Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

RM-NET: RASTERIZING MARKOV SIGNALS TO IMAGES FOR DEEP LEARNING

by
Kajal Gupta

Statistical machine learning approaches are quite famous for processing Markov signal data. They can model unobserved states and learn certain characteristics particular to a signal with good accuracy. However, with the advent of Deep learning the novice ways of solving a problem has shifted towards this more sophisticated algorithm, which is much better, powerful and more accurate. Specifically, Convolutional Neural Nets (CNN) have shown many promising results on images and videos. Here we illustrate how CNN can be applied to a 1D numeric signal using signal rasterization technique. We start by rasterizing a 1D numeric Markov signal into an image followed by applying CNN to perform two basic tasks: signal classification and error localization. We call this process as RM-Net. We demonstrate the performance of our approach on simulated data benchmarked against baselined statistical models. We also illustrate the supremacy of our technique on real word dataset 1000 Genomes Project Phase 3 SV where we try to estimate the location of Copy Number Variant (CNV) in a chromosome. Finally, we conclude using the metrics obtained on both the datasets that our proposed approach is much better, shows promising results and has scope for future improvements over traditional statistical machine learning approaches.

RM-NET: RASTERIZING MARKOV SIGNALS TO IMAGES FOR DEEP LEARNING

by
Kajal Gupta

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Data Science

Department of Computer Science

May 2021

APPROVAL PAGE

RM-NET: RASTERIZING MARKOV SIGNALS TO IMAGES FOR DEEP LEARNING

Kajal Gupta

Dr. Zhi Wei, Dissertation Advisor Professor of Computer Science, NJIT Associate Chair for Graduate Studies of Computer Science, NJIT	Date
Dr. Usman Roshan, Committee Member Associate Professor of Computer Science, NJIT	Date
Dr. Antai Wang, Committee Member Associate Professor of Mathematical Sciences, NJIT	Date

BIOGRAPHICAL SKETCH

Author: Kajal Gupta

Degree: Master of Science in Data Science

Date: May 2021

Undergraduate and Graduate Education:

- Master of Science in Data Science,
  New Jersey Institute of Technology, Newark, NJ, 2021

- Bachelor of Technology in Information Technology,
  JSS Academy of Technical Education, Noida, India, 2016

Major: Data Science

ACKNOWLEDGMENT

The process of doing research is difficult and certainly cannot be done alone. Two years that I spent at NJIT concluded with the most fruitful work that I will treasure for life and has given me something to be proud of when this journey ends. I would like to sincerely thank those people who provided me their guidance, support, and constantly kept me motivated even in my tough times.

First and foremost, I would like to thank my thesis advisor Dr. Zhi Wei, who believed in me and under his guidance got the opportunity to learn and enhance my knowledge. There have been undoubtedly numerous times when I felt lost in my research and didn't know what to do, but during those times he would be there helping me to move forward in the right direction. I always felt reassured that if something goes wrong he would be there to guide me and walk me through the crucial facts that I might have overlooked. I would really like to thank him and show my earnest gratitude for all his help, advice, encouragement and expertise.

Next, I would like to acknowledge my friends Sonal Gupta, Kajal Singh, Deepak Kumar, Abhishek Sachan and Vikas Gupta for constantly being there for me all the times. Although they were thousand miles far, they were always there to turn to. I would like to thank my US friends Tanmay Gupta and Ankita Talwar who were here at NJIT and walked together with me.

I would like to thank mom and dad for loving and trusting me and rooting for me from behind. Lastly, I thank God, for giving me all the wonderful opportunities to be able pursue things that I cherish and value the most in life.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Objective

The main objective of this thesis is to present a novel approach of solving an old statistical problem[1, 2] of signal processing using techniques of Computer Vision[3] and Deep Learning[4]. It has been almost a few decades when HMM[5] was proposed to find underlying hidden states in an observation sequence and models such as SVM[6] and decision trees[7] were proposed to perform classification to classify these signals as per state models. However, with advent of deep learning, there has been a revolution[8] on how a statistical problem which once thought to be solved only using above methodologies can be modified to be solved using more robust algorithms such as neural networks[4]. Hence, in this work we explore deep learning algorithms to build models that are capable of understanding these signals composed of numeric observation sequences following Markov Processes, identifying its hidden states and using their knowledge of representation learning[9] to not only classify the signals on basis of presence or absence of these states but also localize[10, 11] where exactly those states were found in the given signal, if present. We propose a technique of signal rasterization where we convert a numeric signal to an image and then apply computer vison techniques to solve the problem at hand. Finally, we present our comprehensive findings on simulated data and close our discussion by sharing the results on a real-world dataset 1000 Genomes Project Phase 3 SV[12] to show superiority of our proposed method as compared to statistical machine learning algorithms.

1.2 Background Information

The use of existing knowledge of statistics in various fields such as computational biology and bioinformatics[13], gesture recognition[14], finance[15] and computer science[16, 17] has been widely popular since decades. Researchers, scientists and students have heavily relied on it to propose novel algorithms and insights[18, 19] using them on huge amounts of data. Though these machine learning algorithms are capable enough to make accurate decisions, they suffer with a drawback that they strongly rely on certain assumptions regarding the model they build and the data they work upon. These assumptions sometimes do not cater to actual representation of data and may lead to development of models that might be misleading or erroneous. Occasionally these errors can be ignored, while other times they may be catastrophic for example in cancer detection. With the introduction of deep learning models[20] the assumption regarding model parameters has been removed entirely and the responsibility has been shifted on algorithm to tune itself to the data while training. This not only removed the tedious task of feature selection and parameter searching to build accurate models, but also improved the accuracy[21] of generated deep learning models which were much better than its preceding statistical models.

Another important factor to consider while building a machine learning model is data. Working with real-world observable output needs attention since they might be holding unobserved patterns that could not be easily deduced. When such observations are exposed to classical machine learning algorithms which are not precisely crafted suiting the needs of these observations, those inherent properties that might be influencing greatly to the decision boundaries might be subsided and thus lead to reduced performance. The breakthrough in learning the representation of these unobserved features was achieved

through deep learning's representation learning capabilities. Deep learning methods are representation learning based methods obtained by composing simple but non-linear[22] modules thus holding the ability to learn very complex functions[23].

Convolutional Neural Networks[24] have shown significant improvements in performances in area such as natural language processing[25], speech recognition[26], object detection[27], image segmentation[28, 39], cancer detection[30], genomics[31] and many more. There are various forms in which one can employ CNN in their architecture. It can be in form of 1D[32] or 2D[24] CNN. 1D is employed mostly when we have observations in form of sequence such as signal. However, the drawback associated with 1D convolution is that it is not able to describe the relationships and dependencies in the observation and its neighbors since it works in one dimensional space. So, in order to exploit the relationships of a sequence as a function of its neighbors 2D convolution works best.

2D convolution works finest for images and videos which are inherently 2+ dimensions. It uses the concept of kernel[24] to learn the complex function exploring the spatial relationships between a pixel and its surrounding neighbors using convolution and parameter sharing. However, to work with 2D convolution one needs to represent the 1D data as 2D. There have been very few works where this concept has been explored. The first work around this was proposed in 2018 by Ma S. and Zhang Z. where they took advantage of deep learning for representing high dimensional omics data as an image. In their work, they rearranged the omics data in 2D space considering molecular features related in function, ontologies, and other relationships were organized in spatially adjacent and patterned locations[33]. Then they used deep learning models to classify the images. Although the results presented showed decent performances the main drawback was that it

used underlying information such as ontologies extracted from Kyoto Encyclopedia of Genes and Genomes and thus cannot be extended to data which are non-omics and have no ontology information.

In 2020, Bazgir O. et al.[34] proposed a feature representation approach termed REFINED to arrange high dimensional data in a form of image for the application of convolution neural networks. In this work two methods for representing high dimensional data as an image were proposed. First via random projection where each value in a vector was placed one after the other in an image matrix. In second approach they used PCA. In this they used the first two major eigen vectors of the data covariance matrix as the feature coordinate set and projected that as an image. This method showed promising results however the only drawback of this is that it is suitable for high dimensional features and both of their proposal cannot be applied to data with just one dimension such as a signal with hidden states.

The latest work that has employed the idea of converting N-dimensional information as an image and applying deep learning on it is DeepCNV[35] proposed by Glessner J. T. et al. in 2021. Although feature representation was not the primary focus of their research work, they did something similar while trying to reduce false positives in CNV (copy number variations) calls. For each CNV call two plots were generated, LRR scatter plot image and BAF scatter plot image. For both plots SNPs in candidate CNV were colored Red and SNPs in surrounding regions were colored blue. A deep neural network was trained to classify the images. The results showed improved performance and reduced false positives when above feature transformation was used. However, the above work is specific to genomics and cannot be extended to other data as is. Further since we don't

know the hidden states or any such visually distinct features, we cannot employ color coding the image and must deal with greyscale images.

So, in this thesis, we present a simple yet universal way of representing one dimensional numeric data as an image. We call it RM-Net: Rasterizing Markov signal, where each numeric observation of the signal is plotted on y axis against time on x axis. We then use deep learning techniques to perform classification and error segment localization which will help us distinguish the type of signal and locate the presence of an error segment in this 1D numeric signal, which we will be discussing in detail in the next two chapters.

CHAPTER 2

METHODS

2.1 Introducing Signal

We work with a Markov signal in our research. A signal is composed of sequence of numeric observations with some unobserved states and is drawn from a probability distribution. It usually consists of three components – signal length, Normal segment and an Error segment. Length is the span of signal for T time instances. At each time instance a signal can be in one of two states- Normal or Error state. These states have certain observation associated with it which is drawn from mutually exclusive probability distribution that the state follows. Typically, it is drawn from Normal distribution with certain parameters specific to states, however we conducted various experiments to study the effects of other distribution as well.

$Signal = O_1\ O_2\ O_3\ O_4\ O_5\ O_6\ O_7\ \ O_8 \dots \dots O_{L-1}, O_L$ where $O = \{N, E\}$; L = Signal Length

$$N \sim PDF(\mu 1, \sigma 1)$$

$$E \sim PDF(\mu 2, \sigma 2)$$

Signal can be of two types:

Type 1: Signal which contains observations from both Normal and Error states.

$$O = O_1\ O_2\ O_3\ O_4\ O_5\ O_6\ O_7\ \ O_8 \dots \dots O_{L-1}, O_L$$

$$S = N_1\ N_2\ N_3\ N_4\ E_5\ E_6\ E_7\ E_8\ E_9 \dots \dots N_{L-1}, N_L$$

Type 2: Signal which contains observations only from Normal state.

$$O = O_1\ O_2\ O_3\ O_4\ O_5\ O_6\ O_7\ O_8 \dots \dots O_{L-1}, O_L$$

$$S = N_1\ N_2\ N_3\ N_4\ N_5\ N_6\ N_7\ N_8\ N_9 \dots \dots N_{L-1}, N_L$$

We base our research on identifying the type of signal and further segmenting the error boundaries if Type 1 is identified i.e. an error segment is present.

2.2 Research Synopsis

In this research we propose a novel way of performing signal classification and error segment localization in a signal composed of numeric observations using signal rasterization technique. Given above signal we propose to convert it to an image as is and use deep learning models to learn the characteristics of the signal using spatial correlation amongst observations over time. The spatial correlation and the signal patterns are learnt by deep learning models as features and we use this representation learning to perform the tasks of classification and object detection aka error segment localization in our case.

The main goal of our research work is to compare the classical statistical machine learning approaches with the novel idea hypothesized using computer vision approach tapping deep learning. We aim to evaluate our computer vision proposed solution for two tasks:

Task 1: Given a *Type 1* and *Type 2* signal, will it be possible for a deep learning model to outperform classical machine learning models like HMM, SVM and Random Forests in signal classification after signal rasterization?

Task 2: Given *Type 1* signal will it be possible for a deep learning model to identify where exactly the error segment is present in the rasterized signal and if yes then with what accuracy?

We aim to find answers for above two tasks and for it we will be conducting series of experiments to compare computer vision model with statistical machine learning models.

2.3 Competing Methods

2.3.1 Classification

In machine learning, classification is considered as an instance of supervised learning and refers to the tasks of identifying the category which an observation belongs to. For example, in our case, classification would mean given a signal categorize it to either *Type 1* or *Type 2* which is nothing but goal of our research work aka *Task 1.*

Two-layer CNN finetuned on top of InceptionResNetV2

For solving *Task 1*, we propose a deep learning-based approach to solve the problem. We used Convolutional Neural Networks[24] which works upon our rasterized signal. Convolutional Neural Networks is a class of deep learning used for working with images and videos. Convolutional neural networks possess the qualities of shift invariance and translation invariance that makes them quite robust to the problem they are working upon in an image or video. During training they learn to optimize convolutional kernels or filters which represents knowledge of that problem. For our experiment we constructed a two-layer model with first dense layer of 128 neurons and second dense layer of 16 neurons finetuned[36] over top of InceptionResnetV2[37] thus transferring the skills learnt by

InceptionResentV2 model to our model instead of training the model from scratch. An output layer is connected to end of last fully connected layer with single neuron and sigmoid activation function. This neuron is responsible for predicting the class of given rasterized signal image. The loss is calculated as:

$$Loss = -\frac{1}{N}\sum_{1}^{N} y_i \,.\, log(y_i') + (1 - y_i) \,.\, log(1 - y_i')$$

Finally, the class decision is made as follows:

$$f^*(x) = \begin{cases} 1 & if\ p_x > 0.5 \\ -1 & otherwise \end{cases}$$

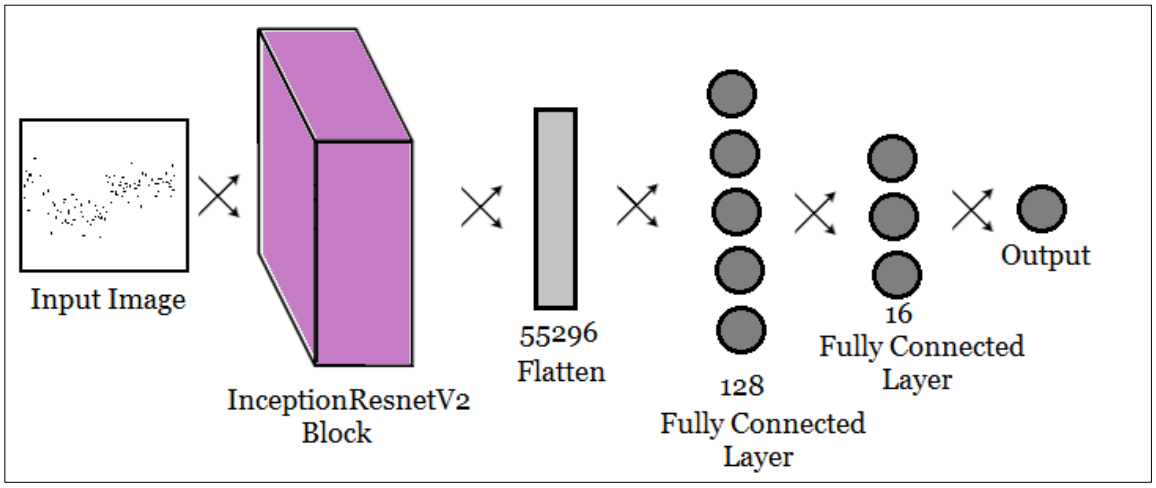

Figure 2.1 CNN with two fully connected layers finetuned over top of InceptionResnetV2.

Metrics for Classification

Since we have a binary classifier, we evaluated the models using accuracy and area under the receiver operating characteristic curve.

Accuracy measures the fraction of prediction that the model got right. It is defined as:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

where *TP= True positive, TN=True Negative, FP=False Positive, FN=False Negative*

AUC measures the area under the two-dimensional ROC curve. A ROC curve plots True Positive Rate (TPR) to False Positive Rate (FPR). This plot is used to find AUC which provides a measure of model's performance across all possible classification threshold. AUC ranges between 0 and 1, with 0 implying worst performance and 1 indicating best performance.

2.3.2 Error Segment Localization

In order to perform *Task 2* i.e. determine where exactly the error segment is present in the signal, we need to find the location of occurrence of an error segment in each signal. Given *Type 1* signal our goal is to accurately locate this error segment. In order to do this, we use computer vision technique as our proposed method for localizing error segment after signal rasterization. This localization of error segment is performed using object detection[36] in computer vision. Object detection is a technique that allows us to identify and locate the presence of an object in an image. In our case the object will be an error segment. In order to perform this error segment localization, we use YOLO (you only look once) framework[38, 39], a popular method for performing object detection in deep learning.

Deep learning model - YOLO (You only look once)

YOLO[38] is an object detection framework for deep learning which is very fast and accurate. It is different from RCNN[40] and Faster-RCNN[41] family which is based on two step

approach for object detection. First being region proposal and next assigning classes to the objects in that region. However, YOLO is a one step process that proposes bounding box of the objects and class probabilities for the objects simultaneously. It can propose multiple instances of objects present in the same image.

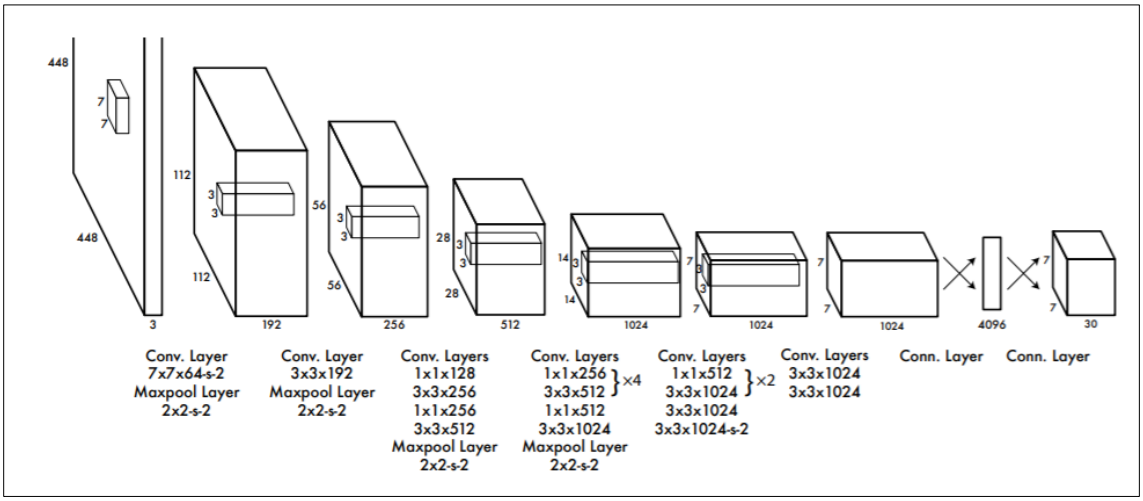

Figure 2.2 YOLO Architecture.

Source: [38]

YOLO starts by dividing the entire image into *S* x *S* grid. If the center of object falls in a grid then that grid is responsible for predicting that object. Each gird predicts B bounding boxes (x, y, w, h) along with its confidence score (c). Each grid also predicts class probabilities conditioned on object $P(class_i | Object)$. In order to predict multiple instance of object in a grid, YOLO uses concept of Anchor boxes. For each anchor boxes x, y, w, h, confidence score and class probability are predicted. The Loss function[38] is calculated as below:

$$Loss = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Since multiple boxes are predicted per grid, Non-Maximal suppression is used to suppress any boxes which has lower confidence score or low IOU with other boxes. Finally, we have non overlapping predicted bounding boxes for a given rasterized signal. These boxes are saved as predicted labels containing error for that signal and is later used to compare performance metrics against ground truth and HMM.

Metrics for Error Segment Localization

Unlike classification, estimating the performance for object detection is not straightforward. Once we have predictions we need to measure it against the ground truth. Our ground truth is the number of observations sequence that makes up an error segment. Since YOLO works on images we need one additional step to convert the boundary locations in graphic coordinates back to index mapping to each observation. Once we reverse transform these coordinates we can find out how many observations fall under this predicted boundary. Then we use Jaccard Index for measuring the model performance against ground truth. Jaccard Index is defined as follows:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Jaccard Index measures the similarity between two sets. It ranges between 0 and 1, with 0 indicating no similarity and 1 indicating sets being perfectly similar. We then use Jaccard Index to define whether a prediction is considered as true positive, true negative, false positive or false negative. We make this decision for per *Type 1* as signal follows:

$$F^{Type1}(O) = \begin{cases} TP & if\ J(A,B) \geq 0.5 \\ FN & if\ J(A,B) \leq 0.1 \\ FP & otherwise \end{cases}$$

Since *Type 2* signal doesn't contain an observation belonging to error states in ground truth, Jaccard Index will be 0 showing no overlap. But let's say model predicted many observations belonging to error state it will skew the performance if this is not penalized. Hence, we give a penalty if the number of observations predicted as being in error state for *Type 2* signal is greater than 10% of the signal length.

$$F^{Type2}(O) = \begin{cases} FP & if\ Length(O \in E) \geq 0.1 * signal\ Length \\ TN & otherwise \end{cases}$$

Once we have TP, TN, FP and FN, we use to calculate accuracy, precision, recall and F1-Score.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

2.4 Baseline Methods

Support Vector Machines

Support Vector Machines are most popular classification algorithms and has robust prediction methods. They can classify both linear as well nonlinear[42] data using kernel trick which makes them widely used and suited to various applications. Given observations with labels SVM tries to find the most optimal hyper plane that maximizes the distance between the hyper plane and the nearest data point. This distance is known as margin and the data points that lies on this margin and contributes to deciding the optimal hyper plane are called support vectors.

In our research since the observations belonging to two states are drawn from probability distribution function with different parameters, we projected them to higher dimensions using kernel trick. This allows the algorithm to work in the transformed space and find the optimal hyperplane. We used Radial Basis Function kernel commonly known RBF kernel. The RBF kernel on two samples **X** and **X'** is defined as:

$$K(X, X') = \frac{-\exp\left(\left|\left|X - X'\right|\right|^2\right)}{2\sigma^2}$$

We train the model on training dataset. The goal of the model is to optimize hinge loss. The optimization function with regularization is given as:

$$\left[\frac{1}{n}\sum_{i=1}^{n} max\big(0, 1- y_i(w^T x_i \ - \ b)\big)\right] + \ \lambda\|w\|^2$$

$\boldsymbol{\lambda}$ determines the tradeoff between increasing the margin and ensuring that X_i lies on the correct side. Once training is done, we test on test datasets. For a given observation in a test dataset and a model we predict the classes as follows:

$$f^*(x) = \begin{cases} 1 & if\ p_x > 0.5 \\ -1 & otherwise \end{cases}$$

Random Forests

Random forest is a supervised ensemble learning method available for classification as well as regression. Random forests add randomness to the model as compared to decision trees. Instead of searching the most important feature in the entire data to make a split, it searches for best features amongst random subset of features thus constructing multiple uncorrelated individual trees. It then outputs the class output as the mode of classes of those individually constructed trees. In our experiment we have used Random Forests of Python's Sklearn library with all default parameters. Given a test observation the final decision is made as follows:

$$f^*(x) = \begin{cases} 1 & if\ p_x > 0.5 \\ -1 & otherwise \end{cases}$$

Hidden Markov Model for Classification

A Hidden Markov Model is a statistical model which models a system following Markov processes. A Markov processes models a Markov chain which describes a sequence of event in which the probability of next event depends on the state attained before it. Formally, consider a sequence of state variables $q_1, q_2 \ldots q_i$, a first-order Markov model embodies *Markov Assumption* on the probabilities of this sequence:

$$\boldsymbol{Markov\ Assumption}: P(q_i = a|q_1 \ldots q_{i-1}) = P(q_i = a|q_{i-1})$$

In our signal we have set of observations however we don't know the underlying states which contributed to it i.e. which probability distribution model it was drawn from. That makes our model ***hidden*** as the events contributing to observations can't be observed directly. So, we use first-order Hidden Markov Model that helps us build a model with both the observed states as well as the hidden states. Along with *Markov Assumption*, HMM instantiates another assumption known as *Output Independence* which states that the probability of an output observation o_i depends only on the state q_i that produced the observation and not on any other states or any other observations:

$$\textbf{Output Independence}: \mathrm{P}(o_i|q_1, q_2 \ldots . q_T, o_1, o_2 \ldots O_T) = P(o_i|q_i)$$

In order to apply HMM to our *Task 1* of classification we follow method suggested by Rabiner[1] in 1989. Given set of observations in a $Signal = o_1, o_2, \ldots \ldots o_T$, and model we calculate the probability of observation sequence given the model:

$$P(O|\lambda)$$

where $\lambda = (A, B, \pi)$; *A = Transition probability, B = Observation probability distribution in a state and* π *= Initial state distribution.*

This is typically the problem 1 of three basic problems[1] of HMM and it can be viewed as the evaluation problem i.e. how do we compute the probability that the observed sequence was produced by the model. It can also be viewed as scoring problem of how well the observation matches a given model. Before solving Problem 1 we optimize the model parameters using HMM training. Once we have found the parameters, we proceed with calculating $P(O|\lambda)$**.**

Following this idea, we build two models one for *Type 1* and other for *Type 2* since both the types are drawn from different probability distribution space. This is type of One-vs-Rest classification also known as One-vs-All. We then calculate the probability of observed sequence with both the models and assign final class to the given signal as one whose probability is greater.

$$P(O|\lambda_1 = (A_1, B_1, \pi_1)) = p_{class1}$$
$$P(O|\lambda_2 = (A_2, B_2, \pi_2)) = p_{class2}$$

In order to compute above probability, we used hmmlearn a Python package, that given an HMM model and observations returns the likelihood aka confidence score of the observation matching that model. Once we have score for each *Types*, we apply a softmax function on the output scores to have consistent baselining along with SVM and Random

Forests in terms of probabilities. This also helps us determining AUC correctly for this method. Given $[score_{class1},\ score_{class2}]$ softmax is applied as follows to get probabilities:

$$\sigma(z_i) = \frac{e^{z_i}}{e^{z_{score_{class1}}} + e^{z_{score_{class2}}}} \ where\ z_i = \{score_{class1},\ score_{class2}\}$$

Once we have the probabilities we assign classes to the given test signal as:

$$f^*(x) = \begin{cases} 1 & if\ \sigma\left(z_{score_{class1}}\right) > \sigma\left(z_{score_{class2}}\right) \\ -1 & otherwise \end{cases}$$

Hidden Markov Model for Error Segment Localization

Above we described Hidden Markov Model for classification task. We utilize the capabilities of HMM in our *Task 2* to perform error segment localization. Given set of observations in a $Signal = o_1, o_2, \dots\dots\ o_T$, and model $\lambda = (A, B, \pi)$ our goal is to determine the hidden state sequences for this observation. Once we have that we can generate a bounding box around error segment to measure performance against the deep learning model YOLO. We start by determining A and $\pi\ from\ the\ training\ data$. The state transition probability A and initial probability π are given as:

$$aij\ =\ P\left[q_{t+1}\ =\ S_j |\ q_t\ =\ S_i\right]\ 1 \leq\ i, j\ \leq\ N$$

$$\pi_i\ =\ P[q_1\ =\ S_i]\ \ 1\ \leq\ i\ \leq\ N$$

we determine the mean and variance of observations belonging to Normal and Error states respectively: $\mu_n, \boldsymbol{\sigma_n}, \mu_e, \boldsymbol{\sigma_e}$. Once we have these six parameters from our training data we attempt to optimize the model parameters to best optimize how the given sequence is generated. We perform HMM training using Baum-Welch algorithm. Once we have optimized out parameters we perform decoding on test data using Viterbi Algorithm:

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \,.\, B_j O_{t+1}$$

$$\text{where } \delta_t(i) = \max_{q_1, q_2 \ldots q_{t-1}} P[q_1, q_2 \ldots q_t = i, O_1, O_2 \ldots O_t | \lambda]$$

After the hidden states of all the observations in a signal are known, we save the start and end location of each error segment by finding the x position around continuous error states. These locations maps to predicted error segment index by HMM which will be used later for performance evaluation.

CHAPTER 3

SIMULATION STUDY

3.1 Generating Signal Data for Simulations

3.1.1 Data for Classification

Given a *Type 1* and *Type 2* signal our goal is to classify these two types correctly in their distinct classes. In order to do so we needed to prepare dataset for training both our statistical as well as well as deep learning model. In order to perform classification; train, test and validation datasets were prepared. To prepare signal dataset, we chose signal length (L) as per our experiment and created signal with observation for Normal states. Most of the experiment have L set to 200 and Normal state observations O_N drawn from N(μ_n, $\boldsymbol{\sigma}$). Later we ingested error segment continuously at a random position in the above signal obtaining a *Type 1* signal. The length of error segment was drawn from a Poisson distribution with lambda($\boldsymbol{\lambda}$) = 75 and Error state observations O_E were drawn from N($\mu_e, \boldsymbol{\sigma}$). For obtaining *Type 2* we didn't ingest the normal state observations with error states. We also introduced sparsity to mimic real world data. Then we randomly removed X% of observations from our signal to make it sparse. We generated 4400 signals in total with 2200 belonging to *Type 1* and 2200 belonging to *Type 2.* We used a split of 2000 signal for training, 2000 for testing and 400 for validation. These observations were then saved in CSV file for later training of our SVM, HMM and Random Forests models.

For preparing data for deep learning model we plotted a scatter plot of the observations against time (x axis). This process can be viewed as rasterizing a signal to create an image. The plots where then saved as 256 *x* 256 image and was further

preprocessed to remove any unwanted grids that might be present in the plot. Titles and axis lines were also removed, and the image was converted to greyscale and saved. These images were then used to train a deep learning model.

3.1.2 Data for Error Segment Localization

Generating data for Error Segment Localization involved one additional step where we also saved the location of an error segment in our signal for *Type 1* data. For each signal four points in graphic coordinate system were recorded corresponding to the error segment– top left, bottom right, width and height.

$$Top\ Left = (x0, y0)$$

$$Bottom\ Right = (x1, y1)$$

$$width = x1 - x0$$

$$height = y1 - y0$$

These points were then used to prepare labels corresponding to each signal in the format specified for training YOLO, an object detection deep learning model. Each label files consists of four normalized information – xcenter, ycenter, width and height; one per an error segment in a signal image.

$$xcenter = \frac{x0 + \frac{width}{2}}{256}$$

$$ycenter = \frac{y0 + \frac{height}{2}}{256}$$

$$width = \frac{width}{256}$$

$$height = \frac{height}{256}$$

These labels were saved for *Type 1* signal in corresponding train, test and validation directory along with rasterized signal image.

3.2 Experiments

As an attempt to investigate how better or worse a deep learning model is on our proposed rasterized signal solution as compared to statistical models, we performed series of experiments on simulated data. These simulated data were generated with various parameters in order to create data as close as to a real-world dataset. Since parameters in real world data are unknown we wanted to study how a model is impacted if certain parameters of signal are changed. Our experiments are organized as follows:

Experiment XX –

Variant 1, Variant 2 …. Variant X

Each of the experiments stresses features of our simulated signal and each variant in that experiment evaluates model performances on a slighter variation of it. For example, studying impact of spatial dependency will be one experiment and signal lengths will be its variants. The reason why we have multiple variants is to mimic real-world signal that could be drawn from infinite possibilities with various permutations and combinations. Therefore, in our research we perform extensive simulations to study various parameters

that can impact signal ultimately improving or degrading model's performance. This helps us to extend our research work to real world signal seamlessly.

3.2.1 Studying impact of Signal to Noise Ratio

A given signal may be impacted by certain amount of noise that can impact model performance either its deep learning or statistical model. Consequently, its useful to study how signal to noise ratio impacts the performance metrics such as accuracy, AUC etc. SNR is defined as mean over variance and higher the SNR, with more certainty model can distinguish the states of the signal. Therefore, we study the impact on performance on changing variance while keeping mean fixed and impact on performance while changing mean and keep variance fixed respectively.

For this experiment we generate a base signal of length L = 200. Let O_N be Normal(N) state observations drawn from mixture model of N(μ_n, $\boldsymbol{\sigma}$) and O_E be Error(E) state observations drawn from N($\mu_e, \boldsymbol{\sigma}$)**.** Let length of O_E be determined from a Poisson distribution with lambda($\boldsymbol{\lambda}$) = 75. Let X% of sparsity ranging from 20-80% be randomly introduced for each signal in the dataset. This specification forms our base signal.

i. Impact of Variances

In this variant we aim to study the effect of signal to noise ratio when mean is fixed, and variance varies.

Variants: Given base signal, we obtain a mixture model of O_N with $\mu_n = \{1, 0.5\ or\ -0.5\}$. We insert O_E with $\mu_e = \{-2\ or\ 2\}$ and study effects on classification by increasing the variance for N and E states as follows:

Variant 1: $\boldsymbol{\sigma} = \mathbf{0.5}$ for both N and E states

Variant 2: $\boldsymbol{\sigma} = \mathbf{1}$ for both *N* and *E* states

Variant 3: $\boldsymbol{\sigma} = \mathbf{2}$ for both *N* and *E* states

Variant 4: $\boldsymbol{\sigma} = \mathbf{3}$ for both *N* and *E* states

Variant 5: $\boldsymbol{\sigma} = \mathbf{4}$ for both *N* and *E* states

Variant 6: $\boldsymbol{\sigma} = \mathbf{5}$ for both *N* and *E* states

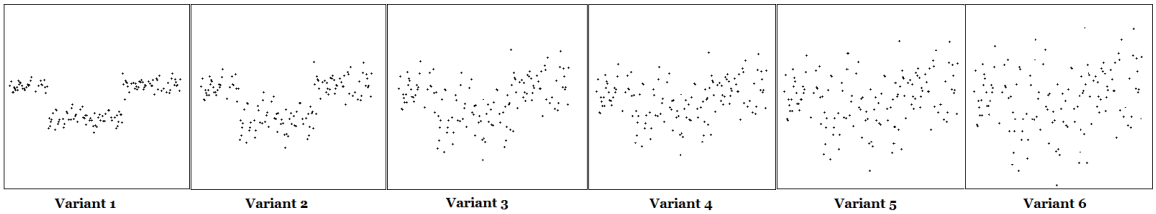

Figure 3.1 Input signal as variance increases.

Results:

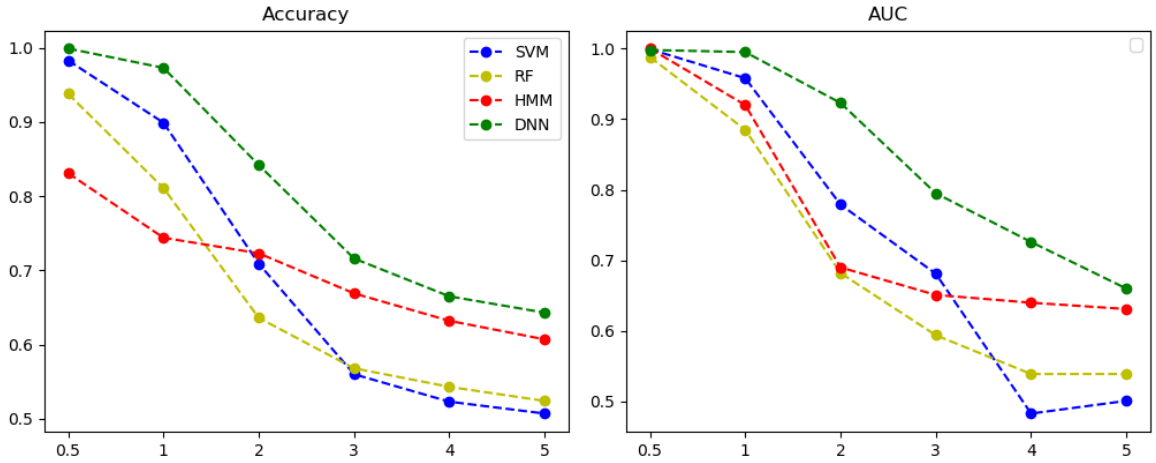

Figure 3.2 Shows the effect on classification accuracy and AUC as signal to noise ratio increases. Performance scores are represented on Y axis and distribution variance on X axis.

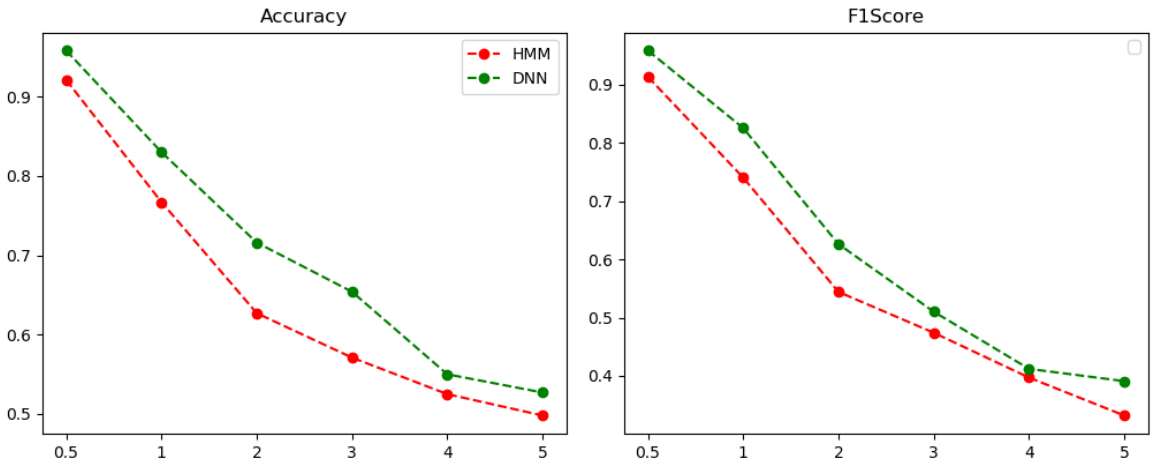

Figure 3.3 Shows the effect on accuracy and F1-Score of detecting error bounding box as signal to noise ratio increases. Performance scores are represented on Y axis and distribution variance on X axis.

From above we can see that as variance increases, model's ability to classify between *Type 1* and *Type 2* class decreases. If we observe the visual representation of the data, we can see that as variance increases it becomes extremely difficult to view the error segment in a signal. Statistical models rely on certain assumptions based on model parameters that is the reason why we see a dip in accuracy drastically for statistical model as compared to DNN. However, we see the deep learning model outperforming statistical models in this experiment. This is because deep learning model tries to learn representation of data rather than relying on certain assumption which might not hold true for a complex data.

So is true for error segment localization, as variance increases, we observe a drop in the accuracy. However, if we compare YOLO's performance with HMM in detecting the location of error segment in a signal we can see that it is still able to perform better than HMM which adds to the point that our suggested method is better than statistical approach.

ii. Impact of Means

In this experiment we aim to study the effect signal to noise ratio when increasing the difference between *N* and *E* means while keeping the variance fixed between these two states.

Variants: Given base signal we fix $\boldsymbol{\sigma}$ to 1 and μ_n *to* 1. We study the effect on signal classification when mean difference between Normal and Error state increases. Let mean difference be defined as:

$$mean\ difference\ =\ |\mu_n\ -\ \mu_e|$$

We study effects on the following variants:

Variant 1: $|\mu_n\ -\ \mu_e|\ =\ \mathbf{0.5}$, *where* $\mu_e\ =\ \{-0.5\ or\ 0.5\}$

Variant 2: $|\mu_n\ -\ \mu_e|\ =\ \mathbf{1}, where\ \mu_e\ =\ \{-2\ or\ 2\}$

Variant 3: $|\mu_n\ -\ \mu_e|\ =\ \mathbf{2}, where\ \mu_e\ =\ \{-3\ or\ 3\}$

Variant 4: $|\mu_n\ -\ \mu_e|\ =\ \mathbf{3}, where\ \mu_e\ =\ \{-4\ or\ 4\}$

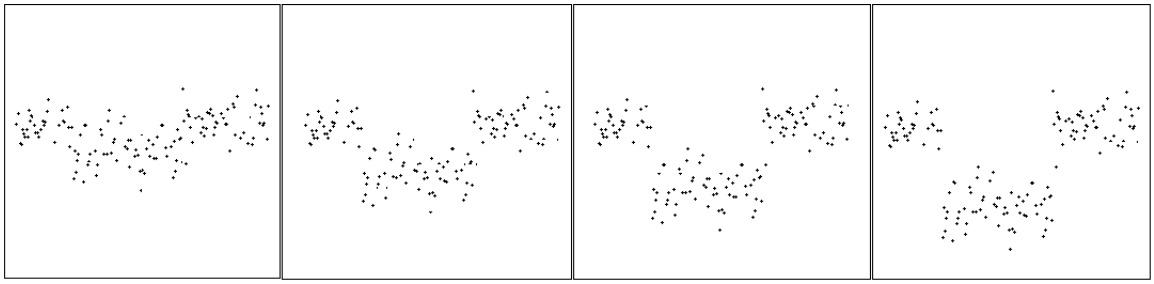

Figure 3.4 Input signal as difference in means of Normal and Error segment increases.

Results:

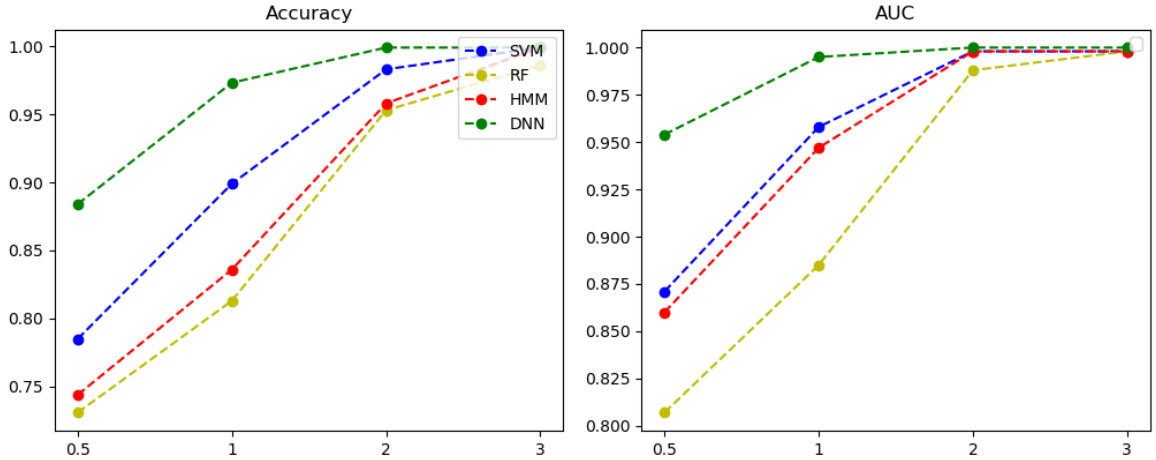

Figure 3.5 Shows the effect on classification accuracy and AUC as difference in *N* and *E* means increase.

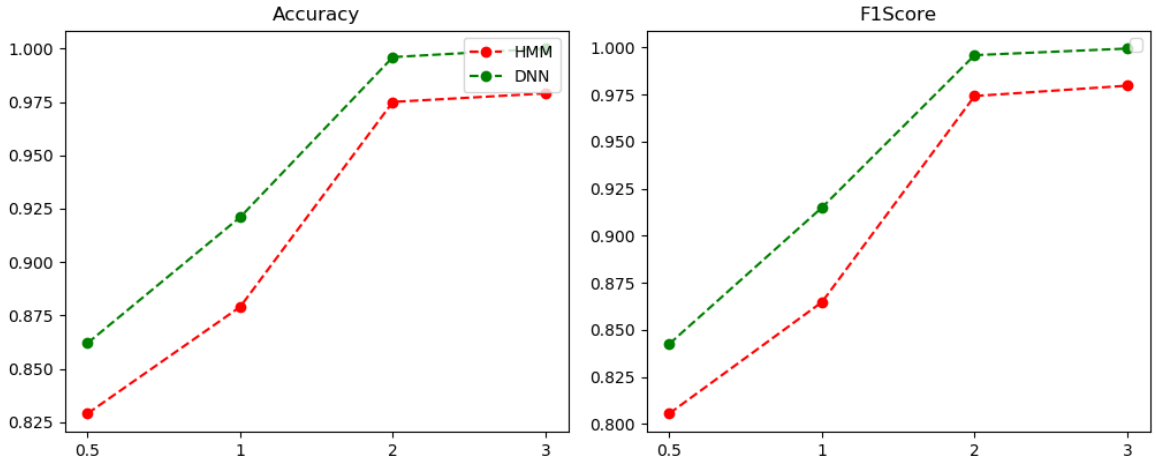

Figure 3.6 Shows the effect on accuracy and F1-Score of detecting error bounding box as difference in *N* and *E* means increase.

As the difference in means of Normal and Error segment increases it becomes easier for both deep learning and statistical model to classify the signal in Type 1 and Type 2. However, when the performance improves for both approaches, one must notice that it's quite significant for deep learning models which has better accuracy and AUC even when the mean difference is low. This might be because of spatial coherency which is visually visible with slightest difference in means. This pattern is utilized by deep learning model to learn to distinguish between both the types. For statistical models too as the difference

in means increases, the model parameters around both the types of signal changes significantly and becomes easier to distinguish. This is evident from the above figure where we can see the performance of HMM, SVM and Random Forests increase when the difference in means becomes significant. Same goes for error segment localization. With increase in difference one can observe that the performance for both the models increases. However deep neural networks seem to be performing much better than HMM. Their F1-Scores also suggests the same. A high F1-Score suggests that the model's precision and recall were high indicating model's ability to locate and identify an error segment correctly.

3.2.2 Studying Impact of Spatial Dependency

There are various factors that can impact a signal such as length, how sparse it is or presence or absence of intra state correlation. In this experiment, we aim to study the impact on performance when such factors change. Studying these factors will help us design our model to incorporate unusual behaviors that might impact its performance. This will also help us to explore the signal characteristics and draw conclusions if similar thing happens in a real-world dataset.

For this experiment we generate base signal with following specifications. Let signal length L = 200. Let O_N be Normal(N) state observations drawn from N(μ_n, $\boldsymbol{\sigma}$) where $\mu_n = 1$ and O_E be Error(E) state observations drawn from N(μ_e, $\boldsymbol{\sigma}$) where $\mu_e = \{-2\ or\ 2\}$. We fix $\boldsymbol{\sigma}$ to 1 for both the states. Let length of O_E be determined from a Poisson distribution with lambda($\boldsymbol{\lambda}$) = 75. Let X% of sparsity ranging from 20-80% be randomly introduced for each signal in the dataset.

i. Impact of Signal Sparsity

In this experiment we aim to study how inter and intra states for Normal and Error segments are correlated and how increase in sparsity effect this correlation and impact model's performance.

Variants: We generate base signal data without sparsity factor and introduce X% of sparsity specific to the variants. We then study effects on classification by increasing the sparsity in a signal as follows:

Variant 1: 10% sparse

Variant 2: 20% sparse

Variant 3: 10% sparse

Variant 4: 80% sparse

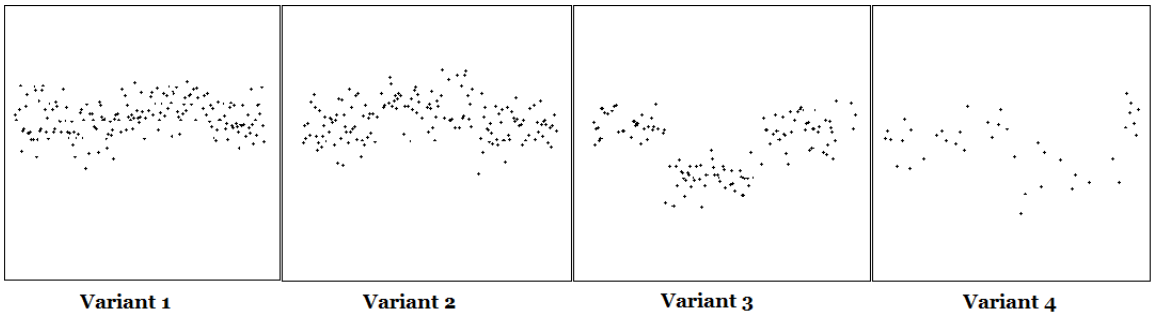

Figure 3.7 Input signal as sparsity increases.

Result:

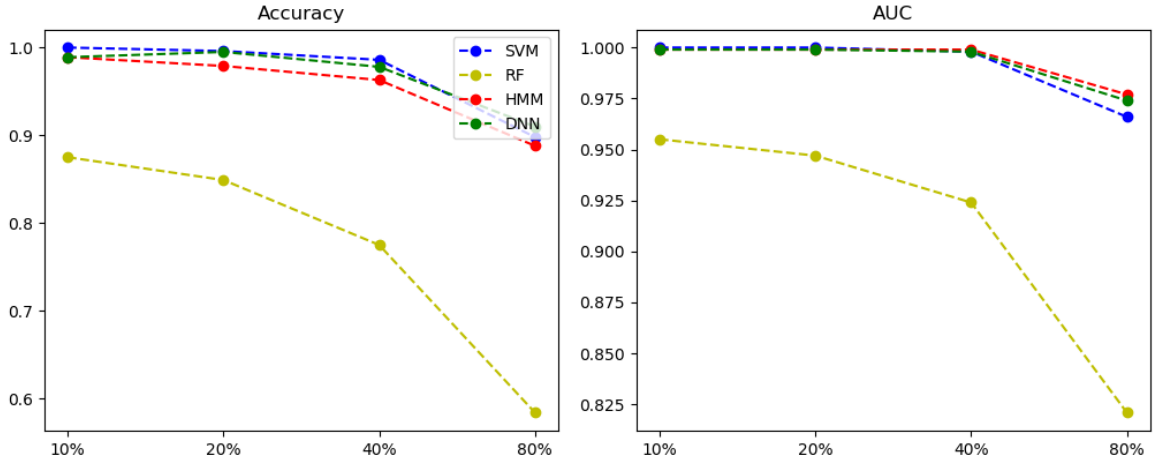

Figure 3.8 Shows the effect on classification accuracy and AUC as sparsity in signal increases.

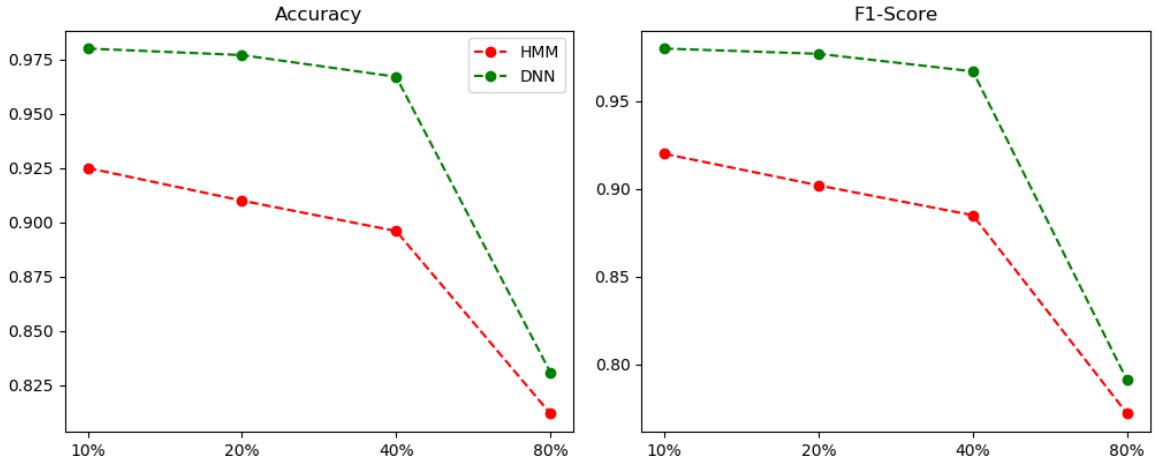

Figure 3.9 Shows the effect on accuracy and F1-Score of detecting error bounding box as sparsity in signal increases.

From above figure one can see that performance decreases as sparsity is introduced. The decrease in performance would be due to loss of information which might be impacting model's ability to classify the signals since certain useful information must have been lost. However, we can see deep learning approach is still better than HMM and Random Forests and is comparable with SVM. This is also true for error segment localization. YOLO

performs significantly better than HMM in localizing error segment even when sparsity increases.

ii. Impact of Signal Length

In this experiment we aim to study the effect of increasing signal length and compare the performance of proposed our deep neural network with statistical models.

Variants: We generate base signal data without length factor and generate signal of length L specific to the variants. We then study effects on classification by increasing the length of a signal as follows:

Variant 1: L= 200

Variant 2: L = 500

Variant 3: L = 1000

Variant 4: L = 2000

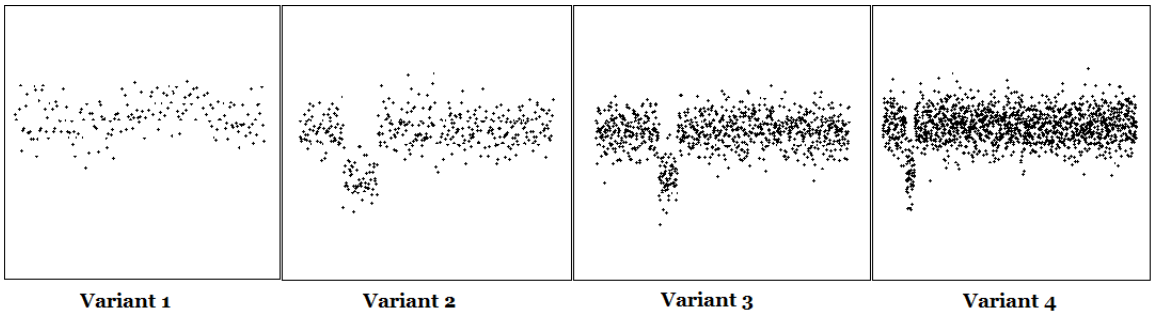

Figure 3.10 Input signal as length of the signal increases.

Result:

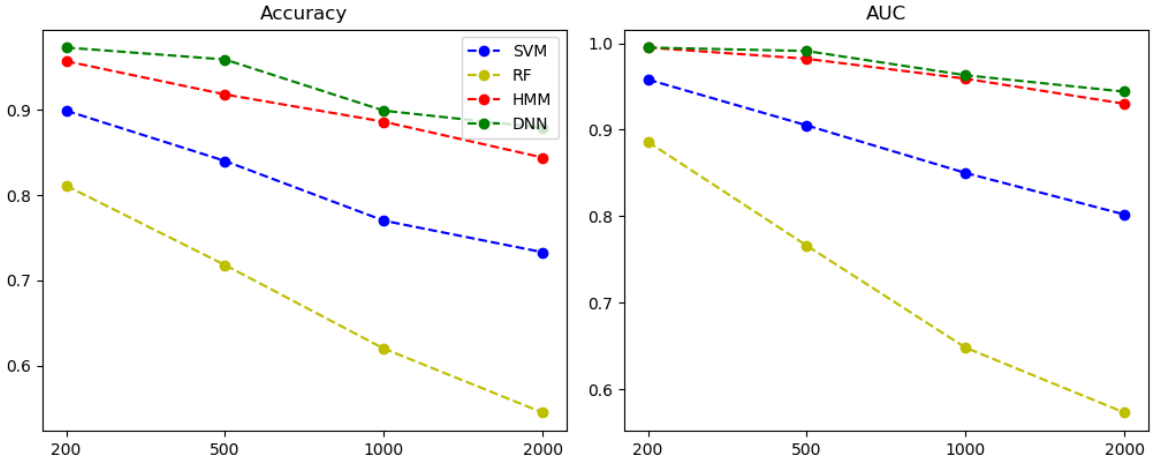

Figure 3.11 Shows the effect on classification accuracy and AUC as signal length increases.

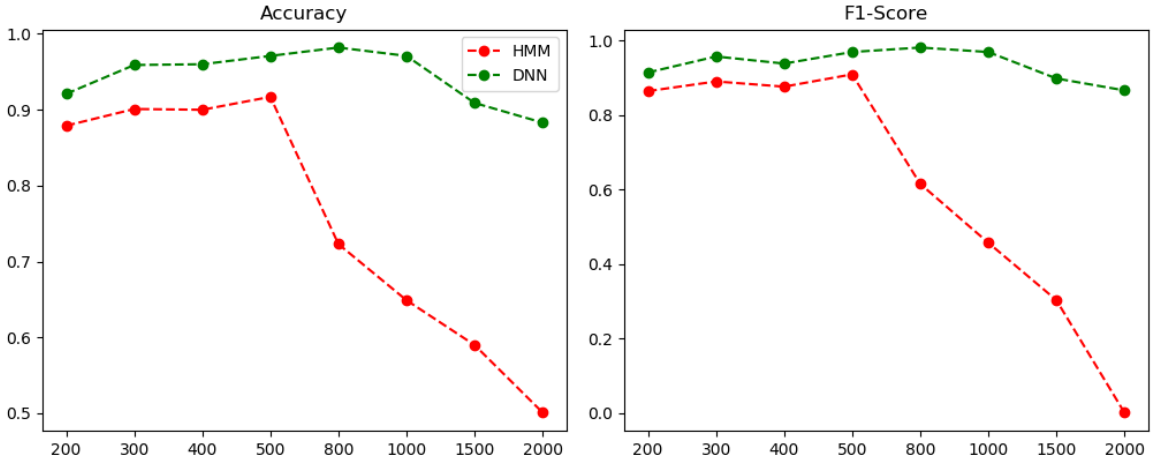

Figure 3.12 Shows the effect on accuracy and F1-Score of detecting error bounding boxes signal length increases.

In this experiment we can see our proposed technique outperforming other statistical model. The performance metrics for the *Task 1* is higher than statistical models which might be attributed to the factor that deep learning models are not sensitive to signal length rather it solely depends on finding visual pattern through which it decides. However, as length of signal increases, statistical models are impacted considerably, since the

parameters might be getting tuned around Normal state values as length increases as compared to Error state observations which is constant; leading to reduced performance.

For error segment localization, we saw an elbow pattern in HMM and Deep learning in which performance increased as signal length increased and after certain threshold it decreased. To confirm this pattern, we tested this experiment with four more variants. One can see that as length of signal increases with constant error length across variants, we see performance improving up to a certain threshold (500 for HMM and 800 for DNN) and with further increase of signal length the performance starts to fall. This might be due to the ratio of error length to signal length. We observed ratio of 0.12 (average of 75/500 and 75/800 where 75 is error length) was an optimal ratio for our experiment where maximum optimal performance was achieved by both the models. Anything other than this threshold resulted in performance degradation.

iii. Impact of Presence of Multiple Error Segments

A signal containing Normal and Error state, presence of multiple error or absence of it might impacts how model learns parameters or features during the training process. In this experiment we aim to study the effect of presence of one or more such error segment in *Type1* signal and compare the performance of proposed our deep neural network with statistical models.

Variants: In this we generate base signal data and introduce one more error segment with same specification but specific to the variant. We then study effects on classification for presence of one or more error segment as follows:

Variant 1: Signal with one error segment

Variant 2: Signal with two error segments

Variant 3: Signal with either one or two segments

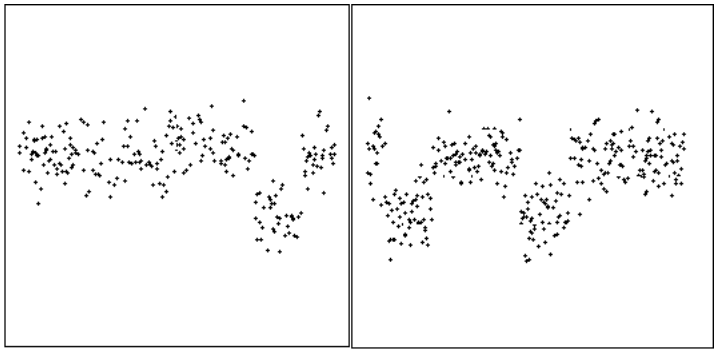

Figure 3.13 Input signal with one and two error segments. Variant 3 is combination of these two.

Result:

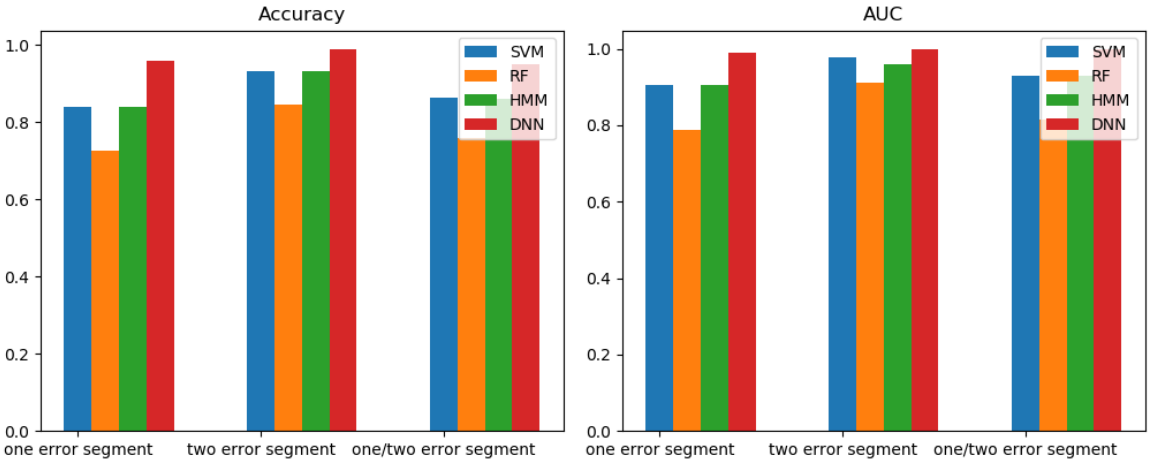

Figure 3.14 Shows the effect on classification accuracy and AUC with one or more error segments. Performance scores are represented on Y axis and no. of error segment on X axis.

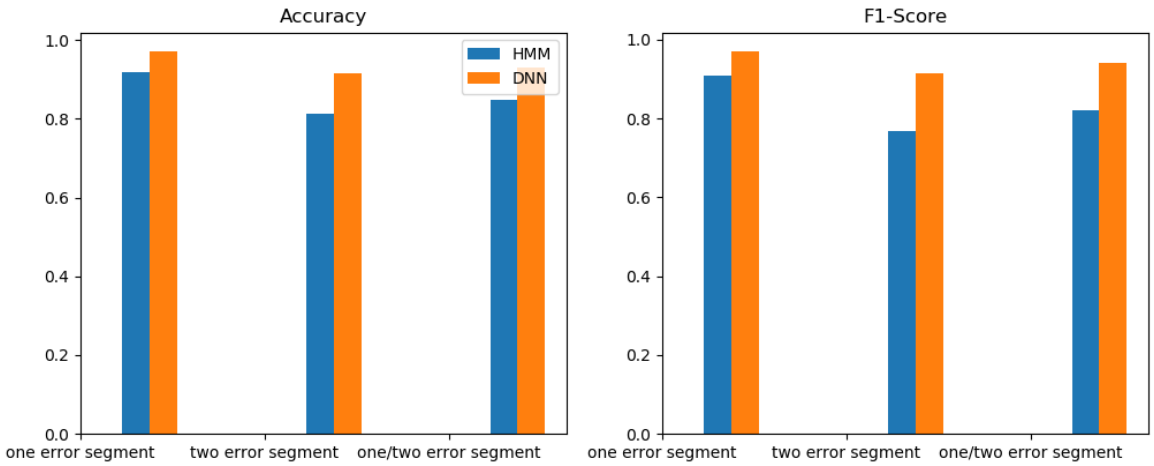

Figure 3.15 Shows the effect on accuracy and F1-Score of detecting error bounding box with one or more error segments.

From the figure we can see that classification performance when there exist exactly two error segments is much better than when there is either one error segment (variant 1) or at most two error segments (variant 3). This might be because presence of two signal helps boost model classification tendency for both deep learning as well as statistical models. For deep learning the pattern is quite visible if you compare *Type 1* and *Type 2* signal and for statistical model, the learnt model parameters are very different from a signal with two Error segment and one containing only Normal segment.

However, in error segment localization, we can see dip in accuracy for two error segments as compared to variant 1 and 2. YOLO's performance for all the variants are quite analogous. In HMM we observed that it made wrong predictions while detecting multi-error segments. Most of the time only one error segment was predicted for both the variants (2 & 3). The other segment was predicted as Normal Segment which impacted model's performance metrics dropping it below YOLO.

iv. Impact of Normal and Error Segment Length and Loss of Spatial Collinearity in an Error Segment

Since we assumed that occurrence of observations following an Error or Normal state in a given signal is continuous, we wanted to experiment with various variants of it such as when length of an Error segment is much large as compared to length of a Normal segment in a signal or a special case when the continuous spatial dependency between the observations following error state is removed. Henceforth we formulated variants including these cases under our simulation study.

Variants: We introduce following variants while changing certain properties of signal.

Variant 1: Base Signal

Variant 2: Signal with $length\ of\ O_N$ ~90% and length of O_E ~10% of the base signal

Variant 3: Signal with $length\ of\ O_N$ ~10% and length of O_E ~90% of the base signal

Variant 4: Signal with distributed error i.e. with no spatial dependency

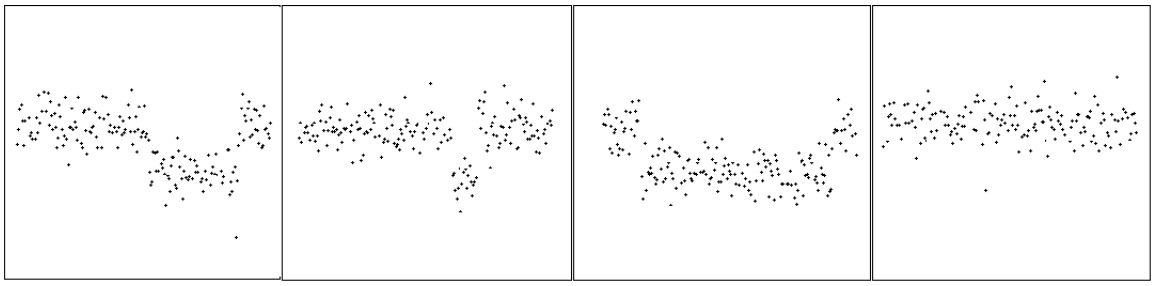

Figure 3.16 Input signal on 1: Signal under study, 2: Large normal length, 3: Large error length, 4: distributed error states.

Result:

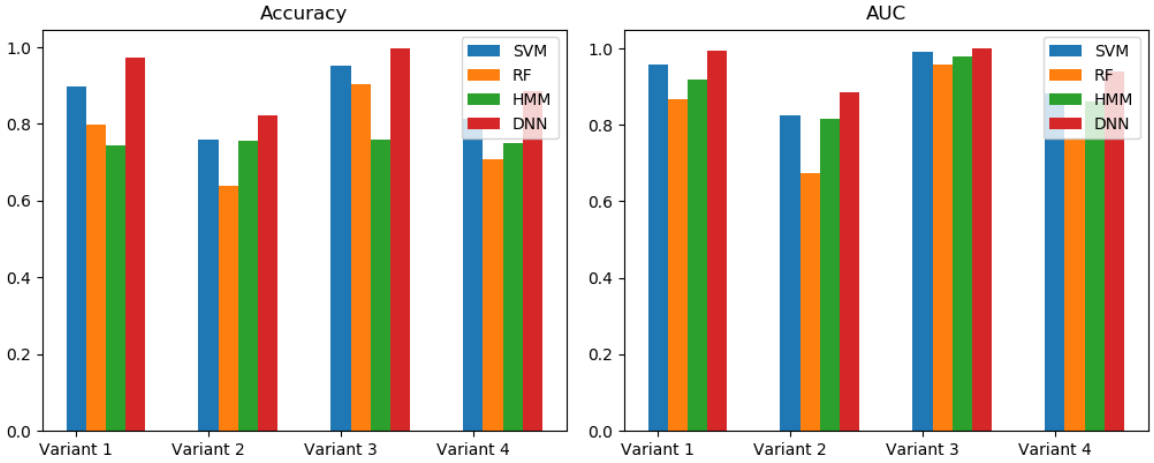

Figure 3.17 Shows the effect on classification accuracy and AUC on various variants.

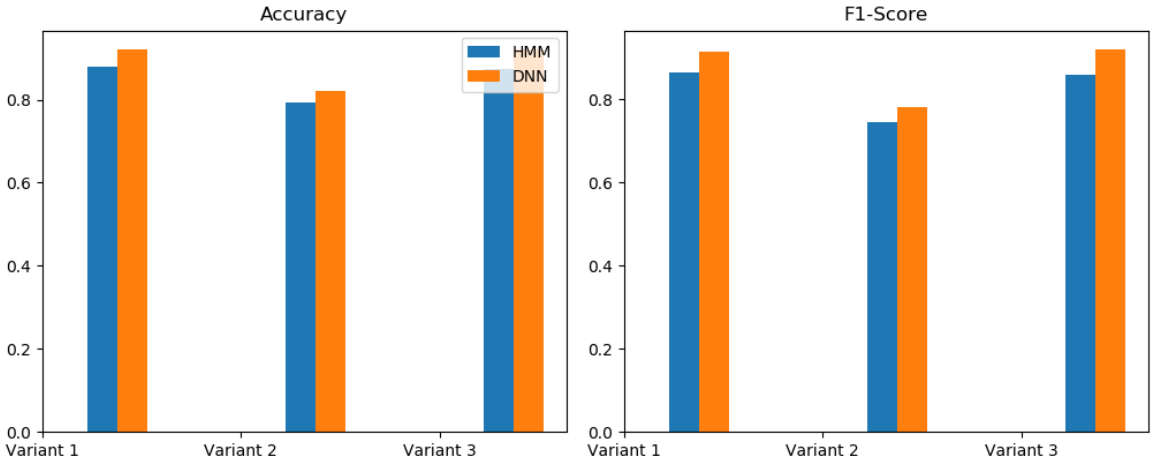

Figure 3.18 Shows the effect on accuracy and F1-Score of detecting error bounding box on various variants.

From above figure, one can see that if we increase normal segment length with respect to error segment (variant 2), it becomes hard form models to distinguish between *Type1* and *Type2* signal because both *Type 1 and Type 2* signal gets similar to each other either visually or parametrically and we see drop of performance from variant 1 to variant 2. Same is true for error segment localization, the model is not able to accurately determine the location of error in each signal.

Alternatively, if we increase error segment length (variant 3), it again becomes much easier for models to distinguish between *Type1* and *Type2* signal because in *Type 1* signal observations of error state are in majority and would be controlling model parameters while in *Type 2* signal, observations of normal state would be controlling the parameters. For deep learning models, a significant difference in pattern might be reason why it is performing slightly better than variant 1 & 2. Same reasoning goes for error segment localization.

Finally, if we remove spatial collinearity between error segments (variant 4), we can see that it becomes harder for models to classify the type of signal and accuracy drops considerable for all the models as compared to variant 1. For error segment localization, loss of spatial collinearity renders object detection model YOLO meaningless and hence we could not train an object detection model for this variant.

3.2.3 Studying Impact of Different Distribution and Model Misspecification

Till now, all the experiments assumed that observations were drawn from Normal distribution following certain mean and variance. However, it is possible that the real-world dataset is drawn from some other distribution. So, we wanted to study the robustness of statistical and our proposed deep learning solution when observations were drawn from different distributions or the states in each signal belongs to two different distributions. We also explore a special case when training is performed on a distribution different than the distribution test observations are drawn from. This is known as model misspecification. This study thus aims to bring out the intrinsic property of model's performance when the distribution is different than the one it expects.

For this experiment we generate base signal with following common specifications. Let signal length L = 200. Let length of O_E be determined from a Poisson distribution with lambda($\boldsymbol{\lambda}$) = 75. Let X% of sparsity ranging from 20-80% be randomly introduced for each signal in the dataset. We choose distribution of the observations specific to the variants.

Variants: We conduct series of experiments to see the effect of different distributions from which the signal state observations are drawn from.

Variant 1: Let O_N be Normal(*N*) state observations drawn from N(μ_n, σ_n) where $\mu_n = 1 \; and \; \sigma_n = 1$ and O_E be Error(*E*) state observations drawn from N($\mu_e, \boldsymbol{\sigma_e}$) where $\mu_e = \{-2 \; or \; 2\} \; and \; \sigma_e = 1$.

Variant 2: Let O_N be Normal(*N*) state observations drawn from N(μ_n, σ_n) where $\mu_n = 1 \; and \; \sigma_n = 1$ and O_E be Error(*E*) state observations drawn from $\Gamma(\alpha, \beta)$ where $\alpha = \{5 \; or \; 6 \; or \; 7.5\} \; and \; \beta = 1$.

Variant 3: Let O_N be Normal(*N*) state observations drawn from $\Gamma(\alpha, \beta)$ where $\alpha = \{5 \; or \; 6 \; or \; 7.5\} \; and \; \beta = 1$ and O_E be Error(*E*) state observations drawn N(μ_e, σ_e) where $\mu_e = \{-2 \; or \; 2\} \; and \; \sigma_e = 1$

Variant 4: Let O_N be Normal(*N*) state observations drawn from $\Gamma(\alpha, \beta)$ where $\alpha = 18 \; and \; \beta = 1$ and O_E be Error(*E*) state observations drawn $\Gamma(\alpha, \beta)$ where $\alpha = \{6 \; or \; 30\} \; and \; \beta = 1$

Variant 5: Train on variant 1 and test on variant 4

Variant 6: Train on variant 4 and test on variant 1

Note: If one plots the signal, Gamma parameters have been selected in a way such that they appear visually same as observations from Normal distribution

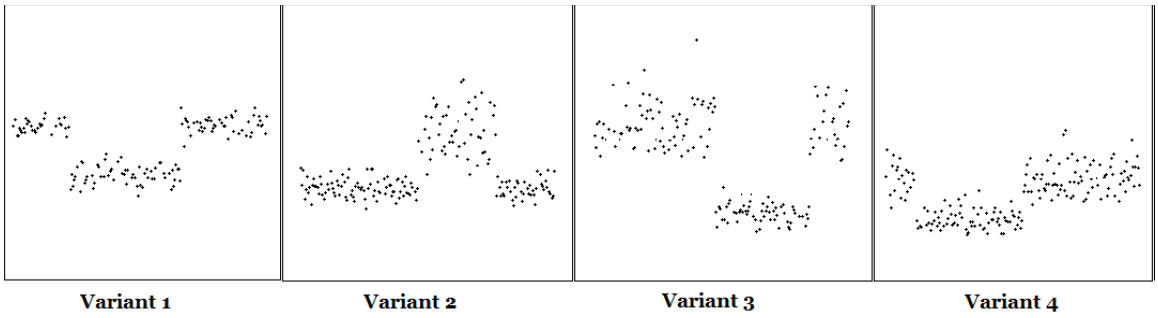

Figure 3.19 Input signal with observations drawn from different distributions.

Result:

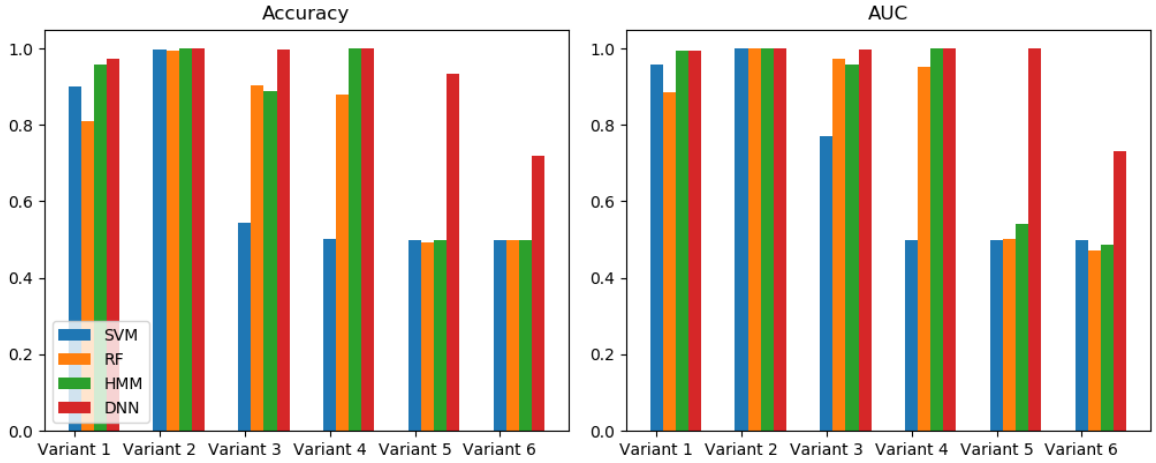

Figure 3.20 Shows the effect on classification accuracy and AUC on various variants.

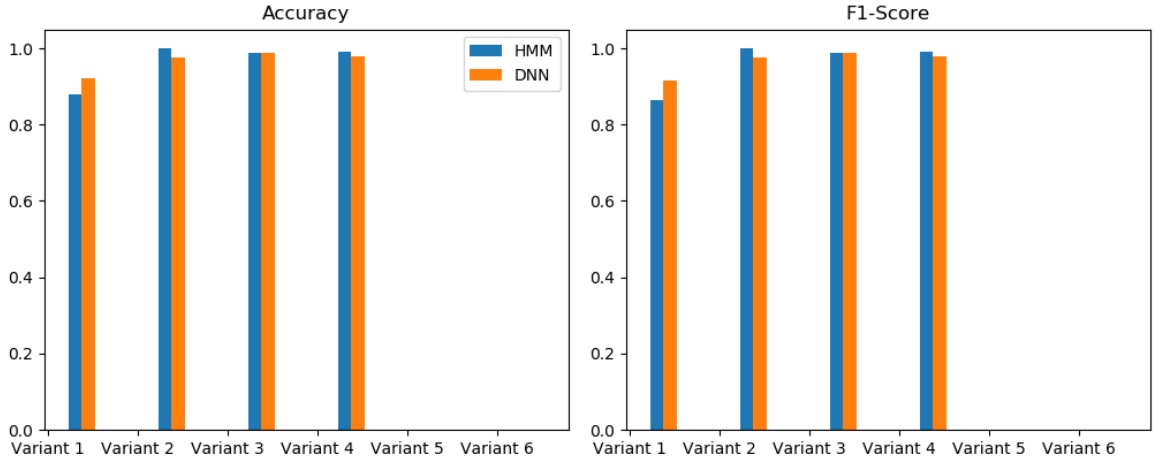

Figure 3.21 Shows the effect on accuracy and F1-Score of detecting error bounding box on various variants.

Few observations from above figure, if we compare: when both Normal or Error state observations are drawn from either Normal (variant 1) or Gamma (variant 4) probability distributions, the performance seems to comparable for HMM and Deep Learning model and not affected much by change in distribution. Same goes for error segment localization. This ensures us that both the models will be robust in case new incoming data changes. Although same is not true for SVM and RF which appears to be affected by change in probability distributions.

If we compare variant 1 and variant 2, when the distribution of Error segment in a signal is different than Normal segment in the same signal, for both classification and localization tasks performance doesn't degrade rather it improves slightly for all the models. This might be because the range of value from which error state observations are drawn from (Gamma distribution) is usually high and might be boosting statistical model's performance and its learning capabilities as it learns and tunes its parameters around it and thus enhancing classification and localization precisely. Same goes for deep learning model and is evident from Figure 3.21 variant 2 that error segment with gamma distribution is much more visually distinctive from signal with both the segments from same distribution.

Similarly, if we compare variant 3 with variant 4, we see model performance is quite comparable. This could be because Normal state observations which are drawn from Gamma distributions and whose observation values and segment length is quite large as compared to Error state observations which are small both in length and value (since they are drawn from Normal distribution) doesn't affect much the model parameters tuned by statistical models while training. Thus, leading to comparable performance. Deep learning model's performance is again comparable with each other and slightly better than variant1.

Finally, we see the performance in case of model misspecification in variant 5 and variant 6 in which you train on one distribution and test on other. Both variants see performance degradation since statistical models makes certain parametric assumption while modeling which gets invalidated if you test on dataset drawn from completely different distributions and deep learning model is not trained to recognize patterns present in test datasets since it has learnt to recognize patterns in training dataset. Although with low accuracy and AUC, for classification task deep learning model is still able to categorize the type of signal in classes. This proves the point that deep learning models are highly robust to underlying observation distributions. However, for error segment localization both YOLO and HMM fails to make any predictions and end up with zero Accuracy and F1-Score. This adds to the point that achieving *Task 2* is much harder as compared to *Task1*.

3.3 Conclusion

The performance metrics on the experiments conducted above for *Task 1* and *Task 2* suggests that the performance of our proposed signal rasterization technique using deep learning is much better than its competing baseline statistical models. For all the complex experiments and its variants, deep learning models have significant edge over the metrics of the baseline methods. First, this might be due to inherent property of deep learning models that learns to observe patterns in an image and uses it to make further decision unlike statistical models that makes certain assumptions while modeling which may or may not hold true for a given signal during testing. Second, deep learning models do not work directly with observations due to which they are immune to aspects where values interfere with the learning process. They learn features via representation learning and by tuning

weights propagated forward in the network and adjusting them on basis of loss propagated backwards enabling them to refine their learning process. Lastly, in our experiment we kept the model hyper-parameters and architecture of our deep learning model same for all the experiments and its corresponding variants. However, if we had changed them suiting to the needs of respective data, we are confident that for the cases where it performed at par with statistical models it would have certainly outperformed them. For now, using the performance metrics above we can successfully conclude that our method achieves higher performance on any given signal and is highly robust to change in signal distribution and intrinsic properties of the signal.

CHAPTER 4

REAL DATASET

4.1 1000 Genomes Project Phase 3 SVs

Structural Variants as the name suggests are the variations[43] in organism's chromosomes and are accountable for many diseases[44, 45] and genomic disorders[46] in humans. Structural Variants are generally defined as a region on DNA 1 kb or larger in size and includes many kinds of variations such as deletions, insertions, duplications, inversions, copy-number variation and translocations. Here in this research we deal with Copy Number Variation (CNV). CNV refers to deletion or duplication of reference DNA compared to reference genome assembly[47]. Presence of CNV has shown associations with diseases[48] and disorders[49] and comprehensive analysis of presence of CNVs will benefit genetics in accounting human genome variations as well as identifying diseases and disorders in a wider population and human diversity. Hence, we try to detect CNVs in 1000 Genomes Project Phase 3 integrated SV release set[12] which was published by Sudmant et al. in 2015. We show the performance of our suggested signal rasterization technique over statistical modelling by attempting to discover the presence of a CNV in a chromosome sample. The dataset we work upon was constructed through series of steps. Following figure describes the dataset construction process by Sudmant et al.

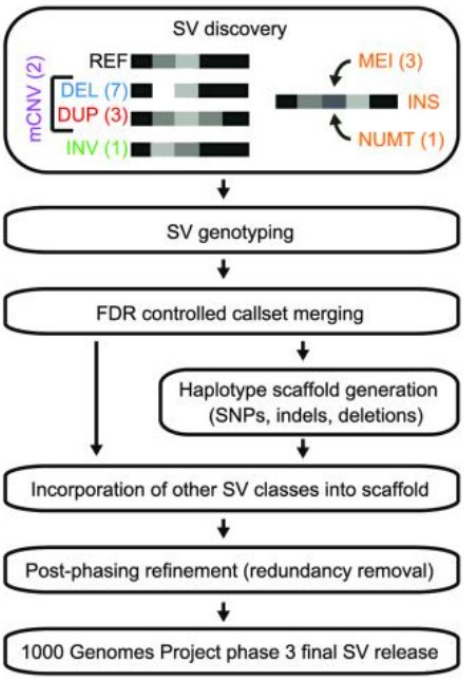

Figure 4.1 Approach used for constructing Phase 3 integrated SV release set.

Source: [12]

The dataset consists of 1000 chromosomes samples with 500 labeled as positive which indicates presence of Copy Number Variation treated as an error segment for our case (*Type 1*) and 500 labeled as NEG which indicates absence of a Copy Number Variation representing a normal segment i.e. signal without an error segment (*Type 2*). Each sample represents of SNP at a chromosome location. We have three information for each location: SNP position in a Chromosome, Log R Ratio: measure of normalized total signal intensity, B Allele Frequency: measure of normalized allelic intensity ratio. Our main task is to segment CNV in a chromosome sample. We split the entire data set in training, testing and validation sets for both the tasks. For *Task 1*, 560 samples were used for training, 140 for validation and 300 samples were used for testing. For Task 2, since YOLO is trained on data containing only error segment we used 300 POS samples for

training, 50 POS samples for validation and remaining 150 POS samples and 150 NEG samples were used for testing.

4.2 Results of Signal Classification and Error Segment Localization

We began training the models by preparing the datasets in the format suitable for respective tasks as described in chapter 2. Each of sample's log R Ratio was mapped against index and rasterized image was produced. Similarly labels for training YOLO was also generated corresponding to each error segments.

Once the data preparation was done we trained all the models for error classification and error segment localization tasks. Our signal rasterization technique outperformed statistical learning approaches by a good margin. It achieved an accuracy of 0.947 and AUC of 0.984 in classification task. Localizing an error segment was more complicated due to the complexity of samples and would be difficult even for human experts if they did the localization manually, so achieving an accuracy of 0.80 by YOLO was quite good. YOLO achieved slightly high precision relative to recall suggesting that it was not able to localize the CNVs but segmented it perfectly if it did identify it. We tried to change the hyper parameters such as batch size, learning rates, anchors but it didn't have much impact on the performance. We didn't replace YOLO architecture to keep it consistent with simulations, that is something which can be explored in future.

Table 4.1 Classification Performance Summary on 1000 Genomes Project Phase 3 SVs

Algorithm	Accuracy	AUC
SVM	0.737	0.944
RF	0.880	0.950
HMM	0.820	0.905
DNN	0.946	0.984

Table 4.2 Performance Summary of Error Segment Localization on 1000 Genomes Project Phase 3 SVs

Algorithm	Accuracy	Precision	Recall	F1-Score
HMM	0.677	0.602	0.645	0.623
DNN	0.800	0.839	0.722	0.776

One problem that we observed while performing error localization using HMM was many signal observations were predicted being in Error state even when they were in Normal state. This made localizing the error segment aka CNVs bit difficult. This resulted in reduced performance of HMM. From our observation, both models suggest the difficulty in localizing error segments is much higher than the simple task of classifying it and thus provides future work for advancements.

We also conducted a one-sided t-tests on Jaccard index obtained from HMM and YOLO for each sample in error segment localization. Our null hypothesis stated no difference in the performance of both the models while alternative stated that performance of YOLO is greater than HMM. We tested this at 95% confidence interval. We obtained t-statistics as 3.0755 on df=299 and p-value of 0.001148. The results show that we can reject null hypothesis and accept alternative hypothesis that YOLO is in fact better than HMM.

We also mapped the distribution of Jaccard Index obtained for *Type 1* signal. From the following figure we can see the prediction of YOLO was very close to ground truth resulting in majority of 1s as compared to HMM where density lies between 0.9 and 1.

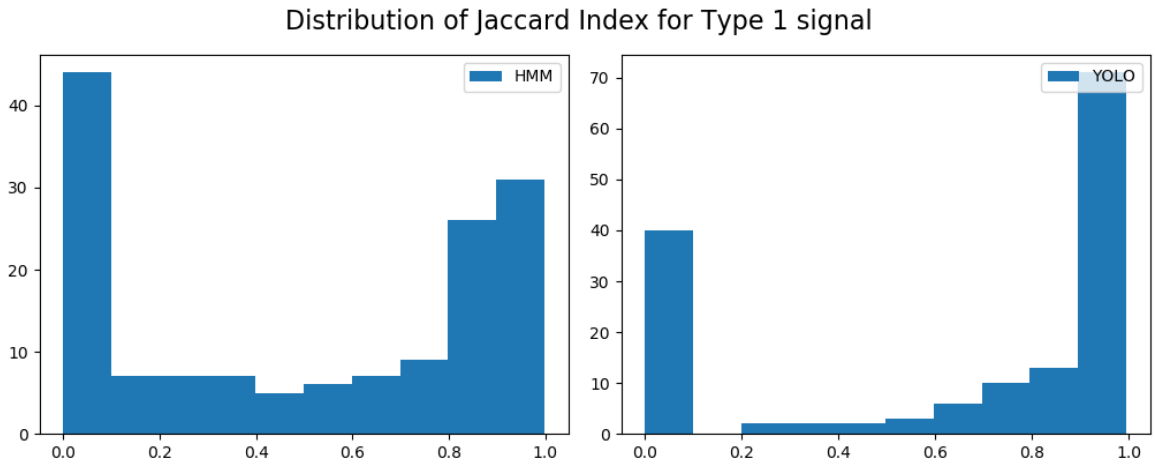

Figure 4.2 Distribution of Jaccard Index for Type 1 sample.

CHAPTER 5

CONCLUSION AND DISCUSSION

Statistical approaches of machine learning depend on underlying model parameters that are assumed from set of observations. In case these hidden states of observations and distributions are unknown, the parameters are heuristically defined that may or may not be true for a given model. In addition to this, there might be chances that in the future the incoming data does not strictly follow the same distribution as well as underlying principles as that of current one. In that case retraining and tuning the parameters becomes quite tedious and expensive task. It can rig the performance metrics and may not be suited to production deployment. Creating a robust model which is not dependent on data's distribution rather than on it features, patterns and visual properties is more suited for such tasks. In this research we proposed a signal rasterization technique for 1D numeric signal data following a Markov process calling it RM-Net. We showed the supremacy of our technique of image rasterization by converting the same problem to a computer vision problem and solving it using deep learning which is more robust and feature driven. We validated the superiority of our performance on simulated as well as real dataset and reported its metrics. We are confident that our approach can further be extended to multi-dimensional signals with correlated neighbors and associations within its observations too. However, currently one can observe that as complexity of real data increases, more work needs to be done in terms of localization and there is much more scope for improvement. This leaves us with possibility of future expansions and enhancements that might be more suited to complex dataset. We leave that probes to future work.

APPENDIX A

COMPREHENSIVE EXPERIMENTAL RESULTS FOR SIMULATED DATA

Following tables summarizes the results of classification and error segment localization on simulated data.

A.1 Studying impact of Signal to Noise Ratio

Table A.1 Classification Performance Summary for Impact of Variances

	Algorithm	Accuracy	AUC
Variant 1	SVM	0.983	0.999
	RF	0.938	0.988
	HMM	0.831	1.000
	DNN	0.999	0.998
Variant 2	SVM	0.899	0.958
	RF	0.811	0.885
	HMM	0.744	0.920
	DNN	0.973	0.995
Variant 3	SVM	0.709	0.799
	RF	0.636	0.682
	HMM	0.723	0.690
	DNN	0.842	0.923
Variant 4	SVM	0.560	0.681
	RF	0.568	0.594
	HMM	0.669	0.651
	DNN	0.716	0.795
Variant 5	SVM	0.523	0.483
	RF	0.543	0.539
	HMM	0.632	0.640
	DNN	0.665	0.726
Variant 6	SVM	0.507	0.501
	RF	0.524	0.539
	HMM	0.607	0.631
	DNN	0.643	0.660

Table A.2 Error Segment Localization Performance Summary for Impact of Variances

	Algorithm	Accuracy	F1-Score
Variant 1	HMM	0.921	0.913
	DNN	0.958	0.958
Variant 2	HMM	0.767	0.740
	DNN	0.830	0.825

	HMM	0.627	0.544
Variant 3	DNN	0.716	0.626
	HMM	0.571	0.474
Variant 4	DNN	0.654	0.510
	HMM	0.525	0.397
Variant 5	DNN	0.550	0.412
	HMM	0.498	0.332
Variant 6	DNN	0.527	0.391

Table A.3 Classification Performance Summary for Impact of Means

	Algorithm	**Accuracy**	**AUC**
	SVM	0.785	0.871
Variant 1	RF	0.731	0.807
	HMM	0.744	0.860
	DNN	0.884	0.954
	SVM	0.899	0.958
Variant 2	RF	0.813	0.885
	HMM	0.836	0.947
	DNN	0.973	0.995
	SVM	0.983	0.998
Variant 3	RF	0.953	0.988
	HMM	0.958	0.998
	DNN	0.999	1.000
	SVM	0.999	0.998
Variant 4	RF	0.986	0.998
	HMM	0.999	0.998
	DNN	0.999	1.000

Table A.4 Error Segment Localization Performance Summary for Impact of Means

	Algorithm	**Accuracy**	**F1-Score**
	HMM	0.829	0.806
Variant 1	DNN	0.862	0.843
	HMM	0.879	0.865
Variant 2	DNN	0.921	0.915
	HMM	0.975	0.974
Variant 3	DNN	0.996	0.995
	HMM	0.979	0.978
Variant 4	DNN	1.000	0.999

A.2 Studying impact of Spatial Dependency

Table A.5 Classification Performance Summary for Impact of Signal Sparsity

	Algorithm	Accuracy	AUC
Variant 1	SVM	1.000	1.000
	RF	0.875	0.955
	HMM	0.989	0.999
	DNN	0.989	0.999
Variant 2	SVM	0.996	1.000
	RF	0.868	0.943
	HMM	0.979	0.999
	DNN	0.995	0.999
Variant 3	SVM	0.986	0.998
	RF	0.839	0.921
	HMM	0.963	0.999
	DNN	0.978	0.998
Variant 4	SVM	0.898	0.966
	RF	0.767	0.846
	HMM	0.888	0.977
	DNN	0.910	0.974

Table A.6 Error Segment Localization Performance Summary for Impact of Signal Sparsity

	Algorithm	Accuracy	F1-Score
Variant 1	HMM	0.925	0.9202
	DNN	0.98	0.98
Variant 2	HMM	0.91	0.902
	DNN	0.977	0.98
Variant 3	HMM	0.896	0.885
	DNN	0.967	0.967
Variant 4	HMM	0.812	0.772
	DNN	0.831	0.791

Table A.7 Classification Performance Summary for Impact of Signal Length

	Algorithm	Accuracy	AUC
Variant 1	SVM	0.899	0.958
	RF	0.811	0.886
	HMM	0.957	0.995
	DNN	0.973	0.995
Variant 2	SVM	0.840	0.905
	RF	0.718	0.766
	HMM	0.918	0.982
	DNN	0.959	0.991

Variant 3	SVM	0.770	0.850
	RF	0.620	0.648
	HMM	0.886	0.959
	DNN	0.899	0.963
Variant 4	SVM	0.733	0.802
	RF	0.545	0.573
	HMM	0.844	0.959
	DNN	0.879	0.944

Table A.8 Error Segment Localization Performance Summary for Impact of Signal Length

	Algorithm	**Accuracy**	**F1-Score**
Variant 1 (200)	HMM	0.879	0.865
	DNN	0.921	0.915
Variant 2 (300)	HMM	0.901	0.891
	DNN	0.959	0.958
Variant 3 (400)	HMM	0.900	0.877
	DNN	0.960	0.939
Variant 4 (500)	HMM	0.917	0.909
	DNN	0.971	0.971
Variant 5 (800)	HMM	0.723	0.616
	DNN	0.982	0.981
Variant 6 (1000)	HMM	0.649	0.458
	DNN	0.971	0.970
Variant 7 (1500)	HMM	0.590	0.303
	DNN	0.909	0.899
Variant 8 (2000)	HMM	0.501	0.018
	DNN	0.883	0.868

Table A.9 Classification Performance Summary for Impact of Presence of Multiple Error Segments

	Algorithm	**Accuracy**	**AUC**
Variant 1	SVM	0.839	0.905
	RF	0.726	0.787
	HMM	0.839	0.905
	DNN	0.959	0.991
Variant 2	SVM	0.932	0.979
	RF	0.845	0.913
	HMM	0.932	0.961
	DNN	0.989	0.999
Variant 3	SVM	0.863	0.929
	RF	0.759	0.816
	HMM	0.862	0.929
	DNN	0.949	0.999

Table A.10 Error Segment Localization Performance Summary for Impact of Presence of Multiple Error Segments

	Algorithm	Accuracy	F1-Score
Variant 1	HMM	0.917	0.909
	DNN	0.971	0.971
Variant 2	HMM	0.812	0.769
	DNN	0.915	0.914
Variant 3	HMM	0.849	0.822
	DNN	0.930	0.942

Table A.11 Classification Performance Summary for Impact of Normal and Error Segment Length and Loss of Spatial Correlation in an Error Segment

	Algorithm	Accuracy	AUC
Variant 1	SVM	0.899	0.958
	RF	0.798	0.868
	HMM	0.744	0.920
	DNN	0.973	0.995
Variant 2	SVM	0.760	0.826
	RF	0.639	0.675
	HMM	0.755	0.817
	DNN	0.821	0.885
Variant 3	SVM	0.951	0.991
	RF	0.903	0.957
	HMM	0.760	0.980
	DNN	0.997	1.000
Variant 4	SVM	0.814	0.882
	RF	0.707	0.765
	HMM	0.751	0.862
	DNN	0.887	0.940

Table A.12 Error Segment Localization Performance Summary for Impact of Normal and Error Segment Length and Loss of Spatial Correlation in an Error Segment

	Algorithm	Accuracy	F1-Score
Variant 1	HMM	0.879	0.865
	DNN	0.921	0.915
Variant 2	HMM	0.794	0.745
	DNN	0.822	0.782
Variant 3	HMM	0.875	0.860
	DNN	0.920	0.920
Variant 4	HMM	-	-
	DNN	-	-

A.3 Studying impact of Different Distribution and Model Misspecification

Table A.13 Classification Performance Summary for Different Distribution and Model Misspecification

	Algorithm	Accuracy	AUC
Variant 1	SVM	0.899	0.958
	RF	0.811	0.886
	HMM	0.957	0.995
	DNN	0.973	0.995
Variant 2	SVM	0.998	1.000
	RF	0.993	1.000
	HMM	1.000	1.000
	DNN	1.000	1.000
Variant 3	SVM	0.545	0.770
	RF	0.905	0.972
	HMM	0.889	0.958
	DNN	0.997	0.998
Variant 4	SVM	0.502	0.500
	RF	0.878	0.952
	HMM	1.000	1.000
	DNN	1.000	1.000
Variant 5	SVM	0.500	0.500
	RF	0.494	0.502
	HMM	0.500	0.510
	DNN	0.924	1.000
Variant 6	SVM	0.500	0.500
	RF	0.500	0.472
	HMM	0.680	0.635
	DNN	0.720	0.730

Table A.14 Error Segment Localization Performance Summary for Different Distribution and Model Misspecification

	Algorithm	Accuracy	F1-Score
Variant 1	HMM	0.879	0.865
	DNN	0.921	0.915
Variant 2	HMM	1.000	1.000
	DNN	0.977	0.976
Variant 3	HMM	0.989	0.989
	DNN	0.989	0.989
Variant 4	HMM	0.990	0.990
	DNN	0.980	0.979
Variant 5	HMM	0.000	0.000
	DNN	0.000	0.000
	HMM	0.000	0.000

Variant 6	DNN	0.000	0.000

REFERENCES

1. Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

2. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

3. Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, *2018*, 1–13.

4. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

5. Baum, L. E., & Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, *37*(6), 1554–1563.

6. Silva, C., & Ribeiro, B. (2007). Combining active learning and relevance vector machines for text classification. *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*.

7. Hastie, T., Friedman, J., & Tisbshirani, R. (2017). *The Elements of statistical learning: data mining, inference, and prediction*. Springer.

8. Dean, J. (2020). 1.1 The Deep Learning Revolution and Its Implications for Computer Architecture and Chip Design. *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*.

9. Bengio, Y., Courville, A., & Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828.

10. Bazzani, L., Bergamo, A., Anguelov, D., & Torresani, L. (2016). Self-taught object localization with deep networks. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*.

11. Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015). Efficient object localization using Convolutional Networks. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

12. Sudmant, P. H., Rausch, T., Gardner, E. J., Handsaker, R. E., Abyzov, A., Huddleston, J., … Korbel, J. O. (2015). An integrated map of structural variation in 2,504 human genomes. *Nature*, *526*(7571), 75–81.

13. Yoon, B.-J. (2009). Hidden Markov Models and their Applications in Biological Sequence Analysis. *Current Genomics*, *10*(6), 402–415.

14. Ren, Y., & Zhang, F. (2009). Hand Gesture Recognition Based on MEB-SVM. *2009 International Conference on Embedded Software and Systems*.

15. Dias, J. G., Vermunt, J. K., & Ramos, S. (2009). Mixture Hidden Markov Models in Finance Research. *Advances in Data Analysis, Data Handling and Business Intelligence*, 451–459.

16. ZHOU, X., WU, Y., & YANG, B. (2010). Signal Classification Method Based on Support Vector Machine and High-Order Cumulants. *Wireless Sensor Network*, *02*(01), 48–52.

17. R., S. (2020). Sampling Distributive Discriminant Random Decision Tree Classification for Spatio-temporal Pattern Prediction. *Journal of Advanced Research in Dynamical and Control Systems*, *12*(SP3), 1429–1440.

18. Almanjahie, I. M., Khan, R. N., Milne, R. K., Nomura, T., & Martinac, B. (2015). Hidden Markov analysis of improved bandwidth mechanosensitive ion channel data. *European Biophysics Journal*, *44*(7), 545–556.

19. Jojoa, M., & Garcia-Zapirain, B. (2020). Forecasting COVID 19 Confirmed Cases Using Machine Learning: the Case of America.

20. Shrestha, A., & Mahmood, A. (2019). Review of Deep Learning Algorithms and Architectures. *IEEE Access*, *7*, 53040–53065.

21. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

22. Nwankpa, C. E. (2020). Advances in Optimisation Algorithms and Techniques for Deep Learning. *Advances in Science, Technology and Engineering Systems Journal*, *5*(5), 563–577.

23. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

24. Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*.

25. González, J. Á., Hurtado, L.-F., & Pla, F. (2021). TWilBert: Pre-trained deep bidirectional transformers for Spanish Twitter. *Neurocomputing*, *426*, 58–69.

26. Kundu, S., Sim, K. C., & Gales, M. J. F. (2016). Incorporating a Generative Front-End Layer to Deep Neural Network for Noise Robust Automatic Speech Recognition. *Interspeech 2016*.

27. Pang, Y., & Cao, J. (2019). Deep Learning in Object Detection. *Deep Learning in Object Detection and Recognition*, 19–57.

28. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *Lecture Notes in Computer Science*, 234–241.

29. Xia, A., Li, D., Cai, J., Gu, H., & Qin, P. (2020). QNet: A Quick Deep Neural Network for Real-Time Semantic Segmentation. *2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP)*.

30. Kleppe, A., Skrede, O.-J., De Raedt, S., Liestøl, K., Kerr, D. J., & Danielsen, H. E. (2021). Designing deep learning studies in cancer diagnostics. *Nature Reviews Cancer*, *21*(3), 199–211.

31. Poplin, R., Chang, P.-C., Alexander, D., Schwartz, S., Colthurst, T., Ku, A., … DePristo, M. A. (2018). A universal SNP and small-indel variant caller using deep neural networks. *Nature Biotechnology*, *36*(10), 983–987.

32. Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, *151*, 107398.

33. Ma, S., & Zhang, Z. (2019, May 23). *OmicsMapNet: Transforming omics data to take advantage of Deep Convolutional Neural Network for discovery*. [1804.05283v2] OmicsMapNet: Transforming omics data to take advantage of Deep Convolutional Neural Network for discovery.

34. Bazgir, O., Zhang, R., Dhruba, S. R., Rahman, R., Ghosh, S., & Pal, R. (2020). Representation of features as images with neighborhood dependencies for compatibility with convolutional neural networks. *Nature Communications*, *11*(1).

35. Glessner, J. T., Hou, X., Zhong, C., Zhang, J., Khan, M., Brand, F., … Wei, Z. (2021). DeepCNV: a deep learning approach for authenticating copy number variations. *Briefings in Bioinformatics*.

36. Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018, August 6). *A Survey on Deep Transfer Learning*. [1808.01974v1] A Survey on Deep Transfer Learning.

37. Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016, August 23). *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. arXiv.org.

38. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

39. Redmon, J., & Farhadi, A. (2018, April 8). *YOLOv3: An Incremental Improvement*. [1804.02767] YOLOv3: An Incremental Improvement.

40. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*.

41. Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149.

42. Suykens, J. A. K. (2001). Support Vector Machines: A Nonlinear Modelling and Control Perspective. *European Journal of Control*, *7*(2-3), 311–327.

43. Feuk, L., Carson, A. R., & Scherer, S. W. (2006). Structural variation in the human genome. *Nature Reviews Genetics*, *7*(2), 85–97.

44. International Schizophrenia Consortium (2008). Rare chromosomal deletions and duplications increase risk of schizophrenia. *Nature*, *455*(7210), 237–241.

45. Lupski, J. R. (1998). Genomic disorders: structural features of the genome can lead to DNA rearrangements and human disease traits. *Trends in Genetics*, *14*(10), 417–422.

46. Wang, K., Li, M., Hadley, D., Liu, R., Glessner, J., Grant, S. F. A., … Bucan, M. (2007). PennCNV: An integrated hidden Markov model designed for high-resolution copy number variation detection in whole-genome SNP genotyping data. *Genome Research*, *17*(11), 1665–1674.

47. Deng, F.-Y., Zhao, L.-J., Pei, Y.-F., Sha, B.-Y., Liu, X.-G., Yan, H., … Deng, H.-W. (2010). Erratum to: Genome-wide copy number variation association study suggested VPS13B gene for osteoporosis in Caucasians. *Osteoporosis International*, *21*(8), 1455–1455.

48. Deng, F.-Y., Zhao, L.-J., Pei, Y.-F., Sha, B.-Y., Liu, X.-G., Yan, H., … Deng, H.-W. (2009). Genome-wide copy number variation association study suggested VPS13B gene for osteoporosis in Caucasians. *Osteoporosis International*, *21*(4), 579–587.

49. Rucker, J. J., & McGuffin, P. (2013). Copy Number Variation in Neuropsychiatric Disorders. *Oxford Handbooks Online*.