2022 International Conference on Engineering and Emerging Technologies (ICEET) | 978-1-6654-9106-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICEET56468.2022.10007093

Machine Learning Classifiers Based on HoG Features Extracted from Locomotive Neutral Section Images

Christopher Thembinkosi Mcineka
Dept. of Electronic & Computer Engineering
Durban University of Technology
Durban, South Africa
21959502@dut4life.ac.za

Nelendran Pillay
Dept. of Electronic & Computer Engineering
Durban University of Technology
Durban, South Africa
trevorpi@dut.ac.za

***Abstract*— This paper presents a comparative study on machine learning algorithms for neutral section image classification. The classifiers are trained by employing the Histogram of Oriented Gradient features that are extracted from the neutral section dataset [1]. A neutral section is a phase break that is used on the Transnet freight rail system to separate the single-phase supply from the 25kV three-phase overhead traction supply. The 25kV is a stepped-down voltage from an 88kV three-phase supply coming from the national grid. While the main purpose of the neutral section is to separate phase voltages, electric locomotives can traverse through these phases by switching On and Off. This auto-switching is possible through induction magnets installed in between the rails and with magnet detection sensors installed underneath the locomotives. However, a computer vision model has been developed, trained, and tested with a neutral section dataset containing images having open and close markers [1]. This paper, therefore, utilises this dataset to provide performance comparison on several machine learning classification algorithms viz. Decision Tree, Discriminant Analysis, Support Vector Machine, K-Nearest Neighbors, Ensemble, Naïve Bayes, and Convolutional Neural Network. A confusion matrix, F1-measure and computation time are employed to measure the performance of each classifier. The MATLAB Classification Learner application was used to obtain the results. The results show that the Linear Support Vector Machine performs best when considering performance and prediction speed. The Linear Support Vector Machine achieved a training accuracy of 93.40% with a test accuracy reaching 94% at a prediction speed of 75 objects per second (computation time).**

Keywords*— *Neutral section dataset, Machine Learning Classifiers, Histogram of Oriented Gradient, Computer Vision, MATLAB, Confusion matrix, F1-measure.

I. Introduction

Transnet embarked on a strategy called Transnet 4.0, which aimed at aligning its strategy with the Fourth Industrial Revolution (4IR) technologies. In keeping with this ethos, Mcineka and Reddy [1] developed a model that employed Machine Learning (ML) algorithm to automatically switch off/on the electric locomotives as they traverse through the Neutral Section (NS). Therefore, the conventional onboard switching scheme deployed in the Transnet railway lines can be replaced with a Computer Vision (CV) based system. In [1], the authors were able to achieve an overall accuracy of 72% on their model. ML classifiers have been deployed in different sectors such as in transportation viz. traffic sign detection and Automatic Number Plate Recognition (ANPR). While several pieces of literature have employed ML classifiers, few have focused on the automatic switching of electric locomotives in railway industries [1]. Subsequently, the latter implies that there is a scarcity of available datasets that can be used to train and test any classifier being proposed for auto-switching electrical locomotives through a CV system. Artificial Intelligence (AI) technology is part of the 4IR industrial changes and CV being the field of AI is in line with the Transnet 4.0 strategy. The CV enables a computer to derive meaningful information from objects, specifically images or video images. The CV does this by extracting information and a computer can interpret this data meaningfully through a trained ML classifier. Therefore, the objective of this paper is to provide a comparative study on various ML classifiers that are commonly employed in CV using the dataset in [1]. The selection of this dataset is mainly due to the scarcity of available datasets used in railway neutral sections.

This paper is structured as follows: Section II provides a literature review on each of the seven ML classifiers used for this study. Section III is an illustration of the conventional NS found in Transnet freight rail. Section IV presents an overview of the dataset used. Section V details the mathematical model of each classifier. Section VI discusses the results obtained. Lastly, Section VII concludes this study.

II. Literature Review

Mitrofanov and Semenkin [2] discussed an approach to training Decisions Trees (DTs). The authors describe the Iterative Dichotomiser 3 (ID3) and the Classification And Regression Tree (CART) as the two main decision tree learning algorithms. In the ID3 an entropy criterion and information gain are employed to grow the tree [2]. Entropy measures the amount of uncertainty in the (data) set $\boldsymbol{R}$. Information gain measures the difference in entropy from before to after the set. CART employs a Gini impurity method: it measures a randomly chosen element from the set $\boldsymbol{R}$ on how often would it be incorrectly labelled if it was randomly labelled according to the distribution of labels in the subset. The authors [2], reported that the standard DT algorithm is a greedy learning scheme since it sequentially builds a tree from the top node to the bottom.

Alzubaidi et al. [3], reviewed Deep Learning (DL) concepts based on Convolutional Neural Network (CNN) where they discussed the architecture, challenges and applications. The authors alluded that the main advantage of employing a CNN is the reduction in the number of trainable network parameters by using weight sharing, subsequently enhancing generalisation and avoiding overfitting [3].

Tharwat [4], discusses the Discriminant Analysis (DA) classifier where a tutorial of Linear DA (LDA) and Quadratic DA (QDA) are presented. The ovarian dataset was collected from the Pacific Northwestern National Lab and Johns Hopkins University. The author provided a basic mathematical and visual explanation of the DA steps. The author then concluded that the QDA achieved better results than the LDA classifier [4].

Seth and Banka [5], proposed a cost-effective technique employing Naïve Bayes classifier on a low-cost hardware implementation. The hardware implementation used is a Raspberry Pi 2 with a Quad-core processor (ARMv7) running at 900MHz@ 1GB on-board RAM with a Linux Debian OS. Furthermore, several data sets are used viz. Iris, Diabetes, Coal mine, Hepatitis, Breast cancer and Glass in testing performance of Naïve Bayes on the Raspberry Pi2. The results show that the Naïve Bayes achieves an average accuracy of 86.40% while running on a low-cost machine[5]. This implies that the Naïve Bayes is not a computationally expensive algorithm. Furthermore, the results show that the algorithm performs differently with various data sets, for instance, from lowest to highest the Coal mine data set achieved an accuracy of 73.87% while the Glass data set achieved 98.62%.

Marriette and Rahu [6], discuss the Support Vector Machine (SVM) for classification. The authors provided SVM formulations for linearly and non-linearly separable data. Firstly, one commonly available kernel function: Linear (LSVM), Polynomial: Quadratic (QSVM)/Cubic (CSVM), Radial Basis Function (RBF) and Sigmoid is selected to transform non-linearly separable data to feature space. The algorithm then finds the optimal hyperplane that separates the classes by employing the Lagrangian and soft margin. The computational complexity of the SVM is optimised by the Sequential Minimal Optimisation (SMO) method: which divides the optimisation problem into two Quadratic Program (QP) problems [6]. This decomposition allows for the SVM algorithm to be trained on a large dataset without the constraints of large memory requirements.

Barstuğan and Ceylan [7], conducted a comparison study based on two ensemble classifiers with biomedical datasets. They compared DT and SVM-based AdaBoost ensembled classifiers. They calculated the accuracy of each ensembled classifier by using 10-fold cross-validation. The authors recorded an average of 76.47% on the AdaBoost-Decision tree and 81.43% on the AdaBoost-SVM classifiers. The results suggest that SVM ensembled classifiers achieve better performance when compared to those with DT.

Yigit [8], proposed an optimal weighting approach for K-Nearest Neighbours (K-NN) classifier in an Artificial Bee Colony (ABC) algorithm. The author tested the validity of the algorithm called the distance-weighted ABC K-NN (dW-ABC K-NN) with the UCI dataset. The author describes the ABC algorithm as an algorithm that mimics the foraging behaviour of honeybees that seek quality food. In the context of computation, the K-NN is a selection of K-values and the Euclidean distance of an unlabelled object to all the labelled objects within the training set. Furthermore, class labels are then determined with the majority votes by considering the weightings of the distances. The advantage of employing a K-NN is its simplicity in the implementation, but this is outweighed by the fact that K-value needs to be selected.

III. Overview of a Neutral Section

The conventional on-board switching scheme employed in the Transnet railway has three main components:

1. Induction magnets (In-between the rails).
2. Arthur Flury NS (Phase break).
3. On-board magnet sensors

Fig. 1, illustrates the current setup of NS in the Transnet railway line: (A) is the "N" marker, (B) Arthur Flury phase break and (C) induction magnets. While the magnet sensors are not shown, they are installed underneath the locomotives [1].

IV. Dataset

The dataset contains a total of 311 images split into 228 training data set and 83 test data set. The 228-training data set contains, open (131), close (60) and negative (37) images. The test data set contains, open (48), close (22) and negative (13) images. Fig. 2 illustrates samples of the dataset images used in this study. The open marker ("N") is used to switch a locomotive off when entering a NS and the close ("C") turns the locomotive on as it traverses to another phase of the NS: employing a CV system. The negative or invalid ("I") images are used to train the classifier to recognise any invalid Region of Interest (RoI). In [1], they employed a muli-class ("N", "C" and "I") rather than a binary class ("N" and "C") allowing the model to distinguish between the two valid classes from the invalid class.

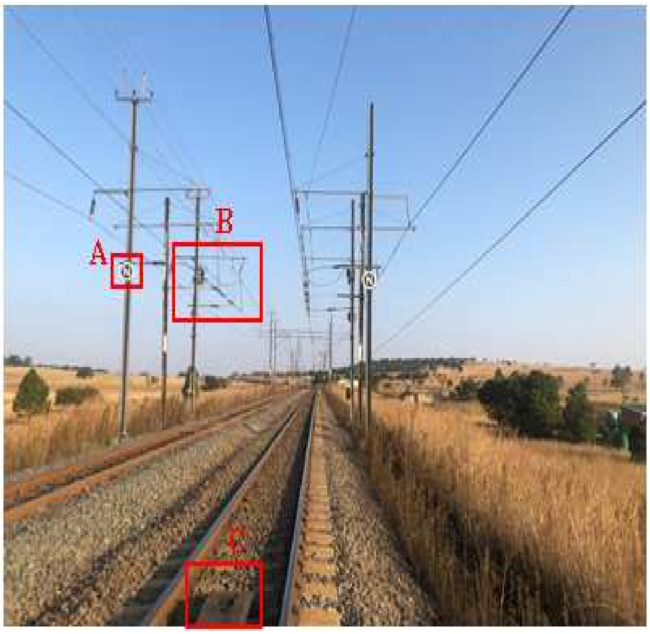


Fig. 1. Arthur Flury NS setup

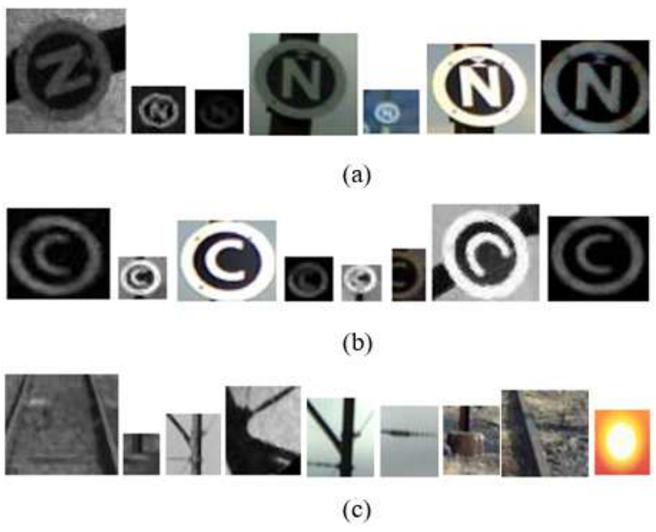


Fig. 2. Dataset sample. Open (a), Close (b) and Negative (c) images

V. Mathematical implementation and algorithms of ML classifiers

This paper presents the performance of seven ML classifiers which were reviewed in the literature section. In this section the focus is on the mathematical equations that define each classifier along with algorithms:

A. *Decision Tree (DT) classifier*

Fig. 3 illustrates the basic structure of a DT classifier with root, internal and leaf nodes.

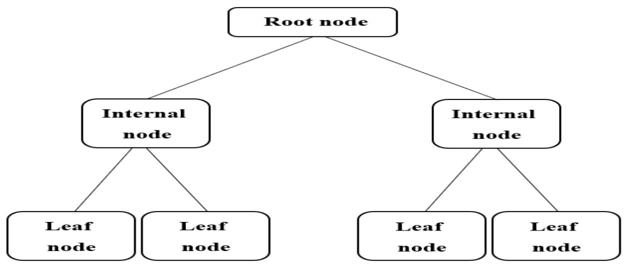


Fig. 3. DT classifier structure [2]

When an ID3 is employed in a DT classifier, an entropy criterion is used: equation 1 defines the entropy criterion. Where $\boldsymbol{H(R)}$ denotes the entropy criterion with $\boldsymbol{p_k}$ being the proportion of objects of the class ($\boldsymbol{k}$) in set/node $\boldsymbol{R}$, and $\boldsymbol{K}$ describes the number of classes. The information gain can therefore be obtained from equation 1 as the difference between the entropy calculated before and the entropy calculated after.

$$H(R) = -\sum_{k=1}^{K} p_k \, log\, p_k \qquad (1)$$

A DT classifier that employs a CART algorithm also mentioned in the literature, uses a Gini impurity also denoted as $\boldsymbol{H(R)}$. Equation 2 describes the mathematical expression that governs a DT classifier that employs a CART algorithm.

$$H(R) = -\sum_{k=1}^{K} p_k \,(1 - p_k) \qquad (2)$$

$$= 1 - \sum_{k=1}^{K} p^2{}_k$$

TABLE I. is a summary of a DT classifier with ID3 or CART algorithm:

TABLE I. Algorithm: DT classifier (ID3 or CART)

```
Algorithm: DT classifier
1. Set the data set **R** as the *initial root node.*
2. **for** every data set **R**:
3.   Calculate $H(R)$ in unused attribute (**A**) [equ.1 or 2]
4. **end**
5. **if** $A < H(R)$: *(new root node)*:
6.   Split data set **R** into subsets *(internal nodes)*
7.   Make a decision tree node *(leaf nodes)*
8.   Recurse on each unused subset
9. **end**
```

B. *Convolutional Neural Network (CNN) classifier*

Fig. 4 depicts the basic construction layers involved in a CNN network. While the CNN can extract feature maps of the markers (RoI's) from the input image; in this study, HoG features were used.

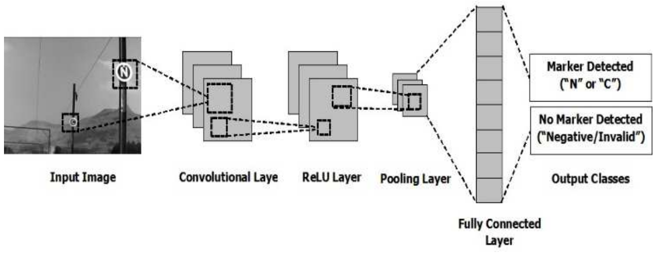


Fig. 4. CNN classifier structure [3]

The input images (RoI's) are firstly resized and then convolved with kernel/filters ($\boldsymbol{k}$) resulting in feature map $\boldsymbol{F(map)}$ defined in equation 3. Where image input size is $\boldsymbol{m}$, kernel size is $\boldsymbol{n}$, the padding is $\boldsymbol{p}$, and the stride is denoted by $\boldsymbol{S}$ in the equation. The ReLU activation function ($\boldsymbol{f}$) in equation 4 models a neuron's output as a function of the input ($\boldsymbol{x}$) with an activation threshold at zero. Equation 5 then outputs new feature maps after pooling which are then used to create a fully connected layer. In between the fully connected layer and output classes, a loss function is employed to calculate the predicted error. Commonly, a Softmax function is used [3]. TABLE II. is a concise algorithm of the CNN network implementation.

$$F(map)_{conv} = \frac{m - n + 2p}{S} + 1 \qquad (3)$$

$$f(x) = max(0, x) \qquad (4)$$

$$F(map)_{pool} = \frac{m-n}{S} + 1 \qquad (5)$$

TABLE II. ALGORITHM: CNN CLASSIFIER

Algorithm: CNN classifier

```
1. Resize RoI's to m*m*r.
2. for every resized RoI :
3.   Compute $F(map)_{conv}$ using [equ. 3].
4.   Compute $f(x)$ using [equ. 4].
5.   Compute $F(map)_{pool}$ using [equ. 5].
6. end
7. Compute a fully connected layer.
8. Classify RoI's.
```

C. *Discriminant Analysis (DA) classifier*

In [4], discuss the structure of a DA classifier which is illustrated in Fig. 5 below.

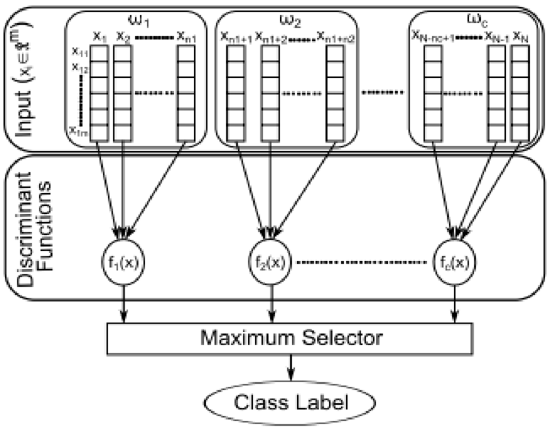


Fig. 5. DA classifier structure [4]

The predicted classification output of a multi-class dataset is described in equation 6 below [9]. The posterior probability denoted $\hat{\boldsymbol{P}}(\boldsymbol{k}|\boldsymbol{x})$ is defined as the product of prior probability $\hat{\boldsymbol{P}}(\boldsymbol{k})$ and multivariant normal density $\hat{\boldsymbol{P}}(\boldsymbol{x}|\boldsymbol{k})$. While $\boldsymbol{C}(\boldsymbol{y}|\boldsymbol{k})$ is the cost of classifying an observation as $\boldsymbol{y}$ when class $\boldsymbol{k}$ is its true class. The number of classes (***K***) gives range of $\boldsymbol{y}$, where {***y=1,…., K***}. Equation 7 defines the multivariant normal density or $\hat{\boldsymbol{P}}(\boldsymbol{x}|\boldsymbol{k})$ with mean $\boldsymbol{\mu_k}$ of ***1-by-d*** and covariance $\sum_{\boldsymbol{k}}$ of ***d-by-d*** at ***1-by-d*** point **x** [4, 9]. TABLE III. presents a generic LDA and QDA classifier algorithm [4].

$$\hat{y} = arg(min) \sum_{k=1}^{K} \hat{P}(k|x)\, C(y|k) \qquad (6)$$

$$P(x|k) = \frac{1}{((2\pi)^d |\sum_k|)^{1/2}} e^{\left(-\frac{1}{2}(x-\mu_k)\sum_k^{-1}(x-\mu_k)^T\right)} \qquad (7)$$

TABLE III. ALGORITHM: LDA AND QDA CLASSIFIER

Algorithm: DA classifier

```
1. Spilt dataset into training and testing.
2. if dataset has high dimensions:
3.   Employ Regularised LDA:
4.     Regularisation parameter $(1 > \eta > 0)$ or,
5.   Employ Subspace method:
6.     Principal Component Analysis (PCA).
7. end
8. Train model using the training data set [equ. 7].
9. Predict the class of the test data [equ. 6].
```

D. *Naïve Bayes classifier*

Equations 8-11 are the mean ($\boldsymbol{\mu}$), standard deviation ($\boldsymbol{\sigma}$), Gaussian density function ($\boldsymbol{g}(\boldsymbol{a},\boldsymbol{\mu},\boldsymbol{\sigma})$) and a prediction ($\boldsymbol{P}(\boldsymbol{a_K}|\boldsymbol{\gamma_k})$) respectively. TABLE IV. presents an algorithm for a Naïve Bayes classifier [5].

$$\mu = \frac{1}{K}\sum_{k=1}^{K} \alpha_k \qquad (8)$$

$$\sigma = \frac{1}{(K-1)}\sum_{k=1}^{K} \sqrt{(\alpha_k - \mu)} \qquad (9)$$

$$g(a,\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(a-\mu)^2}{2\sigma^2}} \qquad (10)$$

$$P(a_K|\gamma_k) = g\left(\alpha_k, \mu_{\gamma_k}, \sigma_{\gamma_k}\right) \qquad (11)$$

TABLE IV. ALGORITHM : NAÏVE BAYES CLASSIFIER

Algorithm: Naïve Bayes classifier

```
1. Spilt dataset into training and testing.
2. Separate data set by class.
3. Calculate mean [equ. 8].
4. Calculate standard deviation [equ. 9].
5. Calculate Gaussian density function [equ. 10].
6. Calculate class probability.
7. Predict [equ. 11].
```

E. *Support Vector Machine (SVM) classifier*

Fig. 6 is a depiction of an SVM with non-linear features transformed by the kernel to feature space where the hyperplane separates the two classes.

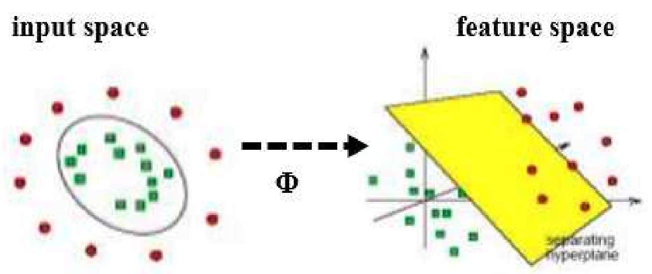


Fig. 6. SVM classifier with kernel and hyperplane [10]

Equation 12 defines the hyperplane, while equation 13 is the Lagrangian function ($\boldsymbol{L_p}$) that finds the optimal hyperplane. Equation 14 is the kernel function ($\boldsymbol{K(x \cdot x_i)}$): separates non-linear features to linearly separable feature space. The training data is defined as $\{\boldsymbol{x_i};\ \boldsymbol{y_i}\}$, where $\boldsymbol{i} = \boldsymbol{1,\ldots,}\ l$ (being training points or features) with ($\boldsymbol{x_i \in R^d}$), where $\boldsymbol{d}$ represents inputs and $\boldsymbol{y_i} \in \{\boldsymbol{-1,+1}\}$ denote classes for open ("N") or close ("C") markers. The weight vector, bias constant and Lagrange multiplier are denoted ($\boldsymbol{w}$), ($\boldsymbol{b}$)and ($\boldsymbol{\alpha}$) respectively. TABLE V. is a concise algorithm of the SVM classifier implementation.

$$y_i(w \cdot x_i + b) - 1 \geq 0\ \forall_i \tag{12}$$

$$L_p \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L} a_i y_i (w.x_i + b) + \sum_{i=1}^{L} a_i \tag{13}$$

$$K(x \cdot x_i) = \big(\Phi(x).\Phi(x_i)\big) \tag{14}$$

TABLE V. ALGORITHM: SVM CLASSIFIER

Algorithm: SVM classifier
1. Spilt dataset into training and testing.
2. **for** and class $\{\boldsymbol{-1,+1}\}$ **:**
3.   Map $\boldsymbol{x_i \in R^d}$ [equ. 14].
4.   Obtain optimal hyperplane [equ. 12 & 13].
5. **end**
6. Predict.

F. *AdaBoost classifier*

The AdaBoost algorithm as stated previously is an ensembled classifier which trains multiple classifiers to improve the classification accuracy. The literature [7] covered the DT and SVM; therefore, since these classifiers have already been covered, only the performance of AdaBoost will be discussed.

G. *K-Nearest Neighbors (K-NN) classifier*

Since the K-value of the K-NN algorithm is selected; the paper discusses the Euclidean distance ($\boldsymbol{d}$) in equation 15 as it is commonly used [8]. Considering ($\boldsymbol{x_i,\ x_j, y_i, y_j}$) $\in \boldsymbol{D}$ as the training set where ***i, j=1…., l***: and (***x, y***) are the feature points. TABLE VI. is also a concise algorithm of the K-NN classifier implementation.

$$d(x,y) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{15}$$

TABLE VI. ALGORITHM: K-NN CLASSIFIER

Algorithm: K-NN classifier
1. Select K-value.
2. Calculate Euclidean distance for $\boldsymbol{K}$ number of neighbours [equ. 15].
3. Calculate K-NN as per the calculated Euclidean distance.
4. Count the number of data points and as them.
5. Predict.

VI. PERFORMANCE RESULTS

The results obtained were performed using the Classification Learner application in MATLAB 2022a: on a Lenovo ThinkPad with an Intel core processor (i5-10210U) running at 1.60GHz@16GB RAM with a Windows 10 OS.

The study used the dataset in Fig. 2 and employed a HoG feature extractor with a [4 4] cell-size [1]. Furthermore, the selection of HoG features was based on comparing other feature extractors such as SURF, SIFT and LBP. Fig. 7 demonstrates that LBP features at [32 32] cell-size computes faster at 0.69s (even with cell-size below), and HoG [4 4] follows behind at 1.39s. SIFT shows it is computationally taxing at 3.43s with a Block Size (BS) of 3. The latter is followed by SURF features, viz. BS at 23 and Feature Size (FS) at 64, the computation time is 2.57s. However, Fig. 8 shows that HoG features perform better with an accuracy of 85.71% while the LBP features at [32 32] cell-size are the least performing at 65.48% accuracy.

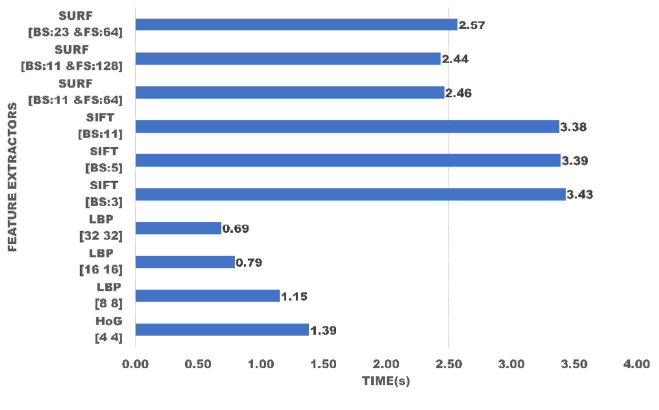


Fig. 7. Computational: HoG [4 4] vs SURF, SIFT and LBP

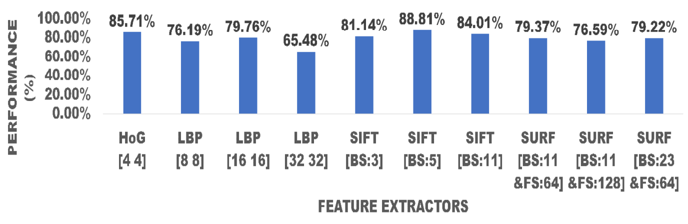


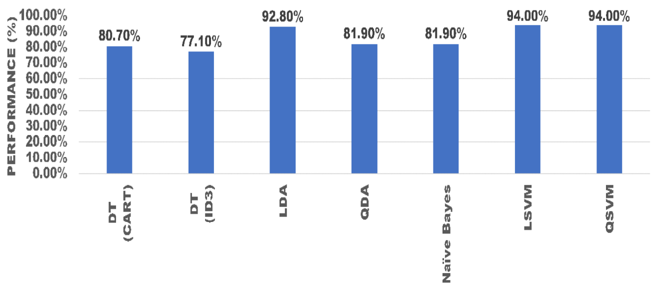
Fig. 8. Performance: HoG [4 4] vs SURF, SIFT and LBP

The HoG features are used to train and validate the performance of each ML classifier as shown in TABLE VII. In each classifier during training and testing a confusion matrix with 2-fold cross-validation was employed. The accuracy of each classifier was then obtained by employing an F1-measure [1]. Fig. 9 is a graphical representation of each classifier's performance during testing, however, TABLE VII. can be referred to for training performance. The focus is on the performance of each classifier with new data, hence the graphical representation of test performance.

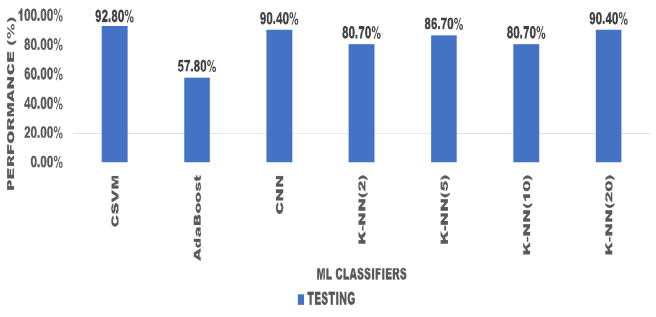
Firstly, considering the performance of each classifier in Fig. 9(a) and (b): LSVM and QSVM take first place with an accuracy of 94%, second place is LDA and CSVM at 92.80%, and third is CNN plus K-NN(20) achieving 90.40%. Note, that the number within the parentheses in K-NN denotes the selected K-value. The AdaBoost results show that it is the worst performing classifier at 57.80%; even during training it still performed last at an accuracy of 57.50% when compared to the other classifiers. Secondly, considering the prediction speed/computation time summarised in TABLE VII. : CNN takes the lead at 82 objects per second (obs/sec), followed by LDA at 78 obs/sec and in third place is LSVM at 75 obs/sec. The K-NN is the worst performing classifier in terms of prediction speed even at different K-values (2, 5, 10, and 20) reaching 12 – 14 obs/sec.

TABLE VII. ML CLASSIFIERS: TRAINING AND TESTING ACCURACY

	Classifier	Training	Testing	Prediction Speed		Classifier	Training	Testing	Prediction Speed
1.	DT (CART)	75.40%	80.70%	72 obs/sec	**8.**	CSVM	93.00%	92.80%	74 obs/sec
2.	DT (ID3)	74.10%	77.10%	73 obs/sec	**9.**	AdaBoost	57.50%	57.80%	68 obs/sec
3.	LDA	94.70%	92.80%	78 obs/sec	**10.**	CNN	90.80%	90.40%	82 obs/sec
4.	QDA	85.10%	81.90%	71 obs/sec	**11.**	K-NN(2)	82.00%	80.70%	13 obs/sec
5.	Naïve Bayes	85.10%	81.90%	26 obs/sec	**12.**	K-NN(5)	82.90%	86.70%	12 obs/sec
6.	LSVM	93.40%	94.00%	75 obs/sec	**13.**	K-NN(10)	84.60%	80.70%	14 obs/sec
7.	QSVM	93.90%	94.00%	68 obs/sec	**14.**	K-NN(20)	83.30%	90.40%	12 obs/sec



(a)



(b)

Fig. 9. Performance of ML classifiers

VII. CONCLUSION

This paper employed HoG features with cell-size of [4 4] in comparing seven ML classifiers with different parameters. The LDA outperformed the other classifiers by achieving a 94.70% during training. However, after testing it achieved 92.80% while the LSVM and QSVM achieved 94%. Fig. 9 shows that LSVM and QSVM are the best performing classifiers when new data is introduced, and AdaBoost is the least performing at 57.80%. In terms of the obs/sec, CNN predicts faster at 82 obs/sec, LDA behind at 78 obs/sec followed by LSVM predicting at 75 obs/sec. The K-NN prediction speed at different K-values was the slowest classifier reaching 12 – 14 obs/sec. Therefore, considering both criteria (performance and prediction speed) the LVSM is a better choice for the dataset presented in Fig. 2 and also used in [1].

REFERENCES

[1] C. T. Mcineka and S. Reddy, "Automatic Switching of Electric Locomotives in Neutral Sections," in *Conference on Information Communications Technology and Society (ICTAS)*, 10-11 March 2021, pp. 97-102.

[2] S. Mitrofanov and E. Semenkin, "An Approach to Training Decision Trees with the Relearning of Nodes," in *International Conference on Information Technologies (InfoTech)*, 16-17 Sept. 2021, pp. 1-5.

[3] L. Alzubaidi *et al.*, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data,* vol. 8, no. 1, pp. 1-74, 2021.

[4] A. Tharwat, "Linear vs. quadratic discriminant analysis classifier: a tutorial," *International Journal of Applied Pattern Recognition,* vol. 3, no. 2, pp. 145-180, 2016.

[5] H. R. Seth and H. Banka, "Hardware implementation of Naïve Bayes classifier: A cost effective technique," in *3rd International Conference on Recent Advances in Information Technology (RAIT)*, 3-5 March 2016, pp. 264-267.

[6] M. Awad and R. Khanna, "Support vector machines for classification," in *Efficient learning machines*: Springer, 2015, pp. 39-66.

[7] M. Barstuğan and R. Ceylan, *Comparison of Decision Tree and SVM Based AdaBoost Algorithms on Biomedical Benchmark Datasets*. 2014.

[8] H. Yigit, "A weighting approach for KNN classifier," in *International Conference on Electronics, Computer and Computation (ICECCO)*, 7-9 Nov. 2013, pp. 228-231.

[9] MathWorks, "Prediction Using Discriminant Analysis Models," in *Help Center*, ed. [Online]: https://www.mathworks.com/help/stats/prediction-using-discriminant-analysis-models.html.

[10] H. Bhavsar and M. H. Panchal, "A review on support vector machine for data classification," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET),* vol. 1, no. 10, pp. 185-189, Dec 2012.