Characterizing the effectiveness of COVID-19 mitigation measures: A data-centric approach

Lukman O. Olagoke
adderbyte@icloud.com

Ahmet E. Topcu
College of Engineering and Technology
American University of the Middle East, Kuwait
ahmet.topcu@aum.edu.kw

ABSTRACT

COVID-19 represents a serious threat to both national health and economic systems. To curb this pandemic, the World Health Organization (WHO) issued a series of COVID-19 public safety guidelines. Different countries around the world initiated different measures in line with the WHO guidelines to mitigate and investigate the spread of COVID-19 in their territories. The aim of this investigation is to use a data-centric approach to quantitatively weigh the effectiveness of these control measures. We begin our investigation with a simple text analysis of coronavirus-related articles and show that reports on similar outbreaks in the past strongly proposed similar control measures. This reaffirms the fact that these control measures are in order. Subsequently, we propose a simple performance statistic that quantifies general performance in light of the different measures that were initiated and implemented. This would help caregivers identify effective control measures and subsequently reduce COVID-19 spread while at the same time preventing undue stress on intensive care units. Lastly, a COVID-19 web-based time-line visualization that enables comparison of performances and cases across continents and subregions is presented.

KEYWORDS

COVID-19, performance statistic, pandemic, data-centric engineering, clustering, SARS-CoV-2

1 INTRODUCTION

The novel coronavirus 2019-nCov was first isolated in December 2019 from samples from a cluster of patients in Wuhan, China, who had shown unknown symptoms of pneumonia [1]. 2019-nCov is different from severe acute respiratory syndrome coronavirus (SARS-Cov) and Middle East respiratory syndrome coronavirus (MER-Cov), and it is the seventh member of the family of coronaviruses that infects humans [1]. This new virus was later designated as severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) [2].

On January 30, 2020, the World Health Organization (WHO) [3] assessed the risk of the COVID-19 outbreak as very high for China and high at the global level. In the aftermath of that assessment, new cases of COVID-19 continued to emerge across different regions as reported by the web-based interactive online tracker proposed by Dong et al. [4].

Concerned by the increasing number of cases and the severity of the disease, the WHO declared COVID-19 a pandemic on March 11, 2020 [5]. They also proposed a series of public measures to help curb its spread [6].

The governments of different countries responded with various implementations of the proposed measures in order to mitigate the spread of COVID-19. Datasets about such measures for each country have been provided on the platform of the United Nations Office for the Coordination of Humanitarian Affairs (OCHA) [7]. These pandemic control measures can impact the effective reproduction number (R_e) [8], which is the expected number of new infections caused by a single infected individual at time t in a partially susceptible population [9], and can thus help mitigate the spread of the virus.

Moritz et al. [10] have shown that these control measures have been important for mitigating the spread of COVID-19 in China. Border control measures have been reported to be efficient against the spread of the virus from mainland China but are insufficient to contain the global spread of COVID-19 [11]. Anderson et al. [8] opined that individual control measures (for example, early self-isolation) are as important as government actions to mitigate the spread.

A number of uncertainties about COVID-19 have made estimating the effectiveness of the measures or performance under specific control measures difficult. These uncertainties include the rate of cases of infection, identifying the onset of infectiousness, fatality rates, the population size of asymptomatic patients, and the duration of the infectious period. In this paper, a data-centric approach is proposed for the analysis of performance under such control measures. More specifically, we introduce a simple statistic that uses the daily numbers of cases of infection and recovered cases for characterizing performance. In addition, a web-based interactive visualizer is provided to visualize the results. This performance measure could help understand the impact of a specific control measure or understand when to ease or re-instate certain control measures.

2 BACKGROUND

The online real-time COVID-19 tracker provides a good visualization of coronavirus cases across the globe on a daily basis [4]. The numbers of confirmed cases, recovered cases, and cases of death are reported daily. In addition, a web-based visualization of control measures is also available [12]. The measures identified include the following:

- Social distancing
- Government and economic measures
- Lockdown
- Movement restrictions
- Public health measures
- Humanitarian measures

The datasets from both of the above platforms were utilized to carry out the analysis reported in this paper. We also present a novel web-based interactive platform that provides both the number of cases (recovered cases, cases of death, and performance measures) and information on control measures. This will be explained in a later section.

Control Measure Investigation Prestudy

In this prestudy, the goal was to understand the major conclusions from previous coronavirus-related outbreaks. To achieve this, the COVID-19 Open Research Dataset [13] was used. We carried out a text analysis of the conclusions of the articles in that dataset.

First, sentence data were tokenized. Secondly, a list of words such as COVID-19 and coronavirus were used as our test strings. Only sentences containing our test strings were retained and ranked based on frequency while the rest were discarded. The top most frequent sentences were then saved.[1] Some of the particularly interesting conclusions found are listed below:

- Preventive measures are important, such as vaccinations, hand-washing, and isolation of affected individuals in hospitals and long-term care facilities [14].
- New coronaviruses that affect both animals and humans continue to emerge or reemerge because various animal reservoirs sustain them, particularly birds and bats, which exist in populations with high volumes and are extremely mobile. [15].
- It affects all individuals regardless of their age and adherence to preventive measures against respiratory infections including use of face mask, hand hygiene, social distancing, and vaccination [16].
- The symptoms that are most commonly reported in association with coronaviruses, although not unique to them, are as follows: chills, headache, malaise, cough, sputum production, sore throat, and nasal congestion. Among older groups of patients, shortness of breath was also commonly seen, while among younger groups of patients, muscle pains were more common [17].

This prestudy shows that similar preventive measures have been proposed for similar outbreaks (for example, MERS-CoV) in the past. The preventive measures proposed are also found to be in line with the recommendations of the WHO [6]. Other conclusions from the prestudy can be found in the linked GitHub repository.

3 METHOD

Intuition for Performance Statistic

Consider the illustration below: We know that control measures influence the rate at which new cases arrive at the "recovered" section (color red) in Figure 1. Furthermore, there are factors and uncertainties that influence the rate at which new cases arrive at the "confirmed case" section (color blue). If the marginal (or incremental) confirmed cases become greater than the marginal recovered cases, the slider shifts to the right. In the same way, the slider shifts to the left if the marginal recovered case is greater than the marginal confirmed case. The target is to keep both rectangles empty or perpetually make the recovered rectangle have a high marginal increase rate. Note that we are interested in the daily change in the contents of both rectangles. This is achieved by subtracting the total value at t_{n-1} from the total value at t_n. This means for example, if the total reported confirmed case for today is x and y for yesterday, the marginal or incremental difference becomes $x - y$. Hence, performance can be calculated as shown below:

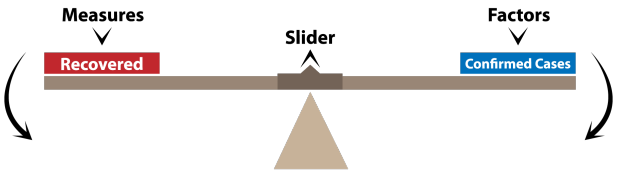


Figure 1: Recovered versus new cases.

$$\mathcal{P} = \Delta R - \Delta C, \tag{1}$$

where $\mathcal{P}$ is the performance statistic, ΔR is the difference in recovered cases, and ΔC is the difference in confirmed cases for time interval Δt.

It is assumed that both ΔR and ΔC are nonnegative integer values. This makes sense since the confirmed or recovered cases would either remain as they were yesterday or else increase today.

[1] To reproduce this result, visit the repository at https://github.com/adderbyte/covid_19_response/blob/master/computationalModelling/Preprocesing/Covid_Viz.ipynb.

Performance Metric

To define the performance metric, we normalize the formulation for $\mathcal{P}$:

$$\mathcal{P} = \frac{\Delta R - \Delta C}{\Delta R + \Delta C}. \tag{2}$$

Note the following:

- The normalizing ensures that the values for the performance are restricted between 1 and -1. This way, the slider moves within a specified range.
- If both ΔR and ΔC are zero, then the performance statistic is not defined. We will adjust to accommodate for this case.
- Extreme values are reached if either ΔR or ΔC is 0.
- If the cases are at nonzero equilibrium (same rate of change), then movement between the recovered and confirmed cases is allowed (factors or measures might cause such movement to occur and push the system away from equilibrium).

To address the problem of ΔR and ΔC, we introduce ϵ. In addition, we take the exponential in order to use the value 1 as baseline and also to provide for easily visualizable figures:

$$\mathcal{P} = \exp^{\left(\frac{\Delta R - \Delta C}{\epsilon + \Delta R + \Delta C}\right)}, \tag{3}$$

where ϵ is of the order 10^{-12}.

4 RESULTS

Performance Visualization

Figure 2 shows the performance statistic plots for different countries. A Savitzky–Golay filter [18] was applied to smooth out the data points without distorting the underlying signal. Red dots signify the date on which social distancing (a control measure) started in respective countries. It is observed that the performance for most controls declined in the early days of March 2020. In response to this decline, a large number of these control measures were initiated at about the same time. Perhaps knowledge of this performance time-line could have helped caregivers and policy-makers respond earlier with appropriate measures at the start of the decline of performance.

Web-based Interactive Platform

The interface of the web-based interactive platform is shown in figure 3. The platform provides the following:

- Selection of multiple countries across different continents and subregions, and visualization of their performance statistics. Upon hovering, it displays the plot for cases of death (first difference) and confirmed cases (first difference) for the selected country.
- Display of the control measure of interest in the red plot. This can be changed with the provided drop-down menu.
- Option to switch between log and linear scales.
- Option to select the time window to display using the date-time range slider.

The aim is to provide an interface for visualization not only using the performance index but also using the daily deaths and recovered cases. Accordingly, performance plots with the cases of death and recovered cases can be compared.

5 DISCUSSION

Performance Metric Validity

Given a control measure that influences the performance statistic positively, we expect the number of cases of death to decrease marginally. Such a positive impact will induce an increase in the recovery rate, thereby causing a decrease in cases of death and a marginal decrease in confirmed cases. If the marginal decrease were to increase, then the performance would not show a steep increase.

We used a bootstrap approach [19] to show that the performance method achieves this negative correlation. In other words, we compute the correlation between the performance and cases of death using bootstrap samples. The outline of the algorithm used is shown in Algorithm 1.

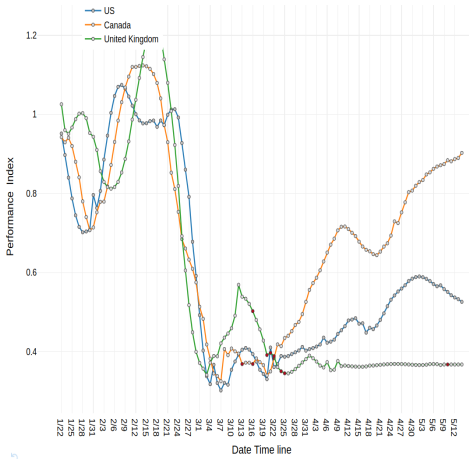
The correlation coefficient is a product-moment Pearson correlation coefficient. It is a measure of the strength of the relationship between two variables and values range from -1 to 1. A value of 0 indicates that there is no correlation between the variables, a value greater than 0 indicates a positive correlation, and a value less than 0 indicates a negative correlation.

The results for the Pearson coefficient are shown in Figure 4. As expected, the performance statistic has a significant negative correlation with the cases of death.

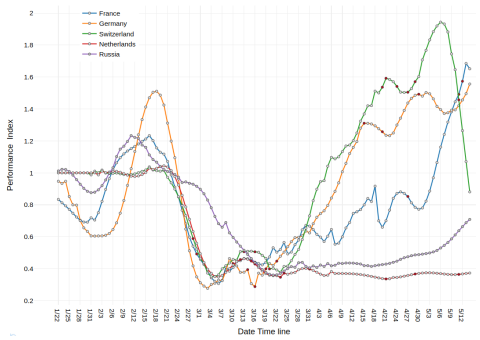
Clustering Results

It would be interesting to cluster countries based on observed performance. The first step is choosing the best algorithm for such a purpose. Both a form of hierarchical clustering, hierarchical density-based spatial clustering of applications with noise (HDBSCAN) [20], and K-means clustering were tested. K-means clustering requires specification of the number of clusters and is more sensitive to noise in the dataset. HDBSCAN is robust to noise and can automatically uncover the number of clusters in the data without prior setting of the number of clusters. HDBSCAN was therefore selected for clustering the time-line of performance scores. HDBSCAN considers the data as a weighted graph where the data points are the vertices. The edges between any 2 points are weighted by the mutual reachability distance between those points. The steps for HDBSCAN are outlined below [20]:

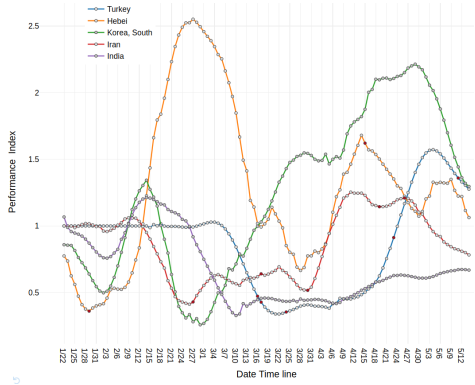
- Input transformation (based on density/sparsity).
- Build the minimum spanning tree of the density graph.



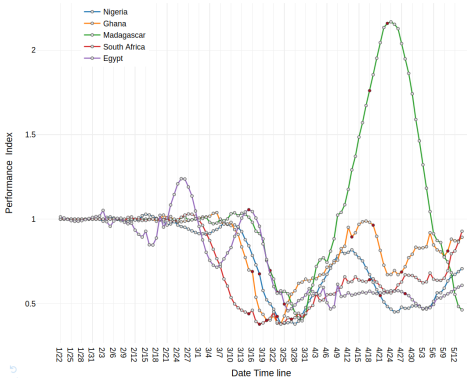
(a) US, UK, and Canada performance plot



(b) Countries across Europe



(c) Countries across Asia



(d) Countries across Africa

(e) Latin American subregion

Figure 2: Performance plots from 3/12/2020 to 5/13/2020 for some countries across different continents and subregions

- Condense the cluster hierarchy based on minimum cluster size.
- Extract the stable clusters from the condensed tree.

The clustering of countries based on performance is shown in Figure 5. This was achieved using a minimum cluster size of 10, minimum samples of 5, and leaf size of 270. The minimum cluster size is the size of the smallest grouping that will be considered a cluster and it is the most important parameter that needs careful fine-tuning. In Figure 5, the results of the clustering are presented. Countries and subregions falling within each cluster were retrieved. Countries

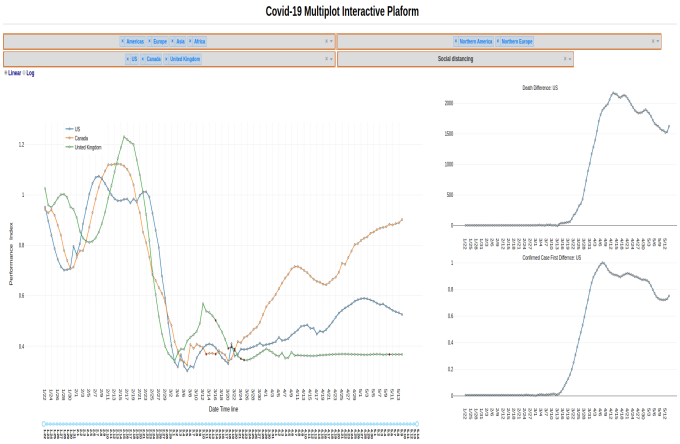


Figure 3: Interface of the interactive multiplot platform.

Algorithm 1: Proposed Algorithm.

Output: correlation_coefficient
Input: iter, N, countriesList, confirmed_case_data, death_case_data, recovered_data, corrcoeff
Initialize: $metricSave = [] , diff = [], i = 0$

```
/* loop over iter times                                    */
for ( i < iter ) do
    sampleCountry = select N random countries from
      countriesList ;
    /* iterate over all selected countries (bootstrap
       samples)                                            */
    for (j in sampleCountry) do
        /* the compuation of P is as defined in this
           report and it makes use of the data on
           recovered cases and confirmed cases             */
        compute P ;
        /* the death_diff makes use of the cases of
           death. Normal first difference                  */
        compute death_diff;
        /* append computations to list                     */
        put P in metricSave ;
        put death_diff in diff ;
    end
    i = i + 1
end
/* mean along rows of metricSave and diff                  */
sumMetric = mean(metricSave)
diffMetric = mean(diff)
/* compute correlation coeff using corrcoeff function
   */
return corrcoeff(sumMetric, diffMetric)
```

like the United Kingdom, Italy, and Spain fall into cluster 1.

Figure 4: Correlation plot.

The USA, Beijing, Macau, South Korea, Thailand, Hubei, and some other Chinese provinces fall into cluster 2. Clusters 1 and 2 could be regarded as representing those countries and subregions that are in a steady or stagnant state or trying to move up the performance scale. Perhaps they started off well but suddenly had a spike in the number of cases. France, Singapore, Vietnam, and Hong Kong are among the countries and subregions in cluster 3 and have very high performance scores. Cluster 0 is the largest and represents countries that are on good footing, including New Zealand, Turkey, Austria, and others. The outliers belong to cluster -1 and this includes countries like Canada, Germany, Belgium, and a few others. Further fine-tuning may reduce the number of elements in the outlier cluster. The outliers are also not doing bad on the performance scale.

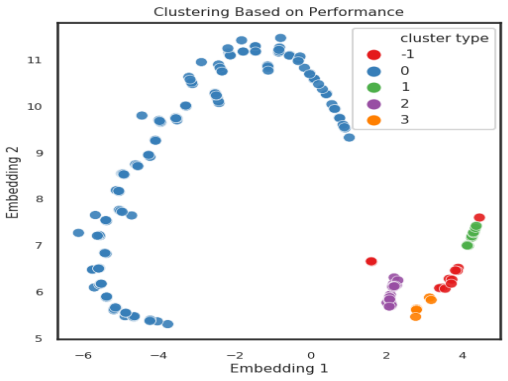


Figure 5: Clustering of the performance index for all countries and subregions in the dataset.

It should be noted that while HDBSCAN was used for clustering, the rendering of the plot was done using T-distributed stochastic neighbor embedding (t-SNE) [21]. For this reason, the x- and y-axes carry the labels of embedding 1 and 2, respectively, because they are the output from the low-dimensional projection of the cluster points using t-SNE.

An iteration of 5000 steps and perplexity of 80 was used as advised by Wattenberg et al. [22].

6 CONCLUSION AND FUTURE WORKS

Conclusions

We have proposed a performance statistic that is useful for quantifying the effectiveness of government measures. In addition, we have shown that this statistic has one very important characteristics: a negative correlation with the daily cases of death. The aims of governments across the globe, first and foremost, are to minimize cases of death and flatten the area under the epidemic curve of the COVID-19. The performance index can help quantify how response measures have helped achieve these aims. The interactive web visualizer provides easily digested and quick feedback to augment decision-making processes in the COVID-19 response measures evaluation. In trying times such as these, there is a need for effective response and evaluation of the different efforts by caregivers to stabilize the situation. There is also the need to avoid undue stress on the health system and caregivers themselves. All these would require some methodologies to critically evaluate COVID-19 control measures and countermeasures. We believe that this study provides a step in the right direction for such a quantitative evaluation.

Future Work

It is desirable that future research investigate the effects of the measures and factors directly on the performance index as illustrated in Figure 6.

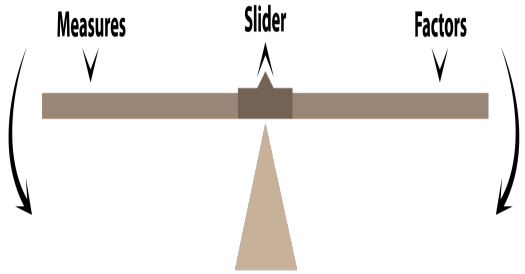


Figure 6: Illustration of new model

With respect to Figure 1, one observes that Figure 6 assumes the confirmed and recovered COVID-19 cases are no longer present and are therefore not used as input into the model. A possible future research path based on this illustration may be summarised follows:

- Building a model that uses the factors and measures to predict the performance.
- Measures and factors importance analysis.

REFERENCES

[1] Na Zhu, Dingyu Zhang, Wenling Wang, Xingwang Li, Bo Yang, Jingdong Song, Xiang Zhao, Baoying Huang, Weifeng Shi, Roujian Lu, Peihua Niu, Faxian Zhan, Xuejun Ma, Dayan Wang, Wenbo Xu, Guizhen Wu, George F. Gao, and Wenjie Tan. A novel coronavirus from patients with pneumonia in china, 2019. *New England Journal of Medicine*, 382(8):727–733, 2020. PMID: 31978945.

[2] Coronaviridae Study Group of the International et al. The species severe acute respiratory syndrome-related coronavirus: classifying 2019-ncov and naming it sars-cov-2. *Nature Microbiology*, page 1, 2020.

[3] World Health Organisation. *WHO update Timeline*, 2020 (accessed May 18, 2020).

[4] Ensheng Dong, Hongru Du, and Lauren Gardner. An interactive web-based dashboard to track covid-19 in real time. *The Lancet infectious diseases*, 2020.

[5] D Cucinotta and M Vanelli. Who declares covid-19 a pandemic. *Acta bio-medica: Atenei Parmensis*, 91(1):157–160, 2020.

[6] World Health Organisation. *Coronavirus disease (COVID-19) advice for the public*, 2020 (accessed May 18, 2020).

[7] ACAPS. *"COVID19 Government Measures Dataset"*, 2020 (accessed May 18, 2020).

[8] Roy M Anderson, Hans Heesterbeek, Don Klinkenberg, and T Déirdre Hollingsworth. How will country-based mitigation measures influence the course of the covid-19 epidemic? *The Lancet*, 395(10228):931–934, 2020.

[9] Wikipedia contributors. Basic reproduction number — Wikipedia, the free encyclopedia, 2020. [Online; accessed 18-May-2020].

[10] Moritz UG Kraemer, Chia-Hung Yang, Bernardo Gutierrez, Chieh-Hsi Wu, Brennan Klein, David M Pigott, Louis du Plessis, Nuno R Faria, Ruoran Li, William P Hanage, et al. The effect of human mobility and control measures on the covid-19 epidemic in china. *Science*, 368(6490):493–497, 2020.

[11] Chad R Wells, Pratha Sah, Seyed M Moghadas, Abhishek Pandey, Affan Shoukat, Yaning Wang, Zheng Wang, Lauren A Meyers, Burton H Singer, and Alison P Galvani. Impact of international travel and border control measures on the global spread of the novel 2019 coronavirus outbreak. *Proceedings of the National Academy of Sciences*, 117(13):7504–7509, 2020.

[12] MapAction. *COVID-19 - Government measures*, 2020 (accessed May 18, 2020).

[13] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Michael Kinney, Ziyang Liu, William. Merrill, Paul Mooney, Dewey A. Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex D. Wade, Kuansan Wang, Christopher Wilhelm, Boya Xie, Douglas M. Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. Cord-19: The covid-19 open research dataset. *ArXiv*, abs/2004.10706, 2020.

[14] Won Mo Jang, Sanghyun Cho, Deok Hyun Jang, Un-Na Kim, Hyemin Jung, Jin Yong Lee, and Sang Jun Eun. Preventive behavioral responses to the 2015 middle east respiratory syndrome coronavirus outbreak in korea. *International journal of environmental research and public health*.

[15] Rolf Hilgenfeld and Malik Peiris. From sars to mers: 10 years of research on highly pathogenic human coronaviruses. *Antiviral research*, 100(1):286–295, 2013.

[16] Benjamin J Cowling, Kwok-Hung Chan, Vicky J Fang, Calvin KY Cheng, Rita OP Fung, Winnie Wai, Joey Sin, Wing Hong Seto, Raymond Yung, Daniel WS Chu, et al. Facemasks and hand hygiene to

prevent influenza transmission in households: a cluster randomized trial. *Annals of internal medicine*, 151(7):437–446, 2009.

[17] Geoffrey J Gorse, Mary M Donovan, Gira B Patel, Sumitra Balasubramanian, and Rodney H Lusk. Coronavirus and other respiratory illnesses comparing older with young adults. *The American journal of medicine*, 128(11):1251–e11, 2015.

[18] Ronald W Schafer. What is a savitzky-golay filter?[lecture notes]. *IEEE Signal processing magazine*, 28(4):111–117, 2011.

[19] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[20] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017.

[21] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[22] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 1(10):e2, 2016.

Learning Non-Metric Visual Similarity for Image Retrieval : A Reappraisal

Lukman Olagoke
pennsylvester2@gmail.com

ABSTRACT

An efficient similarity measure is an essential core of any similar image retrieval model since it is responsible for capturing the (dis)similarity of the perceptual stimuli from the acquired vectors of visual descriptors. However, it is not often clear how the commonly used standard metric measure on this feature vector space corresponds to the observed differences in the perceptual space. The reviewed paper argues that a non-metric visual similarity based on neural network performs better than standard metric distance in measuring visual similarity of images. The aim of this review is to study the impact of non-metric visual similarity and standard metric in similar image retrieval.

CCS CONCEPTS

• **Information systems** → *Similarity measures.*

KEYWORDS

deep neural networks, similarity measure, content based Image retrieval, visual similarity, metric learning, Convolutional Neural Network.

1 INTRODUCTION

The paper under review [5] focuses on content based image retrieval approach called search by example, which requires learning of features and similarity metrics necessary to distinguish between objects within the same category - i.e., *fine-grained object similarity* . Another approach would be *search by category*, where two images are considered similar if they belong to the same category -i.e.,*category level similarity* [11, 18].

Early approaches to fine-grained object retrieval requires the use of hand-crafted features as discussed in [4, 14]. These features are subsequently used to learn similarity [2, 3]. In this review, deep neural network especially Convolutional Neural Network (CNN) and its variants have been used for feature extraction as discussed in [7, 8, 19]. Subsequently, image similarity is learnt using these features.

Paper Review , August 11, Moscow.
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 ARCHITECTURE SUMMARY

The non-metric visual similarity network proposed by the authors was trained on an existing CNN visual feature extractor called regional maximum activation of convolution (R-MAC) described in [19]. R-MAC was itself built on the pre-trained CNN model described in [16]. R-MAC effectively discards the fully connected layer of the pre-trained model and uses the Maximum Activation of Convolution technique for feature extraction. Summarily, the authors fed these visual descriptors into a similarity network called *SimNet*. The SimNet consisted of a set of fully connected layers with dimensions as shown in Table 1.

Table 1: Composition of SimNet

Layers	Size	Comments
Input layer	$1 \times K \times 2$	
First layer	$1 \times K \times 2 \times Ch$	Ch is number of channels
Hidden layers	$1 \times Ch \times 2 \times Ch$	
Output layer	$1 \times Ch \times 2 \times 1$	This is followed by ReLu

Training on Architecture

The R-MAC and SimNet networks were trained end-to-end using easy and difficult image pairs respectively. Four different configurations with different numbers of hidden layers and channels were tested. The training involved feeding two images into separate R-MAC architectures. The resultant features are then concatenated and fed into the SimNet for similarity computation. The training required that image pairs were labelled as similar pair (or not) and that an annotated similarity score is assigned to the pairs.

3 STRONG AND WEAK POINTS

Strong Points

The strong points of the paper are highlighted below:

- The authors were able to motivate the need for a non-metric similarity.
- The adaptability of the model - The SimNet could be adapted for use with other state-of-the art architectures.
- The results from the paper extended the state-of-the-art using well known data sets.

- The model training used both easy and hard data sample pairs in order to achieve good performance.
- The model was evaluated on both off-shelf and fine-tuned state-of-the-art models for comprehensive comparison.

In addition, the authors proposed a new loss function for training similarity :

$$\mathcal{L} = |s_{i,j} - l_{i,j}(sim(x_i, x_j) + \Delta) + (1 - l_{i,j})(sim(x_i, x_j) - \Delta)| \quad (1)$$

where $s_{i,j}$ is the similarity score. $sim(x_i, x_j)$ is the cosine similarity between the output feature vectors from R-MAC-SimNet architecture. Δ is the margin parameter. If I_i and I_j are 2 images, then:

$$l_{i,j} = \begin{cases} 1, & \text{if } I_i \text{ and } I_j \text{ are similar} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Weak Points

Training Evaluation.

- There is an issue with using OASIS [3] in the evaluation of results. OASIS is not based on CNN architecture and CNN architectures are known to have good representation power [6]. How do we decide if the improvement in performance is due to the R-MAC representational power or the added SimNet or both? Evaluation of the OASIS could have been done using OASIS versus OASIS-SimNet architecture. It is then possible to attribute the improvement in performance to the SimNet.
- OASIS is applicable to online large scale retrieval. The authors did not mention or show that the end-to-end architecture could be adapted for large scale retrieval. As noted in [19], the aim of R-Mac was not large scale retrieval but to present the representational power of the activation layers of CNN. How could R-MAC-SimNet be adapted to large scale retrieval and then consequently compared to an architecture such as OASIS for fair comparison?
- It would have been interesting to incorporate SimNet into deeprank model discussed in [21] by replacing the euclidean distance of the deeprank model with the SimNet to test for any improvement in performance. Deeprank uses CNN architecture for feature representation and it outperforms OASIS. If deeprank-SimNet architecture performs better than the deep rank architecture: then one could attribute the improvement in performance to SimNet. The same performance test could also be applied to the end-to-end architecture of [9].

Loss Computation and distance measure.

- The architecture uses cosine similarity. It is known that the cosine similarity equals the euclidean distance when the features are l_2 normalized. Note that for l_2 normalized vectors $\mathbf{x}$ and $\mathbf{y}$, then

$$\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$$

It follows that :

$$\begin{aligned} \|\mathbf{x} - \mathbf{y}\|_2^2 &= (\mathbf{x} - \mathbf{y})^T(\mathbf{x} - \mathbf{y}) \\ &= \mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{y} + \mathbf{y}^T\mathbf{y} \\ &= 2 - 2\mathbf{x}^T\mathbf{y} \\ &= 2(1 - \cos \angle(\mathbf{x}, \mathbf{y})) \end{aligned} \quad (3)$$

Thus we are back to the case of the euclidean distance which is a metric as defined by the properties mentioned in [5] itself.
- The loss function in equation 3 is defined as an absolute value. It is well known that it is not differentiable everywhere (specifically at zero). This is very easy to observe in the case where $\Delta = 0$ and images are similar ($l_{i,j} = 1$). In such case, we have that:

$$\mathcal{L} = |s_{i,j} - (sim(x_i, x_j)|$$

It was not stated how this would be tackled or was tackled. An alternative would be to use max-margin loss or propose certain heuristics at the zero point or use sub-gradient method [12].

Metric similarity and perceptual representation.

- First, as mentioned in [23] perceptual space is the space "out there" that we can often feel with our body and is characterised by three-dimensionality. Conceptual space is formulated in terms of abstract continuum [23], defined by mathematics and physics. A psychological space is space-as-experienced and can not be readily measured using any of the sense data that are used to define perceptual or conceptual space.
- Secondly, as discussed in [13] object similarity is intimately connected with the idea of geometric representation of stimuli in perceptual space. However, [20] opined that the closeness of stimuli and (object) similarity in a geometric representation can not be effectively captured by metrics that are based on segmental addition (eg euclidean distance metric).
- Thirdly , as discussed in [13], similarities based on Shepard law of universal similarity [15] are not affected by this criticism in [20]. Hence, it could be argued that given the transformations induced on the image space by the neural network, the metric distance between the psychological space and conceptual space representations decreases exponentially as discussed

in [13, 15]. The sketch of the argument is shown below:

d_p = distance induced on the psychological l_p space

The Shepard law says that the distance between the psychological lp norm and the measured similarity is exponentially decreasing:$K(x, y) = \exp(-d_p(x, y)^q)$

$$\begin{aligned} \bar{d_p} &= \|\phi(x) - \phi(y)\|^2 \\ &= \phi(x)^T\phi(x) - 2\phi(x)^T\phi(y) + \phi(y)^T\phi(y) \\ &= 2 - 2K(x, y) \\ &= 2 - 2\exp(-d_p(x, y)^q) \end{aligned} \tag{4}$$

In the above, q is an hyperparameter, x, y are feature vectors, and ϕ represent transformation on this feature space by neural net (say the final hidden layer). In essence we expect the difference in dissimilarity between the real measurement and geometrical distance to decrease exponentially.

- Lastly, as hypothesized in [25], perceptual similarity is not a special function but a consequence of visual representation tuned to be predictive about important structures in the world. Representations that are effective at semantic prediction tasks are also representations in which euclidean distance is highly predictive of perceptual similarity judgments.

4 FOLLOW UP STUDY TO THE PAPER

General consideration

- The paper introduces the similarity network. However, it will be worth while to focus on end-to-end architecture that seamlessly incorporates visual descriptor architecture and prediction in one pipeline. The impact of the similarity network could be tested by evaluating against standard metric distances.
- Models such as [9, 21, 25] have provided a general framework from which one could get an intuition. One would simply use a supervised learning method that given any 2 image feature vectors, predicts the similarity score. In fact, [25] has a prototype similarity model. The truth value could be generated using the *structural similarity index* as discussed in [22] or the perceptual or difference Hash as discussed in [24]. The challenge would be doing the supervision on the CNN architecture.

Proposed Models

First Model. A new model, shown in figure 1, is proposed. The intuition for this architecture comes from [5, 8, 25]. The architecture in fig1 is similar to [8]. The region proposal network from [8] is retained. However, the important differences are discussed below:

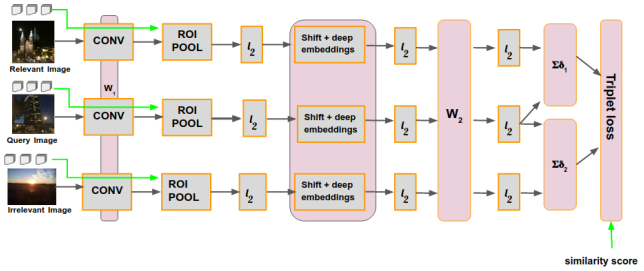


Figure 1: Proposed architecture. End-to-End similarity computation and margin maximization. $\mathbf{W}_1, \mathbf{W}_2$ are shared weights. $\Sigma\delta_1$ are the difference computation

- The CNN architecture extracts feature stack which are then normalized in the channel dimension.
- The activations are scaled channel-wise with the vector $\mathbf{W}_2$ which is shared across the triplet architecture.
- Given a feature output of L layers, if we designate $y_q^l, y_i^l \in \mathcal{R}^{H_l, W_l, C_l}$ for layer l,then the distance becomes the average spatial summed across each channel given as:

$$\Sigma\delta_1 = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|y_q^l - y_i^l\|_2^2$$

- For any pair of images or regions x_1, x_2 we evaluate their structural similarity score [22] and difference hash [24]. Then the similarity score becomes:

$$\text{similarity\_score}(x_1, x_2) = \frac{\text{structural similarity index}(x_1, x_2)}{\text{difference Hash}(x_1, x_2)}$$

- The loss function is computed as below:

$$\mathcal{L}(I_q, I_r, I_i) = \max(0, \text{similarity\_score} + \Sigma\delta_1 - \Sigma\delta_2)$$

q, i, r signify query,relevant and irrelevant images. The sub-gradients computation follows as defined in [1].

In general a Siamese architecture (2 input network) could also have been an excellent proposal. In this case one needs to train on easy and difficult sample pairs in order to enhance performance.

To evaluate the model, the pipeline in figure 2 is proposed. The following salient points about this architecture:

- A Siamese architecture is assumed.
- Given a query image, to generate similar image, select any image at random from the data base.
- Compute the similarity score. If the similarity score is above a defined threshold, we refer to this as a positive sample. Re-rank the images in the data base such that images with close difference hash values to positive samples are the once that will be tested for comparison with query image. Continue to evaluate, each time selecting and re-ranking till only best samples are at

the top. Select best top-k ($k \in \mathcal{N}$) from the collected best samples.

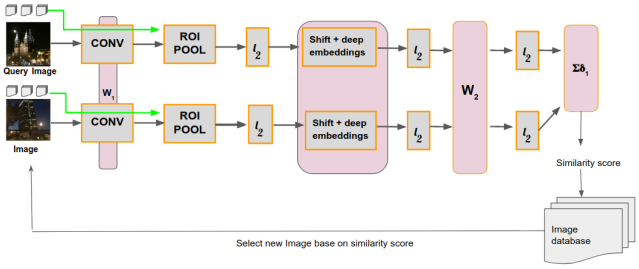


Figure 2: Proposed architecture. End-to-End similarity model evaluation.

Second Model. The second model proposed is based on [21]. A multilayer perceptron (MLP) is added to the architecture as shown below. The aim of the MLP layer is to take a linear combination of the feature vectors and compute a score. This score is compared to the pre-computed similarity score (as given in first model). The difference in similarity score is incorporated in the computation of max-margin loss.

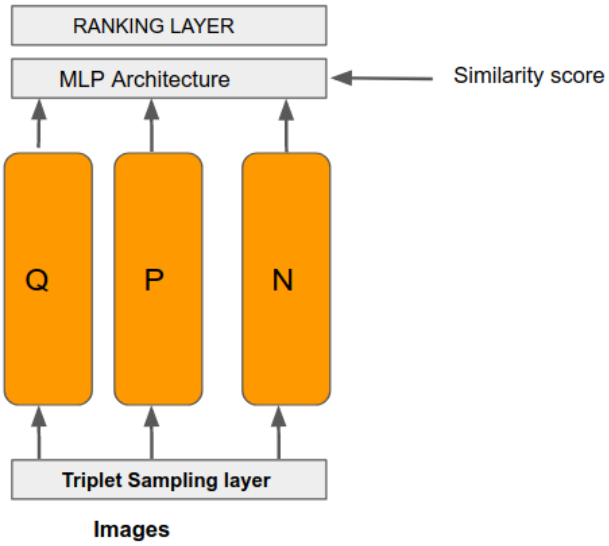


Figure 3: Proposed architecture based on deep rank model. An MLP layer has been added.

5 CONCLUSION

The authors have introduced a new idea for the robust computation of similarities given any pair of visual descriptors. This idea could be incorporated in state-of-the-art architecture in image retrieval. Overall, the idea presented by the authors is a crucial one that still needs to be revisited and studied.

REFERENCES

[1] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. 2018. Automatic differentiation in machine learning: a survey. *Journal of machine learning research* 18, 153 (2018).

[2] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. 2010. Learning mid-level features for recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Citeseer, 2559–2566.

[3] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. 2010. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research* 11, Mar (2010), 1109–1135.

[4] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection.

[5] Noa Garcia and George Vogiatzis. 2017. Learning Non-Metric Visual Similarity for Image Retrieval. *CoRR* abs/1709.01353 (2017). arXiv:1709.01353 http://arxiv.org/abs/1709.01353

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

[7] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. 2017. End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision* 124, 2 (2017), 237–254.

[8] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. 2017. End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision* 124, 2 (2017), 237–254.

[9] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. 2017. End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision* 124, 2 (2017), 237–254.

[10] Gregory Griffin, Alex Holub, and Pietro Perona. 2007. Caltech-256 object category dataset. (2007).

[11] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. IEEE, 1735–1742.

[12] Michael Held, Philip Wolfe, and Harlan P Crowder. 1974. Validation of subgradient optimization. *Mathematical programming* 6, 1 (1974), 62–88.

[13] Frank Jäkel, Bernhard Schölkopf, and Felix A Wichmann. 2008. Similarity, kernels, and the triangle inequality. *Journal of Mathematical Psychology* 52, 5 (2008), 297–303.

[14] David G Lowe et al. 1999. Object recognition from local scale-invariant features.. In *iccv*, Vol. 99. 1150–1157.

[15] Roger N Shepard. 1957. Stimulus and response generalization: A stochastic model relating generalization to distance in psychological space. *Psychometrika* 22, 4 (1957), 325–345.

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

[17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[18] Graham W Taylor, Ian Spiro, Christoph Bregler, and Rob Fergus. 2011. Learning invariance through imitation. In *CVPR 2011*. IEEE, 2729–2736.

[19] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. 2015. Particular object retrieval with integral max-pooling of CNN activations. *arXiv preprint arXiv:1511.05879* (2015).

[20] Amos Tversky and Itamar Gati. 1982. Similarity, separability, and the triangle inequality. *Psychological review* 89, 2 (1982), 123.
[21] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1386–1393.
[22] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.
[23] John Welwood. 1977. On psychological space. *The Journal of Transpersonal Psychology* 2 (1977), 97–118.
[24] Christoph Zauner. 2010. Implementation and benchmarking of perceptual image hash functions. (2010).
[25] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 586–595.

A EXPERIMENTAL DETAILS

In order to proof the concepts discussed in this paper, a very brief training and evaluation model was done on the caltech dataset 101 [10] which contains 250 categories of objects. From each category 9 objects were selected.

Training

Each object is transformed to a feature output vector of size 1x2048 using a pre-trained ResNet model [17]. This is followed by randomly selecting 2 pairs of feature vectors and computing the square of their differences. The square of their differences is used as the input into an MLP similarity network.

The target value was computed using the structural similarity index divided by the difference hash as described in the first proposed model. In summary, the MLP takes in square of the difference between any pair of image feature vector and output the target value.

Evaluation

This simple model was evaluated against an exactly similar model but which uses the euclidean distance as a measure of similarity instead of the MLP similarity network. To evaluate the model, one selects a query image randomly and let the network output the top-10 similar objects. It should be noted that feature vector is first extracted from the query image. Afterwards, the query feature vector is compared against the database of feature vectors using euclidean distance and the MLP network as similarity measure respectively.

It should be stated that the computation was carried out on a personal computer without GPU capability. This means the MLP architecture might not be as robust as expected but it still looks very promising.

The query image is shown below:

Figure 4: Query image for model evaluation

The retrieved image using euclidean distance as metric:

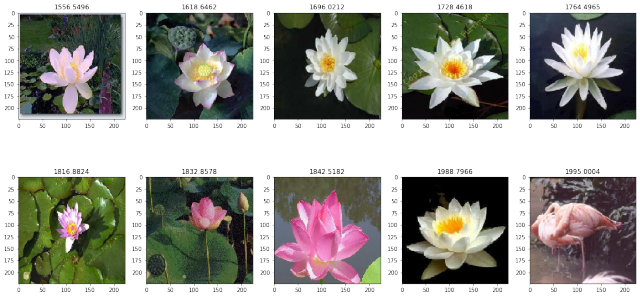


Figure 5: Evaluation of Model using euclidean distance

The retrieved image using the MLP similarity network:

Figure 6: Evaluation of Model using MLP similarity network

As usual with any deep neural network model, the parameters still needs to be tuned to achieve a better result. In addition more samples and training epochs will be needed which the time constraint will not permit me. However, it is a step in the right direction. The code for the model is on my Github reporistory. [1]

[1] https://github.com/adderbyte/content_based_image_retrieval

Learning to Fly (L2F)
Reinforcement Learning for Autonomous Piloting

L. O. Olagoke[1]

Learning to Fly Project
March 2019

AIRBUS

Table of Contents

Motivation.

- Good news: " Long-haul commercial flights could see reduced cockpit crews from 2023, shortly after cargo planes," analysts at UBS Group AG wrote in an extensive July report. They estimated a profit potential of $15 billion for flying with a single pilot and $35 billion if airplanes were to fly themselves "[1]
- This is just one step in the right direction

[1] Justin Bachman. *Autonomous Flights Are One Step Closer to Reality*. https://www.bloomberg.com/news/articles/2018-10-10/will-you-fly-when-your-pilot-is-a-robot. [Online; accessed 18-March-2019]. Oct. 2018.

Simulation Based Approach

- A simulation environment is needed XPlane.
- Algorithmic Approach: Reinforcement Learning.

XPlane-Agent Connection Diagram

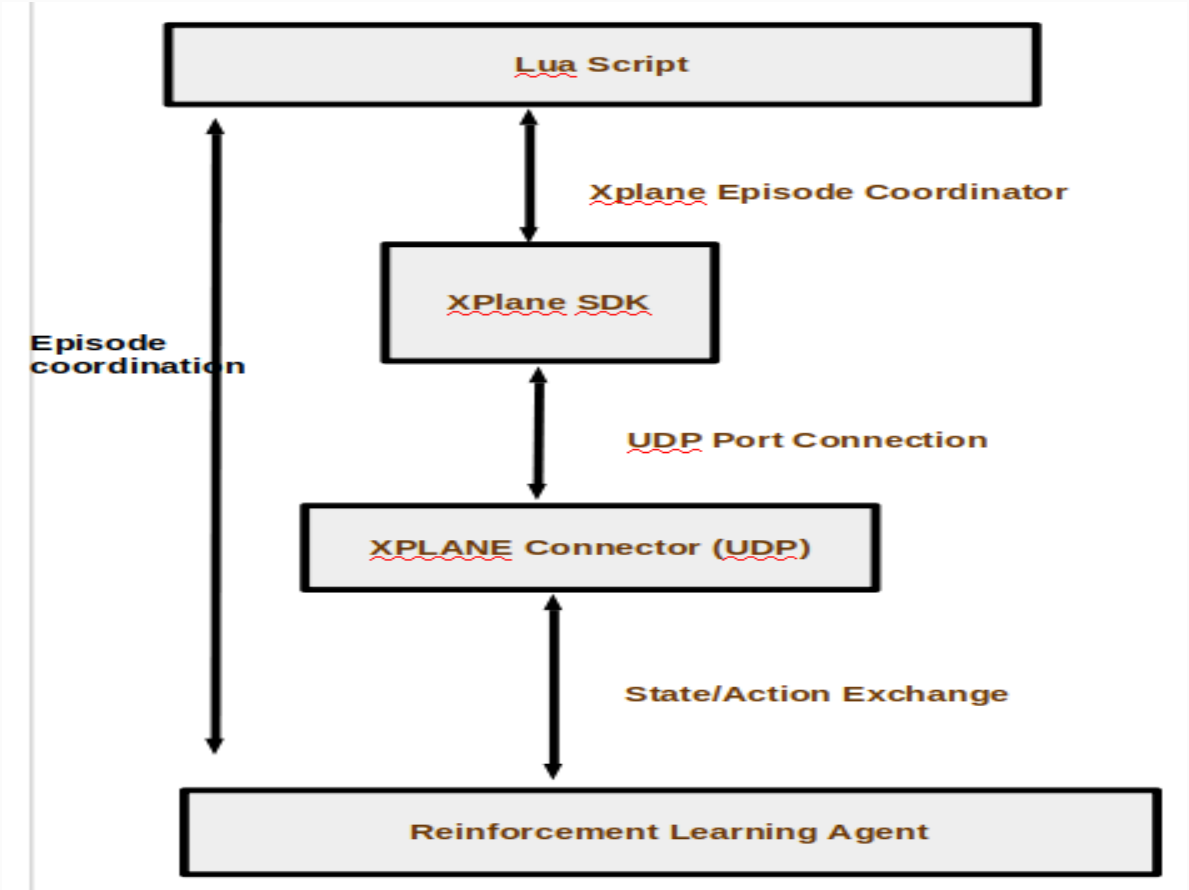


Figure: XPlane Connnection Diagram

L2F Solution Space

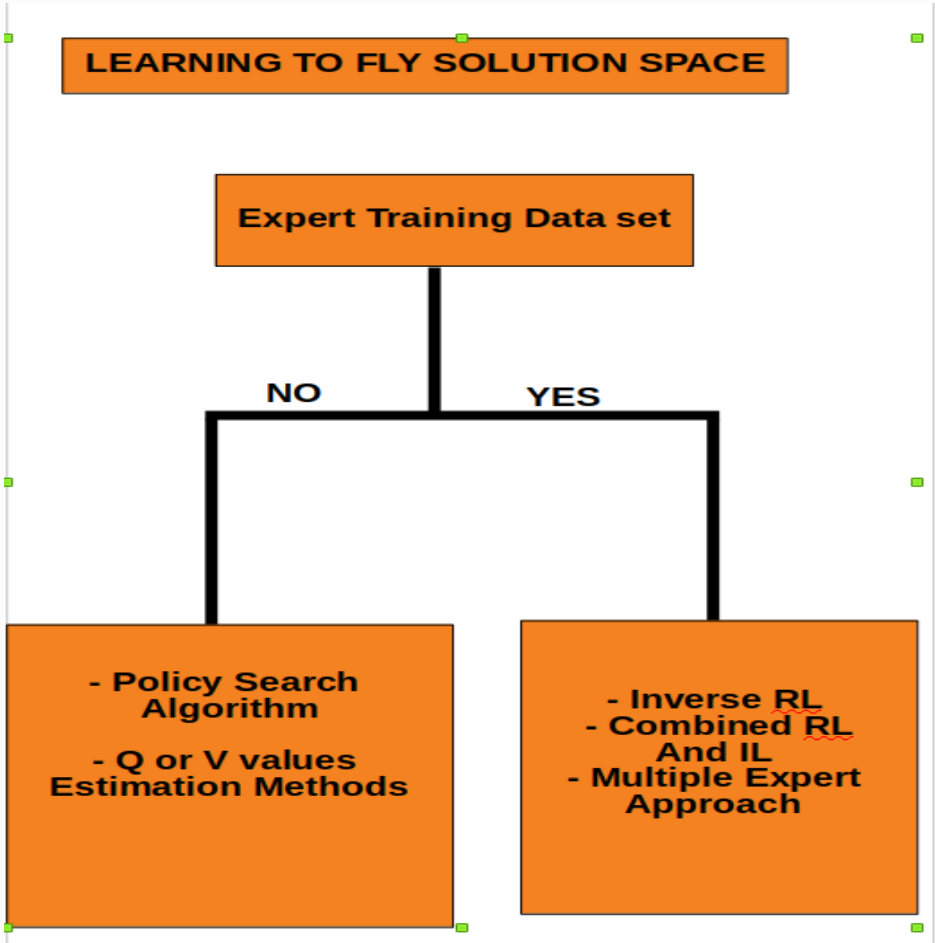


Figure: Possible Flight Simulation solution Space

No Pre training Data was used in this project

Reinforcement Learning Simplified

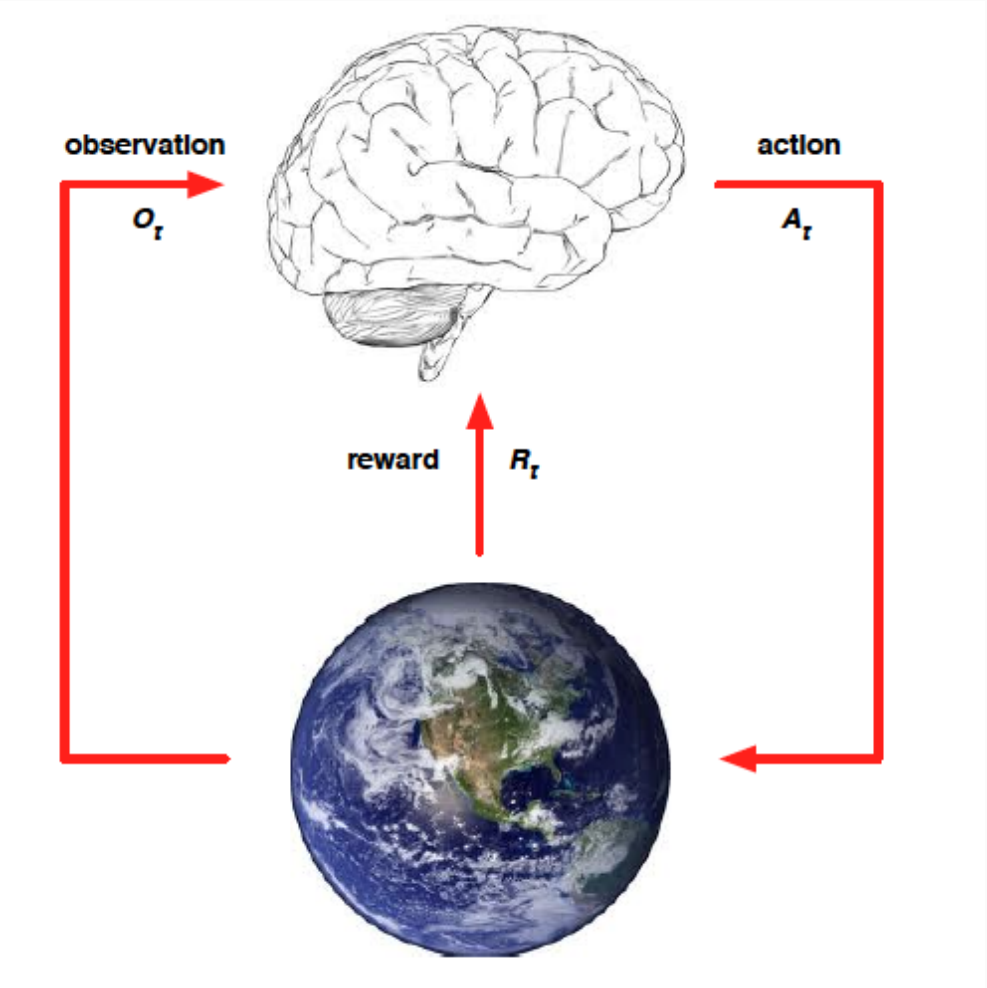


Figure: Agent and Environment

- At each step t the agent:
  - Executes action A_t
  - Receives Observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at environment step.

[2] David Silver. *Reinforcement Learning*.

Reward

A Reward is a scalar feedback signal. It indicates how well the agent is doing at step t. The agent goal is to maximise cumulative reward. Reinforcement Learning is based on reward hypothesis[3].

Definition - Reward Hypothesis

All goals can be described by the maximization of expected cumulative reward. ??

[3] David Silver. *Reinforcement Learning*. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html. [Online; accessed 28-September-2018]. 2015.

RL Agents Components

- Policy.[4]

Definition

Agent behaviour function. Roughly, we say it is a mapping from perceived states to actions to be taken when in those states. It is sufficient enough to determine agent's behaviour.

- deterministic policy: action taken in state s under deterministic policy π.

$$a = \pi(s)$$

- Stochastic policy: probability of taking action a in state s under stochastic policy π.

$$\pi(a|s) = P(A_t = a|S_t = s)$$

[4] 3.

RL Agents Components

- Policy.[4]
- Value Function

Definition

Value function of a state is the total reward an agent is expected to accumulate over the future starting from that state . It specifies how good a action or function is. E.g

$$V_\pi = E_\pi \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... | \ S_t = s \right)$$

[4] 3.

Return

Definition

The Return $\mathcal{G}_t$ is the total discounted reward from time-step t

- $\mathcal{G}_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

- γ is the discount rate $\gamma \in [0,1]$
- value of receiving reward R after k+1 step is $\gamma^k R$
- This value immediate reward above delayed reward.
  - γ close to 0 leads to "myopic" evaluation.
  - γ close to 1 leads to "far-sighted" evaluation.[5]

[5] David Silver. *Reinforcement Learning*. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html. [Online; accessed 28-September-2018]. 2015.

Value Function

state-value function

A state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following a policy π

$$v_\pi(s) = \mathbb{E}_\pi \left[\mathcal{G}_t | S_t = s \right]$$

The Bellman equation for $v_\pi(s)$ is:

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s \right]$$

Action-Value Function

The action-value function $q_\pi(s,a)$ is the expected return starting from state s, taking action a, and then following policy π.

$$q_\pi(s, a) = \mathbb{E}_\pi \left[\mathcal{G}_t | S_t = s, A_t = a \right]$$

$$q_\pi(s, a) = \mathbb{E}_\pi [R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

Value Estimation Methods

How do we make decisions on next action?

Solve the MDP using Value Estimation Methods.

- Q Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$$

- SARSA Algorithm

Problems[6]

- Hard to scale to high dimension
- Approximations might destroy policy
- For many MDPs Q values can be complicated and difficult to approximate while policies might be simple

[6] Pieter Abbeel , Gerhard Neumann. *Policy Search: Methods and Applications*. https://icml.cc/2015/tutorials/PolicySearch.pdf. [Online; accessed 28-September-2018]. 2015.

Policy Gradient Search[7]

- Policy Search Methods: Methods that learn a parameterized policy that can select actions without consulting a value function.
- $\pi(a|s,\theta) = P(A_t = a|S_t, \theta_t = \theta)$
- Explanation: Probability that an action is taken at time t, given that the environment is in state s at time t with parameter θ .
- $\theta \in R^d$ is the parameter vector , while $J(\theta)$ is the performance measure

Then we seek to maximize $J(\theta)$, so their updates approximate gradient ascent in J:

- $\theta_{t+1} = \theta_t + \alpha \Delta J(\theta_t)$

[7] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction(Online Draft)*. MIT press, 2017.

OpenAI GYM

- toolkit for developing and comparing reinforcement learning algorithms.
- No assumption about structure of agent
- Goal: Make the RL environment compatible of the gym interface:
  - step
  - reset
  - render
  - close

GYM XPLANE

- Adapting XPlane for Gym Interface
- 2 scenarios tested:
  - Keeping a Predefined Heading
  - Keeping heading and Altitude.
- gymXplane-v2 (Standalone for RL on Xplane)

State Action Space

Xplane Data and command Ref files : Contains all action Parameters

- Action Space (Controls)
  - Longitudinal Stick (Roll control)
  - Latitudinal Stick (Pitch control)
  - Rudder Pedals (Rudder control)
  - Throttle

AIRBUS

State Action Space

Xplane Data Ref : Contains all state Parameters

- State Space Parameters
  - Angular Momentum (p_dot, q_dot,r_dot)
  - Speed
  - Horizontal Stability
  - Vertical Stability
  - Pitch
  - Roll
  - Change in Altitude/Heading
  - This could change depending on scenario or observation

AIRBUS

Reward Function

Very tricky to get right

- Canberra distance:

$$d(targetState, presentState) = \Sigma_i \frac{|targetState_i - presentState_i|}{|targetState_i| - |presentState_i|}$$

- Apply Gaussian Kernel on the canberra distance : $e^{-\frac{d(targetState, presentState)^2}{\sigma^2}}$
- σ is an hyperparamater [0,1]: Such that the closer the present values to the target the bigger value chosen.
- Canberra distance: is chosen to motivate the agent to learn

Relevant Distance Measure

Other distance measure that could be used:

- Squared Euclidean
- Cosine similarity
- Braycurtis
- Standardized Euclidean

Simulation Initialization Parameters

Parameters	Values
Altitude	1200 Ft MSL
Speed	90 Knots True
Pitch	0-3
Heading	164 degrees
Episode Length	90-93

Libraries: Stable Baseline, Random Agent, Sarsa Actor-Critic
Other Baselines (Tensorforce,RLLIB) are possible but they might be slow or h

AIRBUS

Case For Pretraining

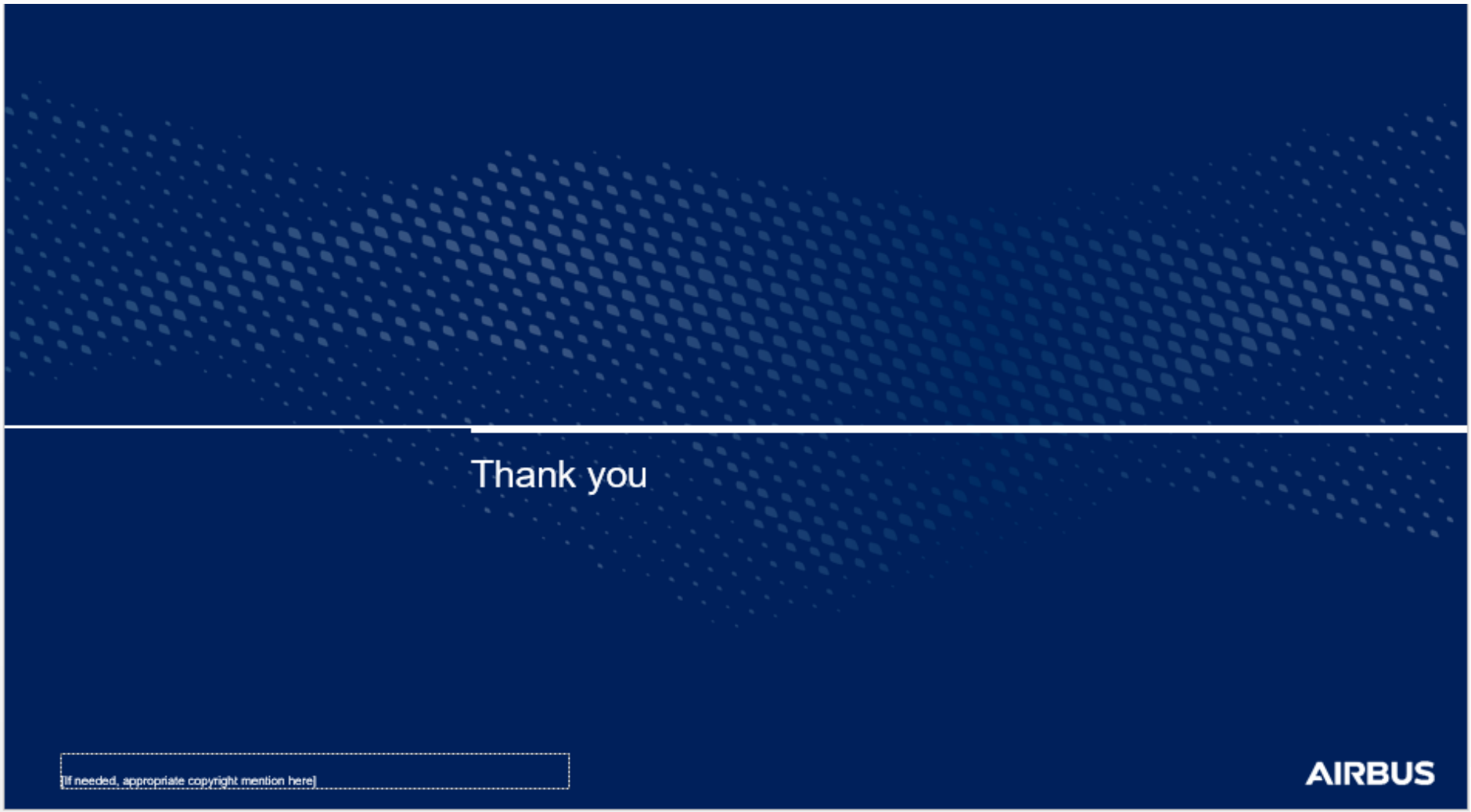
- Use collected trajectory dataset to approximate reward function
- Pre-train agent before actual training
- Reward specification is important. Agent training depends on it

Conclusions

- X Plane can not use vectorised environment (Slows down training)
- Deep learning models (Time to Converge)
- Traditional RL models are being tested.(shows more stable flight)
- Hyperparameter tuning still necessary
- Appriate state space variables (More variables, more trianing time)
- master-slave approach (multiple agents)

AIRBUS

Questions / Discussions



AIRBUS

References I


Justin Bachman. *Autonomous Flights Are One Step Closer to Reality*. https://www.bloomberg.com/news/articles/2018-10-10/will-you-fly-when-your-pilot-is-a-robot. [Online; accessed 18-March-2019]. Oct. 2018.


Pieter Abbeel , Gerhard Neumann. *Policy Search: Methods and Applications*. https://icml.cc/2015/tutorials/PolicySearch.pdf. [Online; accessed 28-September-2018]. 2015.


David Silver. *Reinforcement Learning*. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html. [Online; accessed 28-September-2018]. 2015.


Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction(Online Draft)*. MIT press, 2017.

L2F Status Update: Architecture and Algorithm

L. O. Olagoke[1]

Learning to Fly Project
Nov. 2018

AIRBUS

Table of Contents

AIRBUS

RL for Complex Task

How can RL solve complex task like autonomous piloting ?

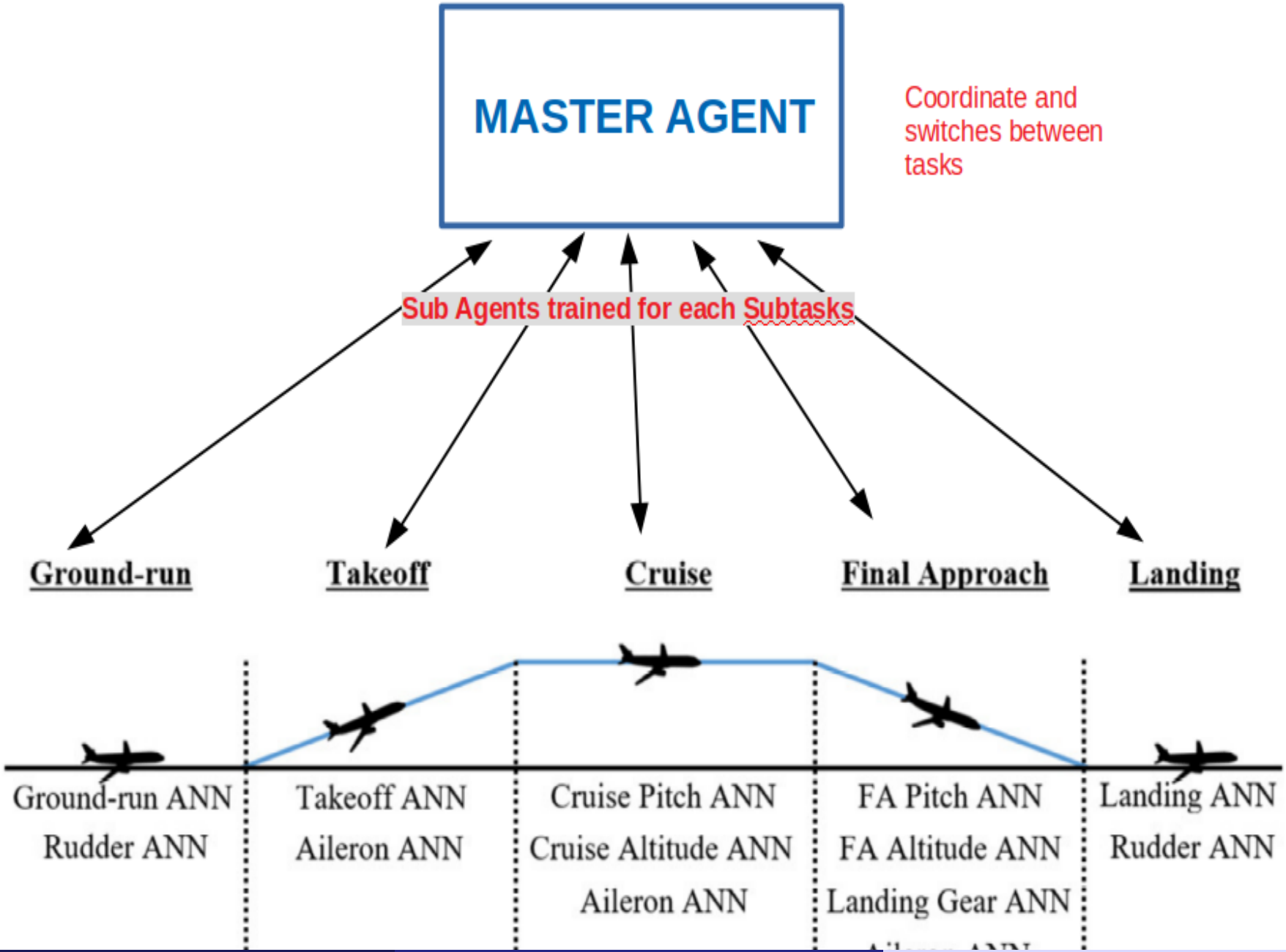
- Humans solve complicated challenges by breaking them up into small, manageable components .
- Grilling pancakes consists of a series of high-level actions, such as measuring flour, whisking eggs, transferring the mixture to the pan, turning the gas cooker on, and so on
- Idea: Break complex tasks into **Sub-tasks**.

AIRBUS

Hierarchical RL

- Break problems into sub-task .
- Extend traditional reinforcement learning to more complex tasks
- How many sub-tasks?
- What are the sub-tasks ?
- We can start simple.

Architecture



Sub-Agents Training

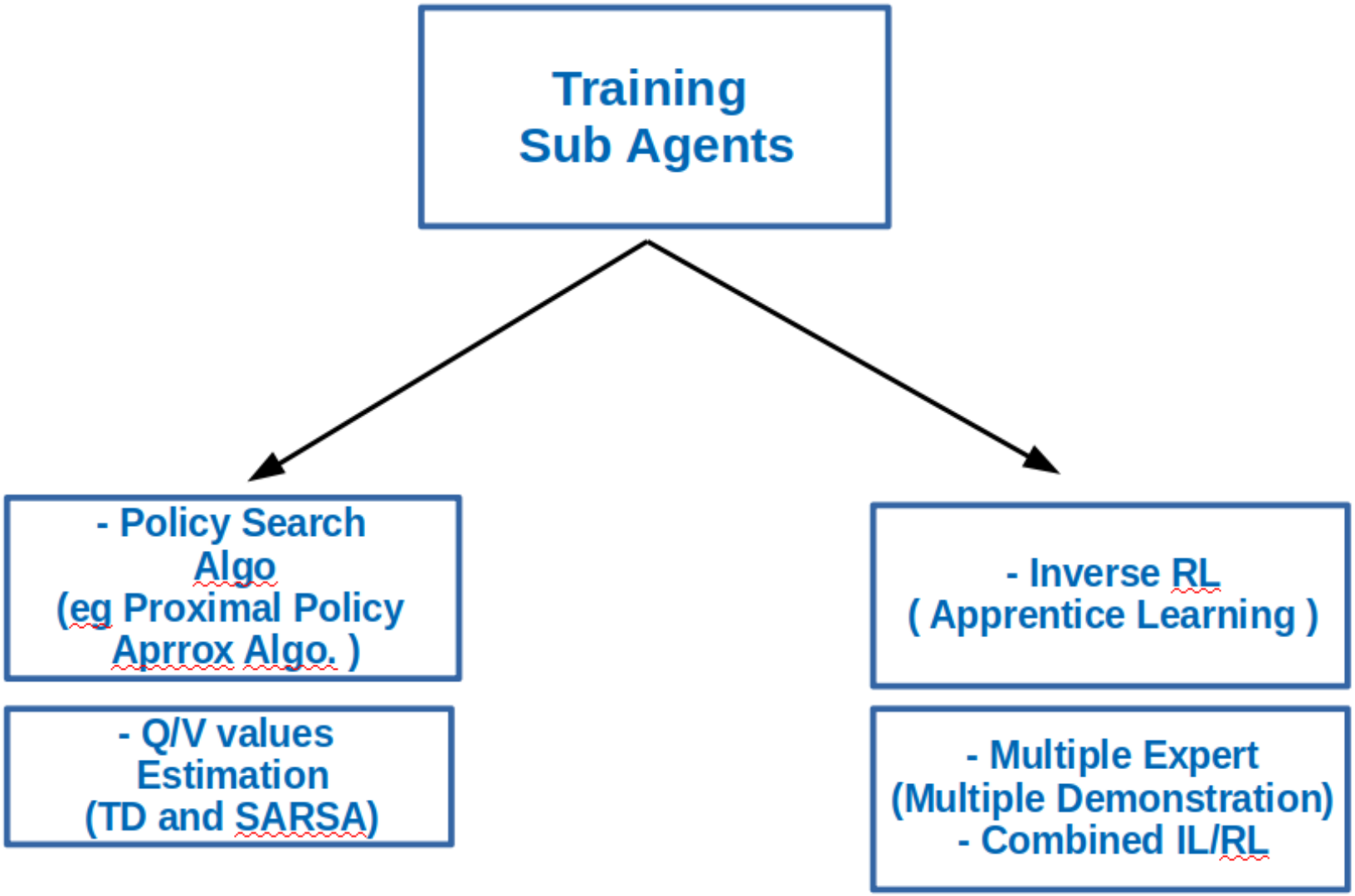
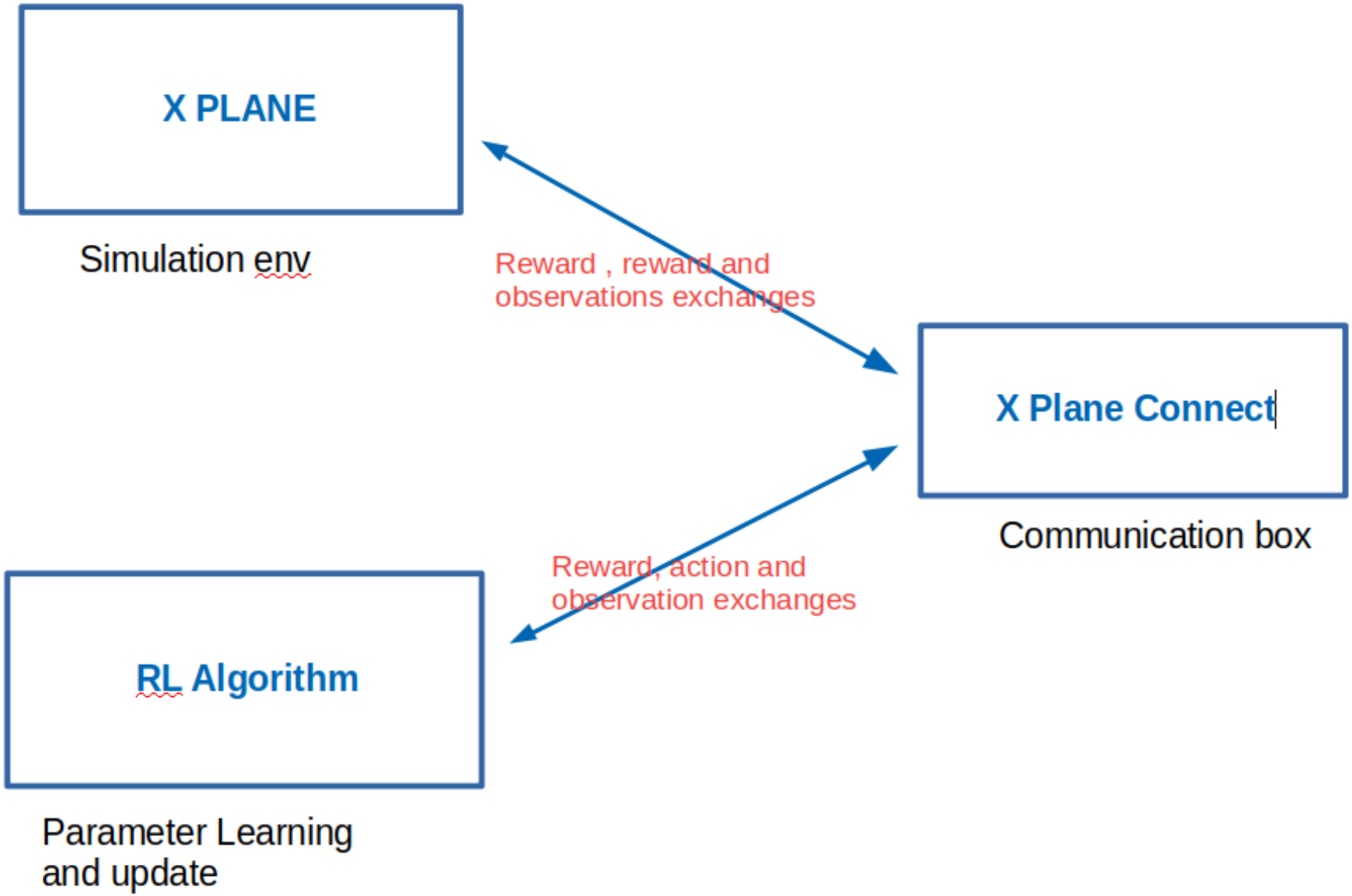


Figure: Architecture

Two Training Approaches

APPROACH 1: FAST TRAINING PIPELINE



Fast RL Training Pipeline

Other Considerations

- X Plane not built with RL in Mind. (API needed)
  1. X Plane SDK
     https://developer.x-plane.com/sdk/
  2. FlywithLua (Lua programming - fast prototyping)
     https://forums.x-plane.org/index.php?/files/file/38445-flywithlua-ng-next-generation-edition-for-x-plane-11
  3. FlyWithLua needs Lua Programming language.(easy)
     https://www.lua.org/start.html
  4. Python Interface (Not tested, no good support)
     http://www.xpluginsdk.org/python_interface.htm
  5. Other API's as needed.

AIRBUS

Fast RL Training Pipeline

Other Considerations Continued

1. X Plane Connect from NASA
   https://github.com/nasa/XPlaneConnect
2. RL Algorithm : Tensorforce[8] , a Tensorflow library for applied RL. (no need to reinvent the wheel)
   https://github.com/reinforceio/tensorforce
3. Why Tensorforce ?: *test sub-task on available RL benchmarks, reuse modules for new algorithms*

[8] Michael Schaarschmidt, Alexander Kuhnle, and Kai Fricke. *TensorForce: A TensorFlow library for applied reinforcement learning*. Web page. 2017. URL: https://github.com/reinforceio/tensorforce.

Where are the Parameters?

The Parameters Reference:

1. X Plane Parameters (Data Reference)
   http://www.xsquawkbox.net/xpsdk/docs/DataRefs.html
2. Command Reference
   http://siminnovations.com/xplane/command/index.php
3. Wiki reference
   http://www.xsquawkbox.net/xpsdk/mediawiki/Main_Page
4. X Plane SDK Sample code complete Download.
   http://www.xsquawkbox.net/xpsdk/mediawiki/Category:Sample_Code

AIRBUS

X Plane Flight School

Flight School for training Agent: (General Aviation)

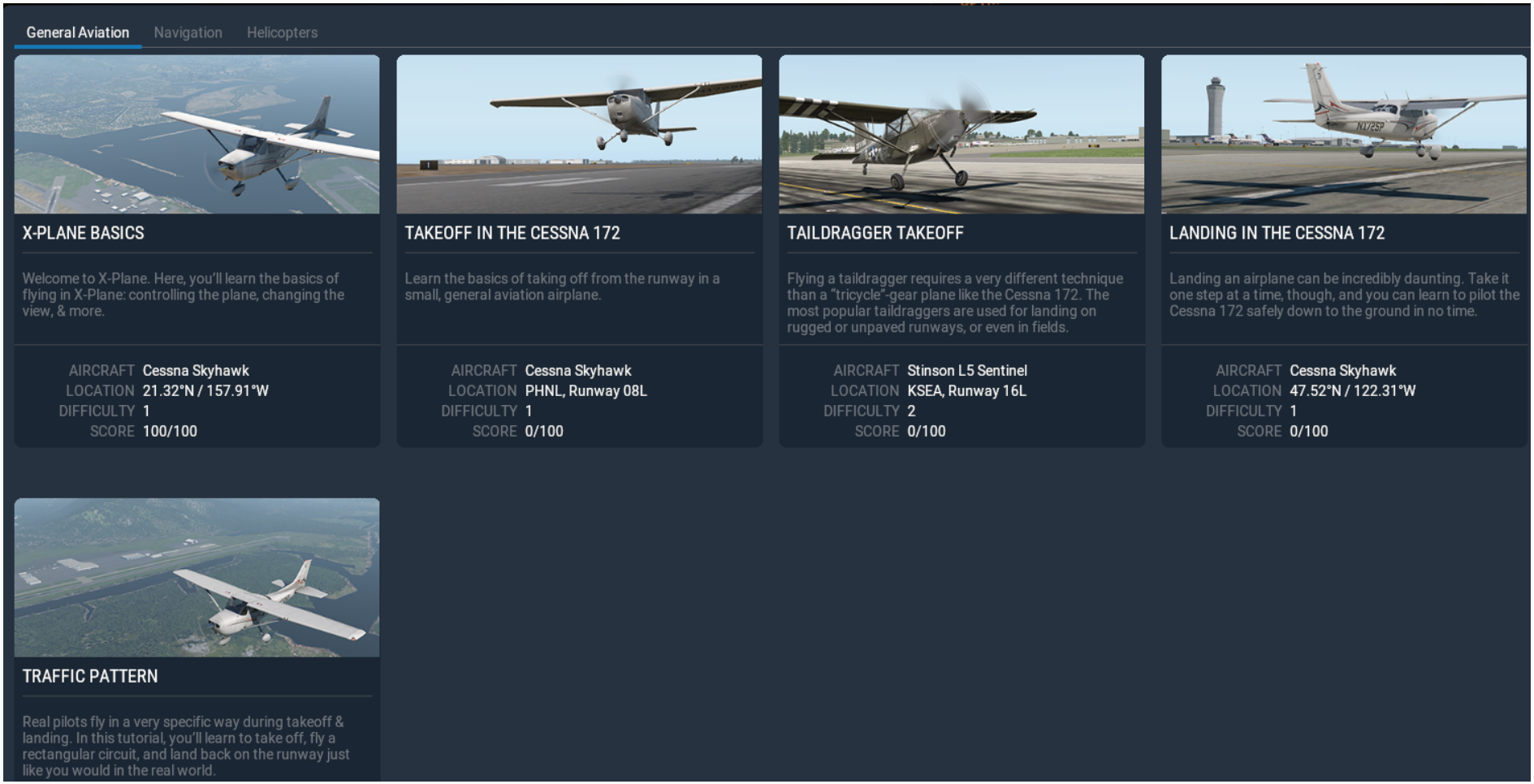


Figure: Training Set up

X Plane Flight School

Flight School for training Agent: (Navigation)

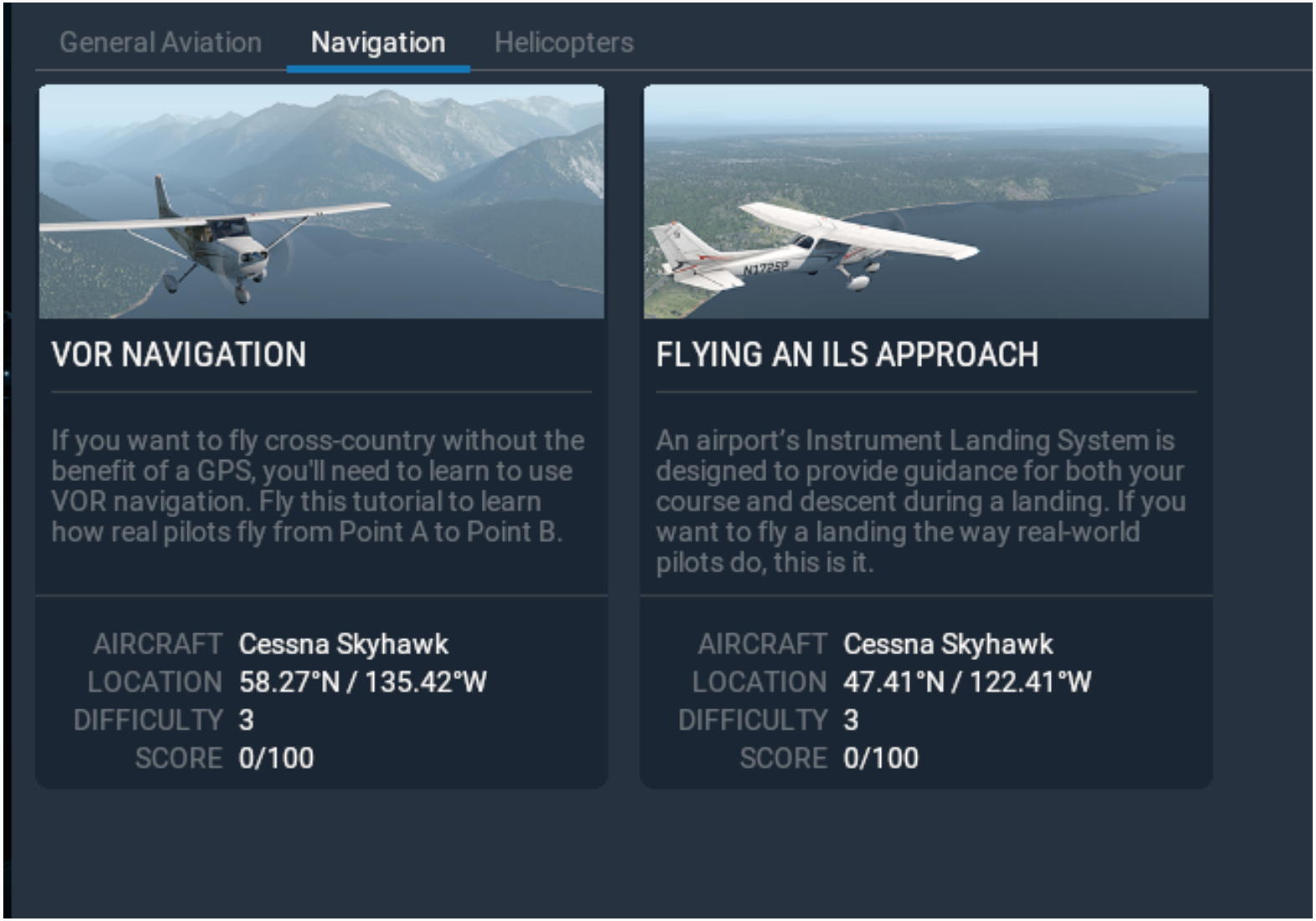


Figure: Training Set-up

Approach 2: Robust Training Pipeline.

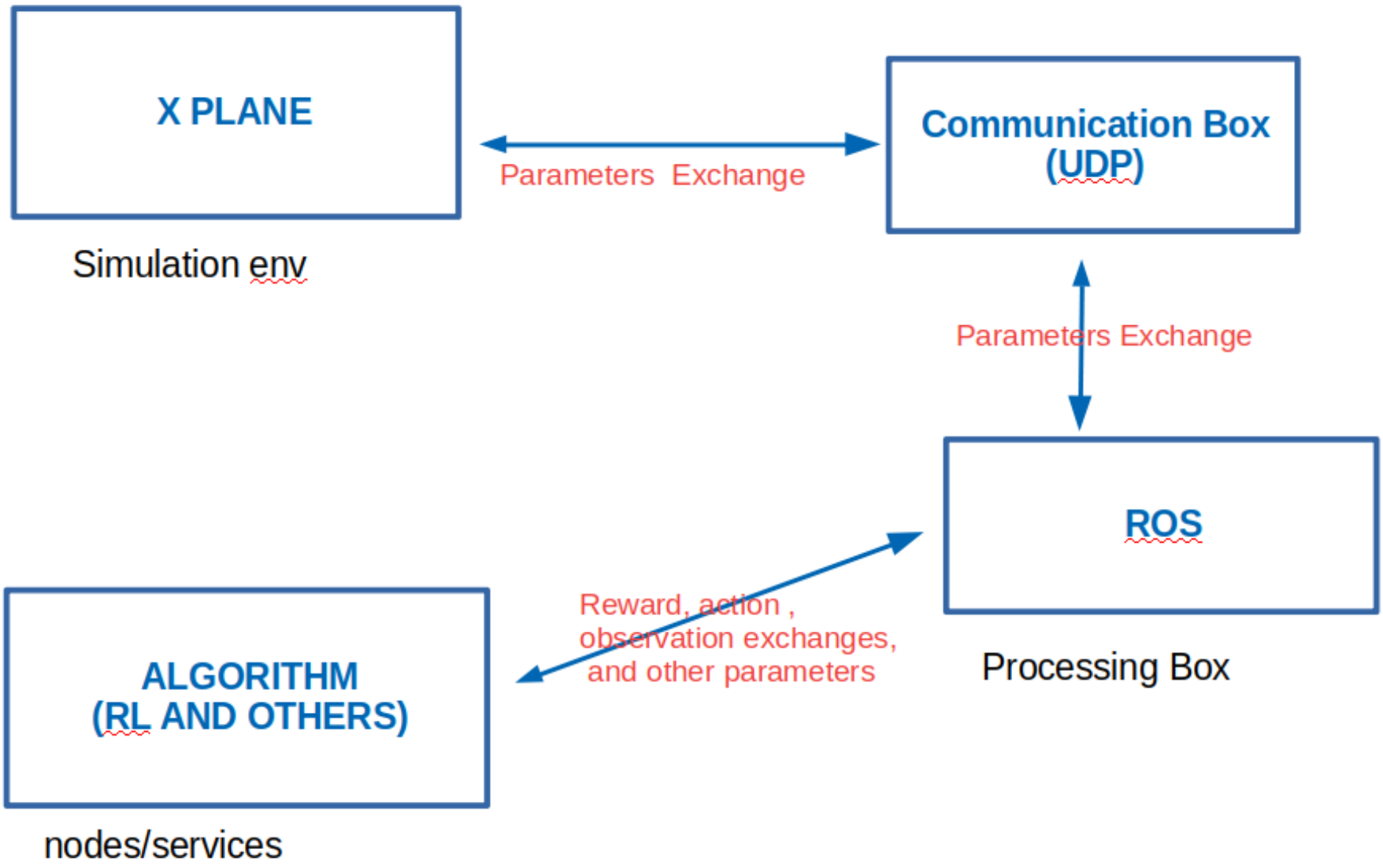


Figure: Robust Set up

Robust Pipeline

Other Considerations:

- Vision modules (to identify objects - taxing) (YOLO algorithm) https://github.com/pjreddie/darknet/wiki/YOLO:-Real-Time-Object-Detection
- Identify the airport/track (Landing Phase)
- BlackMagic module for video stream quality enhancement
- Other modules/devices as needed

AIRBUS

Testing Flight And Other Features

Test and Emergencies module:

- Add Other Parameters (Data Ref) (Aircraft weight etc)
- Add Constraints (Amount of fuel Data Ref)
- Simulate failures at different phases (Aircraft settings)
- Testing : test model on "Normal Flight"
- Add Master Agent (Hybrid learning)
- Other considerations as necessary

AIRBUS

Where is the Reward ?

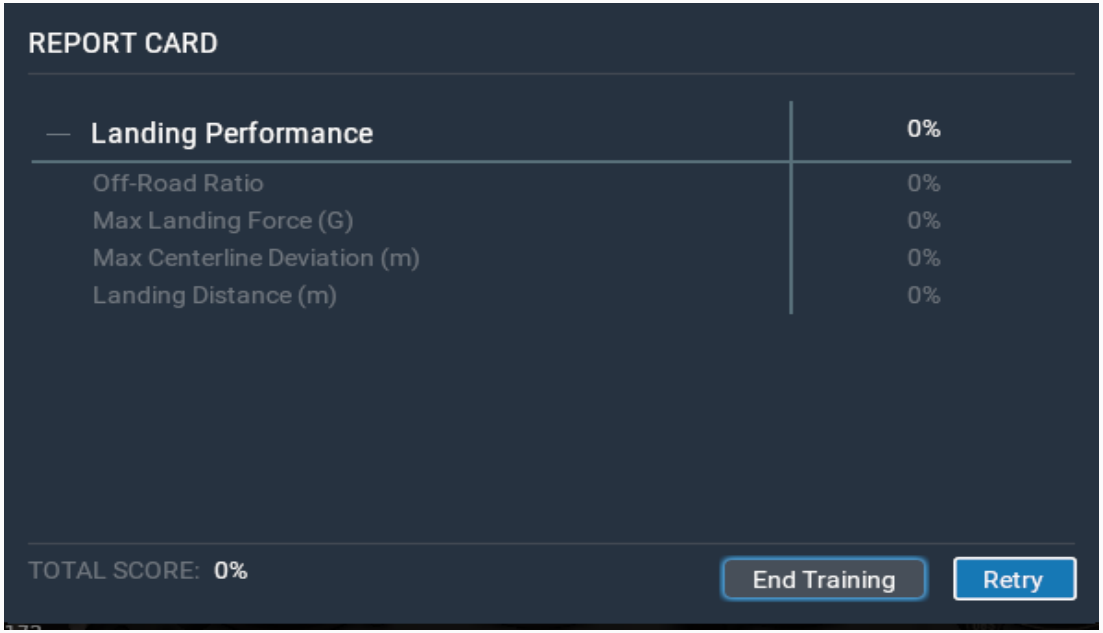


Figure: Reward: Can we use this ?

- Do we have a reward function or do we define one ?
- Detecting and penalizing crash ?

Conclusions

- Multiplayer System (let agent play against itself)
- Wiki Documentation (for project follow-up)
- Still a work in progress ...

References I

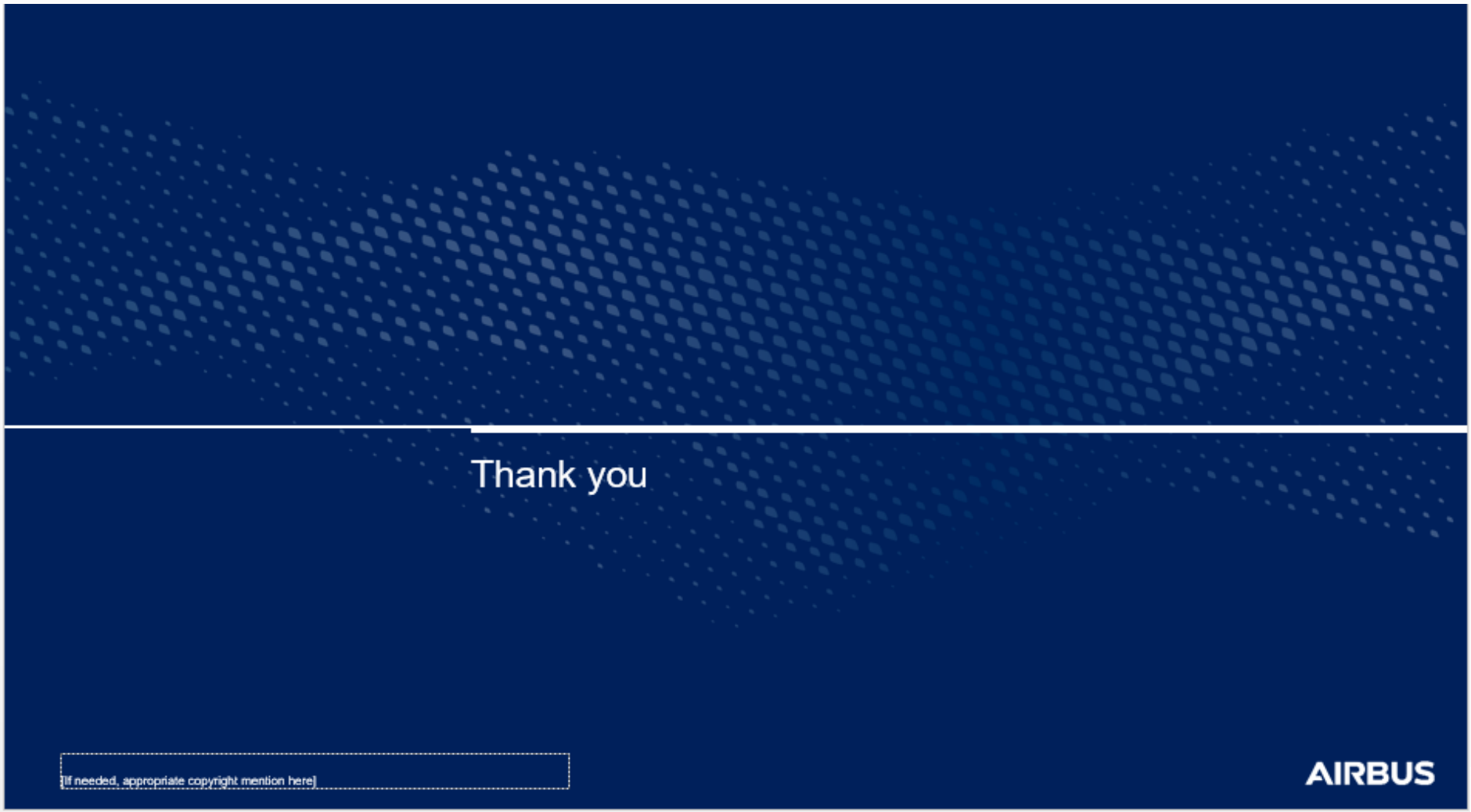
- Pieter Abbeel. *Reinforcement Learning*. http://ai.stanford.edu/~pabbeel/thesis/thesis.pdf. [Online; accessed 28-September-2018]. 2015.
- Pieter Abbeel and Andrew Y Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 1.
- Haitham Baomar and Peter J Bentley. "Autonomous Navigation and Landing of Airliners Using Artificial Neural Networks and Learning by Imitation". In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2017.
- Adam Coates, Pieter Abbeel, and Andrew Y Ng. "Learning for control from multiple demonstrations". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 144–151.

AIRBUS

References II

- Kevin Frans et al. "Meta learning shared hierarchies". In: *arXiv preprint arXiv:1710.09767* (2017).
- Ofir Nachum et al. "Data-Efficient Hierarchical Reinforcement Learning". In: *arXiv preprint arXiv:1805.08296* (2018).
- Michael Schaarschmidt, Alexander Kuhnle, and Kai Fricke. *TensorForce: A TensorFlow library for applied reinforcement learning*. Web page. 2017. URL: https://github.com/reinforceio/tensorforce.
- Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction(Online Draft)*. MIT press, 2017.

Questions / Discussions



AIRBUS

L2F state of the Art Reinforcement Learning for Piloting

L. O. Olagoke[1]

Learning to Fly Project
Sept. 2018

AIRBUS

Table of Contents

Autonomous Flight And Reinforcement Learning?

Can we train a system to perform autonomous flight?

- Sequentially programming the system base on different scenario will be ineffective .
- We need learning algorithm.
- Supervised, Unsupervised or Reinforcement learning (RL).
- RL is the best option (Mapping Situation to Action).

Agent and Environment

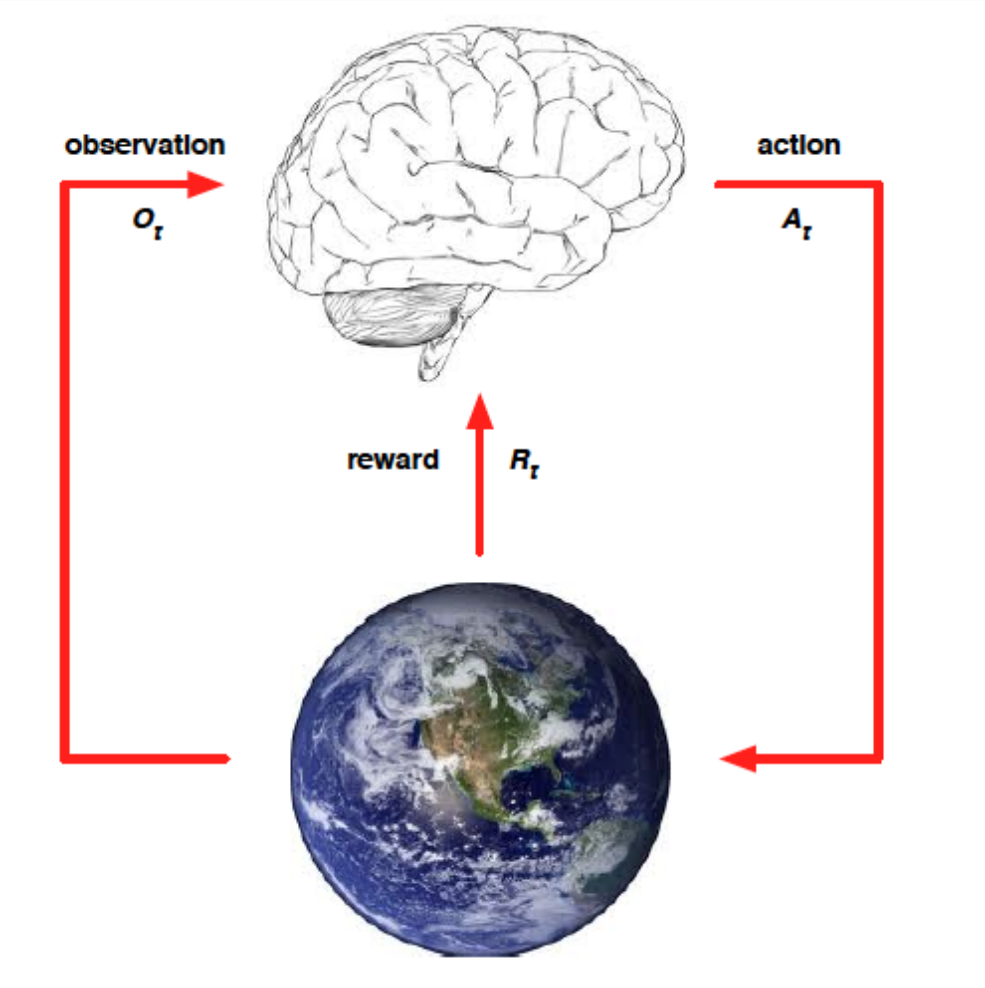


Figure: Agent and Environment

- At each step t the agent:
  - Executes action A_t
  - Receives Observation O_t
  - Receives scalar reward R_t
- The environment:
  - Receives action A_t
  - Emits observation O_{t+1}
  - Emits scalar reward R_{t+1}
- t increments at environment step.

[1] David Silver. *Reinforcement Learning*.

Reward

A Reward is a scalar feedback signal. It indicates how well the agent is doing at step t. The agent goal is to maximise cumulative reward. Reinforcement Learning is based on reward hypothesis[2].

Definition - Reward Hypothesis

All goals can be described by the maximization of expected cumulative reward. ??

[2] David Silver. *Reinforcement Learning*. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html. [Online; accessed 28-September-2018]. 2015.

Partially And Fully Observable Environments

- Full observability: agent directly observes environment state.
- Agent State = Environment State = Markov State
- This defines Markov Decision Process

- Partial observability: agent indirectly observes environment.
- Agent state $\neq$ Environment state
- This defines a Partially Observabe Markov Decision Process (POMDP)

In POMDP, the agent must construct its own state representation S_t^a. Using for E.g.[3]

- Complete History : $S_t^a = H_t$
- Beliefs of the environment : $S_t^a = (\mathbb{P}[S_t^e = s^1], ...\mathbb{P}[S_t^e = s^n])$
- Recurrent neural network. $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

[3] David Silver. *Reinforcement Learning*. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html. [Online; accessed 28-September-2018]. 2015.

RL Agents Components

- Policy.[4]

Definition

Agent behaviour function. Roughly, we say it is a mapping from perceived states to actions to be taken when in those states. It is sufficient enough to determine agent's behaviour.

- deterministic policy: action taken in state s under deterministic policy π.

$$a = \pi(s)$$

- Stochastic policy: probability of taking action a in state s under stochastic policy π.

$$\pi(a|s) = P(A_t = a|S_t = s)$$

[4] 12.

RL Agents Components

- Policy.[4]
- Value Function

Definition

Value function of a state is the total reward an agent is expected to accumulate over the future starting from that state . It specifies how good a action or function is. E.g

$$V_\pi = E_\pi \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... | \ S_t = s \right)$$

[4] 12.

RL Agents Components

- Policy.[4]
- Value Function
- Model

Definition

Model: is the agent representation of the environment. It is whatever an agent can use to predict how the environment will respond to its actions.

- $\mathcal{P}$ predicts the next state [5]:

$$\mathcal{P}^{a}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$$

- $\mathcal{R}$ predicts the next immediate reward.

$$\mathcal{R}^{a}_{s} = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

[4] 12.

[5] We assume that the Markov property is satisfied

Return

Definition

The Return $\mathcal{G}_t$ is the total discounted reward from time-step t

- $\mathcal{G}_t = R_t + \gamma R_{t+2} + ... = \sum_{k=1}^{\infty} \gamma^k R_{t+k+1}$

- γ is the discount rate $\gamma \in [0,1]$
- value of receiving reward R after k+1 step is $\gamma^k R$
- This value immediate reward above delayed reward.
  - γ close to 0 leads to "myopic" evaluation.
  - γ close to 1 leads to "far-sighted" evaluation.[6]

[6] David Silver. *Reinforcement Learning*. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html. [Online; accessed 28-September-2018]. 2015.

Bellman equation Simplified[7]

$$
\begin{aligned}
v(s) &= \mathbb{E}[G_t | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + ...) | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma (\mathcal{G}_{t+1}) | S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s]
\end{aligned}
$$

[7] David Silver. *Reinforcement Learning*. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html. [Online; accessed 28-September-2018]. 2015.

Categorizing RL Agents (1)

- Value Based
  - Value Function
  - No Policy (Implicit)
- Policy based
  - Policy
  - No Value Function
- Actor critic
  - Policy
  - Value Function
- Model Free
  - Policy and/or Value Function

Exploration Vs Exploitation

- Exploration: involves finding more information about the environment
- Exploitation: involves exploiting known information to maximise reward
- Balance between exploitation and exploration should be sought
- Prediction : given a policy, evaluate the future
- Control : optimise the future by finding best policy[8]

[8] David Silver. *Reinforcement Learning*. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html. [Online; accessed 28-September-2018]. 2015.

L2F Solution Space

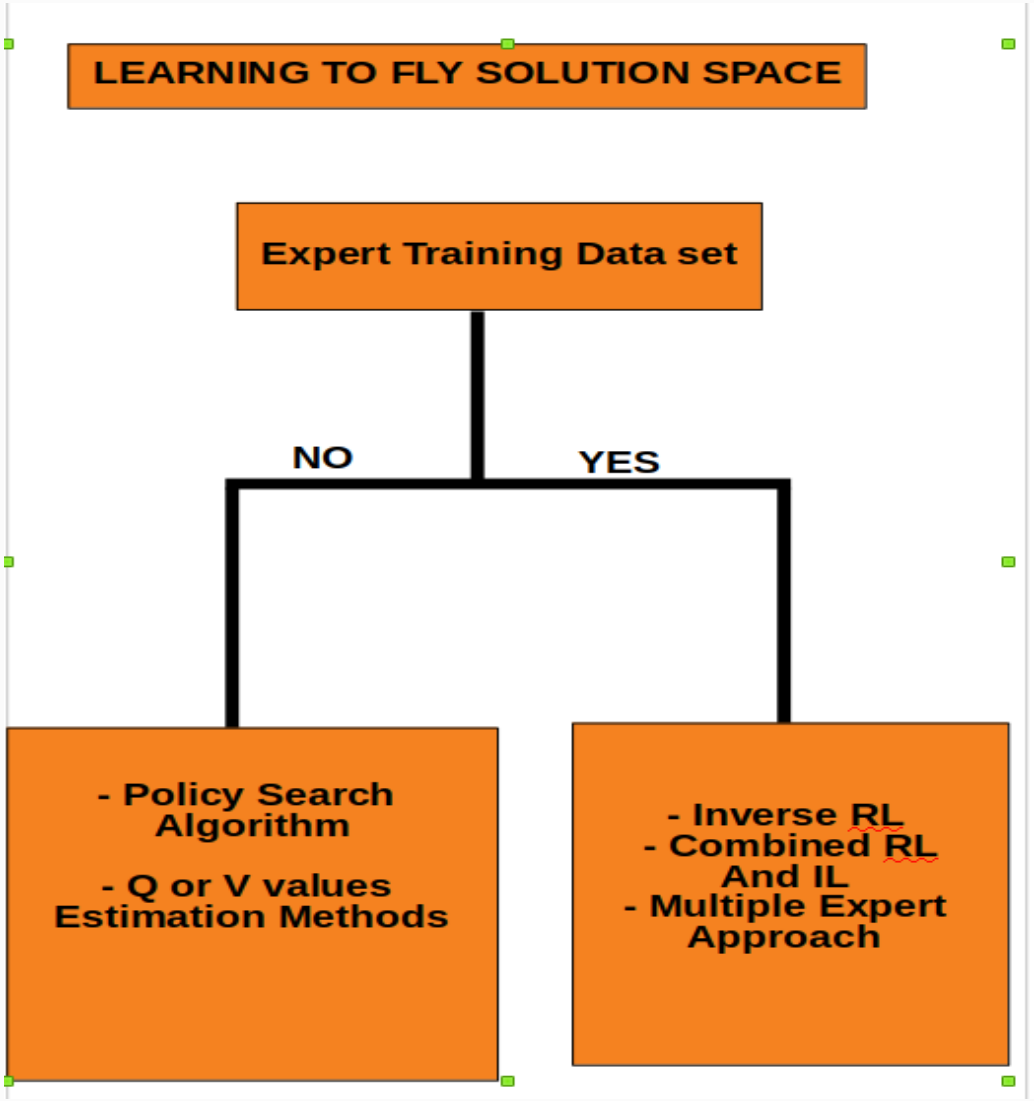


Figure: Possible Flight Simulation solution Space

Markov Decision Process

- How do we solve the task at hand ? (Learning to Fly)
- (An idea) Specify a reward function $\rightarrow$ Markov Decision Process

Definition

A Markov Reward process is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{A}, \mathcal{R}, \gamma, \mathcal{D} \rangle$ where

- $\mathcal{S}$ is a (finite) set of states.
- $\mathcal{P}$ is a state transition probability matrix: $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$
- $\mathcal{A}$ is the action space
- $\mathcal{R}$ is a reward function: $\mathcal{R}_s = \mathbb{E}[R_{t+1} | S_t = s]$
- γ is discount factor, $\gamma \in [0,1]$
- $\mathcal{D}$ is the initial state probabilities

9

[9] Pieter Abbeel and Andrew Y Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

Agent And its Environment: Policy search

- A diagram is worth a million !

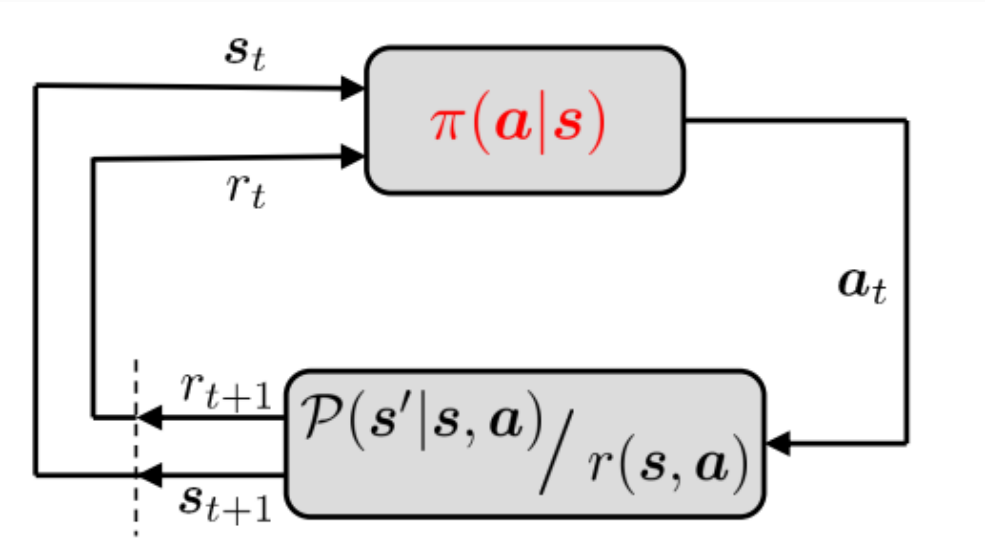


Figure: Agent and Environment

Figure taken from:[10]

[10] Pieter Abbeel , Gerhard Neumann. *Policy Search: Methods and Applications*. https://icml.cc/2015/tutorials/PolicySearch.pdf. [Online; accessed 28-September-2018]. 2015.

Value Function

state-value function

A state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following a policy π

$$v_\pi(s) = \mathbb{E}_\pi \left[\mathcal{G}_t | S_t = s \right]$$

The Bellman equation for $v_\pi(s)$ is:

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s \right]$$

Action-Value Function

The action-value function $q_\pi(s,a)$ is the expected return starting from state s, taking action a, and then following policy π.

$$q_\pi(s, a) = \mathbb{E}_\pi \left[\mathcal{G}_t | S_t = s, A_t = a \right]$$

$$q_\pi(s, a) = \mathbb{E}_\pi [R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

Value Estimation Methods

How do we make decisions on next action?

Solve the MDP using Value Estimation Methods.

- Q Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \arg\max_a Q(S_{t+1}, a) - Q(S_t, A_t))$$

- SARSA Algorithm

Problems[11]

- Hard to scale to high dimension
- Approximations might destroy policy
- For many MDPs Q values can be complicated and difficult to approximate while policies might be simple
- Number of states grows exponentially in number of state variables.

[11] Pieter Abbeel , Gerhard Neumann. *Policy Search: Methods and Applications*. https://icml.cc/2015/tutorials/PolicySearch.pdf. [Online; accessed 28-September-2018]. 2015.

Policy Gradient Search[12]

- Policy Search Methods: Methods that learn a parameterized policy that can select actions without consulting a value function.
- $\pi(a|s,\theta) = P(A_t = a|S_t, \theta_t = \theta)$
- Explanation: Probability that an action is taken at time t, given that the environment is in state s at time t with parameter θ .
- $\theta \in R^d$ is the parameter vector , while $J(\theta)$ is the performance measure

Then we seek to maximize $J(\theta)$, so their updates approximate gradient ascent in J:

- $\theta_{t+1} = \theta_t + \alpha \Delta J(\theta_t)$

[12] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction(Online Draft)*. MIT press, 2017.

Solving the Policy Gradient Problem

Many Approaches but same goal ...

- Policy Gradient (Monte Carlo)
- Neural Network and Monte Carlo Tree Search (AlphaGo Zero)[13]

[13] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

Inverse Learning

Two Challenges in applying RL for Simulation.

- Appropriate reward function
- Learning fairly accurate dynamics of model

Possible Solution

1. Use IL to learn the reward function and dynamics
2. Use Model Based Methods RL for control

Imitation Learning Illustrated

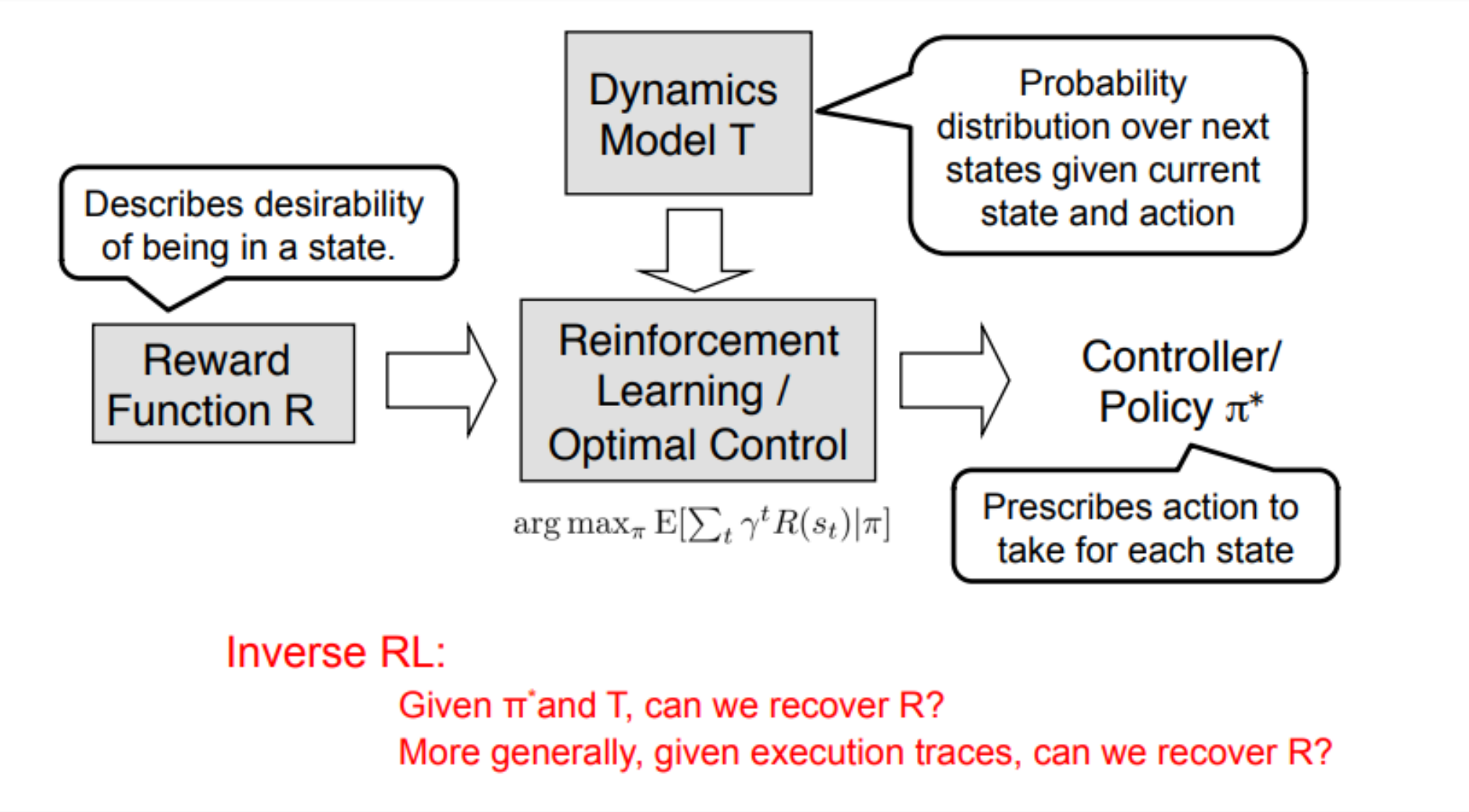


Figure taken from [14]

[14] Pieter Abbeel. *Inverse Reinforcement Learning*.

IL Reward Function Estimation: Why?

- Reward Function provide most succinct representation of task.
- Assume we have other parameters except the reward function. We also have in addition the Trace of the teacher's policy (s_0, a_0, s_1, a_1 ...)
- The aim is to recover R and use this to define a good policy.
- This approach has been more successful in application to autonomous flight system[15]

[15] Pieter Abbeel. *Inverse Reinforcement Learning*. https://people.eecs.berkeley.edu/~pabbeel/cs287-fa15/slides/lecture6-inverseRL.pdf. [Online; accessed 28-September-2018]. 2010.

Inverse Reinforcement Learning: Going deep[16]

- Given Some vector of features: $\phi : S \rightarrow [0,1]^k$ over the state space.
- $R^* = \omega^{*T}\phi(s)$ is the true reward function. $\omega^* \in \mathcal{R}^k$
- Compute the policy and feature expectation or expected discounted accumulated feature value .

- $E[V^\pi(s_0)] = \omega^t \mu(\pi)$
- $\mu(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \in \mathcal{R}^k$
- Given a set of m trajectories $\{s_0^i, s_1^i ...\}_{i=i}^m$ given by the expert

$$\mu_E = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{\infty}\gamma^t \phi(s_t^i)$$

- Algorithm finds a policy such that

$$\left\|\mu(\pi) - \mu_E\right\|_2 \leq \epsilon$$

[16] Pieter Abbeel. *Inverse Reinforcement Learning*. https://people.eecs.berkeley.edu/~pabbeel/cs287-fa15/slides/lecture6-inverseRL.pdf. [Online; accessed 28-September-2018]. 2010.

Learning Flight Dynamics with IL[17]

Given an initial pilot flight demonstration data, can we learn flight dynamics?

1. Have a pilot demonstrate what is to be learned and we record the flight trajectories of the demonstration
2. Use all state-action trajectories seen so far to learn a dynamics model for the system
3. Test the policy by running on a real system/simulation. If the performance is as good as the teacher, stop. Else, add the state-action trajectories from the (unsuccessful) test and go back to step 2.

[17] Pieter Abbeel. *Inverse Reinforcement Learning*. https://people.eecs.berkeley.edu/~pabbeel/cs287-fa15/slides/lecture6-inverseRL.pdf. [Online; accessed 28-September-2018]. 2010.

Pitfalls of IL[18]

1. Assumes expert is optimal
2. Assumes we can enumerate all policies
3. Reward function ambiguity (R=0)

What are possible solutions ?

- Learn from multiple expert demonstration
- Combining Reinforcement Learning & Imitation Learning

[18] Adam Coates, Pieter Abbeel, and Andrew Y Ng. "Learning for control from multiple demonstrations". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 144–151.

Learning from Multiple experts[19]

Goal: Learn Given multiple expert models that might be inaccurate (Noisy expert models)

- Let the estimated dynamics for estimated MDP be $\mathcal{T}^*$ while $\mathcal{T}$ is dynamics for True MDP.
- MDP $\mathcal{M}^*$ for $\mathcal{T}^*$ now includes a local policy improvement algorithm $\mathcal{A}$
- $\mathcal{A}$ iteratively makes local policy improvement upon current policy eg using policy gradients

[19] Adam Coates, Pieter Abbeel, and Andrew Y Ng. "Learning for control from multiple demonstrations". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 144–151.

Learning from Multiple experts: Algorithm[20]

1. Initialise model estimate $\mathcal{T}^0 = \mathcal{T}$
2. Find local optimum policy π_{θ_0} using local improvement algorithm on $\mathcal{M}^*$
3. Execute the policy in the real MDP and record the state-action trajectory
4. Construct new model by adding bias term to $\mathcal{T}^0$
5. Use the policy improvement algorithm to find local policy improvement direction (utility function)
6. Use line search to find next optimal policy $\theta^{i+1} = \theta^i + \alpha d^i$
7. If line search did not improve policy , return current policy and exit, otherwise go to step 3.

[20] Pieter Abbeel. *Reinforcement Learning*. http://ai.stanford.edu/~pabbeel/thesis/thesis.pdf. [Online; accessed 28-September-2018]. 2015.

Cost Shaping[21]

- Reward shaping is the practice of modifying a reward function to provide guidance or give hints to the learning agent.
- Formally, we assume we have a transformed MDP where $R' = R + F$ is the bounded shaping reward function.
- $F = \phi(s') - \phi(s)$ where $\phi(s)$ indicate value of a state

[21] Andrew Ng. *Shaping and Policy Search in Reinforcement Learning.*. http://rll.berkeley.edu/deeprlcoursesp17/docs/ng-thesis.pdf. [Online; accessed 28-September-2018]. 2013.

Finite Horizon And Reward Shaping[22]

Assume access to:

- Reward shaping function
- Imitation Learning as above (Oracle in form of expert)

1. Instead of searching for policies that optimize total cost over entire horizon, we search for policies that minimize cost over truncated horizon.
2. The oracle or expert model can help shorten the horizon.

[22] Wen Sun, J Andrew Bagnell, and Byron Boots. "Truncated Horizon Policy Search: Combining Reinforcement Learning & Imitation Learning". In: *arXiv preprint arXiv:1805.11240* (2018).

RL Example Use Case for Aeronautic Flight

1. Stanford Autonomous Helicopter Demonstration (Learning from multiple experts)
   `http://heli.stanford.edu/`
2. Hierarchical Reinforcement learning for flight Simulation.[23]
3. Improvement on Hierarchical learning[24][25]

[23] Malcolm Ryan and Mark Reid. "Learning to fly: An application of hierarchical reinforcement learning". In: *In Proceedings of the 17th International Conference on Machine Learning*. Citeseer. 2000.

[24] Kevin Frans et al. "Meta learning shared hierarchies". In: *arXiv preprint arXiv:1710.09767* (2017).

[25] Tuomas Haarnoja et al. "Latent Space Policies for Hierarchical Reinforcement Learning". In: *arXiv preprint arXiv:1804.02808* (2018).

Conclusions

- We can benefit from available training data set
- Policy Search algorithm have been used in similar setting and it works.
- we can compute the Q or Value Functions while doing policy search.
- RL and IL can be combined to make a robust model
- We can also learn from noisy expert data set.[26]

[26] David Silver et al. "Deterministic policy gradient algorithms". In: *ICML*. 2014.

References I

- Pieter Abbeel. *Inverse Reinforcement Learning*. https://people.eecs.berkeley.edu/~pabbeel/cs287-fa15/slides/lecture9-inverseRL.pdf. [Online; accessed 28-September-2018]. 2010.
- Pieter Abbeel. *Reinforcement Learning*. http://ai.stanford.edu/~pabbeel/thesis/thesis.pdf. [Online; accessed 28-September-2018]. 2015.
- Pieter Abbeel and Andrew Y Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 1.
- Adam Coates, Pieter Abbeel, and Andrew Y Ng. "Learning for control from multiple demonstrations". In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 144–151.

References II

- Kevin Frans et al. "Meta learning shared hierarchies". In: *arXiv preprint arXiv:1710.09767* (2017).
- Tuomas Haarnoja et al. "Latent Space Policies for Hierarchical Reinforcement Learning". In: *arXiv preprint arXiv:1804.02808* (2018).
- Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).
- Pieter Abbeel , Gerhard Neumann. *Policy Search: Methods and Applications*. https://icml.cc/2015/tutorials/PolicySearch.pdf. [Online; accessed 28-September-2018]. 2015.
- Andrew Ng. *Shaping and Policy Search in Reinforcement Learning*.. http://rll.berkeley.edu/deeprlcoursesp17/docs/ng-thesis.pdf. [Online; accessed 28-September-2018]. 2013.

AIRBUS

References III


Andrew Y Ng and Michael Jordan. "PEGASUS: A policy search method for large MDPs and POMDPs". In: *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc. 2000, pp. 406–415.


Malcolm Ryan and Mark Reid. "Learning to fly: An application of hierarchical reinforcement learning". In: *In Proceedings of the 17th International Conference on Machine Learning*. Citeseer. 2000.


David Silver. *Reinforcement Learning*. `http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html`. [Online; accessed 28-September-2018]. 2015.

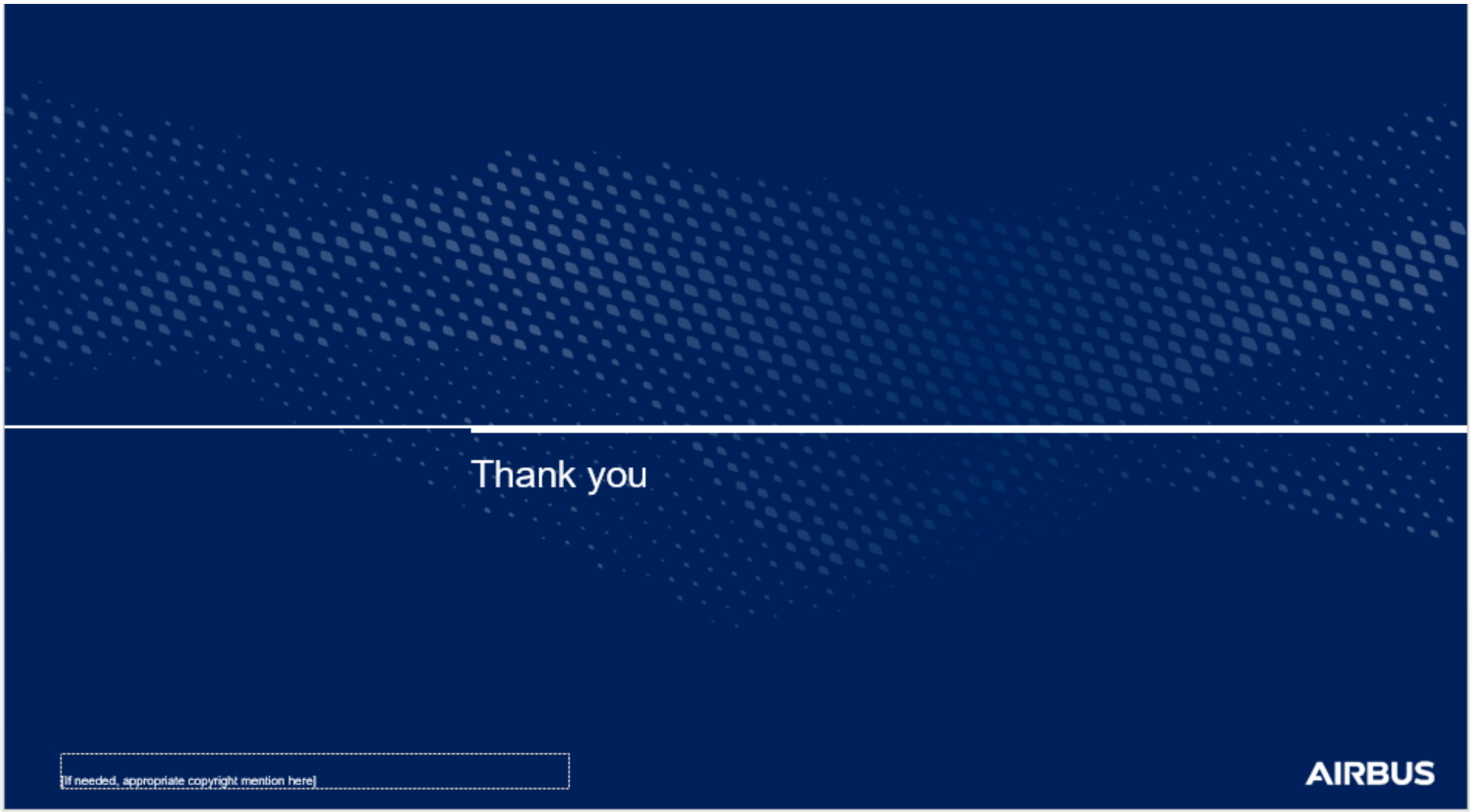

David Silver et al. "Deterministic policy gradient algorithms". In: *ICML*. 2014.

References IV

Wen Sun, J Andrew Bagnell, and Byron Boots. "Truncated Horizon Policy Search: Combining Reinforcement Learning & Imitation Learning". In: *arXiv preprint arXiv:1805.11240* (2018).

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction(Online Draft)*. MIT press, 2017.

Questions / Discussions



Modelling Dynamic Temporal Behavior Using Recurrent Neural Network

by

Olagoke Lukman Olabisi

A **Semester Project** submitted

in the
School of Computer and Communications System
Laboratory of Artificial Intelligence

Supervised by: Prof Boi Faltings
Assisted by: Igor Kulev

June 2018

“ It always seems impossible until it’s done. ”

Nelson Mandela

Abstract

In recent times air quality and its temporal and spatial variations has been of major concerns to all stakeholders in urban areas. This is obviously due to the increase in anthropogenic factors that affects the concentrations of the major air pollutants in these regions. Hence there is need to monitor the concentration of these pollutants. However, the resources to maintain air-quality-monitor stations across the length and breadth of even a small region can be quite challenging and would be influenced by a number of factors- traffic volume, meteorological condition.

In this project we propose to learn to predict the dynamics and evolution of the air quality index using Recurrent Neural Network (RNN) based on pre-existing data collected in the city of Skopje, Macedonia from sensors placed at Five different locations. In particular, we use a variant of the RNN called the long short term memory (LSTM) which has proven to be powerful and effective for such task.

We tested our models against 3 standard models in Time series prediction- Autoregressive model(AR), Moving Average (MA) and Autoregressive Integrated Moving average (ARIMA) and we recorded a significant performance.

Acknowledgements

There are many people who have helped shaped my ideas and from whom I have learnt a lot. Definitely I wont be able to mention all of them here. I hope those whose names don't appear here would still remember that I have them too at heart.

I thank Professor Boi Faltings of the Laboratory of Artificial Intelligence, EPFL, for offering me the opportunity and support to work on this project. A lot of thanks also goes to the assistant supervisor, Mr.Igor Kulev (Ph.D Supervisor at EPFL) for his constant support and mentoring while I work on the project.

I will also like to thank my parents for the good moral upbringing they gave me and Mr Shareefdeen Ashade for teaching me to know the importance of good education.

Contents

List of Figures

List of Tables

Abbreviations

Acronym	**W**hat (it) **S**tands **F**or.
AR	**A**uto**R**egressive
MA	**M**oving **A**verage
ARIMA	**A**uto**R**egressive **I**ntegrated **M**oving **A**verage
LSTM	**L**ong **S**hort **T**erm **M**emory
RNN	**R**ecurrent **N**eural **N**etwork
AQI	**A**ir **Q**uality **I**ndex
ppm ppb ppt	**P**arts **P**er **M**illion / **B**illion / **T**rillion
PM	**P**articulate **M**atter
VIM	**V**isualisation and **I**mputation of **M**issing values
ACF	**A**uto **C**orrelation **F**unction
PACF	**Partial A**uto **C**orrelation **F**unction

Dedicated to teachers and teachers everywhere.

Chapter 1

Background

1.1 Air Pollution And Air Quality Prediction: A General Overview

Air pollution is the process of emitting solid, liquid or gaseous material into the air from stationary or mobile sources, moving subsequently through an aerial path and perhaps being involved in physical or chemical transformations before it can have any environmental impacts[3]. It can either be Anthropogenic that is occuring as a result of human action or biogenic which occurs as a result of activities of other living organisms. The term air quality means the state of the air around us. Good air quality refers to clean, clear, unpolluted air. Generallly **Air quality** and **its temporal and spatial variations** are largely influenced by the nature of anthropogenic activities associated with various gaseous and particulate emissions and the set of prevailing meteorological conditions there.

To ensure human safety of humans, particularly in urban areas it is important to monitor the daily ambient air quality on a regular basis.To save cost of man-power, financial and instrumentation requirements of such monitoring , it is desirable to develop models which are robust, viable and of high predictive capability. To do such we turn to various machine learning models.Researchers have also established relationships between air pollutants and daily excess in mortality [4]. Though no model can be perfect, however a good model can help individuals and policy-makers in making right decisions will make the our planet habitable. In particular, children and elderly people are the most vulnerable to air quality hazards: this study can help render helpful decisions that can help prevent death and health impact of bad air quality to this vulnerable group.

Classically, atmospheric dispersion models[5][6], have been used to predict the ground level concentration of air pollutants near the sources. Such model requires precise knowledge of several source parameters and meteorological conditions. This does not a flexible option. However by using Neural Network we might be able to obtain a functional approximation for the non-linearirities between input parameters (meteorolrological parameters and pollutant past history) and with that be able to model future values for the poluutans.

1.1.1 Air Pollutants

We have defined air pollution and air quality. We now turn attention to the various air pollutants that can impact the quality of air thereby affecting the equilibrium in the ecological cycle or the well being of humans, plants or other organisms- thus causing air pollution. It is one or more of such pollutants that we will be building our model to learn and predict.

Furthermore, it should be noted that it's the quantity (or concentration) of a chemical in the air that makes the difference between "harmlessness" and "pollution" [1]. Carbon dioxide (CO2), for example, is present in the air around us at a typical concentration of less than 0.05 percent and breathing it in usually does no harm (we breathe it out all day long); but air with an extremely high concentration of carbon dioxide (say, 510 percent) is toxic. Any gas or particles could qualify as pollution if it reached a high enough concentration to do harm.

We also note in the passing that the phenomenon of air pollution usually involves a sequence of events: the generation of pollutants at and their release from a source; their transport and transformation in the atmosphere; and their effects on human beings, materials, and ecosystems.

We note that of Particulate matter (PM) is a term used to describe the mixture of solid particles and liquid droplets in the air. It can be either human-made or naturally occurring. PM2.5 means the mass per cubic metre of air of particles with a size (diameter) generally less than 2.5 micrometres (m). PM2.5 is also known as fine particulate matter. The same interpretation holds for PM10 but of larger diameter size.

Air Quality Index (AQI) is an important indicator for general public to understand easily how bad or good the air quality is for their health and to assist in data interpretation for decision making processes related to pollution mitigation measures and environmental management. It is defined as an index or rating scale for rating the daily combined effect of ambient air pollutants recorded at monitoring site. An increases

Table 1.1: Table of Air Pollutants [1]

Impurity	Source	Effects	Units of Measurement (Gravimetric)
SO2	Domestic and industrial burning of coal.	Irritation of the nerves in the nose, throat and airways. May also lead to constriction of the airways.	ppm/ppb/ppt
CO	Incomplete combustion of organic materials (Carbon containing). Wood, coal, oil, gas. Outdoors: vehicle exhausts, heating appliances. Indoors: smoking, heaters (unvented).	Reduces the oxygen-carrying capacity of the blood leading to headache, nausea, vomiting, eventually collapse and death.	ppm/ppb/ppt
O3(Ozone)	Product of chemical reaction between other pollutants (Nitrogen Oxide and hydrocarbons in the presence of sunlight).	Running eyes, throat irritation, breathing difficulties	ppm/ppb/ppt
NO2	Combustion of fossil fuels, road vehicles' power generation, industrial processes. Indoors: unvented gas cookers and other appliances	Throat and eye irritation (also involved in photochemical smog formation).	ppm/ppb/ppt
PM10 and PM2.5	Combustion processes and natural sources such as dust, diesel and smoke	Small particles can penetrate deep into the lungs and cannot be expelled. They may cause irritation and/or carry with them toxic or carcinogenic substances	ppm/ppb/ppt

in AQI is likely to have an adverse effect on human health, in particular vulnerable people –senior citizens, allergic or sick citizens– are particularly at risk. Computing the AQI requires an air pollutant concentration from a monitor or station.Our data source contains concentrations in ppm but they can be easily converted to AQI. We however we stick with the ppm as unit of measurement for pollutant concentration. We provide below table of pollutants that have serious effects on humans[1].

1.2 Time Series Prediction

A *time series* is a sequence S of historical measurements y_t of an observable variable y taken at equal time intervals. In this project the obevable variable is the particular pollutant we are interested in. With the variable observable we consider the problem of forecasting futre value of the pollutant. An important aspect forecasting is **the size of horizon**. If one step forecasting of time series is is already a challenging task, performing multistep-forecasting is more difficult[7] due to additional complications like accumulation of errors, reduced accuracy and increased uncrtainty. In this project we take on the muti-step forecasting challenge, paraphrased as:
Given the previous X hr/day(s)/week(s) obervations/hr pollutant concentration, do a muti-step forecast for the next 24 hrs ?
We note that the forecasting domain has been influenced, for a long time, by linear statistical methods such as ARIMA models. We will use the ARIMA model as our base model in this project. In recent years, machine learning models have drawn attention and have established themselves as serious contenders to classical statistical models in the forecasting community [8]. These models, also called *data-driven models* are examples of *nonparametric nonlinear models* which use only historical data to learn the stochastic dependency between the past and the future. Of these data driven models we will use a variant called *Long Short Term Memory* which is a *RNN* endowed with a long memory capacity.

1.3 Times Forecast and Modelling

1.3.1 Introucing Time Series Dynamics

We present a brief formalism of our modelling problem as applied to a time series. We adopt the emergent view in dynamic systems theory that random behavior may be generated by deterministic systems with only a small number of degree of freedom, interacting non-linearly. This complicated and aperiodic behavior is called *deterministic chaos.*[9]

It is assumed in this work that many classes of experimental time series may be analyzed within the framework of a dynamical systems approach. We thus interpret the time series as the observable of a dynamic system whose state s evolves in a state space $\Gamma \subset \mathbb{R}^g$ according to the law:

$$s(t) = \mathbb{F}^t(s(0))$$

where $\mathbb{F} : \Gamma \to \Gamma$ is the mapping representing the dynamics, $\mathbb{F}^t$ is the iterated versions, and $s(t) \in \Gamma$ denotes the value of the state at time t. In the absence of noise the time series ia related to the dynamic systems according to the relation

$$y_t = \mathbb{G}(s(t))$$

where $\mathbb{G} : \Gamma \to \mathbb{R}^D$ is called the *measurement function* and D is the dimension of the series. In this project we restrict $D = 1$.

The functions $\mathbb{F}$ and $\mathbb{G}$ are unknown and so we cannot reconstruct them in their original state. However we might try to reconstruct a state space that is in some sense equivalent to the original[10].

1.3.2 State Space Reconstruction Problem

The *state space reconstruction* problem consist in reconstructing the state as described above when the only available information is contanined in the sequence of observations of y_t. The *Takens Theorem* suggest that for a wide class of deterministic systems, there exists a mapping (*delay reconstruction map*) $\Phi : \Gamma \to \mathbb{R}^n$:

$$\Phi(s(t)) = \{\mathbb{G}(\mathbb{F}^{-d}(s(t))), \ldots, \mathbb{G}(\mathbb{F}^{-d-n+1}(s(t)))\}$$

between a finite window of the time series $y_{t-d}, \ldots, y_{t-d-n+1}$ (embedding vector) and the state of the dynamic system underlying the series, where d is the time lag and n is the number of past values taken into consideration. Consequently, Taken showed that Φ is an embedding. From this fact it can be asseted that if Φ is an embedding then a smooth dynamics $f : \mathbb{R} \to \mathbb{R}$ is induced in the space of reconstructed vectors:

$$y_t = f(y_{t-d}, y_{t-d-1}, \ldots, y_{t-d-n+1}) \qquad (1.1)$$

From this , we conclude that the reconstructed states can be used to estimate f and consequently f can be used as alternative to $\mathbb{F}$ and $\mathbb{G}$ in time series forecasting[10]. However, the representation in (1.1) does not take into account any noise component, since it assumes a deterministic process can accurately describe the time series. We extend this deterministic formulation into Statistical *Nonlinear Autoregressive* formulation:

$$y_t = f(y_{t-d}, y_{t-d-1}, \ldots, y_{t-d-n+1} + w(t)) \qquad (1.2)$$

where the missing information is here lumped into a noise term which we denote by $\mathbf{w}$. In the rest of this work we will refer to the formulation (1.2) as our general representation

of a time series. Furthermore, we note that the success of a reconstruction approach starting from a set of observed data-the air pollutant history, depends on:

- Choice of hypotheisis that approximates f
- the choice of order n
- and the time lag.

Our choice of f the approximation function will be LSTM (and linear regression), and we use order of 1 (univariate series) and the time lag will be 24 hours covering a full day. Justification for our choice here will be discussed further in later chapters.

1.3.3 From Dynamic System model to Supervised Learning Model

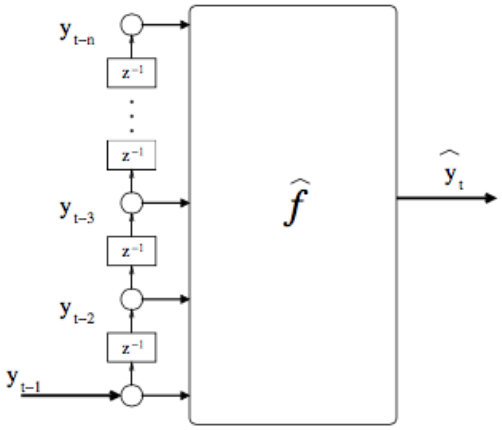
The embedding formulation of (1.2) suggests that, if there we have a historical record S, the problem of one-step forecasting can be tackled as a problem of supervised learning. Supervised learning modelling involves evaluating or approximating, on the basis of a finite set of observation, the relationship between *input* variables and one or more *output* variables, given that the output variables are dependent on the input variable [10]. In the forecasting setting the training set is derived by the historical record S by creating $[(N-n-1)\times n]$:

$$X = \begin{bmatrix} y_{N-1} & y_{N-2} & y_{N-3} & \cdots & y_{N-n-1} \\ y_{N-2} & y_{N-3} & y_{N-4} & \cdots & y_{N-n-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_{N-n} & y_{N-1} & y_{N-2} & \cdots & y_1 \end{bmatrix}$$

and the [(N-n-1)*1] output vector:

$$Y = \begin{bmatrix} y_{N-1} \\ y_{N-2} \\ \vdots \\ y_{N-n} \end{bmatrix}$$

Figure 1.1: One Step Forecasting Training



[The approximator $\hat{f}$, which is LSTM in this project, returns the prediction value at the time step t+1 as a function of n previous values the rectangular box containing z^{-1} represent unit delay operator i.e $y_{t-1} = z^{-1}y_t$]

1.4 Strategies for Multistep Time series forecast

We discuss here how supervised learning approach or learning technique can be extended to tackle the *Multi-step* forecasting[10].

A multi-step time series forecasting task consist of predicting the next H values $[y_{N+1}, \ldots, y_{N+1}]$ of a historical time series $[y_1, \ldots, y_N]$ composed of N observations, where H>1 denotes the time horizon. This section will discuss the strategies to adopt machine learning for time series forecast.

1.4.1 Recursive Strategy

1.4.2 Recursive strategy

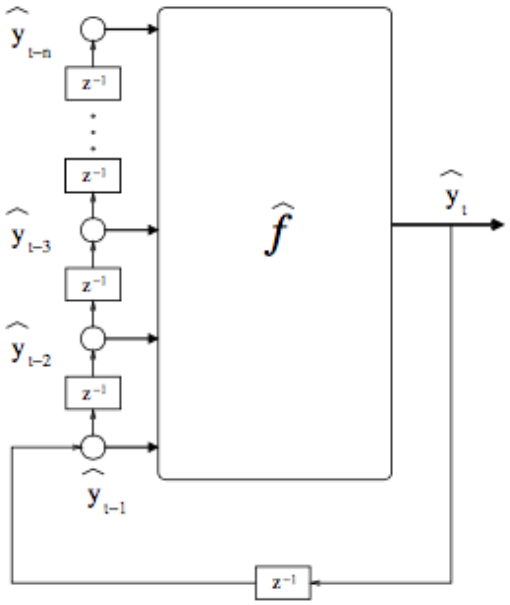
Recursive strategy trains first a one step model f:

$$y_t = f(y_{t-d}, y_{t-d-1}, \ldots, y_{t-d-n+1} + w_t)) \qquad (1.3)$$

with t∈(n,...,N-1) and then uses it recursively for returning a multistep prediction as shown in the diagram fig(1.2):

A well known draw back of the recursive approach is its sensitivity to the estimation error, since estimated values, instead of actual values are more pr less used when we get further in the future. However, the recursive models as been shown to be very successful in real-world time series forecast][11] when implemented with RNN. It is this approach that wil be adopted in this project.

Figure 1.2: Iterated Prediction



[The approximator $\hat{f}$ returns the prediction value at the time step t+1 by iterating the predictions obtained in the previous steps]

1.4.3 Direct Strategy

The direct strategy learns independently H models f_h

$$y_t = f_h(y_{t-d}, y_{t-d-1}, \ldots, y_{t-d-n+1} + w_{t+h})) \qquad (1.4)$$

. where t$\in n, \ldots, N-h$ and h$\in 1, \ldots, H$ and returns a multi-step forecast by concatenating the H predictions.Since the strategy does not use any approximated values to compute forecast it is not prone to accumulation of errors. But since the H models are learned independently no statistical dependencies between predictions is considered. In addition the direct method demands a large vomputational time since the number of models to learn equal the size of the horizon.

1.4.4 DirRec Strategy

The DirRec strategy [12] combines the architectures and the principles underlying the Direct and the Recursive strategies. DirRec computes the forecasts with different models(like direct strategy) for every horizon, and at each time step, it enlargesthe set of inputs by adding variables corresponding to the forecasts of the previous step(Recursive strategy). Unlike the last two models, the embedding size is not the same for all the horizons.

1.5 Conclusion

In this chapter we provided introduced air pollution and the motivation to have a model for predicting it.. Further details especially on the background domain knowledge can be found on the github page "https://github.com/adderbyte/Project_LSTM_Air_Pollution/blob/master/Air_Pollution_essentials/5_Air_pollutants.md". We emphasised the need to monitor air pollutants and the effects it might have on us. In addition we provide a mathematical formalism for the topic of discussion. With these tools,*domain knowledge, motivation and mathematical formalism*, we are ready to go dive into the more crucial topics of this study.

In the next chapter we will discuss LSTM, the function approximator we will use in our prediction.

Chapter 2

Functional Approximator: Long Short Term Memory

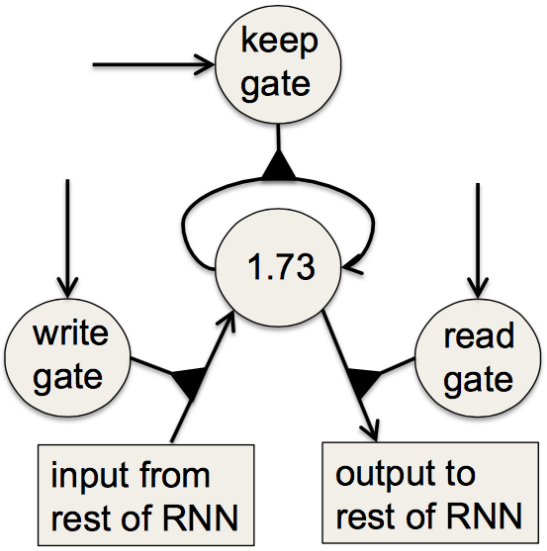
2.1 Introducing LSTM

In this section we develop in-depth discussion of the *Long Short Term Memory*, which is the *Functional approximator* (see fig 1.2) used in this work. Appendix A provides an informal and easy-to-learn introduction to LSTM.

Long Short-Term Memory (LSTM) is a specific recurrent neural network (RNN) architecture designed to model temporal sequences or time series sets and their `long-range` dependencies more accurately than conventional RNNs[13]. Both LSTM and RNN have successfully being applied to various sequence prediction and sequence labelling task. However, vanilla RNN suffers form vanishing and exploding gradient which limits its use as long term memory cell – it can't capture long term dependencies [13]. Below are some historical facts about LSTM:

- Hochreiter and Schmidhuber [14] solved the problem of getting an RNN to remember things for a long time (like hundreds of time steps).
- They designed a memory cell using logistic and linear units with multiplicative interactions.
- Information gets into the cell whenever its write gate is on.
- The information stays in the cell so long as its keep gate is on.
- Information can be read from the cell by turning on its read gate.

Figure 2.1: Simple Memory Cell



[Image from Lecture Note by Geoffrey Hinton [13].]

From here we turn to how we can make simple long term memory. We will use fig (2.1).

2.1.1 Simple Memory cell

To preserve information for a long time in the activities of an RNN, we use a circuit that implements an analog memory cell as in fig 2.1 and fig 2.2.

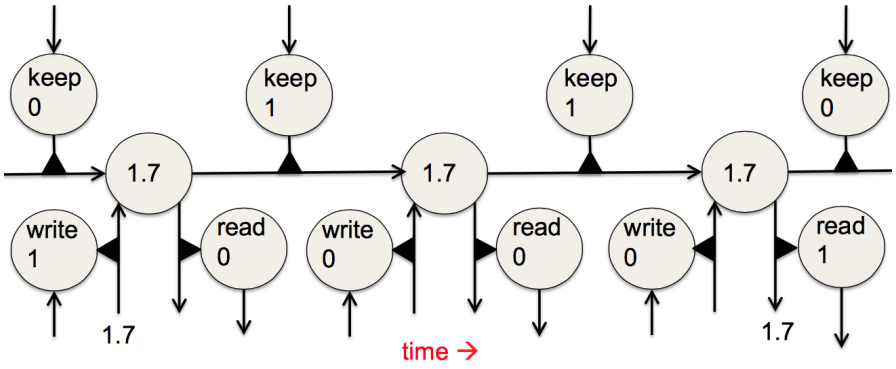
- A linear unit that has a self-link with a weight of 1 will maintain its state.
- Information is stored in the cell by activating its write gate.
- Information is retrieved by activating the read gate.
- We can backpropagate through this circuit because logistics are have nice derivatives.

Having understood the simple intuition behind the memory cell. We go on to give a more detailed architecture.

2.2 From Memory cell to Full LSTM Architecture

The LSTM contains special units called *memory blocks* in the RNN layer. The memory blocks contain memory cells with self-connections storing the temporal state of the

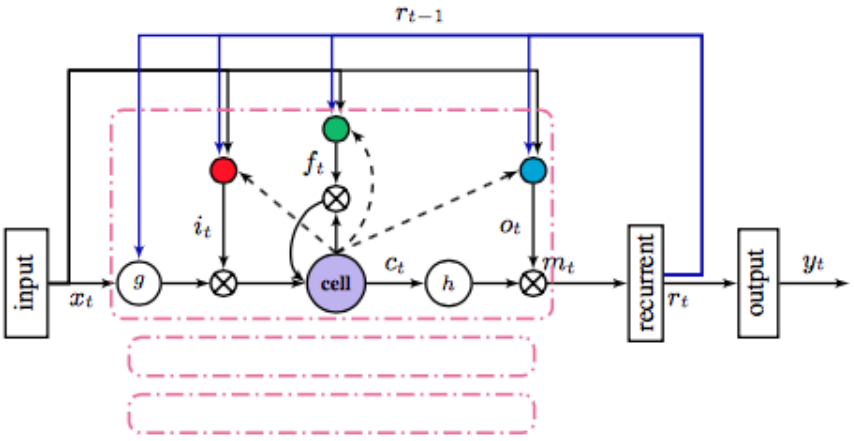
Figure 2.2: Back propagation in Memory cell



[Image from Lecture Note by Geoffrey Hinton [13]. This can be read as a sequence of reading, writing through the network. Keep 0 implies no information stays in the cell. We can read when read is 1 or write to the cell when write is . We can also backpropagate through the network.]

network.The special multiplicative units called *gates* control the flow of information. Each memory block in the conventional architecture had an input gate and an output gate. The input gate controls the flow of input activations into the memory cell(see Fig 2.1 ,2.2, qnd 2.3). The output gate controls the output flow of cell activations into the rest of the network[2].Later, the forget gate was added to the memory block[15]. This addressed a weakness of LSTM models preventing them from processing continuous input streams that are not segmented into subsequences. The forget gate scales the internal state of the cell before adding it as input to the cell through the self-recurrent connection of the cell, therefore adaptively forgetting or resetting the cells memory.In addition, the modern LSTM architecture contains *peephole* connections from its internal cells to the gates in the same cell to learn precise timing of the outputs [16].

Figure 2.3: Typical LSTM Cell Architecture



[Image from [2].]

An LSTM network carries out the supervised learning as described in the chapter 1. It basically maps input sequences from input to output by calculating network the network input activations using the following equations iteratively:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1}W_{ic}c_{t-1} + b_i) \tag{2.1}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f) \tag{2.2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \tag{2.3}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \tag{2.4}$$

$$m_t = o_t \odot h(c_t) \tag{2.5}$$

$$y_t = \phi(W_{ym}m_t + b_y) \tag{2.6}$$

Explanation for the terms in the equations are provided in the table below :

Table 2.1: Table of LSTM Euation Notations[2]

Equation Symbol	Definition	Explanation
$i, f, o\ , c$	These are respectively input, forget , output and cell activation gates	
m	the cell output activation vector	
$\odot$	element-wise product of vectors	
ϕ	Network output activation function	typically Softmax is used.
W	Weight	W_{ix} implies weight matrices from the input gate to the output gate.
g and h	input and output activation function	Generally tanh function
W_{ic}, W_{fc}, W_{oc}	Diagonal weight matrices for peep hole connection	
b	bias vector	b_i is read as the input gate bias vector
σ	logistic sigmoid function	

2.3 Linear Regression Regression

As discussed earlier, LSTM is our function approximator. In our actual implementation we will combine this LSTM network with linear regression. The aim is to use LSTM

has our memory to learn the dependencies in the time series. As shown in the Appendix A where we discussed on LSTM extensively, we get cell state output and current output. With both output, the recent cell state or cell output vector, we implement linear regression to predict the next value in the time series .The model thus learns the appropriate weight in the linear prediction part such that performance(see Appendix A) is maximised. Thus it is necessary for us to provide a little treatise on linear regression.

2.3.1 Simple Linear Regression

We assume a model:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

where β_0 and β_1 are two unknown constants that represent the *slope* and the *intercept* also known as *coefficients* or *parameters* and ϵ is the error term.We here emphasise that the X is the output we get from the neural net. Given some estimates of β_0 and β_1 as $\hat{\beta}_0$ $\hat{\beta}_1$, we predict the future value:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

where $\hat{y}$ indicate a prediction of Y on the basis $X = x$. The *hat* symbol denotes an estimated value.

We then define:

$$e_i = y_i - \hat{y}_i$$

as the i^{th} residual. The aim of the prediction part of the model is to estimate β_0 and β_1 that can be used for future prediction. The sum of residuals is then :

$$RSS = e_1^2 + e_2^2 + e_3^2 \dots e_n^2$$

2.3.2 Multiple Regression

We can extend the simple regression to multiple regression. The multiple regression was actually used in our implementation. We define this as:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n + \epsilon$$

We estimate $\beta_0, \beta_1, ..., \beta_p$ as the values that minimize the sum of squared (performance factor) residuals:

$$RSS = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

In this function we also used the the coefficient of determination, denoted R2 or r2 and pronounced "R squared", to measure the performance of our model. The coefficient of determination is indicates the proportion of the variance in the dependent variable that is predictable from the independent variable. A value of R^2 close to 1 is considered good .

2.4 Conclusion

In this cahpter we discussed the functional approximator that will form the core of our model. Mor details about the function approximator is provided in appendix A. In the next chapter, we provide details about the data we used for the model prediction. We also provide details about the feature engineering implemented to ensure our model works well.

Chapter 3

Data Analysis And Feature Engineering

3.1 Data Sets Analysis

We present the analysis of data that was used in our model. The data sets consist of consist of impurities concentrations *PM10, PM25,* NO_2*, CO,* O_3 collected from 3 regions (*Estern Region, Western Region, Skopje Region*) of Macedonia. In each region, measuring devices/sensors were placed at different locations and the readings recorded. **We will focus on PM10 prediction for one region (Skopje) and one location (Rektorat)**.

As noted earlier each region contain data sets from different location. We will answer the following question in this study:

- How does the PM10 concentration from other locations affect the PM10 prediction for the location of interest?
- How does the concentration of other impurities affect the prediction of PM10 for the region of interest?
- How do we handle missing values for the PM10?
- What improvement do we get if we have weather data?

To start with a brief look at the data set for Skopje is provided below 3.1. Note the presence of **missing values** and the fact that the column name bears the readings for different locations. There are 7 locations present in the Skopke region data: *'Centar',*

Figure 3.1: Data set for Skopje Region

	date	Lat	Lon	PM10	NAME
1	26.01.2007 10	42.003600	21.463600		Gazi Baba
2	26.01.2007 10	42.006667	21.386944	61.67	Karpos
3	26.01.2007 10	41.977821	21.464474		Lisice
4	26.01.2007 10	41.999200	21.440800	95.57	Rektorat
5	26.01.2007 10	42.015800	21.650600		Mrsevci
6	26.01.2007 10	41.987500	21.652500		Miladinovci
7	15.01.2007 01	41.992500	21.423611		Centar
8	15.01.2007 01	42.003600	21.463600		Gazi Baba

'Gazi Baba', 'Karpos', 'Lisice', 'Miladinovci', 'Mrsevci', 'Rektorat' We are interested in **Rektorat** location.We will exploit this dataset further and get some summaries. First lets look at the summary for missingness in the data 3.2 set:

Figure 3.2: Figure showing Missingness in the data set

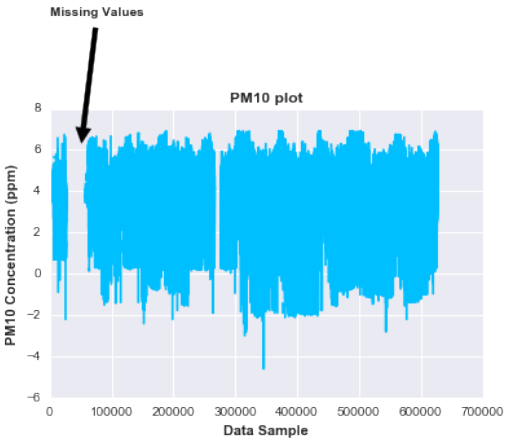
```
date          0
Lat           0
Lon           0
PM10     255713
NAME          0
dtype: int64
```

This include all the locations.

This constitute 40.816% of the PM10 concentrations that were collected. A quick plot for PM10 values will also reveal the missingness spread[1]:

[1]We observe the presence of negative readings in the data set and we have to comment on this. While the official Environmental Protection Agency (EPA) regulation does not address negative numbers, the EPA, nevertheless, has a standing convention of allowing negative data into Air Quality System. The default minimum drift/range Quality Check check is -4.9 ppb for Ozone, -4.9 μg/m3 for PM10, and -4.99 for PM2.5. While negative concentrations can never really exist because the atmosphere cannot have a negative amount of PM, we often validate these hours to account for instrument drift/noise - this is outside the scope of this study. Negative values can however be used for computational purposes in both the 24-hour average and Air quality Index concentration. This is done because removing these concentrations or setting them to zero would bias the computation.[17]

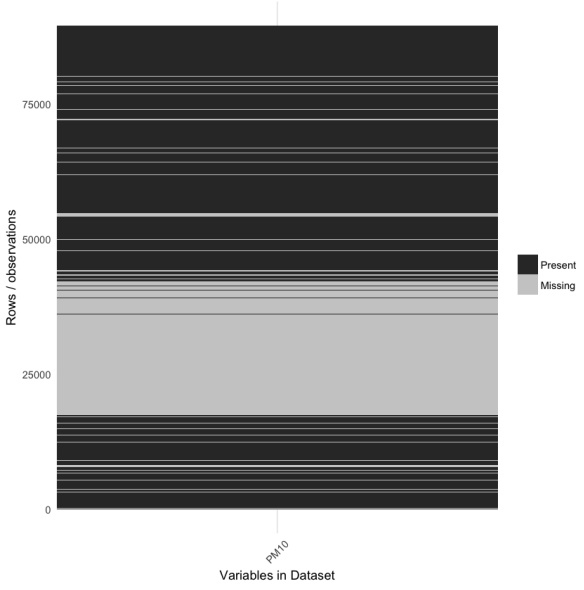
Figure 3.3: Plot for PM10



Note missing value

To even dig data and see missingness across the data set clearly we used the adapted version of VIM package in R and we obtain a plot shown below:

Figure 3.4: Plot showing Misingness in the data set



A routine cleaning of the data set fig 3.1 again gives us some basic insight about the data set:

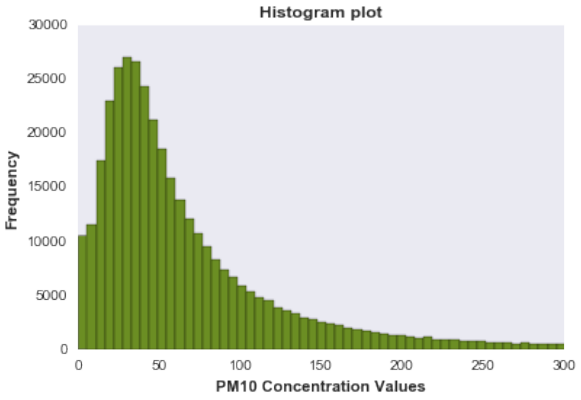
Figure 3.5

	date	Lat	Lon	PM10	NAME	time	month	day	daysInterval	days_interval
0	2007-01-26 10:00:00	41.992500	21.423611	NaN	0	10	1	26	0 days 00:00:00	0
1	2007-01-26 10:00:00	42.003600	21.463600	NaN	1	10	1	26	0 days 00:00:00	0
2	2007-01-26 10:00:00	42.006667	21.386944	61.67	2	10	1	26	0 days 00:00:00	0
3	2007-01-26 10:00:00	41.977821	21.464474	NaN	3	10	1	26	0 days 00:00:00	0
4	2007-01-26 10:00:00	41.999200	21.440800	95.57	6	10	1	26	0 days 00:00:00	0
5	2007-01-26 10:00:00	42.015800	21.650600	NaN	5	10	1	26	0 days 00:00:00	0
6	2007-01-26 10:00:00	41.987500	21.652500	NaN	4	10	1	26	0 days 00:00:00	0
7	2007-01-15 01:00:00	41.992500	21.423611	NaN	0	1	1	15	-12 days +15:00:00	-12

In the above the locations have now be encoded with numbers $0-6$ under the column name 'Name'. In addition we can see the 'day', 'month', and 'days Interval' in which the data were collected. The Longitude and Latitude columns ('Lon', 'Lat' in the figure) will be removed since we are maing predictions within the same geographical region. And moreover our data set bank is not large enough to make us take the Longitude and Latitude so that we need to add only features that are important for the function approximator.

We also have a glance at the histogram plot:

Figure 3.6: Plot showing Misingness in the data set



We observe there are tail values with high PM10 concentration. But most PM10 concentration falls within the range $0-150$ at most,which makes sense since high values should occur only when there are factors (such as increase in C0) level that could drive it increase .

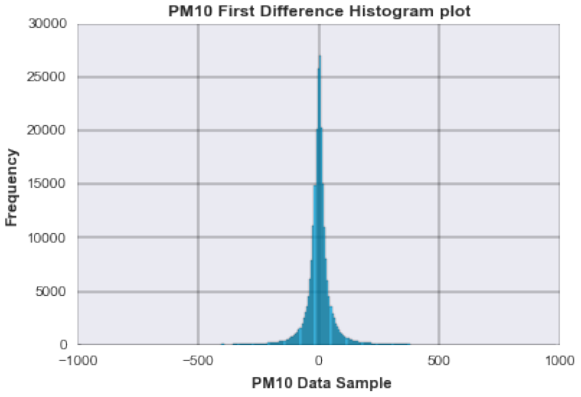
Visual inspection shows that the histogram distribution is "right skewed" and we would need to apply a normalizing normal transformation. Skewed distributions bring a certain philosophical complexity to the very process of estimating a "typical value" for the distribution[18]. To be specific, suppose that we wish to summarize the PM10 observations by a "typical value". What does typical value mean? If the

distribution is symmetric, the typical value is unambiguous - it is a well-defined center of the distribution. For example, for a bell-shaped symmetric distribution, a center point is identical to that value at the peak of the distribution.

For a skewed distribution, however, there is no "center" in the usual sense of the word. Skewed data often occur due to lower or upper bounds on the data. That is, data that have a lower bound are often skewed right while data that have an upper bound are often skewed left.We can see that the PM10 values have upper bounds so it is skewed right to the right.

We also studied the histogram plot for the *differenced or lagged* PM10 concentration: This shows that the first difference shows a symmetric distribution and it is stationary as we will show later :

Figure 3.7: PM10 first difference plot



This is a good news. Since stationarity of data set is a useful property for statistical analysis , modelling and prediction. A simple statistic of the cleaned data set for PM10 is provided below:

Figure 3.8: PM10 statistics at a glance

```
count    370787.000000
mean         75.526098
std          86.283927
min           0.000000
25%                NaN
50%                NaN
75%                NaN
max        1093.080000
Name: PM10, dtype: float64
```

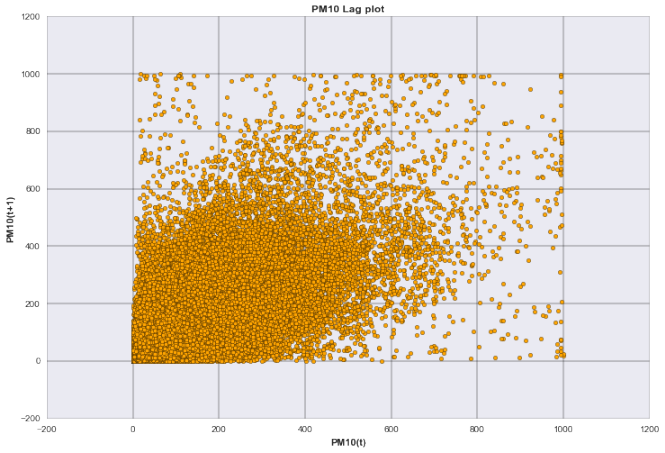
From fig 3.8 , we note that we might need to normalize or standadize our data set before training (the maximum value reported is 1093) to avoid the effect of outliers. In addition we still have to do data transformation. More on this to come. Looking at the missingness plot we can observe why median value id reported as Nan - there are much Nan in the data set

3.1.1 Lag plot

A lag plot is a special type of scatter plot with the two variables (X, Y) lagged . A lag is a fixed amount of passing time; One set of observations in a time series is plotted (lagged) against a second, later set of data - in the figure below that is why the axes are labelled PM10(t) and PM10(t+1) [19].

Lag plot is used to help evaluate whether the values in a data set or time series are random. If the data are random, the lag plot will exhibit no identifiable pattern. If the data are not random, the lag plot will demonstrate a clearly identifiable pattern. The type of pattern can aid the user in identifying the non-random structure in the data. Lag plots can also help to identify outliers.

Figure 3.9: Lag plot for PM10



- *Outliers*: In the lag plot we can definitely observe numerous outliers (e.g values in the range $900 - 1000$).
- *Randomness*: We can obeserve a visible linear trend at the origin of the axis up to approximately the value 800. That is why we have chosen ARIMA, AR and MA as base model
- *Correlation*: From the plot we can hypothesize that the data set will show weak correlation.

3.1.2 Missing data Mechanisms

Presented below is a table of taxonomy of missing data[20]:

Table 3.1: Table of Taxonomy of Missing data

Taxonomy	Definition	Example
Missing Completely at random (MCAR)	Missing value y neither depends on x nor y	This implies hypothesizing that missing values in PM10 in the data set does not depend on the month or PM10 itself
Missing at Random (MAR)	Missing value y depends on x but not y	This implies hypothesizing that missing PM10 values depend on the month but not on PM10 itself.
Missing not at random (MNAR)	The probability of a missing value depends on the variable that is missing	This implies hypothesizing that the probability of missingness in PM10 is conditioned on PM10 itself.

In this study, it is assumed that the missing values on PM10 are missing at random. The assumption stems from the fact that observation of the data set shows regions of large missingness, which we can attribute to a device fault or error in collection. Both of these features (device fault or error in collection) are not available to us nor can we make any meaningful assumption about them. Moreover these two factors were not also reflected in the feature sets that were collected.There are several ways to deal with missing data[20]:

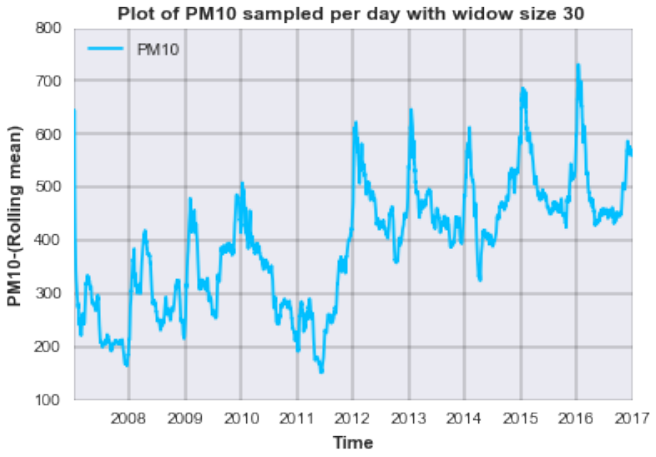
- Deletion Method (Pairwise and List wise deletion)
- Single Imputation method (for example Mean/Mode substitution, Dummy variable control, Conditional mean substitution/Regression Imputation.
- Model based methods (Maximum likelihood, multiple imputation)

The name the each of these methods are indicating of how it handles missing data. For example Dummy Variable control replaces missing values with *indicator variable*. Mean and median replaces missing values with and and mode of the data set. In multiple Imputation methids data is filled in with imputed values using specified regression model, this process is repeated m times. We thus have m different data set. In maximum likelihood approach, we identify the set of parameter values that produces the highest log-likelihood[20]. On each of the model we shall propose we will indicate how we deal with missing values.

3.1.3 Time Series plot

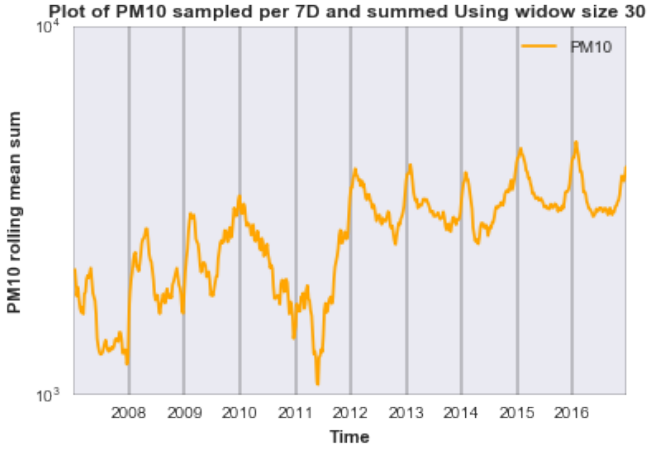
We provide here a time series plot of the data set. We resampled the data for differentt window-size. Thereafter a rolling mean was use across the the concentration in order to observe the trends in pollutant concentration.

Figure 3.10: Lag plot for PM10



We can observe an upward rise in the the PM10 pollutant across the years. Applying a sampling period of 7 days and using a rolling mean across the data gives the following trend:

Figure 3.11: Lag plot for PM10



A visual inspection of this plot indicates that a simple linear fit should be sufficient to remove this upward trend [18] - the difference plot 3.7 confirms this fact since it was obtained by doing a simple linear shift.

3.1.4 Stationarity and Autocorrelation plot

3.1.4.1 Standardization

For this part we assume have standardized our data set. Since the range of values of raw PM10 concentrration varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. Thus the need for Normalization. *Feature standardization* makes the values of each feature in the data have zero-mean (when subtracting the mean in the numerator) and unit-variance. The simple statndardization equation was used:

$$PM10_{standardized} = \frac{PM10 - PM10_{mean}}{PM10_{std}}$$

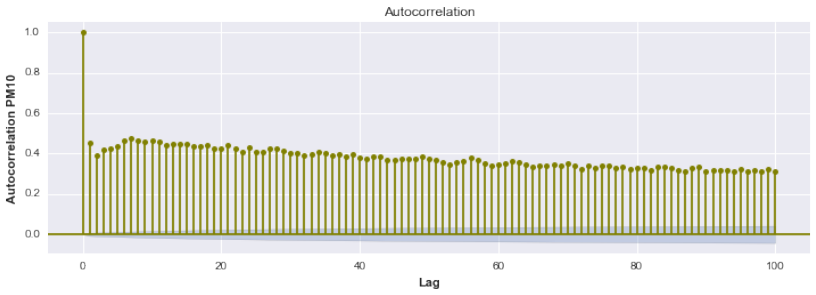
In particular we have standardized the data set before carrying out the following analysis that follows. In addition null values were filled with zero to enable meaningful deduction. In later part where we build the model we would indicate how missing values were treated

3.1.4.2 Autocorrelation

Autocorrelation plots are often used for checking randomness in time series. This is done by computing autocorrelations for data values at varying time lags. If time series is random, such autocorrelations should be near zero for any and all time-lag separations.

If time series is non-random then one or more of the autocorrelations will be significantly non-zero. The blue shade lines at the bottom of the plot gives the confidence bands (95-99). The autocorelation plot is given below:

Figure 3.12: Autocorrelation plot



This plot has been generated using Statmodels statistical tool. Different packages not so similar ACF plot. The plots were obtained after the data sets has been standardized

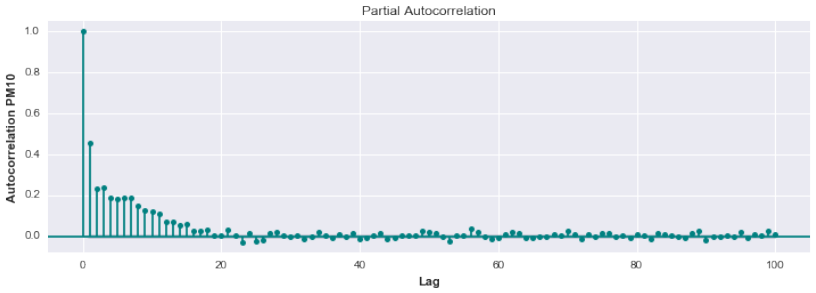
From the plot above it is evident that the ACF plot decays slowly to zero. We infer from this that there is visible trend in the plot[21]. We are also confident that the data set is not some random white noise.

3.1.4.3 Partial Autocorrelation plot

In general, a partial correlation is a conditional correlation.It is the correlation between two variables under the assumption that we know and take into account the values of some other set of variables.For instance, consider a regression context in which y = response variable and x_1, x_2 and x_3 are predictor variables. The partial correlation between y and x_3 is the correlation between the variables determined taking into account how both y and x_3 are related to x_1 and x_2.

For a time series, the partial autocorrelation between PM10t and PM10$(t-h)$ is defined as the conditional correlation between PM10 and PM10$(t-h)$, conditional on PM10$(t-h+1), \ldots,$PM10$(t-1)$, the set of observations that come between the time points t and th. The PACF for the PM10 data set concentration is shown below:

Figure 3.13: Autocorrelation plot

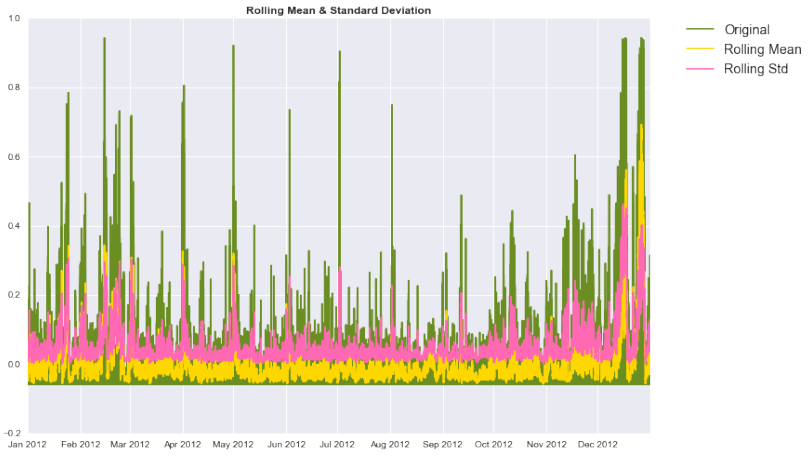


From this plot we can answer the question how large should the window size be?, or how far back in time should we look to make prediction for the next PM10 concentration/hr for next 24 hours? We can hyothesize that since for lag of 24 the PACF shows significant dependency3.13, we will work with a window size of **24**. Thus we reframe the question asked earlier to: *Given the past 24 value of PM10 concentration we want to predict the next 24 hour.*

3.1.4.4 Stationarity

A common assumption in most time series technique is that the time series is stationary. A stationary process has the property that the mean, variance and autocorrelation structure do not change over time [18]. A stationarized series is relatively easy to predict: we simply predict that its statistical properties will be the same in the future as they have been in the past. Another reason for trying to stationarize a time series is to be able to obtain meaningful sample statistics such as means, variances, and correlations with other variables. Such statistics are useful as descriptors of future behavior only if the series is stationary. For example, if the series is consistently increasing over time, the sample mean and variance will grow with the size of the sample, and they will always underestimate the mean and variance in future periods. And if the mean and variance of a series are not well-defined, then neither are its correlations with other variables. For this reason we should be cautious about trying to extrapolate regression models fitted to nonstationary data. Thus it is important we test for stationarity. Especially since we would also be using Linear regression for the prediction part once we are outside the LSTM cell. We can check for stationarity by plotting the rolling mean and doing a Dickey-fuller test[22][23]:

Figure 3.14: Rolling mean and std plot



We present a result of the Dickey-Fuller start on the plot above: Results of Dickey-Fuller Test:

Table 3.2: Dfuller Unit Toot Test Result

Test Result	Statistical Value
Test Statistic	-11.25387
p-value	1.685844e-20
Lags Used	6.100000e+01
Number of Observations Used	6.258800e+04
Critical Value (1%)	-3.430454
Critical Value (5%)	-2.861586e+00
Critical Value (10%)	-2.566795e+00

The null hypothesis of the Augmented Dickey-Fuller is that there is a unit root (non-stationaroty), with the alternative that there is no unit root (stationarity). From the returned result for the p-value we can safely reject the null value.

Doing the same thing for the shifted(by 1) PM10 values we obtain the p value of zero:

Table 3.3: Dfuller Unit Root test for Shifted PM10 values

Test Result	Statistical Value
Test Statistic	-40.473470
p-value	0.000000
Lags Used	61.000000
Number of Observations Used	62588.000000
Critical Value (10%)	-3.553318
Critical Value (5%)	-3.832484
Critical Value (1%)	-4.371315

3.2 Standardization and Data Transformation

Up to this moment we have only discussed about PM10. Now we want to add other air pollutants to the feature set for use in the prediction and dp some transformation of the data set.

3.2.1 Box Cox transformation

The aim of the Box-Cox transformations is to ensure the usual assumptions for Linear Model hold: $\mathbf{y} \sim \mathcal{N}(X\beta, \sigma^2 I_n)$ - that is transform the data to normal.[2] Draper and Cox (1969) in their work this [24] concluded that even in cases that no power-transformation could bring the distribution to exactly normal, the usual estimates of boxcox parameter λ will lead to a distribution that satisfies certain restrictions and thus will be usually symmetric. We use the version of the Box Cox Transformation that is not sensitive to negative values and Nan values.

[2]Linear regression by itself does not need the normal (gaussian) assumption, the estimators can be calculated (by linear least squares) without any need of such assumption, and makes perfect sense without it.

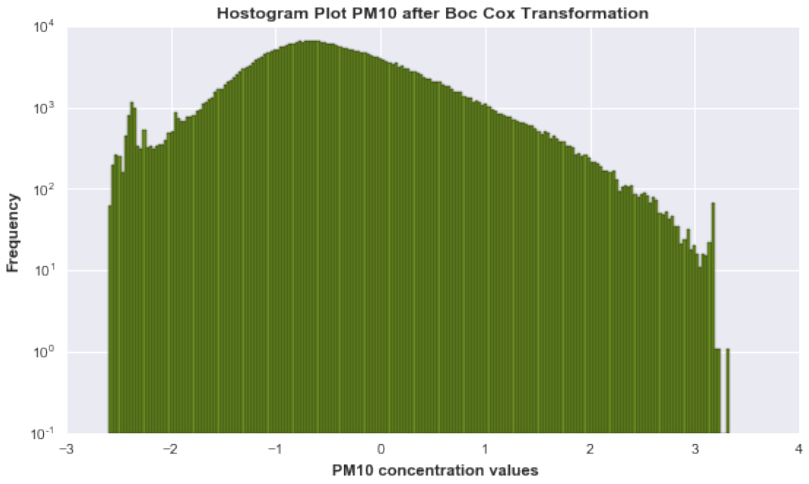
But then, we will want to understand some of the properties of this method,and provide answers to questions such as: are the least squares estimators optimal in some sense? or can we do better with some alternative estimators? Only under the normal distribution of error terms, we can show that these estimators are, indeed, optimal - for instance they are "unbiased of minimum variance", or "maximum likelihood". No such thing can be proved without the normal assumption.

Also, if one wants to construct (and analyze properties of) confidence intervals or hypothesis tests, then we use the normal assumption

$$y(\boldsymbol{\lambda}) = \begin{cases} \frac{(y+\lambda_2)^{\lambda_1}-1}{\lambda}, & \text{if } \lambda_1 \neq 1; \\ \log y, & \text{if } \lambda_1 = \text{otherwise} \end{cases} \tag{3.1}$$

where y is the PM10 concentration in our study. In practice, we could choose λ_2 such that $y + \lambda_2 > 0 \forall y$. So,we could only view λ_1 as the model parameter. Summary of details on Boxcox transformation can be found in [25]. Applying Boxcox transofrmation to the PM10 data set gives the following histogram plot[3], note the symmetry and compare with fig 3.9.

Figure 3.15: Rolling mean and std plot



In summary, standardization was applied to the whole feature set but for the target variable which is PM10, we applied. We now have the following:

[3]It should be noted that we have to conduct a test for the best that fits the model. The details of this parameter search and box cox function can be found in the git repo `https://github.com/adderbyte/Project_LSTM_Air_Pollution`

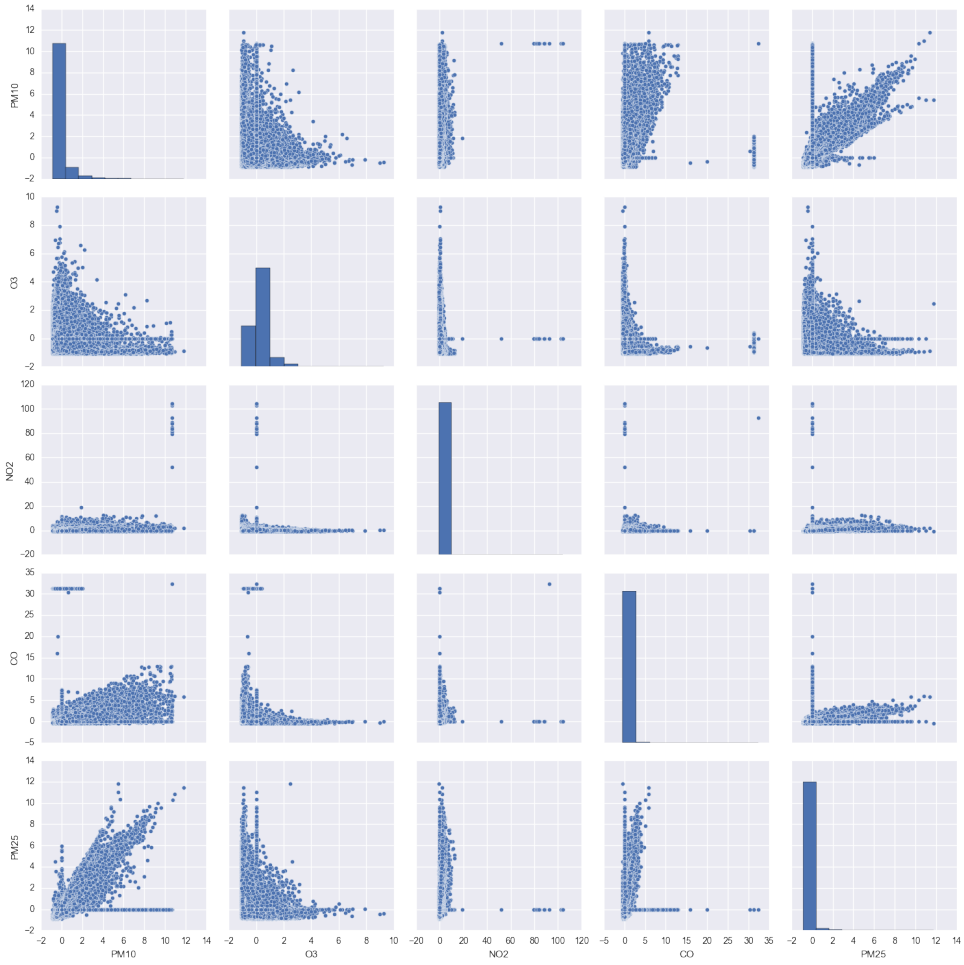
Figure 3.16: Feature set information

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 626500 entries, 0 to 626499
Data columns (total 19 columns):
date                  626500 non-null datetime64[ns]
PM10                  370787 non-null float64
NAME                  626500 non-null object
PM10_null_pointers    626500 non-null int64
CO                    352851 non-null float64
CO_null_pointers      626500 non-null int64
NO2                   222714 non-null float64
NO2_null_pointers     626500 non-null int64
O3                    291676 non-null float64
O3_null_pointers      626500 non-null int64
PM25                  79371 non-null float64
PM25_null_pointers    626500 non-null int64
time                  626500 non-null object
month                 626500 non-null int32
day                   626500 non-null int32
hour                  626500 non-null int64
daysInterval          626500 non-null timedelta64[ns]
days_interval         626500 non-null int64
hour_interval         626500 non-null int64
dtypes: datetime64[ns](1), float64(5), int32(2), int64(8), object(2), timedelta64[ns](1)
memory usage: 86.0+ MB
```

3.2.2 Covariance Plot

The covariance of two variables x and y in a data set measures how the two are linearly related. A positive covariance would indicate a positive linear relationship between the variables, and a negative covariance would indicate the opposite. Since we now have a huge data set consisting of PM10 values and other air pollutants as shown in the summary above, we would like to have a look at the correlation plot:

Figure 3.17: Pair plot for feature set



In the plot above we can observe visually a good correlation between concentration levels of PM10 , PM25 and CO. This fact can be exploited in cases when predicting for any of these pollutants and there are numerous missing values. We can separate the the missing value from the recorded values and try to use the model to predict accurately the missing value (e.g in trying to predict PM10 we separate Nan from recorded values). To the feature set we add the values from the other 2 pollutants (e.g add values from CO and PM25 concentration) and the shifted version of the pollutants (say PM10 for example) itself - we can take first and second difference. If there are any missing values in the other 2 (say CO and PM25, we can do local mean filling or just mean filling as discussed in the section on missing data mechanism). The missing values in the target (PM10) will be used for testing and be filled appropriately.

Provided below is the pair plot and joint plot of for PM10, PM25 and CO.

Figure 3.18: Pair plot for PM10,PM25 and C0

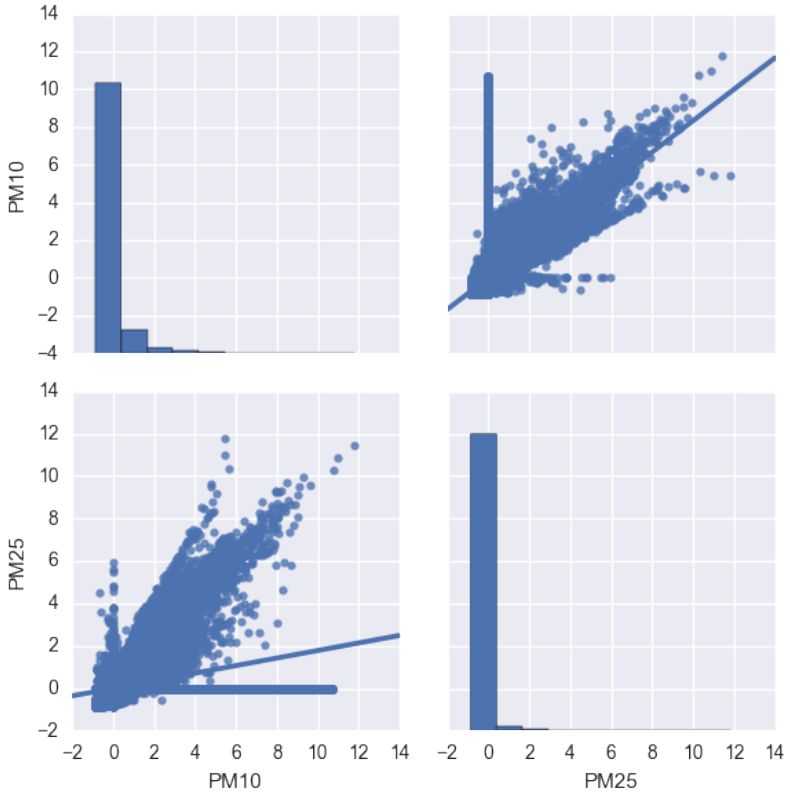


Figure 3.19: Joint plot PM25 and PM10

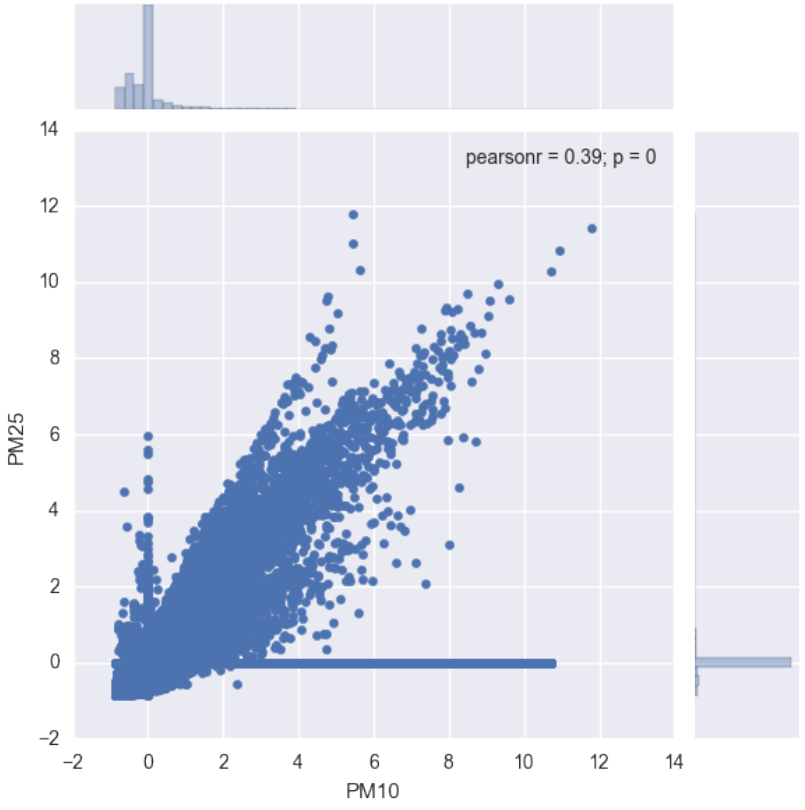
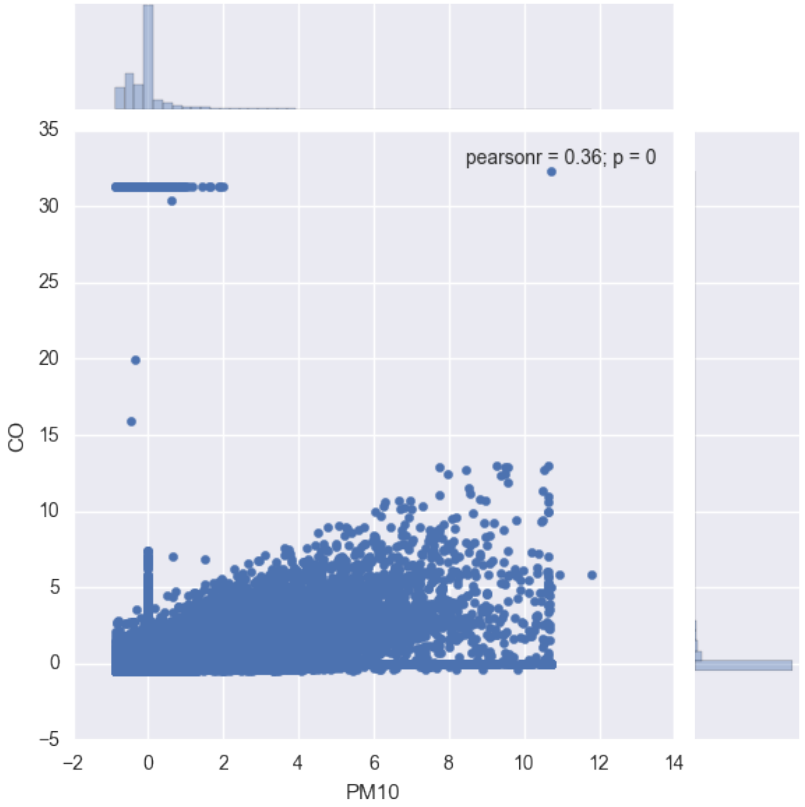


Figure 3.20: Joint Plot C0 and PM10



3.3 Conclusion

In this chapter we provided a concrete analysis of the data set up for training. Next we will use the data set in the LSTM model.

Chapter 4

Experimental Results and Model Analysis

4.1 BASE MODELS

In this section we discuss the base model that was used. And also report the accuracy on prediction that was observed.

4.1.1 ARIMA MODELS

ARIMA models are, in theory, the most general class of models for forecasting a time series which can be made to be stationary by differencing (if necessary).The ARIMA forecasting equation for a stationary time series is a linear (i.e., regression-type) equation in which the predictors consist of lags of the dependent variable and/or lags of the forecast errors. The predictors depend on the parameters (p, d, q) of the ARIMA model:

- Number of AR (Auto-Regressive) terms (p): AR terms are just lags of dependent variable. For instance if p is 5, the predictors for x(t) will be $x(t-1), \ldots, x(t-5)$.
- Number of MA (Moving Average) terms (q): MA terms are lagged forecast errors in prediction equation. For instance if q is 5, the predictors for x(t) will be$e(t-1), \ldots, e(t-5)$ where $e(i)$ is the difference between the moving average at ith instant and actual value.
- Number of Differences (d): These are the number of nonseasonal differences, i.e. in this case we took the first order difference. So either we can pass that variable

and put d=0 or pass the original variable and put d=1. Both will generate same results.

Figure 4.1: Moving Average Summary Statistics

ARIMA Model Results

Dep. Variable:	D.PM10	No. Observations:	89475
Model:	ARIMA(0, 1, 2)	Log Likelihood	-4766.115
Method:	css-mle	S.D. of innovations	0.255
Date:	Tue, 23 May 2017	AIC	9540.230
Time:	05:17:35	BIC	9577.837
Sample:	01-02-2007 - 06-02-2008	HQIC	9551.701

	coef	std err	z	P>|z|	[95.0% Conf.	Int.]
const	-7.028e-06	0.001	-0.009	0.993	-0.002	0.002
ma.L1.D.PM10	0.0035	0.003	1.031	0.302	-0.003	0.010
ma.L2.D.PM10	-0.1006	0.004	-27.686	0.000	-0.108	-0.094

Roots

	Real	Imaginary	Modulus	Frequency
MA.1	-3.1350	+0.0000j	3.1350	0.5000
MA.2	3.1699	+0.0000j	3.1699	0.0000

Figure 4.2: Auroregressive model Summary Statistics

ARIMA Model Results

Dep. Variable:	D.y	No. Observations:	89475
Model:	ARIMA(2, 1, 0)	Log Likelihood	-4797.924
Method:	css-mle	S.D. of innovations	0.255
Date:	Wed, 24 May 2017	AIC	9603.847
Time:	15:56:15	BIC	9641.454
Sample:	1	HQIC	9615.317

	coef	std err	z	P>|z|	[95.0% Conf.	Int.]
const	-7.091e-06	0.001	-0.009	0.993	-0.002	0.002
ar.L1.D.y	0.0185	0.003	5.548	0.000	0.012	0.025
ar.L2.D.y	-0.0892	0.003	-26.787	0.000	-0.096	-0.083

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	0.1035	-3.3467j	3.3483	-0.2451
AR.2	0.1035	+3.3467j	3.3483	0.2451

Figure 4.3: Arima Model Summary Statistics

ARIMA Model Results

Dep. Variable:	D.PM10	No. Observations:	89475
Model:	ARIMA(2, 1, 2)	Log Likelihood	-3045.742
Method:	css-mle	S.D. of innovations	0.250
Date:	Tue, 23 May 2017	AIC	6103.484
Time:	05:20:43	BIC	6159.895
Sample:	01-02-2007 - 06-02-2008	HQIC	6120.690

	coef	std err	z	P>|z|	[95.0% Conf.	Int.]
const	-5.816e-06	0.000	-0.030	0.976	-0.000	0.000
ar.L1.D.PM10	0.7295	0.036	20.443	0.000	0.660	0.799
ar.L2.D.PM10	0.0966	0.033	2.967	0.003	0.033	0.160
ma.L1.D.PM10	-0.7536	0.035	-21.374	0.000	-0.823	-0.685
ma.L2.D.PM10	-0.2058	0.035	-5.873	0.000	-0.274	-0.137

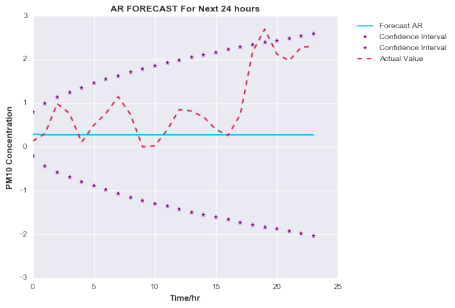
Roots

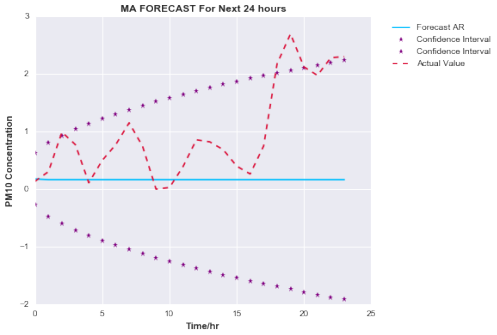
	Real	Imaginary	Modulus	Frequency
AR.1	1.1849	+0.0000j	1.1849	0.0000
AR.2	-8.7357	+0.0000j	8.7357	0.5000
MA.1	1.0346	+0.0000j	1.0346	0.0000
MA.2	-4.6971	+0.0000j	4.6971	0.5000

The summary of attribute from the output of AR,MA, and ARIMA models returns a significant amount of information, we will focus only on the table of coefficients. The coeficientf column shows the weight (i.e. importance) of each feature and how each

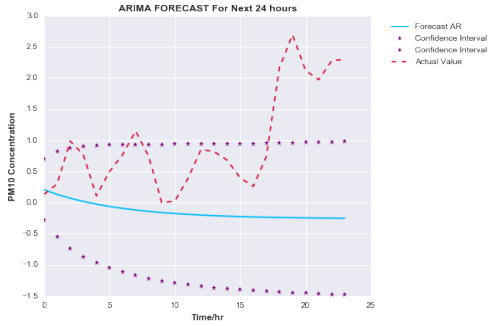
one impacts the time series. The $P > |z|$ column informs us of the significance of each feature weight. For each of the model we can accept the coefficient with p-values less than 0.05. Prediction with these models yield:

Figure 4.4: AR Model Forecast Plot





(a) MA prediction



(b) Arima Forecast

Table 4.1: Base Model And LSTM (Using only PPM10 as LSTM input)

Model	RMSE	R^2
MA	1.061809	0.0496
AR	1.139356	0.05798
ARIMA	1.11608	0.2330
$LSTM_{pm10}$	1.6754	0.39751

4.2 LSTM

In this section we give details about the LSTM models that were used for the prediction. The general approach here is to use a window of size 24 - that is it contains one value per time step for 24 time steps. At each training step we use the values in this window to predict the 25th value - which is the next time-step.

In the next training step we shift the window by 1 by dropping the first value in the window. We then add value 25th value of the feature set to to the window to make the window size back to 24. This *sliding window* effect is continued till the training set is exhausted.

The training data set was divided into "training and validation set" to reduce overfitting. For testing we actually emptied the wondow to make sure sure we are not learning from the previous time steps, and fill it with the previous values for 24 time step. Our task is then to predict the next 24 time steps

To do this at each iteration we predict the next value and add this same value to the prediction set while also dropping the first value from the window. This patterm continues until the we have predicted for 24 time steps: and hence the window becomes filled with our new prediction. The only exception to this pattern was when we use pollutants different from the Target pollutant in the feature set. Then we took the liberty to add real values of the other pollutants to predict the target. This is safe since the actual target pollutant is not in the feature set.

We included the target pollutant (PM10) in the feature set in some of the models. Since we assume we have access to the pollutant concentration up to a time period t then our aim is to predict $t + 24$ steps ahead. Often we will also add other relevant features to this pollutant concentration to model the get a more robust model. Our aim is to model the prediction capability given that there are missing values and that there are other feature sets that can aid the prediction power of the model.

4.2.1 Prediction Model With PM10 Input

The first attempt at the LSTM model we made was just to feed in the PM10 concentration and see how the model performs. A table of comparison of this model with the base model is shown in table4.1. A table of characteristics of the model is shown below:

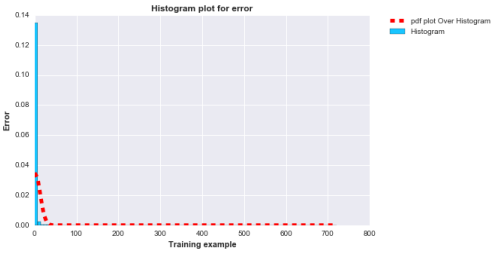
Table 4.2: LSTM model Summary Using only PM10 as input

Model	Architecture	Input	Nan Treatment	Performance Function	Learning rate
LSTM	24 *by* 24 LSTM cells	PM10	Forward filling and Median imputation	RMSE	Expoentially Decaying learning rate

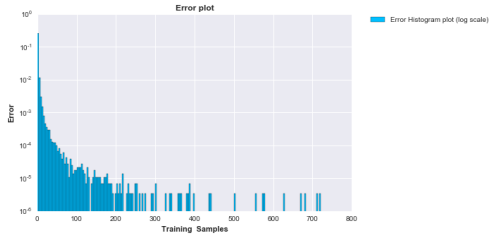
We also note that we used PM10 concentration from only the Region and Location of interest (Skopje and Recktorat). The model does well in the prediction for a horizon or time steps of about 10 after which it grossly fails to model well the PM10 behaviour. This can be attributed to the fact that each time we accumulate error in predicting the next time step. So that for far horizons the error term affects the prediction model.

The AR , ARIMA and MA models seem to have had a good low RMSE than LSTM but this can be due to the fact that we also had additional parameters (p, q, r) for the models which were absent in the LSTM model. A combination of LSTM and the shifted value would have also performed very well than the base models. Nevertheless we observe the LSTM model stated off well better than the base models. And it had a better R^2 value, which might mean it models the PM10 better than the base model.[1]

The RMSE plot obtained during train shows a a decreasing value as the training sample increases which is what we would want.

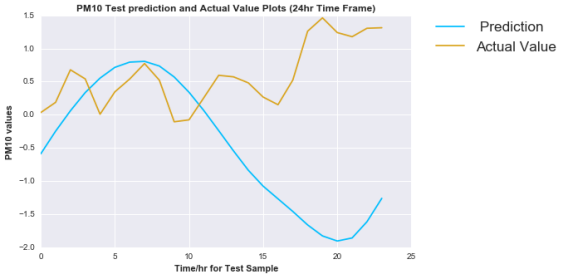


(c) Error Plot For LSTM with pdf



(d) RMSE plot for LSTM model with only PM10 as input

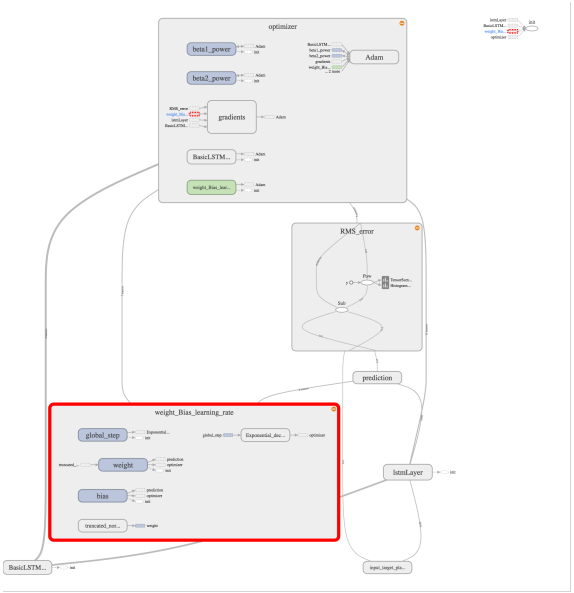
Figure 4.5: LSTM forecast After Training



The architecture of the model was generated using Tensorbiard and is shown below:

[1]There are controversies over the use of R^2 nevertheless for our we can assume it is still a good indicator of performance

Figure 4.6: LSTM Model Architecture Overview (Using Tensorboard)



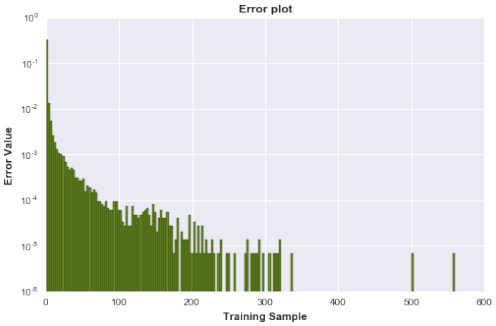
It was ensured that during training the model could learn from each training epoch. This is done by initializing each epoch with the output from previous epoch $initialstate(t+1) = previousstate(t)$. And to avoid overfitting we "zerorize" the initial state - that is we don't set it to the previous state.

4.2.1.1 Conclusion Drawn.

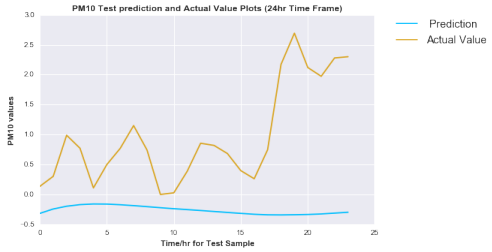
We already observe that the shifted PM10 concentration is stationary. This also explains why the AR, ARIMA and MA models. In situations where there are obvious missing values as in this study a combination of LSTM with input as Fist and secnd difference of the not-Nan values can serve as a better predictor for the Nan columns. This can aid model performance and accuracy.

4.2.2 Model Prediction with PM10 and Local Average Filling

The table of charateristics of this model is same as for the previous model except that Nan values were handled differently here (deletion andf local mean filling):*first, we delete regions of the data set with large gap of missing value - see figure 3.4 in chapter 3 - then at each training epoch we will the missing values with the local mean for each training window /batch.* This makes sense since the local average for each day would then be used to fill the missing values. The model was actually a way to showcase how we could handle missing values differently.

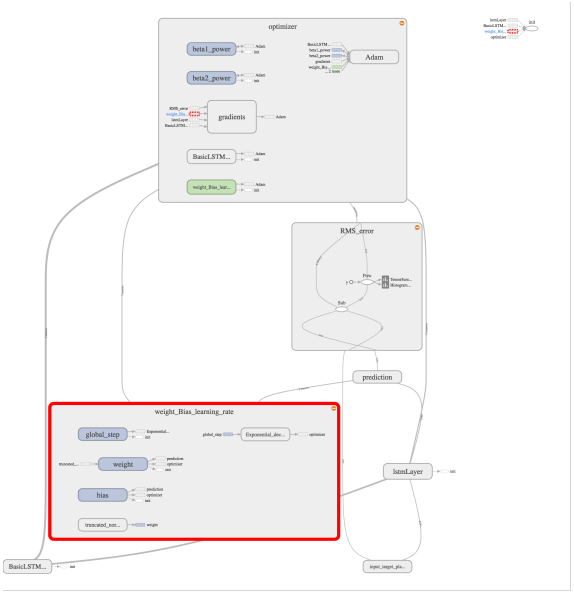


(a) RMSE for local mean filling model-red line is the pdf imposed on the RSME



(b) Prediction with local mean filling model

Figure 4.7: LSTM Model Architecture Overview (Using Tensorboard)



4.2.2.1 Conclusion Drawn

The model offers a new perspective to handle missing data during training. It steps from the intuition that values within a day should be much similar or at least have a genuine local mean more appropriate for filling the missing values. The model had an RMSE of 1.48916 for testing and R^2 values of 0.22108. Howver given more training time it and parameter tuning it can outperform the forward filling and mean filling model. This is because forward models constructed on this method actually outperforms other filling method as we shall see later.

In addition this model, can be used for missing value imputation and can be combined with the first or second difference.

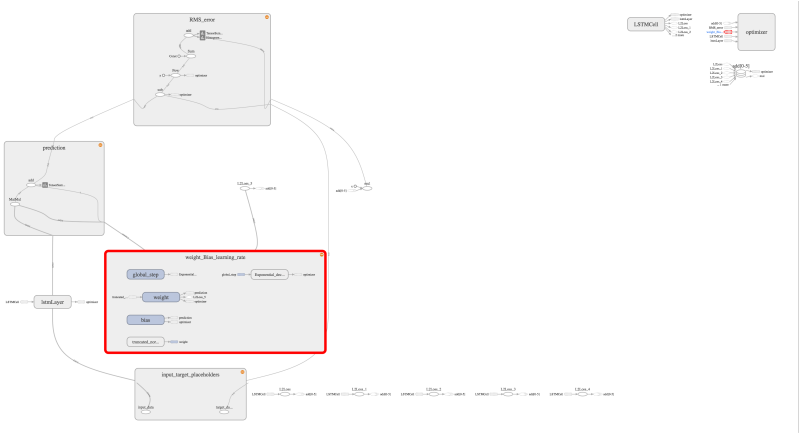
4.2.3 LSTM Model Plus Other Impurities as Input

As noted in chapter 3 we would like to know how the model prediction for one air pollutant can be improved when provide information about concentration of other pollutants in the same location[2]. This forms the motivation for this model.[3]. We used local median because the median is always less bias to outliers.

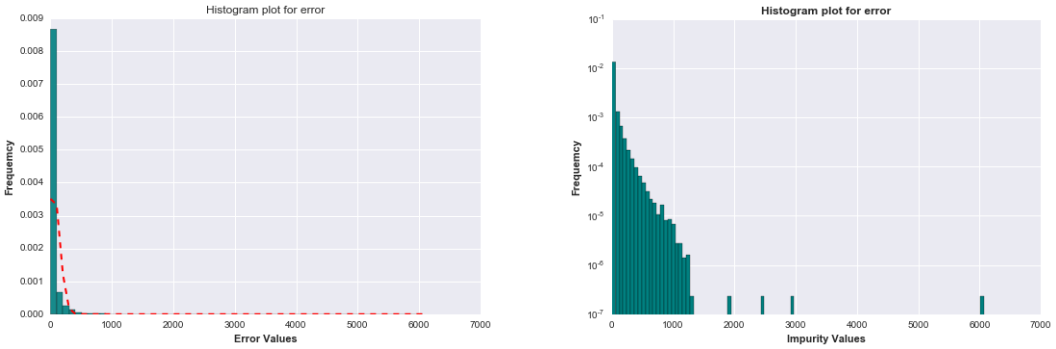
Table 4.3: LSTM with PM10 and other air pollutants as inputs

Model	Architecture	Input	Nan Treatment	Performance Function	Learning rate
LSTM	312 *by* 312 LSTM cells	PM10 and Other Pollutants Collected within same location	Deletion and Local median filling	RMSE	Expoentially Decaying learning rate with Regularizer

Figure 4.8: Model architecture from Tensorboard



[2]The pollutants included are $CO, NO2, O3, PM25$ within the Rektorat Location.
[3] Note that in this model we are not using pollutant concentration from other locations



(a) Training RMSE plot

(b) Prediction with model Using mask

Figure 4.9: Trainging Prediction plot

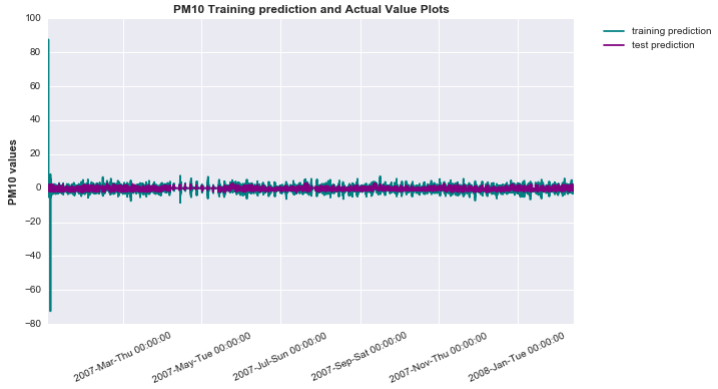
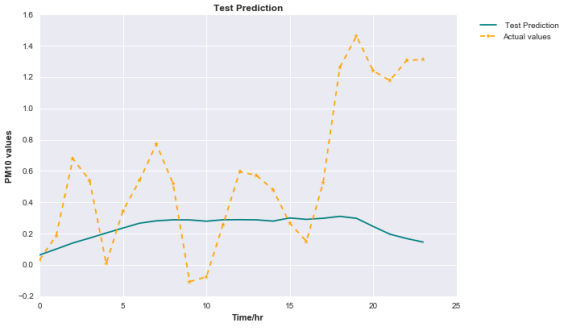


Figure 4.10: LSTM plus ther Impuritis with local mean filling



Model	RMSE	R^2
LSTM with Local Mean Filling and Other Pollutants	0.5790	$3.6626585469385197e-05$

4.2.3.1 LSTM Model Plus Other Pollutants Inputs Using Mask for Nan

This model is same as the previous one only difference again is how we handled missing values:*For each time step or training epoch we use a mask to label identify missing values. The input becomes the missing values indicator and the variable values*

This model has the advantage that we don't have to do any mean filling or forward filling as missing values are directly handled by the model using the indicators. The model outperforms the base model.

Figure 4.11: Trainging Prediction plot

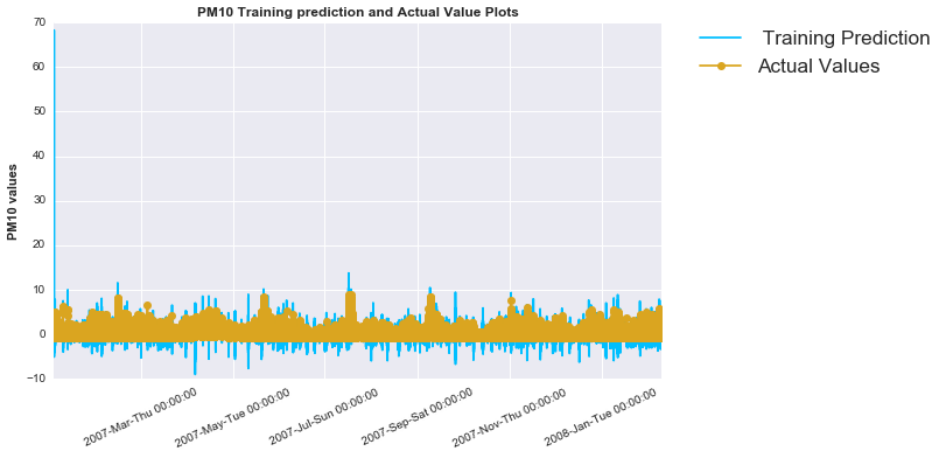
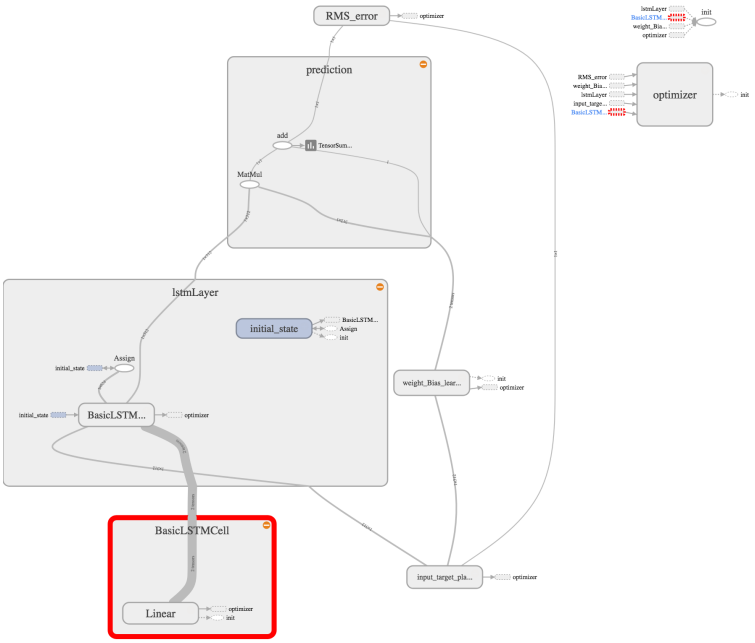
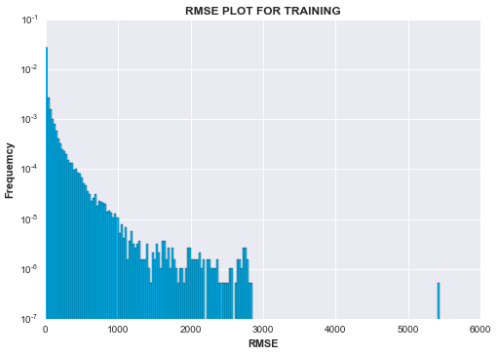
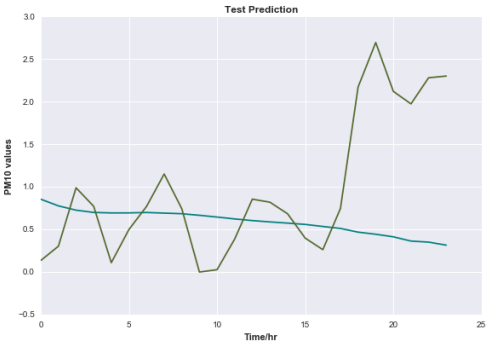


Figure 4.12: Model architecture from Tensorboard





(a) Training RMSE plot



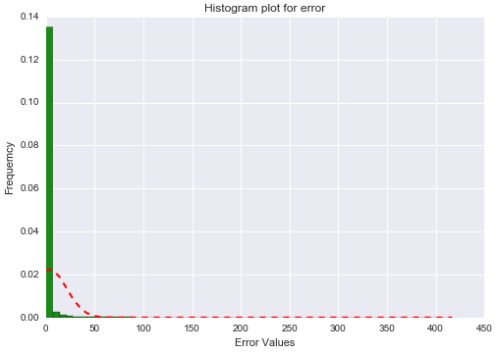
(b) Prediction with model Using mask

Table 4.4: Prediction Other Impurities and Mask

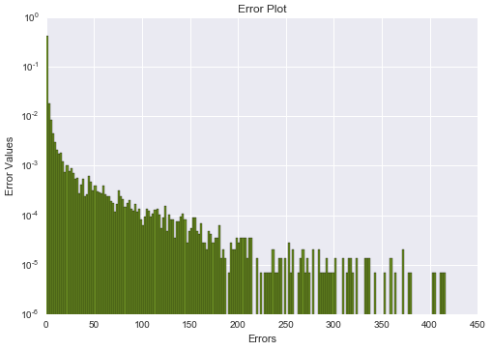
Model	RMSE	R^2
LSTM with MASK	0.994426473	0.613850

4.2.3.2 Missing Values Using Other Impurities

We will also use this section to discuss another method with which we could fill missing values. This involves using only the impurities collected as input and make the pollutant of interest as Target (in this project we generally used PM10). This was first investigated in this paper [26] using Artificial Neural Net. We expect LSTM to do better. The model generated gave the following RMSE plot:



(c) Training RMSE plot for PM10



(d) Prediction with model Using mask

We obtain an RMSE of 1.0260 and R^2 value of 0.136148. It outperforms the base models by a small margin.

Figure 4.13: Prediction plot Using other Impurities to predict PM10

4.2.3.3 Conclusion Drawn

We emphasize that in the entire discussion the aim has been to make a model that could assist in real world prediction of air pollutants concentration and stop or reduce the menace they cause. It is for this reason we have also included here a new approach - earlier we discussed an approach- with which we can deal with missing data and fill missing value more meaningfully.

We observe that both models perform better than the base model. It should also be noted that we use regularizer in this model to avoid overfitting. In LSTM using tensorflow care needs to the taken when defining the regularise: a name scope should be used that tags the weight parameters separately from the bias, since it is the weight terms that are to be regularized. In addition to this we tried to change the unit type from the tany layer to Relu and Relu6 but the model shows no improvement. In anyways the tanh unit seem to be the best. We also exploited the use of peep holes discussed in chapter 2 on LSTM. This only seem to give improvement when we have large input features.

4.2.4 LSTM Model Using Other Locations Plus Impurities From Other Locations

Up to this moement we have only used featuire set from the Rektorat location of the Skopje region. Now we exploit how the model behave when we add input features from other locations. The LSTM description is provided below:

Table 4.5: LSTM with PM10 and other air pollutants as inputs

Model	Architecture	Input	Nan Treatment	Performance Function	Learning rate
LSTM	1032 *by* 1032 LSTM cells	PM10 and Other Pollutants Collected in Other locations	Deletion and Local median filling	RMSE	Expoentially Decaying learning rate with Regularizer

Figure 4.14: RMSE for training set

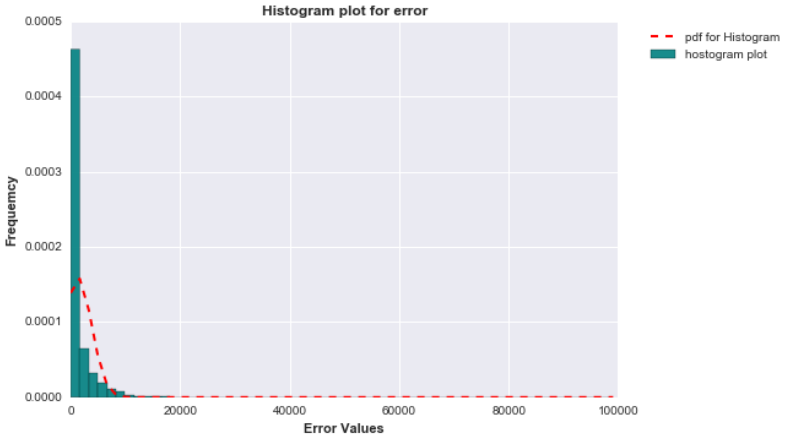


Figure 4.15: Prediction plot for model

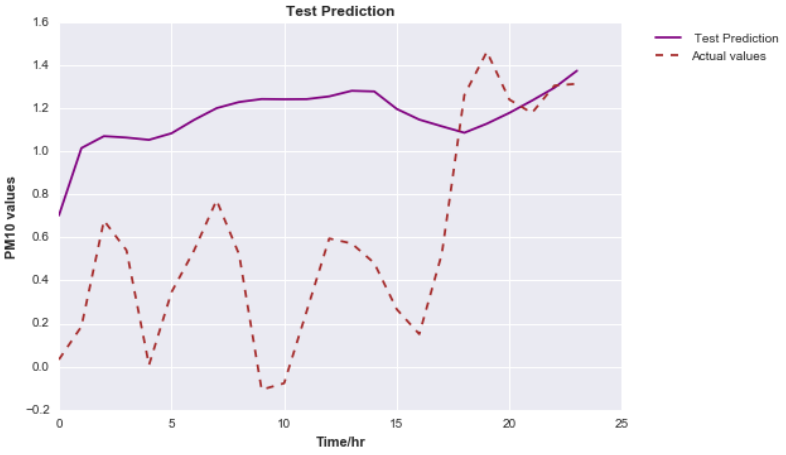
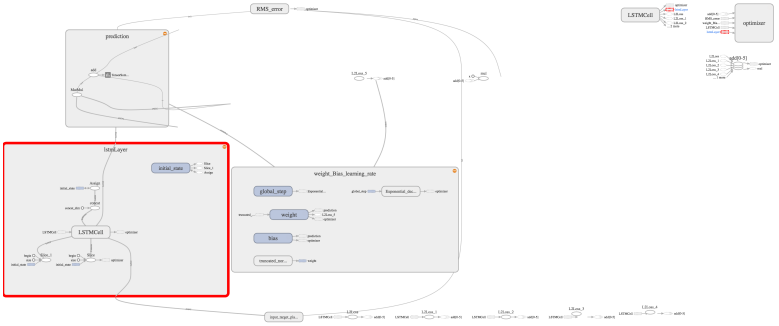


Figure 4.16: LSTM Graph from Tensorboard



4.2.4.1 Conclusion Drawn

We can that though the RMSE value is lower than than the model with only other impurities reported from same locations above, the model actually was able to model the dynamics of the variation of PM10 accurately. We observe that even in after like 20 horizons ahead, the model was still able to follow the curvature of the PM10 concentration. We should note that the model was ran just once and there was no extensive parameter tuning (this model was run a MacBook Pro, 2009 version). With large computational power and parameter tunning this model will of course outperform the previous model. Moreover it as better R^2 value. In real practical application of the model, the margin of error in the tail prediction for this model is quite low and it can be trusted. For the last model less confidence will be given to tail predictions

4.2.5 Weather and Pollutant prediction

The Concentration of air pollutants is influenced by meteorological conditions. We identified five features: temperature, humidity, baometer, wind speed and weather (such as cloudy , foggy , suuny and speed) . Figure below shows the correlation between PM10 and the first four features. using the data collected from August to Dec 2012 in Beijing where each row/column denote one feature and a plot means the AQI level of the location.

In particular, a high wind speed disperses the concentration of PM10 and high humidity usually cause high concentration. A high pressure will result in a good AQI. The impact of temperature is not clear in this case , but a good AQI is more likely when temperature is high and humidity is low , or when pressure is high and temperature is low.

Should we consider: Traffic related features Human Mobility features Road network related features Point of interest related features

Chapter 5

Model Improvement And new Model Proposal

This chapter serves two purposes. First it presents ways by which the extant model can be improved - which can be another project itself. Secondly, I present the air quality prediction from a different perspective and then propose how we can build a new model based on this.

5.1 Mixture of Models

Up to this point we have focused on modelling the dynamics of the air pollutant using lSTM as our function approximator. Now, we will ask the question, can we actually do better? what if we combine this seemingly different models we have modelled to make a new model. But then we are faced with the problem as to whether such *mixture of expert* models as they are called are can be better than using a single function approximator as we have done so far. How do we mix models? We can mix the models in the following ways[27]:

- Average over many different models. (for example Gaussian process, LSTM, ARIMA)
- Use LSTM architecture but average over predictions made by many different weight vectors

An example would be to combine model "using only PM10 to predict PM10", "PM10 plus it differenced or shifted values" and predictions from "PM10 using only Other

impurities". This combine model can help fill the vast missing values and the output can be fed for another combined model that makes future predictions.

How would this combined predictor be better? On any test case some individual predictors may perform better than the combined predictor. However if the individual predictors disagree a lot, the combined predictor is typically better than all the individual predictors when we average over test cases. Then our goal would be to make the individual predictor disagree without making them worse individually.

This also boils down to the bias and variance terms of the RMSE used in the model. The squared error can be conveniently decomposed into bias and variance terms. The bias term is big if the model has too little capacity to fit the data. On the other hand the variance term is big if the model has so much capacity that it is good at fitting the the sampling error in each training data. Mathematically we express this as follows:

$$\hat{y} = \mathbb{E}(y_i) = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Where i represent the number of models and y_i is the prediction of each model. This means expression above reads that the combined model prediction is now the average of the prediction from each individual model[27]. Then we define the squared error as follows:

$$\begin{aligned}\mathbb{E}_i[(t-y_i)]^2 &= \mathbb{E}_i[(t-\hat{y})-(y_i-\hat{y})]^2 \\ &= \mathbb{E}_i[(t-\hat{y})^2-(y_i-\hat{y})^2-2(t-\hat{y})(y_i-\hat{y})] \\ &= (t-\hat{y})^2+\mathbb{E}_i(y_i-\hat{y})^2-2(t-\hat{y})\mathbb{E}_i(y_i-\hat{y})\end{aligned} \tag{5.1}$$

The first term is the error we get by comparing the combined prediction model(average of model) and the target. The second term is the variance of the y_i - expected squared difference between y_i and $\hat{i}$. The last term vanishes because we are multiplying two terms that have zero mean and uncorrelated; so that we expect to get zero on average. In essence the expected error we get by picking a model at random is greater than the expected error of the combined model by the *variance of the output of the models*.

We stated earlier that we should aim towards making individual model in the combined architecture differ but not perform worse. This seem to be a contradictory requirement. How can we hope to achieve this?

- By using different number of hidden layers

- Different number number of units per layer (ReLu, tanh)[1]
- Different types of units per layer

5.2 A new proposed Model

I here present what I think is my own perspective for the reformulation of the problem. The model I proposed here would not be something new but a combinaton of ideas from *Hidden Markov models* and *Partially Observable Markov Decision process*. First we will revise ourselves what the AQI index is.

AIr Quality Index (AQI) is a number used by Government agencies to communicate to the public how polluted the air is currently is. The function used to convert from air pollutant concentration to to AQI varies by pollutants and is different from countries to countries. Air quality index values are divided into ranges and each range is assigned a descriptor and a colour code. We will use a standard issued by United States Environmental Protection Agency as shown in table below:

Table 5.1: AQI table

AQI Values	Levels of Health Concern	Colors
0 – 50	Good(G)	Greem
51 – 100	Moderate(M)	Yellow
101-150	Unhealthy for sensitive groups (U-S)	Orange
151 – 200	Unhealthy (U)	Red
201 – 300	Very Unhealthy (VU)	Purple
301 – 500	Harzadous	Maroon

A *Hidden Markov* model is same as *Markov model*. The only difference is in whether or not we can observe the current state of the process. In a Hidden Markov models we add a set of observations to the model. To account for the fact that we cant observe the states directly, we add a set of observations to the model. So instead of directly observing the current state, the state gives us an observation which provides a

[1] see appendix to view different types of units that are possible

hint about what state it is in. The observations can be probabilistic; so we need to also specify the observation model. This observation model simply tells us the probability of each observation for each state in the model.

More formally, a Hidden Markov model is a doubly embedded stochastic process with an underlying stochastic processes that is not observable (hidden) but can only be observed through another set of stochastic processes that produce the sequence of observation[28].

5.2.1 Model Introduction

We assume that the AQI levels are the hidden states of our hidden Markov model. However at each time step the random process furnishes an observation, which are the concentration levels that are being recorded. Then we reformulate the problem as thus:

Given an observation of air pollutant concentration how do we choose a corresponding sate sequence which is optimal in some meaningful sense (ie best explains the observations)? Given a sequence of observation of air pollutant concentration how can we predict the next 24 hour per time step pollutant concentration ? In essence we want a mapping from observation to the underlying state. *How do we generate adjust and find optimal parameter for the model*

5.2.1.1 Elements of the model

The HMM specification of the prediction of air pollutant concentration has 5 elements:

- Number of states $S \in S_1, S_2, S_3, S_4, S_5, S_6, S_7$ corresponding to the AQI level.
- The Number of observation per state. In this model it is the set positive real values of pollutant concentration we can observe denoted V.
- The transition probility of S, denoted π
- Transition probability in V denoted p
- Initial probability π

We will lump V, p, π together and denote it as λ.

$$\lambda = (V, p, \pi)$$

This is the model parameter.

5.2.1.2 Model Prediction

Tackling the problems formulated earlier directly by seem daunting. The classical way to solve this is to keep a complete history of the trajectory of the state-observation sequence. However, this can be inefficient. Moreover since we have a continuous observation, there might be need to discretize the obervation space. Fortunately, it turns out that simply maintaining a probability distribution over all of the states provides us with the same information as if we maintained the complete history.

We will refer to a probability distribution over states as a *belief state* $\mathbb{B}$ and the entire probability space (the set of all possible probability distributions) as the *belief space*.

5.2.1.3 Model Solution

- Step 1, we already have a set of data set that has the continuous sequence of observations. We will then estimate the belief state over these observations - there are already standard equations that transforms ppm to AQI equivalent. We no have have the sequence of probaility distribution over state. Using LSTM, we feed this belief state estimates to the neural net and aim to stochatic forecast the dynamics of future beilief state.
- Since the Neural networks and discriminant function approximators, step 1 should be fairly straight forward. However the output of the model, should now be treated as such.
- Given a sequence of belief state (output of the lstm neural nets), we can choose a sequence of states (AQI levels) which is optimal in some meaningful sense using the veterbi algorithm [28]
- We can optimize model parameters,λ, using Expectation maximization algorithm .[28]
- Lastly Given a set of belief state, we can compute the probability of observing the sequence using Forward-Backward Algorthm.

5.3 Conclusion

In this chapter we have suggested an improvements to the model. In addition we suggested a new perspective of interpreting or formulating the problem. The perspective

can be more interpretable and more robust. It is hoped that someone else will find this improvement interesting and try to work on them.

Appendix A

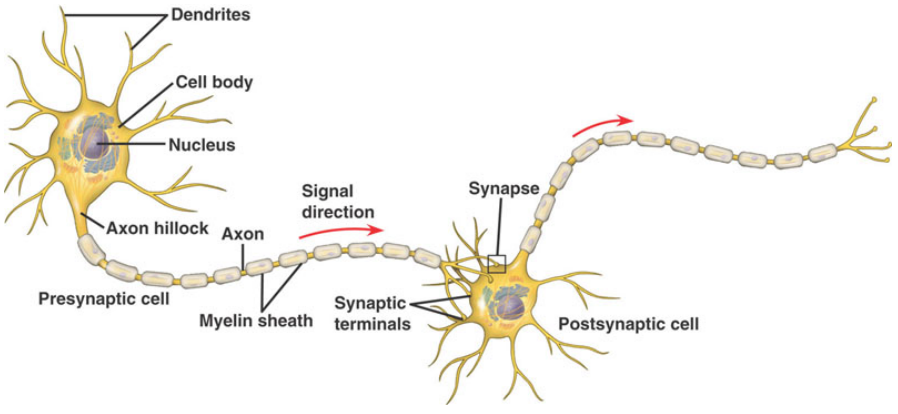
Appendix : Gentle Introduction to LSTM

A.1 Neural Nets: Biologically inspired

Inspiration for this appendix comes from lecture note by Prof Patrick Henry Winson of the MIT, Artificial Intelligence Laboratory [29] .

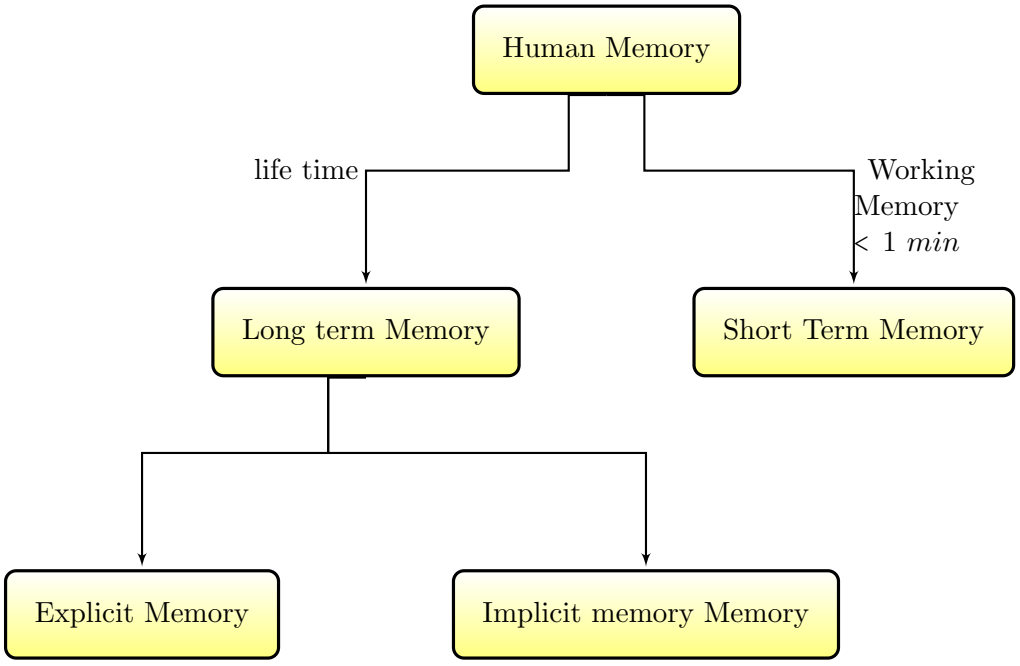
Our **memory** gives us the ability to store, retain and recall information and experiences. In more physiological or neurological terms, memory is, at its simplest, a set of encoded neural connections in the brain. It is the recreation or reconstruction of past experiences by the synchronous firing of neurons that were involved in the original experience.The figure A.1 shows a simple neuron. Millions of these building blocks lie in our brains.

Figure A.1: Simple neuron chain



The **short term memory** or the *Working Memory* act as a scratch pad for temporary recall of information which is being processed at any point in time. It has been dubbed the **the brain's post-it note**.

Figure A.2: Simple Human Memory Chart



The **Long term memory** gives us the ability to store information for long period of time. It is divided into memory that can be consciously recalled (explicit) and those that can be recalled without any effort (implicit).

Our task or the task of neural nets is to come up with a **computational units** that can model this memory taxonomy in the human's brain. And at the same time endow those with some capabilities and functions that the different memory classes and sub-classes perform . We will come to see that **Recurrent Neural nets** are the computational mimicry of **Short term memory** and **Long Short Term memory** are computational mimicry of **Long term memory**.

A.1.1 Computational neural building blocks

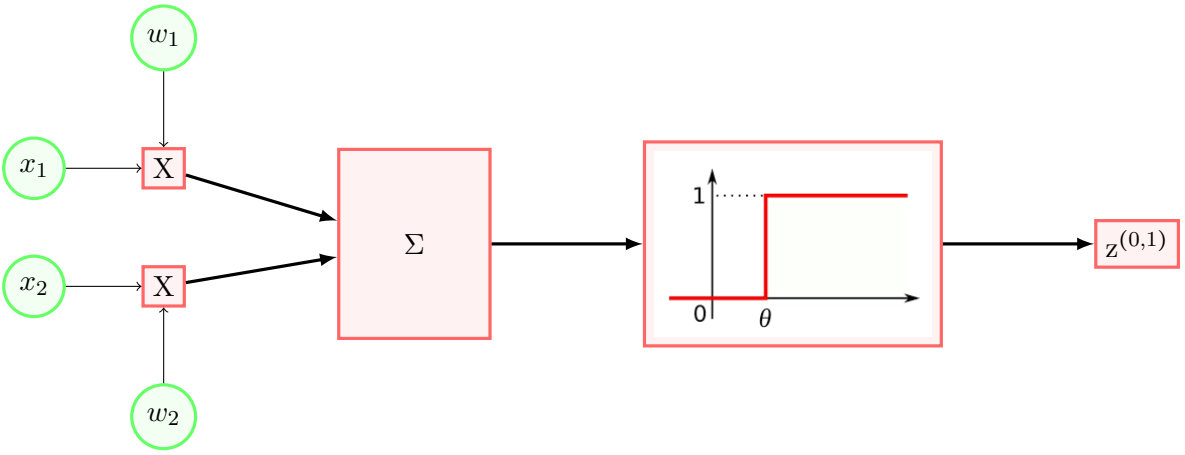
Fig A.2 shows a neuron chain: the building block of the brain that enables our memory. How can we model this building block?

First let us explain the neuron figA.1, note that the **synapse** is the region o between an axon of a neuron and the dendrite of another as labelled. The signal direction in red goes down from cell body to the dendrite of the other cell. The other neuron either get fired (termed *Firing of next neurone*) if the signal strength of the

incoming signal is above a **threshold** θ, otherwise signal dies away- as it is not strong enough to excite the next neurone.

Then we proceed to model this like below:

Figure A.3: Simple neuron model



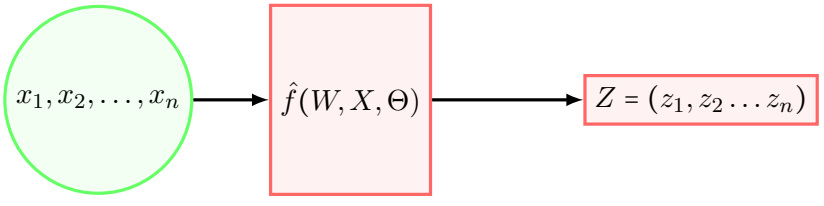
To explain this model we let input x_1 - for simplicity we can assume binary input- be multiuplied by weight w_1. By this we model the *synaptic connection*. If the strength of the incoming signal x_1 is high the *weight* w_1 goes up and vice versa. The same argument holds for x_2 and weight w_2.

The Σ is the "summer". It represents a way to measure the collective influence of both x_1 and x_2 weights given the weights. This we can think of as "confluence" of neurons trying to fire another neuron.

The we have to decide if the sum of these inputs is sufficient enough to make the neuron fire. We do this with the step function as depicted in the diagram. Note the threshold θ implying that if the sum is greater than the threshold the neuron is excited otherwise nothing happens (*all or none* effect as in typical organism's neurons).

What happens if we colection of these neural models:

Figure A.4: Neural Model for Mutiple inputs



In this clean diagram, we have captured the whole essence of neural nets. It is a *function approximator*. The output vector(Z) is a function of the weights(W), inputs(X),

Thresholds vector(Θ). We formalize this and we write:

$$Z = \hat{f}(W, X, \Theta)$$

When we train neural nets our aim we basically adjust these parameters so that it get as close as possible to the real or desired output , $\mathcal{G}$.

A.1.2 Performance Function

How do we gauge how well our approximator Z is doing compared to the real output $\mathcal{G}$? To gauge this we use the performance function defined as:

$$\mathcal{L} = -\frac{1}{N}|Z - G|^2$$

where N is the Normalization factor. Also note the negative(-)sign attached to the cost function. With this, we aim to minimize the squared difference between Z and $\mathcal{G}$ - zero being the best. The plot for two-paramater $\mathcal{L}$ is shown below. As the

Figure A.5: Simple Performance function (2 parameters)

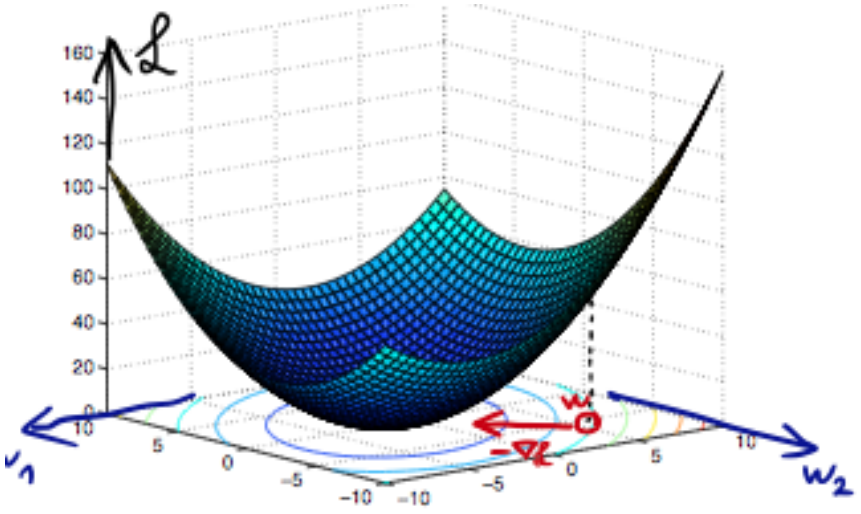


figure shows we are trying to adjust our weight parameters to maximise performance. The red arrow shows that we are trying to take "steps-called *Hill climbing*" each time towards the weights combinations that gives minimizes the *"error"* - or maximise the *performance*.

What we do because we can't do hill climbing each for a large parmaters- this explodes exponentially and is intractable. Instead we take some *"Partial derivative"* and *follow the gradient*-Note that if we remove the negative sign of the cost function we

take step in opposize direction of the gradient.

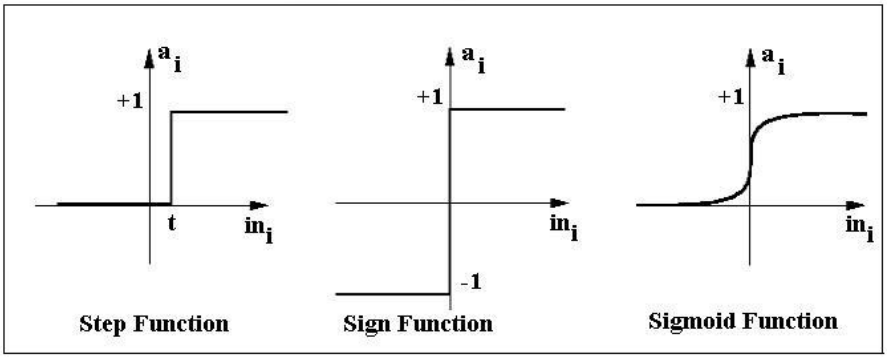
$$\Delta\mathcal{L} = \gamma(\frac{\delta\mathcal{L}}{w_1}i + \frac{\delta\mathcal{L}}{w_2}j)$$

The equation depicts how much improvement we get by making movement along w_1 and w_2. γ is the step size - how much we move each time. However since we are taking gradient, we make 2 modifications to the diagram above.

- Replace the *step function* with *Sigmoid function.*
- Sigmoid function is continuous and has simple derivative - a good property since we will be taking gradient. The step function is discontinuaous.
- We then add a *bias term* so thatwe can get rid of the *Threshold.* Thus we add this as w_0 to weight parameters

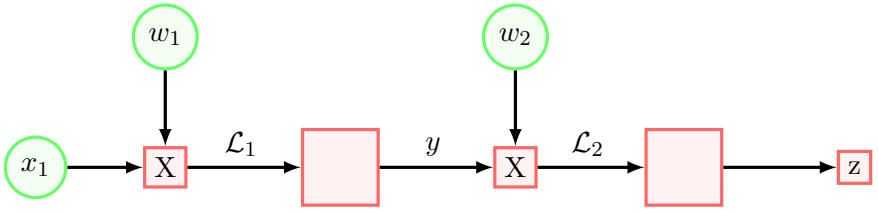
As the figure below shows, we add the bias parameter to get sign function and we change the sign function to Sigmoid to compute gradient easily.

Figure A.6: From Step Function to Sigmoid



A.1.3 Back Propagation

Figure A.7: Simplest Neural net



From the diagram above, we can manually use the "back propagation" to compute the gradients along the network. We do use this using the simple chain rule . Our aim is to compute the partial the performance function.

$$\frac{\delta \mathcal{L}}{w_2} = \frac{\delta \mathcal{L}}{\delta z}\frac{\delta z}{\delta w_2} \tag{A.1}$$

$$\frac{\delta \mathcal{L}}{\delta w_2} = \frac{\delta \mathcal{L}}{\delta z}\frac{\delta z}{\mathcal{L}_2}\frac{\mathcal{L}_2}{w_2} \tag{A.2}$$

$$\tag{A.3}$$

The equations above can be derived by simple following the gradient along the network. We also compute for $w1$:

$$\frac{\delta \mathcal{L}}{\delta w_1} = \frac{\delta \mathcal{L}}{\delta z}\frac{\delta z}{\delta w_1} \tag{A.4}$$

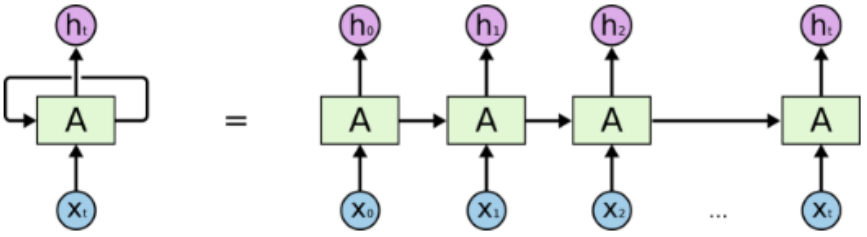
$$\frac{\delta \mathcal{L}}{\delta w_1} = \frac{\delta \mathcal{L}}{\delta z}\frac{\delta z}{\delta \mathcal{L}_2}\frac{\delta \mathcal{L}_2}{\delta y}\frac{\delta y}{\delta \mathcal{L}_1}\frac{\delta \mathcal{L}_1}{\delta w_1} \tag{A.5}$$

From the simple network we can build complex neural nets. We now turn to RNN and LSTM.

A.2 RNN And LSTM

We model s**short term memory** using **recurrent neural net (RNN)**. A recurrent neural network in it simplest form can be described as multiple copies of the simple network described in previous section , each passing a message to a successor.

Figure A.8: RNN unenrolled



An unrolled recurrent neural network.

From Colah's blog http://colah.github.io/posts/2015-08-Understanding-LSTMs/

However, if we unenroll the network for a large time step behind and try doing the back propagation computed in previous section we observe one draw back:which is down as we do very far back in time we might incur **vanishing** or **exploding gradient**. This implies we might not be learning the right dependency available in previous time

steps for **vanishing gradient** or we might just be learning something completely irrelevant or killing other weights that might also be important for performance (**exploding gradient**).
How do we solve these two problems

- Solution to Exploding gradient
  - Truncated Back propagation through Time
  - Clipping gradient at threshold
  - RMSprop(adaptive learning rate) to adjust learning rate
- Solution to Vanishing gradient
  - Harder to track
  - weight initialization
  - ReLu activation function
  - RMSprop (oprimization algorithm)
  - LSTM and GRU

A.2.1 LSTM- Long Term Memory Mimicry

The way LSTM solves this problem is to introduce a memory unit called **cell unit** into the neural architecture. See Figure A.9. Though it might seem difficult to track,there is a simple intuition behind what is going on. First,there are now 3 inputs into the network at each time step:

- x_t is the input of current time step
- h_{t-1} is the input from the previous time step - hence it is the output of the previous time step.
- c_{t-1} is the input of the previous cell unit.

Therefore this single unit makes decision by taking into consideration these 3 different inputs. After this , it gives out h_t which is the output of its own (current) time step and $c_t$is the output of the (current) its own cell unit. Concatenating these single cell unit like we did in the simple neural model we get the fig A.2:

The way the internal memory, c_t ,is updated over time is similar to water flow in pipe system. If we assume that memory is the water flow along the pipe. We might

Figure A.9: LSTM cell

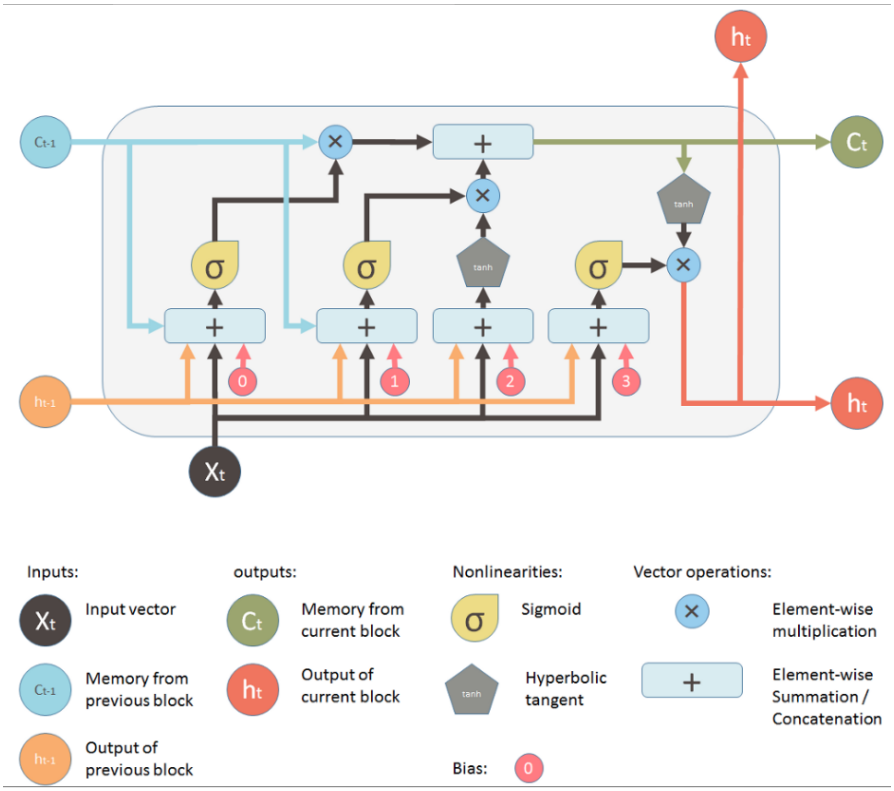


Image from
https://medium.com/@shiyan/understanding-lstm-and-its-diagrams-37e2f46f1714

Figure A.10: LSTM Neural net

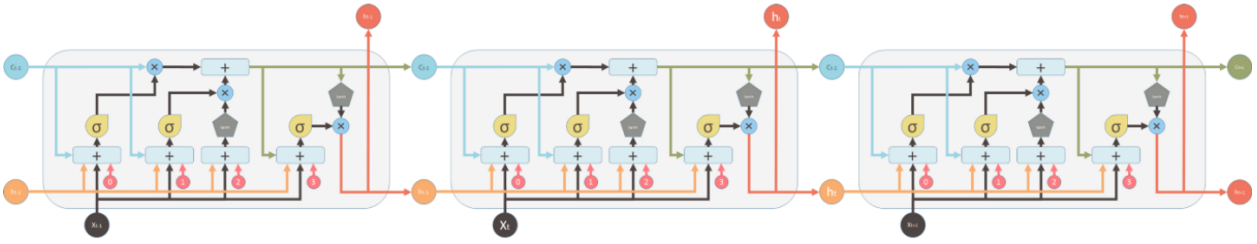


Image from
https://medium.com/@shiyan/understanding-lstm-and-its-diagrams-37e2f46f1714

want to change or adjust the flow . This can be effected using *valves*. One valve will be the *forget valve*. When shut , no old memory will be kept. If you fully open this valve, all old memory will pass through. The second valve is the new memory valve where new input gets in.

A.2.2 Analysing the LSTM Cell

Looking at the LSTM diagram in figA.11, the top pipe is the memory pipe. The input is the old memory c_{t-1} (a vector). The first cross $\times$ it passes through is the *forget valve*. It is actually an element-wise multiplication operation. So if you multiply the old memory c_{t-1} with a vector that is close to 0, that means you want to forget most of the old memory. If your forget valve equals 1, everything passes through. Then the second

Figure A.11: LSTM Neural cell

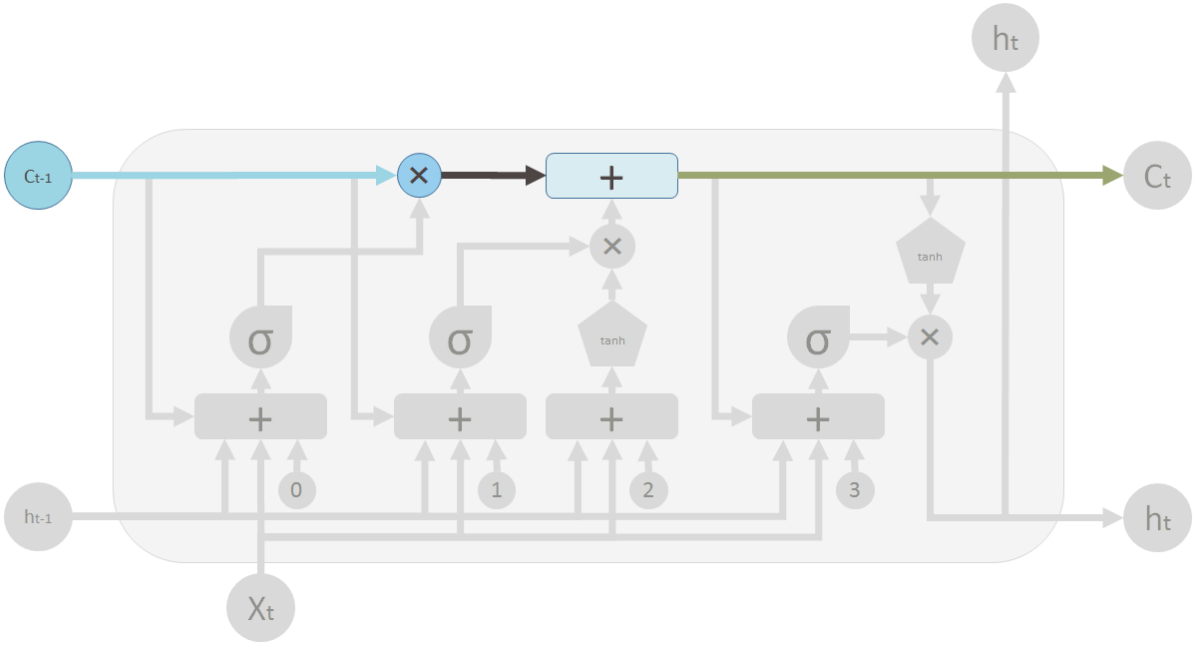


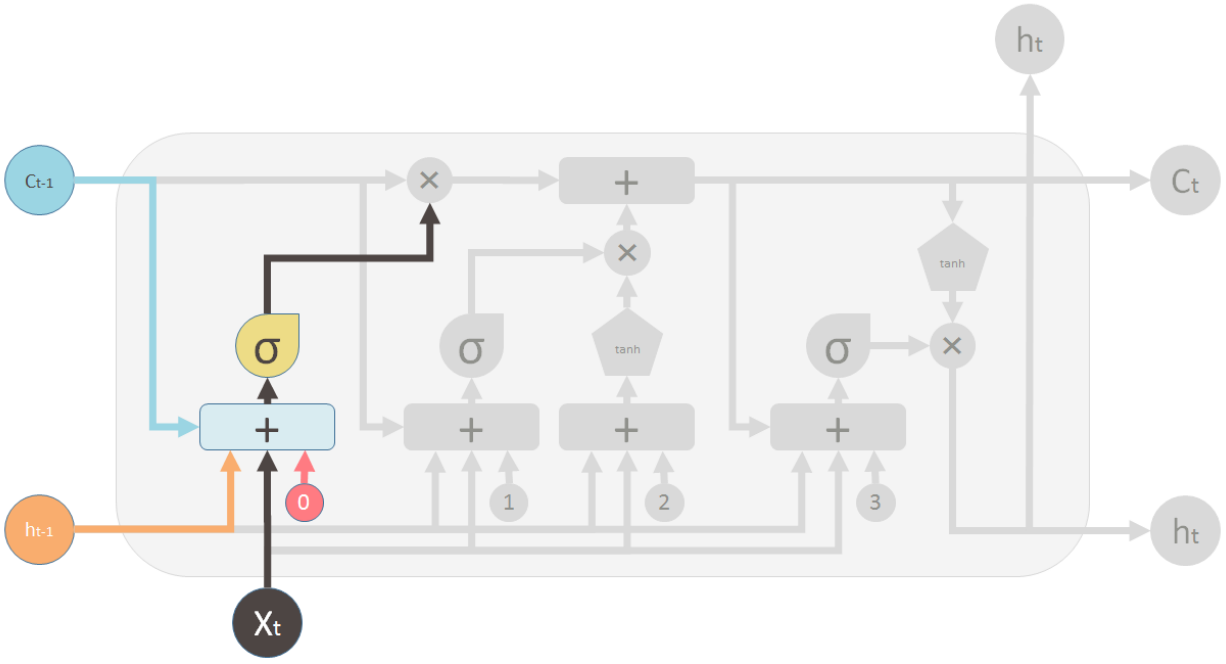
Image from
https://medium.com/@shiyan/understanding-lstm-and-its-diagrams-37e2f46f1714

operation the memory flow will go through is this $+$ operator. This operator means piece-wise summation. How much new memory should be added to the old memory is controlled by another valve, the $\times$ below the $+$ sign.

After these two operations, we get transform the old memory C_{t-1} changed to the new memory state c_t.

The valves of LSTM Cell: The first valve is called the *forget valve*. It is controlled by a simple one layer neural network. The inputs of the neural network is h_{t-1}(the output of the previous LSTM block), x_t(the input for the current LSTM block) c_{t-1} (the memory of the previous block) and finally a bias vector b_0. This neural network has a sigmoid function as activation, and its output vector is the *forget valve*, which will applied to the old memory c_{t-1} by element-wise multiplication. This is shown fig A.12: The second valve is the new memory valve. Again, it is a one layer simple neural network

Figure A.12: LSTM: First forget valve



Forget Valve .Image from
https://medium.com/@shiyan/understanding-lstm-and-its-diagrams-37e2f46f1714.

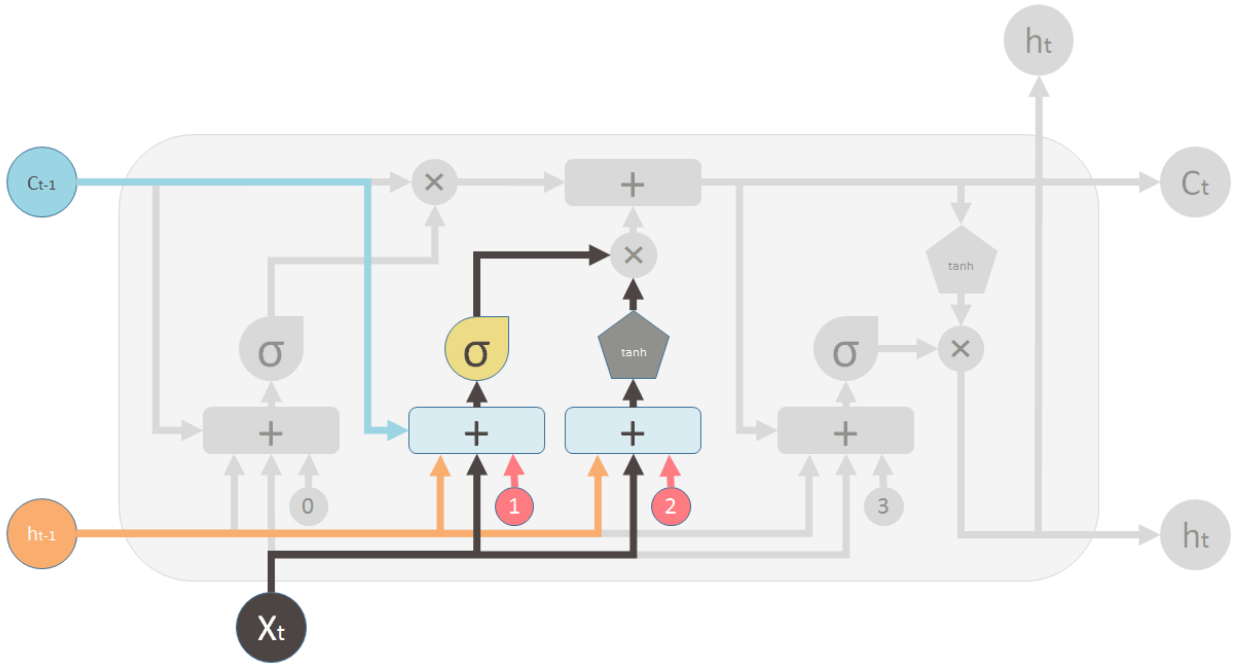
that takes the same inputs as the forget valve. This valve controls how much the new memory should influence the old memory. This is shown in fig A.13:

A.2.3 Output from the cell

The *new memory itself* c_t , is generated by another neural network. It is also a one layer network, but uses tanh as the activation function. The output of this network will element-wise multiple the new memory valve, and add to the old memory to form the new memory as in fig A.14.

And finally, we need to generate the output for this LSTM unit h_t. This step has an output valve that is controlled by the new memory, the previous output h_{t-1}, the input x_t and a bias vector. This valve controls how much new memory should output to the next LSTM unit.

Figure A.13: LSTM: New Memory valve



New memory pipeline .Image from
https://medium.com/@shiyan/understanding-lstm-and-its-diagrams-37e2f46f1714

Figure A.14: LSTM cell state output

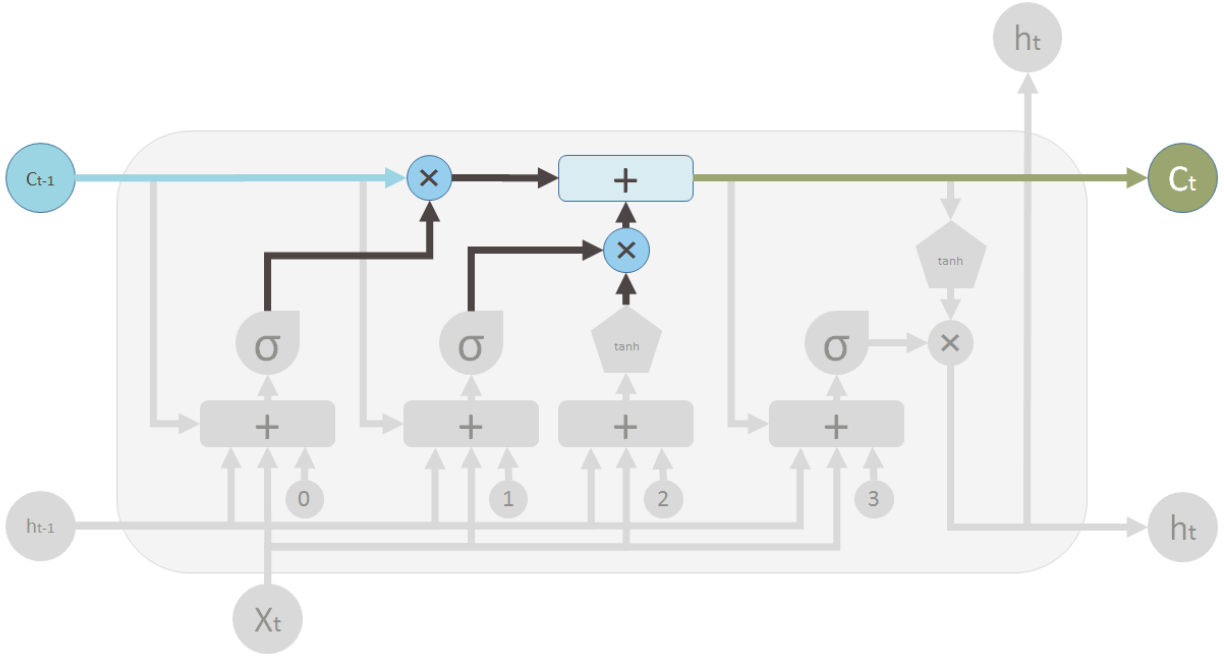


Image from
https://medium.com/@shiyan/understanding-lstm-and-its-diagrams-37e2f46f1714

Figure A.15: LSTM output

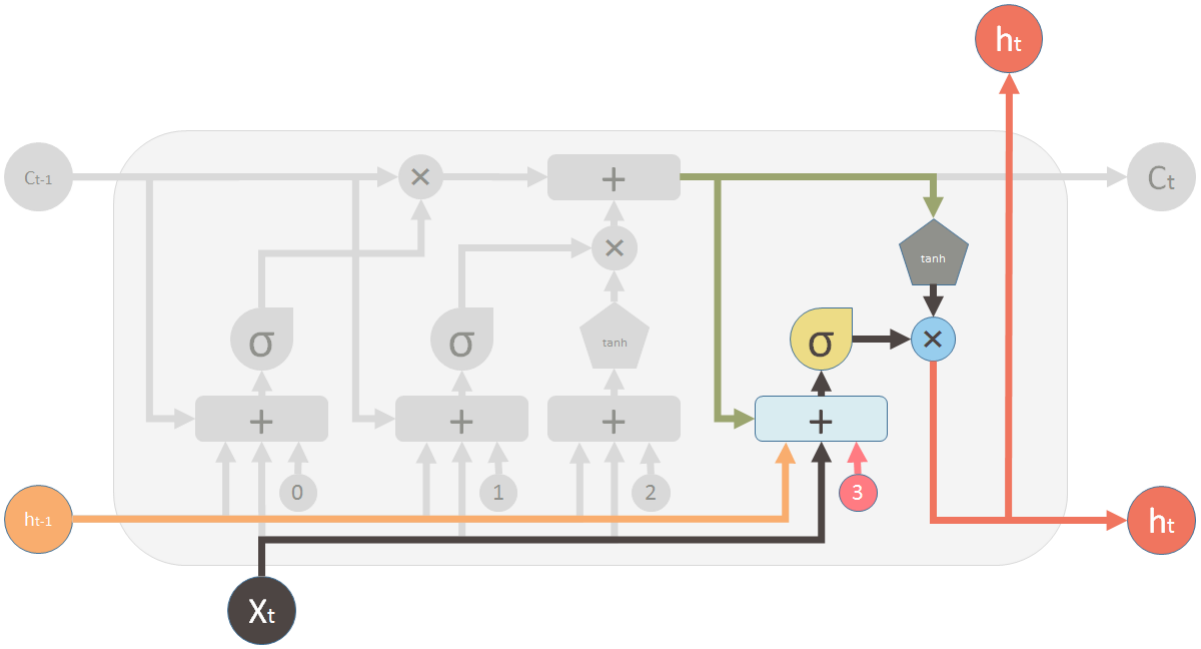


Image from
https://medium.com/@shiyan/understanding-lstm-and-its-diagrams-37e2f46f1714

Appendix B

Activation Functions

In this project and especially in our discussion of LSTM we made rference to activation function. The activation function of a node defines the output of that node given an input or set of inputs. As an analogy, a standard computer chip circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on the input.

Figure B.1: Various Activation Functions

Name	Plot	Equation	Derivative (with respect to x)	Range	Order of continuity	Monotonic	Derivative Monotonic
Adaptive piecewise linear (APL) [15]		$f(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s)$	$f'(x) = H(x) - \sum_{s=1}^{S} a_i^s H(-x + b_i^s)$ [2]	$(-\infty, \infty)$	C^0	No	No
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$	C^∞	Yes	No
Bent identity		$f(x) = \frac{\sqrt{x^2+1}-1}{2} + x$	$f'(x) = \frac{x}{2\sqrt{x^2+1}} + 1$	$(-\infty, \infty)$	C^∞	Yes	Yes
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$	C^{-1}	Yes	No
Exponential linear unit (ELU)[13]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\alpha, \infty)$	C^1 when $\alpha = 1$, otherwise C^0	Yes iff $\alpha \geq 0$	Yes iff $0 \leq \alpha \leq 1$
Gaussian		$f(x) = e^{-x^2}$	$f'(x) = -2xe^{-x^2}$	$(0, 1]$	C^∞	No	No
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$	C^∞	Yes	Yes
Leaky rectified linear unit (Leaky ReLU)[10]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0	Yes	Yes
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$	C^∞	Yes	No
Parameteric rectified linear unit (PReLU)[11]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0	Yes iff $\alpha \geq 0$	Yes

Image from `https://en.wikipedia.org/wiki/Activation_function`

Bibliography

[1] Chris Woodford. Air pollution. `http://www.explainthatstuff.com/air-pollution-introduction.html`, Feb 2017. [Online; accessed on 27-May-2017].

[2] Hasim Sak , Andrew W. Senior and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128, 2014. URL `http://arxiv.org/abs/1402.1128`.

[3] Abshiskey Tiwary , Jeremy Colls. *Air Pollution : measurement, Modelling and Mitigation.* Routledge Taylor and Francis group, London and New York, 2002.

[4] D.A. Dockery and C.A. Pope. Acute respiratory effects of particulate air pollution annual review of public health. *Environmental Epidemiology Program, Harvard School of Public Health, Annu. Rev. Public Health*, 15:1–20, 1994.

[5] Alan J. Cimorelli, Steven G. Perry, Akula Venkatra, Jeffrey C. WeilL, et al. Aermod: A dispersion model for industrial source applications. part I: General model formulation and boundary layer characterization. *Journal of Applied Meteorology*, 44:682–693, 2005.

[6] Paolo Zanetti. *Air Plollution Modelling: Theories, Computational Methods and Available Software.* Computation Mechanics Publications,Bookcraft Ltd, United Kingdom, 1990.

[7] Tsay R.S. Tiao G.C. Some advances in non-linear and adaptive modelling in time-series. *Journal of Forecasting*, 13(2):109–131, April 1994.

[8] Atiya A.F. El Gayar N. El-Shishiny H. Ahmed, N.K. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29:5–6, 2010.

[9] H.G. Schuster. *Deterministic Chaos:An introduction.* Weinheim Physik, 1988.

[10] Gianluca Bontempi, Souhaib Ben Taieb and Yann-Ael Le Borgne. Machine learning strategies for time series forecasting. *Lecture Notes in Business Information Processing, European Summer School, ACM Computing Classification*, pages 63–67, March 2013. ISSN 1865-1348. URL `https://link.springer.com/chapter/10.1007%2F978-3-642-36318-4_3`.

[11] Prokhorov D. Wunsch D. Saad, E. Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on Neural Networks*, 9(6):1456–1470, 1998.

[12] Lendasse A. Sorjamaa, A. Time series prediction using dirrec strategy. *European Symposium on Artificial Neural Networks*, 9(6):143–148, 2006.

[13] Geoffrey Hinton. Advanced Machine Learning:Lecture 10: Recurrent Neural Network. `https://www.cs.toronto.edu/~hinton/csc2535/notes/lec10new.pdf`, 2016. [Online; accessed 27-May-2017].

[14] Sepp Hochreiter , Jurgen Schmidhuber. Long short term memory. *Neural Computation Journal*, 9(8):1735–1780, 1997.

[15] M.Wollme , F.Eyben , B.Schuller and G.Rigoll. A multi-stream asr framework for blstm modeling of conversational speech. *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference*, 4860-4863, May 2011.

[16] Felix A. , Schraudolph , Nicol N. Schmidhuber Gers and Jürgen. Learning precise timing with lstm recurrent networks. *J. Mach. Learn. Res.*, 3:115–143, March 2003. ISSN 1532-4435. doi: 10.1162/153244303768966139. URL `http://dx.doi.org/10.1162/153244303768966139`.

[17] PM25 continuous monitoring: Aceessing the data, March 2014. URL `https://www3.epa.gov/ttn/amtic/files/2014conference/monpmpart3.pdf`. [Online: accessed on 5 July 2017].

[18] NIST/SEMATECH. e-handbook of statistical methods, 2017. URL `http://www.itl.nist.gov/div898/handbook/`. [Online: accessed on 10 May 2017].

[19] NCCS Manual. Lag plots. *NCSS statistical software documentation*, 2017. URL `https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Lag_Plots.pdf`. [Online: accessed on 5 July 2017].

[20] Little Roderick J., Donald Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons Inc, Hoboken, New Jersy, 2002.

[21] European Statistical course e learning. Auto-correlation plot, 2017. URL `https://ec.europa.eu/eurostat/sa-elearning/acf`. [Online: accessed on 5 July 2017].

[22] Robert F. Nau. Statistical forecasting: notes on regression and time series analysis, 2017. URL http://people.duke.edu/~rnau/whatuse.htm. [Fuqua School of Business Duke University, Online: accessed on 10 May 2017,].

[23] Wikipedia. Dickeyfuller test — wikipedia, the free encyclopedia, 2017. URL https://en.wikipedia.org/w/index.php?title=Dickey%E2%80%93Fuller_test&oldid=763298912.

[24] N. R. Draper and D. R. Cox. On distributions and their transformation to normality. *Journal of the Royal Statistical Society. Series B (Methodological)*, 31(3):472–476, 1969. ISSN 00359246. URL http://www.jstor.org/stable/2984350.

[25] Pengfei Li. Box-cox transformations: An overview, 2005. URL https://www.researchgate.net/file.PostFileLoader.html?id=59032e92dc332dd7ed1e88a7&assetKey=AS%3A488088414756864%401493380754638.

[26] Mohammad Arhami , Niman Kamali , Mohammad Mahdi. Predicting hourly air pollutant levels using artificial neural networks coupled with uncertainty analysis by monte carlo simulations. *Environ Sci Pollut Res*, Sept 2013. doi: 10.1162/153244303768966139. URL https://link.springer.com/article/10.1007/s11356-012-1451-6.

[27] Geoffrey Hinton. Mixtures of experts. https://www.youtube.com/watch?v=yIIFnTkvhrQ&index=45&list=PLoRl3Ht4JOcdU872GhiYWf6jwrk_SNhz9, 2016. Accessed 12/05/17.

[28] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *IEEE proceedings in speech recognition*, pages 267–296, 1990. URL http://dl.acm.org/citation.cfm?id=108235.108253.

[29] Patrick Henry Winson. Neural nets. https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/readings/MIT6_034F10_netmath.pdf, Oct 2008. [Online; accessed on 27-May-2017].

Multi Scale Object Detection Using Convolutional Neural Net

by

Olagoke Lukman Olabisi

A **Project** submitted

in the
School of Electrical Electronics Engineering
Laboratory of Signal Processing 5

Supervised by: Prof Jean-Philippe Thiran
Assisted by: Damien Matti

December 2018

“ It always seem impossible until it is done”

Nelson Mandela

Abstract

In autonomous vehicle or robots design, the environmental perception and modeling module is responsible for making meaning of the object in the surroundings. This has two inherent challenges: first it is important to identify when an object is present in an image captured by sensors and second is to accurately classify the object into appropriate object class. Added to this, is the problem of classifying multiple objects of different scales and classes captured in a given frame.

Thus, there are a lot of complexities to deal with: both from the algorithmic and modelling perspective. An algorithm that takes too long to converge optimally might never be deployed and a model too clumsy to understand might never be accepted.

The aim of this project is to develop and test algorithmic models that are able o detect object at different scales. This will be achieved by leveraging the effectiveness of convolutional neural nets (conv net). In particular, it is desired to know, how the depth of the conv net affect performance for multi-scaled object detection. This is important so as to be able to obtain which model performs best for a particular object scale. The result of this analysis can be exploited in ensemble model for object detection.

Acknowledgements

Many people have helped shape and train my ideas over the years. Verily, writing all your names out here would fill an entire book in itself. Thus I hope you will all forgive me if I don't: but as always be assured that I love, cherish and admire you all. You have all given me strong motivation and constructive criticism with which my life has thrived.

In particular, I will like to specially thank Professor Jean-Philippe Thiran of the Laboratory of Signal Processing 5, EPFL, for offering me the opportunity and support to work on this project. A lot of thanks also goes to the assistant supervisor, Mr. Damien Matti (Ph.D Supervisor at EPFL) for his constant support and mentoring while working on this project.

Contents

List of Figures

List of Tables

Abbreviations

Acronym	**W**hat (it) **S**tands **F**or.
IoU	**I**ntersection **O**ver **U**nion
LIDAR	**L**ight **D**etection **A**nd **R**anging
Conv Net	**C**oncolutional **N**eural **N**etwork
RPN	**R**egion **P**roposal **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
HOG	**H**istogram of **O**riented **G**radient
FC	**F**ully **C**onnected Layer
SIFT	**S**cale **I**nvariant **F**eature **T**ransform

Dedicated to my ***mum*** *(who passed away during my Masters studies)*

and

Mr. Ashade (for his constant mentoring)

Chapter 1

Background

1.1 Introduction

In order to be able to transfer specific capabilities of human visual perception to autonomous vehicle or robots , we need to gather information about the real world using different sensors types (for example: LiDAR,Camera). The different sensor types establishes a mapping from scene onto data [1]. Accordingly, the different sensor types would result data in in the form of 2D imagery and 3D point cloud. The measured data from different sensor types contain information that might help us reduce the uncertainty about the task at hand: object detection.

This chapter starts with a discussion of how object detection fits into the autonomous system design architecture. Next, a simple perception model - using simple the pin hole camera - is described to provide a general abstraction of what a "hypothetical" image capture or sensor device should do. From this simple abstraction, we can infer what parameters or problems are inherent for such devices. The remainder of the chapter then shifts to the main task of this project : the detection of objects captured by such devices - since the images have already been captured (by these devices) and were acquired online from kitti Vision Bench mark suite `http://www.cvlibs.net/datasets/kitti/`.

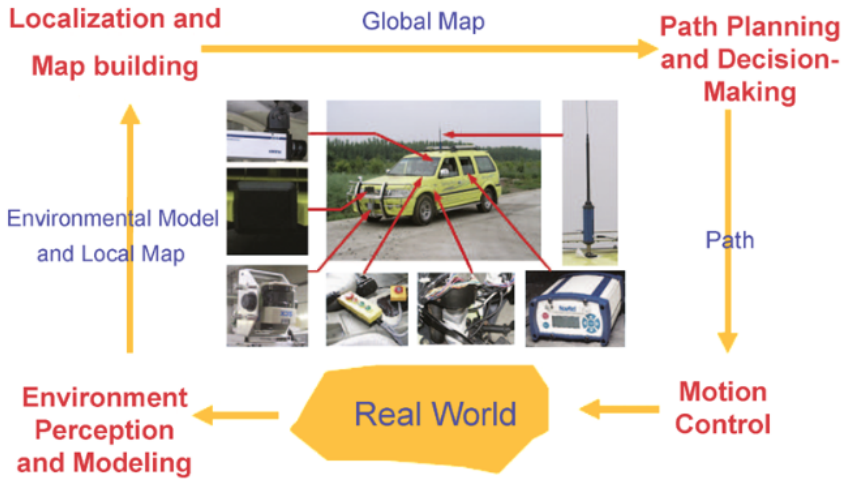
1.2 The Big picture

1.2.1 Autonomous Robots or Vehicles: A General Overview

Autonomous vehicle [1] is a vehicle that is capable of sensing its environment and navigating without human input [2]. It basically consists of four fundamental technologies (Fig 1.1):

- environment perception and modeling
- localization and map building
- path planning and decision-making
- motion control.

Figure 1.1: framework for autonomous vehicle



[Image Source [3]]

The environment perception and modeling module is responsible for sensing environment structures in a multi-sensor way and providing a model of the surrounding environment[3]. The environmental model typically includes a list of objects in the environment - moving objects, that of static obstacles, vehicle position relative to the current road, the road shape, pedestrians etc. This module typically passes the environment model and the local map to the localization and map building module after processing the original data.

[1] We assume autonomous vehicle is an instance of autonomous robot. So that the concepts discussed here can be extended to any generalized autonomous robots

The goal of the second module, localization and map building, is to use geometric feature location estimate in the map to determine the vehicles (or robot's) position, and to interpret the sensor information to estimate the locations of geometric features in a global map . Thus, the second module yields a global map based on the environment model and a local map [3].

The path planning and decision-making module is to assist in ensuring that the vehicle is operated in accordance with the rules of the ground, safety, comfortability, vehicle dynamics, and environment contexts. Hence, this module aims to improve mission efficiency and generate the desired path [3].

The final module, motion control, is to execute the commands necessary to achieve the planned paths, thus yielding interaction between the vehicle and its surrounding environment[3].

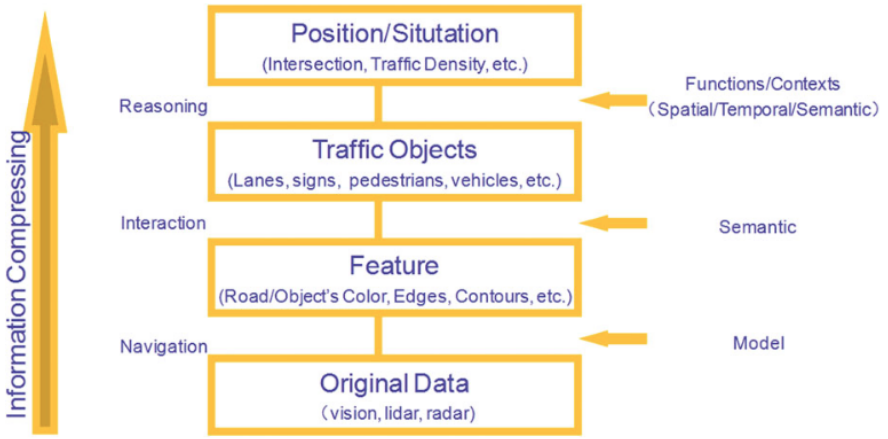
This project will focus on the Environmental perception module and modelling aspect.

1.2.1.1 Multi-sensor Based Environment perception and Modeling

Figure 1.2 illustrates a general environment perception and modeling framework. The frame is explained below

1. The original data are collected by various sensors
2. Various features are extracted from the original data, such as road (object) colors, lane edges, building contours.
3. Semantic objects are recognized using classifiers, and consist of lanes, signs, vehicles, pedestrians
4. We can deduce driving contexts, and vehicle positions

Figure 1.2: Environmental Perception and modelling



[Illustration of basic component of imaging device. Source [3]]

- Multi-sensor Fusion

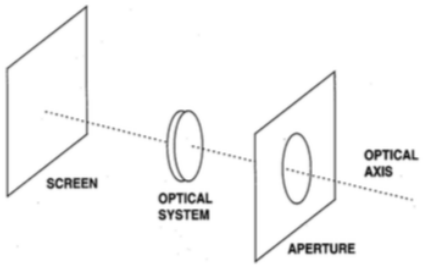
  Multi-sensor Fusion is the basic framework of intelligent vehicles for better sensing surrounding environment structures, and detecting objects/obstacles [3]. Sensors for detecting surrounding environment perception can either be active or passive. Active sensors include Lidar, Radar, ultrasonic and radio, while the commonly-used passive sensors are infrared and visual cameras. These various sensors are capable of providing different detection precision and range, and yielding different effects on environment. In particular, combining various sensors could cover not only short-range but also long-range objects/obstacles, and also work in various weather conditions.

  In general, in a driving environment, we are interested in static/dynamic obstacles, lane markings, traffic signs, vehicles, and pedestrians. This project focuses on developing algorithmic models for detecting such objects.

1.2.1.2 Basis Element of Image Device

The process for image formation in the simple model for perception system begins with light rays which enter the camera through an *angular aperture or pupil*, and hit a screen or *image plane*, which is the vision's system photosensitive device which register's image intensities, see Fig 1.3. We note that most of these rays resulted from reflections of the rays emitted from a light sources and hitting the object surfaces [4].

Figure 1.3: Composition of a Simple Vision Device



[Illustration of basic component of imaging device. Source [4]]

Any single point of a scene reflects light coming from possible all directions. Thus it is possible that many rays reflected from same point enters the imaging device eg (camera). However to obtain a sharp image, all rays coming from the same point should converge onto a single point on the image plane. When this happens we say the image is properly focused. Focusing can be achieved broadly in 2 ways [5]:

- Reducing the camera aperture to a point. This gives the *pin-hole camera.*
- using an optical system such as lenses, and other elements designed to make all rays from the same 3D point converge to a point.

1.2.1.3 Simple Model for Perception: The Pin Hole Camera

To get a good start on how perception mechanism actually works, we start by building an intuition using a very simple model: the pin hole camera. Firstly, we note that the core focus of any imaging system are *digital images.* We draw a division between intensity images and range images [4]. *Intensity images* are the familiar, photograph-like images encoding light intensities acquired by cameras for example. Often, they measure the amount of light impinging on photosensitive device. *Range images* encode shape and distance between a known reference frame and a visible point in the scene. They are often acquired by special sensors like LIDAR, RADAR and SONARS. We emphasize that any digital image irrespective of type consist of a 2-D array of numbers. The numbers might represent light intensities, distances or other physical quantities, depending on the nature of actual image[4].

1.2.1.4 Mathematical Model Of Pin Hole Camera

The Figure above (fig 1.3) shows a pinhole camera. The operation of the pinhole camera consist of a small hole, *pinhole or optical centre* through which rays from object

in real world passes through to form an inverted image on the back of the image box, called image plane. For convenience we consider a virtual image generated by placing the image plane in front of the pin hole camera.

Points in 3D world are represented as, $\mathbf{w} = \begin{bmatrix} u, & v, & w \end{bmatrix}^T$, and we assume that the optical centre is at the centre of this 3D world. The virtual image is formed on the image plane, which itself is slightly displaced from the optical centre along the *w-axis*. The *principal* point defines the point where the optical axis strikes the image plane.

The point where the optical point strikes the image plane is called *focal length*. The task consist in establishing correspondence between the point $\mathbf{x} = \begin{bmatrix} x, & y \end{bmatrix}^T$ in the image plane, and the the corresponding 3D point $\mathbf{w} = \begin{bmatrix} u, & v, & w \end{bmatrix}^T$ of the real world image. The image position $\mathbf{x}$ can be found by observing where the rays from the 3D real image strikes the image plane. This process is called *perspective projection*. In a *Normalized camera*, we have the focal length to be 1 and assume that the origin of the 2D coordinate system of the image plane, is centred at the principal point. A 3D point $\mathbf{w} = \begin{bmatrix} u, & v, & w \end{bmatrix}^T$ is then projected onto the image plane, $\mathbf{x} = \begin{bmatrix} x, & y \end{bmatrix}^T$, by using using similar triangle relationship [5]. This gives:

$$x = \frac{u}{w} \qquad \text{and} \qquad y = \frac{v}{w}$$

.

To make the model more realistic we will correct the assumptions that the focal length is one. Further to this, the final position in the image plane is measured in pixels and not distance so that we have to factor in photo-receptor spacing. To correct both of these assumptions, we add a scaling factor along each axis of image plane [5]. This gives:

$$x = \frac{\phi_x u}{w} \qquad \text{and} \qquad y = \frac{\phi_y v}{w}$$

These parameters are called the *focal length parameters*.

We are not yet done; we know that the pixel position $\mathbf{x} = \begin{bmatrix} 0, & 0 \end{bmatrix}^T$ is at the top left for most images rather than at the centre in which the principal point is positioned. To correct this we add *the offset parameters* [5]. This gives:

$$x = \frac{\phi_x u}{w} + \delta_x \qquad \text{and} \qquad y = \frac{\phi_y v}{w} + \delta_y$$

We also add the *skew parameter* γ that moderates the projected position x as a function of the height v in the real world:

$$x = \frac{\phi_x u + \gamma v}{w} + \delta_x \qquad \text{and} \qquad y = \frac{\phi_y v}{w} + \delta_y$$

Finally we account for the fact that the camera is not usually aligned at the origin of the world coordinate system - in which we assume the optical axis is aligned with w-axis. Thus, we need a coordinate transformation from the world points w to the camera coordinate system [5]. This is achieved with the following transformation:

$$\mathbf{w}' = \mathbf{\Omega}\mathbf{w} + \boldsymbol{\tau}$$

Or expressed in full matrix form as:

$$\mathbf{w}' = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}$$

Incorporating this into the equation for x and y we have:

$$x = \frac{\phi_x(\omega_{11}u + \omega_{12}v + \omega_{13}w + \tau_x) + \gamma(\omega_{21}u + \omega_{22}v + \omega_{23}w + \tau_y)}{\omega_{31}u + \omega_{32}v + \omega_{33}w} + \delta_x$$

$$y = \frac{(\omega_{21}u + \omega_{22}v + \omega_{23}w + \tau_y)}{\omega_{31}u + \omega_{32}v + \omega_{33}w} + \delta_y$$

The parameters $(\phi_x, \phi_y, \gamma, \delta_x, \delta_y)$ are the *intrinsic parameters* because they describe the camera itself. On the other hand, $(\mathbf{\Omega}, \boldsymbol{\tau})$ are the *extrinsic parameters* since they describe the position and orientation of the camera in the world. From the analysis above, we identify the 3 fundamental geometric[5] problems:

- Perspective-n-Point problem: the goal here is to recover the position and orientation of the camera relative to a known scene.
- Calibration: the goal here is to estimate the intrinsic parameter
- Inferring 3D world points (Multi view reconstruction)

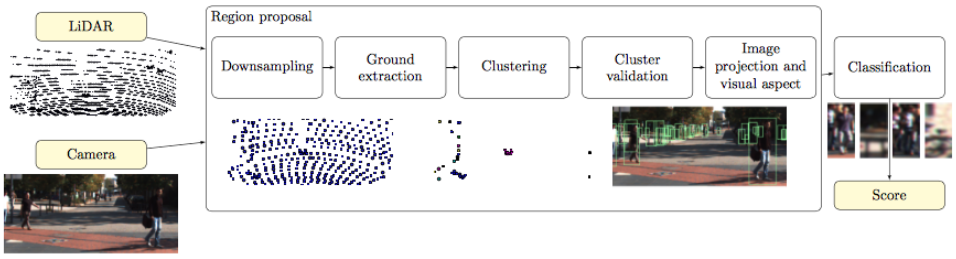
This project won't discuss further the first two problems- since the data set we used, the Kitti data set has taken care of this. [6] - see next subsection.

The last problem is being studied in a parallel project to this one. It is hoped that data from point cloud - of LiDAR sensor -can processed or trained to provide cue for the model described in this project. To acieve this the there should be a calibration from camera to LiDAR system. This is explained in the next section.

1.2.2 Calibration of LiDAR and Camera

The last section introduces the simple model parameters. In order to use LiDAR 3D and camera there is need for some calibration. This is explained below. First the overall architecture (this project and the parallel project on LiDAR integration) should be like a fusion system as below:

Figure 1.4: Camera based system and LiDAR



[LiDAR and Camera in a single architecture]

Although a vison sensor provides rich information, it also has weaknesses, such as a narrow field of view and poor behavior in rapid illumination changes. LiDAR overcomes such drawbacks. LiDAR provides a wider field of view, highly accurate depth measurement and robustness to environmental change. Thus fusion of LiDAR with the camera system gives good performance in real deployment [7].

Generally, the camera and LiDAR should be calibrated in order to convert data between the local coordinate systems of the sensors. Calibration can be divided into two parts: intrinsic calibration, in which the sensors intrinsic parameters, such as focal length, skew parameter and image center, are estimated; and extrinsic calibration, in which the rigid body transformation between the sensors is estimated [7]. Specifically, to optimally combine LiDAR and camera measurements, accurate extrinsic calibration of the six degrees of freedom transformation between the sensors is required. In other words, we should know their relative pose with rotation and translation from each other to effectively use the data from both sensors. Extrinsic calibration between a LiDAR and a camera is roughly classified into two categories: direct and indirect correspondence searches [7].

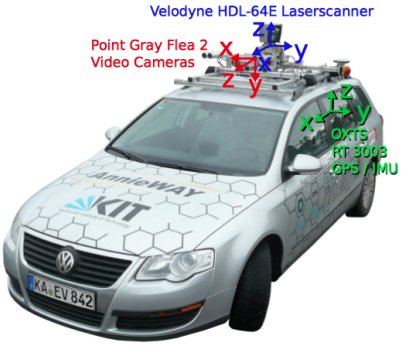
1.3 The Kitti Data set

The section introduces the Kitti data set.

The Kitti dataset was captured from a VW station wagon moving around Karlsruhe (Germany) and has been used extensively in mobile robotics and autonomous driving research. In total, it consist 6 hours of diverse real driving scenario recorded at 10-100Hz using a variety of sensor highlighted. The sensor setup is listed below:

- 2 × PointGray Flea2 grayscale cameras (FL2-14S3M-C), 1.4 Megapixels,1/2″ Sony ICX267 CCD, global shutter
- 2 × PointGray Flea2 color cameras (FL2-14S3C-C), 1.4 . Megapixels, 1/2″ Sony ICX267 CCD, global shutter.
- 4 × Edmund Optics lenses, 4mm, opening angle 90° , vertical opening angle of region of interest (ROI) 35°
- 1 × Velodyne HDL-64E rotating 3D laser scanner, 10 Hz, 64 beams, 0.09 degree angular resolution, 2cm distance accuracy, collecting 1.3 million points/second, field of view: 360° horizontal, 26.8° vertical, range: 120 m.

Figure 1.5: Kitti Dataset Platform

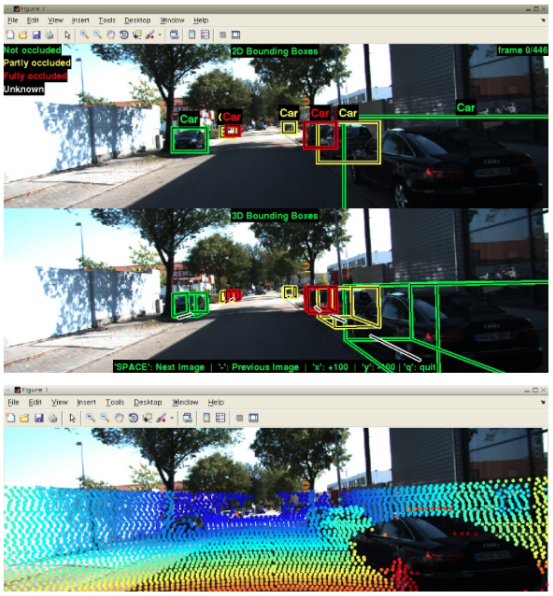


[VW Passat station wagon equipped with four video cameras (two color and two grayscale cameras), a rotating 3D laser scanner and a combined GPS/IMU inertial navigation system, source [6].]

The kitti raw data set is divided into categories: Road, City, Residential, Campus and Person. This project is focused on the Person category. For each category, both gray and color images are stored with loss-less compression using 8-bit PNG files. The engine hood and sky region have been cropped from the images.

The Velodyne scans are stored as floating point binaries. Each point is stored with its (x, y, z) coordinate and an additional reflectance value (r).Per frame sensor readings and corresponding timestamps are provided for each category.

Figure 1.6: image and point cloud from Kitti Data set



[Image Source [6].]

1.4 What is an Object?

In this project, objects are stand-alone things with well-defined regions and centre[8] (such as cars and bicycles) as opposed to amorphous background stuff such as sky , grass , and ground.

We desire that an object:

- has well-defined closed boundary in space[8]
- sometimes it is unique within the image and stands out as salient.[8]
- Object forming background will be labeled as neutral and won't correspond to any object class

1.5 Object classification and Object detection

The goal in this project is to train a model that performs Object detection. Objection detection is quite more challenging [9] than object classification since it requires more complex methods[9] to solve.

Complexity arises because detection requires localization of each object in a frame and subsequent categorization or classification of such object. Localization itself is has a little overhead::

- First, there are numerous candidate locations - set of all proposed window positions for probable object location [9] called proposals must be pre-processed[9]
- Second, these candidates proposals provide rough localization that must be further refined to provide more accurate localization. [9]

Figure 1.7: Object Localization.

[Usually green rectangle represent the real object location (ground truth). The red/blue rectangle represent the proposed locations. The algorithm should learn the objectness of the red rectangle window (given the ground truth) using metrics such as Intersection of union. This is followed by classification of the object. Image Source [8].]

1.5.1 Remarks

In general, it is important to measure and score how a window cover each object in a frame then develop model to classify each of them. This is what makes image recognition different from object detection. In particular, this means that it is required to measure or score the "objectness of a window". The objective of this objectness score is to localize objects in a window before classification.

It is desired that high score should be assigned to windows fitting an object tight - and low score otherwise. How this objectness and classification is done is the central aim of this project.

1.6 Conclusion

In this chapter a general overview of what the environmental perception and modelling aspect of an autonomous vehicle is all about was discussed. In addition a simple model of pin hole camera was used to show how image capture is been simply

realized. In particular, we assume the images of the objects have been acquired and stored publicly online `http://www.cvlibs.net/datasets/kitti/`. Our aim is to develop algorithm that can help us detect object and classify it.The next chapter discusses how we carry out image classification .

Chapter 2

Image Classification

2.1 Introduction

This chapter discusses the intricacies of image classification. It is assumed that the data set are available and what is left is to develop a model or algorithm to do the classification. For simplicity we assume for the moment that there is one object class per image frame. Though there might be multiple such objects to classify. All that is required is that each object class is located in separate frame.

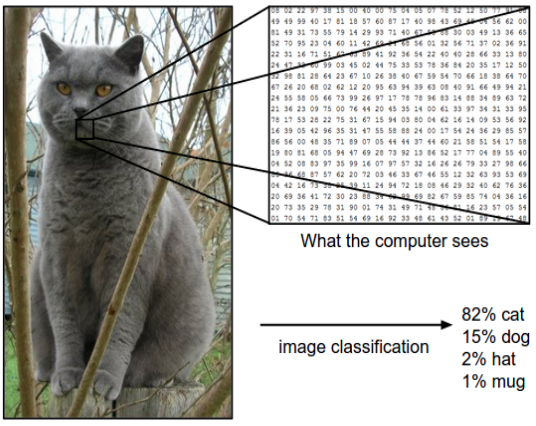
An obvious approach would be to write an algorithm that recognizes each distinct instant class of object. Such algorithmic representation would be required to be invariant to viewpoint variation, scale variation, deformation, occlusion, illumination conditions, background clutter, intra-class variations [10]. It turns out that such approach would not be robust and has yet it is not obvious how one might write an algorithm for identifying same object class in different scenario [10].

It turns out that the data driven approach would be a viable option. In this approach, we a provide the computer with the scenes we have acquired from various sensors and then develop learning algorithms that look at these examples and learn about the visual appearance objects in the scenes. This approach is the data-driven approach, since it relies on first accumulating a training data-set of labeled object images from the sensors. This makes it obvious the reason a standard vision benchmark suite such as kitti[6] is needed.

2.1.1 Simple Image Classification Model

As noted in the previous chapter, to a computer an image is represented as one large 3-dimensional array of numbers as shown in the image, fig 2.1. The task is to intelligently use those "dumb numbers" for decision making: object classification.

Figure 2.1: Simple Cat Classification



[Image from Stanford Lecture note on: Convolution neural network for Visual recognition [10].]

A simplified pipeline for object classification using data driven approach is outlined below:

- Input: the input consists camera captured images of a scene. We refer to this data as the training set. The data from the 3D LiDAR will also be used to augment the input data set. Both the image and point cloud are in the Kitti data set.
- Learning: The aim is to use the training set to learn to recognize pedestrians in the scene. This step is called Learning a model or *training a classifier*[10].
- Evaluation: In the end, it is necessary to evaluate the quality of the classifier by presenting to it a new set of scenes test set, and ask it to detect pedestrians. A score will then be assigned base on how well pedestrians in a scene are detected. Intuitively, we are hoping that a lot of the predictions match up with the true answers (which we call the ground truth [10]).

2.2 Learning Models

2.2.1 Nearest Neighbour and K-Nearest Neighbour

As usual, it is good with a simplified model: the nearest neighbor classifier. The nearest neighbor classifier takes a test scene or image, compare it to every single one of the training set, and try predict if belongs to an object class or not. The aim would be to learn the pixel values that encapsulates the object position in the scene or image. To do the comparison, the L1 or L2 distance will be employed. Given 2 images say I_1 and I_2 , [10], the L1 distance is computed as so:

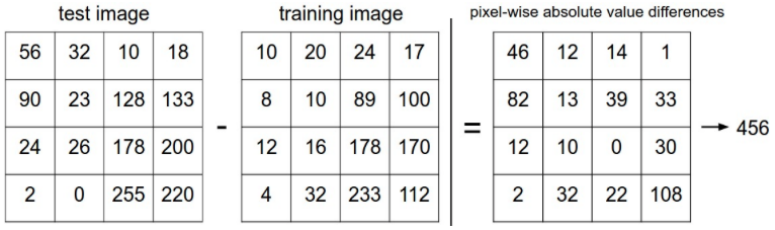
$$d_1(I_1, I_2) = \Sigma_p |I_1^p - I_2^p|$$

and the L2 distance :

$$d_2(I_1, I_2) = \sqrt{\Sigma_p (I_1^p - I_2^p)^2}$$

To put this into context, consider the simple illustration below fig 2.3. Assume the test image correspond to 2D image and we give it to nearest neighbour classifier to check if it is pedestrian or not. To do this it computes the L1 distance between the test image and training image: if the distance is small or zero then the image corresponds likely to a pedestrian.

Figure 2.2: L1 distance Illustration



[Image from online source, [10].]

Suppose there are different labels that an image might correspond to in the training set- for example a proposed image might be pedestrian, car, traffic etc. The idea of the k-nearest neighbour would be to find the closest k images to a given test image; and then vote to pick the correct one. When k=1, we have the nearest neighbour classifier.

In practice, it is required to Split the training set into a training set and a validation set. The validation set is used to tune all hyperparameters. Hyperparameters

are parameters that (for example the L1 and L2) influence the accuracy of our model during training. The test set is required to be a fresh set on which we can evaluate the model.

The nearest neighbour is relatively easy to train but not robust. However, it serves as a useful simple illustration. We now turn to Linear models.

2.2.2 Linear Models

For this section, we define the score function and the loss function. The *score function* defines a mapping from the raw data (of the 2D image) to class scores (where the class represents the possible set of objects to be classified)[10]. The cost function defines the agreement between the predicted scores and the ground truth[10].

2.2.2.1 Linear Classifier Score Function

To each $x_i \in R^D$ of the training set we associate a label y_i. Accordingly, $i = 1, 2 \ldots N$ represent the total number of images in the set (where each image has dimensionality D) and $y_i \in 1 \ldots K$ represents the distinct categories of objects (pedestrians,traffic light,houses etc).

The score function is then defined as

$$f : R^D \rightarrow R^K$$

and maps the raw image pixels to class scores[10]. For the linear classifier, a simple yet useful score function is the linear mapping below :

$$f(x_i, W, b) = Wx_i + b$$

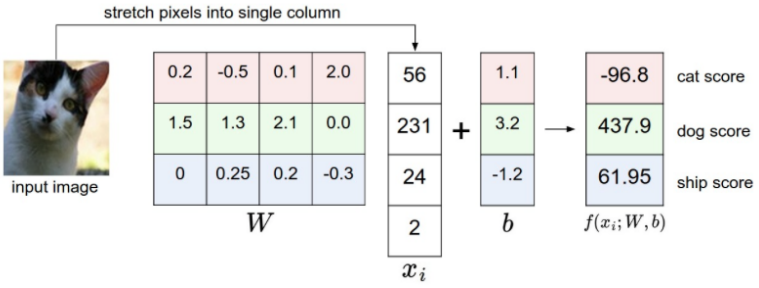
It is assumed that the image x_i has all of its pixels flattened out to a single column vector of shape [D x 1]. To see this in context see the image below:

The matrix W (of size [K x D]) represents the weight, and the vector b (of size [K x 1]) represent the bias.

2.2.2.2 Loss Function for Linear Classifier

Having computed the score for each class. It is appropriate to define a loss function which helps us evaluate the accuracy of the predictions based on the score. A high loss

Figure 2.3: linear score function illustrated



[Image from online source, [10]. Each row of W can be assumed to represent the necessary "weight values" needed to compute the score of a given class label. For any test image, we can compute the corresponding score all class labels. In the example, the classes are cat, ship and dog. Cat however has the highest score has expected]

means the model is not doing well in making correct predictions and a low loss means the predictions of the model are good.

The Multiclass support vector Machine is a simple example of loss function that is widely used. This is given as[10]:

$$\boldsymbol{L_i} = \sum_{\boldsymbol{j \neq i}} \max(0, s_j - s_{y_i} - \Delta)$$

where s_j is the score of the class j and s_{y_i} is the score of the true class, Δ is an hyperparameter. To put this in context, take the output of the cat classification above:

$\boldsymbol{s} = [-96.8, 437.9, 61.95]$. Then to compute the SVM loss function assuming $\Delta = 20$:

$$L_i = max(0, -96 - 437.9 + 20) + max(0, 61.95 - 437.9 + 20)$$

$$L_i = max(0, -) + max(0, -)$$

In general, the SVM loss function wants to ensure the score of the correct class y_i larger than the incorrect class scores by at least by Δ. The formula indicated that it is threshold to zero if the score is negative. In the example above the loss value is zero.

It is also possible to add a regularization parameter to the loss function computation. This is important because if W is the weight we acquired from training. Then any λW would also be a valid weight.Therefore, to encode some preference for a certain set of weights W over others to remove this ambiguity, the regularization parameter is needed. The regularization parameter helps compute the regularization loss [10] and is given below:

$$R(W) = \sum_{n} \sum_{m} W_{n,m}^2$$

Adding the SVM loss and regularization loss (weighted by hyperparameter λ) we have the following objective function [10]:

$$L = \frac{1}{N} \sum_{i} L_i + \lambda R(W)$$

or in full as

$$L = \frac{1}{N} \sum_{i} \sum_{j \neq y_i} [\max(0, f(x_i; W, b)_j - f(x_i; W, b)_{y_i} + \Delta] + \lambda \sum_{m} \sum_{n} W_{n,m}^2$$

where N is the total number of training example and λ an hyperparameter.

In practice it turns out that Δ can be set to 1 (assuming we *normalize* the data set before training). Together both Δ and λ control the tradeoff between the data loss and the regularization loss in the objective [10]. The magnitude of the weights has direct effect of the computed scores, therefore the Δ and λ help prevent the model to generalize (prevent *overfitting*, or say invariant to view point variation).

Another popular choice of loss function is the soft Max defined as:

$$L_i = -\log(\frac{\exp(f_{y_i})}{\sum_j \exp(f_j)})$$

The output of the softmax loss function is a normalized class probabilities. This means it assigns probability value to each class, and the most likely class gets the highest probability value.

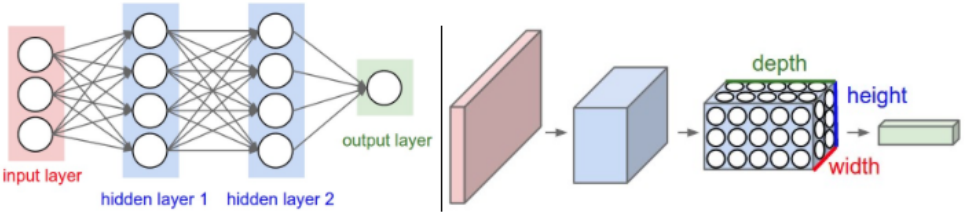
2.2.3 Convolutional Neural network

Conv net is deep-learning architecture which takes its inspiration from the perception mechanism of living organisms. It has recently been very effective in visual pattern pattern recognition and representation.

Like typical neural network it has multiple weighted linear units (discussed earlier) endowed with an activation function which together act on data fed into it from the input node and is usually trained using the back-propagation[11] algorithm. But,

unlike a regular Neural Network, the layers of a Conv Net have neurons arranged in 3 dimensions: width, height, depth [10]. As the name suggest CNN makes use of the usual convolution operation in place of the usual matrix operation of a typical neural net.

Figure 2.4: Neural net Versus Conv Net



Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

[Image from, [10].]

2.2.4 Convolutional Neural Network

The convolution of x and w written as $x * w$ is defined as [12] :

$$s(t) = (x * w)(t) \tag{2.1}$$

$$= \int x(a)w(t-a)da \tag{2.2}$$

If we consider discrete convolution, the equations above becomes:

$$s(t) = (x * w)(t) \tag{2.3}$$

$$= \sum_{a=-\infty} x(a)w(t-a) \tag{2.4}$$

In ConvNets, x would typical be the input to the CNN and w the kernel or Filter. The output is usually called Feature Map. Since the input to the ConvNets are multidimensional array, 2D image for example, then we have to use 2D Kernel. Let I be the 2D image and K the kernel then the 2D convolution is defined as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-m)$$

Since, convolution operation is commutative, the equation above can be rewritten as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(i,j)$$

Most libraries implements cross correlation which is same as convolution but without flipping the the kernel [12]:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(i,j)$$

The figure below shows convolution on 2D tensor:

Figure 2.5: Illustration of Conv

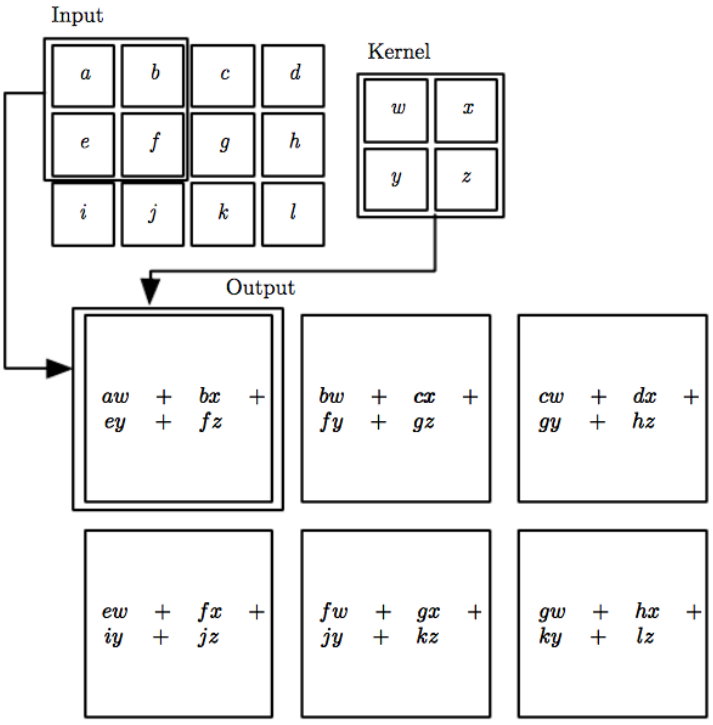


Figure 9.1: An example of 2-D convolution without kernel-flipping. In this case we restrict the output to only positions where the kernel lies entirely within the image, called "valid" convolution in some contexts. We draw boxes with arrows to indicate how the upper-left element of the output tensor is formed by applying the kernel to the corresponding upper-left region of the input tensor.

[Illustration from the deep learning book by Ian Goodfellow, [12].]

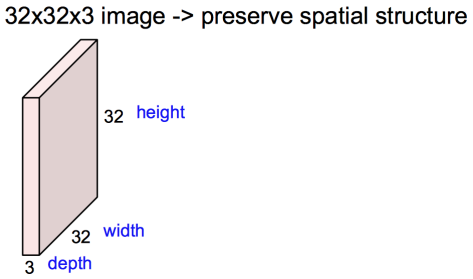
2.2.5 ConvNets Building Blocks

Three main types of layers are used to build ConvNet architectures: Convolutional Layer,Pooling Layer, and Fully-Connected Layer.

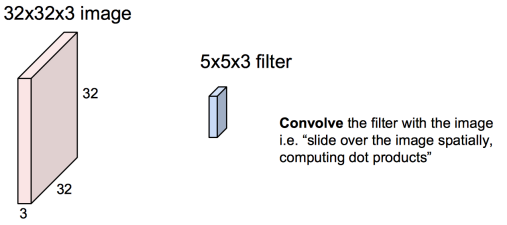
2.2.5.1 Convolutional Layer

The Conv layer is the main computational unit of the network. The CONV layers parameters consist of a set of learnable filters. Each filter is small spatially (along width and height), however it extends through the full depth of the input volume. **Local Connectivity:** Typically, it is impractical to connect neurons to all neurons in the previous volume. Each neuron is connected to only a *local region* of the input volume. The spatial extent of this connectivity is a hyperparameter called the receptive field of the neuron (equivalently this is the filter size). The extent of the connectivity along the depth axis is always equal to the depth of the input volume[10].

There is no way to visualize the convolution layer than with an example:

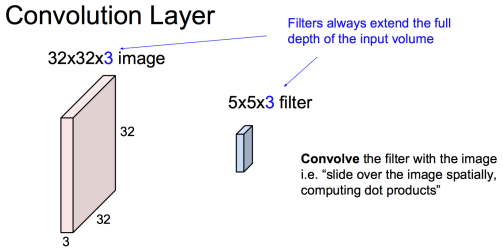


(a) Let's say this is the input to the network

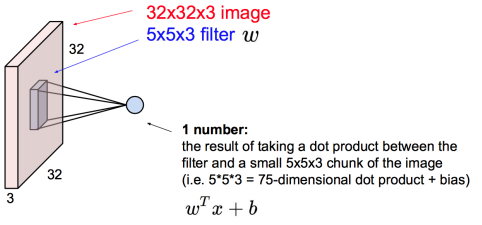


(b) The input and the Filter

[Image Source [10]]

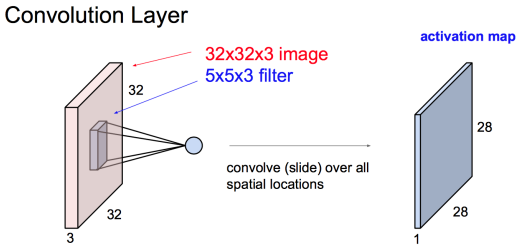


(c) Filter should extend the depth

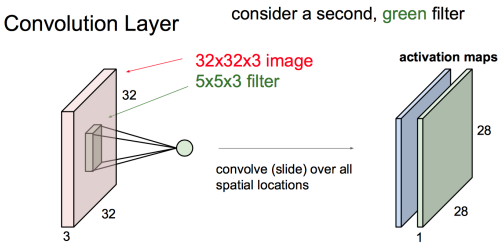


(d) The output: Note use of Linear Function locally, not on entire image

[Image Source [10]]

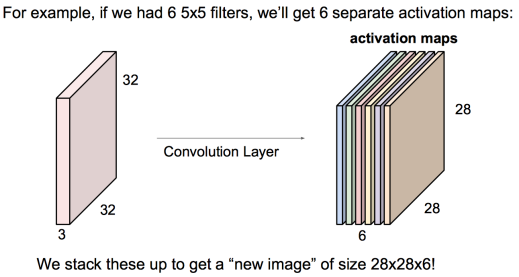


(e) After convolution over the entire image , the output is a Feature map or activation map

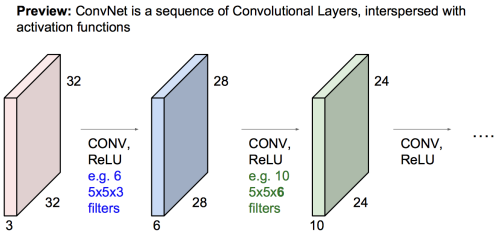


(f) Typically, more than one filter is used

[Image Source [10]]



(g) Each Activation Map correspond to a filter



(h) CNN is sequence of these layers with activation function (ReLu)

[Image Source [10]]

It is important to note that:

- The depth of the output volume is an hyperparameter and reflects the number of filter used in the network [10]
- The stride determines how we slide the filter[10]
- The input volume is often zero padded. This involves padding the input volume with zeros around the border.
- Parameter sharing refers to using the same parameter for more than one function in a model[12]. It is used in convnet to control the number of parameters[10]. In a convolutional neural net, each member of the kernel is used at every position of the input The parameter sharing used by the convolution operation means that rather than learning a separate set of parameter for every location, we learn only one set [12].

Example 2.1. *Consider that we have receptive of size* $55 \times 55 \times 96 = 290,400$ *(width,height and depth) in the first Conv Layer of a network, and each was connected to region of* $11 \times 11 \times 3 = 363$ *size in the input volume. Thus, this gives a total*

363 *weights and 1 bias. Together, this adds up to* $290400 \times (363+1) = 105,705,600$ *parameters on the first layer of the ConvNet alone! Using Parameter sharing and denoting a single 2-dimensional slice of depth as a* ***depth slice*** *(=96), we are going to constrain the neurons in each depth slice to use the same weights and bias. This is equivalent to all* 55×55 *neurons in each depth slice*(96) *using the same parameters - 96 unique set of weights (one for each depth slice). This gives* $96 \times 11 \times 11 \times 3 = 34,848$ *unique weights, or 34,944 parameters (+96 biases).[12]*

- RELU layer will apply an elementwise activation function, such as the max(0,x) thresholding at zero.

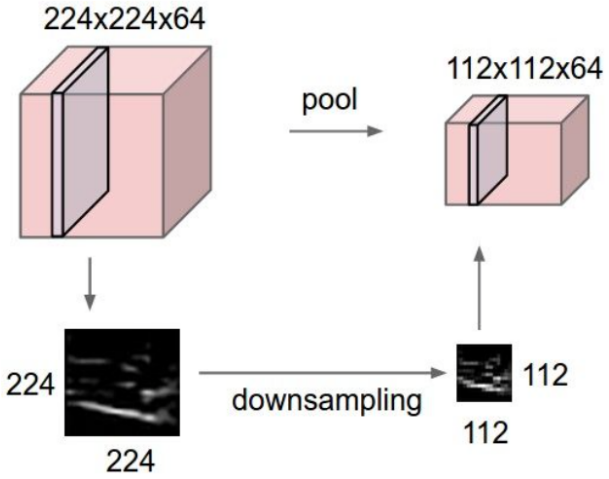
Conv Layer Summary [10]

- Input: $W_1 \times H_1 \times D_1$ as input volume.
- Parameters required:
  - Number of Filters, **K**
  - spatial extent, **F** (Filter size)
  - Stride, **S**
  - Amount of Zero padding, **P**
- Output Volume: $W_2 \times H_2 \times D_2$ where:
  - $\mathbf{W_2} = \frac{\mathbf{W_1 - F + 2P}}{\mathbf{S+1}}$
  - $\mathbf{H_2} = \frac{\mathbf{(H_1 - F + P)}}{\mathbf{S+1}}$
  - $\mathbf{D_2} = \mathbf{K}$
- $\mathbf{F} \times \mathbf{F}$ weights per filter. Total of $(\mathbf{F} \times \mathbf{F} \times \mathbf{D}).\mathbf{K}$ weights and **K** biases.

2.2.5.2 Pooling Layer

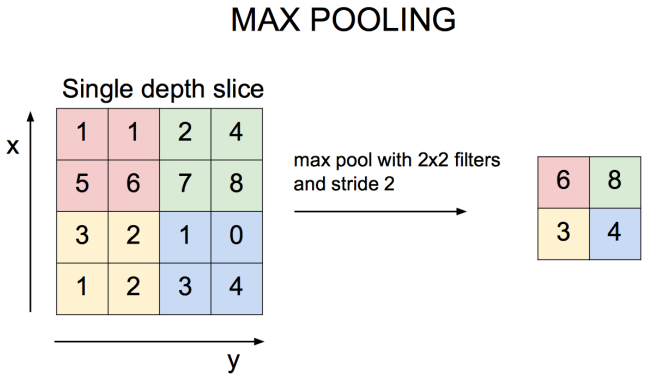
Its function is to progressively reduce the spatial size of the representation thereby reducing the amount of parameters and computation in the network- hence reduce overfitting. It operates on every depth slice of the input and resizes it spatially, using the "max" operation. Thus is illustrated below:

Figure 2.6: Pooling as Downsampling



[Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume [10].]

Figure 2.7: Max Pool using 2×2 filter



[The most common downsampling operation is max, giving rise to max pooling, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2×2 square). Source, [10].]

Pooling Layer Summary[10]

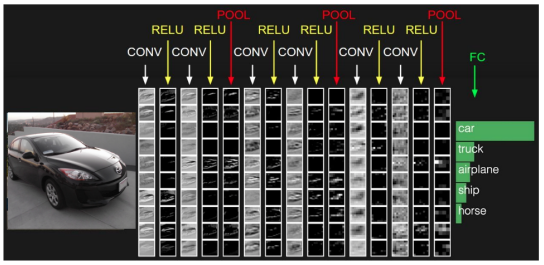
- Input: $W_1 \times H_1 \times D_1$ as input volume.
- Parameters required:
  - spatial extent, $\mathbf{F}$ (Filter size)
  - Stride, $\mathbf{S}$
- Output Volume: $W_2 \times H_2 \times D_2$ where:
  - $\mathbf{W_2} = \frac{\mathbf{W_1 - F}}{\mathbf{S+1}}$
  - $\mathbf{H_2} = \frac{\mathbf{(H_1 - F)}}{\mathbf{S+1}}$
  - $\mathbf{D_2} = \mathbf{D_1}$
- Introduces zero parameters. Usually no zero padding.

2.2.5.3 Fully Connected Layer

Contains neurons that connect to the entire input volume, as in ordinary Neural Networks [10]. It is in this Layer that the class scores for the network is computed.

In general a typical CNN architecture pipeline would involve *(input - conv layer - relu - pool - fully connected layer)*[10]. This is illustrated in the figure below:

Figure 2.8: Typical conv net Architecture pipeline

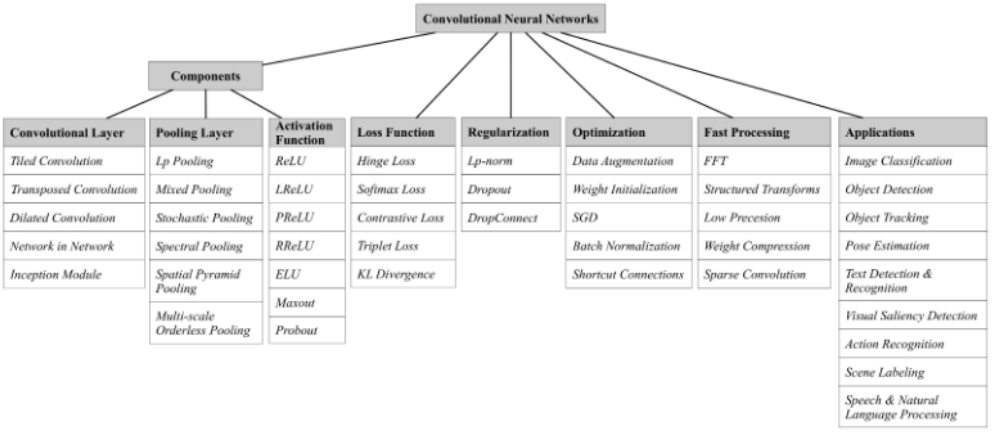


[Image source [10].]

2.2.5.4 Fully Connected Layer

The figure below shows the possible components, techniques and application of conv net:

Figure 2.9: Components, techniques and application of conv net



[Image source [10].]

2.3 Conclusion

This chapter lays the foundation of image recognition. In addition the typical CNN architecture was discussed. In the next chapter we build on this and try to utilize 3D point cloud to reduce uncertainty in the prediction.

Chapter 3

Object Detection

3.1 Introduction

In the last chapter, image classification was extensively discussed. This chapter extends the last chapter further to include image detection. This is a domain of computer vision that has been around in the last few decades. Previously, progress in object detection was based exclusively on the use of HOG (Histogram of gradient) and SIFT (Scale Invariant Feature Transform) algorithms. However, in recent terms, CNN have shown to outperform these other methods[13]. The task ahead then becomes how to adapt CNN for object detection task. Image detection, differs from image detection because it involves object localization.

This chapter starts with a brief discussion on object localization. Furthermore, the technique and metrics for object localization and detection in CNN will be analyzed.[1]

3.2 Object Localization And Detection

In the last chapter, image classification problem was discussed. This can be formulated like so: given the image, the algorithm (CNN) identifies the object (for example cat in the image below):

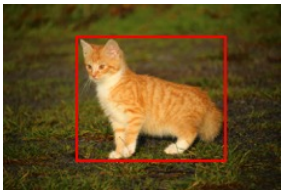
[1]This chapter is based on the Deep learning specialization course I completed on Coursera [14] taught by Prof Andrew Ng

Figure 3.1: image of cat

[Image source [15].]

In the localization problem not only is the algorithm to label or identify the object but it has also to put a bounding box around the position of the object:

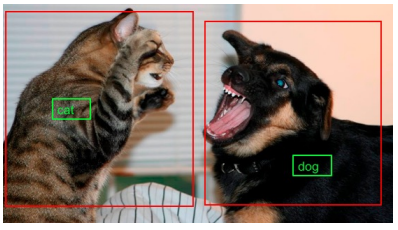
Figure 3.2: Localization problem



[Image source [15].]

In the detection problem, there might be multiple of such objects of different classes at multi-scales in the image, and the algorithm has to localize and label them:

Figure 3.3: Detection problem



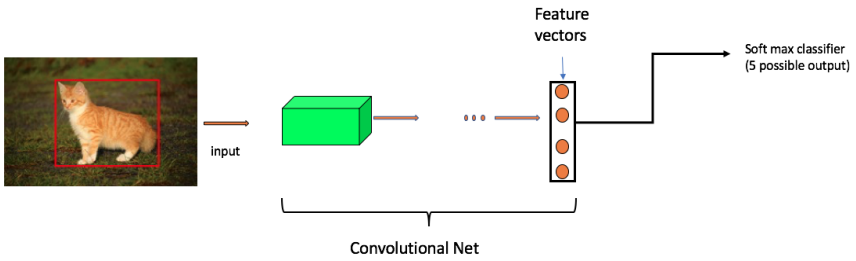
[Image source [16].]

In the detection problem , multiple object of different classes and at different depth can be in the image. In the localization (and by extension classification problem), there is usually single object located at the centre of the image. This is a constraint on the object class in an image and not on the number of object classes in the training set.

3.2.1 Object Localization Problem

The figure below shows the pipeline for classical object identification.

Figure 3.4: Classification problem



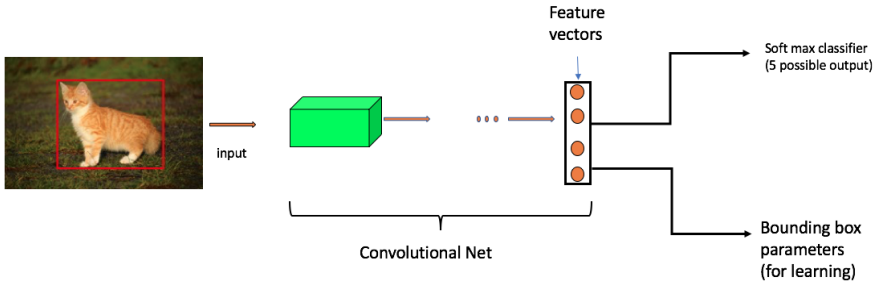
[This pipeline use the conv net discussed in last chapter.]

The output of this pipeline in a feature set. The feature set is input into a soft-max -as discussed in the last chapter- to compute the predictive classes. The predictive classes in this project consist of 5 object categories - there are 5 possible outputs from soft-max, this corresponds to the number of object classes in the input labels , the classes considered are[2] -:

- Cars
- Bicycles
- Big Pedestrians
- Small Pedestrians
- Background (None of the above)

To upgrade this into a localization framework, simply add bounding box as output in the learning pipeline as below:

Figure 3.5: Add localization module



[This pipeline use the conv net discussed in last chapter. In localization we assume there is one object class per image though there might be multiple object in a single image]

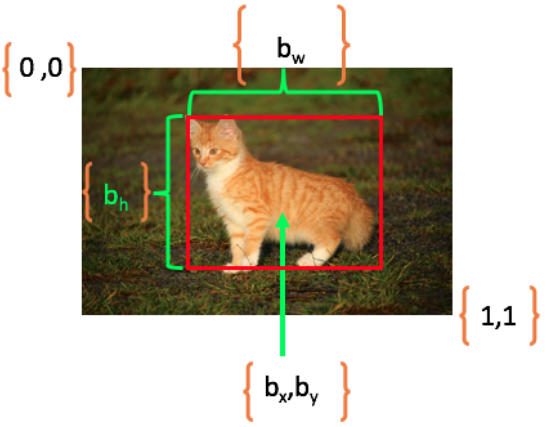
[2]cat class object is not part of the data set. It is used in the example for pedagogical illustration

The output now carries the bounding box information. But how is the bounding box specified? The bounding box is described by 4 variables:

$$[b_x, b_y, b_h, b_w]$$

The b_x, b_y specify the centre of the image, while b_h, b_w specify the height and width of the image. In addition, let's assume the convention that the upper left of the image is co-ordinate [0,0] and lower-right is co-ordinate [1,1]. This is illustrated below:

Figure 3.6: Bounding box parameters



[The bounding box parameters would also be learned. This would be illustrated later]

3.2.1.1 Simple Illustration

Let the input, x, fed into the conv net be any object class defined in the set below:

{x : x ∈ { Small pedestrian, Big Pedestrian, Bicycles , Cars, Background}}

The target label is given as:

$$y = \begin{bmatrix} O_p \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

The component O_p represent "object present". It serves to detect when an object class is in the image. Mathematically, it represent the probability that an object is not "background". Then there are the parameters that define the bounding box for the object detected. Lastly there is object class parameters that signify which object class is present. Thus c_1 represents (probability) class Small pedestrians, c_2 represents (probability) class big pedestrians, c_3 (probability) represent class bicycles, and c_4 (probability) represent class cars.

Example 3.2.1. Let x = car.

Figure 3.7: Input : Object class car

[input to the neural net. Image source [17]]

Then: $y = [1, b_x, b_y, b_h, b_w, 0, 0, 0, 1]^\top$

Thus y implies that an object is present and the object class is car.

Example 3.2.2. Let x = Background (no object class).

Figure 3.8: Input : Object class background

[input to the neural net. Image source [18]]

Then: $y = [1, ?, ?, ?, ?, ?, ?, ?, ?, ?]^\top$

Thus y implies that no object is present. The other parameters are thus not needed: hence the "?" - meaning "I don't care".

This section ends with a discussion of what loss function should be used? For the object classes, cross entropy loss will be appropriate. For the bounding box parameters L2 loss will be appropriate. For the O_p a logistic loss or L2 loss will do the job.

Having discussed object localization in depth, it is now right to extend this model for detection.

3.2.2 Sliding Window Algorithm: Sequential Execution

In the sliding detection algorithm, a closely cropped image of the object classes are designed and input into the conv net. This is illustrated below.

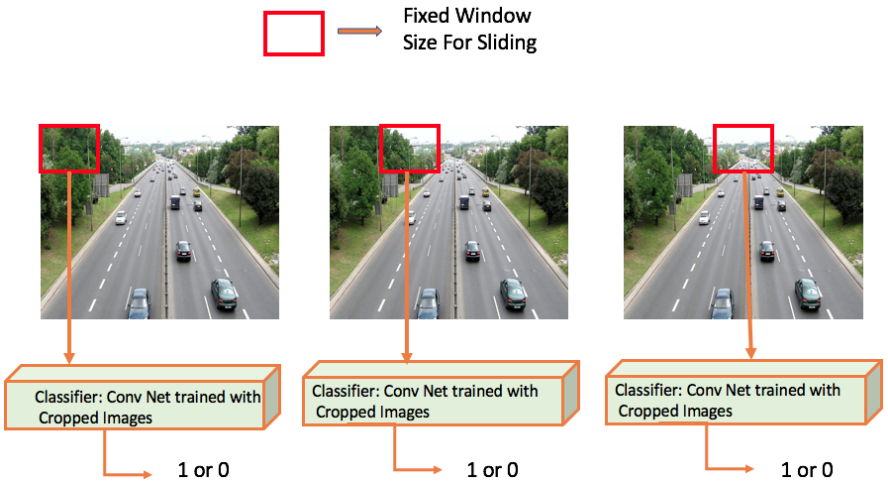
Figure 3.9: Closely cropped image for use in sliding window algorithm

[Label 1 signifies this is a car. And 0 signifies background. Image 2 source [19] Image 4 source [20] [17]]

Train a conv net as illustrated in 3.4 to identify the cars. Then use the trained conv net in sliding window detection as below:

1. Select a fixed window size.
2. Pass the fixed window frame over the image.
3. At each instance feed the region cover by the window into conv net classifier
4. The classifier determines if it is an image or not (0 or 1)

Figure 3.10: Sliding window technique Illustration



[Label 1 is car . Label 0 is background. This illustration implies that the conv net had been trained with cropped image. During testing, the sliding window of same size as cropped image used for training is sequentially applied over test image. A each instant, the conv net is used to test if an image is present or not.]

Lastly, resize the fixed window into a larger size and feed into the conv net - the entire process is repeated with larger window size. The name "sliding window" comes from using the square box and sliding it over the image at specified stride - stride is how much the rectangle in previous region over lap or shift as it moves to the next position. The draw back of this method is

1. It is computationally expensive when used with conv net. The conv net takes appreciable time to classify objects. Now, doing this for many window regions will be prohibitively expensive.

2. It might not be suitable for real time object detection for reason(1).

When used with HOG or SIFT this approach might be be less computationally expensive. However, the aim in this project is to leverage the use of state-of-the heart technique. The next section describes a more optimal way use this technique with conv net.

3.3 Convolutional Sliding Window: One Forward Pass Execution

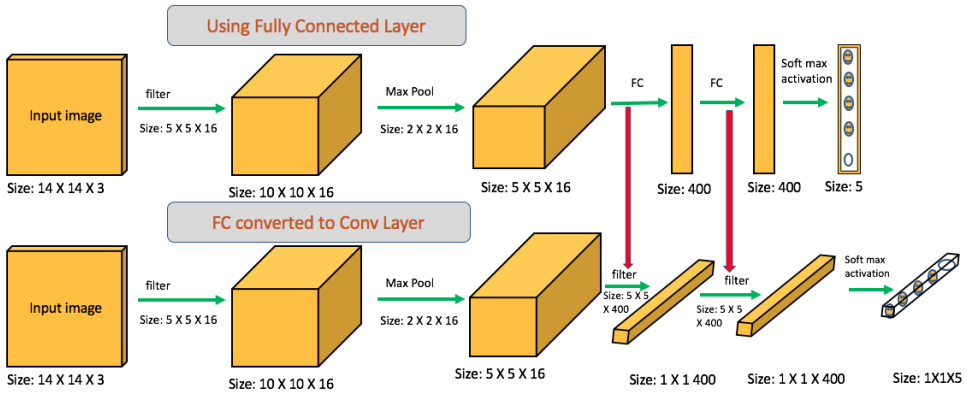
This section shows how to implement, in a computational efficient way, the sliding window algorithm with conv net as classifier. But before this, it is important to know

how to convert fully connected (see last chapter) layer to convolutional layer. This section takes inspiration from [21].

3.3.1 From Fully Connected Layer to Convolutional Layer

From last chapter, the fully connected layer was shown to be part of of the conv net architecture. This section documents how to make this into convolutional layer. First, consider the figure below:

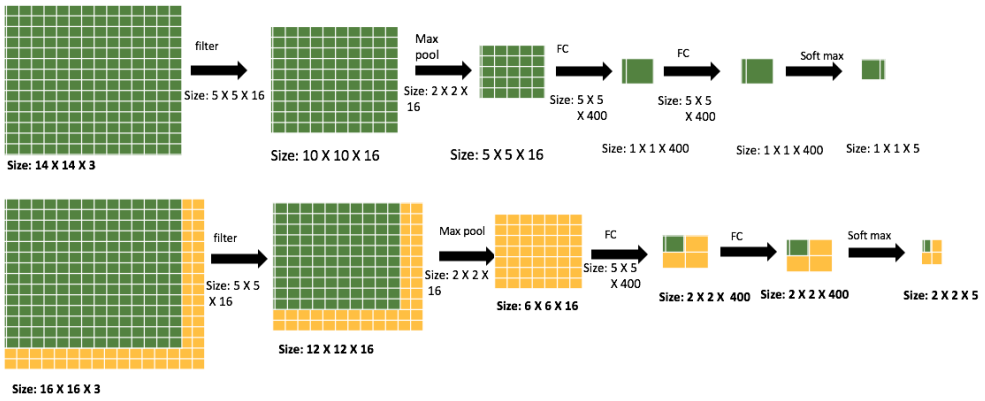
Figure 3.11: Conversion of Fully connected layer to Full convolutional layer



[There are 5 object casses in the classification problem. Hence the output of the softmax is 5. The empty circle signifies background or no object class. In actual implementation the size of the input image will be far bigger than the one illustrated.]

The pink downward pointing arrow shows the point at which the FC layer is replaced with Conv layer. The benefit is that it makes for efficient use of the sliding window in the conv net. In particular it can be observed that certain the operations of the network are duplicated. This can be exploited to achieve efficiency - by sharing repeated operations across different parsing of the sliding window. Consider the figure below:

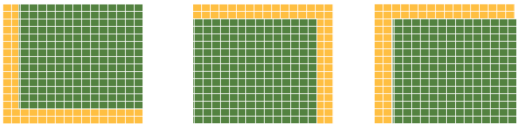
Figure 3.12: Implementation of sliding window algorithm with conv net



[The architecture leverages the FC conversion to Filter operation. It should be noted that the the dimensions have be represented by rectangular dimension - the channel depth has been omitted]

In the figure above, it is assumed that we trained with cropped input image of size $14 \times 14 \times 3$. Let's despicts the scenario where FC is used - no conversion of FC to filter. In the testing phase, assume test image of size $16 \times 16 \times 3$ and window size of $14 \times 14 \times 3$ corresponding to the size of image used during training. The window will fill in the position shown in the superimposed green on yellow grid which represent the test image. In the next iterations or epoch,the windows are shifted as depicted in the figure below:

Figure 3.13: Image shift positions in sequential execution Model

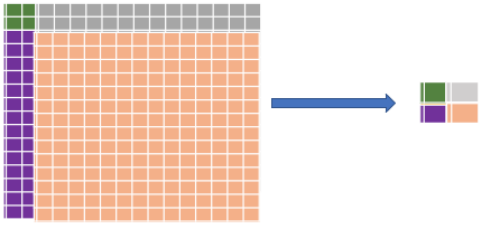


[Shift positions assume using fully connected layer was not transformed to conv layer. There are 4 shift in total. The first is represented in green grid on yellow background in fig 3.12.Each shift is run separately as new training epoch. So the operation is not optimal]

However when the FC is converted to conv layer, the entire 4 operation becomes a single operation, as depicted in the second conv net of fig3.12. This saves a lot of computation cost - in the actual model the size will be a lot bigger, hence this transformation is needed. The output of fig 3.12 is $2X2X4$, and it corresponds to the output from the four filter positions - thanks to the FC to Conv layer conversion, this is now a single operation.

To see this observe the figure below. Each filter position now corresponds to a particular point in the $2X2X4$, output. This is possible because most of the operations are duplicated:

Figure 3.14: Window position superimposed: Optimal execution



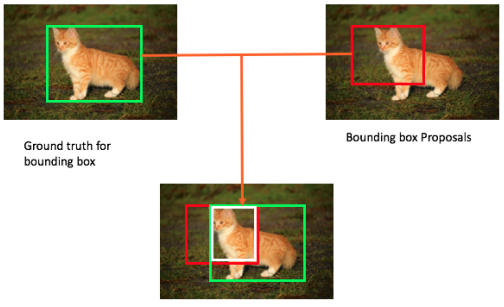
[In the left rectangular grid, the colors represent the 4 window shifts positions in 3.13 superimposed to show . Thus the 4 shifts are trained as single epoch now. This is possible thanks to FC to conv layer conversion. In addition there is sharing of shared operations implied.]

This section discusses the convolutional implementation of the sliding window. By using conv layer instead of FC we can go from the sequential sliding window implementation to one-forward pass implementation. To end this section, it should be observed that no mention was made of the bounding box position. Indeed, the bounding box position - using the technique discussed up till now - won't be accurate. In the next section, we discuss how to a performance metric for bounding box

3.3.1.1 Evaluating Object Localization: Intersection Over Union

The intersection of Union (IoU) is a metric for evaluating bounding box. To understand IoU consider the figure below:

Figure 3.15: Intersection of union



[The Ground truth is green rectangle. The bounding box proposal is red. The intersection is the white rectangular region]

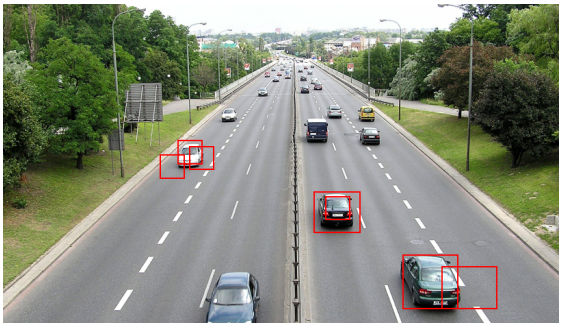
Given the ground truth bounding box and the bounding box proposals. The IoU is given as the area of the size of the intersection of the two rectangles (that is white rectangle) divided by the size of the union of the rectangles. It is a measure of overlap of the 2 bounding boxes.

$$IoU = \frac{\text{size of intersection region - represented by white rectangle}}{\text{size of union of the ground truth and proposal}}$$

If the ground truth matches the bounding box proposal then the IoU is 1. Typically, the proposal is accepted if the IoU ≥ 0.5, otherwise it is rejected. The bigger the IoU the more stringent is the the model and the better the bounding box. The model will also have to train on this metric: this will be explained later.

3.3.1.2 Non Maximum Suppression

Figure 3.16: Non maximum suppression



[The Non maximum suppression is use in situation when there are numerous detection for a given object. Note in the image this is denoted by the red rectangles around the cars. Only few cars were selected for illustration purposes]

As illustrated in the figure above, the Non max suppression is cleaning up multiple bounding box detection for a given object: it does so by first discarding bounding box with IoU ¡ 0.6, or any predefined threshold; then it selects the bounding box with the highest predictive class while deleting the bounding boxes that has high over lap with the selected one. The selected box is thereafter, proposed for the prediction algorithm. The predictive class represent the output probability of whether it contains an object or not - regardless of the class. This was discussed in earlier section. The non-max suppression algorithm can be carried on each of the object classes available during the training epochs.

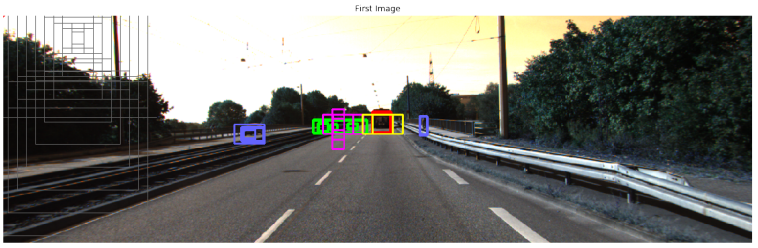
For example, In the middle vehicle for example the centre rectangle - which would we assume has low IoU and low class prediction compared to the outer one - will suppressed since it overlaps with the outer rectangle. Then the algorithm moves to other

rectangle that don't overlap with the centre vehicle. For example far left vehicle, and repeat the same process.

3.3.1.3 Anchor boxes

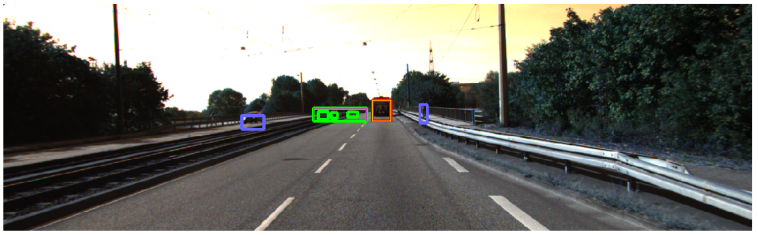
In other to optimize the detection of objects and bounding box prediction, this project makes use of default bounding box similar to [22]. Fixed-size collection of bounding boxes and scores for the presence of object class classes in those boxes, followed by a non-maximum suppression step to produce the final prediction. This is followed the classification of the object present in the box using the neural net. The bounding box is trained with a linear regression model. This is illustrated below:

Figure 3.17: Default Bounding Box Anchors



[Different bounding box anchors to capture various object in an image]

Figure 3.18: Default Bounding Box After non maximum suppression



[Default bounding box Normalized]

This approach enables multiple objects that fall into overlapping bounding box to be detected easily. An example would be when a pedestrian is in front of a car: this might just be treated as single bounding box because they fall into the object classes to be detected and the bounding box are within same area. Using anchor boxes in such scenario of various sizes however, these two objects can be detected separately.

3.4 Conclusion

In this chapter the focus was on a shift from a image classification to image detection algorithmic analysis. The fundamentals idea were discussed followed by the state-of-the-heart approach (Con net sliding window). The next chapter will focus on the experimental results from putting all this description in multiple implementation.

Chapter 4

Theoretical Model Analysis

4.1 Introduction

[1] Up until recently, most neural net architecture have been treated as black box with no theoretical explanation of why they work so well. The aim of this chapter would be to give some theoretical explanations of neural net architecture using concepts from information theory. This will help understand the behavior of some of the the models and experimental results that would be discussed in the next chapter.

This chapter starts by defining the learning problem from rate distortion theory perspective. Next, this is extended to describe the neural net inner architecture. Lastly, the training behavior of the model is explained using compression and estimation phase of the stochastic gradient descent.

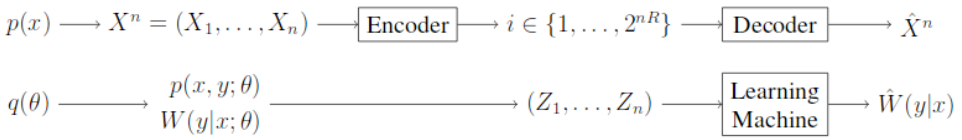
4.1.1 Supervised learning and rate distortion theory

It turns out that a good place to start studying the theoretical underpinnings of the learning mechanism discussed so far is by using analogy from rate distortion theory. This will help clarify further why a performance metric is important and also establish a bound on the achievable learning capacity [23].

The figure below is important for the explanation in this section:

[1] This section is motivated by my study on rate-distortion theory. Somehow, I reasoned that there is absolutely correlation between the rate distortion theory and supervised learning paradigm. Fortunately, I was able to find related papers that discusses such relationship exactly as I had imagined it in my mind. This might help us understand the neural net black box appropriately.

Figure 4.1: rate distortion encoder and decoder versus supervised learning



[Image source [23].]

4.1.1.1 Rate Distortion for encoder and decoder

In rate distortion theory, given a source distribution p(x), which produces an n-length sequence X^n, the encoding function is defined as :

$$f_n : X^n \to \{1, 2, ..., 2^{nR}\}$$

where R is the encoding Rate. The decoding function does an approximate or quantized reconstruction (reconstruction problem) of X^n like so:

$$g_n : \{1, 2, ..., 2^{nR}\} \to \hat{X}_n$$

Typically, a distortion function :

$$d : x \times \hat{x} \to R^+$$

measures the cost of representing the symbol x by $\hat{x}$: consequently the average per symbol distortion of the elements of the sequence is then given as [24]:

$$D = Ed(X^n, g_n(f_n(X^n)))$$

- In summary it is desired that the distortion function is bounded.
- At each instance the smallest distortion region that best approximate the sent sequence is selected
- This is similar to supervised learning where our distortion function would normally be the performance function described in the previous chapter.
- Again like the previous chapter , the aim is to minimize the distortion (increase performance or decrease error).

Next, an elaborate bayesian statistical analogy is described in next section.

4.1.1.2 Bayesian learning perspective

Using fig 4.1, for simplicity we assume a parametric statistical framework - by this we mean that the sample data comes from a population that follows a probability distribution based on a fixed set of parameters [25].

Consider a simple classification model: let the data points X $\in \mathbf{X} \subset R^d$ and its label Y $\in [M]$ be distributed according to $p(x, y|\theta)$ where $\theta \in \Lambda \subset \theta$ indexes a parametric family of distributions $\mathbf{D} = \{p(x, y|\theta) : \theta \in \Lambda\}$.

Suppose there is a gennie that chooses $\theta \in \Lambda$ - this is analogous to choosing each source symbol in a sequence randomly according to distribution p(x) as described in last section. The learning machine (like the decoder) obtains a sequence of n samples $Z^n = (X^n, Y^n)$ where each pair $Z_i = (X_i, Y_i)$ is drawn i.i.d according to $p(x, y|\theta)$ [23]. The task of the learning machine is to select a classifier defined by [23]:

$$w : \mathbf{X} \to [M]$$

(this is the analogy of the reconstruction problem described above). Just like in the encoder-decoder analogy, there is need to define a performance metric.

To do this, define $q(\theta)$ to be a prior distribution over the parametric family. If $q(\theta)$ is the true distribution over the parameter space, then the performance metric collapses to average loss over many instances of learning problem. Furthermore let,

$$W(y|x, \theta = p(y|x, \theta)$$

describes a regression function that takes in x $\in \mathbf{X}$ and produce as output M dimensional vector of probability that x belongs to class y. Then $\hat{W}(y|x)$ is an estimate of W. The loss is then defined as L_p distance between the M-dimensional vector formed by $W(.|x; \theta)$ and $\hat{W}(.|x)$[23]:

$$L_p(x, \theta; W, \hat{W}) = \left(\sum_{y=1}^{M} |W(y|x, \theta) - \hat{W}(y|x)|^p \right)^{\frac{1}{p}}$$

The L_p loss provide a comprehensive sense of classifier performance. Hence, the justification for using this in the model - this will be discussed in later section. Lastly, the Kullbeck-Leibler divergence between W and $\hat{W}$ [23]:

$$L_{KL}(x, \theta; W, \hat{W}) = \sum_{y=1}^{M} W(y|x, \theta) \log\left(\frac{W(y|x, \theta)}{\hat{W}(y|x)}\right)$$

This corresponds to the cross entropy loss or log loss mentioned in the previous chapter. The pinker's inequality states that:

$$L_{KL} \geq \frac{L_1^2}{2}$$

Hence there is a relationship between L1 loss and the L_p loss. Thus using this L_{KL} is perfectly valid. Moreover a lower bound on L_1 risk can be translated to a bound on the L_{KL} divergence between W and $\hat{W}$ averaged over θ [23].

The following lemma follows from the rate distortion analysis : There exist a Bayesian learning rule with Bayes Risk $< \epsilon$ only if

$$I(X^n, Y^n; \theta) \geq \min_{\substack{p(W|\hat{W}) \\ E[L_1(W|\hat{W}] \leq \epsilon}} I(W; \hat{W})$$

where I is used to denote the mutual entropy. Thus to achieve a given low error rate the mutual information on the left hand side which is a function of the joint distribution (X,Y) must be greater than mutual information from the map rule. As will be shown seen, this is analogous to the information bottleneck - information bottleneck is a function of jount distribution of X,Y. The aim of training is to make ϵ as small as possible and make learning approach the right hand side. As will be shown in the later section of this chapter, when this error becomes small, an efficient representation of input is then possible - thus the model becomes more optimal.

- Summary of results [23]: Following the analysis in [23], it has been shown that , in order to derive the average L_p error below a threshold ϵ the mutual information between training samples and the parameters θ must be at least as great as the differential entropy of the posterior plus a penalty term that depends on ϵ and a sample theoretic term quantity called the interpolation dimension .The interpolation dimension is the cardinality of the smallest (finite) interpolation set. It measures the number of data points from the posterior distribution needed to uniquely interpolate the entire function [23].

To give an overall picture ,the true model parameter θ is not always known a priori. However, we can choose a parameter say $\hat{\theta}$ from a parametric family $q(\theta)$ and try to estimate how well it describes the true parameter which describes the data or model - the Kullbeck divergence is meant to test this "how well".

In summary, it is explained (statistically speaking) that the problem at hand is same as trying to model a stochastic phenomenon by a regular parametric family of

distributions $\mathbf{F} = \{F_\theta : \theta \in \Theta\}$ where $\Theta \in R^p$. However, we observe "N" iid outcomes from the phenomenon $\{X_1...X_n\} \overset{iid}{\sim} F_\theta$ for some $\theta_0 \in \Theta$ (for example consider the image classes randomly selected) but do not know/observe the $\theta \in \Theta$ that generated them - we don't observe the weights and biases that will correspond to a correct classification of the randomly selected images by the algorithm. The obvious inference task is : how can the θ that generated the sample $\{X_1...X_n\}$ be estimated? Since the $\{X_1...X_n\}$ is all we have, we can use a function of the input $\{X_1...X_n\}$ - called estimator [2]- to learn this . The estimator, used in this project is the Conv net and the parameters would correspond to the weights and biases discussed in chapter 2. The KL distance is used to test how well the functional approximator is doing. This is similar to the reconstruction problem discussed in the last subsection.

However wouldn't it be nice if the conv net can be opened to observe how it works? The next section does that.

4.2 Opening the Neural net Black Box

First, the section starts with discussion on Information Bottleneck and then proceed to leverage this concept to have a deeper understanding of how the black box (neural net) described in the previous chapter works in depth.

4.2.1 Information Bottleneck (IB)

The Information bottleneck is introduced here, to express the relevant information that an input random variable $X \in \mathcal{X}$ contains about an output random variable $Y \in \mathcal{Y}$. Given the joint distribution $P(X, Y)$, this defines the mutual information $I(X; Y)$ [24].

Statistically, it represents the minimum sufficient statistics of X wrt Y, denoted $\hat{X}$ [26] - the minimmum representation that captures the mutual information. Assuming the markov chain

$$Y \to X \to \hat{X}$$

From Data processing inequality[24] we have:

$$I(X : \hat{X}) > I(Y : \hat{X})$$

Thus finding an optimal representation (or reconstruction) is equivalent to the Lagragian minimization:

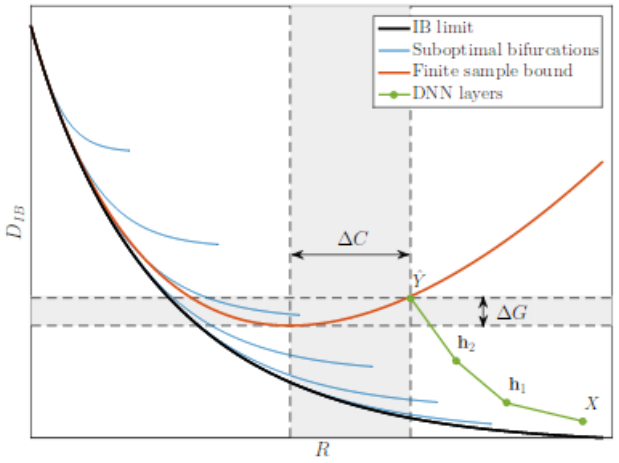
[2]the estimator is required to be sufficient statistics for learning the true parameters

$$\mathcal{L} = I(X : \hat{X}) - \beta I(Y : \hat{X})$$

where β is tradeoff parameter between representation rate, $R = I(X : \hat{X})$ and relevant information $I(Y : \hat{X})$ [26].

Of relevance to this project is the IB curve, shown below. The IB curve shows that it is possible to have bifurcations to sub optimal curves at critical values of β. This will be very important in explaining why some models might be better for certain object scale than others - in essence some models might perform sub optimally.

Figure 4.2: Information curve for Deep Neural net



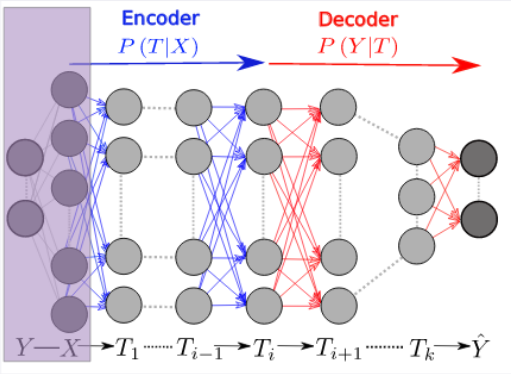
[The black line is the optimal achievable IB limit.The blue line are sub-optimal bifurcations obtained by remaining in the same representation. The green line shows hypothesized path for the hidden layers. Though the training distortion may be low (green line) the actual distortion may be as high as the red line. The aim in training model is to shift the DNN layers as close as possible to the optimal curve and obtain lower complexity and good generalization. **Source**[26].]

The bifurcations represent the phase transitions between different topological representations of $\hat{X}$ and are properties of joint distribution independent of the model [26]. At optimal point, the neural model should be able learn to extract the most efficient informative features, or approximate minimal sufficient statistics, with the most compact architecture[26]. In fact as will be shown in later chapter varying the size of the hidden layer will affect model accuracy and give better representation of $\hat{W}$ for different object class input.

4.2.2 Stochastic Gradient Descent and Information Bottleneck

The figure below will be important for this section:

Figure 4.3: Markov View of DNN layers



[Image source [27].]

This needs little explanation. The layers labelled T form the hidden layers and the arrows signifies they form a Markov chain of successive internal representation of input X [27]. The desired output layer is Y; it is seen only during training,through a finite sample of the joint distribution, $p(X, Y)$, and is used for learning the connectivity matrices between consecutive layer . The output layer is $\hat{Y}$.

The representation is grouped into an encoder, $P(T|X)$, quantified by

$$I_x = I(X;T)$$

and a decoder, $P(Y|X)$, quantified by :

$$I_y = I(T;Y)$$

Interestingly, the IB can be viewed as representing the optimal representation: that compress the input for any desired mutual information on the output Y. The total information stored in the network is given : $\frac{I(Y;\hat{Y})}{I(X;Y)}$ [27].

The information path along the network follows directly from the Data processing inequality [24], as shown below:

$$I(X;Y) \geq I(T_1;Y) \geq I(T_2;Y) \geq I(T_3;Y) \geq I(T_4;Y)...I(T_k;Y) \geq I(\hat{Y};Y)$$

- Summary of results from [27]: In [27] it is shown that Stochastic Gradient Descent with which our model will be trained consist of 2 different phases:

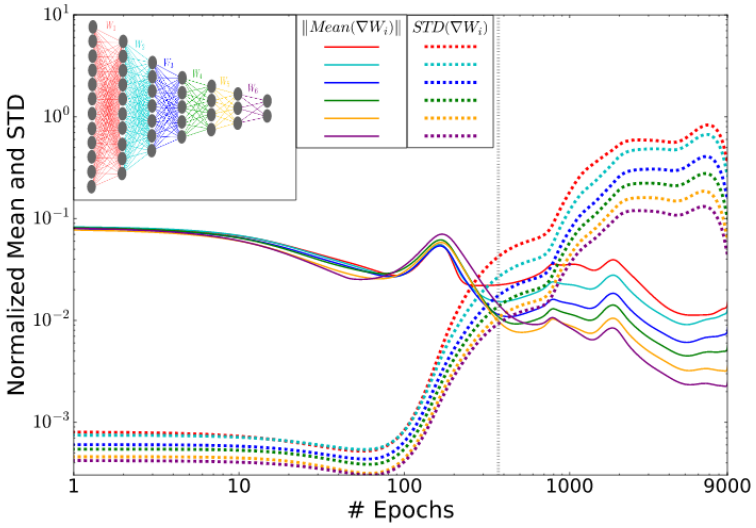
– *empirical error minimization phase (ERM) or drift phase*

– *compression phase or diffusion phase*

The ERM last for a few hundred epochs, and during this phase the neaural layers tend to increase the information on the labels (increase I_Y) while preserving the Data Processing Inequality order. In the compression phase, gradient fluctuation is high, causing reduction in mutual information on variable X.

- The cost of the training epochs in standard deep learning are spent on compression of the input to efficient representation and not on fitting the training labels
- The representation compression phase begins when the training errors becomes small. Additionally, the Stochastic Gradient Decent (SGD) epochs change from a fast drift (smaller training error) into a stochastic relaxation, or random diffusion, constrained by the training error value .
- Adding hidden layers reduce the number of training epochs for good generalization.

Figure 4.4: Graph of Stochastic gradient descent phases across epoch



[Image source [27].]

Above is a plot of the normalized mean and standard deviations of the weights stochastic gradients (in the samples batches), for every layer of a DNN (shown in the inset), as function of the SG epochs.

The grey line marks the transition between the first phase, with large gradient means and small variance and the second phase, with large fluctuations and small means (diffusion, low SNR) Shwartz-Ziv and Tishby [27].

4.3 Conclusion

In this section, an information theoretic view of the internal mechanism of the model is presented. This gives us a good theoretical underpinnings with which behavior of the models in the next chapter will be explained.

Indeed the topic in this chapter represent a recent development in deep learning. It is believed that in the nearest future deeper analysis of some of the behavior of the network will be available.

Chapter 5

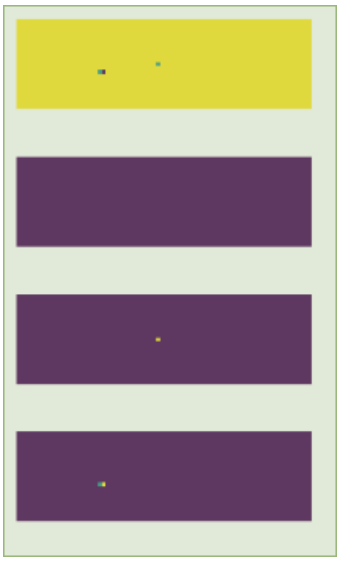
Experimental Results

In this chapter, the results of the detection model training is discussed. The chapter starts with additional optimization techniques used in actual implementation of the model. This is followed by an results on the behavior of the model performance as the sale of the object varies. Lastly, more results from different runs of the model will presented. The explanation of the model behavior will rely mostly on theoretical analysis from previous chapter and in some cases discussions on discussions from other chapters. This part of the project put everything together to achieve implementation results.

5.1 Optimization Technique Used in Implementation

Generally the label (desired output,Y, used during training) of the network consist of a image mask for different object classes plus a mask for background (negatives).

Figure 5.1: Input



[The label used for the network is a image mask. The negative signifies the background - plain purple of second rectangle.]

This makes the entire model discussed in this chapter quite different from other public models - it is not known either if this difference might help performance. The object classes considered are : Small pedestrian, big pedestrian, cars, bicycles.

5.1.1 Dropout And learning rate

Dropout technique has been effective in reducing model overfitting. A dropout of 0.5 was used generally. The effect of the drop out is to make the model implementation able to generalize and force the network to be more accurate. This is done by dropping out input and output connection of neural unit with probability set to 0.5. This is equivalent to randomly sampling "a network" from the full network.

A learning rate of 0.5 was used during training. The update of weights is generally given as $W := W + \Delta\mathcal{L}$, where $\mathcal{L}$ is the cost function, Δ is the learning rate, and W the weight. Thus the learning rate, determines how the model parameters are updated - or how often the model learns.

In some implementation, an exponentially decaying learning rate is used. This is defined as :

$$\text{learning rate} \times \text{decay rate} \times \frac{global\_steps}{decay\_steps}$$

Other parameters aside the learning rate are arbitrarily tweaked to give optimal performance.

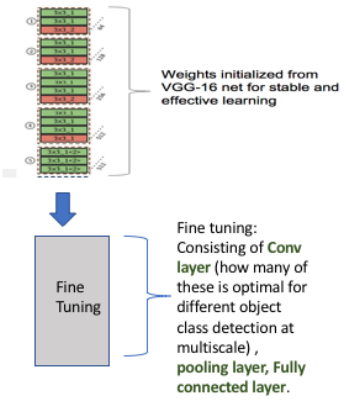
5.1.2 Data Augmentation

Since the project takes a data driven approach to object detection. One way to solve the scarcity of data for learning the large number of model parameters is through data augmentation. The general approach consist in using simple geometric transformations (such as rotating, shifting). In the initial training of the models, no augmentation was used but later training were done with augmented set

5.1.3 Weight Initialization

Conv net has huge parameters which might be difficult to train. To facilitate training and convergence, the parameters are initialized using a pre-trained model - specifically VGG 16 [28]. Pr3e-trained weights have been shown to give better performance [29].

Figure 5.2: Model Finetuning



[With the initialisation set as above, the implementation focuses on finetuning training for object detection.]

5.1.4 Mini Batch Stochastic gradient

Model parameter update is achieved through gradient descent using the back propagation algorithm. The (stochastic) batch gradient descent provides a more optimal way to train. The bacth size is set to 64 for each training epoch

5.1.5 Batch Normalization

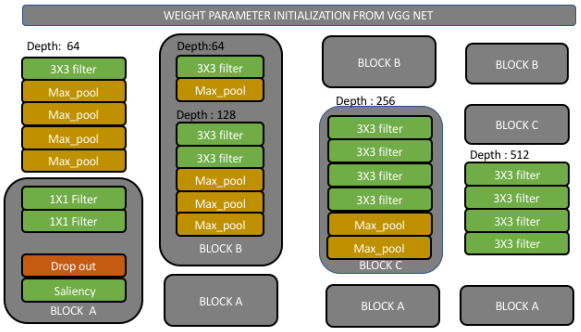
Batch normalization has been shown to improve improve covariant shift [30], reduce dependence of gradients on parameter size or initial values[30]. Thus for each training epoch and batch size 64 we perform batch normalization as below where x is input data in a batch:

$$\hat{x_i} \leftarrow \frac{x_i - mean_b}{\text{standard\_deviation}_b}$$

5.2 Model Results

5.2.1 First Model Results

Figure 5.3: Network Architecture

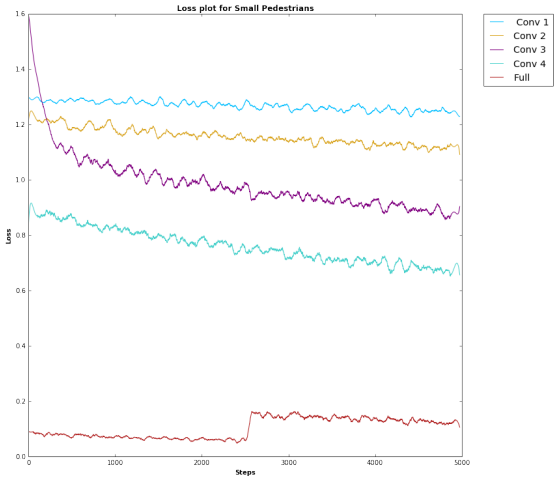


[From left to right the architecture is referencede as conv 1, conv 2, conv 3, conv 4. From left to right observe that the network depth is increasing. The aim is to find which architecture better captures varous object class]

The aim of this architecture is to understand the variation of model accuracy for multiscale object as the conv layer becomes deeper. From left to right, the figure shows a progression from a simple model to a more deeper architecture. Each architecture was trained separately. Modules that are repeated are made into blocks. They are then used as part of the next architecture. A full network consisting of the Depth:512 block above replicated twice was also trained . A plot of error was made for each model. The plot was smoothened using savizkygolay[1] filter using a window size of 51 and polynomial of degree 3. For these architectures, Adam gradient Optimizer was used with a decaying learning rate.

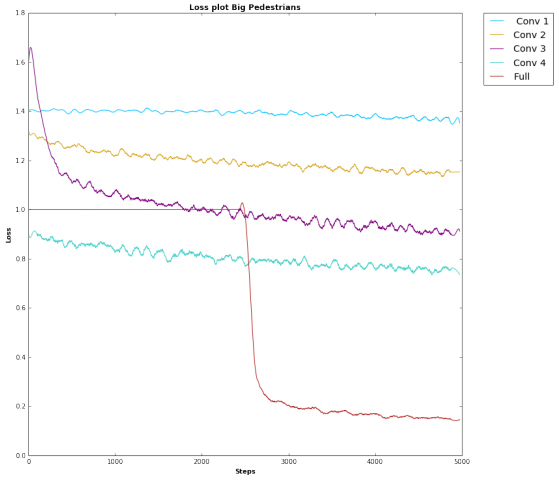
[1] "https://en.wikipedia.org/w/index.php?title=Savitzky%E2%80%93Golay_filter&oldid=807248668"

Figure 5.4: Cross entropy loss for small pedestrians



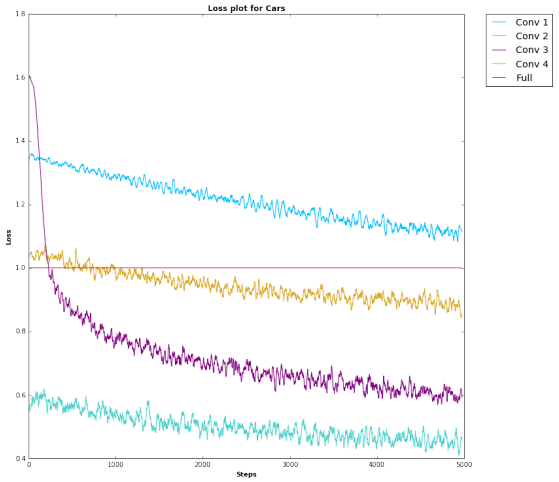
[Each of the architecture in fig 5.3 has been labelled conv 1 to 4 in this figure. The last architecture is the full layer.]

Figure 5.5: Cross entropy loss for Big pedestrians



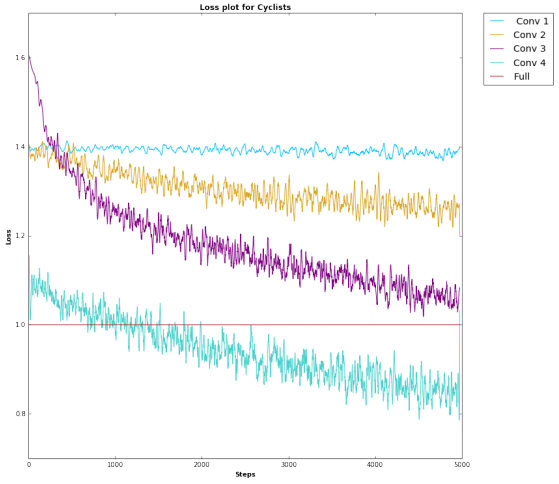
[Each of the architecture in fig 5.3 has been labelled conv 1 to 4 in this figure. The last architecture (not shown) is the full layer.]

Figure 5.6: Cross entropy loss for small pedestrians



[Each of the architecture in fig 5.3 has been labelled conv 1 to 4 in this figure. The last architecture is the full layer]

Figure 5.7: Cross entropy loss for cyclists



[Each of the architecture in fig 5.3 has been labelled conv 1 to 4 in this figure. The last architecture is the full layer]

The following conclusions can be made:

- The Full network is not always the best for all class labels
- The full network performs well on at least small and (after long iteration) big pedestrians
- For Cars Conv 4 performs absolutely better

Using the understanding from last chapter it is possible to explain the model behaviour. The total mutual information the full network retains $I(X^n, Y^n; \theta)$ about the sample input is much more than other models - since the parameter size θ is bigger. This induces a small training error. With a small error , entering into the compression phase during batch training will result in good internal representation of object classes. Since the Conv 4 (the last architecture in 5.3 is quite similar to the full network it is possible for similar performance to occur.

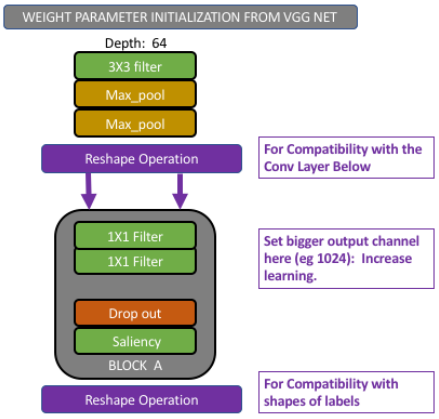
Since the models conv 1, conv 2,and conv 3 have less hidden units. Their poor performance can be attributed to bifurcation to suboptimal region in the information bottleneck curve during training. Thuis might have induced a greater error on the model. Thus, the compression phase will not result in good internal representation of data. Note that they also have less parameters than the conv 4 and full network

In particular the architecture in fig 5.3 above has serious caveat: the model architecture are in order of decreasing weight parameters. This causes the models with more parameters to perform well. In partiucular there is heavy use of max pooling operation which might affect model performance for smaller architecture. The next model has same architecture as conv 1 but with increased parameters at the last layer. This causes it to have similar performance with the full model.

5.2.1.1 Single Conv 1 Model Architecture

Indeed we need a model that it simple to train for use in real time object detection. The model in this section consist of single model architecture (similar to first architecture,conv 1, in 5.3) that works well on all object classes.

Figure 5.8: Single Conv architecture



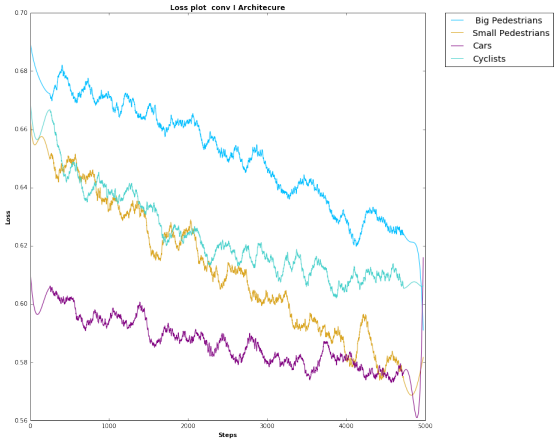
[New Architecture 1 with good performance]

Again, using the understanding from the last chapter, the total information stored in the network is $\frac{I(Y;\hat{Y})}{I(X;Y)}$. In additon we know that since the parameter space is now much larger,the joint mutual information $I(X^n, Y^n; \theta)$ will be high. Expanding this gives :

$$I(X^n, Y^n; \theta) = I(\theta; X^n) + I(Y^n; \theta | X^n)$$

It is easy to infer that this will also cause an increase in IB: $\mathcal{L} = I(X^n : \hat{X}^n) - \beta I(Y^n : \hat{X}^n)$ since the IB depends on the joint distribution of X,Y. This in turn causes the information stored in the network $\frac{I(Y;\hat{Y})}{I(X;Y)}$ to be high, and a good output layer representation is thus delivered.

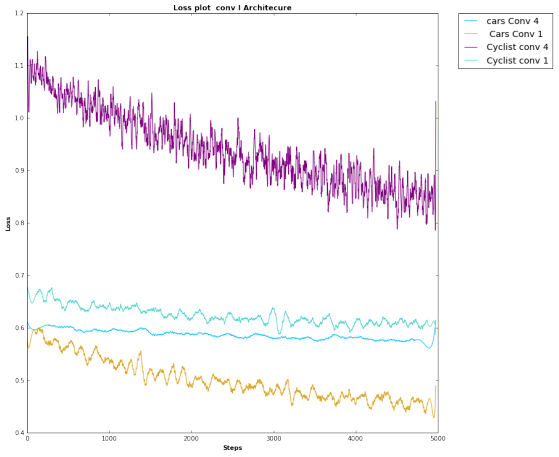
Figure 5.9: Training Loss plot for New Conv 1 Architecture



[Observe the overall good performannce on all lables]

In the figures below the performance of this model is compared with that of the best performance obtained in last architecture. Conv4 performed bast for class object cars and cyclist, thus it desirable to compare conv 4 for these object classes and the new conv1. The result is the figure below:

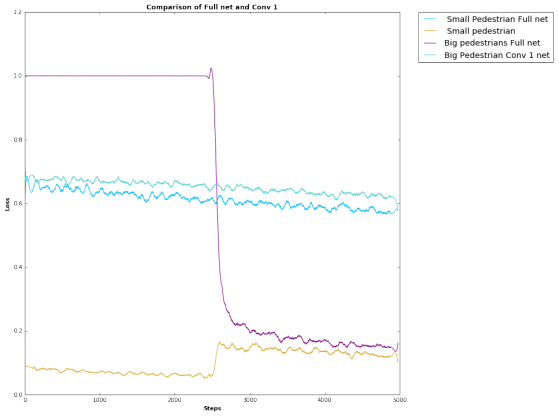
Figure 5.10: New Conv1 versus Conv4: Training loss



[Object classes cars and cyclist were considered here]

Similarly the full network performed best for class object big and small pedestrians , thus it desirable to compare full network for these object classes and the new conv1. The result is the figure below

Figure 5.11: New Conv1 vs Full network: Training loss



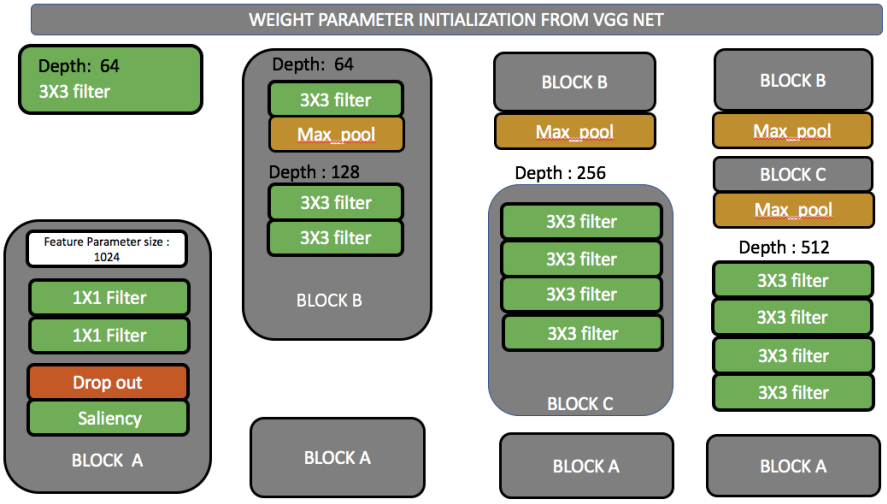
[Object classes Small and Big Pedestrians were considered here]

5.2.2 Second Model Results

The model in this section is motivated by the fact that it is not usually desirable to reduce the feature parameters : as done in this previous section for the smaller architecture using the max pool operation. The implementation uses mean absolute error (MAE) to measure model performance and reduces the max pool operation on smaller architectures. Rather than decrease the parameter size, the function tf.nn.max_pool - from tensorflow library - was used to pool features from the labels and adjusting label

sizes to the prediction output. Apart from these differences, the model architecture remains same as illustrated fig 5.3.

Figure 5.12: Mean absolute error for training



[New model architecture retaining feature parameter size.]

The Mean absolute error plot for this architecture is shown below:

Figure 5.13: MAE Training Error plot

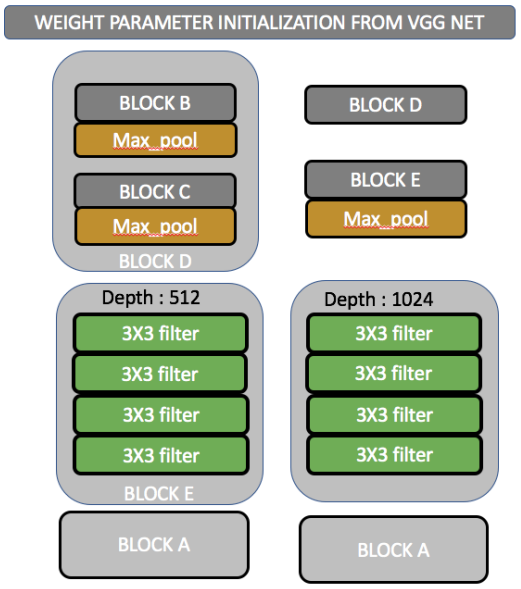
[x axis represent the training epoch]

The labels from onet1 to onet4 represent each architecture in fig 5.12 from left to right. The onetf represent the full network. Next, the explanation for be observed behavior goes along the line of the previous explanation using the IB, mutual information and parameter size arguments.

5.2.3 Augmented Model

The model here improved on the full net by increasing the depth of the full network to 1024. This also implies the entire previous model have to be adjusted for augmented parameter input - the model were built earlier with maximum parameter initialization size of 512 in mind. The model architecture is shown below:

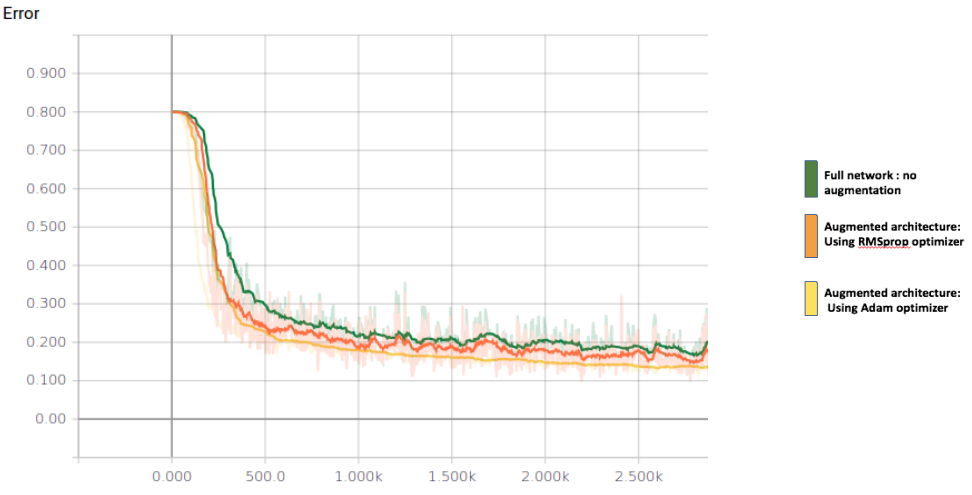
Figure 5.14: Mean absolute error for training



[from left to right the onet4 to augmented model. The depth 1024 is why this model differs from others]

The error plot for this model is shown below:

Figure 5.15: MAE training error plot for Aumented Architecture and full net



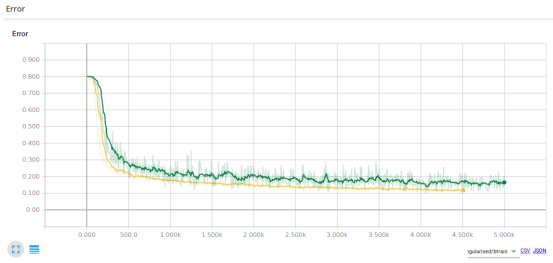
[x axis represent the training epoch]

The explanation for model performance is same as earlier explained.

5.2.3.1 Augmented Regularized Model

We present the augmented regularized model. Regularization helps model performance and convergence as discussed in chapter 2. Below is the plot for the regularized model:

Figure 5.16: Regularized vs unregularized model



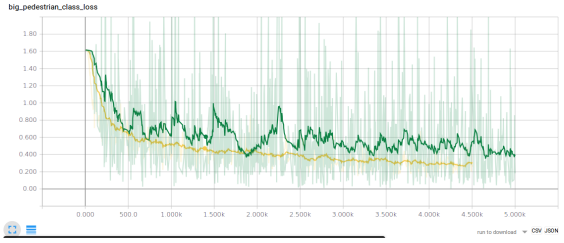
[The regularized model is in yellow. x-axis is the epoch]

The best way to observe the effect of regularization is to observe the error plot for various object classes:

Figure 5.17: Regularized vs unregularized MAE error for bicycles and cars

[x axis represent the training epoch. Regularized augmented model in yellow]

Figure 5.18: Regularized vs unregularized MAE error for Big pedestrian



[The regularized model is in yellow. x-axis is the epoch]

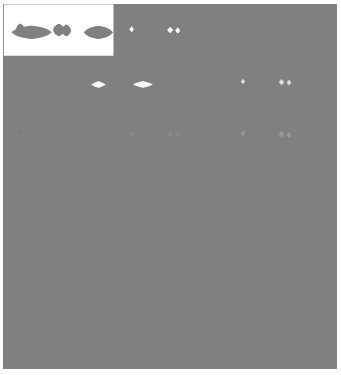
Appendix A

Appendix : Model Image Capture From Tensor board

Some noteworthy plots and images generated in tensorboard will be documented here:

A.1 Training Images

Figure A.1: Training label Captured during training.

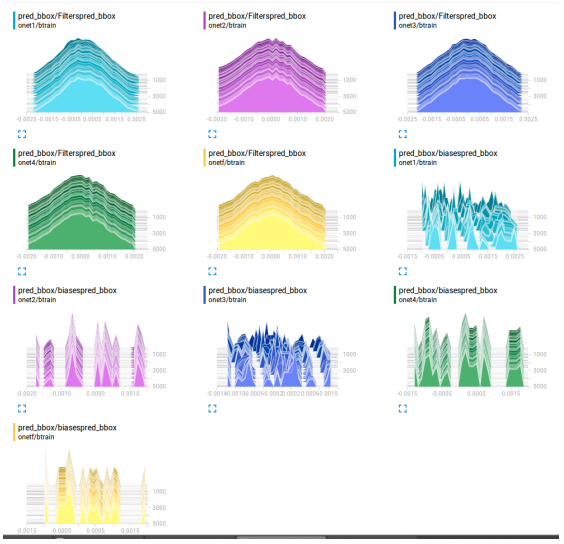


[mask are use as labels]

Figure A.2: View of a transformations of input image

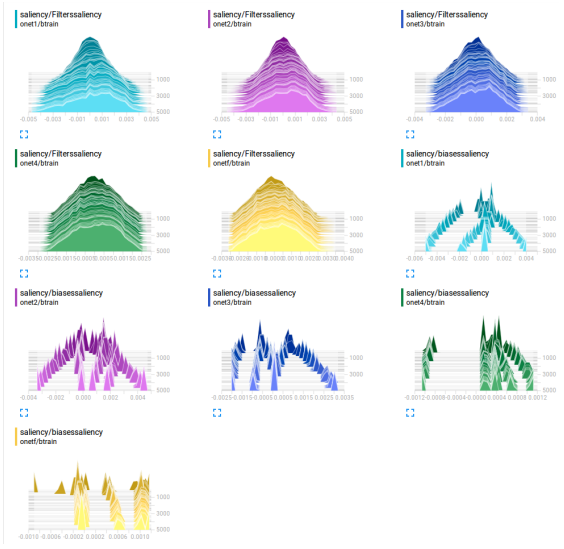
[This is captured from the network. The pattern of objects in the image are quite visible. This shows a gradual capture of object features as the image goes through the network]

Figure A.3: histogram plot of prediction box parameters



[The histogram plot assumes a normal distribution which is good.]

Figure A.4: histogram plot of prediction parameters for object classes



[The histogram plot assumes a normal distribution which is good. Normal distribution help in analyzing errors. The biases represent short stub in the last 5 plots, they might be equivalent to the separate object classes. Each plot corresponds to one of the architecture onet to one4 to augmented model]

Figure A.5: Image input to the network

[The model should detect each of the object classes in the image]

Bibliography

[1] Martin Weinmann. *Reconstruction and Analysis of 3D Scenes: From Irregularly Distributed 3D Points to Object Classes*. Springer Publishing Company, Incorporated, 1st edition, 2016. ISBN 3319292447, 9783319292441.

[2] Wikipedia. Autonomous car — wikipedia, the free encyclopedia, 2017. URL `https://en.wikipedia.org/w/index.php?title=Autonomous_car&oldid=796221467`. [Online; accessed 19-August-2017].

[3] Hong Cheng. *Autonomous Intelligent Vehicles - Theory, Algorithms, and Implementation*. Advances in Computer Vision and Pattern Recognition. Springer, 2011. ISBN 978-1-447-12279-1. doi: 10.1007/978-1-4471-2280-7. URL `https://doi.org/10.1007/978-1-4471-2280-7`.

[4] Emanuele Trucco , Alessandri Verri. *Introductory techniques for 3D vision*. Prentice-Hall, New Jersey, April 1998.

[5] Simon J. D. Prince. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 2012. doi: 10.1017/CBO9780511996504.

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[7] Sungdae Sim, Juil Sock, and Kiho Kwak. Indirect correspondence-based robust extrinsic calibration of lidar and camera. *Sensors*, 16(6), 2016. ISSN 1424-8220. doi: 10.3390/s16060933.

[8] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34: 2189–2202, 11 2012. ISSN 0162-8828. doi: 10.1109/TPAMI.2012.28. URL `doi.ieeecomputersociety.org/10.1109/TPAMI.2012.28`.

[9] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL `http://arxiv.org/abs/1504.08083`.

[10] Convolutional neural net for vision recognition, April 2017. URL http://cs231n.github.io/classification/. [Online: accessed on 22 August 2017, CS231N, Stanford University].

[11] R Hecht-Nielson. Theory of back propagation neural network. *Neural networks*, supplement 1, 1998.

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

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[14] Andrew Ng. Deep learning specialisation course, coursera, 2016. URL https://www.coursera.org/learn/convolutional-neural-networks/lecture/6UnU4/convolutional-implementation-of-sliding-windows. [Online; accessed 19-December-2017].

[15] pixabay. Cat.kitten, 2017. URL https://pixabay.com/en/cat-kitten-cat-baby-young-cat-1074644/. [Online; accessed 19-December-2017].

[16] Digital Image. Funny cat dog, 2017. URL https://i.ytimg.com/vi/daaoTw4VtGE/maxresdefault.jpg. [Online; accessed 19-December-2017].

[17] Digital Image. Cat.kitten, 2017. URL http://www.zastavki.com/eng/Auto/Audi/wallpaper-64028.htm. [Online; accessed 19-December-2017].

[18] Digital Image. road, 2017. URL http://www.peency.com/images/2015/05/13/road-sunset-wallpaper.jpg. [Online; accessed 19-December-2017].

[19] Digital Image. car, 2017. URL http://felixwong.com/2017/01/cars-in-cuba/. [Online; accessed 19-December-2017].

[20] Digital Image. car, 2017. URL http://pbrown.co.uk/photos/photos-5/files/page7-1007-full.html. [Online; accessed 19-December-2017].

[21] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using

convolutional networks. *CoRR*, abs/1312.6229, 2013. URL `http://arxiv.org/abs/1312.6229`.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. URL `http://arxiv.org/abs/1512.02325`.

[23] Matthew Nokleby, Ahmad Beirami, and Robert Calderbank. A rate-distortion framework for supervised learning. *IEEE International Symposium on Information theory*, 09 2015.

[24] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006. ISBN 0471241954.

[25] Wikipedia. Autonomous car — wikipedia, the free encyclopedia, 2017. URL `https://en.wikipedia.org/wiki/Parametric_statistics`. [Online; accessed 19-Dec-2018].

[26] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. *CoRR*, abs/1503.02406, 2015. URL `http://arxiv.org/abs/1503.02406`.

[27] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

[29] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. *Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 153–160, 2009. URL `http://jmlr.csail.mit.edu/proceedings/papers/v5/erhan09a/erhan09a.pdf`.

[30] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, and Gang Wang. Recent advances in convolutional neural networks. *CoRR*, abs/1512.07108, 2015. URL `http://arxiv.org/abs/1512.07108`.