Analysis of stock market recommendations using computer vision

Submitted in partial fulfilment of the requirements
of the degree of

Masters of Technology

by

Pronoy Mandal (Registration no. 20 - 27 - 09)

Supervisor:
Dr. (Mr.) Dasari Srikanth

Co - supervisor:
Mr. Himanshu Chaudhary

Department of Applied Mathematics
Defence Institute of Advanced Technology (Pune)
2021 - 22

Approval sheet

Thesis titled: Analysis of stock market recommendations using computer vision

Undertaken By: Pronoy Mandal (Registration no. 20 - 27 - 09) is approved for the award of the degree of Masters of Technology in Data Science programme.

(Internal supervisor)

(External supervisor if any)

(Member from allied department)

(External subject expert)

(Chairman)

Date:
Place:

CERTIFICATE

This is to certify that the dissertation entitled **"Analysis of stock market recommendations using computer vision"** submitted by **Pronoy Mandal** (Registration no. 20 - 27 - 09) is in partial fulfilment of the requirement for the award of degree of **Masters of Technology in Data Science**.

Dr. (Mr.) Somanchi VSSNVG Krishnamurthy
Internal supervisor
Associate professor
Department of Applied Mathematics
Defence Institute of Advanced Technology (DIAT)
Girinagar, Pune - 411025
Date:
Place:

Name:
External supervisor
Date:
Place:

Countersigned:
Dr. (Mr.) Somanchi VSSNVG Krishnamurthy
Head of the Department of Applied Mathematics
Defence Institute of Advanced Technology (DIAT)
Girinagar, Pune

Declaration

This is to certify that the dissertation report titled **"Analysis of stock market recommendations using computer vision"** comprises of original work and due citations have been made in the text to all other material used. The dissertation has not been submitted in any form for another degree or diploma at any other institute/university.

Date:
Place:

Pronoy Mandal
Registration no. 20 - 27 - 09
Department of Applied Mathematics
Defence Institute of Advanced Technology (DIAT)
Girinagar, Pune

Dr. (Mr.) Somanchi VSSNVG Krishnamurthy
Internal supervisor
Head of the Department of Applied Mathematics
Defence Institute of Advanced Technology (DIAT)
Girinagar, Pune - 411025
Date:
Place:

Name:
External supervisor
Date:
Place:

Acknowledgements

I am thankful to my institute Defence Institute of Advanced Technology (Pune) for considering my dissertation and providing help at all stages necessary during my work collecting information on my dissertation.

It gives me immense pleasure to express my deep and sincere gratitude to Dr.(Mr.) Dasari Srikanth (Dissertation guide) for his kind help and valuable advice during the development of the dissertation synopsis and his guidance and suggestions. Apart from that, I would like to thank Mr Himanshu Chaudhary (co-guide), Assistant General Manager from the Integrated Surveillance Department (ISD) of the Securities and Exchange Board of India (SEBI) for guiding me throughout this project and giving his valuable suggestions wherein necessary. I would like to express my deepest sense of gratitude to the employees of the Integrated Surveillance Department (ISD) and Information Technology Department (ITD) of SEBI who have put in their time and efforts to give additional guidance and suggestions during the project.

I would also like to thank the employees of Hewlett Packard Enterprise (HPE), a contractor of SEBI, who have allowed me to access their development environments, sophisticated hardware and cloud platforms, and associated tools and have been an immense help throughout this project.

I am deeply indebted to the Head of the Applied Mathematics Department, Dr. (Mr.) Somanchi VSSNVG Krishnamurthy, for giving me this valuable opportunity to do this project. I express my hearty thanks to him for his assistance, without which it would have been difficult to finish this project and review the project successfully. I would like to convey my deep sense of gratitude to all the teaching and non-teaching staff for their constant encouragement, support, and selfless help throughout the dissertation work. It is my great pleasure to acknowledge the help and suggestions that I received from the Department of Applied Mathematics.

Abstract

There are a large number of business NEWS channels broadcasting a variety of topics throughout morning and evening primetime. These range from simple stock market recommendations (strategies to BUY/SELL/HOLD) to detailed fundamental and technical analyses of various companies to in-depth quarterly performance reviews of various parties in the market. With the availability of such a volume of information from public sources; it becomes difficult to assimilate all the information in a single place and analyse it for meaningful information.

This project encompasses the ability to analyse NEWS which gives stock market recommendations (a subset of business NEWS) and presents the results in a tabular format to the user for analysis. The tabular format includes basic information from the broadcast such as the telecast date of the show, the analyst presenting the same, the stock which is being discussed, the NSE listing symbol of the concerned stock, the recommendation etc. (totalling 8 parameters). The exact reason why the Securities and Exchange Board of India (SEBI) is attempting to analyse NEWS videos is something that can't be specified within the scope of this report due to its confidentiality. However, it can be said that this project's output forms the base as well as the input of many anomaly (in the context of violations in the securities market) detection models at a later stage.

This project goes a step ahead to ease and automate the job of SEBI officers from the surveillance department by deploying the same onto an MLOPs platform and creating an equivalent free-running workflow in a virtual environment.

The aforementioned reason forms the principal motivation for doing this project i.e. it helps automate a large part of a task that is prone to human errors and mistakes such as collecting data manually by watching the NEWS shows and at the same time reduces the time required to carry out such tasks. Apart from that, since an MLOPs workflow is included in the project, this project helps establish various principles which need to be followed for any ML project in general so that it can be deployed into production.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Dissertation name and title

My dissertation is titled **Analysis of stock market recommendations using computer vision**.

1.2 Problem statement

There is a large no. of daily broadcasts which take place throughout primetime (morning and evening) as well as outside of it. This project aims to digitize as well as summarise the recommendations cumulatively and present them to the end-user at the same time this project and its associated machine learning system are input to more sophisticated models for anomaly detection (in the context of violations in the securities market) at a later stage.

1.3 Proposed solution

The proposed solution is a deep learning model aimed at solving two fundamental machine learning problems at hand:

(a) Object detection – Detect and classify important regions of interest in the frames of various NEWS broadcasts (or NEWS shows) i.e. regions having (but not restricted to) the name of the analyst/presenter, the recommendation being given, the company, commodity or the market segment being discussed and its corresponding listing symbol (if any), the various metrics of the recommendation such as stop loss, target price etc.

(b) Optical Character Recognition (OCR) – Bilingual (English and Hindi) text obtained from the above regions of interest is passed onto sophisticated OCR models which have a text detection model and an actual language model(s) (corresponding to the concerned languages) working under the hood to obtain or extract the required information.

The final output consists of a CSV (or) an excel file as required which summarises the above-obtained information.

1.4 Scope and report contents

Chapter 2 goes into a detailed discussion of the literature which has been studied while preparing this report. It includes several details such as (but not restricted to) how **YoLo** models or in general object detection models are evaluated and which performance metrics and common datasets are referred to for their speed and accuracy. It has additional details regarding how advanced deep learning models like **YoLov4** can be optimised for special use cases wherein necessary. The work presented in this dissertation may be completed sequentially or parallelly and is described as follows. Additionally, the chapters to which they pertain to are also mentioned.

(a) Machine learning and computer vision: There are a total of 8 NEWS shows which are currently being targeted under this project. Machine learning used within this project is further divided into two parts: the one dealing with the recognition of various numerical parameters from the video frames of a broadcast (wherein active work is going on to increase efficiency as well as the accuracy of models) are random forest classifiers which have been trained on a large number of video frames which were earlier trained on **YoLov3** output (but needed additional training due to lesser accuracy) and templates followed by a much more robust and reliable framework i.e. YoLov4 which has been trained only for one NEWS show (i.e. *PehlaSauda* from CNBC Awaaz).

The second part of ML goes into OCR models wherein **no active work is being pursued** rather readymade models and OCR engines that are doing the job within tolerable error limits are being used. Both Tesseract – OCR and EasyOCR provided models have been used for this purpose and their performance metrics analysed. It should be noted that **no part of the report** goes into the details of the inner workings of the OCR models which are deep learning models in themselves.

As opposed to standard ML projects wherein a considerable amount of time is used for training a particular model; this project uses more time and space to carry out a variety of image processing operations on the captured video frames. This includes histogram equalization, simple grey level transformations, and complex erosion and dilation processes. After carrying out sufficient thresholding operations, these are fed into the ML models described above.

The details for the above i.e. the process of collecting the data, preparing the data for training and testing and using appropriate deep learning models for the same are discussed in chapter 3 in detail.

(b) Deployment into production: Rarely do data science projects in academia ever reach production and into a streamlined architecture. There are several reasons which prevent it from happening so. Such details, as well as reasons, have been given in brief in chapter 4's opening sections.

Fortunately, for this project, the deployment of the data models falls within its current scope. All the ML models which would be trained and tested would be loaded onto the proprietary **Ezmeral MLOPs** platform by **HPE**. The servers for the same reside alongside the required hardware resources on SEBI's internal servers. Doing this (without going into the details) would make the entire process streamlined i.e. the process of uploading videos, and training models on new data as soon as it is available and at the same time would decrease the manual work involved while doing so. As soon as the deployment has been completed, whether for all the NEWS shows or even a few of them, the internal server would be able to serve requests from within the organization domain (*POST* methods with *JSON* body) and at the same time would be able to abstract all the complex code and scripts running in the background.

Apart from the MLOPs deployment, it is more desirable for SEBI to have a continuously running workflow that shall consist of processing a bulk or a batch of videos at once. A workflow about this has been deployed on a standalone virtual machine as a part of this project. Which projects require an MLOPs workflow and which don't, as a result, have also been discussed in chapter 4.

Chapter 5 would then go on to discuss the various results which have been obtained at the end in terms of speed and accuracy of the YoLo models involved which metrics are suitable for evaluating projects spanning multiple ML models. Chapter 6 of the report would then finally present important conclusions drawn from the entire project.

Chapter 2

Literature review

2.1 YoLo – A brief discussion on v3 and v4 for real-time object detection

YoLo standing for You Only Look Once is(are) a set of advanced deep learning models for real-time object detection. The focal point for YoLo is that it manages to perform bounding box regression and classification at the same time. The architecture or methodology which separates YoLo and other object detection frameworks is that other sophisticated frameworks (RCNN and even Fast RCNN) attempt to repurpose existing classifiers and localisers to perform object detection. They apply a single model to multiple locations within an image with variations in scale and other properties of the image multiple times. High scoring regions are labelled and classified accordingly.

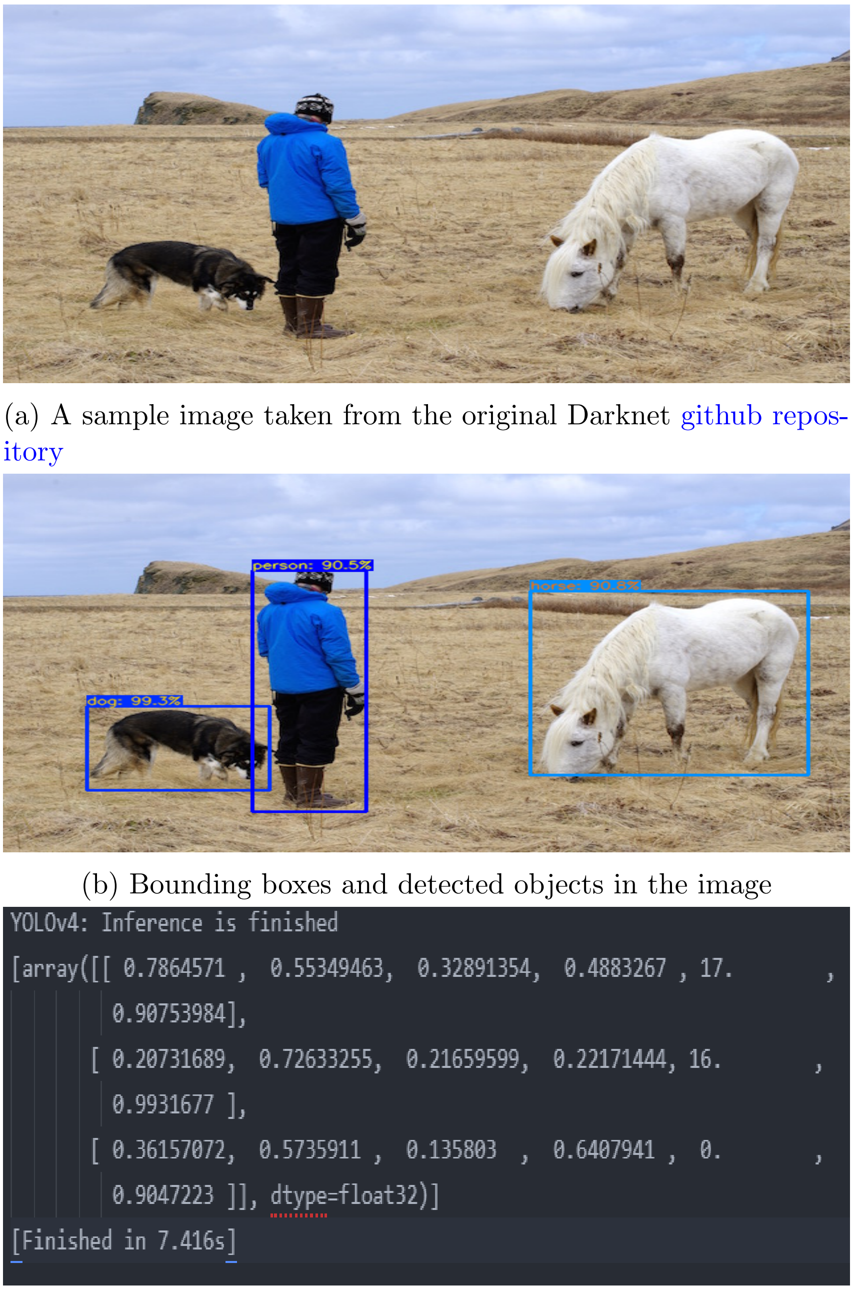
YoLo frameworks take a drastically different approach: they divide the region into a large number of regions (or boxes) and simultaneously apply a single convolutional neural network to the entire image. The bounding boxes and associated probabilities are received at the output. These regions are then weighted according to these probabilities. The reason why this method supersedes other popular methods is that other neural network-based deep learning architectures (such as R-CNN) may end up using multiple network evaluations (in the order of thousands) on a single image that takes up a considerable amount of processing time. Additionally, YoLo models look at the entirety of the image during the detection phase because of which their predictions are based on the global context of the image. Although the differences don't end here, these fundamental differences are enough to give YoLo a second to third order speed advantage over Fast R-CNN and R-CNN models.

The neural network for YoLo along with its entire compilation system is known as Darknet. A typical darknet run on an image, set of images, video or a real-time stream of image content (such as through a webcam) leads to the following three things as output:

- The detected classes of objects (e.g. horse, car etc.)
- The prediction confidence (e.g. 0.91, 0.86 etc.) and
- The time it took to carry out the prediction or detection (e.g. 6s, 12s etc.)

- Sophisticated high-level libraries will also produce the bounding boxes corresponding to each

Following is the output when a high-level library like yolov4 is used to process an image as shown below.



(a) A sample image taken from the original Darknet github repository

(b) Bounding boxes and detected objects in the image

```
YOLOv4: Inference is finished
[array([[ 0.7864571 ,  0.55349463,  0.32891354,  0.4883267 , 17.        ,
          0.90753984],
        [ 0.20731689,  0.72633255,  0.21659599,  0.22171444, 16.        ,
          0.9931677 ],
        [ 0.36157072,  0.5735911 ,  0.135803  ,  0.6407941 ,  0.        ,
          0.9047223 ]], dtype=float32)]
[Finished in 7.416s]
```

(c) The bounding box coordinates obtained

Figure 2.1: A sample Darknet run on an image

2.1.1 YoLov3

Joseph Redmond is credited with designing the original neural network architecture called Darknet. This was made using all low-level languages so that it could be made flexible as it can be. This architecture produced a series of computer vision wonders in the field of object detection named YoLo: the original one, v2, v3 and then v4. Even YoLov5 has been developed (the first one to be developed by an organisation as opposed to an individual) at the time of writing this report. However, we won't be discussing it here since it hasn't been used in the concerned project.

YoLov3 is the first version using an objectivity score for the prediction of bounding boxes. It also proceeds to add further connections to the backbone of the entire architecture. Additionally, detections are also carried out at three levels of granularity which greatly increased the inferencing accuracy on smaller objects as well as objects/ smaller objects placed close to each other. There are some important parts or sections in YoLov3 which proves its robustness to a variety of real-life object detection scenarios. They relate to better bounding box prediction, class prediction, their feature extractor as well better prediction across multiple scales provided in the input. We would be discussing each of them in brief in the following paragraphs.

Bounding box prediction

Any bounding box in a prediction is represented by a set of four parameters namely t_x and t_y: coordinates of the centre of the bounding box and its width and height represented by t_w and t_h respectively. In the event of the top left corner of the bounding box being displaced by (c_x, c_y) respectively, the following overall changes are calculated in the dimensions (let p_w and p_h be the width and height of the bounding box from the previous iteration).

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

Unlike previous systems (or previous YoLo versions), the training phase assigns an objectness score to each bounding box. This score equals one if the current bounding box overlaps a ground truth object to an extent greater than any other previously obtained bounding box. If any previously obtained bounding box doesn't overlap the ground truth to a greater extent or overlaps only by a certain threshold (say 0.5 as stated in this paper) then the objectness score is simply zero. Following is a representation of all the important dimensions which are involved in this process.

Class prediction

SoftMax is one of the most common types of methods used for multi-label classification, however, the method has been leading to decrementing results for the class prediction accuracy in YoLov3 as well as has been unnecessary. Therefore, simple multiple logistic classifiers are being used, and **binary cross-entropy** has been used as a loss function. This different approach helped in the case of much larger and more sophisticated datasets such as the Open Images Dataset wherein labels may overlap frequently. SoftMax algorithm assumes that a single bounding box corresponds to exactly one label which is often not the case.

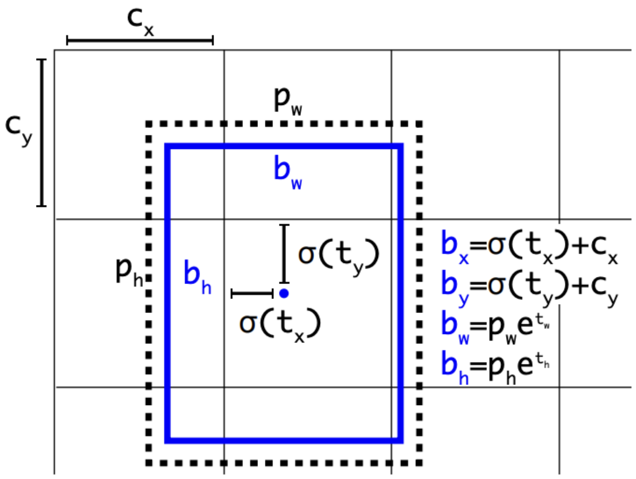


Figure 2.2: An illustration showing all the important bounding box dimensions being used in YoLov3

Feature extraction

Feature extraction in YoLov3 has been a hybridisation of the one in YoLov2, Darknet – 19 (19 convolutional layers) and another residual network. But the most important part of this network is shortcut connections which increase the size of the network significantly, however, continue to supersede ResNet variants in terms of efficiency. Since this leads to a total of **53** convolutional layers it's named Darknet-53.

2.1.2 YoLov4 – improvements over YoLov3

The fundamental difference between YoLov4 and all its previous variants is that the detector stage has been bifurcated into two distinct sections consisting of a single stage – detector and sparse prediction stage. Following is an illustration of the same.

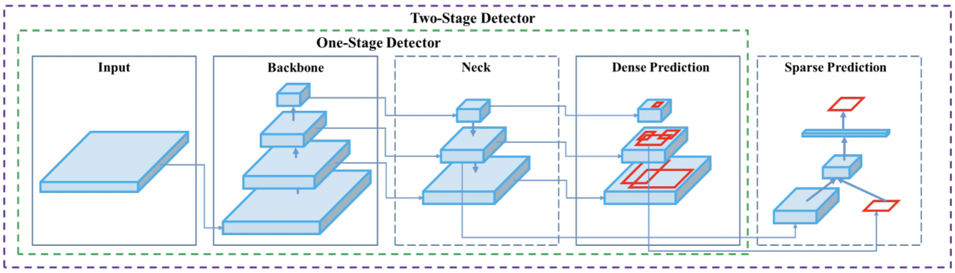


Figure 2.3: Two-stage detector setup used in YoLov4

After the initial input phase, we have **3** main sections as follows.

Backbone

This is the common name for the feature extractor section of the entire neural network architecture. It should be noted that all backbones are essentially classification models. With VGG16 being the most common and the earliest deep learning classifiers, SqueezeNet, ShuffleNET and MobileNET are used along with it. These classifiers are meant for the CPU only.

Neck

The neck is the common name for a feature aggregator in YoLo frameworks i.e. it collects the feature maps from various sections of the backbone stage. FPN, PAN, NAS-FPN, Fully-connected FPN, BiFPN, ASFF, and SFAM are all examples of feature aggregation blocks used in YoLov4.

Head

This is the final stage of the entire framework: a common name for the object detector stage in YoLo frameworks. It should be noted that this stage exclusively tells the region(s) in which (an) object(s) may be located, however, doesn't tell the class or label to which the object belongs. As stated earlier, these could be a single or two-stage detector, both anchor-based and anchor-free. Some common examples are YOLO, RetinaNet, SSD, Faster – RCNN etc.

As the number of examples in each stage shows, a typical YoLov4 framework can be implemented in any combination of input, backbone, neck and head. When there are such a large number of combinations possible, the best architecture should be an optimal combination of all the sections. This optimal choice would alone make it superior to YoLov3 in terms of **performance and accuracy**. This optimal combination should be arrived at by looking at the following parameters:

- Input resolution of images and their size
- Number of convolutional layers
- Number of parameters (hyperparameters) to be optimised
- Number of output layers or filters
- YoLov4 also provides something known as *Bag of Specials* and *Bag of Freebies* which are methods or functions for increasing the respective fields and mappings between backbone levels to detector levels. This gives us another region wherein optimisation can be done.

The final architecture that was deemed optimal for YoLov4 is as follows (with backbone, neck and head in order).

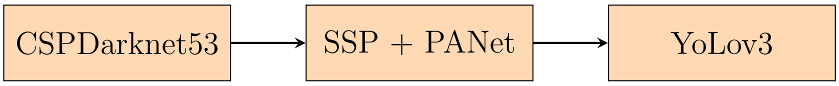


Figure 2.4: The final architecture presented in the paper

A couple of other combinations were derived but the FPS of the above combination superseded everything else. A summary of the FPS values of a couple of other possibilities is given below. Now we are going to see some special sections of the architecture (but are not limited to) in detail which make it more efficient than YoLov3.

Table 2.1: Neural network parameters for image recognition for 512 × 512 network resolution

Backbone model	Receptive field size	Parameters	Average layer size	BFLOPs	FPS
CSPResNext50	425 × 425	20.6 mn	1058 K	31	62
CSPDarknet53	725 × 725	27.6 mn	950 K	52	66
EfficientNet	1311 × 1311	12	668 K	11	26

Cross Stage Partial connection (CSP)

In any large neural network-based architecture, it is common for the last and the second to last layers to lose out on a lot of contextual features seen by the initial layers. A way out of this is to introduce the concept of skip connections so that the backpropagation of gradients to the initial layers can be done easily. DenseNet was initially considered for this but it had skip connections between every other layer which proved to be inefficient. So, the researchers stuck with their initial choice of CSPResNext50 and the CSPDarknet53 as far as the architecture is concerned. Following is an illustration of the same.

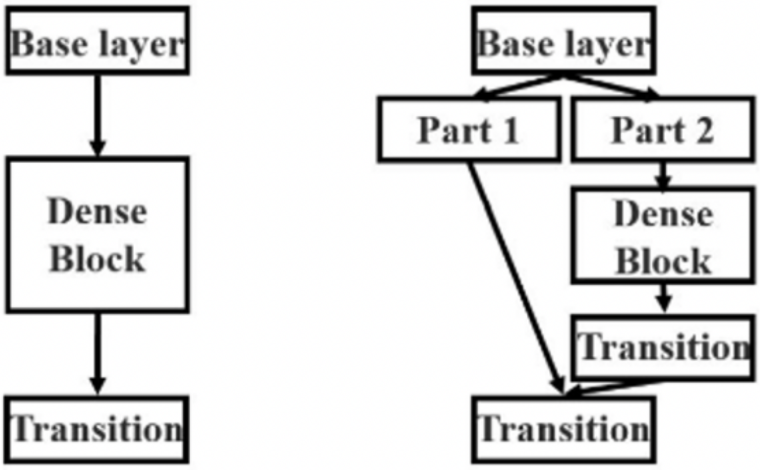


Figure 2.5: The two possible dense architectures: DenseNet and CSPDensnet

Self-Adversarial Training (SAT)

Deep learning is very much susceptible to *adversarial data* so YoLov4 uses SAT to introduce precise amounts of perturbation in the data such that the predicted label stays the same as the original label. This helps it easily achieve good accuracies for even augmentations of simple images.

All these discussions prove the benefits and advantages of YoLov4 over YoLov3 and hence it's choice for the project. As always, all things are better quantified in a graph of mAP versus execution time as shown below.

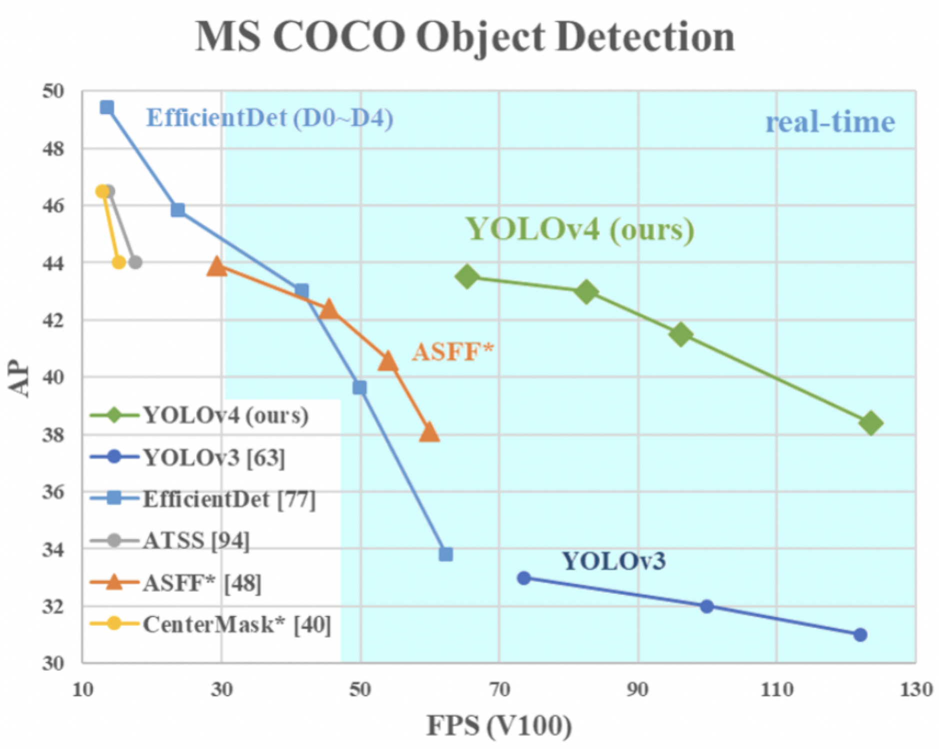


Figure 2.6: YoLov4 vs other state of the art object detectors

2.2 Performance metrics for evaluating object detection models [1]

The entirety of this report deals with a large number of performance metrics used to evaluate a variety of ML/DL models. Some are popular and have been used to evaluate such models for a long time while some metrics are empirical in nature because their output is a combination of a variety of models dealing with different classes of inputs and outputs which are working together. However, we would be specifically discussing metrics used for the evaluation of the performance of object detection models in this section.

2.2.1 Intersection over Union (IoU)

This metric equals the amount of overlap between the ground truth denoted by g_t and the predicted value by a particular model i.e. prediction denoted by p_d. It is to be noted that since this value is a ratio, it is a fraction consisting of numerator and denominator areas in appropriate units. Mathematically, it is given by

$$IoU = \frac{area(g_t \cap p_d)}{area(g_t \cup p_d)} \tag{2.1}$$

As it is very obvious for any arbitrary situation $0 \leqslant$ IoU $\leqslant 1$ wherein 0 corresponds to a null overlap and 1 corresponds to a full overlap. However, in almost all ML/DL models we don't go for such strict definitions rather we define what we call the IoU at α (or abbreviated as IoU@α). This means any value of IoU $\geqslant \alpha$ is considered a true positive whereas any value otherwise is considered a true negative.

Before we go ahead and define the next important metric we need to know a couple of different terms as stated below:

- **Precision** – The degree to which a model can identify only the relevant objects. It is simply the ratio of true positives and all the detections made by the model. It is given by
$$P = \frac{True\ positives}{All\ detections\ made\ by\ the\ model} \tag{2.2}$$
- **Recall** – The degree to which the model can detect all true positives amongst all ground truths. Mathematically, it is the ratio of the number of true positives and all ground truths and is given by
$$R = \frac{True\ positives}{All\ ground\ truths}$$

(2.3)

A good model has a high precision as well as high recall.

- **Precision–recall curve** – It is simply the plot of the variation of precision and recall against the variation of confidence values. A good model will give high precision values even when confidence is varied significantly.
- **Average precision** – When the area under the precision-recall curve is calculated at the IoU@α threshold then we get this value. Formally, it is defined using the following integral.
$$AP@\alpha = \int_0^1 p(r)\,\mathrm{d}r \tag{2.4}$$

Now we are at a stage where we can understand the next important metric

2.2.2 Mean Average Precision (mAP)

For each class, the average precision as defined above is calculated. This roughly translates to No. of average precision values ≈ No. of classes. When the average of these average precision is taken we get what we know as the mean average precision. For n classes we can simply calculate it as follows:

$$mAP@\alpha = \frac{1}{n}\sum_{i=1}^{n} AP_i \tag{2.5}$$

2.3 Guidelines for a good YoLo project

The following rules for a good project using YoLo are mere *thumb – rules* and are not some strict guidelines to be followed and should be evaluated on a case to case basis for every project. Additionally, it should be noted that such rules may not be applicable for implementation in every object detection project. The following rules are bifurcated into those undertaken during the training and those in the testing (detection) phase.

2.3.1 For training

(a) The values of `random` should be set to 1 in your `.cfg` file which allows training for multiple image or video resolutions simultaneously.

(b) Every distinct object that is liable for detection must have an appropriate label in the dataset.

(c) Precision may be increased by keeping the height and width of images or video frames as a multiple of 32.

(d) The training dataset should be such that every object to be detected corresponds to an exactly similar object in the training dataset. Similarity should be in terms of size, no. of classes to be detected (c), overall spatial orientation, overall illumination, augmentation (if any) etc.

If the size of the training dataset is N and the no. of classes is stated as above then training must run for at least Nc iterations.

(e) Training datasets should have as many **positive** examples as there are **negative** examples (i.e. images which don't have any object to be detected). When the detection or testing run is executed, such negative examples shall return no bounding boxes. These ensure an equal sensitivity of the model to both types of images as well as eliminate a lot of post-processing operations.

(f) Sometimes it is desirable to run your detections with the `-show_imgs` option at the end so that it can be manually verified whether the predicted bounding boxes are correct or not. Seeing the detections and detecting some anomaly could be a direct implication of training runs going wrong or some inherent problem in the dataset.

2.3.2 For testing or detection

(a) Increase network resolution in the same way as mentioned in 1. c.

(b) It should be noted that retraining is not required in the event of loss in your dataset or any other unintended corruption. Only the *darknet* command should be available which can be used to perform detections using the pre-trained `.weights` file which was trained on the 416 × 416 resolution images.

(c) To further enhance the accuracy, dataset training must proceed onto higher multiples of 32 such as 608 × 608 or 832 × 832. In the event of a memory overflow (`Out of memory`), `subdivisions` in your `.cfg` file must be increased from 16 to 32 to 64 and so on.

Chapter 3

Methodology and development

3.1 Video data

Chapter 4

Deployment in production

4.1 ML models in production

Rarely do data science projects in academia ever reach production and into a streamlined architecture. There are several reasons which prevent it from happening so. Indeed statistics say that roughly 87% of all data science projects never make it into production [2]. There are so many issues that hamper companies and developers to do so. We will be discussing only a few of them here.

It is seen that even in professional work environments, data scientists rarely have the access to the right kind of data. Either they face issues due to the network policy in a corporate environment or the data is simply out of their reach. Another issue that quickly catches up with this issue is that a large amount of data already exists in the public domain, however, it is unorganised and exists in a completely unstructured format. More precisely, the data exists in a mixture of various formats namely structured, unstructured and semi-structured data. Such types of data need to go through rigorous cleaning and upkeep mechanisms, the process flow of which is sometimes difficult to figure out. Along with these two issues comes a very humane issue regarding the very basis of how data scientists typically work: in an individual manner. However, putting data models into production happens to be the best example of a collaborative effort by a company and hence can't be single-handedly attempted by an individual. Apart from that technical issues of choosing the right framework, selecting the appropriate storage setup for your data and models etc. continue to exist.

We realise that the problems discussed above are as technical as they are humane. All these requirements gave rise to a completely new role of an MLOPs engineer [3]

Fortunately, for this project, the deployment of the data models falls within its current scope. All the ML models which would be trained and tested would be loaded onto the proprietary Ezmeral MLOPs platform by HPE. The servers for the same reside alongside the required hardware resources on SEBI's internal servers. Doing this (without going into the details) would make the entire process streamlined i.e. the process of uploading videos, and training models on new data as soon as it is available and at the same time would decrease the manual work involved while doing so. As soon as the deployment has been completed, whether for all the NEWS shows or even a few of them, the internal server would be able to serve requests from within the organization domain (POST methods with JSON body) and at the same time would be able to abstract all the complex code and scripts running in the background. Following is an illustration of the complete MLOPs lifecycle which is allowed by (or within) the Ezmeral MLOPs platform.

(insert appropriate image here)

4.2 A two-part deployment

This section deals with the needs of SEBI requiring a two-part deployment and the importance of each type in detail.

4.2.1 Ezmeral MLOPs

The remaining part of the literature survey deals with a proper yet brief study of the very exhaustive HPE Ezmeral official documentation to send these models to production. It should be noted that the Ezmeral platform is quite large and only a small part of it deals with proper MLOPs. An even smaller part concerns the project at hand due to the nature of the license procured by the governing organization (SEBI) i.e., we won't be looking at complicated use cases involving Kubernetes clusters or anything similar. Only the relevant parts of the study have been mentioned below. The entire Ezmeral architecture looks something like this:

(insert image of Ezmeral ml dl solution)

As per the above architecture following are some sequential steps to be followed while deploying a model to production: [4]

(a) Usually, the process begins with the setting up of a proper environment within the Ezmeral cluster. This involves the allocation of appropriate system privileges to various users in a system. Since it's a server-based system, it is customary to have multiple users and to allocate their various levels of access to the system. The platform admin is responsible for this job. After doing so, various required apps (which are essentially docker images) relevant to the application may be downloaded.

(b) After this the project repository is created and appropriate data loaded. This may include pre-written scripts, datasets for training and testing as well as pre-trained models (which may be improvised later on). It should be noted that all of these things belong to persistent storage and there would be no data loss in the case of a server outage.

(c) Followed by this an appropriate training cluster is to be allocated for a given project: the software concerned currently allows bindings with a variety of CPUs and VCPUs (called flavours). These have different specifications in terms of memory and speed.

(d) This is followed by actual programming in a Jupyter notebook attached to this cluster wherein after the initial phase, dedicated training regimes would utilise these cluster resources.

(e) Followed by the creation and training of a model within the attached Jupyter notebook, the programmer is expected to register the model for further training at a later stage. At this point, the model gets stored in the persistent section of the Ezmeral cluster and could be utilised later for further and more advanced training. Beyond this, in typical industrial use cases (such as this), scheduled scripts are just run which re-run this training procedure as many times as required.

(f) After a successful model registry, the server should be in a position to serve requests from the expected domains (in this case only requests from within the SEBI domain). To do so, again appropriate CPUs/VCPUs are assigned to balance the load, serve REST API requests etc.

(g) It should be noted that only the scoring script written in Python is run for any given request with the appropriate arguments provided in the JSON request. Hence, apart from the complex steps described above, the entire process boils down to a single Python script execution at the end.

Followed by this the server is ready to respond to a variety of requests from the concerned domain.

This concludes the discussion on the entire ML – DL workflow to be used on the Ezmeral MLOPs platform. However, we didn't go into details about server availability in terms of uptime or downtime nor did we go into the details of the amount of storage available to SEBI who has purchased this platform. It so turns out that although this discussion was not done, this is one of the most principal aspects of the deployment.

The Centos VM on top of which the entire Ezmeral cluster resides is used for a lot of tasks apart from the one described above. This leads to a low-frequency and high-latency solution which is not good for any MLOPs service use case. Therefore, it was decided that although this setup would be developed in its entirety, this is reserved for only single-usage queries for obtaining the output to only specific videos at a time.

The greater requirements at the time were to perform massive batch processing workflows which are freely running and would manage to process several hundreds of videos in sequence. Hence, the second part of the deployment concerns a single VM workflow as described in the following section.

4.2.2 Single VM workflow

To overcome the single-usage and as-in-required constraints described above, we have decided to go for an automation script that calls the main Python script repeatedly to process the bulk of videos that are in turn fetched from a remote server. In this case, it is SEBI's Datalake server.

The concerned setup involves a Linux VM running Centos 8.2 which is relatively ideal and is not involved in unnecessary workflows apart from the concerned one. All the required assets i.e. code (the main Python script aka the scoring script in the above section), data (ML models and image templates or masks) and excel files containing a readymade listing of companies are to be made available on the VM. Apart from this the automation script (which is also written in Python) resides on the VM. The logic inside the automation script is segmented into the following parts.

- Connecting and disconnecting to the remote server.
- Fetching and sending files from and to the remote server respectively.
- Performing normal command-line actions on the VM on which it resides: running the Python scoring script repeatedly with command-line arguments and deleting unnecessary files after a single execution run.
- Show the current stage of the workflow which is under progress and display a final summary of the total no. of videos that have been processed successfully and(or) the number of videos that have failed.

We shall elaborate on the last point a little further. Execution of the scoring script on a single video file leads to the production of a single CSV file at the location of the VM and produces a very large log file wherein the entire output of the script was redirected. The CSV file and the log file need to be sent back to the remote server followed by which they are to be deleted from the VM. Since the video file also doesn't have any further use, it is also deleted subsequently. The correct order in which this is to be done is elaborated in the last section of this chapter as a guideline for similar scenarios. Upgrades have been planned for this workflow as detailed further in section 5.6.3.

4.3 Overall software flow

(insert appropriate flowchart)

Step 1: Open a connection with the remote SEBI Datalake server. Exit the program if a timeout error occurs with an appropriate exit code.

Step 2: Fetch the next video file in sequence from the remote server.

Step 3: Run the Python scoring script on the same. This will lead to the production of a log file. And a CSV file only if the execution is successful.

Step 4: Delete the video file.

Step 5: Send a copy of the log file to the server and delete its instance from the VM.

Step 6: If **Step 3** was successful, send a copy of the generated CSV file to the remote server and delete its instance from the VM.

4.4 Guidelines for deployment

This section presents a set of guidelines for the two types of deployment detailed in sections 4.2.1 and 4.2.2.

(a) Concentrate on three fundamental aspects of the MLOPs solution namely how the data is going to be stored and retrieved, what frameworks are applicable and(or) available for your job and finally the iterative processes involved in your MLOPs solution [3].

(b) Condense your workflow to a few specific data types in which your input data is to be stored e.g. .mp4 files as in this case.

(c) Be careful with the amount of data to be stored: look for the best solution fitting your budget.

(d) Classify whether you would be dealing with batch workflows or predictions by your ML models in real-time as detailed in sections 4.2.2 and 4.2.1 respectively as this shall greatly affect your outcome.

(e) If you are dealing with batch workflows as in section 4.2.2, the order of fetching and deleting files is paramount. Classify files into input, files produced irrespective of the status of the result and files produced only on successful execution. Order them by size and the sequential points of failure. Delete the largest input file(s) first followed by sequential cycles of sending files to the remote server and deleting them.

Chapter 5

Results and discussions

The following subsections shall discuss in detail, a variety of important comparisons which affect the project work, a brief overview of the methodologies learned, deviations from the ideal case (if any) and a creator's road map for implementation of a similar project and other moderately or less important details. It also has a section for the work which is being currently done at SEBI to improve the existing system. The initial two sections deal with metrics of accuracy and execution time across two different bases for comparison while the section that follows discusses why the metrics are chosen and defined as they are.

5.1 Comparison between OCR models

Both Tesseract and EasyOCR are sophisticated OCR engines and are available as open-source in their default versions. However, there exists a large number of differences in how they have been developed. While Tesseract [5] happens to be a quite superior alternative right from the beginning which can use even PDF files as input apart from the standard images, it turns out that there doesn't exist any suitable interface with common programming languages (such as Python) which leads to a lot of problems. The first and foremost problem is that Tesseract needs to be set up in a way similar to how a particular program is set up on a remote server i.e. the only link between a program using Tesseract and the actual engine is the file path. Customisations can be done nonetheless, but are difficult to implement due to the lack of a high-level interface. This leads directly to another issue: the Ezmeral MLOPs platform is set up on a legacy Centos 7.5 virtual machine cluster and won't allow arbitrary software to be installed in it due to the presence of an enterprise-grade security architecture. So it may be the better OCR engine in general, however, this use case is unsuitable for it.

EasyOCR offers as many capabilities [6] as Tesseract (sans the direct processing of PDF files) and has a high-level library directly available for use in Python. This means that no preliminary server like set-up needs to be done to access the OCR engine, rather everything is on board once the program and all its imported modules are loaded onto memory. Hence, EasyOCR was chosen as the OCR for this project.

It should be noted that EasyOCR is a ready-to-use OCR that is invariant to colour images while the Tesseract documentation does say things about providing a greyscale image as input. This feature alone avoided a lot of extra image processing operations within the codebase. After using EasyOCR, the following changes took place:

- The size of the codebase was greatly reduced.
- The execution time dropped significantly.
- Major improvements were not visible in accuracy although EasyOCR still managed to perform better.

Amongst all decoded values at the end of a particular run, the following results were obtained and are summarised below.

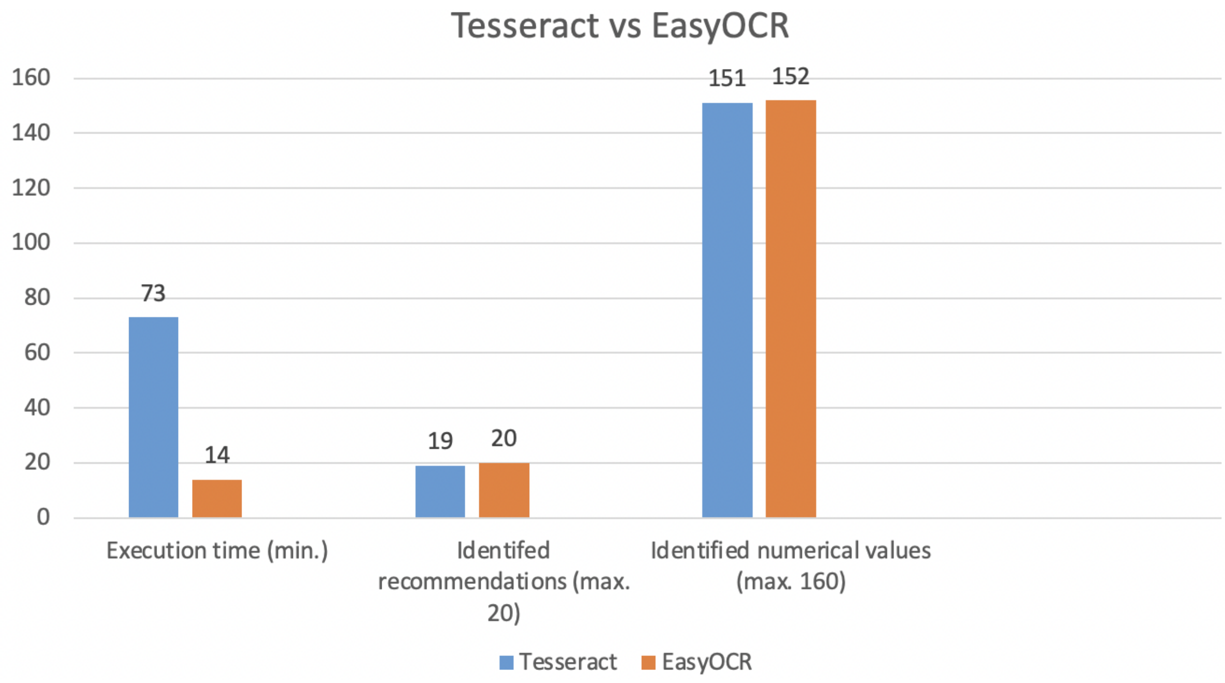


Figure 5.1: Comparison of Tesseract and EasyOCR for a sample 30 minutes telecast

Still, EasyOCR (and the project) benefits even further from a possible change in hardware configuration which is detailed in the following section.

5.2 Comparison between hardware architectures

It is a known fact that almost all deep learning systems (inclusive of computer vision models) benefit greatly from the presence of GPUs in the development environment. Although as of now the Ezmeral cluster doesn't have GPU access, testing was still performed on a local workstation with a RTX 2060 graphics card with the equivalent Nvidia CUDA toolkit installed.

It should be noted that no execution environment discussed earlier in chapter 4 had any mention of GPUs so there was a decrease in execution time by a large margin for the same test video. The decrease of roughly by a factor of 4.5 w.r.t. to the closest competitor i.e. for execution recorded on a standalone VM. The reason for this was obvious: the documentation of EasyOCR mentions that their pipeline is more performant in the presence of a GPU [7]. Following are the results summarised in the form of a bar chart.

(insert appropriate bar graph)

5.3 Current workflow progress and testing procedures

5.3.1 Workflow progress

As stated in earlier chapters, the work at hand is to process a total of 7 NEWS shows in their entirety (excluding only one for the lack of recommendations in it) to summarise all the recommendations being provided in them. This shall ensure that when more sophisticated anomaly detection models are run on them, the input (produced as an output from my project) is sufficient for them to run properly. Additionally, it should also be noted that the systems succeeding the ones used in this project also use several data cleaning procedures before arriving at a final output so a picture-perfect output is not necessary although desirable.

The following table lists the shows on one side and the percentage of total videos of each show that has been processed.

(insert appropriate table)

5.3.2 Testing procedures

Usually, there are no clearly defined testing procedures for such custom projects in deep learning and computer vision. A typical testing procedure that was used to verify the authenticity and the accuracy of the output is that after filtering out frames at the FPS rate, the recommendations being made in a handful of videos were noted down manually. This was then compared to the output obtained from the program. If there is a perfect match (which was mostly the case), then the output is deemed to be suitable otherwise no separate improvements are made midway in a batch processing workflow. The reason for this is two-fold:

(a) Batch processing workflows involve a very large number of videos at once, stopping them in between drastically decreases the efficiency of such workflows and the more important reason being

(b) Anomaly detection systems or models which follow this project already employ sufficient data cleaning procedures to obtain a reliable and usable output within tolerable error limits.

5.4 Inferences and learnings

5.4.1 Possible usage of only an OCR engine

It would be worthwhile to reason why an OCR engine is not directly used on the extracted frames since it would detect the required text followed by which the programmer is supposed to consolidate it to a usable form.

The reason lies in the abilities of EasyOCR as an OCR engine which is both an advantage as well as a disadvantage. It should be known that, unlike Tesseract which uses page segmentation modes [8] to detect text in an image structurally arranged in specific configurations, EasyOCR doesn't have any such facility. It directly returns a list of bounding box coordinates, the text detected along with a confidence score across the entire image [7]. Following is an example of what would happen if signboards containing multiple texts are passed to EasyOCR.

(insert appropriate image)

We would realise that all texts scattered throughout the image are returned, however, we are unable to draw any correlation between them at first look. This could prove to be a major challenge in this project wherein multiple related numerical and alphanumeric values are placed close to each other. In an extreme case where multiple recommendations are detected in the same frame, it would be very difficult to detect the mapping of the numerical values (such as stop-loss or target price) and the listing symbol of a company share. Hence, the choice of using an object detection model followed by an OCR engine.

5.4.2 Procurement of data from a contractor

It has been clearly stated in section 3.1 that the entirety of the video data was obtained from Clipbyte. It would be worthwhile to argue that all videos could be directly downloaded from a platform like YouTube. However, there are two major reasons for not doing so:

(a) All broadcasts of the concerned NEWS shows are not available on such video platforms and nor do the respective channels upload videos in a timed and orderly manner.

(b) There is no official way to download videos from such a platform so availing videos from such a platform by a Government regulatory body would imply improper means of accessing publicly available information in the situation wherein the InVestigations Department (IVD) of SEBI has to file a case against a suspected entity. The data source needs to be mentioned in the case file and hence a separate contractor has been tended for this job.

5.5 Accuracy, empirical accuracy and execution time

A lot has been discussed about the various performance metrics used to evaluate object detection models in section 2.2, however, we still stick to the concept of simple empirical accuracy in the above two sections. The reason for the same is that there are multiple ML/DL models which are involved in this project. Each model has been designed (or is being used) to complete only a specific task amongst a large number of tasks that are to be executed sequentially. Each task involves different kinds of data and hence produces different kinds of output.

A simple e.g. being that the YoLo models discussed throughout the report output a set of mixed values as given in section 2.1 after taking in an image as an input. However, the OCR models that follow require an image as an input, but produce a textual output (i.e. of string datatype). These models within them have two deep learning models i.e. a text detection and a language model in a cascaded fashion out of which only the former receives an input in the form of an image while the second model deals with textual input and output.

Each of these models named above has its own set of accuracy metrics which may be discussed however, they would be irrelevant to the final output of the project. Hence, we have stuck to the concept of simple empirical accuracy which is defined by the set of values obtained at the end from the OCR and summarised in the CSV file and is defined as follows

$$A_e = \frac{No.\ of\ NA\ values\ in\ the\ CSV}{Total\ no.\ of\ values\ in\ the\ CSV}$$

(5.1)

Measuring execution time remains the same i.e. in seconds although at a couple of places wherein the execution time is large, seconds have been replaced by larger units of time.

5.6 Future work

The current system can be improvised in two different areas as given below. A workflow improvement can also be made.

5.6.1 Accuracy and efficiency

The current system does filtering at only the FPS rate which manages to reduce the number of frames under consideration to a much smaller value (a 25^{th} of it to be precise). However, as we have seen in chapter 3, the frame selection is still not inherently intelligent and still leads to the production of ≈ 1400 frames on an average, amongst which frames would be manually selected for annotation.

Theoretically, we can say that for the problem statement at hand; there exists a program P that extracts and processes exactly N frames such that the total no. of recommendations being made, R is contained in exactly N keyframes.

To do so we can use an adaptive frame extraction algorithm [9] which shall extract only the frames which are important to us. This method is based on matching histograms of successive frames, a correlation that exists amongst their colour channels, etc. This shall greatly reduce the number of frames to be processed at a time (apart from the already existing FPS rate filtering). This method, however, could be tending to fail because similar-looking frames need not be redundant to us just because they appear close to each other in the actual video. In that case, we can go for an alternative approach wherein the frame numbers are output according to a simple implementation of any multi-output regressor [10]. Since there could be inherent inaccuracies in the training of the same, we can use a spreading factor δ which shall extract keyframes starting from $f_k - \Delta$ to $f_k + \Delta$ instead of just the frame f_k.

5.6.2 Usability

Currently, a two-part deployment and its necessity are deemed to be confusing to many SEBI officers who work in the surveillance department and don't have any prior programming knowledge. Both these deployments could be integrated into a single native cross-platform desktop app developed either using PyQt or Electron which shall have an intuitive front end design and the same set of calling scripts as its backend. This shall help concerned officers keep track of ongoing workflows as well as use them when required without interfering with ongoing batch workflows and being aware of the complexity of the ML/DL code behind it.

5.6.3 VM workflow

The landing server of SEBI Datalake would soon be modified in a way that it will subscribe to an MRSS feed on the Clipbyte platform wherein any new video once pushed shall automatically trigger a workflow. The VM workflow would be modified accordingly so that it either runs as a cron job at scheduled intervals or it executes only when a new video is available for processing.

Chapter 6

Conclusions

The various recommendations being made in NEWS shows were successfully detected (within tolerable error limits across a large size of input data), extracted and analysed for anomalies in the concerned process. The input to the program script was simply the video file itself and the telecast date (which was reflected as it is in the last column). The summarised data was available to the end-user in either an excel file or CSV file for further analysis. The final summarised data contained important fields like the Name of the analyst, the share/stock regarding which recommendation was being made: its name as well as its listing symbol in the appropriate exchange, the actual recommendation being made (buy, sell or hold), the target price and the associated stop loss. A total of eight NEWS shows were processed in this manner wherein only a single NEWS show was skipped due to the lack of recommendations being made in the video dataset available to us.

As stated earlier in chapter 4, the deployment of the above ML/DL models was parallelly done onto two separate systems, namely the HPE Ezmeral MLOPs server for single usage low-frequency requests and on a separate Linux virtual machine for large free-running batch processing workflows which would keep on running a single script continuously on a large dataset.

I got to learn a lot of things regarding machine learning, deep learning for computer vision, functioning of OCR engines, MLOPs deployment in different environments and as well as different guidelines which are to be followed for gathering a video dataset (e.g. sampling at FPS rate, annotating a dataset, partitioning the dataset properly etc.) and processing it accordingly for further analysis. Overall it turned out to be a complete knowledge enriching experience in a multitude of domains.

Bibliography

[1] Kiprono Elijah Koech. *Object Detection Metrics With Worked Example*. English. Aug. 2020. URL: https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e.

[2] VB staff. *Why do 87% of data science projects never make it into production?* English. Venturebeat. July 2019. URL: https://venturebeat.com/2019/07/19/why-do-87-of-data-science-projects-never-make-it-into-production/.

[3] Rising Odegua. *How to put machine learning models into production*. English. Stackoverflow. Oct. 2020. URL: https://stackoverflow.blog/2020/10/12/how-to-put-machine-learning-models-into-production/.

[4] Terry Chiang. *Tutorial 1: Income Prediction (XGBoost)*. English. HPE. Sept. 2020. URL: https://docs.containerplatform.hpe.com/52/reference/epic/working-with-ai-ml/tutorials/Tutorial_1_Income_Prediction.html.

[5] Google. *Tesseract OCR*. English. Google. July 2015. URL: https://tesseract-ocr.github.io/.

[6] Jaided AI. *EasyOCR*. English. Jaided AI. Mar. 2020. URL: https://github.com/JaidedAI/EasyOCR.

[7] Jaided AI. English. Jaided AI. URL: https://www.jaided.ai/easyocr/tutorial/.

[8] Adrian Rosebroc. *Tesseract Page Segmentation Modes (PSMs) Explained: How to Improve Your OCR Accuracy*. English. pyimagesearch. Nov. 2021. URL: https://pyimagesearch.com/2021/11/15/tesseract-page-segmentation-modes-psms-explained-how-to-improve-your-ocr-accuracy/.

[9] Naveed Ejaz, Tayyab Bin Tariq, and Sung Wook Baik. “Adaptive key frame extraction for video summarization using an aggregation mechanism”. In: *Journal of Visual Communication and Image Representation* 23.7 (2012), pp. 1031–1040. ISSN: 1047-3203. DOI: https://doi.org/10.1016/j.jvcir.2012.06.013. URL: https://www.sciencedirect.com/science/article/pii/S1047320312001095.

[10] Jason Brownlee. *Deep Learning Models for Multi-Output Regression*. English. Machine Learning Mastery. Aug. 2020. URL: https://machinelearningmastery.com/deep-learning-models-for-multi-output-regression/.