Automated Analysis of Videos Towards Compliance Monitoring

A PROJECT REPORT

Submitted by

Mr. Suraj Nandkishor Kothawade

in partial fulfillment of
the requirements for the award of the degree
of

Bachelor of Technology
In
Computer Science and Engineering

Under the guidance of

Prof. Ganesh Ramakrishnan
Prof. A. M. Bainwad

At

SHRI GURU GOBIND SINGHJI INSTITUTE OF ENGINEERING AND TECHNOLOGY
VISHNUPURI, NANDED
(MAHARASHTRA STATE)
PIN 431 606 INDIA

May 2018

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

APPROVAL SHEET

This project entitled "**Automated Analysis of Videos Towards Compliance Monitoring**" by **Mr. Suraj Kothawade, 2014BCS066** from **Shri Guru Gobind Singhji Institute of Engineering and Technology, Nanded** is approved that the internship is carried out under **Prof. Ganesh Ramakirshnan's** guidance at the **Department of Computer Science and Engineering, Indian Institute of Technology, Bombay** during 18th December, 2017 to 10th May, 2018 successfully.

_____

Guide
Prof. Ganesh Ramakrishnan

Place: IIT Bombay, Mumbai
Date: 10th May, 2018

Department of Computer Science and Engineering

SHRI GURU GOBIND SINGHJI INSTITUTE OF ENGINEERING AND TECHNOLOGY, NANDED

Certificate

This is to certify that **Suraj Nandkishor Kothawade (2014BCS066)** has been carried out the project work entitled ***Automated Analysis of Videos Towards Compliance Monitoring*** for the award of degree of Bachelor of Technology in Computer Science and Engineering from Shri Guru Gobind Singhji Institute of Engineering & Technology, Vishnupuri, Nanded (M. S.) under my supervision. The project embodies results of original work, and studies are carried out by the student himself and the contents of the project work do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/ Institution.

Guide	**Head**
Mr. A.M. Bainwad	**Prof. Dr. U. V. Kulkarni**

Date:

Acknowledgements

Prima facea, I am grateful to the God for the good health and wellbeing that were necessary to complete this work. I want to thank my guide **Prof. A.M. Bainwad** for his continuous help and generous assistance. He helped in a broad range of issues from giving me direction, helping to find the solutions, outlining the requirements and always having the time to see me.

My deepest thanks to **Dr. U.V Kulkarni** for encouraging me time and again to pursue research and giving me a strong foundation in Machine Learning and Neural Networks.

Furthermore, I would like to express my sincere gratitude to my advisor **Prof. Ganesh Ramakrishnan** for the continuous support of my study and related research, for his patience, motivation, and immense knowledge. I could not have imagined having a better advisor and mentor for my work.

Besides my advisors, I would like to thank **Dr. Rishabh Iyer** and **Mr. Vishal Kaushal** for introducing me to various real-world problems in the intersection of Computer Vision and Machine Learning and guiding me to try multiple approaches to solve them. Their commendation truly kept me motivated at all times. I am really grateful to Rishabh for being a caring friend and guide.

My sincere thanks also goes to **Mr. Anurag Sahoo**, **Mr. Rohan Mahadev** and **Mr. Pratik Dubal** for mentoring me and the bonds of brotherhood. Special thanks to Pratik for helping me out in all situations. I also thank my friends **Manasi, Kunjan, Sahil** and **Furkhan** for their friendship, faith and the fun we had in college.

I would like to express my immense gratitude to my parents **Nandkishor** and **Ujwala**, for their unconditional love, encouragement and support, and for being there with me physically,emotionally and mentally, every single day of my life. I sincerely thank them for careful nurturing and upbringing they provided to mould me into what I am, and for being there with me whenever I needed them.

Suraj Nandkishor Kothawade
2014BCS066

Contents

List of Tables

List of Figures

Abstract

In this report, we explore various techniques from machine learning, deep learning and computer vision in order to solve peculiar real-world problems in analysing diverse videos. We illustrate various state-of-art methodologies and pipelines using convolutional neural networks that can be commonly used to solve domain specific challenges in object detection and classification, scene classification, etc. We primarily focus on feature engineering, fine tuning, transfer learning, region based and single pass object detection models. For each of the mentioned techniques, we investigate in-depth their relevance from a theoretical as well as application perspective. Towards this end, we focus on the efficacy of customized deep learning models using data from the deployment scenarios and compare them with off-the-shelf models using generic data from the Internet. Finally, since most of these models are deployed on-site where resources are limited, it is imperative to have models which do not require GPUs. We exhibit customization of resource constrained models and deploy them on embedded devices without substantial loss in accuracy.

To demonstrate the effectiveness of these techniques we consider multiple compliances from a real-world classroom scenario and develop an understanding on the pros and cons of using these measures. The following challenges are considered:

1. Counting the number of students precisely
2. Classifying classroom scenarios into class started | not started at a frame level
3. Detecting if each student is wearing uniform

Chapter 1

Introduction

1.1 Video Analytics — The Need of the Hour

Visual Data, in the form of images, videos and live streams, has been growing at an unprecedented rate in the last few years. While this massive amount data is a blessing for Data Science, as it helps in improving the predictive accuracy, it is also a curse since humans are unable to consume this large amount of data. Moreover, today, machine-generated videos (via Drones, Dash-cams, Body-cams, Surveillance cameras etc.) are being generated at a rate higher than what we as humans can process. Among machine-generated videos, surveillance videos are one of the largest contributors to this growth. Surveillance cameras are deployed in several verticals, including office facilities, road intersections for traffic monitoring, ATMs and Banks, Hospitals, Manufacturing Facilities, Industrial Plants, Construction Sites, Educational Institutions, Retail stores and Malls, Hotels and Restaurants etc. Each of these verticals have their own unique video analytics applications. In most scenarios, video analytics is used for security purposes (detecting loitering and intrusion, asset tampering, suspicious activity, object detection or face recognition). In other scenarios, video analytics is used for process compliance, e.g. if an event in a manufacturing plant has happened on time, or whether it was done as desired. In retail scenarios and hotels, the information from video analytics is used for getting insights in customer pattern (e.g. heat-map, flow-map, counts, dwell-times etc.) In Scene classification cases like classrooms, where it is needed to detect when the class has started based on the different parameters that help differentiate multiple scenarios. While all these applications sound very different, the analytics building blocks are the same.

Figure 1.1 demonstrates the process clearly. The analytics engine consists of several building blocks, including object detection, tracking, face and human detection, human and face sub-attribute recognition, vehicle detection and vehicle sub-attribute recognition etc. The information from the analytics engine

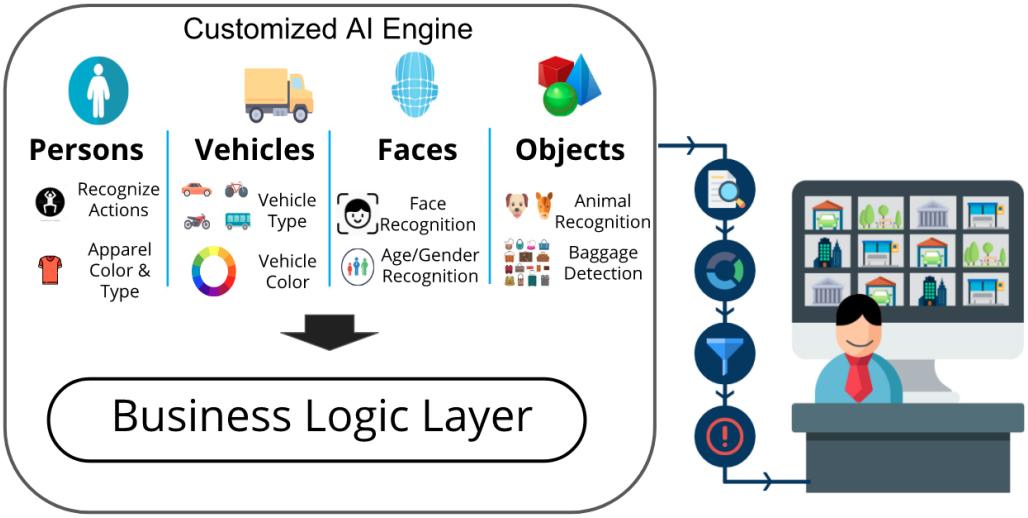

Figure 1.1: End-to-End process for analytics

is then passed on to a business logic layer, which applies rules based on the analytics results to produce the required output. For example, using human detection (localizing where a human is in the video frame) if a human enters a demarcated area, it sends out a real-time alert. Similarly, by tracking the paths of the human in the video, we can compute the heat-map and flow-map of human movement.

The following sections outline the advancement of deep learning in computer vision, followed by the recent advances and challenges of video analytics for surveillance applications.

1.2 Advancement of Deep Learning in Computer Vision

Current approaches to all but a few Computer Vision tasks involve the use of Deep Convolutional Neural Networks (CNNs). CNNs generated a lot of interest after the successful performance of AlexNet [14] in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 competition [24]. Following its triumph, there was an upsurge in the number of deep CNN models that were being used across the Computer Vision community. The winner of ILSVRC 2014 was the even deeper GoogLeNet, which was the first CNN model to have a fundamentally different architecture than AlexNet. It was followed by ResNet [7], the winner of ILSVRC 2015, which was an astonishing 152 layers deep. It won the competition by achieving an error rate of 3.57%, beating humans at the image

classification task.

Similarly, for Object Detection tasks, there has been a significant advancement in the use of CNNs in the last lustrum. It started with the introduction of the Region based family of networks [3, 2, 23]. Recently, the Single Pass family of networks, consisting of YOLO and Tiny Yolo [22, 21], along with the Single Shot MultiBox Detector (SSD) [18] have emerged as the state-of-the-art models in object detection. Their extremely fast inference times allow for object detection to take place in real-time, thus broadening the areas of application where Deep Learning can be used for vision tasks.

While there has been a remarkable improvement in the performance of Object Detection models for real-time tasks, an important role is still played by Object Tracking algorithms. Tracking algorithms are much faster than detection algorithms and help preserve the identity of an object being tracked when detection fails. Object Tracking is highly dependent on the quality of object detections. With good detections, the performance of a simple tracking algorithm increases drastically.

The following subsections give a brief overview on a few initial Deep Convolutional Neural Networks and Object Detection models.

1.2.1 Deep Convolutional Neural Networks

Convolutional neural networks are deep artificial neural networks that are used primarily to classify images (e.g. name what they see), cluster them by similarity (photo search), and perform object recognition within scenes. They are algorithms that can identify faces, individuals, street signs, tumors, platypuses and many other aspects of visual data.

Convolutional networks perform optical character recognition (OCR) to digitize text and make natural-language processing possible on analog and handwritten documents, where the images are symbols to be transcribed. CNNs can also be applied to sound when it is represented visually as a spectrogram. More recently, convolutional networks have been applied directly to text analytics as well as graph data with graph convolutional networks.

The efficacy of convolutional nets (ConvNets or CNNs) in image recognition is one of the main reasons why the world has woken up to the efficacy of deep learning.[15] They are powering major advances in computer vision (CV), which has obvious applications for self-driving cars, robotics, drones, security, medical diagnoses, and treatments for the visually impaired.

Since object detection is one of the core problems in computer vision that people have cared about for a very long time, there has been monolithic amount of progress. However, deep neural networks have proved to be a path breaking solution to this problem. 1.2 shows the progression of object detection performance on the Pascal VOC [1] dataset. The performance started to stagnate up until 2012 and in 2013 when one of the first deep learning approaches in object

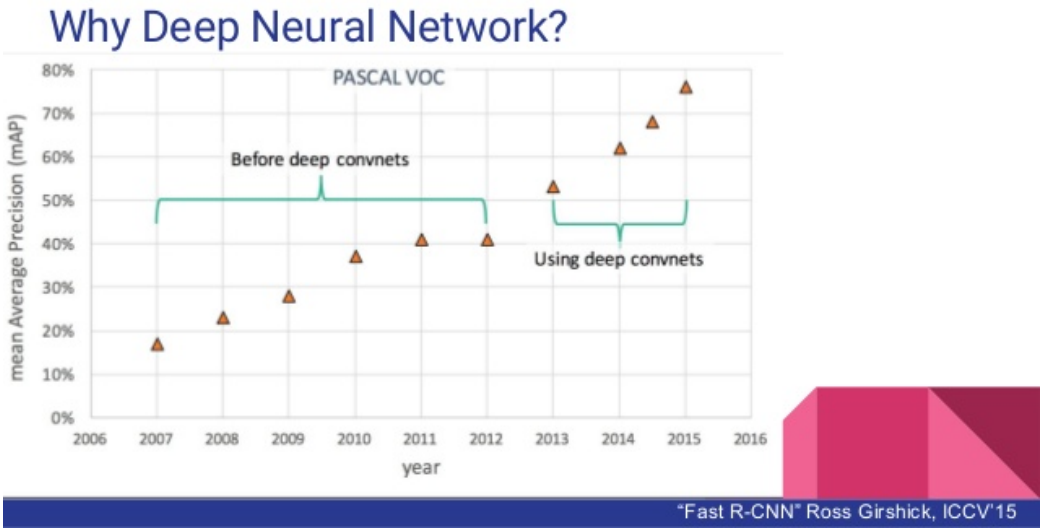

Figure 1.2: Impact of deep learning on object detection[2]

detection came around, the performance shot up very quickly and got better and better ever since.

1.2.1.1 AlexNet — 2012

Even though some say that LeCun et al. [16] in 1998 was the real pioneering publication, AlexNet [14] is the most fundamental network architecture in the deep learning community.

This neural network developed by Krizhevsky, Sutskever, and Hinton in 2012 was the coming out party for CNNs in the computer vision community. It was the first time a model performed so well on a historically difficult ImageNet

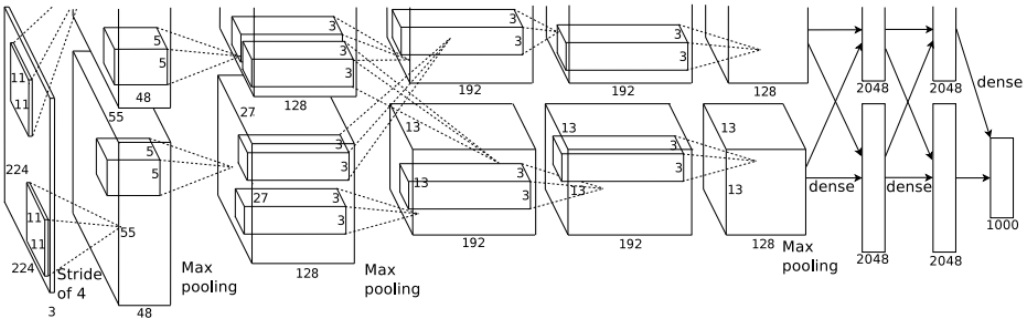

Figure 1.3: Architecture of AlexNet[14]

dataset. Utilizing techniques that are still used today, such as data augmentation and dropout, this paper really illustrated the benefits of CNNs and backed them up with record breaking performance in the ImageNet competition.

The most imperative factors that define AlexNet are as follows:

- Trained the network on ImageNet data, which contained over 15 million annotated images from a total of over 22,000 categories.
- Used ReLU for the nonlinearity functions (Found to decrease training time as ReLUs are several times faster than the conventional tanh function).
- Used data augmentation techniques that consisted of image translations, horizontal reflections, and patch extractions.
- Implemented dropout layers in order to combat the problem of overfitting to the training data.
- Trained the model using batch stochastic gradient descent, with specific values for momentum and weight decay.
- Trained on two GTX 580 GPUs for five to six days.

After AlexNet won ILSVRC in 2012, other architectures came up in the succeeding years which substantially dropped the error rate. ZF Net [35] came out in 2013 with a structure similar to AlexNet with more of fine tuning that improved the performance. ZF Net also introduced a way to visualize the learning in CNNs using deconvolutional layers. This summit of progress on ILSVRC [24] was lead by VGG Net [25] in 2014 and Google Net [28] in 2015.

1.2.1.2 ResNet — 2015

ResNet startled the deep learning community with depth not even close to any preceding network architectures. ResNet is a 152 layer network architecture that set new records in classification, detection, and localization through one incredible architecture. Aside from the new record in terms of number of layers, ResNet won ILSVRC 2015 with an incredible error rate of 3.6%. Depending on their skill and expertise, even humans generally hover around a 5-10% error rate.

And this is why ResNet is important:

- *Ultra Deep* — says Yann LeCun — **152 Layers!**
- Only the first 2 layers, the spatial size gets compressed from an input volume of 224x224 to a 56x56 volume.
- A naÃŕve increase of layers in plain nets result in higher training and test error. (Figure 1 in [7])

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 1.4: Architecture of ResNet [7]

- Microsoft Research Asia also tried a 1202-layer network, but got a lower test accuracy, presumably due to overfitting.
- **3.6%** error rate
- Trained on an 8 GPU machine for two to three weeks.

1.2.2 Object Detection Models

Object Detection is something that the Computer Vision community has been caring about for a very long time. Tremendous progress is being achieved on this problem by the region-based family [2, 23] and the single-pass family i.e. YOLO [22, 21] and SSD [18]. Even though the region based family (see Section 1.2) provides high detection accuracy, they prominently rely on 'Selective Search'[30] for region proposals which hampers the detection speed. Even the fastest, highest accuracy region based detection algorithm, Faster R-CNN [23] can achieve only 7 FPS, which is not a viable solution to problem scenarios that require real-time object detection. On the other hand, single-pass detectors like YOLO and SSD do not rely on bounding box proposals and still give significantly better results in terms of both speed and accuracy. Recently, Redmon and Farhadi released YOLOv3 [21], which claims to be 3x faster than SSD with the same accuracy. Moreover, YOLO has a smaller derived network called Tiny YOLO which is capable of operating on a CPU. This work builds upon the YOLO family of networks for performing object detection in real-time. Depending on the use case, we custom train the object detectors on the classes of objects relevant to the needs of the deployment. For example, for monitoring safety in construction sites, we might care about objects such as humans, helmets, safety shoes etc. In these cases, we do not need to consider other objects such as cars, buses or

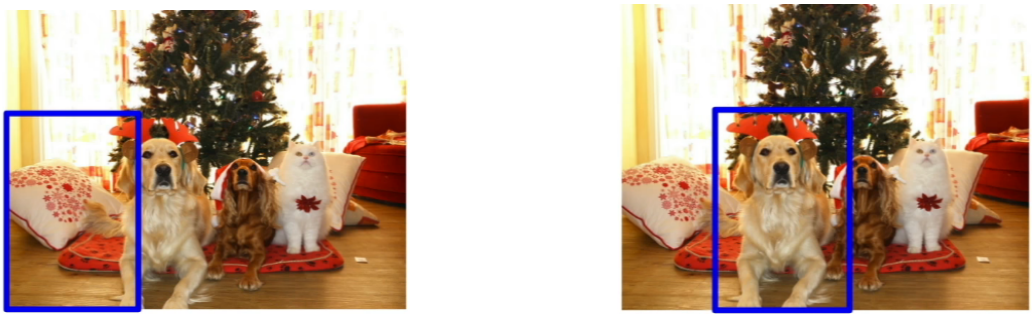

Figure 1.5: Sliding window approach

bags etc. On the other hand, if it is a traffic scenario, the focus will be on vehicle classes, such as cars, trucks, buses, motorbikes etc.

1.2.2.1 Region Based Family

One of the early and naive solutions to object detection problem was the **sliding window approach**. The intuition is to take crops from the input image and feed it to the CNN to get a classification decision on the input crop.

For example, in 1.5 we take an input crop (denoted by blue bounding box in the left hand corner) and pass it to the CNN to get a classification based on the classes that we care about. ("Cat", "Dog" and an additional category called "Background" for cases it does not see any of the required classes.) In the first crop the network should predict the background class. The second crop accurately surrounds the dog and hence the network should predict it as the dog class.

However, the major drawback in the sliding window approach in selecting these crops. These crops could be of numerous possibilities with combinations of various locations, size and aspect ratios in the image. Hence, this is a brute-force approach wherein there can be "n" possible crops that would be needed to pass through the network which is computationally intractable. To solve this issue, there was a need to select only a few crops from the image which could possibly be objects that we care about. This exact solution was proposed by region proposals using selective search [30] which implements traditional computer vision and image processing techniques to find "blobby" image regions. Region proposal networks give roughly 2000 probable boxes where objects might be present for a given input image. Since the number of crops are now limited, processing them became much more computationally tractable.

All of this came together in 2015 in a paper proposing a network called **R-CNN** (Regions with CNN features) [3]. The advent of R-CNNs has been more impactful that any of the previous papers on new network architectures. With the first R-CNN paper being cited over 5000 times, Girshick et al. at UC Berkeley

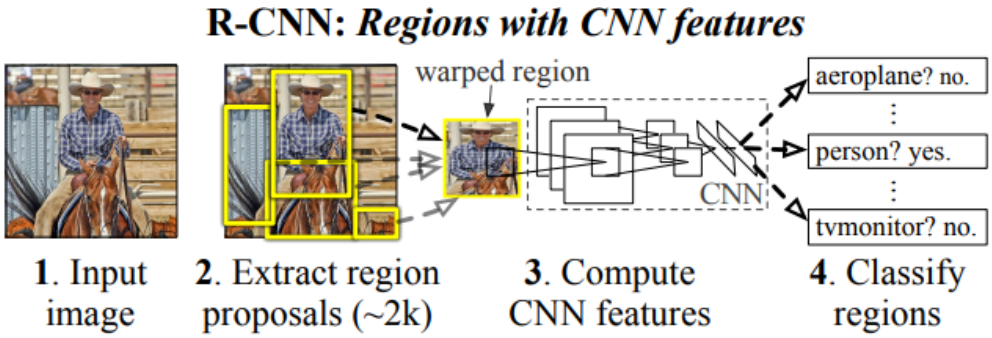

Figure 1.6: Object Detection using R-CNN [3]

created one of the most impactful advancements in computer vision

Selective Search [30] is used in particular for RCNN. Selective Search performs the function of generating 2000 different Regions of Interest (RoI) that have the highest probability of containing an object. Since we get a set of region proposal that may be of different sizes, these proposals are then âĂIJwarpedâĂİ into a unique image size that can be fed into a trained CNN that extracts a feature vector for each region. This vector is then used as the input to a set of linear SVMs that are trained for each class and output a classification. The vector also gets fed into a bounding box regressor to obtain the most accurate coordinates. Non-maxima suppression is then used to suppress bounding boxes that have a significant overlap with each other.

Despite of being quite faster than the traditional sliding window approach, R-CNN has the following drawbacks:

- Still computationally expensive, 2000 proposals for each image
- Relies extensively on the region proposal network
- Training is slow (84h), takes a lot of disk space
- Inference is slow — 47s/image with VGG16 [26]
- Fixed by SPP-net [9]

Fast R-CNN [2] was able to solve the problem of speed by basically sharing computation of the convolution layers between different proposals and swapping the order of generating region proposals and running the CNN. In this model, the image is first fed through a ConvNet, features of the region proposals are obtained from the last feature map of the ConvNet (check section 2.1 [2]), and lastly the model has fully connected layers as well as regression and classification heads.

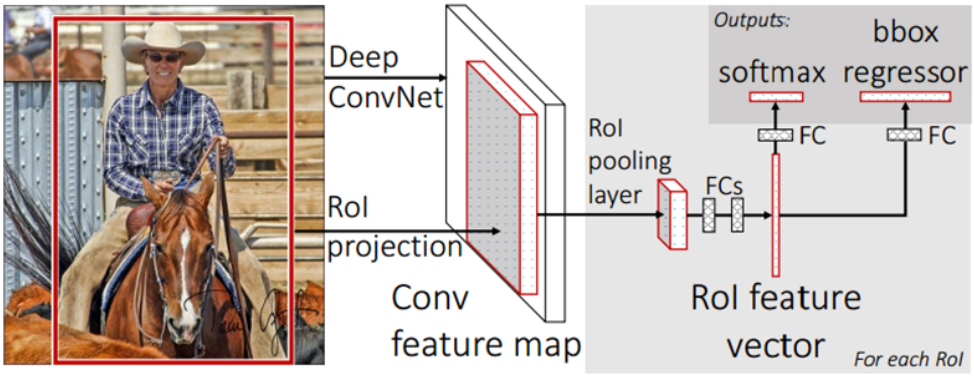

Figure 1.7: Work flow of Fast R-CNN

Figure 1.7 illustrates the pipeline that Fast R-CNN follows. Rather than taking crops directly from the image, Fast R-CNN projects these proposal on the feature map which allows us to reuse a lot of the convolutional computation. These RoIs are then warped using a pooling layer and passed to the fully connected layers. Finally, we can predict the classification scores and the offsets of the boxes.

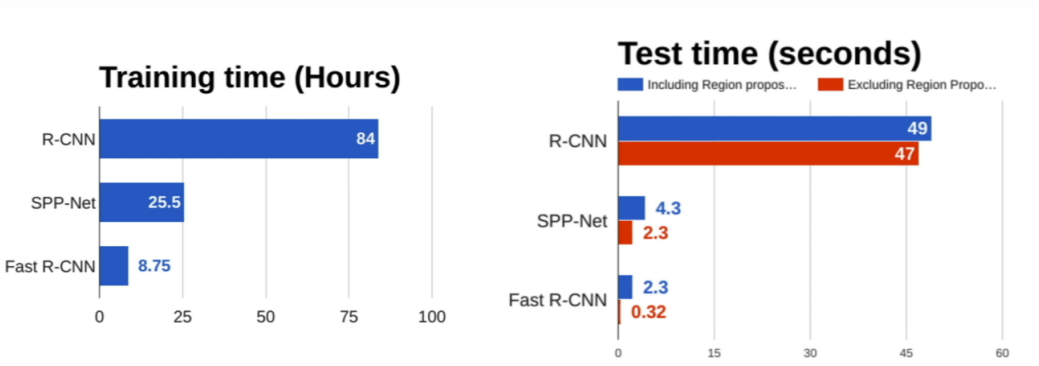

Figure 1.8: R-CNN vs SPP vs Fast R-CNN

Fast R-CNN proved to be a significant improvement to R-CNN. It was so fast at test time that its computational time is dominated by computing region proposals (See Figure 1.8 — Computing region proposals takes around two seconds). Hence, Fast R-CNN was bottlenecked because of computing region proposals.

Forunately, **Faster R-CNN** [23] solved this problem by inserting region proposals network (RPN) to predict proposals directly from features (See Figure 1.9).

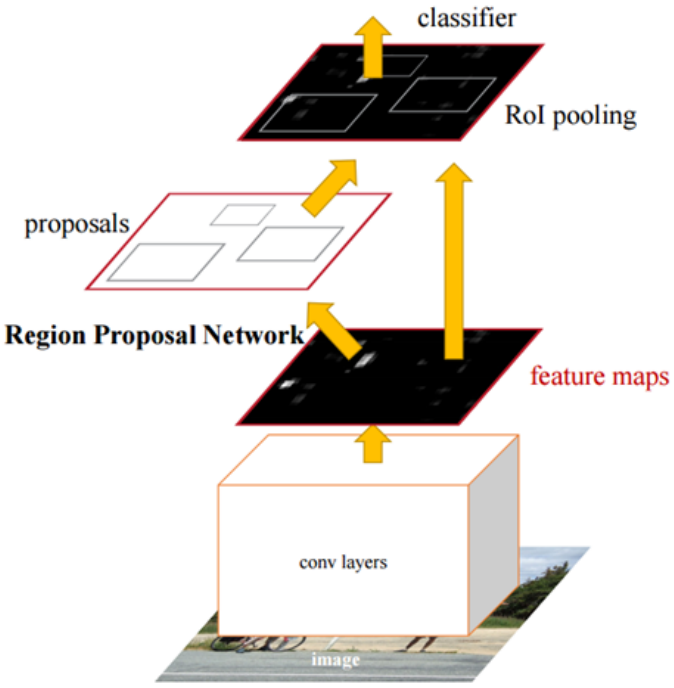

Figure 1.9: Work flow of Faster R-CNN

Faster R-CNN jointly trains with 4 losses:

- RPN classify object | not object
- RPN regress box coordinates
- Final Classification score (object classes)
- Final box coordinates

With all the above advancements in the region based family, Faster R-CNN is the fastest network in the family that has an inference time of approximately 0.2 seconds! (See Figure 1.10)

1.2.2.2 Single Pass Models

Apart from the region based family, there is another family of networks for object detection which is prominently feed forward in a single pass (No back propagation!). **YOLO** (You Only Look Once) [22] and **SSD** (Single Shot Detector) [18] are such single pass models which came around in 2016.

The core idea of single pass models is that rather than doing independent processing on these potential region proposals, instead treat it as a regression

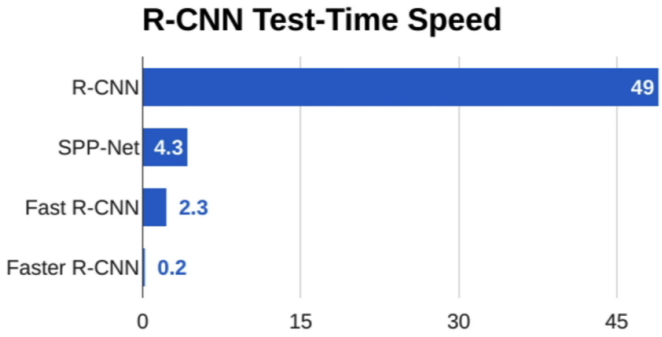

Figure 1.10: Time comparison of the region based networks

problem and make all the predicts in a single pass with a big CNN. (See Figure 1.12)

Figure 1.11 illustrates the working of the YOLO model. Given an input image, the model divides the input image into a $S \times S$ grid (7×7 in Figure 1.11). If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. If no object exists in that cell, the confidence scores should be zero. Otherwise the confidence score is to equal the intersection over union (IOU) between the predicted box and the ground truth.

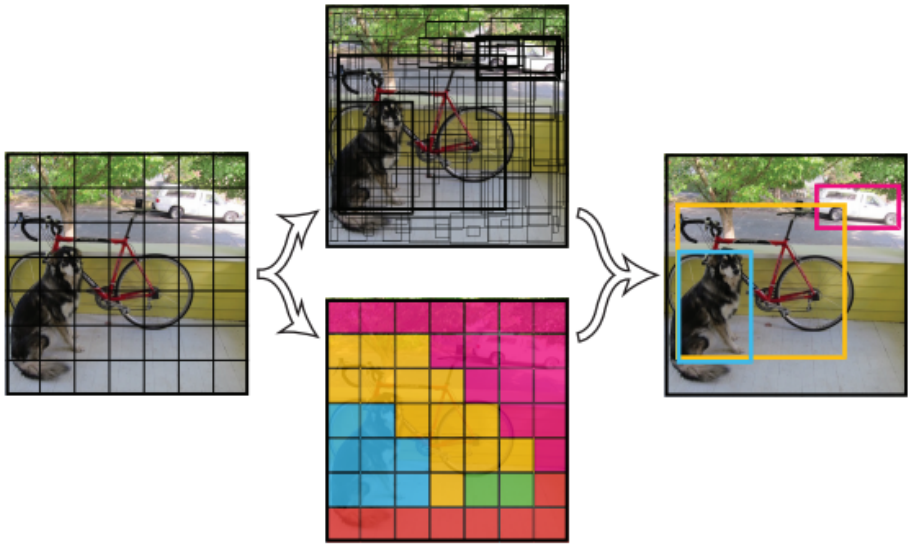

Figure 1.11: The YOLO Model [22]

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box.

These predictions are encoded as an $S \times S \times (B \times 5 + C)$ tensor. For evaluating YOLO on PASCAL VOC, we use $S = 7, B = 2$. PASCAL VOC [1] has 20 labelled classes so $C = 20$. Our final prediction is a $7 \times 7 \times 30$ tensor.

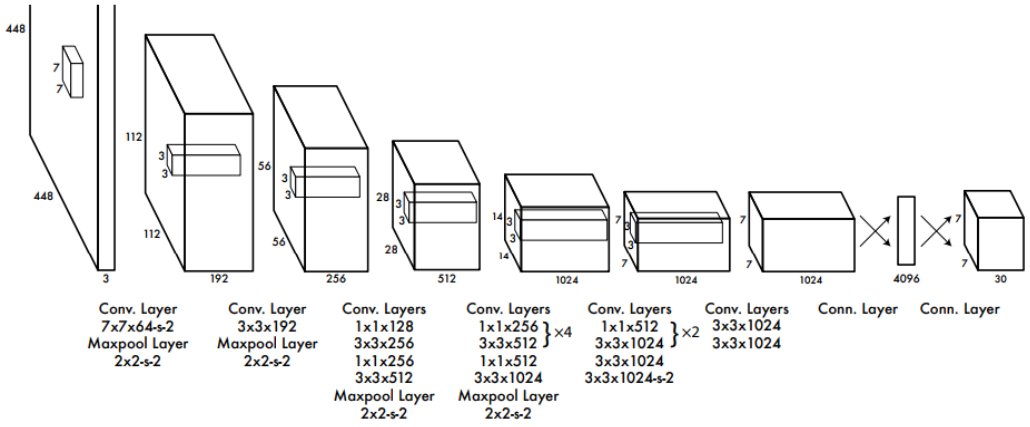

Figure 1.12: The YOLO Architecture [22]

YOLO was inspired by the GoogLeNet model for image classification [28]. YOLO has 24 convolutional layers followed by 2 fully connected layers. However, instead of the inception modules used by GoogLeNet, YOLO uses 1×1 reduction layers followed by 3×3 convolutional layers. There is also a faster version of YOLO called Tiny YOLO which has 9 convolutional layers but it comes with a compromise in accuracy (See Table 2.1).

1.3 Challenges of Video Analytics in Security and Surveillance

A lot of research has gone into video analytics systems for security and surveillance applications. Over the past two decades, video analytics companies have been providing solutions and products for video analytics in several domains. Gong, Loy, and Xiang [5] and Gouaillier and Fleurant [6] provide a good summary of the technology, problems, as well as the companies which are building analytics products in this space. Most surveillance cameras have a fixed angle of view, and for this reason, video analytics on these surveillance cameras are slightly easier than other forms of video analytics on moving cameras. Many video analytics problems can therefore be solved by background subtraction algorithms, which essentially use motion information to generate contours and motion blobs. Sobral and Vacavant [27] provide a very comprehensive survey of background subtraction algorithms for motion analytics. These algorithms work well in low traffic situations, and where one wants high sensitive alerts for problems like intrusion detection, motion detection and asset tampering. However, background subtraction algorithms are mostly unsupervised algorithms and are not trained to specifically detect humans or other objects of interest. As a result, they cannot distinguish between motion caused by shadows or leaf movements, viz-a-viz a human or animal intrusion, and often generate a lot of false alarms. However, background subtraction algorithms are extremely fast and scale very well in embedded applications. Due to privacy and bandwidth issues, it is often not feasible and prohibitively costly to deploy video analytics solutions on the cloud. As a result, it is essential to develop resource constrained video analytics solutions which can be deployed on premise. Deep learning has dominated the landscape of computer vision for the past few years, and almost all video analytics applications can be solved with high accuracies via deep learning. However, deep learning algorithms are resource hungry and require expensive GPU cloud servers to deploy. Given this, developing resource constrained, locally deployable and embedded deep learning solutions is critical. Several very recent advances like the MobileNet family of models [10], X-NOR networks [19] and Tiny-YOLO [21] have enabled deployment of resource constrained on embedded devices.

Chapter 2

Research, Analysis, Findings and Implementation

2.1 Impact of Network Customization

In this paper, we use YOLO for illustrating the importance of custom training such networks, by providing a head-to-head comparison between off-the-shelf models trained on generic datasets and models trained on custom datasets. Also, we show that there is not a huge gap in performance between custom trained YOLO and Tiny YOLO models from a deployment perspective by providing a similar head-to-head comparison.

These state-of-the-art networks are extensively used to solve object detection problems in various scenarios like counting students in a classroom, detecting vehicles running on the highway, etc. These problems get more complicated when subcategories like boy/girl student for the first case and vehicle make/model/color for the second case are needed to be identified. Generally these models are trained on ImageNet [24], PASCAL VOC [1], Microsoft COCO [17] or any such standard dataset, these pre-trained models give good results when it comes to detecting objects in generic scenarios. Huang et al. [11] provide a detailed comparison of these networks trained on the COCO dataset. However, they might not work well in all the real world scenarios due to the fact that the dataset used for training these models are very generic.

Below we describe four datasets that have been used throughout our experiments.

1. Classroom Dataset (736 images): This dataset consists of classroom images with varied seating arrangements, surroundings, class strength, etc. The main customer requirements for this deployment were getting accurate student counts, detecting whether class has started or not, and uniform compliance.

2. Community Center Dataset (5336 images): This dataset consists of indoor and outdoor images of a community place encompassing dense and sparse crowds of different age groups thereby providing an aggregated data for detecting the person class. The customer requirement was to get various statistics including counts, age/gender distribution, heat-map/flow-map etc.

3. Traffic Dataset (999 images): This dataset showcases running roads and highways consisting of various vehicles (car, bus, truck, bicycle, motorbike, three-wheelers), with different perspectives and densities.

4. Multi-Scenario Surveillance Dataset (8191 images): The Multi-Scenario Surveillance dataset is a blend of data from several customers, including the ones above. This dataset consists of the the most common classes seen in surveillance videos including persons, cars, buses, trucks, motorbikes etc. This dataset is similar to the PASCAL VOC dataset, except that it is focused on data from surveillance videos, instead of images downloaded from the Internet.

Table 2.1 illustrates the performance of YOLO and Tiny YOLO models trained on the PASCAL VOC dataset, Multi-Scenario Surveillance dataset, and the above mentioned custom datasets in terms of their Mean Average Precision (mAP) at an Intersection over Union threshold (IoU) of 0.5, along with their inference times in milliseconds.

The following are the main takeaways from the results:

1. In all cases, we see that the customized models perform better on held-out test sets compared to the off-the-shelf PASCAL VOC and Multi-Scenario Surveillance models, even though they are trained only with a fraction of the data.

2. As expected, the Multi-Scenario Surveillance dataset performs much better compared to the PASCAL VOC dataset. This is expected, since the Multi-Scenario Surveillance dataset consists of surveillance images, rather than images downloaded from the Internet.

3. Even though a custom trained YOLO model performs the best, the CPU latency is too high to run it in real-time. On the other hand, a custom trained Tiny YOLO model performs well from an accuracy perspective and yet takes about one fifth the time for inference compared to YOLO (on CPU).

Table 2.1: Comparison between models trained on PASCAL VOC, Multi-Scenario Surveillance (MSS) and Customized Datasets

Target Dataset	Train	YOLO			Tiny YOLO		
		mAP	Inference Time (in ms)		mAP	Inference Time (in ms)	
			GPU	CPU		GPU	CPU
Classroom	VOC	9.04	53	371	4.6	11	60
	MSS	58.79	43	310	36.7	14	98
	Custom	**66.16**	43	314	**54.3**	14	96
Community Center	VOC	40.9	34	276	26.8	10	80
	MSS	75.2	26	343	59.8	11	60
	Custom	**80.72**	25	343	**70.36**	12	61
Traffic	VOC	18.29	14	271	16.61	5	45
	MSS	59.1	12	110	40.10	5	30
	Custom	**71.45**	10	106	**72.54**	5	31
Multi-Scenario	VOC	9.10	55	410	4.28	16	84
	Custom	**47.22**	51	334	**32.47**	6	27

2.2 Classroom Compliance Monitoring

Here we consider a classroom scenarios to solve the following three compliances:

1. Counting the number of students precisely
2. Classifying classroom scenarios into class started | not started at a frame level
3. Detecting if each student is wearing uniform

For each compliance, we discuss all the approaches we tried and elaborate results, justifications and learnings. Towards the end, we summarize techniques that we tried but did not work and those that worked. Finally, we conclude each compliance by stating winning technique with reasonings and results.

2.2.1 Student Count using Customized Object Detection

Counting the number of students in a class may seem to be very analogous to counting persons in a general scenario. Hence, our first approach was try various network using weights directly out of the box. Moreover, since this feature was

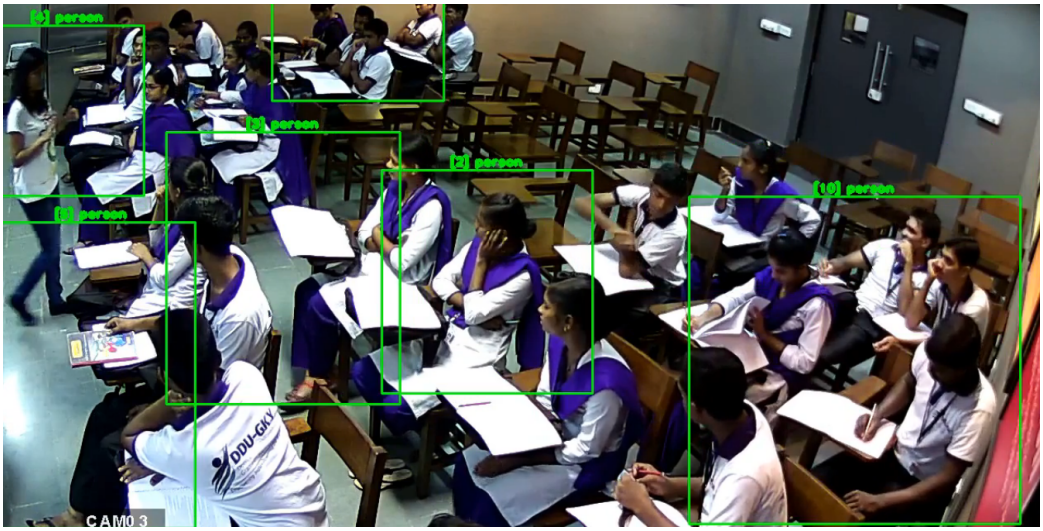

Figure 2.1: Person detection using MobileNet SSD

supposed to be deployed in a classroom where resources are constrained, we first experimented with models that operated on embedded devices.

Firstly, we considered using a MobileNet SSD [10] implementation and analyse its detection on multiple classroom scenarios. Figure 2.1 is an output frame obtained using MobileNet SSD. Clearly, the model gives terrible detections. There is a huge undercount, few false positives and inaccurate bounding boxes. Moving on, we tried using other models like HAAR Cascades and off-the-shelf YOLO and Tiny-YOLO models. HAAR [31] did not perform well in most of the scenarios as it is primarily used to detect faces. Unfortunately, most classrooms had cameras situated at the rear end and the facial features were not clear in any case.

Secondly, since YOLO trained on PASCAL VOC claimed to generalize well for most of the generic scenarios [22], we started with using YOLO for detecting human heads. (See Figure 2.2)

Even after tons of hyper parameter tuning, there was gargantuan over count due to a lot of false positives. This totally wasn't the solution. Next attempt was using YOLO for detecting person using off the shelf models trained on PASCAL VOC [1] and Microsoft COCO [17]. On tuning the confidence thresholds we could obtain a few accurate bounding boxes but still a severe undercount (See figure 2.3). Clearly, we can comprehended that using these off-the-shelf models trained on general data is not the correct solution. Hence it was necessary to customize these models to work specifically for the classroom scenarios.

Training Deep CNNs being a supervised task, it is mandatory to provide ground truth labels for training data. Generally, images annotated with bounding boxes are used for custom training deep models like YOLO for object detection.

Figure 2.2: Head detection using YOLO

Selecting images for training data is a very crucial task. The following factors are influential for creating a vital training and validation dataset.

- Number of images in training data
- Diversity of training images
- Resolution of each image
- Augmentation done at a batch level (Indirectly relates to the batch size set in the configuration file)

Since, annotating these images involves manually drawing bounding boxes on each object in a frame, the time and man power required is tremendous. Hence, we were looking for smart and user friendly video annotation (because all the data was in the form of videos) tools to reduce the labor costs and time.

A very good tool that satisfies all of the above criteria is VATIC — Video Annotation Tool from Irvine, California [32]. VATIC makes it easy to build massive, affordable video data sets and is used by labs around the world. It samples a video into frames automatically and provides a user friendly GUI to draw and drag bounding boxes for multiple class objects across the video. However, VATIC samples the video at a very high FPS (frames per second) by default, thereby hindering diversity in the training data. We know that object detection models are large and require significant computational power to operate. Hence, it was not tractable to use a replicated dataset for training. Diversity in the dataset

could be ensured by finding distinct frames across the video. This magic can be done by summarizing the video using submodular functions [12] [33]. The summarized video can then be used as a diverse set of frames. Further, we annotated these frames individually using BBox (Bounding Box) Label Tool.

Figure 2.3: Comparison between Custom Trained YOLO, Custom Trained Tiny-YOLO, off-the-shelf YOLO (trained on PASCAL VOC) models and Motion Analytics using MLBGS for object detection

Finally, we trained YOLO and Tiny-YOLO models using BBox annotated frames. Training YOLO involves various steps but one of the most important ones is to modify the configuration (cfg) file which contains 24 Conv layers for YOLO (See Figure 1.12) and 9 Conv layers for Tiny-YOLO. It is mandatory to modify the number of classes (1 in this case — person class) in the region layer and according modify filters in the final convolutional layer — $(classes + 5) \times 5$. The cfg file is also responsible for setting the training batch size and number of subdivisions which can be changed for resource constrained environments. Moreover, YOLO internally augments the data at a batch levels using parameters like angle, hue, saturation, exposure, etc.

Figure 2.3 shows a clear comparison between different object detection methodologies and helps us draw the following conclusions:

- Custom YOLO model — 24 Conv layers
  - High Accuracy (See Table 2.1 for mAP)
  - Tighter bounding boxes and smooth tracking
  - Requires GPU
- Custom Tiny-YOLO model — 9 Conv layers
  - Decent Accuracy (See Table 2.1 for mAP)
  - Acceptable bounding boxes and average tracking
  - Can operate on embedded devices
- Off-the-sheld YOLO model — Trained on PASCAL VOC
  - Low accuracy (See Table 2.1 for mAP)
  - Negligible detections (Severe undercount)
  - Does not work well because of generic data used for training
- Motion analytics using MLBGS
  - Does not detect any object if motion is not present.

2.2.2 Class Started — Class notStarted Classification using Feature Engineering

The objective of this compliance was to detect if the classroom has begun or not. The data consisted of 4 different classrooms in a single SDC (Skill Development Center). This was a complicated problem to solve since there could be "n" number of scenarios for both the situations (Class Started — notStarted). For example, a class may have different number of students everyday, wearing different apparels everyday and various types of seating arrangements. Moreover, there might be multiple cameras at different locations in the classroom resulting in various combinations of the same instance. Pragmatically, it was impossible to create a set of training data that encompasses all the scenarios.

Due to endless possibilities of class started — class not started scenarios, it was reasonable to train a deep CNN on a set of diverse classroom contexts. A typical classroom scene has identical patterns which a relatable to not all but a few real-world scenarios. Therefore, it was quite intuitive to use a deep CNN trained on a generic dataset like ImageNet, PASCAL VOC or Microsoft COCO and use the knowledge gained from this network as a foundation. This concept is widely known as Transfer learning or Inductive transfer. The machine learning community defines transfer learning as follows:

- Transfer learning and domain adaptation refer to the situation where what has been learned in one setting — is exploited to improve generalization in another setting [15].
- Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned [29].

2.2.2.1 Transfer Learning

Briefly, we choose a pre-trained CNN model, ideally trained for a contextually similar problem. We then choose a layer of the model, which is used to extract the features of an image when it is forward passed through the network. This extraction of feature vectors is performed for all images in the training set. Upon the completion of the feature extraction, we train a multinomial logistic regression model. On the positive side, Transfer Learning allows us to quickly train models without a GPU. It generally achieves a high accuracy on a held-out test set, when trained on a small training set. However, Transfer Learning requires us to possess domain specific knowledge, and intricacies of the base CNN model to be used, in order to identify the feature extraction layer. Moreover, the identification of the layer involved quite some experimentation and heuristics.

Table 2.2 illustrates transfer learning accuracies for various networks. Only the final and pre-final fully connected layers are considered for respective net-

Table 2.2: Transfer learning results for Equalize dataset (1992 train and 1199 test images) and Randomize dataset (2591 train and 600 test images)

Network	Layer	Equalize Test Accuracy	Randomize Test Accuracy
Alexnet	fc6	77.0642	**89**
	fc7	76.7306	88.5
ResNet	res5b branch2c	76.4804	87
	res5c branch2c	76.7306	87.6667
VGG 16	fc6	76.7306	86
	fc7	77.5646	84.6667
VGG 19	fc6	76.3136	84.8333
	fc7	76.6472	84.5
Mobile Net	pool6	77.0642	87.6667
Googlenet	loss3 classifier	73.5613	72.8333

works because CNN features are more generic in early layers and more original-dataset-specific in later layers [13]. We primarily used two different splits (equalize and randomize) of the same dataset for experimentation. Alexnet [14] worked well in most of the scenarios in both the splits. Since, the fully connected layer 6 gave better results than other layers and models we considered to use it for inference on other unseen data. Gradually after a lot of analysis on unseen data, we found that the model overfit to the training data and was unable to generalise for new classroom scenarios. Since there could be "n" possible classroom scenarios, it was not feasible to provide all of them in the training data.

2.2.2.2 Feature Engineering

Relying completely on a Convolutional neural network could solve only peculiar classroom scenarios provided in the training data. At this point of time, it was necessary to hand craft features that significantly contribute as a differentiating factor for humans while categorizing a class into started and not started. The following factors were decided at a frame level for every unique classroom:

- Spatial arrangement of students
- Intensity of motion
- Attendance

The **spatial arrangement** of students is a very important factor in deciding whether the class has started or not. The idea is to use a feature extractor to fetch the seating arrangement of students and train a classifier using this feature vector for prediction. In this case, we use YOLO [20] as a feature extractor and logistic regression as a classifier. We create a 10×10 grid analogous to YOLO's grid on the frame (See Figure 2.4). We then use a customized YOLO model to detect persons from a particular frame (See custom classroom in Table 2.1 for accuracies). The next step is to compute an intersection over union (IOU) of each bounding box enclosing the "person" class and grid cell. In case there are multiple bounding boxes intersecting with the same grid cell, we add up the IOU values for that particular cell. A similar process is repeated for each grid cell to obtain a $10 \times 10 \Rightarrow 100$ valued sparse feature vector containing IOU values for each grid cell. The sparse feature vector intelligently stores only the important information, i.e the index and corresponding IOU value, thereby saving tons of spatial and computational resources. Further, we compute sparse feature vectors for multiple frames in the training data and use them for training the classifier.

The second important and obvious factor is **motion** in the classroom. Apparently, a class with a lot of motion is not in progress because majority of people are either walking in/out or perhaps just wandering around. On the other hand, a class that is in progress definitely has negligible motion. Various background subtraction algorithms like GMG [4] and MLBGS [34] amalgamated with image processing techniques like morphological analysis are used for analysing motion between consecutive frames in a video. We used GMG in this case as it performed better than MLBGS. GMG algorithm combines statistical background image estimation and per-pixel Bayesian segmentation. It uses first few (120 by default) frames for background modelling. It employs probabilistic foreground segmentation algorithm that identifies possible foreground objects using Bayesian inference. The estimates are adaptive; newer observations are more heavily weighted than old observations to accommodate variable illumination. Several morphological filtering operations like closing and opening are

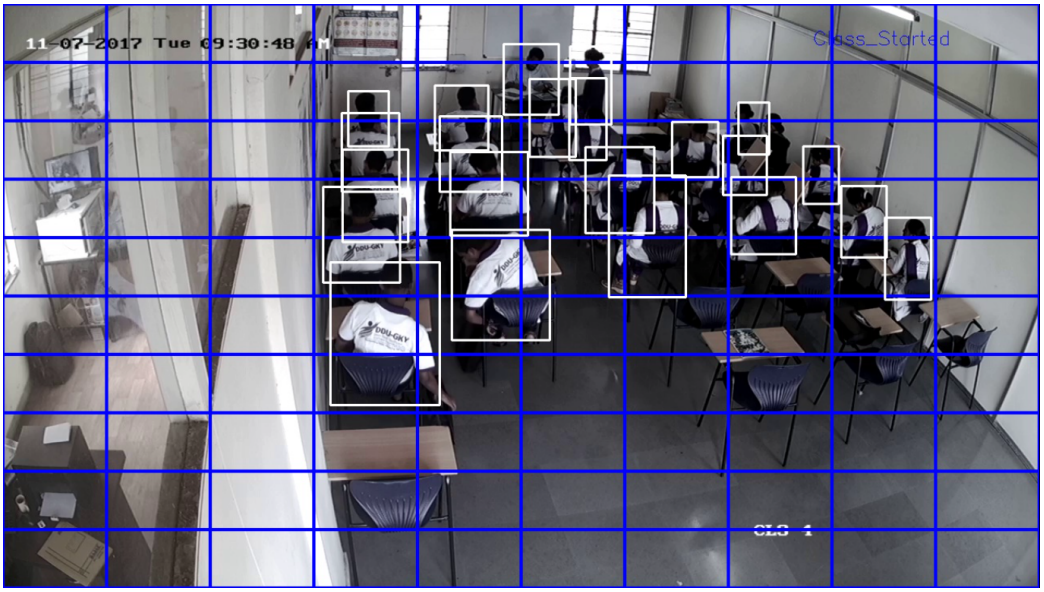

Figure 2.4: Detected boxes and 10 × 10 Intersection Over Union (IOU) Grid used to get sparse feature vector and train Logistic Regression

done to remove unwanted noise. This enables us to draw motion blobs around moving objects. An area with high motion is surrounded by a larger motion blob, thereby capturing more area in the frame. The sum of areas of all such motion blobs in the frame is used as a feature for class started — not started classification. The third feature is **person count** (classroom attendance) which is based on heuristics of classrooms when they have started and particularly operates as a threshold.

Taking into consideration, all the three features, namely, spatial arrangements (teaches the model different orientations of students in the classroom during class started and not started events), motion (considering that a started class has low motion) and person count(heuristics for thresholding minimum percentage of students for a class started scenario) we were able to solve this problem and generalize well for all scenarios concerned with a particular SDC (Skill Development Center).

2.2.3 Uniform Detection using Customized Multi Class Object Detection

The third compliance and final compliance is to analyse if the detected students are wearing uniform or not. This seemed to be a trivial and straightforward problem beforehand but turned out to be convoluted due to different challenges like uncertain lighting conditions, overlapping students in the frame, etc.

One of the early approaches that seemed to be plausible were cosine similarity and color histograms. In the cosine similarity approach, we cropped out a few students wearing uniform from the frame as reference data. Further, we used a custom trained YOLO model to detect persons during inference and create a vector representing each detected person. Cosine similarity (See Eq. 2.1)is then computed between the vector of each detected object and reference image vectors.

$$\cos\varphi = \frac{\left[\overrightarrow{CA'}, \overrightarrow{CB}\right] \cdot \left[\overrightarrow{CA'}, \overrightarrow{CD}\right]}{\left|\left[\overrightarrow{CA'}, \overrightarrow{CB}\right]\right| \cdot \left|\left[\overrightarrow{CA'}, \overrightarrow{CD}\right]\right|}. \tag{2.1}$$

However, this technique did not work due to the significant difference in each vector. Apparently, each vector cannot be directly compared because of various real-world challenges mentioned in paragraph one that lead to significant differences even during the same class. The second approach that failed due to similar reasons was the color histogram. We computed color histograms for RGB and HSV channels and then discretized the colors in the image into a number of bins, and counting the number of image pixels in each bin. For example, a redâĂŞblue histogram can be formed by first normalizing color pixel values by dividing RGB values by $R+G+B$, then quantizing the normalized R and B coordinates into N bins each. Since a color histogram is nothing but a representation of the distribution of colors in an image it would fail in cases with even average change in light leading to a completely different color histogram.

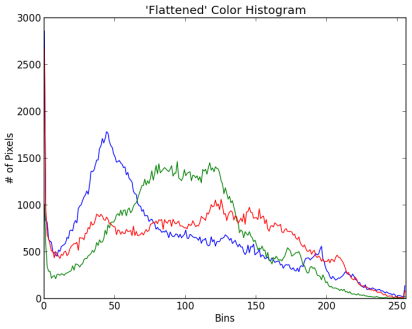

Figure 2.5: Example of a color histogram

To mitigate challenges caused by lighting changes and eliminate noise included in the bounding boxes detected by YOLO, we tried cropping out students analogous to segmentation techniques. Ideally, it should create a binary mask containing segments of students. To do so, we used MOG for background subtraction between an empty classroom frame and a frame on which we want to run inference. Since these two frames are not consecutive, there was a significant change in pixel values across the frame between the empty classroom frame and the inference frame. Hence, all the background subtraction algorithms failed.

Clearly, solving this problem using traditional image processing techniques was not working out. Therefore, we decided to use Deep Convolutional Neural Network to solve this problem. We trained a YOLO model to detect three classes namely, boyUniform, girlUniform and noUniform. Since the number of classes is now different, the number of filters in the network (See figure 1.12) has to be modified to $(3+5)\times 5 \Rightarrow 40$ filters. We trained this model using 1200 images for 60000 iterations for about 2 days.

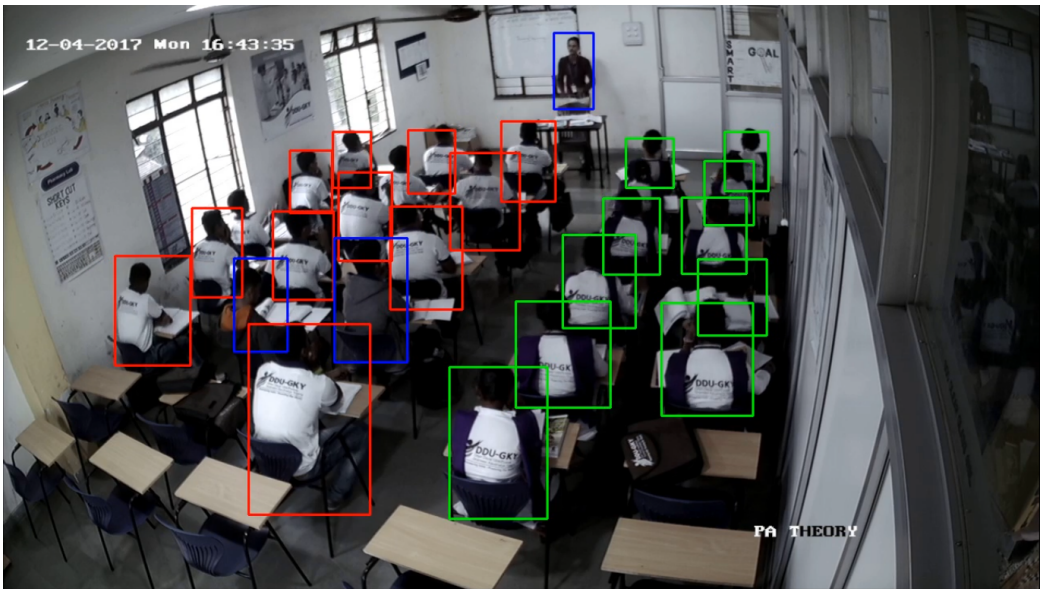

Figure 2.6: Uniform detection and classification using YOLO. (Green boxes are classified as girlUniform, red as boyUniform and blue as noUniform)

Figure 2.6 illustrates the performance of custom trained YOLO model for uniform detection. The model clearly outperforms all the approaches attempted earlier and operates at a very high accuracy.

Chapter 3

Future Work

With recent advancements in computer vision and the ones to come, there can be various techniques that can prove to give better results for the classroom compliances. A few approaches that can be taken up as experiment and can potentially give equal or better results are as follows:

- Using Mask R-CNN [8] for segmentation can plausibly lead to a more generalized model for detection "person" and "uniform". However, annotating data for a segmentation task is extremely arduous. Nevertheless, Microsoft COCO [17] has data annotated at a joint level for humans which is generally used to detect if a human is standing, sitting, etc.
- Evaluate performance on custom training SSD models instead of YOLO
- Use Transfer Learning to classify detections from YOLO/SSD into girlUniform, boyUniform, noUniform.

Chapter 4

Conclusion

This work is focused upon automating analysis of different real-world problems and addresses the challenges that intricate such problems. We work through various techniques and try multiple approaches and justify each failed or successful approach with results. We go over the data collection tricks and training pipelines for fast experimental turn around. We also share the significant amount of practical experience we have gained by deploying models at customer locations.

The major takeaways from this work are as follows:

- Deep CNNs are the state of art for Computer Vision tasks and have been growing unprecedentedly in the past few years.
- Deep Learning does not necessarily require GPU cloud servers. It is possible to get high accuracy using on-premise CPU deployments.
- Customization will always give better results than off-the-shelf models.
- Customization provides superior accuracies compared to off-the-shelf models even with fraction of the training dataset.
- Tricks such as data summarization in the customized training pipeline ensures high accuracy with a smaller training set due to diversity.

References

[1] M. Everingham et al. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338.

[2] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1440–1448.

[3] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587.

[4] Andrew B Godbehere, Akihiro Matsukawa, and Ken Goldberg. “Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation”. In: *American Control Conference (ACC), 2012*. IEEE. 2012, pp. 4305–4312.

[5] Shaogang Gong, Chen Change Loy, and Tao Xiang. “Security and surveillance”. In: *Visual Analysis of Humans*. Springer, 2011, pp. 455–472.

[6] Valérie Gouaillier and A Fleurant. “Intelligent video surveillance: Promises and challenges”. In: *Technological and commercial intelligence report, CRIM and Technôpole Defence and Security* 456 (2009), p. 468.

[7] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *arXiv preprint arXiv:1512.03385* (2015).

[8] Kaiming He et al. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2980–2988.

[9] Kaiming He et al. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In: *european conference on computer vision*. Springer. 2014, pp. 346–361.

[10] Andrew G Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861* (2017).

[11] Jonathan Huang et al. “Speed/accuracy trade-offs for modern convolutional object detectors”. In: *CoRR* abs/1611.10012 (2016). arXiv: 1611.10012.

[12] Rishabh Iyer. “c Copyright 2015”. In: ().

[13] Andrej Karpathy. “Cs231n convolutional neural networks for visual recognition”. In: *Neural networks* 1 (2016).

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. 2012, pp. 1097–1105.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), p. 436.

[16] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[17] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *CoRR* abs/1405.0312 (2014). URL: http://arxiv.org/abs/1405.0312.

[18] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: 2016.

[19] Mohammad Rastegari et al. “Xnor-net: Imagenet classification using binary convolutional neural networks”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 525–542.

[20] J. Redmon and A. Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

[21] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: *arXiv* (2018).

[22] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788.

[23] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[24] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[25] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2014).

[26] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[27] Andrews Sobral and Antoine Vacavant. “A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos”. In: *Computer Vision and Image Understanding* 122 (2014), pp. 4–21.

[28] Christian Szegedy et al. “Going Deeper with Convolutions”. In: *Computer Vision and Pattern Recognition (CVPR)*. 2015.

[29] Lisa Torrey and Jude Shavlik. “Transfer learning”. In: *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques* 1 (2009), p. 242.

[30] Jasper RR Uijlings et al. “Selective search for object recognition”. In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[31] P. Viola and M. Jones. “Rapid object detection using a boosted cascade of simple features”. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, pp. 511–518. DOI: 10.1109/CVPR.2001.990517.

[32] Carl Vondrick, Donald Patterson, and Deva Ramanan. “Efficiently scaling up crowdsourced video annotation”. In: *International Journal of Computer Vision* 101.1 (2013), pp. 184–204.

[33] Kai Wei, Rishabh Iyer, and Jeff Bilmes. “Submodularity in data subset selection and active learning”. In: *International Conference on Machine Learning*. 2015, pp. 1954–1963.

[34] Jian Yao and Jean-Marc Odobez. “Multi-layer background subtraction based on color and texture”. In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[35] Matthew D. Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *CoRR* abs/1311.2901 (2013). arXiv: 1311.2901. URL: http://arxiv.org/abs/1311.2901.