Vision-based Obstacle Detection and Autonomous Navigation of Unmanned Ground Vehicles

Project Report

Master of Science

by

Sujeet Kumar Singh

Roll No: 214363009

Under the guidance of

Prof. S.K. Dviweedy & Dr. Prithwijit Guha

Nov.,2022

Department of Electronics and Electrical & Mechanical Engineering

INDIAN INSTITUTE OF TECHNOLOGY

GUWAHATI - 781 039

Abstract

In the recent period, the application of unmanned ground vehicles (UGVs) have increased rapidly in every field. Due to fast growth in artificial intelligence technology, UGVs are more capable of doing tough tasks very efficiently and accurately without human intervention. In the recent, period Covid19 pandemic and Russia Ukraine war UGVs have been used extensively.The application of UGVs broadly increased in every field whether it is an agricultural field, medical field, defense field, or space field. Due to the advent of deep learning, works that are not solvable can be easily solved by deep learning. Many computer vision tasks with the help of deep learning can be handled effectively and efficiently, these are possible because of the availability of large datasets and the hardware capability of computers. Obstacle detection, classification and predicting depth of the obstacles is one of the most important and difficult tasks for an unmanned ground vehicle. Vision-based approaches are popular for this task because it is cost-effective and makes human-like perception. Although there are many cues to detect the depth of the object, here depth is calculated with the help of monocular cues with single camera model. So many artificial neural network techniques developed in the past few years to mimic the human visual system. Convolutional neural networks (R-CNN family, YOLO family, Retina Net etc.) are widely used in object detection.

Contents

List of Figures

Chapter 1

Introduction

1.1 Unmanned Ground Vehicles

Unmanned Ground Vehicle (UGV) Unmanned ground vehicles are robotic systems that operate on the ground without the onboard human operator. Nowadays UGVs are widely used for civilian and military purposes, particularly in environments that are harmful or unpleasant to humans. Based on its application, unmanned ground vehicles will generally include the following components: platform, sensors, control systems, guidance interface, communication links, and systems integration features.

Platform The platform can be based on an all-terrain vehicle design and includes the locomotive apparatus, sensors, and power source. Wheels, tracks, and legs are the common forms of locomotion. wheels are power efficient and allow the highest speed on flat ground but are not suitable for traversing off-road and uneven terrain, as they get stuck or sink a low contact area and thus higher pressure. The track is good for rugged terrain but less efficient and low speed due to mechanical complexity and high vibration. Legs types can be used in a variety of terrain but it has low speed and require complex control and stability hardware.

Control systems UGVs are controlled remotely or autonomously or by both with the help of artificial intelligence technology.

Guidance Interface Depending on the type of control system, the interface between machine and human operator can include a joystick, computer programs, or voice command.

Sensors Sensors are used for navigation and to perceive the environment. Some commonly deployed sensors on UGVs are lasers, ultrasound, RADAR, camera, odometer, gyroscope, inclinometer, etc. Sensors are crucial for efficient work UGVs. Sensors can be classified into two types

- **Exteroceptive Sensors (ESs)**
- **Proprioceptive Sensors (PSs)**

Exteroceptive sensors are used to perceive external environmental information, e.g., LiDAR, millimeters-wave, ultrasonic, and cameras. While Proprioceptive sensors are

used for real-time information about the platform itself, e.g., vehicle speed, acceleration, altitude angle, wheel speed, and position to ensure real-time state estimation of UGV itself. Common Proprioceptive Sensors are GNSS and IMU.

Light Detection and Ranging (LiDAR) LiDAR is used for object position, orientation, and velocity information by transmitting and receiving laser beams and calculating time differences. The collected data type is a series of 3D point information called a point cloud, more specifically the coordinate of the object to the radar coordinate system and echoes intensity. Lidar is mainly used in SLAM, point cloud matching and localization, object detection, trajectory prediction, and tracking. Lidar cannot collect the color and texture information of the target.

Radio Detection and Ranging (RADAR) The working principle of RADAR is similar to LiDAR, the key difference here we used radio waves to detect the target. Compared with Lidar, Radar has a longer detection range, smaller size, lower price, and is not easily affected by light and weather conditions. Ultrasonic Ultrasonic detects objects by emitting sound waves and is mainly used in the field of ships. Ultrasonic is small in size, low in cost, and not affected by weather and light conditions, but its detection distance is short, the accuracy is low, it is prone to noise, and it is also easy to interfere with other equipment.

1.2 Computer Vision

Computer vision is the area of study in which computers are empowered to visualize, recognize and process what they see in a similar way as that of humans [24]. The main aim of computer vision is to generate relevant information from image and video data in order to extract something about the world [25][26]. It can be classified as a sub-field of artificial intelligence and machine learning. Applications of computer vision include image classification, visual detection, 3D scene reconstruction from 2D images, image retrieval, augmented reality, machine vision and traffic automation

1.2.1 Monocular Camera

Monocular cameras store environmental information in the form of pixels by converting optical signals into electrical signals. Advantages monocular cameras as compared to Lidar, Radar, ultrasonic can generate high-resolution images having environmental color and texture information, and also low cost. To obtain depth information from monocular camera is very tedious task, it is highly susceptible to illumination conditions and weather conditions, and the high computing power required for high-quality images challenges the real-time performance of the algorithm.

1.2.2 Stereo Camera

The working principle of a stereo camera is similar to a mono camera, compared to a mono camera stereo camera is equipped with an additional lens at a symmetrical position, the depth information and movement of the environment can be captured by taking two pictures at the same time through multiple viewing angles information. However, it is also susceptible to weather conditions field view is narrow, and more computing power is required.

1.2.3 Infrared Camera

Infrared cameras collect environmental information by receiving signals of infrared radiation from objects. The infrared camera is sensitive to the wavelength of 0.15 m to 15 m in practical applications, a corresponding infrared camera is selected according to the wavelength of different detection targets. Computer vision is the subfield of computer science, artificial intelligence, and machine learning, it mimics the human vision system, because of deep learning and neural networks computer vision accuracy increases significantly.

1.2.4 Simultaneous Localization and Mapping (SLAM)

SLAM tries to construct a map of an unknown environment, either in 2D or 3D, while also employing the same map for navigation. This paper focuses on visual SLAM, which uses monocular cameras to examine the environment,However, in practise, a wide range of sensors can be used.By deploying stereo-cameras in the SLAM system, a real metric scaled 3D map is generated because depth can be estimated by triangulation.The biggest disadvantage of stereo cameras is the sophisticated and expensive hardware setup, as well as the rigorous calibration [43] that is required.The arrangement can be simplified by employing monocular cameras instead of binocular cameras. be made easier and less expensive When only a single monocular camera is employed, this types of SLAM system is frequently referred to as mono-SLAM [12]. There are two fundamental concerns in mono-SLAM. The first point to mention is that the 3D map is originally constructed on an unknown scale factor, implying that the link between The map's points are correct, but their link to the real world is uncertain. The second concern is that the scale may shift with time, resulting in a geometrically incorrect map [40], a phenomenon known as scale drift.Small-baseline stereo matching between consecutive frames is one method for addressing these difficulties. Assuming that the camera moves over time, neighbouring frames can be viewed as stereo pictures, and the baseline can be obtained during the initial phase. However, because pure camera rotations provide no baseline and the metric scale is only obtained once, the system is still susceptible to scale drift.Another option to addressing the aforementioned concerns is to introduce an external scale reference, i.e. prior knowledge of a set of pre-defined objects of known size, a technique that fails in the

absence of these items. [41].Finding a technique to obtain depth maps of the images would be advantageous since they could be used to update the inconsistent map and offer the real metric scale, as in the case of the stereo-camera. The task of predicting depth maps from monocular pictures can be viewed as a sophisticated high-dimensional function, with an image as input and a depth map as output.The human brain has an exceptional ability to comprehend visual sceneries and can solve this function by utilising picture monocular cues such as vanishing points, lighting, shading, perspective, and relative scale [19].The monocular cues, learning experiences, and knowledge about the environment, as well as the size of recognised objects, are processed in the brain, resulting in a 3D impression of the scene.This is a simple activity for humans; even persons with monocular vision (just one eye) can complete it. However, developing a system that can estimate depth using monocular inputs is a difficult problem [29]. Artificial neural networks [20] are deep learning algorithms that replicate the brain and its ability to learn complicated tasks by modelling the neural system's structure.These methods let computers to learn from data without being explicitly written, and they have proven beneficial for solving issues that hard-coded algorithms cannot solve [20].Neural networks have been used to solve a variety of difficult computer vision problems in recent years, including object recognition and segmentation, image classification, and image reconstruction. The results are promising, with networks outperforming humans in some activities [21]. Neural networks operate by feeding input data to the network and returning some output data. The internal weights of the network, often known as its parameters, dictate how the input is transferred to the output. The process of determining the appropriate weights is referred to as network training or optimization. Supervised neural networks are trained on inputs with corresponding desired outputs, or ground truths, which teach the network how it should act. The goal is for the network to learn to return accurate outputs, or predictions, given an input it has not seen before

1.2.5 Motivation

Having access to depth maps in SLAM-systems has proven advantageous [41, 43]. because they provide the proper metric scale and can also be used to rebuild geometrically consistent maps, as well as in other applications such as Accurate depth estimate is a critical task in robot navigation and 3D modelling [4].The ability to estimate depth maps from monocular pictures would make these systems more robust, simpler, and less expensive. The main of this this is to predict the depth from monocular camera with good accuracy so that can be use in navigation in UGV's.

1.2.6 Aim

The goal of this master's thesis study is to implement, train, and test a deep convolutional neural network that predicts depth from monocular RGB images. This research looks

into the possibilities of improving the performance of a cutting-edge network for depth estimation.

1.3 Theory and state of arts in computer vision

1.3.1 Artificial neural network (ANN)

Artificial Neural Networks (ANNs) are inspired by the biological neural system and are made up of several nodes, known as neurons, that are linked together.A general ANN has an input layer, one or more hidden layers, and an output layer. Each layer has a number of neurons that are linked to neurons in the next layers.Each neuron in each layer is connected to all neurons in the subsequent layers if the layer is fully connected(FC).Artificial neural networks that have two or more hidden layers are called deep neural networks, DNNs [32].The number of hidden layers and neurons required depends on the task's and application's complexity.An artificial neuron, like a biological nerve cell, takes input from other neurons and transmits output to other neurons. Certain weights, which can be thought of as the strength of the connections, describe how much a given input influences the output. The weight tells that how inputs are mapped with outputs.The output is thus a weighted sum of all the connected input neurons. The output is then converted by passing this linear combination through an activation function to obtain the final output that is passed on to the other neurons.In supervised learning, the hidden layers of the network try to learn a mapping between the input and the matching ground truth. Complex functions can be learned by joining numerous neurons with the right weights and bias.The difference between the ground truth and anticipated output is utilised to adjust the learn able parameters (weights and biases) in order to minimized the error.A loss function is used to calculate the error to be minimised during training.

1.3.2 Convolutional neural networks (CNNs)

In CNNs kernel is used so that spatial information of the input is not lost. It widely used in image processing.Convolutions, activation functions, and pooling are the three fundamental components of a CNN.CNNs frequently use two additional techniques known as batch normalisation and drop-out. These technique allow the network to learn the critical features required for job completion during the training period.

Convolutional layers

The input, which is generally a picture, is convolved with a filter that slides over the entire image in the convolutional layer. As a result, each neuron in the hidden layer will be linked to various sections of neurons in the previous layer. The receptive field is the size of the region indicated by the filter size [2]. The stride is the number of pixels the

filter moves in each new step. The outcome of the convolution is known as a feature map since it captures features or patterns in the image. A feature might be a horizontal edge or line, but it can also be a more sophisticated structure.The weights and biases of the filters are learned during training and determine which features to extract.This is prompted by the idea that qualities that are useful in one section of the image are likely to be useful in other regions as well.

Activation function

The resulting linear combination of inputs is passed via an activation function, which translates the values to a different range and introduces non-linearities.This enables the model to learn high-order polynomials between the input and the ground truth, which is required because the underlying problem is unlikely to be linear.There are various types of activation functions, the most common of which is ReLU [32], but its version PReLU is also popular.

Rectified linear unit (ReLU) The rectified linear unit, or ReLU, is a non-linear activation function that is extensively employed due to its simplicity and dependability [33]. For an input x, ReLU is defined as

$$\text{f(x)} = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \qquad f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$$

The benefit of ReLU is that it leads to quick and easy computations, which results in faster training [30]. The problem of disappearing gradients as x grows is also avoided because the gradient for x 0 is constant. ReLU only activates positive input values, whereas negative values are mapped to zero and so do not activate. Back propagation does not update the parameters of non-activated, or dead, neurons, which leads to slow learning [33].

Parametric Rectified Linear Unit (PReLU) He et al. [21] proposed a variation of ReLU termed parametric ReLU, or PReLU, to alleviate the problem of dead neurons and increase model fitting. This activation function allows for activation even when the input is negative. PReLU is defined in below figure , where a is a coefficient determining the function's negative slope.

$$\text{f(x)} = \begin{cases} a_i x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \qquad f'(x) = \begin{cases} a_i & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$$

If a = 0, PReLU becomes the regular ReLU. The coefficient a is a parameter that is learned during training. It can be taught channel-by-channel or shared across channels,

as indicated by the subindex i [21].

Pooling layers

A pooling layer is often placed after the feature map has been routed through the activation function to reduce its spatial size.This minimises the amount of parameters the network must learn, improving the network's computational efficiency [20].It also expands the receptive area and renders the network insensitive to tiny input translations.The pooling layers do not contain learnable parameters, but instead a fixed function is applied separately on each of the feature maps [2].During this procedure, a small filter, often 2x2 or 3x3, glides over the feature map and produces a summary statistics of the closest values. The most popular pooling options are max-pooling and average-pooling [20]. Max-pooling involves taking the maximum of the values within the filter region. Because the same feature is detected over the entire feature map, returning the greatest value in a specific region indicates whether the feature is present or not.As a result, the network becomes insensitive to small translations of the input image, because the maximum value is always extracted inside a small region.However, the precise spatial location of the feature is lost as a result. Average-pooling, like max-pooling, returns the average of all values within the filter region. Because just one value is returned for a region with numerous values, the spatial size of the feature map is minimised. The filter size and stride decide how much it is decreased, or the downsampling factor. Zero padding can be used to obtain the required downsampling factor.

1.3.3 Training a neural network

Loss function

The loss function is used to measure the difference between the ground truth value, y, and the predicted value y^. When predictions are made pixel by pixel, then prediction error is also determined pixel by pixel. During the training process, the weights and biases are changed in order to minimising the loss function. Choosing an adequate loss function is thus critical for the network to successfully learn its parapeters. L1, L2, and the BerHu loss function [6] are some common loss functions for regression situations.

L1 loss The L1 loss is also called the mean absolute error and is defined

$$L_1 loss = \frac{1}{n}\sum_{i=1}^{n} |y_i - x_i|$$

Here, yi is the true value in pixel i and yi is the anticipated value in that pixel. The loss is averaged over all pixels and divided by the total number of pixels, n. The L1 loss is a linear function that grows linearly with the size of the prediction mistake.

L2 loss The L2 loss is also called the mean square error and is defined as

$$L_2loss = \frac{1}{n}(\sum_{i=1}^{n}(x_i - y_i)^2$$

Here, yi is the ground truth value in pixel i and yi is the predicted value in that pixel. The squaring of the prediction error results in a loss function with a non-linear behaviour. L2 loss returns a higher loss value for larger prediction error.Squaring the loss, simplifies the computations of the gradient of the loss function during back propagation.

BerHu loss B(y) $= \begin{cases} |y - y_i| & |y - y_i| \leq c, \\ \frac{(y-y_i)^2+c^2}{2c} & |y - y_i| > c, \end{cases}$ If the absolute prediction error is less than or equal to c, the function returns the L1 loss; otherwise, it produces a scaled version of the L2 loss. The linear qualities of L1 and the non-linear properties of L2 are implemented in this manner, which is stated in [29] as an advantage of utilising the BerHu loss. The value of c not only determines where the transition between the two loss functions occurs, but it also scales the quadratic loss function.

Cross entropy loss

$\sum_{c=1}^{M} y\_o, c \log(p\_o, c)$

M : number of classes
log : the natural log
y :binary indicator (0 or 1). if class label c is correct classification for observation o
p: predicted probability observation o is of class c

Cross entropy is an excellent loss function for classification problems since it minimizes the difference between two probability distributions: expected and actual.

1.3.4 Optimization

During the training phase, minimizing the loss function with respect to the weights of the network is known as "optimization." The training process begins with forward propagation, in which the input is sent through the network to generate the prediction. A loss function is used to compare the prediction against the truth. The gradient of the loss function is calculated with respect to the weights and the information is sent back through the network via back propagation

Stochastic gradient descent

Gradient descent (GD) computes the gradient over the whole training set and updates the weights after each epoch. An epoch is complete when all training samples have been propagated forward and backward across the network. To speed up the training process

considerably, Stochastic-Gradient-Descent (SGD) can be used instead [20]. One advantage with SGD is that, even if the number of training samples increase, the computation time for each update remains the same. This allows for convergence even for large training sets [20].The SGD is less likely than the GD to become converge in local minimums. Gradient decent formula defined as

$w^{(t+1)} := w^{(t)} + \gamma \Delta w^t$

γ : learning rate

Δw^t : gradient of weight at t_{th} iteration

The learning rate is a user-defined hyper parameter that has a significant impact on training performance. Because significant steps are completed each time, using a high learning rate results in faster training. If the learning rate is excessively high then the system may overshoot the minimum and diverge instead.A lower value of learning rate will result in slower convergence and the possibility of becoming stranded in local minimums. In general, a good strategy is to train the network with a higher learning rate for a number of epochs before decreasing the learning rate [32]. A popular SGD variant is the Adaptive Moment Estimation (ADAM) optimizer. we can used adaptive learning rate, initially higher value of learning rate then smaller value of learning rate at the end. The approach has shown excellent performance in terms of computing efficiency and memory needs, and it is especially well suited for issues requiring huge datasets and high-dimensional parameter spaces.

Hyper parameters Tuning and Regularization

A network's should be able to perform well not only on training data, but also on data it has never seen before. There is two major problem in network is overfiting and underfiting. Underfitting occurs when a network has too few parameters and cannot adapt successfully to either the training or test sets.The network will be able to learn a more sophisticated mapping between the input and output with a larger number of parameters. However, this does not guarantee that it is a good model because it may get overly adapted to the training data and fail to generalise to the test set [32]. This is referred to as overfitting. Deep neural networks are more prone to overfitting due to their complexity and large amount of parameters [39].To overcome this problem we used regularization.

Drop-out Drop-out is a regularisation approach in which a random fraction of hidden neurons is briefly off during the training process. The parameters of the off neurons will not be updated during backpropagation [32].Drop-out is very useful for training deep networks, when overfitting is a concern.

Batch normalization Ioffe et al. proposed batch normalisation to reduce covariance shift by normalising the input to each layer on a batch-by-batch basis.Batch normalisation prevents saturated activations, makes learning more stable, and allows for a higher learning rate.

Data augmentation

Deep learning model work best on large datasets, in case of less data we can used data augmentation. which improves the model efficiency.Simple and effective augmentation procedures include rotation, flipping, and cropping [36]. Translation and scaling have also proven to be effective. A little amount of Gaussian noise with zero mean can sometimes be introduced to the input images to prevent the network from learning ineffective high frequency structures, making the network more resistant. [20]

1.3.5 Datasets

KITTI dataset

The KITTI dataset [18] is widely used for training and assessing depth prediction networks. It has been used to train the majority of the networks [5, 15-17, 19] mentioned in section 2.5. The collection contains roughly 93 thousand photos of outdoor sceneries organised into five categories: city, residential, road, campus, and individual. The data was collected utilising a high resolution colour stereo camera rig and a Velodyne 3D laser scanner (LiDAR), which sampled the scene's depth.In total, 151 sequences were recorded, with raw data from the left and right cameras provided for each frame. A corrected version of the RGB photos, as well as the raw LiDAR scans, are available for download. Because the raw LiDAR scans are sparse, they do not contain data for all pixels. Uhrig et al. [42] created denser depth maps by constructing a large scale dataset of depth annotated RGB images from sparse scan data. They were prompted to do their research after seeing that typical convolutional neural networks perform badly when trained on sparse input. The depth maps given are four times denser than the raw LiDAR scans, containing data for four times the number of pixels [42].

NYU V2 datasets

The NYU-Depth V2 data collection consists of video sequences from various indoor situations captured by the Microsoft Kinect's RGB and Depth cameras. Its features follows as:

•1449 densely labeled pairs of aligned RGB and depth images

•464 new scenes taken from 3 cities

•Each object is labeled with a class and an instance number. The dataset has several components

•Labeled: A subset of the video data accompanied by dense multi-class labels. This data has also been preprocessed to fill in missing depth labels.

Chapter 2

Literature Survey

.

Monocular Depth Estimation pixel by pixel Depth estimation is a dense prediction task that assigns a depth value to each pixel in the image. It is closely related to the problem of semantic segmentation, which seeks to assign a class name from a set of categories to each pixel [26]. For dense prediction jobs, it is preferable to produce an output prediction with the same spatial size as the input, such that each input pixel has a corresponding prediction. This can be accomplished by employing an encoder-decoder network design. The encoder extracts important features from the input image and reduces the feature maps in size. Following that, the decoder upsamples the feature maps, recovers the image resolution, and generates pixel-wise predictions [8]. ResNet [22] and VGG [38], which were originally built for image classification, do not include a decoder. ResNet is used as an encoder by Laina et al. [29], Fu et al. [16], Jiao et al. [27], and Cao et al. [5].

2.1 State of the arts in depth prediction

Multi-scale deep neural network, 2014

Eigen et al. [15] used a multi-scale deep neural network in one of the first successful attempts to predict depth from monocular pictures using deep neural networks.This work demonstrates the value of integrating information from many scales and the importance of integrating the global context of the image in order to take use of monocular cues. The approach is built on two networks: coarse and fine. Using the full image, the coarse network predicts the depth of the scene at a global level. The output from the coarse network, together with the original image, is passed as input to the fine network that refines the prediction by using local information. The network produced state-of-the-art results on all error metrics for both the NYU-Depth V2 dataset and the KITTI dataset.

Embedding of focal length

He et al. [23] present a unique deep convolutional neural network integrating information about the camera's focal length for estimating depth in monocular images.The approach tries to assign a depth value to each pixel and employs an encoder-decoder structure based on VGG initialised with pretrained weights, followed by fully connected layers and upsampling. The focal length information is embedded in the network's final fully connected layers.They employ the BerHu loss function to take advantage of the benefits of both the L1 and L2 losses.

Unsupervised monocular depth estimation with left-right consistency, 2016

Godard et al. [19] demonstrate that just taking the reconstruction loss into consideration results in good picture reconstruction but poor depth estimates. Using only the left image and epipolar geometry constraints, their network can generate disparity maps for both pictures (left to right and right to left) at the same time.

Unsupervised CNN for single view depth estimation, 2016

They pose the depth estimation problem as an image reconstruction challenge, training on a pair of images from a stereo camera using an auto-encoder configuration. The network predicts the inverse depth of the left picture, then reconstructs it using inverse warping, the disparity, and the right image. The rebuilt image is matched to the input, and the reconstruction loss is reduced throughout the training phase. Their network is built on AlexNet, and the last completely linked layers are replaced with fully convolutional layers, as in [29]. They use skip-connections to achieve more precise predictions. The anticipated disparity (inverse depth) map is upsampled to the original image s to evaluate their method.

Semantic booster and attention driven loss, 2018

The pixel depth values in most situations are irregularly distributed and have a lengthy tail distribution. Jiao et al. [27] present a solution for network supervision that accounts for the imbalanced depth distribution by employing an attention driven loss. Simultaneously, they take advantage of the fact that depth and semantics in the scene share context information and can benefit from each other. They use a multi-task deep convolutional neural network with depth prediction and semantic labelling sub-networks. The input is routed through a backbone encoder based on ResNet-50 that has been pretrained on ImageNet and produces high-dimensional feature maps. The feature maps are passed to the sub-networks, which anticipate the depth and semantic labels. A Lateral Sharing Unit (LSU) is employed for information sharing between the two tasks, and the network learns this sharing method.This work shows that using a loss function adapted to the dataset distribution and incorporating semantic information can improve the depth predictions.

Deep Ordinal Regression Network (DORN), 2018

Fu et al. [16] present a Deep Ordinal Regression Network that predicts depth from monocular pictures using a multi-scale method. They use a space-increasing discretization method to divide the depth values into intervals (SID). This method takes into account the fact that the uncertainty in depth prediction rises as depth values increase, allowing for higher errors for deeper values. The depth prediction problem is modelled as an ordinal regression problem, which combines the principles of regression and classification and seeks to assign each pixel to one of a set of ordered categories [10]. They compare VGG-16 and ResNet-101 and found that ResNet-101 performs better then VGG-16 on all error metrics.

Depth estimation as a classification task, 2017

The majority of depth prediction networks approach the problem as a regression task.However, various attempts have been made to recast the problem as a classification task instead. Cao et al. [5] present the depth estimation problem as a pixel-wise classification job, with post-processing utilising a fully convolutional residual network and fully connected conditional random fields (CRF:s) for post processing.They mean that depth values are hardly exactly regressed to the ground truth and that it is easier to predict a pixels depth interval instead of the exact value. They also emphasise that one issue with regression is that it tends to forecast the mean depth value, which can lead to big errors for locations that are either extremely close or very far away. Cao et al. [5] experiment with three distinct network architectures: VGG-16, ResNet-101, and ResNet-152. ResNet outperforms VGG-16, with ResNet-152 outperforming ResNet-101 significantly better.

One Stage Object Detection

Deep learning-based object detection techniques are typically classified into two types: One of them is to detect targets using classification area proposals, such as RCNN[2], Fast-RCNN[3], Faster-RCNN[4], and so on. This is called two-stage detection approach. First, a sliding window method is used to acquire a candidate area on the image, then the feature vector of the candidate area is retrieved, and finally, the classifier's scoring result is used to determine the target category of the candidate area. The other type of strategy is one-stage methods such as Retina net YOLO, SSD, etc., which use a method of returning to the target category directly. Although detecting speed is substantially enhanced, accuracy suffers. RetinaNet, a one-stage detector, introduces a new focal loss function and a feature pyramid network (FPN) to achieve the accuracy of the most advanced two-stage approach while maintaining detection speed. One of the most widely used detection methods. RetinaNet, on the other hand, is constrained by its backbone network and continues to perform poorly in dealing with complex surroundings and low-resolution problems.

Retina net.

RetinaNet is divided into three sections: the backbone network network (ResNet) for feature extraction, feature pyramid networks (FPN), classification and regression sub-networks. The picture characteristics are first retrieved by ResNet, then restructured in FPN, and lastly transmitted to the classification and regression sub-networks to acquire the final detection findings.

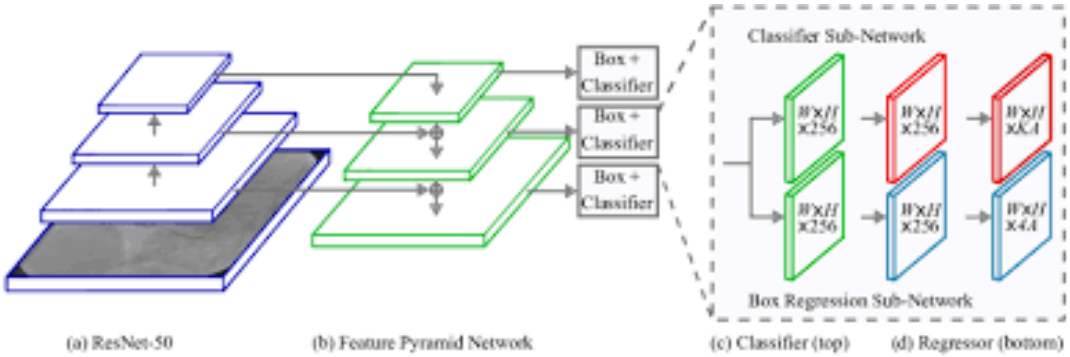

Figure 2.1: Retina net.

2.1.1 R-CNN Model Family

The R-CNN family of methods refers to the R-CNN, which stands for "Regions with CNN Features" or "Region-Based Convolutional Neural Network," developed by Grishick et al. This includes the techniques R-CNN, Fast R-CNN, and Faster-RCNN designed and demonstrated for object localization and objects recognition.

The R-CNN was described in the 2014 paper by Ross Girshick, et al. It has been one of the first large and successful applications of convolutional neural networks to the problem of object localization, detection, and segmentation. The approach was demonstrated on standard datasets, achieving then state-of-the-art results on the VOC-2012 dataset and the 200-class ILSVRC-2013 object detection dataset.

Their proposed R-CNN model has contained three modules; they are:

• **Module 1:Region Proposal.** Generate and extract category-independent region proposals, e.g., candidate bounding boxes.

• **Module 2: Feature Extractor.** Extract features from each candidate region, e.g. using a deep convolutional neural network.

•**Module 3: Classifier.**Classify features as one of the known classes, e.g., linear SVM classifier model. The architecture of the model is summarized in the image below, taken from the paper

The feature extractor used by the model was the Alex Net deep CNN image classification competition. The output of the CNN was a 4,096-element vector that describes the contents of the image that is fed to a linear SVM for classification, specifically one SVM is trained for each known class. It is a relatively simple application of CNNs to

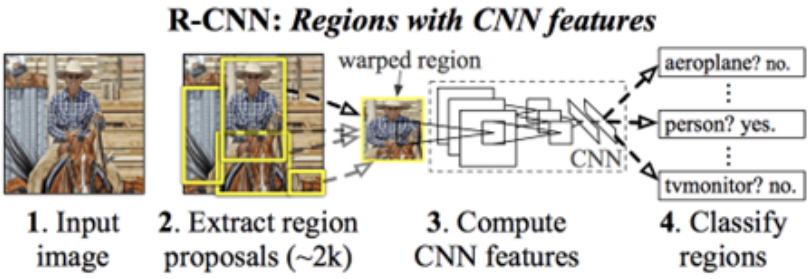

Figure 2.2: R-CNN Mode

the problem of object localization and recognition. A downside of the approach is that it is slow, requiring a CNN-based feature extraction pass on each of the candidate regions generated by the region proposal algorithm. This is a problem as the paper describes the model operating upon approximately 2,000 proposed regions per image at test time.

Fast R-CNN

Fast R-CNN is proposed as a single model instead of a pipeline to learn and output regions and classifications directly. The architecture of the model takes the photograph of a set of region proposals as input that are passed through a deep convolutional neural network. A pre-trained CNN, such as a VGG-16, is used for feature extraction. The end of the deep CNN is a custom layer called a Region of Interest Pooling Layer, or RoI Pooling, that extracts feature specific to a given input candidate region. The architecture of the model takes the photograph of a set of region proposals as input that are passed through a deep convolutional neural network. A pre-trained CNN, such as a VGG-16, is used for feature extraction. The end of the deep CNN is a custom layer called a Region of Interest Pooling Layer, or RoI Pooling, that extracts feature specific to a given input candidate region. The output of the CNN is then interpreted by a fully connected layer then the model separates into two outputs, one for the class prediction via a softmax layer, and another with a linear output for the bounding box. This process is then repeated multiple times for each region of interest in a given image. The architecture of the model is summarized in the image below, taken from the paper.

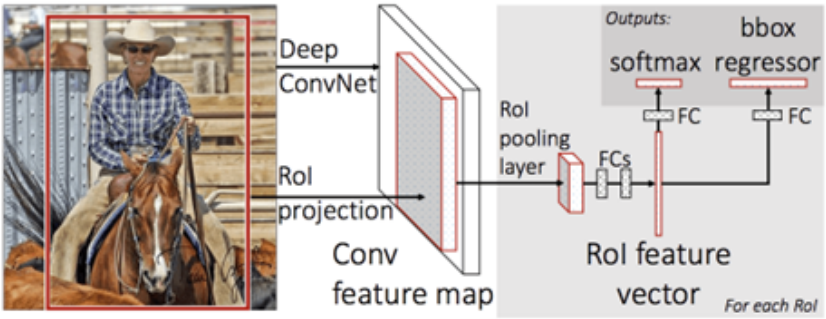

Figure 2.3: Fast R-CNN Model

The model is significantly faster to train and make predictions, yet still requires a set

of candidate regions to be proposed along with each input image.

Faster R-CNN

The model architecture was further improved for both speeds of training and detection by Shaoqing Ren, et al. at Microsoft Research in the 2016 paper. The architecture was designed to both propose and refine region proposals as part of the training process, referred to as a Region Proposal Network, or RPN. These regions are then used in concert with a Fast R-CNN model in a single model design. These improvements both reduce the number of region proposals and accelerate the test-time operation of the model to near real-time with then state-of-the-art performance. it is a single unified model; the architecture is comprised of two modules:

• Module 1: Region Proposal Network. Convolutional neural network for proposing regions and the type of object to consider in the region.

• Module 2: Fast R-CNN. Convolutional neural network for extracting features from the proposed regions and outputting the bounding box and class labels.

Both modules operate on the same output of a deep CNN. The region proposal network acts as an attention mechanism for the Fast R-CNN network, informing the second network of where to look or pay attention. The architecture of the model is summarized in the image below, taken from the paper.

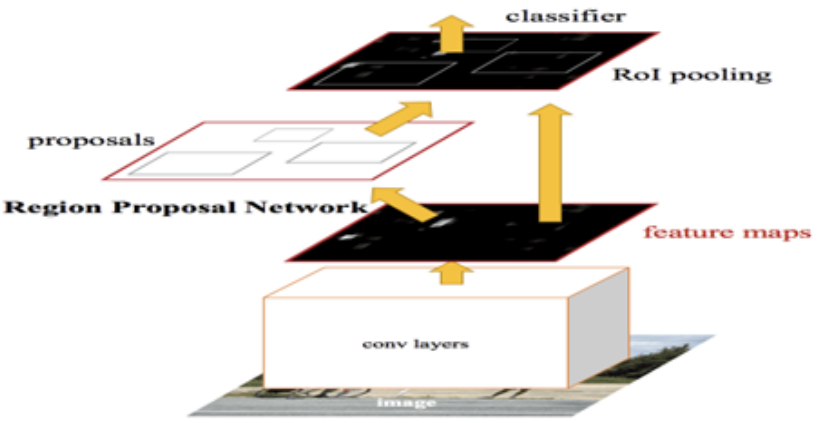

Figure 2.4: Faster R-CNN Model

The RPN works by taking the output of a pre-trained deep CNN, such as VGG-16, passing a small network over the feature map, and outputting multiple region proposals and a class prediction for each. Region proposals are bounding boxes, based on so-called anchor boxes or pre-defined shapes designed to accelerate and improve the proposal of regions. The class prediction is binary, indicating the presence of an object, or not, the so-called "objectness" of the proposed region. A procedure of alternating training is used where both sub-networks are trained at the same time, although interleaved. This allows the parameters in the feature detector deep CNN to be tailored or fine-tuned for both tasks at the same time. This Faster R-CNN architecture is the peak of the family of models and continues to achieve near state-of-the-art results on object recognition tasks.

A further extension adds support for image segmentation, described in the paper 2017 paper "Mask R-CNN

2.1.2 YOLO Model Family

Another popular family of object recognition models is referred to collectively as YOLO or "You Only Look Once," developed by Joseph Redmon, et al. [] The R-CNN models are generally more accurate, yet the YOLO family of models are fast, much faster than R-CNN, achieving object detection in real-time.

YOLO

The YOLO model was first described by Joseph Redmon, et al. in the 2015 paper titled "You Only Look Once: Unified, Real-Time Object Detection." Note that Ross Girshick, developer of R-CNN, was also an author and contributor to this work, then at Facebook AI Research. The approach involves a single neural network trained end to end that takes a photograph as input and predicts bounding boxes and class labels for each bounding box directly. The technique offers lower predictive accuracy (e.g., more localization errors), although operates at 45 frames per second and up to 155 frames per second for a speed-optimized version of the model. The model works by first splitting the input image into a grid of cells, where each cell is responsible for predicting a bounding box if the centre of a bounding box falls within the cell. Each grid cell predicts a bounding box involving the x, y coordinate, the width and height, and the confidence. A class prediction is also based on each cell. For example, an image may be divided into a 7×7 grid and each cell in the grid may predict 2 bounding boxes, resulting in 94 proposed bounding box predictions. The class probabilities map and the bounding boxes with confidences are then combined into a final set of bounding boxes and class labels. The image taken from the paper below summarizes the model's two outputs.

YOLOv2 (YOLO9000) and YOLOv3

The model was updated by Joseph Redmon and Ali Farhadi to further improve model performance in their 2016 paper titled "YOLO9000: Better, Faster, Stronger." Although this variation of the model is referred to as YOLO v2, an instance of the model is described that was trained on two object recognition datasets in parallel, capable of predicting 9,000 object classes, hence given the name "YOLO9000." Several architectural changes were made to the model, such as the use of batch normalization and high-resolution input images. Like Faster R-CNN, the YOLOv2 model makes use of anchor boxes, pre-defined bounding boxes with useful shapes and sizes that are tailored during training. The choice of bounding boxes for the image is pre-processed using a k-means analysis on the training dataset. Significantly, the predicted representation of the bounding boxes is changed to allow small changes to have a less dramatic effect on the predictions, resulting in a more stable model. Rather than predicting position and size directly, offsets are predicted for moving and reshaping the pre-defined anchor boxes relative to a grid cell and dampened by a logistic function. Further improvements to the model were proposed by Joseph Redmon

and Ali Farhadi in their 2018 paper titled "YOLOv3: An Incremental Improvement." The improvements were reasonably minor, including a deeper feature detector network and minor representational changes.

Chapter 3

Methodology

The implemented method is explained in this chapter.In this phase involves retraining a pretrained network on a widerface datasets and evaluating the performance. In the next phase, I will make changes to the network design in order to increase performance on the depth estimation task. I have chosen the Retina net model to retrain the network because it is a single stage object detector that is fast and accurate.

3.1 RetinaNet architecture

RetinaNet. can decomposed in three parts.

•The backbone network called Feature Pyramid Net, which is built on top of ResNet. or MoboNet. and is responsible for creating convolutional feature maps of an entire image.

•A subnetwork that performs object classification utilising the output of the backbone.

•A subnetwork responsible for performing bounding box regression using the backbone's output.

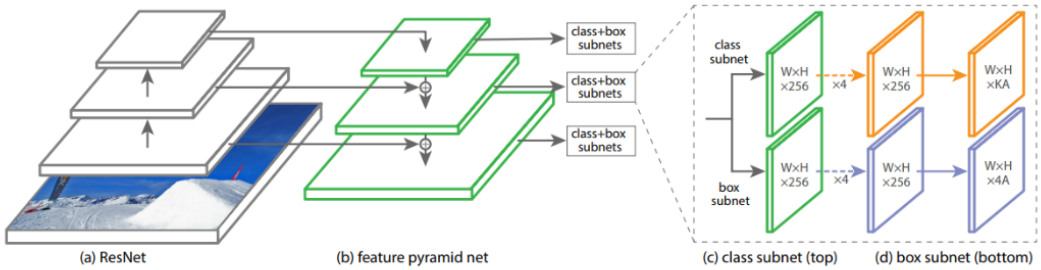

Figure 3.1: RetinaNet model architecture

The baseline design is based on an encoder-decoder structure. The encoder is based on ResNet or MoboNet. and begins with convolution, batch normalisation, a ReLU activation, and a max-pooling layer. Convolution and batch normalisation are used after the Residual blocks to reduce the feature maps. we designed architecture such that we get features map at every stage, for that we used feature pyramid network which is laterally

connected to every statge of resnet. we get different scaled features maps. After that we used two subnetworks. One is for classification of the objects, and other is for the regression network.

3.1.1 Network training

This section discusses the network training process. PyTorch was used for the implementation, training, and evaluation. Model is trained in server GPU of 16 GB. which is installed in EEE department of IIT Guwahati.

3.1.2 Datasets

For training and evaluation, the public NYU v2 dataset was used. The NYU-Depth V2 data collection consists of video sequences from various indoor situations captured by the Microsoft Kinect's RGB and Depth cameras. Its features follows as:

•1449 densely labeled pairs of aligned RGB and depth images

•464 new scenes taken from 3 cities

•Each object is labeled with a class and an instance number. The dataset has several components

•Labeled: A subset of the video data accompanied by dense multi-class labels. This data has also been preprocessed to fill in missing depth labels.

•Toolbox: Useful functions for manipulating the data and labels.

3.1.3 Data preprocessing and augmentation

Data augmentation was used during the training and validation phases. In terms of data augmentation, the photos were subjected to horizontal flipping with a 50 percent chance and random rotation. This was done after the subsampling but before the random crop was extracted. The ground truth images were converted to float images, and the depth in metres was calculated by dividing each pixel value by 256. As a result, the network was trained on ground truths in metres and, as a result, predicts in meters.

3.1.4 Mask function

To deal with the sparsity of the depth maps and avoid interfering with the training, a binary mask was created for each depth map.The mask contained ones where depth values were legitimate and zeros where depth values were invalid. In the loss function, the mask was multiplied by the expected image to extract just the pixels having a matched ground truth, and the loss was calculated exclusively over these pixels.

3.1.5 Initialization of weight and frozen layers

To make the network converge faster, the baseline network was initialised with pretrained weights. To reduce training time even more, the settings of all layers up to the second ResNet block were frozen.This was inspired by the fact that the first layers of CNNs collect low-level features that were thought to be identical between the two datasets.

3.1.6 Optimization

The loss function is minimised with respect to the parameters as given during the optimization. The parameters that the user can control, such as the learning rate, batch size, number of epochs, and so on, are known to as hyperparameters and can have a significant impact on the training results. This section explains how the hyperparameters were selected.Using a large batch size, according to Goodfellow [20], is preferable since it provides a more accurate estimate of the gradient. During training, the maximum possible batch size was chosen, which was limited by GPU memory. Because it combines the positive qualities of both the L1 and L2 losses, the BerHu loss was chosen as the loss function. Laina et al. [29] and He et al. [23] found that utilising BerHu over L2 and L1 loss produced more accurate depth estimates, which justified the use of this loss function.

3.1.7 Evaluation

The model with the lowest loss on the validation set was chosen and utilised for testing on the test set. The examination was conducted both numerically and qualitatively, using the error metrics described, Over the whole test set, the average of each error metrics was calculated. The photos were subsampled in the same way as during the training phase, but no data augmentation was used. To ensure that the network constantly evaluates the same photos, a centre crop was extracted. Because the network was fed subsampled and cropped images with smaller spatial sizes than the originals, the network predicted these sizes as well. To measure network performance, the raw outputs were compared to the subsampled and cropped ground realities without any post-processing. As a result, the evaluation was performed at a lower resolution than the original.

Chapter 4

Results and Discussion

4.1 Qualitative Results

The primary purpose of this master's report is to implement, train, and evaluate a neural network that predicts depth from monocular RGB images.

The quality and size of the dataset on which a network is trained have a substantial influence on its performance. Retina Net, the proposed network, is fairly complex and has a significant number of parameters.To avoid overfitting, a large dataset is required.The network had to learn to fill in the missing values because the ground truth data was sparse and did not contain values in all pixels. The network struggled to predict appropriate values for locations that were never provided any depth information.

The search for hyper-parameters was not extensive. Some hyper-parameters, like as learning rate and batch size, were predicted by trial and error.

Figure 4.1: Nyu test image.

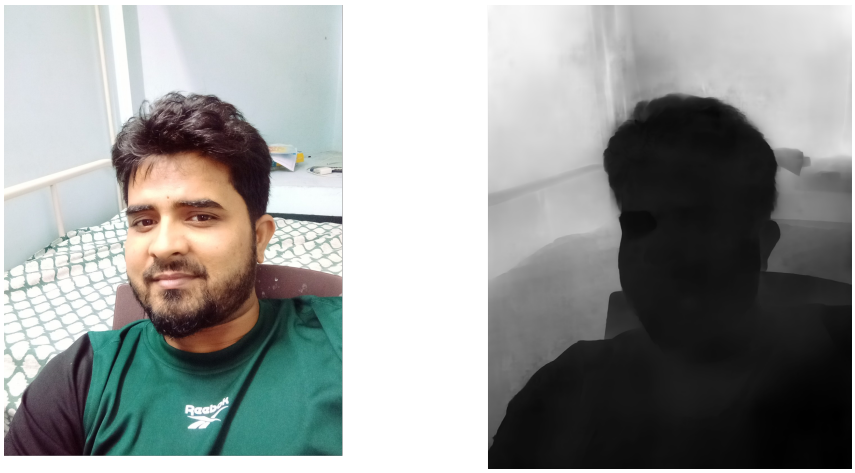

Figure 4.2: Self image test on model

4.2 Quantitative Result

Table 4.1: Quantitative Results

Method	Rel.	RMSE
Eigen et al. 2014	0.158	0.618
Laina et.al 2014	0.194	0.790
Fu et al. 2018	0.115	0.509
FPN network on top of resnet 50	0.056	0.46

Chapter 5

Future Work

Increasing the accuracy of the model by hyperparameter tuning and modeling the best loss function and making mobile applications for depth map. Finally integrating the predicted depth maps into a mono-SLAM system

Bibliography

[1] Niall O' Mahony (Institute of Technology Tralee), Sean Campbell (Institute of Technology Tralee), Lenka Krpalkova (Institute of Technology Tralee), et al (2018) Deep Learning for Visual Navigation of Unmanned Ground Vehicles; A review

[2] Karami E, Prasad S, Shehata M Image Matching Using SIFT, SURF, BRIEF, and ORB: Performance Comparison for Distorted Images

[3] Adit Deshpande A Beginner's Guide To Understanding Convolutional Neural Networks – Adit Deshpande – CS Undergrad at UCLA ('19). https://adeshpande3.github.io/ABeginner

[4] Tsai FCD (1994) Geometric hashing with line features. Pattern Recognit 27:377–389. https://doi.org/10.1016/0031-3203(94)90115-5

[5] Angelina M, Gim U, Lee H PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition

[6] Campo Seco F, Cohen A, Pollefeys M, Sattler T Hybrid scene Compression for Visual Localization

[7] Loghmani MR, Planamente M, Caputo B, Vincze M Recurrent Convolutional Fusion for RGB-D Object Recognition

[8] Clément M, Kurtz C, Wendling L (2018) Learning spatial relations and shapes for structural object description and scene recognition. Pattern Recognit 84:197–210. https://doi.org/10.1016/J.PATCOG.2018.06.017

[9] Bonaccorso G (2018) Machine Learning Algorithms Popular Algorithms for Data Science and Machine Learning, 2nd Edition. Packt Publishing Ltd

[10] Mahony NO, Murphy T, Panduru K, et al (2017) Improving controller performance in a powder blending process using predictive control. In: 2017 28th Irish Signals and Systems Conference (ISSC). IEEE, pp 1–6

[11] O'Mahony N, Murphy T, Panduru K, et al (2017) Real-time monitoring of powder blend composition using near-infrared spectroscopy. In: 2017 Eleventh International Conference on Sensing Technology (ICST). IEEE, pp 1–6

[12] O' Mahony N, Murphy T, Panduru K, et al (2016) Adaptive process control and sensor fusion for process analytical technology. In: 2016 27th Irish Signals and Systems Conference (ISSC). IEEE, pp 1–6

[13] Koehn P, Koehn P (1994) Combining Genetic Algorithms and Neural Networks: The Encoding Problem.

[14] Schöning J, Faion P, Heidemann G (2016) Pixel-wise Ground Truth Annotation in Videos - An Semi-automatic Approach for Pixel-wise and Semantic Object Annotation. In: Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods. SCITEPRESS - Science and Technology Publications, pp 690–697

[15] Alhaija HA, Mustikovela SK, Mescheder L, et al (2017) Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes

[16] Meneghetti G, Danelljan M, Felsberg M, Nordberg K (2015) Image Alignment for Panorama Stitching in Sparsely Structured Environments. Springer, Cham, pp 428–439

[17] Alldieck T, Kassubeck M, Magnor M (2017) Optical Flow-based 3D Human Motion Estimation from Monocular Video

[18] Zhang X, Lee J-Y, Sunkavalli K, Wang Z (2017) Photometric Stabilization for Fast-forward Videos

[19] Schöning J, Faion P, Heidemann G (2016) Pixel-wise Ground Truth Annotation in Videos - An Semi-automatic Approach for Pixel-wise and Semantic Object Annotation. In: Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods. SCITEPRESS - Science and Technology Publications, pp 690–697

[20] Ioannidou A, Chatzilari E, Nikolopoulos S, Kompatsiaris I (2017) Deep Learning Advances in Computer Vision with 3D Data. ACM Comput Surv 50:1–38. https://doi.org/10.1145/3042064

[21] Devries T, Taylor GW (2017) Dataset Augmentation in Feature Space. arXiv Prepr arXiv 170205538v1

[22] Dvornik N, Mairal J, Schmid C Modeling Visual Context is Key to Augmenting Object Detection Dataset

[23] Zheng L, Yang Y, Tian Q SIFT Meets CNN: A Decade Survey of Instance Retrieval

[24] Ng H-W, Nguyen D, Vonikakis V, Winkler S Deep Learning for Emotion Recognition on Small Datasets Using Transfer Learning. https://doi.org/10.1145/2818346.2830593

[25] AlDahoul N, Md Sabri AQ, Mansoor AM (2018) Real-Time Human Detection for Aerial Captured Video Sequences via Deep Models. Comput Intell Neurosci 2018:1–14. https://doi.org/10.1155/2018/1639561

[26] CS231n Convolutional Neural Networks for Visual Recognition. http://cs231n.github.io/transfer-learning/. Accessed 9 Mar 2018

[27] Highlander TC Efficient Training of Small Kernel Convolutional Neural Networks using Fast Fourier Transform

[28] Highlander T, Rodriguez A (2016) Very Efficient Training of Convolutional Neural Networks using Fast Fourier Transform and Overlap-and-Add

[29] Wang J, Perez L The Effectiveness of Data Augmentation in Image Classification using Deep Learning

[30] Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet Classification with Deep Convolutional Neural Networks. NIPS'12 Proc 25th Int Conf Neural Inf Process Syst 1:1097–1105

[31] Nash W, Drummond T, Birbilis N (2018) A Review of Deep Learning in the Study of Materials Degradation. NJ Mater Degrad 2:37. https://doi.org/10.1038/s41529-018-0058-x

[32] Hayou S, Doucet A, Rousseau J (2018) On The Selection of Initialization and Activation Function for Deep Neural Networks. arXiv Prepr arXiv 180508266v2

[33] Adit Deshpande A Beginner's Guide To Understanding Convolutional Neural Networks – Adit Deshpande – CS Undergrad at UCLA ('19). https://adeshpande3.github.io/ABeginner