# V3C-NET: End to End Traffic Volume Estimation

By

TAYYBA NAZ

00000170404

Supervisor

DR HASAN SAJID

Robotics and Intelligent Machine Engineering

SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

JULY 2019

# V3C-NET: End to End Traffic Volume Estimation

TAYYBA NAZ

00000170404

A thesis submitted in partial fulfillment of the requirements for the degree of

MS ROBOTICS AND INTELLIGENT MACHINE ENGINEERING

Thesis Supervisor:

DR HASAN SAJID

Thesis Supervisor's Signature: ________________________________________

ROBOTICS AND INTELLIGENT MACHINE ENGINEERING

SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,

ISLAMABAD

JULY, 2019

## Declaration

I certify that this research work titled "*V3C-NET: End to End Traffic Volume Estimation"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student

TAYYBA NAZ

2016-Nust-MS-RIME-00000170404

## Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

Signature of Student

TAYYBA NAZ

170404

Signature of Supervisor

DR HASAN SAJID

## Copyright Statement

## Acknowledgements

I would like to thank many individuals who helped me with finishing my thesis. Top of all. I want to Thank Allah for provision of knowledge, health and the strength to finish the research.

I would like to express my gratitude to my friends, especially Kashif for helping with thesis related things here in Abu Dhabi. I would like to thank Tayyab who helped me with completing the thesis related formalities during my absence in Pakistan. I also would like to thank my company, AiFi Technologies Abu Dhabi, for providing me with all the resources I needed to work on my thesis.

I would like to Thank Sir Hasan for encouraging and supporting me with my out of country job. I would like to Thank him for all the supervision he provided despite me being away.

My thanks and appreciation go to all people who helped me one way or another with my work.

*To my friends, I couldn’t have done this without you. Thank you for all the support.*

## Abstract

Visual object counting has been a well-researched and solved topic over the past few years in computer vision. Previous computer vision solutions fail to provide a robust solution to changes in illumination, occlusion, varying weather conditions and camera/vehicle orientation. These problems can be solved by providing machine learning solution. We propose an end to end method for counting of vehicular volume. We aim to replace the manual collection of the count carried out by civil engineers and surveyors. This data is important in identifying frequently used routes, roads that need more lanes, length of the lanes/roads and suitable traffic control methods. Our method uses 3D Convolutional Neural network which learns spatial-temporal features of vehicular volume and count them as the vehicle leaves the frame. Our method is able to learn to count vehicles without the use of any extra pre-processing, or applying any object tracking methods, and counting them when they pass through a line of interest or region of interest. Our method is robust to occlusions, camera/vehicle orientation, lighting condition and various weather conditions.

**Key Words:** *3D CNN, Vehicular volume, Spatial-temporal data*

# Table of Contents

# List of Figures

# List of Tables

# CHAPTER 1: INTRODUCTION

Traffic volume is very significant with proposing effective and rapid traffic solutions. Increase in traffic volume means the need for more lanes, more maintenance, effective timer settings for traffic signals, possible bridge replacements, new roads and other suitable traffic control measures. Moreover, this data can be used to determine the number of lanes needed on a certain road, width of lane, set timer for traffic lights. Traffic volume data can also indicate any possible change in traffic patterns. Roads with less vehicular volume can be dealt with single lane roads or bridges but for huge volume of traffic, the city municipality has to come with several laned roads and possible bridges solution which are very expensive to build and maintain. Hence, its crucial to have an accurate estimation of vehicular volume.

Vehicular Volume is defined as the estimated count of vehicles passing a certain section of road in a any given unit time. Over the past few years, there has been considerable increase in vehicular population all over the world as business and cities expand continuously. Since the first automobile was introduced [1] in 1885 we have seen ground breaking advancement in vehicular industry. Now we have cars. SUV's, Bikes, Trucks and vans. More Vehicles on the road mean, more traffic jams, demand of more roads and effective traffic solutions. The cities are already changing into digital societies with the conception of smart city projects. With advancement in smart city projects, intelligent transport system (ITS) is a very essential component. Intelligent transport system can be used to make traveling more efficient and smarter. The data collected can be used to improve traffic flow and analyze solutions for traffic related problems. Density estimation of vehicular traffic is very essential as it can play a major role in intelligent transport systems for smart city projects all over the world. This volume data helps with creation of effective traffic control measures, check existing control measures effectiveness, designing of intersections, signal timer duration, channelization.

Vehicular traffic count is done by civil engineers or surveyors to identify the most frequently used routes or excessive traffic. It is usually carried out by using mechanical count boards or electric count boards. Surveyors usually keep a tally sheet with them. Ticks are marked on a preprocessed form to carry out manual count. The surveyor has stand at the road side possibly under harsh weather conditions or either video is used to perform the task of counting. Usually small sample of taken and the data is extrapolated for the longer time periods. The desired count interval is measured by either using a watch or a stopwatch. Its very time consuming and requires lot of human work hours. Plus, one cannot ignore the possibility of human error in the total vehicular count. Other methods include having portable or permanent automatic counters. These are expensive solutions and their area of coverage isn't quite wide. Other popular methods [2] are Pneumatic Road tube counting, Piezoelectric Sensor, Inductive Loop, Magnetic Sensor, Acoustic Detector. Passive infrared, Doppler and Radar Microwave sensors. To replace the conventional method of counting, researchers have tried to automate the process by offering other hardware dependent solutions such as inductive loop detector [3] state of the art sensor with embedded system and WSN (wireless sensor network) [4] and portable magnetic sensor [5].

Hardware solutions involving sensors are more expensive and unreliable as the hardware components require constant maintenance. To avoid the dependency on the hardware solutions, pure software solutions backed by computer vision algorithms were proposed. The gaussian mixture model performance was optimized for counting by adjusting the region of interest [6]. In another work [7] vehicles were detected using active basis model and reflection symmetry is used to verify the vehicle detected. Majority of the computer vision-based solutions work on static images and most methods use region of interest (ROI) methods to count vehicles making the solutions dependent on static camera position. The purely software-based solutions aren't robust as they fail to recognize vehicles in different weather condition, illumination changes, occlusions and camera/vehicle orientations.

In this paper, we propose a novel 3D Convolutional Neural Network architecture for end to end traffic volume estimation. The network learns to collect vehicle features only from the real-world scene and count them as the vehicles leave the frame. Our approach requires no preprocessing of any kind of vehicular raw videos. The network learns by itself to distinguish between a vehicle and non-vehicle. We also provide no region of interest or line of interest to the network for the counting task. This is inspired from human's approach to solving this problem. Although 3d convolutional neural networks have been extensively used for classification problems, in this work we show that 3D convolutional neural network is well suited for the regression problem of counting vehicles.

*The rest of the paper is organized as follows. The related work has been discussed in chapter 2. Dataset has been briefly discussed in chapter 3. In chapter 4, we in detail the proposed the method. In section 5 and 6, we have discussed the results. And finally in chapter 7, we conclude our results.*

# CHAPTER 2: LITERATURE REVIEW

This section is divided into two parts, in A, we did a brief literature on computer vision and deep learning techniques aimed at counting objects. In B, the previous works done using 2d and 3d convolutional neural network was briefly reviewed to highlight the remarkable performance of these techniques.

## 2.1. Counting Objects

To overcome the problem of identifying and counting an object with illumination, occlusion, shadows, scale variation and overlapping is very challenging with computer vision algorithms. Machine learning are good alternatives to address these problems but these methods require training on large samples of data. Annotating this huge data is again quite tedious and time-consuming task. To avoid the collection of real-world data and to safe time on labeling the data, authors **[8]** used synthetic images for training their network. for testing purpose, images were taken from the real world. The following method was used to generate a synthetic image of size 128x128; a blank image was created followed by filling it with brown and green colored circles. This brown and green color represent the background (green color) and tomatoes (brown color). Gaussian filter was used to blur the image. Variable sized circles were drawn on random positions on the image. a total of 24,000 images were generated out of which 2400 were kept for testing purposes. Their proposed novel network has several layers of convolution and pooling operations with addition of modified Inception-ResNET layer. This architecture has previously performed very well image recognition challenges. The modification has three layers concatenated onto one layer. The modified version captures features in multiple scales. The last layer outputs the number of tomatoes in the given image. drop out technique was used to avoid the problem of overfitting. The network was trained in 3 epochs with Adam as optimizer. Mean Squared Error was used as cost function and the weights were initialized with Xavier initialization. The network is able to count half ripe and full ripe tomatoes robust to size, shadows, light conditions and partial/complete occlusions. The same network and concept can be used to train other fruits as well. The network performed with 91% accuracy with the ability to count ripe and half ripe tomatoes.

To tackle the problem of counting object visually, two novel deep learning approaches were proposed **[9]**. The first method proposes the use of regression from multiple image scales using Feature Pyramid Network (FPN). This covers for all cases of leaves either big or small sized leaves. the second method is a combination of estimates from the various image sizes based on estimated confidence variance. The variance value for regressor is predicted by the deep network. The latter method performed a 95% average precision on CVPPP 2017 Leaf Counting Challenge dataset. For training purposes, tobacco and Arabidopsis plant images were annotated and used.

Similarly, Oil Palm tree detection and counting **[10]** is important as the data can be used to predict the tree's yield, monitoring the health of current growing palm trees and to maximize the productivity out of crude palm oil. Remote sensing images were acquired for Malaysia. the authors implemented convolutional neural network on TensorFlow framework. To train the network, previously collected and manually interpreted training which were used. classification accuracy was calculated on number of test samples collected separately. The CNN parameters were continuously adjusted to achieve the best performing optimal weights. The best performing CNN was used to predict the label for every sample in the given dataset. the samples which were predicted as palm trees, their coordinates were merged with the same palm tree samples to yield one coordinate i.e. palm tree detection result. the paper used RELU activation function. The train image sample is very huge since it's a high-resolution image obtained by a remote sensing device, to process the image 17x17 pixel sliding window is used to obtain images for the network. the network achieved the highest accuracy of 96.05 %, 96.34%, 98.77% on region 1, region 2 and region 3 respectively.

In computer vision, background subtraction has been one of the most adopted method to identify the objects. It is highly popular method for static environments. Applying this method for object detection in a vehicle is bit challenging. **[11]** proposed a vision-based pipeline to count the vehicles. Background image and video frame were converted to gray scale and fed to background subtraction algorithm. The resultant image is used to establish a region of interest. This region of interest is vital for vehicle detection and count. Using thresholding, hole filling and adaptive morphological operations the centroid of the vehicle is calculated. When the centroid of the vehicle passes the virtual zone, the vehicle is counted. The method achieved an accuracy of 96.85%. But this strategy has some limitations, the vehicles should be in a clear view and not occluded in any way in the virtual zone. Secondly, the width of the virtual zone has to be wide enough to count all the vehicles in the science.

CSRNet **[12]** introduced deep learning inspired approach. It is able to accurately estimate count in highly congested scenes and also generate high quality density maps. It is composed of two parts a frontend feature extractor 2D CNN and a dilated CNN backend. The frontend is actually a pretrained, modified VGG16 without the fully-connected layers, and for the backend they come up with four different variants of the dilated CNN. For training they fine-tune the front-end for CSRNet was demonstrated on four datasets, ShanghaiTech, UCF_CC_50, WorldEXPO'10, USCD dataset, where it outperformed the state of the art, like on the TRANCOS vehicle counting dataset, where it was able to outperform the state of the art by a significant margin. Similarly, it was able to achieve 47.3% lower Mean Absolute Error (MAE) than the state of the art, in the ShanghaiTech Part_B dataset.

To optimize and automate Traffic control and management system, many solutions have been implemented. Vehicle detection using deep neural network and KLT tracker was proposed **[13]**. The pipeline suggests selecting an image from every Nth frame. This image was run through an R-CNN vehicle detection to find the bounding box. The same image was also used to extract new corner points, new trajectories are calculated using the new and old points and, in the end, confusing R (trajectories) is removed. The both results are then assigned R to the bounding boxes. If trajectory is inside bounding box and it has doesn't have a label, mark it as a first-time detected vehicle and increment the count counter. The pipeline missed some of the vehicle passing because of R-CNN network. R-CNN was trained with very less training example hence the algorithm performance is compromised.

To address the issue of failed detections due to varying shadowing conditions and occlusions problems, a six staged pipeline was implemented **[14]**. Firstly, background is removed from the video frames to exclude the non-moving objects. This action was performed pixel by pixel. In the second step, image segmentation is carried out by extracting the contours from the binary mask and bounding box being drawn around vehicle contours. In the next step the shadows are detected and removed. Then, frame by frame difference is used to track the vehicle. Lastly, objects are classified as either cars or non-cars and the detected vehicles are counted. But this paper fails to report the result accuracy and it uses the typical method of counting the vehicle once the centroid of vehicle passes the line of interest.

Traffic congestion is one of the major challenges faced by authorities today. To address this problem by analyzing the road conditions, **[15]** estimates traffic flow from surveillance videos. Background subtraction was carried out with K-nearest neighbor. Bounding box is drawn around the detected vehicle and counted as centroid of bounding box passes the virtual line. The paper reported a success rate of 95%.

Background subtraction has been popular method is computer vision-based solutions. Pixel Based Adaptive Segmenter Method **[16]** uses images taken from videos surveillance images, each image channel is processed separately. Decision threshold is used to carry put background and foreground decision to segment the background from foreground. cvBlob library is used to track the vehicle. Bounding triangle is drawn around the vehicle. This paper again uses the concept of bounding box passing the virtual line for counting of vehicle. This method fails for varying illumination conditions yielding a success rate of only 17% during the evening conditions and 48% during the morning time.

## 2.2. Convolutional Neural Networks

Convolutional networks were first introduced by Yann LeCun [17] in 1998 as LeNet. CNN’s have recently caught a lot since the advent of GPU’s and more RAM availability over the past years. CNN’s can be computationally expensive. Computer vision tasks such as object detection have been overtaken by CNN methods as they have been yielding quite impressive results.

Alexnet [18] was a very deep and large convolutional network trained to classify the one of the most challenging image dataset, ImageNet. ImageNet has over 15 million high resolution labeled images. The images belong to approximatively 22,000 classes. ImageNet Large Scale Visual Recognition Challenge is held every year and AlexNet won this challenge in 2012 with 27.5% and 17.0% top1 and top 5 error rates respectively. The network contained eight layers out of which 5 were convolutional and 3 were fully connected layers. ReLU was used as activation function and this huge network was trained on multiple GPU’s.

R-CNN used selective search to extract 2000 regions from the input image. These are called image proposals. These regions are passed to a CNN to compute the features for classification. This introduced the problem of taking too much time to train the network because of 2000 regions and doesn’t perform well in real time. To address these problem, same author came up with Fast R-CNN [19]. The whole image was fed to the convolutional neural network, regions of proposal were identified from the generated feature maps. Faster R-CNN [20] replaced the conventional method of the usage of selective search for region proposal, it uses a separate network to predict the region proposals. This reduced the speed of object detection algorithm to 0.2 seconds from staggering 49 seconds.

3D U-Net [21] extends the work from the previous U-Net [22] from 2D to 3D CNN architecture, and is able to achieve better Intersection over Union (IoU) than its 2D counterpart. They also define two major use cases of the method, in a semi-automated setting, where the user annotates some slices. The network learns from this sparsely annotated dataset and is able to produce dense 3D segmentation. The second use-case is fully-automated setup, where it is assumed that a user already annotated the dataset in the same manner as semi-automated, and the network is trained on this dataset. The performance of the method is tested on the Xenopus dataset, which is a highly variable 3D structure and complex dataset, and the proposed method is able to achieve good results for both settings.

Processing 3D data by a 2D Convolutional Neural Network is very challenging task as data representation might be a issue. 3D sensors such as LiDAR, RGBD and CAD camera output 3d data and most of the time the data isn't completely utilized. VoxNet [23] is 3D convolutional neural network designed for real time object recognition. The network takes point cloud data, convert it into occupancy grid and then pass it to the 3D CNN. Network outperformed ShapeNet [24] in majority of the tasks.

# CHAPTER 3: DATASET

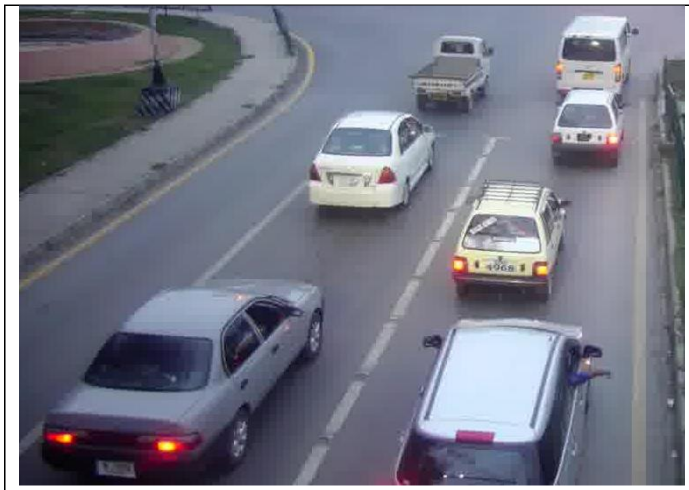
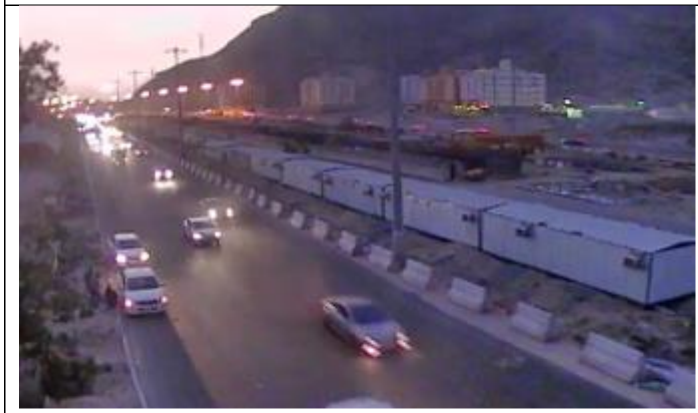
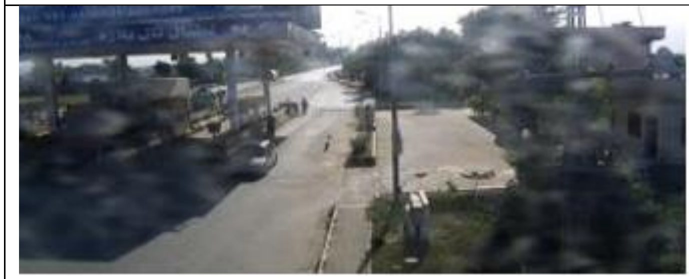
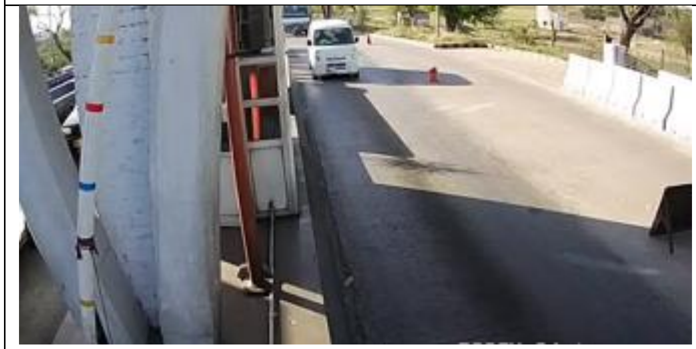
We manually annotated our dataset since currently no dataset is available to provide the right ground truth for static sequences and variable sequence images. The videos/images pretrained to different environments having illumination changes, changes in vehicle or camera orientation, congested/non congested traffic flow, image size variance and occlusions.

Vehicles were manually counted after every 10 second mark and noted down. Majority of the videos were 5-minute length videos. Whole data was arranged in excel sheets to make it readable and accessible during the implementation part. Initially we had a data of 3100 sample folders. This was split in 70:30 chunk for training and testing purposes respectively. The whole network will be trained on 2171 images and 929 images will be reserved for static length video samples testing.

For variable length samples, we were provided with folders having variable length sequence. Folders had images annotated from10,20,30 or 40 second video length. Some of the samples were also taken from NVIDIA AI City Dataset (publicly available).

Since Convolutional Neural Networks take fixed length input image, we couldn’t train the network on variable length samples. Variable length samples were all used for testing purposes to report bias free results. Conversion of variable length sample to be adapted with our network is explained in the next section.

| **Image** | **Description** |
|---|---|
| | • Dark illumination conditions<br>• Light Glare in image |
| | • Congested vehicular traffic conditions<br>• Camera perspective is different |
| | • No traffic in view<br>• Light conditions variance |

| | |
|---|---|
|  | • Change in camera orientation<br>• Light variance |
|  | • Change in weather conditions<br>• Variance in camera orientation<br>• Vehicle traffic congestion |
|  | • Shadowy conditions<br>• Camera view isn't clear |
|  | • Good camera resolution<br>• Clear view<br>• Good light conditions |

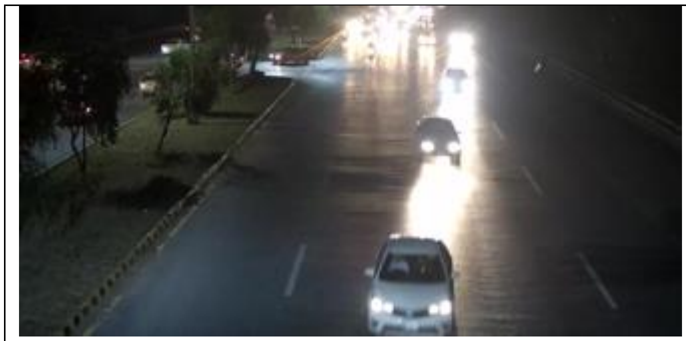
| | |
|---|---|
|  | • Dark Conditions<br>• Abundant Traffic<br>• Light glare |

Table 1 : Dataset snippets

# CHAPTER 4: METHODOLOGY

## 4.1. Convolutional Neural Networks

Since they were introduced in the late 1990s by Yann LeCun [25], Convolutional Neural Networks have gradually taken over the Computer Vision community. Most of the problem in Computer Vision have been invaded by the Convolutional Neural Networks.

### 4.1.1. Convolution

Convolutional Neural Networks are based on the principal of convolution, Convolution is a well-known operation in Image processing, which is given by:

$$I_{new} = \sum k.I$$

Where, $I_{new}$ is the new Image received after applying convolution, $k$ is the kernel, a small image which acts as weights in Convolutional Layers and $I$ is the original image.

### 4.1.2. 2D Convolution

2D convolution in a Neural network is performed to extract high-level features from local neighborhood on feature maps in the previous layer.

$$v_{ij}^{xy} = b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} w_{ijm}^{pq} \, v_{(i-1)m}^{(x+p)(y+q)}$$

$$a = max(v, 0)$$

Where, $\max(v_{ij}^{xy},0)$ is an activation / squashing function (which can be replaced by other activation functions, like, tanh, sigmoid, etc.), $b_{ij}$ is the bias, $m$ is indexes over the set of the feature maps in the *i-1*th layer connected to the current feature map, $w_{ijk}^{pq}$ is the value at the position, *(p, q)* of the kernel connected to the *k*th feature map, and $P_i$ and $Q_i$ are the height and width of the kernel, respectively.

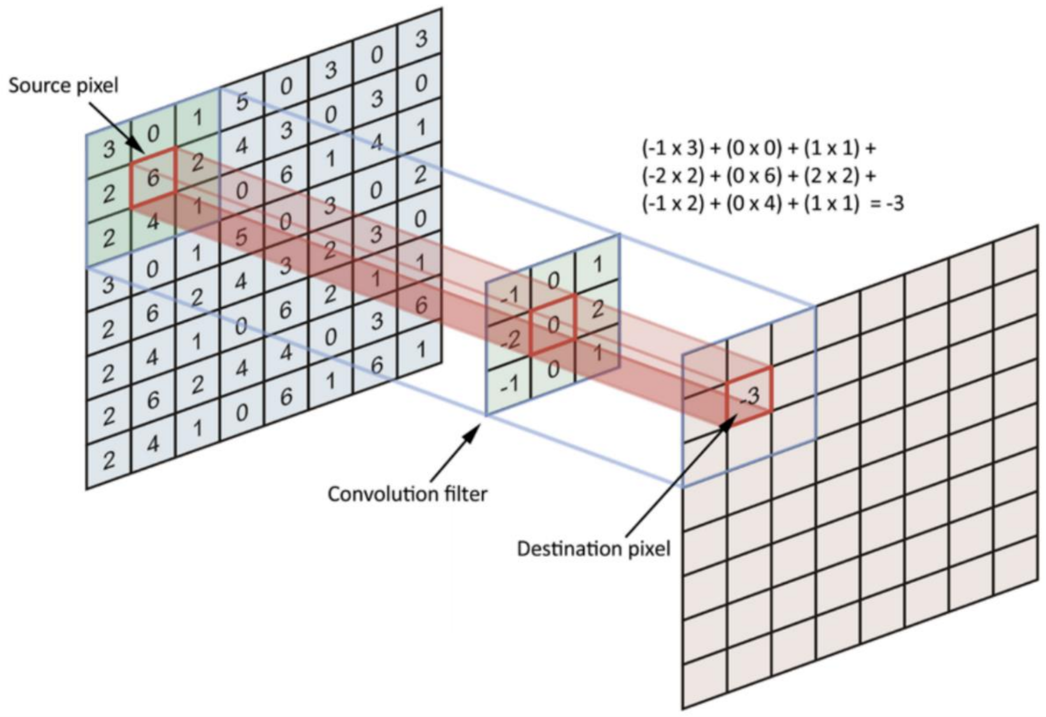

Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
Destination pixel

Figure 1 : Illustration of a 2D convolution [26]

The above equation is an equivalent to what we see in Fully connected layers.

$$y = Wx + b$$

Where, *W* is the Weights, *x* is the input feature and *b* is the bias.

In case of convolutions the total number of weights are a lot less then that of an equivalent fully connected layer because the same kernel is re-applied in case of convolutions all over the image, while in case of fully connected layers there is one weight for one corresponding pixel. This property of convolutions make them more efficient in computations and also they achieve better results. In practice multiple kernels are used to extract high-level features from the input image and then either the resultant image is pass to another convolution layer or to a pooling layer.

### 4.1.3. Pooling

Pooling layers help in reducing the size of the image to make the operations less expensive, there two types of pooling layers used in Convolutional Neural Networks,

Max Pooling and Average pooling, the concept of pooling is to select a value from a window of some size $(k \times k \times k)$ from the input image of size $(N \times N)$ and use that value as the representative of these value in the new image $(N / k \times N / k)$. this method is also commonly known as sub-sampling, in max pooling the maximum value from the window is chosen while in average pooling average of all the values in the window is taken.

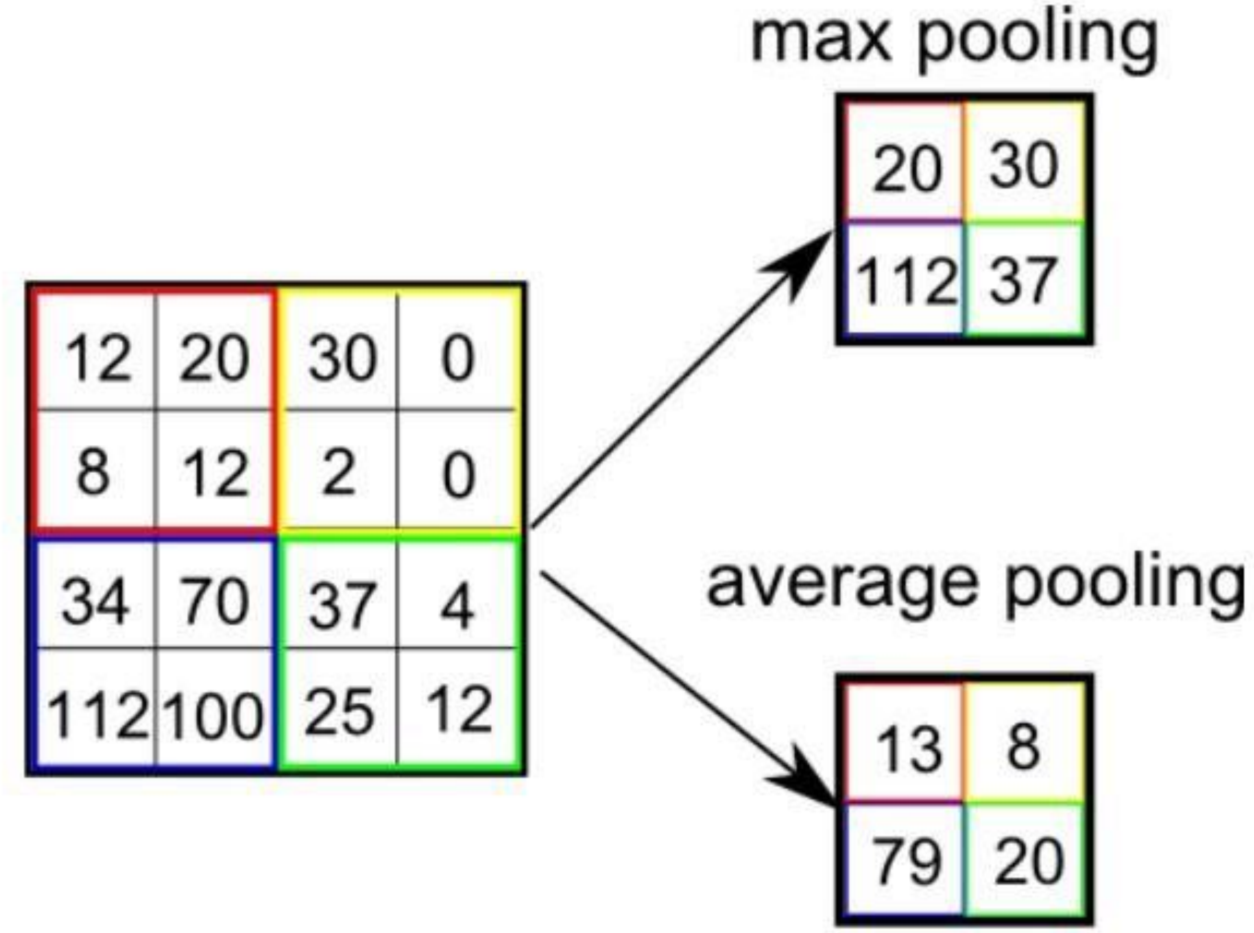


Figure 2 Illustration of 2D pooling (max and average) [27]

This process of stacking convolutional and pooling layers is applied a several times alternatively to extract features from the image then the resultant feature map is passed to the fully connected layers which learn the non-linear representation of the high-level feature map extracted from convolutional layer.

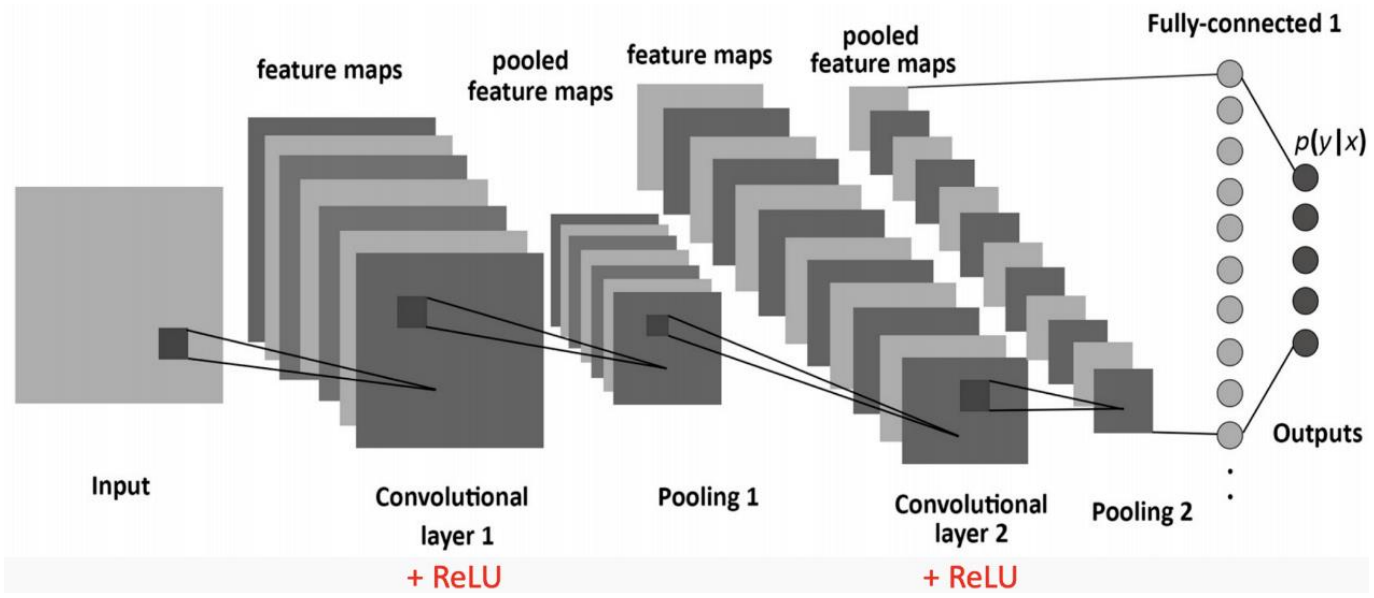


Figure 3  A Convolutional Neural Network [26]

### 4.1.4. 3D Convolution

In 3D Convolutions the input is a 3D feature map or several 3D feature maps, which make 3D Convolutions extract high-level features from the spatial as well as temporal dimension, which makes them better for video analysis then 2D Convolution.

The principal of 3D convolution is similar to that of 2D i.e. to extract high-level features from the input data using a 3D kernel instead of a 2D. The value of *j*th feature map on the position (*x*, *y*, *z*) in the *i*th layer can be formally written as:

$$\mathrm{v}_{\mathrm{ij}}^{\mathrm{xyz}} = b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijm}^{pqr} \, v_{(i-1)m}^{(x+p)(y+q)(z+r)}$$

$$a = max(\, v, 0\,)$$

Where, $R_i$ is the size of the kernel in the third dimension (temporal), $w_{ijm}^{pqr}$ is value at position ($p$, $q$, $r$) of the kernel connected to the feature map $m$ in the previous layer.

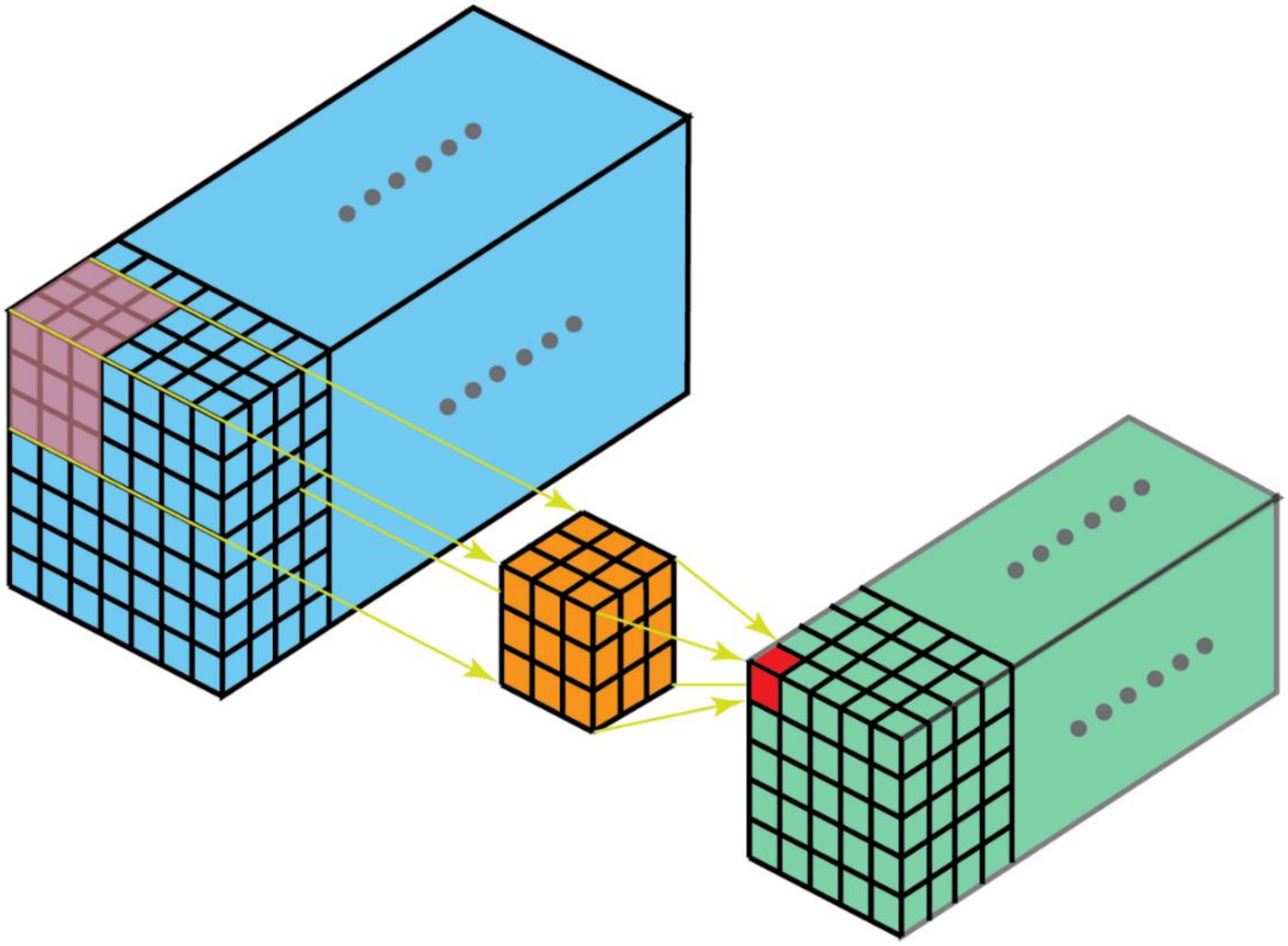


Figure 4 Illustration of a 3D convolution [28]

### 4.1.5. 3D Pooling

3D pooling is applied on a 3D window of the feature map instead of 2D. Just like 2D pooling, 3D pooling is also used for reducing the computation cost by using one value to represent several values in a window, just that this window is $(\mathrm{k} \times k \times k)$ and the output is feature maps $(\mathrm{N} / \mathrm{k} \times N / \mathrm{k} \times N / \mathrm{k})$.

[29]

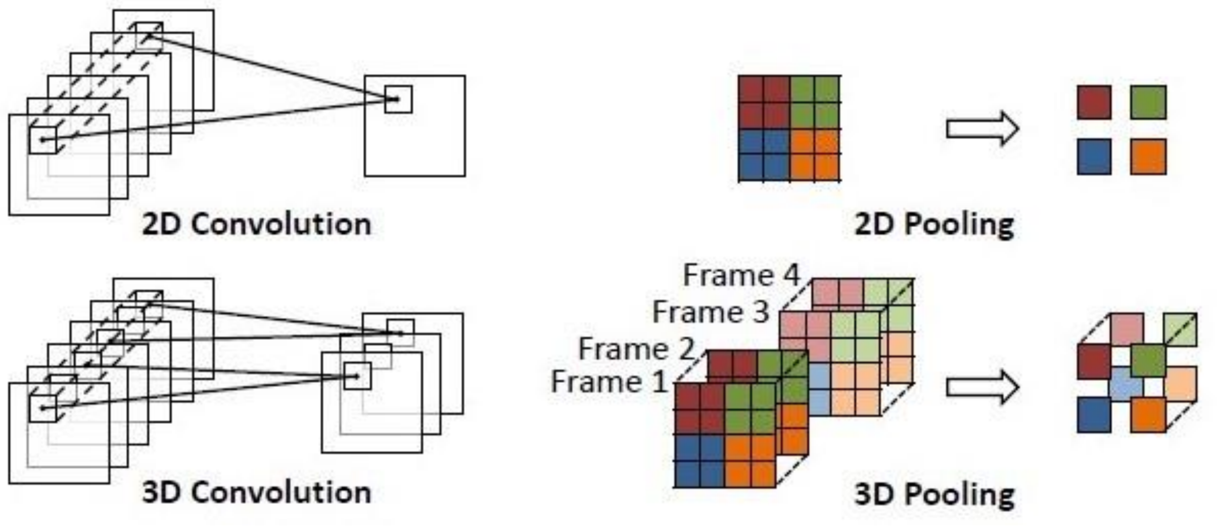


Figure 5 A comparison between 2D and 3D.

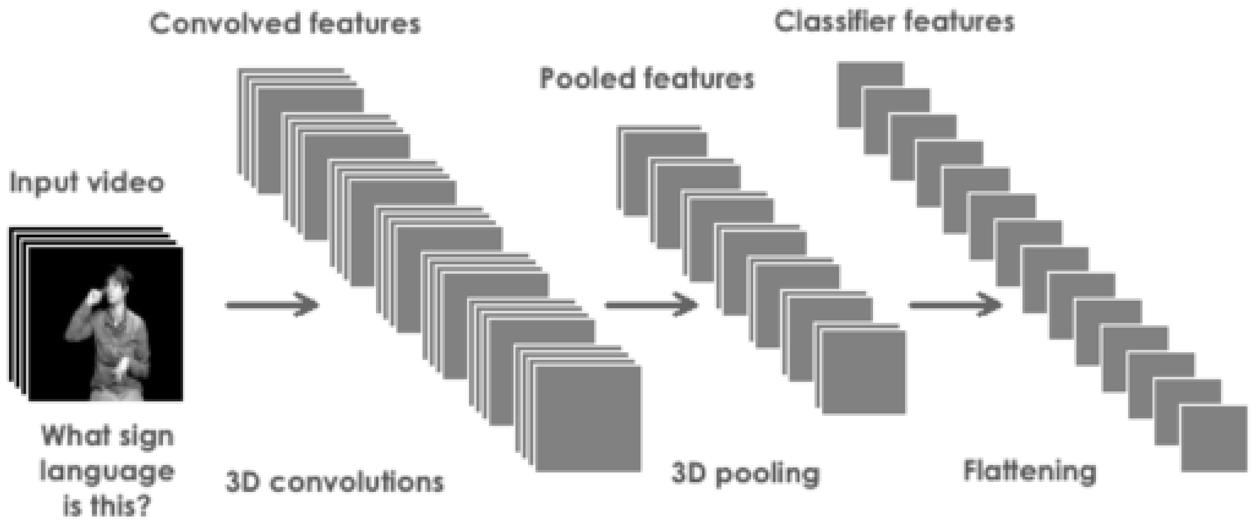


Figure 6 A 3D Convolutional Neural Network [30]

## 4.2. Loss Function

In Machine learning, the network learns by the concept of having a value(s) from a certain loss function to evaluate how good the specific architecture is performing on the given data. Loos function compares the actual value with the predicted value. It will report a high number if the deviation is too much and a small number if the predicted output is near to the actual output. Since Deep learning problems can be divided into two types, either classification or regression; since our work focuses on regression, we will only cover the regression losses.

### 4.2.1. Mean Square Error (L2 Loss)

Mean square error is the average of difference between the predicted output and actual output and the final value squared. Since the actual error is squared, the prediction is quite far away from the actual values.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}\left(x_j - y_j\right)^2$$

### 4.2.2. Mean Absolute Error (L1 Loss)

It is the average sum of absolute difference between the predicted values and the actual values. Since it doesn’t use the concept of squaring the errors, it is robust to outliers.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}\left|x_j - y_j\right|$$

### 4.2.3. Mean Bias Error

This isn't much commonly used in Machine learning domain. Its almost same like L1 loss, the only difference is that no absolute of value is taken. It is not much accurate but it can determine if the current model has a positive or a negative bias.

$$\boldsymbol{MAE = \frac{1}{n}\sum_{j=1}^{n}\left(x_j - y_j\right)}$$

## 4.3. Optimizer

Optimizers are used while training the model to update the weight parameters. Loss functions act as input to the optimizer to indicate if the model training is moving towards local minima or not.

There are lots of common optimizers **[31]** but we will only discuss the most used ones.

### 4.3.1. Stochastic Gradient Descent

Stochastic Gradient Descent only uses the loss gradient of one training example at each iteration, it doesn't use the sum of loss gradient of all training examples. Data shuffling is required in its usage. Since it uses only one training example at a time, its path towards the minima is very noisy and very random.

$$\boldsymbol{W = W - \alpha . dw}$$

$$\boldsymbol{b = b - \alpha . db}$$

Where, $(\boldsymbol{\alpha})$ is the learning rate, $\boldsymbol{dw}$ and $\boldsymbol{db}$ are derivatives of weight $\boldsymbol{W}$ and bias $\boldsymbol{b}$.

### 4.3.2. RMSprop

Stands for root mean square propagation. It resolves adagrad's vanishing learning rate by using moving average of squared gradient. Learning rate gets adjusted on its own and its choses a different learning rate for each parameter.

$$\boldsymbol{v_{dw} = \beta . v_{dw} + (1 + \beta). dw}$$

$$\boldsymbol{v_{db} = \beta . v_{dw} + (1 + \beta). db}$$

Where, $(\boldsymbol{v_{dw}})$ and $(\boldsymbol{v_{db}})$ are the gradient updates for the weights $(\boldsymbol{W})$ and bias $(\boldsymbol{b})$, $(\boldsymbol{\beta})$ is the momentum.

$$\boldsymbol{W = W - \alpha . \frac{dw}{\sqrt{v_{dw}} + \epsilon}}$$

$$\boldsymbol{b = b - \alpha . \frac{db}{\sqrt{v_{db}} + \epsilon}}$$

Where, $(\boldsymbol{\alpha})$ is the learning rate, $(\boldsymbol{\epsilon})$ is the weight decay.

### 4.3.3. Adam

Adam is adaptive moment estimation. It computes the individual adaptive learning rates for every parameter. These learning rates are estimated from the moments of the gradients. It also helps overcome the problem of vanishing learning rates. It is computationally effective hence requiring very less memory.

$$\boldsymbol{m = \beta_1 . m + (1 - \beta_1). g}$$

$$\boldsymbol{v = \beta_2 . v + (1 - \beta_2). g^2}$$

Where, $\boldsymbol{m}$ and $\boldsymbol{v}$ are moving averages of gradient $\boldsymbol{g}$, $\boldsymbol{\beta_1}$ and $\boldsymbol{\beta_2}$ are hyper-parameters with default values of 0.9 and 0.999.

$$\boldsymbol{\widehat{m}} = \frac{\boldsymbol{m}}{(\boldsymbol{1} - \boldsymbol{\beta}_{\boldsymbol{1}}^{\boldsymbol{t}})}$$

$$\boldsymbol{\widehat{v}} = \frac{\boldsymbol{v}}{(\boldsymbol{1} - \boldsymbol{\beta}_{\boldsymbol{2}}^{\boldsymbol{t}})}$$

Where, $\boldsymbol{\widehat{m}}$ and $\boldsymbol{\widehat{v}}$ are bias corrected 1st and 2nd momentum, $\boldsymbol{t}$ is the current iteration.

$$\boldsymbol{w} = \boldsymbol{w} - \boldsymbol{\eta}\frac{\boldsymbol{\widehat{m}}}{\sqrt{\boldsymbol{\widehat{v}}} + \boldsymbol{\epsilon}}$$

Where, $\boldsymbol{\eta}$ is the step size, $\boldsymbol{\epsilon}$ is the weight decay and $\boldsymbol{w}$ are the weights.

## 4.4. Weight initialization

While training a model, it is important to randomly initialize the weight to ensure faster convergence. Initializing weights, the right way is very important, if done wrong it can cause problem of vanishing or exploding gradients. There are several ways to initialize the weights of layers. We will only discuss only a few.

### 4.4.1. Zeros

In zero initialization, all the layers all initialized. This might be problematic as same all the errors propagated through the network will same hence affecting the learning process.

### 4.4.2. Random Normal

It initializes the tensors with a normal distribution.

### 4.4.3. Xavier Normal Initializer

It initializes the weights in network by drawing them from a distribution with zero variance and also a specific variance.

$$\boldsymbol{f}(\boldsymbol{W}) = \frac{\mathbf{2}}{\boldsymbol{n_i} + \boldsymbol{n_o}}$$

Where, $\boldsymbol{W}$ are the weights and $\boldsymbol{n_i}$ and $\boldsymbol{n_o}$ are the number of neurons in input and output layers.

## 4.5. Proposed Method

### 4.5.1. V3C-Net

We are working towards solving a spatial temporal problem. We want to identify a vehicle only out of a real-world environment having other objects as well and we want to count the number of vehicles for a certain time frame as well. 3D CNN seems the best solution as it will help preserve both the spatial as well as temporal data.

We use a complete 3d Convolutional Neural Network (See figure 8). One half of the network extracts the vehicular features from the video and the latter half counts the vehicles from the vehicular features. **[24]**extensively researched and reported the results of 3D convolutional neural network using various size of convolutional kernels. The experiment was carried out on six different benchmarks. Paper reported that convolution network with a homogenous kernel size of 3x3x3 outperformed on 4 benchmarks. Keeping these results in mind and after enough self-experimentation as well, we designed the network to have 3x3x3 kernel size. We also did brief experimentations with varying input sizes and with varying convolutional neural network architecture designs. The network with the best result is chosen for the paper submission and also for thesis.

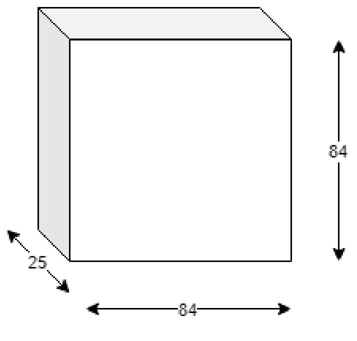


Figure 7 Input Shape

We named the network V3C net; V standing for vehicles, 3 for 3d convolutional network and C for count. The network has eight convolution layers responsible for finding the required vehicular features, five pooling layers to minimize the size of features to make computations faster and seven fully connected layer in end to create a fully connected neural network to count the vehicles from the extracted vehicular features.

The input is of 3d shape (see Figure 7). Spatial domain dimensions are 84x84. The images are really huge in size and they are resized to accommodate our required dimensions. Each input sample contains 250 frames extracted from 25 fps videos. To make up for 25 temporal dimension, every tenth frame is chosen sequentially. All of the input sample is reshaped and send to network as the network's input.

The first convolution layer outputs sixty-four feature maps. The layer has 5248 trainable parameters. The next layer is pooling layer, a max pooling is carried out spatially only on the input from the previous first convolutional layer. The spatial features aren't pooled as we want to preserve the temporal data at this point and not loose it too soon hence affecting the count result.

This Layer is next followed by convolution layer with a 3x3x3 kernel size outputting 256 feature maps. It has 221312 learnable parameters. From this layer onwards, the rest of the pooling layers have the same 2x2x2 kernel size. The 3a and 3b have 884992 and 1769728 learnable parameters respectively. The next convolution layer after pooling has 3539456 trainable parameters. The next three convolution layers have 7078400 parameters. The feature extractor part of our network has a total parameter of 27,655,936 that can be trained. The fully connected or dense are all interconnected with each. This causes a huge increase in the number of parameters. The last seven fully connected layers have total parameters of 102,781,953 trainable parameters.
Before the last pooling layer, zero padding is applied spatially only. Adding it to the temporal data might cause redundancy in final vehicular count.

The fully connected first layer eventually receives a feature map sized 8192. The output dimension from the last layer of dimension 1x4x4x512 is flattened before passing it to the fully connected layers. This network of fully connected layers will learn to identify vehicular features and count them. The complete architecture is shown in the table (Table number)

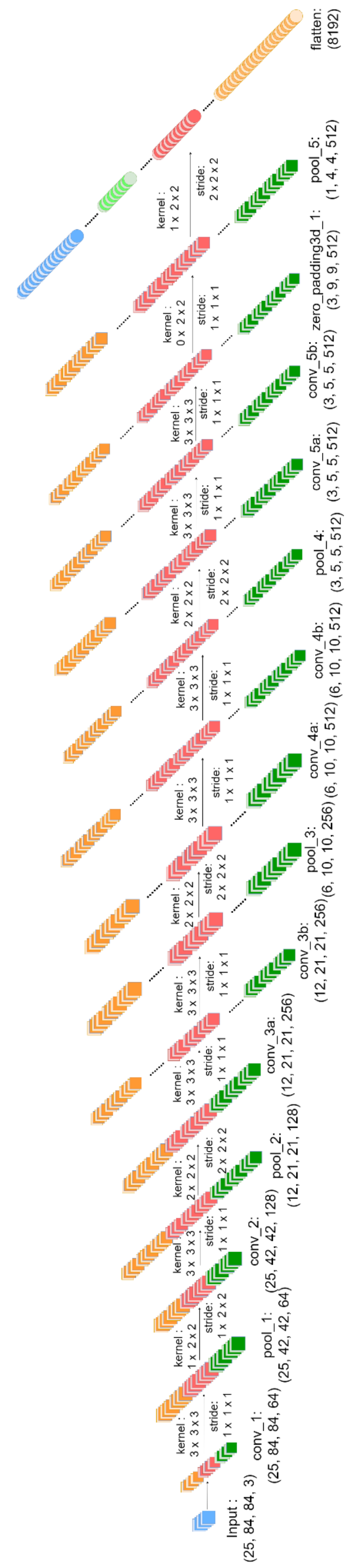


Figure 8 V3CNet Architecture

| ***Layer*** | ***Kernel Size (depth, height, width)*** | ***Output shape (depth, height, height, width, output channels)*** |
|---|---|---|
| Conv-1 | 3x3x3 | 25,84,84,64 |
| Max-pool-1 | 1x2x2 | 25,42,42,64 |
| Conv-2 | 3x3x3 | 25,42,42,128 |
| Max-pool-2 | 2x2x2 | 12,21,21,256 |
| Conv-3 | 3x3x3 | 12,21,21,256 |
| Conv-4 | 3x3x3 | 12,21,21,256 |
| Max-pool-3 | 2x2x2 | 6,10,10,256 |
| Conv-5 | 3x3x3 | 6,10,10,512 |
| Conv-6 | 3x3x3 | 6,10,10,512 |
| Max-pool-4 | 2x2x2 | 3,5,5,512 |
| Conv-7 | 3x3x3 | 3,5,5,512 |
| Conv-8 | 3x3x3 | 3,5,5,512 |
| Zero padding | 0x2x2 | 3,9,9,512 |
| Max-pool-5 | 2x2x2 | 1,4.4.512 |
| Flatten | ~ | ~ |
| Fully-connected 1 | ~ | 8192 |
| Fully-connected 2 | ~ | 4096 |
| Fully-connected 3 | ~ | 4096 |
| Fully-connected 4 | ~ | 4096 |
| Fully-connected 5 | ~ | 4096 |
| Fully-connected 6 | ~ | 512 |
| Fully-connected 7 | ~ | 1 |

Table 2 Architecture details

### 4.5.2. Training the network

To train the network, we used RMSprop for the optimizer. No matter how small the gradients are, RMSprop makes sure that gradients don't die and algorithm training doesn't halt for any reason. Since our data output is a number and the number varies a lot. Sometimes it zero and sometimes it can go as high as 50 plus number of cars. Training of such network can be quite instable.

Mean Absolute Error was our choice for cost function. We want to predict the number closer to the original number as much as we can since it's a regression problem. Having an accurate number is the goal. Mean Absolute Error tries to keep prediction closer to the actual output.
The tensors were initialized using Xavier initialization to help us reach the local minima faster.
The model was trained on Nvidia Dgx work-station. The implementation was done on python using Keras library.

# CHAPTER 5: RESULTS AND EXPERIMENTS

The dataset mentioned in section 3 was used for training and testing. In training and experiment phases, the biggest challenge was finding the right architecture and right best performing settings. Training the network also brought its own issues as initially we faced issues regarding vanishing gradients.

We divided the network into two separate parts for training. All the layers before flatten do the work of feature extractor. The layers after flatten learn to identify the vehicular feature who go out of frame and count them. The training was done using the freezing and unfreezing of layers.
For training our architecture, dataset of 10 second video length sequence was used. Each sequence had 250 frames, we sequentially picked every 10th frame, the images were resized to 84x84. In the end, each chunk has a final shape of 25x84x84x3. This 3d shape is fed into our 3D CNN for training. Since 3D CNN's are very quick to train, the whole network was trained in 8 epochs only.

Testing was done on two different types of video lengths i.e. Static length and Variable length Sequence. Variable length testing was challenging on the trained model. For variable length sequence, each chunk was subdivided into chunks of 50. Each chunk was separately processed; in chunk of 50 images, every second image was picked to get a total 25 images. The chunk was resized and reshaped into shape of 25x84x84x3. Once all the sub-chunks belonging are processed, the count is aggregated in the end for that variable length chunk.

# CHAPTER 6: RESULTS AND DISCUSSIONS

We compared our result with a baseline method which uses the technique of vehicle detection, tracking and then counting it. We used to-the date best object detection algorithm for vehicle detection. The input images were annotated using the bounding box technique. The algorithm was trained till it reached a mean average precision error of 0.80. the calculated metrics were input to the Hungarian algorithm. The tracked vehicles are counted once they cross a virtual line.

Table 3 : Results Comparison

| ***Method*** | ***MAE*** | ***SD*** |
|---|---|---|
| Baseline | 3.40 | 4.29 |
| V3C-NET ~ Static length | **1.39** | **1.72** |
| V3C-NET | **2.43** | **3.11** |

Our algorithm outperforms the baseline method. Our method needs no preprocessing and its pretty much robust to illumination changes, car/vehicle orientation and object occlusion.

Here are some of the results reported:

Table 4 : Static Length Sequence Results

| Actual Vehicular count | Predicted Vehicular count |
|---|---|
| 2 | 2.93 |
| 1 | 1.44 |
| 4 | 3.31 |
| 0 | 0.91 |
| 6 | 5.00 |
| 4 | 4.05 |
| 2 | 2.80 |
| 5 | 5.05 |
| 3 | 3.79 |
| 0 | 0.0004 |
| 2 | 2.09 |
| 6 | 5.63 |
| 3 | 2.86 |
| 0 | 0.0004 |
| 4 | 4.02 |
| 1 | 1.07 |
| 2 | 2.32 |
| 5 | 4.43 |

Table 5: Variable Length Sequence Results

| Actual Vehicular Count | Predicted Vehicular Count |
|---|---|
| 20 | 17.59 |
| 0 | 0.046 |
| 0 | 0.0007 |
| 10 | 11.41 |
| 1 | 1.67 |
| 2 | 2.22 |
| 20 | 18.55 |
| 4 | 4.19 |
| 4 | 5.00 |
| 14 | 12.41 |
| 24 | 26.03 |
| 5 | 4.77 |
| 1 | 1.01 |
| 8 | 9.17 |
| 9 | 9.55 |
| 4 | 4.05 |
| 6 | 7.98 |
| 3 | 3.51 |
| 10 | 10.63 |
| 14 | 13.50 |
| 10 | 9.73 |

# CHAPTER 7: CONCLUSION AND FUTURE WORK

We have proposed a robust network which provides satisfactory results to estimate the vehicular traffic. The importance of vehicular data is briefly discussed in chapter 1. Our algorithm doesn’t need to detect and track the moving vehicle. It doesn’t need any preprocessing hence proving the effectiveness of CNN based solutions.

We also annotated data for vehicular count for each type of vehicle e.g. in a 10 second chunk we may have 2 bikes, 12 cars, 4 vans and 1 truck. In the future, we will experiment with classification of vehicle type and then count the vehicles according to its type hence classification plus regression both done at the same time.

# APPENDIX-A

***Pseudo code ~ Training Phase***

```
 func read_csv:
        subfolder_path = []
        count = []
        Open file as csv file:
                Append row named folder to subfolder_path list
                Append row named count to count list
return subfolder_path and count list
```

```
func get_list_images(folder_name):
        list_images= []
        for all images in folder_name:
                if images name end with jpg:
                        Append images to list_images
Naturally sort the list_images
return list_images
```

```
func read_images(folder_path, array_image_file_path):
    image_array = []
    read_ith_frame= 10
    while( length of folder_path contents):
        read images skipping every 10th
        resize the images to 84,84
        Append image into image_array as datatype float32
return Image_array
```

```
func main():
    subfolder_path, count = read_csv()
    for all data in folder:
        img_list = Get_list_images()
        read_images(subfolder_path, img_list)
        reshape image to 1x25,84x84x3
        model = train(reshaped_image, count)
        save model
        save weights
```

***Pseudo code~ testing phase~ Static length***

```
load model
read test csv
for each test sample:
    reshape the test sample
    predict on model
    write original count in a file
    write predicted count in a file
```

***Pseudo code~ testing phase~ Variable length***

```
load model
read test csv
for each test sample:
    divide it into chunk of 50
        for each chunk
            reshape the test sample
            predict on model
            append count into a list
    add all the elements in the count list
write original count in a file
write predicted count in a file
```

# References

[1] "The first automobile (1885–1886)," [Online]. Available: https://www.daimler.com/company/tradition/company-history/1885-1886.html.

[2] "Vehicle Sensing: Ten Technologies to Measure Traffic," [Online]. Available: https://www.windmill.co.uk/vehicle-sensing.html.

[3] A. D. J. P. G.-C. P. M. C. a. F. J. V.-A. Jose J Lamas-Seco, "System for vehicle classification: Hardware prototype and off-line signal proccesing," in *IEEE EUROCON 2015-International Conference on Computer as a Tool*, 2015.

[4] H. T. a. H. H. R. Walid Balid, "Intelligent vehicle counting and classification sensor for real-time traffic surveillance.," in *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[5] S. T. a. R. Rajamani, "Portable roadside sensors for vehicle counting, classification, and speed measurement.," in *IEEE Transactions on Intelligent Transportation Systems*, 2014.

[6] A. A. e. al, "Gaussian mixture models optimization for counting the numbers of vehicle by adjusting the region of interest under heavy traffic condition," in *International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 2015.

[7] S. K. a. R. Safabakhsh, "Vehicle detection, counting and classification in various conditions," in *IEEE Intelligent Transport Systems*, 2016.

[8] M. a. S. C. Rahnemoonfar, "Deep count: fruit counting based on deep simulated learning," *Sensors*, vol. 17, p. 905, 2017.

[9] Y. a. F. G. a. K. F. a. S. A. a. H. A. B. Itzhaky, "Leaf counting: Multiple scale regression and detection using deep cnns," *CVPPP*, 2018.

[10] W. a. F. H. a. Y. L. a. C. A. Li, "Deep Learning Based Oil Palm Tree Detection and Counting for High-Resolution Remote Sensing Images," *Remote Sensing*, vol. 9, p. 22, 2017.

[11] N. a. W. U. a. N. C. a. K. K. a. O. N. Seenouvong, "A computer vision based vehicle detection and counting system," pp. 224-227, 2016.

[12] Y. a. Z. X. a. C. D. Li, "Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[13] M. A. Abdelwahab, "Accurate Vehicle Counting Approach Based on Deep Neural Networks," in *International Conference on Innovative Trends in Computer Engineering (ITCE)*, 2019.

[14] A. a. C. M. Ramanathan, "Spatiotemporal Vehicle Tracking, Counting and Classification," in *IEEE Third International Conference on Multimedia Big Data (BigMM)*, 2017 .

[15] S. a. M. M. A. a. R. J. a. T. H. Bouaich, "Vehicle counting system in real-time," in *International Conference on Intelligent Systems and Computer Vision (ISCV)*, 2018.

[16] M. B. a. W. E. P. Subaweh, "Implementation of pixel based adaptive segmenter method for tracking and counting vehicles in visual surveillance," in *International Conference on Informatics and Computing (ICIC)*, 2016.

[17] Y. a. B. L. a. B. Y. a. H. P. a. o. LeCun, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998.

[18] A. a. S. I. a. H. G. E. Krizhevsky, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.

[19] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015.

[20] S. a. H. K. a. G. R. a. S. J. Ren, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015.

[21] S. S. L. T. B. a. O. R. Ahmed Abdulkadir, "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," *CoRR,* 2016.

[22] S. P. A. A. V. V. Casamitjana Adria, "3D Convolutional Neural Networks for Brain Tumor Segmentation: A Comparison of Multi-resolution Architectures," in *Second International Workshop, BrainLes*, 2016.

[23] D. M. a. S. Scherer, "A 3D Convolutional Neural Network for Real Time Object Detection," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 922-928, 2015.

[24] L. D. B. R. F. L. T. M. P. Du Tran, "Learning Spatiotemporal Features with 3D Convolutional Neural Network," *IEEE International Conference on Computer Vision ( ICCV)*, pp. 4489-4497, 2015.

[25] L. B. Yann LeCun, "Gradient-Based Learning to Document Recognition," *Proceedings of IEEE,* pp. 2278-2324, 1998.

[26] "towardsdatascience," [Online]. Available: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac .

[27] "towardsdatascience," [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[28] "towardsdatascience," [Online]. Available: https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215.

[29] H. N. H. L. S. Q. Fan, "Reconfigurable Acceleration of 3d-CNNs for Human Action Recognition with Block Floating-Point Representation," in *28th International Conference in Field Programmable Logic and Application*, 2018.

[30] [Online]. Available: http://acl2014.org/acl2014/P14-2/xml/P14-2061.xhtml.

[31] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms," *CoRR,* vol. abs/1609.04747, 2016.