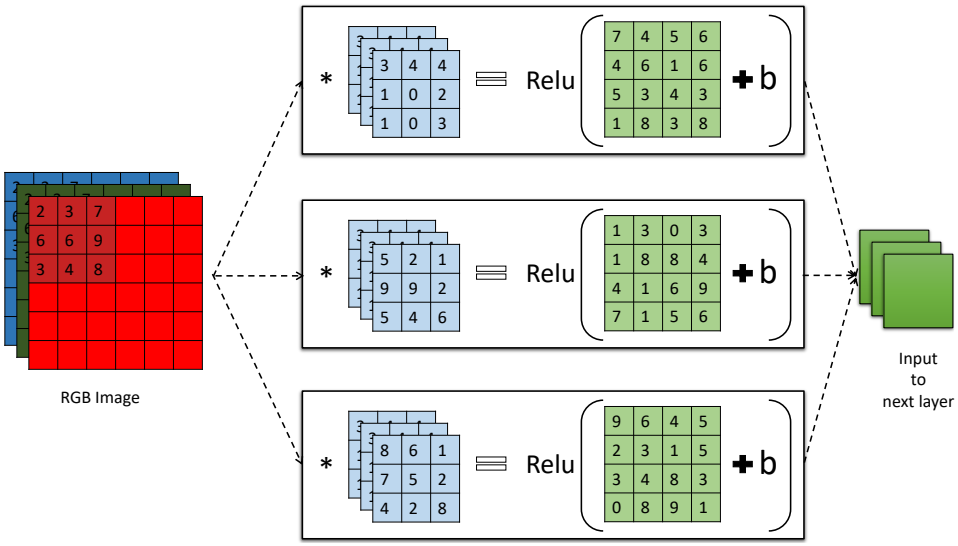

Figure 6: One Convolution layer.

2.6 SGD(Stochastic Gradient Descent)

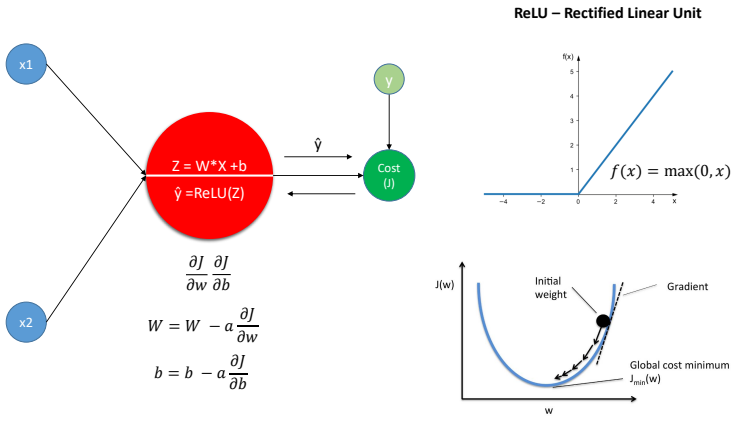

Figure 7: SGD - Stochastic Gradient Descent[2].

Stochastic Gradient Descent is an iterative algorithm for the optimization of an objective function. SGD starts the training with some random weights of the filters and then passes the image available for training through entire CNN. As expected, the results predicted by this network ($\hat{y}$) would be different than the true results (y). Therefore, the square of the difference between these two results $(y - \hat{y})^2$ is termed as an objective function (J). This quadratic form insures that objective function is

convex and gradient descent will reach minimum. The goal of SGD is to adjust down the weights of all the filters in a way that minimizes the objective function as much as possible. SGD takes following steps to minimize the objective function.

1. Initialize the CNN with random filter weights and evaluate the objective function.

$$W = [w_{11}, w_{12}, ...w_{nk}] \tag{1}$$

2. Calculate the gradient (slope) of the objective function with respect to each weight of the filter($\frac{\partial J}{\partial W}$). In other words, evaluate the rate of change of the objective function with respect to every weight in the filter.

$$\frac{\partial J}{\partial w} = \begin{bmatrix} \frac{\partial J}{\partial w_{11}} & . & . & . & . & \frac{\partial J}{\partial w_{1k}} \\ \frac{\partial J}{\partial w_{21}} & . & . & . & . & \frac{\partial J}{\partial w_{1k}} \\ . & . & . & . & . & . \\ \frac{\partial J}{\partial w_{n1}} & . & . & . & . & \frac{\partial J}{\partial w_{nk}} \end{bmatrix} \tag{2}$$

   Where, n = Number of hidden layers and k = number of neurons in each layer.

3. Update the weights of all the filters as follows:

$$W = W - \alpha \frac{\partial J}{\partial w} \tag{3}$$

4. Calculate the objective function with the updated weights.

5. Repeat steps 2 to 4 until the objective function is close to zero.

Figure (7) (left) summarizes the above mentioned steps. The intuition behind the algorithm is that initialization with random weights lends us to a random point on the surface of the cost function as shown in figure 7(Bottom-right). At this point, taking a gradient of the cost function with respect to all weights provides us the direction in which moving the weight would lead to a reduced cost function (shown with arrows in figure (7)). As we know, any function reaches its minimum value when the slope is zero. Hence, SGD attempts to find the minimum of the objective function by moving

the weights of the filter step by step in the direction of the gradient. The size of the step is controlled by a parameter "α" known as a learning rate.

Training a model most often involves dealing with a very big dataset. Evaluating an objective function at each step for all the training data, therefore, is computationally very expansive. To overcome this problem, SGD randomly chooses only one sample form the dataset at every step and performs a gradient descent update. The point when SGD finishes one sweep across the entire dataset it refers to as an "epoch". Since, the selection of the data sample is random in SGD, it is called "Stochastic" Gradient Descent. When the gradient descent update is performed for a small batch of training data instead of just one sample it is known as "Batch Gradient Descent".

2.7 A Simple Network

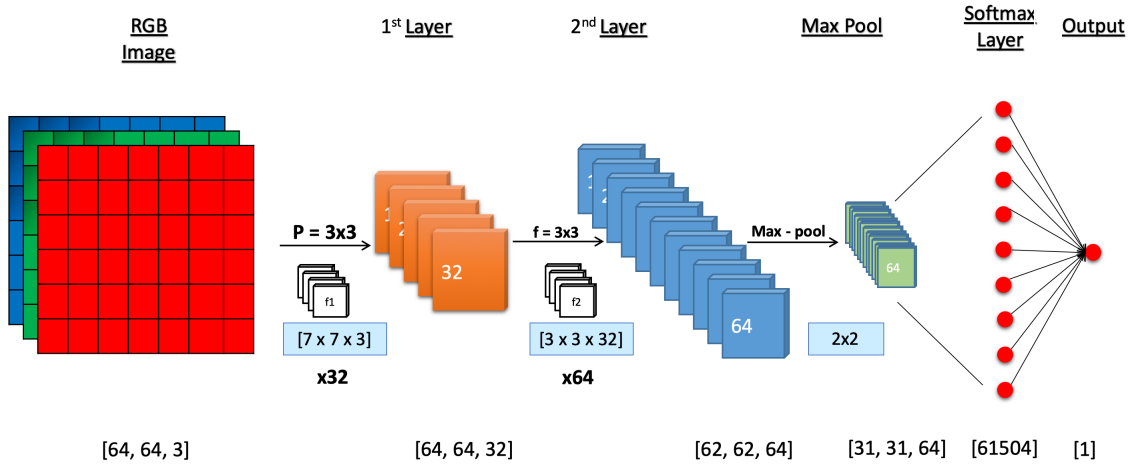

Figure 8: A simple network - Network structure.

This section gives a brief overview of a simple two-layer CNN. Consider a network shown in figure (8). The network has two hidden layers followed by a Max pool and a softmax layer. The RGB image is first padded [3 x 3] and then fed to the first layer with 32 kernel/filter of size [7 x 7 x 3]. The convolution operation in the first layer results in [62, 62, 64] dimensional output which is then passed to the second layer. The second layer has [3 x 3 x 32] sized 64 filters. The important point to note here is

that the last dimension of the filter should match the dimension of the output of the previous layer, which in this case is 32. The [62, 62, 64] dimensional output of the second layer is max-pooled ([2 x 2]) with [2 x 2] padding and the resultant matrix is flattened to a softmax layer. To further illustrate the network better, the figure (9) gives the image of the hidden layers. Notice how the size of the feature map reduces after the Max-pooling.

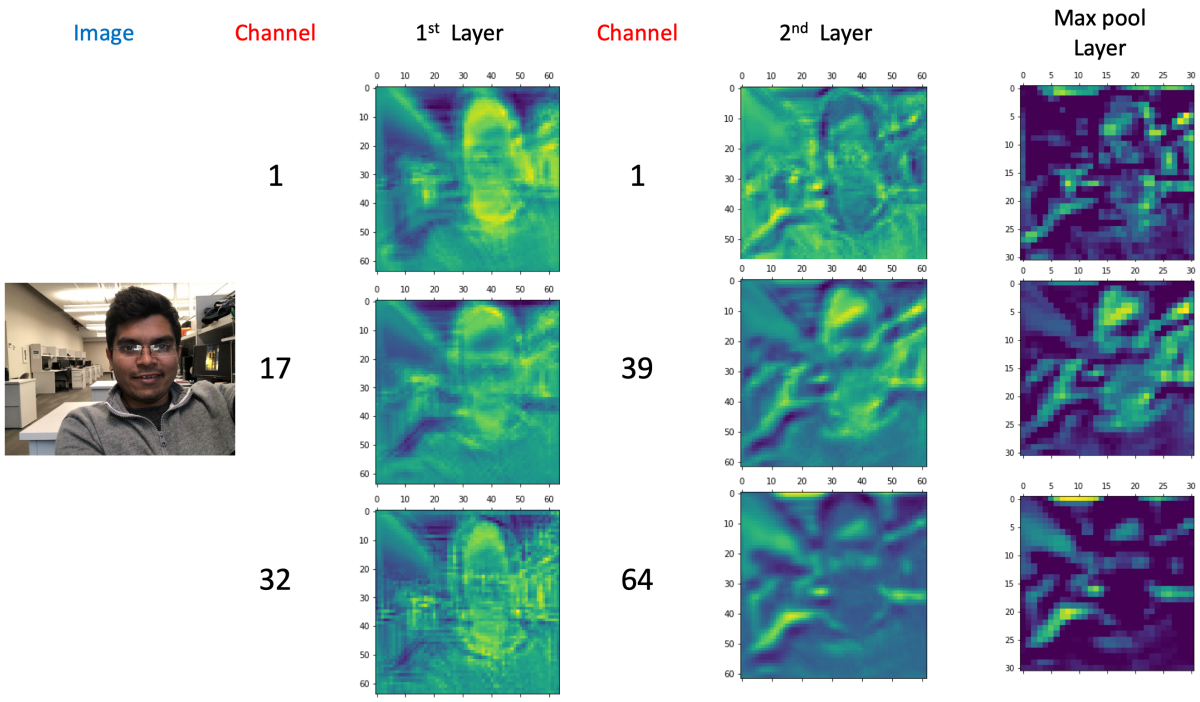

Figure 9: A simple network - Hidden layers.

3 Object detection and segmentation

The task of any object detection algorithm is to address three basic questions. Where the object is located (i.e. center, top, bottom, etc.), what is the class of that object (i.e. person, car, etc.) and where is the boundary of the classified object in the image. These three questions divides the task of object detection into the following sub-tasks:

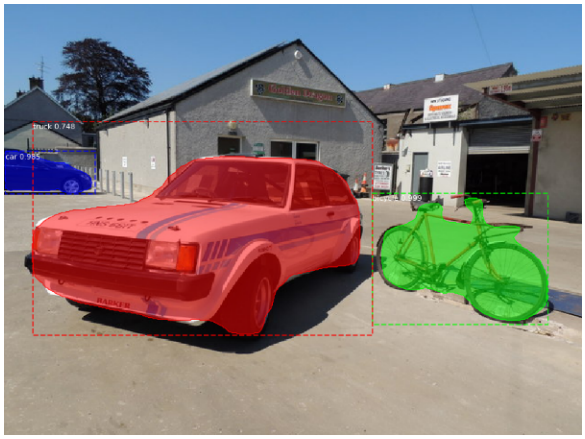

Figure 10: Goals of object detection.

1. Object localization : Locate the object inside an image and draw a bounding box around it.

2. Object classification : Identify the detected object by classifying it to an appropriate class.

3. Instance Segmentation : Identifying the pixel level outlines and to create a mask on top of the object as shown in figure (10).

This work uses a structure that addresses all the above mentioned questions, called the Mask R-CNN[4], proposed by He et al. for water detection and segmentation. The following sections provides a detailed overview of the overall structure and different modules of the Mask R-CNN.

3.1 Mask R-CNN

Mask R-CNN stands for Mask Region based Convolutional Neural Network. It is a state of the art framework and the latest version of the R-CNN family[6, 7, 8](Girshick et al.). The Mask R-CNN takes image as an input, detects objects in the image, draws a bounding box around it and creates a mask. It has a separate module to deal with each three aforementioned tasks of object detection and segmentation. The forthcoming sections elaborates the working of each module and demonstrates how they can be put to gather in a unified framework with the highest efficiency.

3.1.1 Object localization

The goal of object localization, given an input image, is to look for all the possible objects present in the image and give the co-ordinates of the center, height and width of the object as an output. One simple way to do that is to use an image processing based technique that uses predefined filters to detect edges and proposes the objects based on that. Selective Search(SS) is an example of such an algorithm.It proposes around 2000 regions per image with a possibility of containing an object. However, processing 2000 regions for each image is computationally very expensive for classification network and fairly time consuming process. As a result, the localization module ends up being a bottleneck for the entire process.

The Mask R-CNN, on the other hand, utilizes a CNN called "Region Proposal Network(RPN)" to propose regions. The RPN, as shown in figure (11), is a regular CNN specifically trained to propose regions with an object, and an objectness score. Objectness score is a measure of the proposed regions containing an object (vs background). RPN, compare to image processing based methods, stands out due to the fact that instead of operating on an image directly, it uses convolved feature maps of the image as shown in figure (11). In other words, it gets extracted features(such as shapes, edges..etc) directly as an input making the task much simpler. In addition, it doesn't proposes as many as 2000 regions, resulting in significant improvement of classification computation.

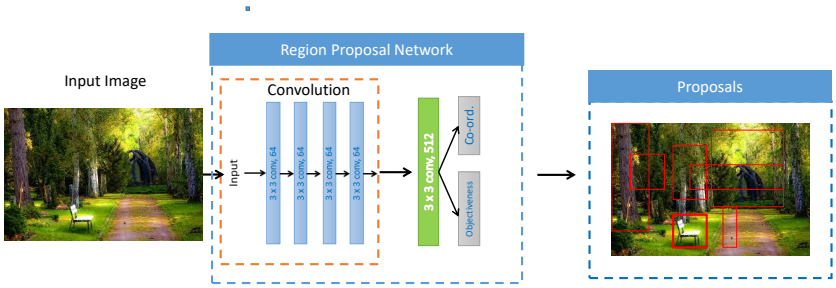

Figure 11: Region Proposal Network

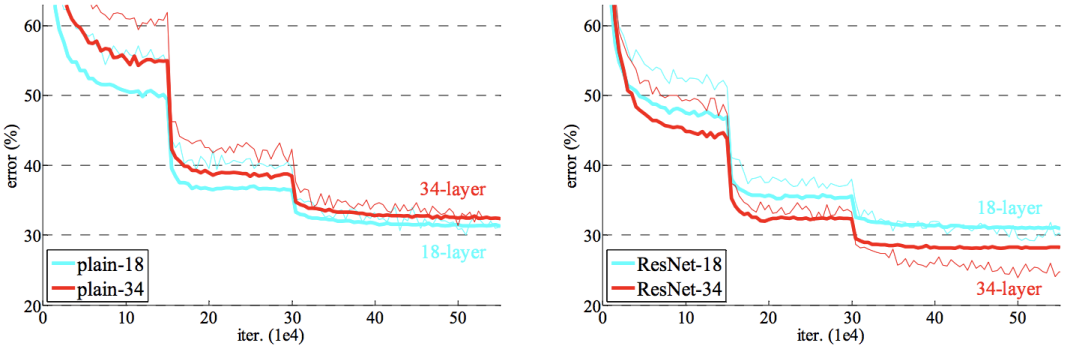

Figure 12: Performance comparison of plain CNN and ResNet[3].

3.1.2 Object classification

Object classification involves learning about the small features in the image to classify it to an appropriate class. As mentioned earlier, each layer in the CNN is responsible for extracting new features. Hence, the basic intuition is that as the number of layer increases the network should become more capable of extracting even small features and its performance should improve(i.e. get batter in classifying different objects). However, as shown in figure (12) (left), in practice the error(training and testing) starts increasing after a certain level as number of layer increases (the performance of 34 layer network is poor than 18 layer network). It is expected that deep CNN should at least match the performance of its shallow counterpart if not any better. On the

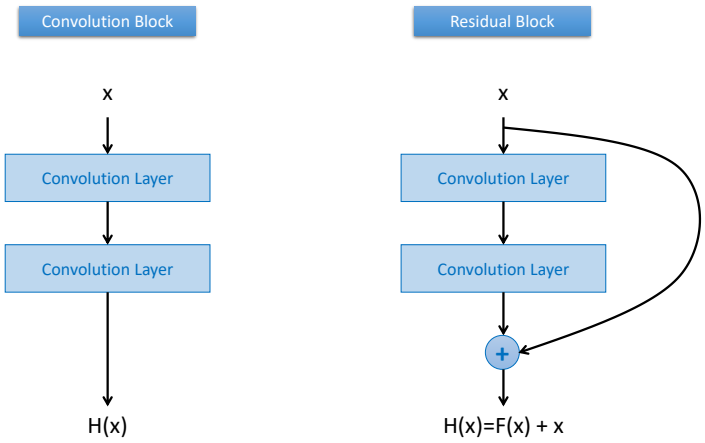

Figure 13: Basic structure of simple CNN(left) and ResNet(right).

contrary, since, the performance is deteriorating upon the addition of the extra layers, a simple guess is that extra layers are not capable of learning the identity mapping. In other words, keeping the learning aside, the deeper layers are unable to even carry forward the results from the earlier layers. This guess lead to a simple solution called 'Residual Network' (ResNet)[9].

In ResNet, the input is passed to a block of CNN and is also added to the output(short connection). Figure 13 shows the comparison of regular CNN and ResNet. The intuition behind this solution was that it may be easier for deeper layers to learn nothing (weights = 0) compare to the identity mapping. Therefore, if there is nothing more to learn the input will simply bypass the layer through the side channel. The results from ResNet are shown in figure 12 (right).It is clear from the figure that even very deep ResNet performs better than the previous shallow networks. In practice, 152 layer deep ResNet(ResNet - 152) trained on ImageNet dataset achieved 3.57 error rate in object classification. It was concluded that ResNet - 152 outperformed even human vision(error rate = 5) at object classification. Because of this excellent performance, the Mask R-CNN leverages ResNet-101 (ResNet with 101 layers) for object classification.

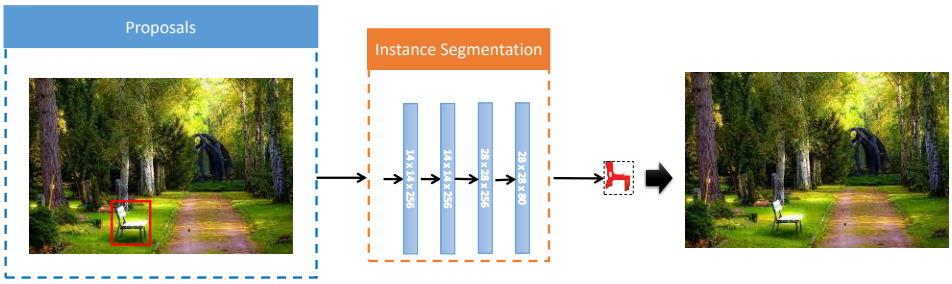

Figure 14: Instance Segmentation: FCN .

3.1.3 Instance Segmentation

Instance segmentation is responsible for generating a pixel-perfect binary mask for each object. As shown in the figure (14), Mask R-CNN uses Fully Convolutional Network(FCN) for this purpose. The positive object proposal (generated by RPN) in the image, also known as 'Region of Interest(ROI)', are passed to two branches working in parallel. One branch is a classification network predicting the object class while the other branch is made up of Fully Convolutional Layers predicting the mask of the object. The FCN is trained to create k x n x n size binary mask(1 for the pixel containing object and 0 for background) for each ROI. Where k is the number of classes and n stands for the number of pixels(height x width) in the image. For a batter intuition of Instance Segmentation kindly refer to figure -(22) in Appendix.

3.1.4 A Unified Architecture: Mask R-CNN

An important point worth noting here is that in all three modules image, first, has to pass through a series of convolution layers. On the other hand, these structures are trained to perform completely different tasks (i.e.Localization, Classification, Segmentation). This could mean that in the initial stages all three modules learn the common features in the image(i.e. shape, edges..etc.) and in the later stages they

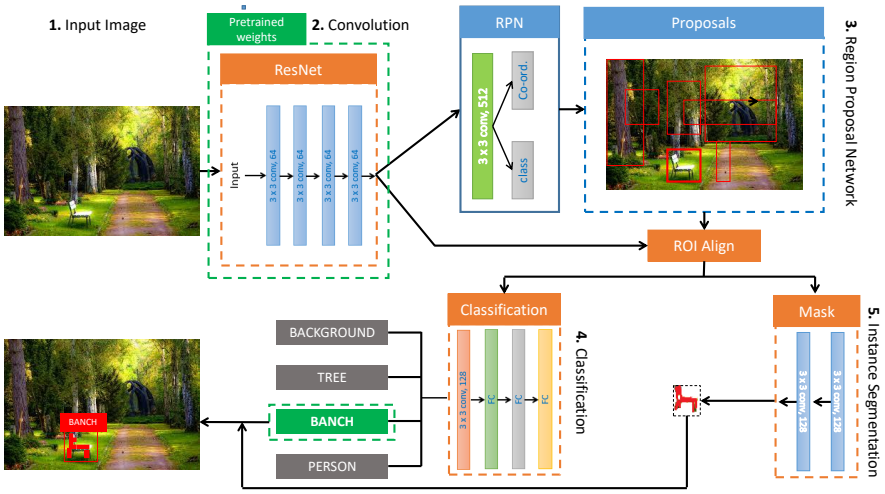

Figure 15: Architecture of Mask R-CNN .

learn task-specific features. The Mask R-CNN leverages this fact to combine all three modules into a unified framework in a most computationally efficient way.

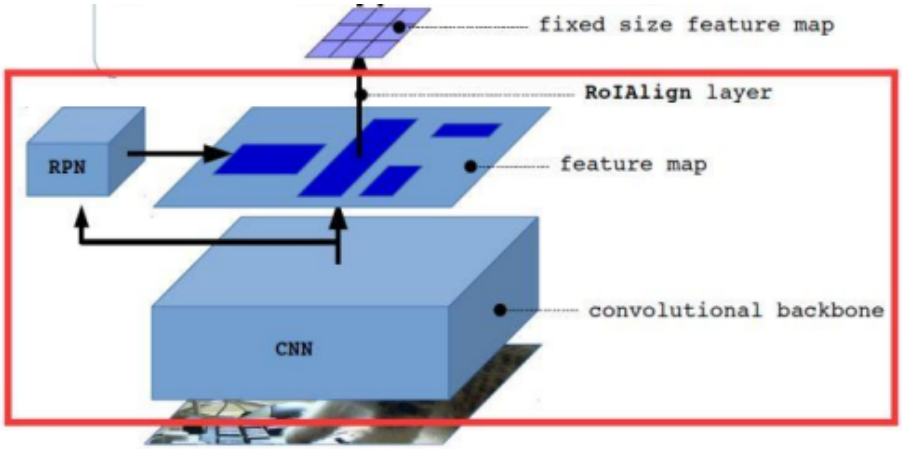

Figure 16: ROI align.

As shown in figure 15, the image first passes through a series of convolutions(i.e. ResNet-101) called "backbone" to generate feature maps. The RPN uses these feature maps as an input and proposes regions with possible objects. These proposals are then passed on to a module called 'ROI Align'. The task of ROI Align is to take the co-ordinates of regions with objects proposed by RPN and map it to the feature maps of the image in the backbone (see figure (16)). In other words, ROI align

takes co-ordinates of the bounding box (center - x, y, height and width), find the corresponding feature maps on back bone and compresses it to a fixed size to pass it to the next stage. These feature maps are then passed on to the two parallel networks for classification and Mask generation. Finally, the results from both units are combined to give the final result.

4 Implementation and Results

The implementation of Mask R-CNN for water curtain detection is done in python3 using Keras and Tensorflow[10]. The implementation can be devided in to three steps as follows:

1. Preparation of Data set
2. Mask R-CNN Training
3. Testing the Model: Results

4.1 Preparation of Data set

Figure 17: Annotated training data-set

The training of the Mask R-CNN, as mentioned earlier, is a supervised machine learning and demands the training data containing the expected output. Therefore, in the present case, the training requires following three point of reference:

- Classification: Labelled Images
- Localization: Bounding boxes around the object

- Instance Segmentation: Pixel perfect masks covering the object

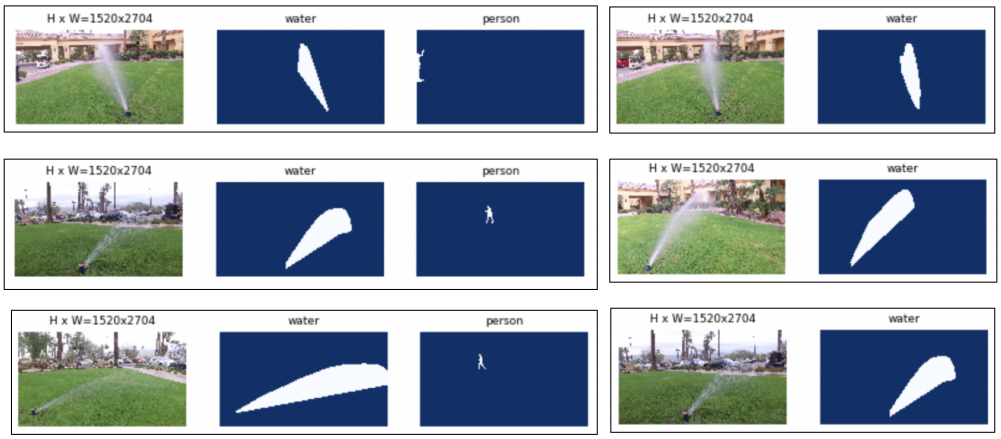

Figure 18: Masks of the training data

This work uses VGG-Image annotator-1.0.6[11] for annotation. The annotation involves labeling and marking the boundary of the object of interest in the image. The annotation tool gives the annotations in .json file containing the x and y co-ordinates of the pin pointed annotated pixels. Figure (17) depicts some examples of the annotated training data. For the present case, the training data has two classes: i) Water ii)Person. These co-ordinates, then, can be used to generate the mask and the bounding box. Connecting the co-ordinates can give the outline of the object and the entire region inside this outline can be used as a mask. Figure (18) shows such masks that are used for training. Similarly, the (highest x, highest y)and (lowest x, lowest y) co-ordinates gives the top left and bottom right point of the object and connecting the horizontal and vertical line passing through these points, as shown in figure (19) gives the bounding box surrounding the object.

It is important to have a verity of background in the training data in order to generalize it well. Here, generalization refers to the capability of the model to detect the object of interest in the verity of situation. For the present case the training data consist 106 annotated images. For very accurate results, the computer vision model usually requires a huge annotated image dataset (100 – 200 k). However, this study uses the transfer learning (explained in next section) to reduce the size of training data without compromising the quality of the results.

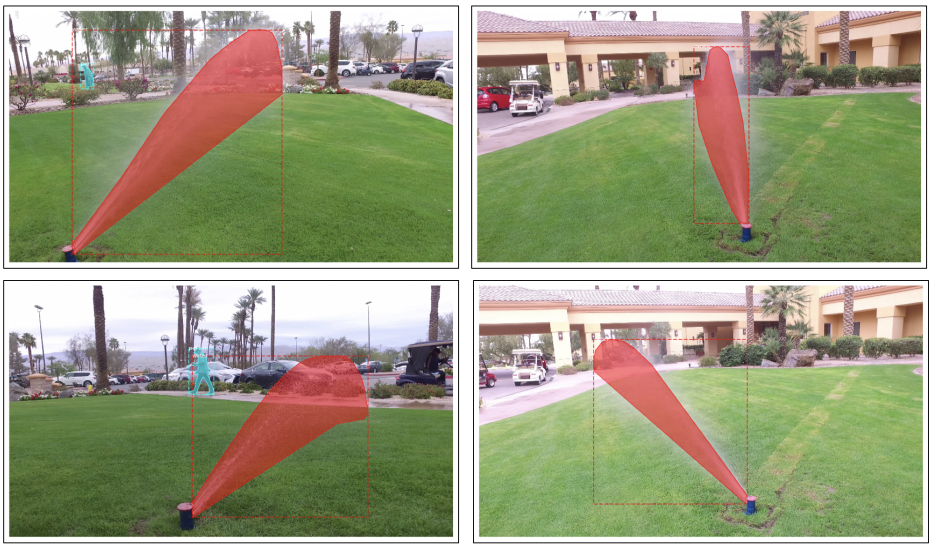

Figure 19: Bounding boxes of the training data

4.2 Mask R-CNN training

The Mask R-CNN is trained using ResNet-101 as a backbone. The model is trained by minimizing the combination of three loss function as follows.

$$L_{ROI} = L_{cls} + L_{box} + L_{mask} \tag{4}$$

Where, L_{cls} = Classification loss, L_box = Bounding box loss (loss for the prediction of the bounding box co-ordinates) and L_{mask} = Mask loss. The training was carried out for 60 epochs.

In this work, the model was trained by leveraging the concept of "Transfer Learning". The Mask R-CNN has a very large structure and training such a network requires a significant amount of computation and training data. However, the initial layer of all CNN, irrespective of the task, learns to detect the basic features(such as shapes, edges, background, etc.). Therefore, it makes much more sense to use pre-trained weights, if available, of a previously trained model for initial layers and to train only the last few layers specific to the task. This idea serves as a key concept for transfer learning which enables us to train a very huge network with comparatively less computations and training data. This work uses pre-trained COCO dataset weights that were trained on 120K images. The model used in this work for water

detection is trained on Dell Precision 7920 XL Rack Server with two Intel Xeon Gold 6138 2.0G CPUs (each with 20 cores, 40 threads), 128GB memory, and 2 NVIDIA Quadro P2000 GPUs.

4.3 Results

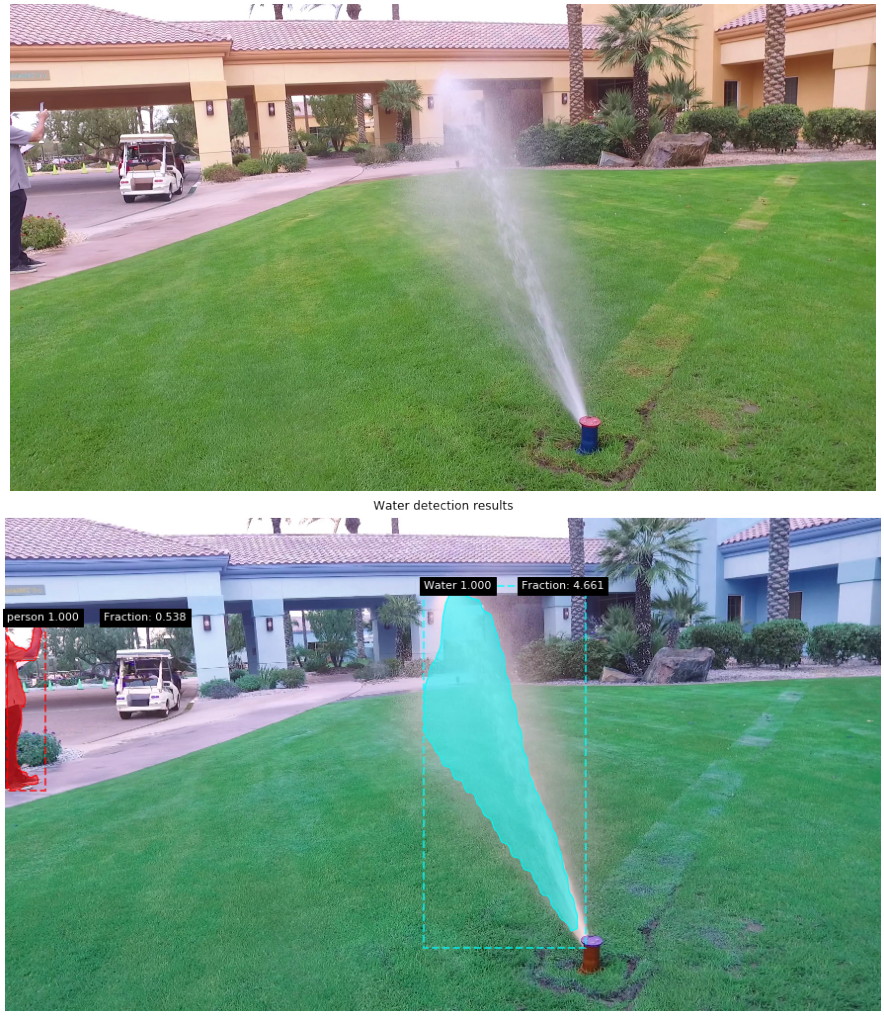

Figure 20: Test results of Mask R-CNN: i)Original Image (Top) ii) Model Output (Bottom)

The output of the Mask R-CNN model trained for water detection is visualized in Figure:(20) which shows the original image (Figure 20(i)) and the result predicted by the model (Figure 20(ii)). It gives the labeled bounding box of each object along with

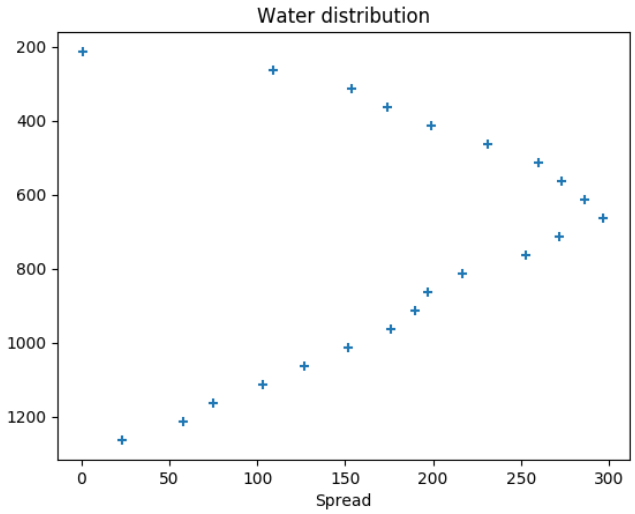

Figure 21: The spread of water as a function of height.

the binary mask separating all objects from each other. The pixel perfect mask is of significant importance here as it is further utilized to infer some useful results. The term "fraction" in the output image refers to the fraction of the image frame occupied by that specific object. The goal is to control the water distribution by adjusting the flow of the water. Therefore, this fraction could be one possible way to quantify the water distribution in the image and can serve as possible feedback for control. It could also help control the water distribution in the presence of the external source of disturbance such as wind. In addition, the mask is further extended to calculate the spread of the water curtain as a function of height (shown in figure (21)). In the figure, the X axis refers to the width of the water curtain in number of pixels, while the Y axis represents the vertical pixel coordinates. The model is trained to detect the person so that if the person comes closer to the sprinkler it can be immediately stopped to prevent the person from getting wet. This can be achieved by putting a threshold on the variable "fraction" for class "person". Here, the fraction can be used as a reference to infer the distance between the person and sprinkler. The fraction going beyond the threshold indicates the presence of the person nearby.

Training the model for a variety of datasets indicated that the masks are more accurate when the model is tested on the image with a background that is similar to

the training data. One possible explanation for this result is that during some light conditions, the detection of the air - water interface is very difficult to track and the similar environment allows the model to discriminate even very fine details. Since the accuracy of the semantic segmentation is essential in this case, training an individual model for a variety of locations, instead of single generalized model, would provide more fruitful results.

5 Conclusion

The study demonstrates the use of supervised machine learning for water curtain detection. A Convolutional Neural Network architecture named Mask R-CNN is deployed to train an object detection model. The model is trained to perform three main tasks i) object localization, ii) object classification, and iii) instance segmentation. Object localization is done by RPN giving fast computation, eliminating the bottleneck image processing based algorithm (i.e. SS). ResNet-101, a hundred and one layer-deep CNN, is used for classification, and FCN to propose the masks. An annotated data set was prepared for model training. The use of transfer learning resulted in a significant reduction of computation and the training data without affecting the quality of the results. The trained model was successfully able to detect the water curtain and person, giving the bounding box and binary mask. The mask is further used to calculate the fraction and water distribution with respect to height. It was observed that training an individual model for different regions gives better prediction compared to a single generalized model.

6 Appendix

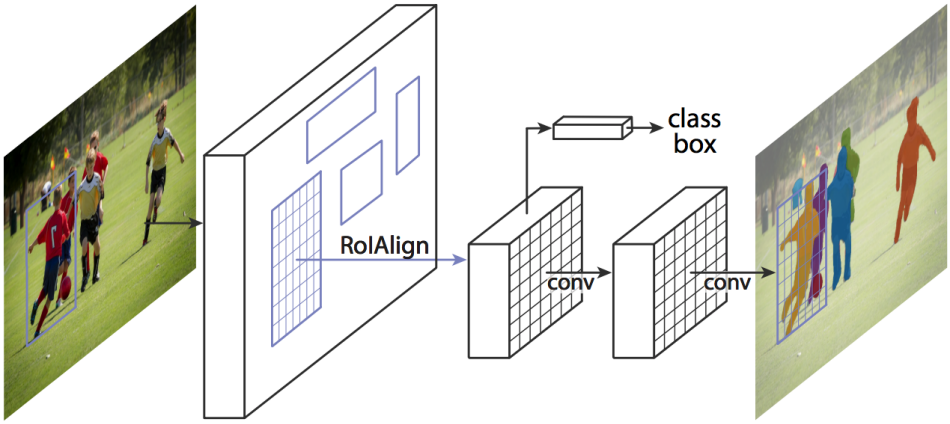

Figure 22: The Mask R-CNN framework for instance segmentation[4].

References

[1] Andrew Ng. *Deep Learning Specialization.* Coursera, 2016. https://www.coursera.org/specializations/deep-learninghowItWorks.

[2] Sebastian Raschka. Graph of stochastic gradient descent optmization. http://rasbt.github.io/mlxtend/user$_g$*uide/general*$_c$*oncepts/gradient* − *optimization/*.

[3] Resnet performance comparision. http://ethen8181.github.io/machine-learning/keras/resnet$_c$*am/resnet*$_c$*am.html*.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016. http://www.deeplearningbook.org.

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[7] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[10] Waleed Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. https://github.com/matterport/Mask$_R$*CNN*, 2017.

[11] A. Dutta, A. Gupta, and A. Zissermann. VGG image annotator (VIA). http://www.robots.ox.ac.uk/ vgg/software/via/, 2016. Version: 1.0.6.