Comparative Performance of Deep Learning and Machine Learning Algorithms on Imbalanced Handwritten Data

A'inur A'fifah Amri, Amelia Ritahani Ismail,* Abdullah Ahmad Zarir
Department of Computer Science, Kulliyyah of ICT
International Islamic University Malaysia
P.O. Box 10, 50728 Kuala Lumpur, Malaysia

***Abstract*—Imbalanced data is one of the challenges in a classification task in machine learning. Data disparity produces a biased output of a model regardless how recent the technology is. However, deep learning algorithms, such as deep belief networks showed promising results in many domains, especially in image processing. Therefore, in this paper, we will review the effect of imbalanced data disparity in classes using deep belief networks as the benchmark model and compare it with conventional machine learning algorithms, such as backpropagation neural networks, decision trees, naïve Bayes and support vector machine with MNIST handwritten dataset. The experiment shows that although the algorithm is stable and suitable for multiple domains, the imbalanced data distribution still manages to affect the outcome of the conventional machine learning algorithms.**

Keywords*—Deep belief networks; support vector machine; back propagation neural networks; imbalanced handwritten data; classification

I. Introduction

Imbalanced class in a dataset occurs when the dataset is not in the same amount of values among the parameters or classes. The majority class of the dataset is when the class has the most instances. The minority class of the dataset is when the class has the least instances. A few disadvantages prompted by imbalanced class data in a classification are over fitting, deficient class model and wrongly classified. Over fitting is a result of accuracy bias due to overwhelming data values in one class compared to missing values of another class. The model might give a high accurate result, but it is biased to the majority class.

The approach that will be focused on this paper is a review on the effects of imbalanced class in a handwritten dataset towards deep learning and machine learning algorithms. Deep learning is a part of machine learning algorithms that are recently introduced to solve complex, high-level abstract and heterogeneous datasets, especially image and audio data. There are several types of deep learning architectures, which are deep neural network (DNN), convolutional Neural Network (CNN), deep belief networks (DBN) and convolutional deep belief networks (CDBN). In this paper, we will focus on two deep learning algorithms, which are CNN and DBN. CNN is composed of one or more convolutional layers with fully connected layers at the end of it. CNNs are used in computer vision and acoustic modeling for automatic speech recognition (ASR). A deep belief network (DBN) is a probabilistic, generative model made up of multiple layers of hidden units. It can be seen as a composition of simple learning modules of Restricted Boltzmann Machine (RBM) that make up each layer.

Conventional machine learning algorithms such as back propagation neural network (BPNN), support vector machine (SVM), Naïve Bayes and decision trees are also included in the experiment to enhance performance comparison value between deep learning and traditional machine learning algorithms when an imbalanced class handwritten data is used as the training set.

This paper is organized as follows. Section 2 clarifies the definitions of imbalanced data, the effects of imbalanced data have for classification tasks and the application of any deep learning algorithms used to counter this problem. Basic concepts and the applications of DBN, CNN, BPNN, SVM, Naïve Bayes and decision tree algorithms are described in the same section. Section 3 explains the experimental setup of imbalanced class data classification using deep learning and machine learning algorithms. Section 4 interprets the result analysis of the experiments and conclusions are presented in Section 5.

II. Related Work

Encouraging results have been received upon the application of deep learning algorithms in text recognition [1], audio classification [2] and even abstract high-level domains such as emotional recognition [3]. However, these are applied to data that are distributed evenly. Not many imbalanced data problems have been solved using a deep learning method.

According to some papers [4]-[7], imbalanced class in a dataset refers to the disparity of data dispensation between the classes. The class that has more training values is called the majority class and the class that has the least or most missing data values are called the minority class [5]. Minority data class is a realistic problem that the real-world situation faced because most of the time, data are scarce, despite its importance. The examples of minority classes in real world problem are credit fraud detection [8] and cancer abnormalities diagnosis [6], [8]. It can be expensive if the new data needs labeling [9]. Unfortunately, most algorithms devised shown stable and promising performance when using

This research is supported by the International Islamic University Malaysia under the Research Initiative Grants Scheme (RIGS): RIGS16-346-0510

balanced data in classification tasks but showed otherwise when imbalanced data is used[4]. Prediction of minority class is presumed to have a higher error rate compared to the majority class and its test examples are often wrongly classified as well [10].

The imbalanced class could cause deficient classification models [6], [7]. The algorithm that performs on balanced dataset will not perform as good when using an imbalanced dataset [4], regardless how good the model is. In a work, an imbalanced multimedia dataset was used on CNN [5], and it shows that the error rate "fluctuate" compared to when using a balanced dataset, where the error rate continues to decrease. In a paper [6], the author used SVM as the main algorithm and showed that the effect of data disparity results in a "high false negative rate". Another paper [11] modified kNN algorithms to counter the effect of imbalanced data to the algorithm. Bootstrapping is often used to improve the algorithm performance when imbalanced data is used [6], [9].

A. Deep Belief Networks

To comprehend DBN, the concept of Restricted Boltzmann Machine (RBM) must first be explained. The architecture of RBM is it consists of a bidirectional connection between hidden layers and visible layers. This feature allows the weight to be connected exclusively and allows deeper extraction between the neurons. RBM is a probabilistic model[2] and a two-layer, bipartite, undirected graphical model with a set of binary hidden random variables (units) h of dimension K, a set of (binary or real-valued) visible random variables (units) v of dimension D, and symmetric connections between these two layers represented by a weight matrix ($W \in R^{D \times k}$) [12]. Two main RBM often used are Bernoulli, where visible and hidden layers are binary, and Gaussian is where the visible units are allowed to use real number values [3].

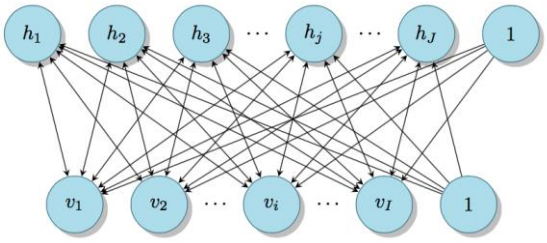

Fig. 1. Example of RBM architecture schematic design [12].

Fig. 1 above presents the schematic design of RBM architecture. RBM is made up of stochastic visible units and stochastic hidden units that are connected to each other [13].

A deep belief network (DBN) is a probabilistic, generative model made up of multiple layers of hidden units. It can be seen as a composition of simple learning modules that make up each layer. DBN is made up of stacked Restricted Boltzmann Machine (RBM) used greedily as depicted in Fig. 2. However, such feature results DBN to be computationally expensive and time-consuming because the number of layers DBN needs to go through is a lot.

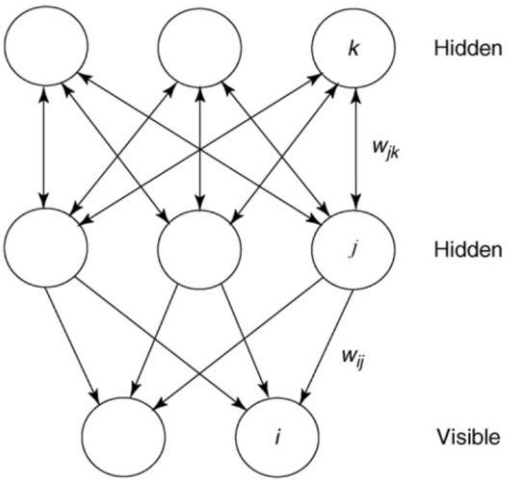

Fig. 2. A stacked RBM or known as DBN [14].

According to Le & Provost [3], training a DBN is expensive in terms of computation because pre-training took 11 minutes per epoch and fine-tuning takes up 10 minutes per epoch. DBN is used in emotions recognition [3] by learning high-level features. Face verification is also using DBN, despite the usage of CNN, the hybrid algorithm aims to achieve robustness in verifying similarities of different faces [15]. DBN is also used to model natural images [16] by learning multiple layers of unlabeled data.

B. Convolutional Neural Networks

Convolutional neural networks (CNN) consists of one or more convolutional layers [4], [5], [17], alternating with subsampling layers and by the end of the network, optionally, a fully connected MLP [4]. Basically, CNN architecture must consist of one or more convolutional, pooling and a fully connected layers on top [5]. The convolutional layers are responsible for feature extraction and is called feature map [4], [5], [17] and sometimes feature detection [18]. After convolutional layer, it is often paired up with a pooling layer that will perform a pooling function based on the inputs it received from the previous convolutional layer [4]-[7]. The pooling layer is also known as a subsampling layer, and it will alternate with a convolutional layer because it computes the statistics of the convolutional layer. The pooling layer will perform pooling functions and is called min-pooling, max-pooling layers or etc. according to its context of problem-solving. The pooling function will *"downsample"* the input it received from its convolutional layer [5]. Such will carry on until the end of the network. At the end of the series of alternation, a fully connected MLP will be added. It works as a classification module for the network [4]. This layer will receive all neurons from its previous layers whether they are convolutional or pooling and connect them with its own neurons [5]. The architecture is depicted in Fig. 3.

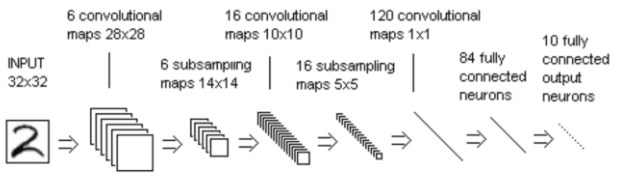

Fig. 3. Example of a convolutional neural network reading an image input [49].

However, the implementation of convolutional and subsampling layers in a CNN plus the method of the network training differs in every CNN [19]. It depends on the context of problems that are attempted to solve.

C. Back Propagation Neural Networks

Artificial neural networks (ANN) are modeled after the human brain networks [20], [21]. It is widely known used for supervised learning and recognizing patterns from input dataset by weight adjustments [20]. There are many examples of ANN such as feed forward and radial basis function (RBF). ANN's ability to scrutinize nonlinear data and to design complex models has allows it to be applied in studies of different fields [21], [22]. Fig. 4 shows an example of ANN architecture.

The most common neural network algorithm used is the back propagation neural network (BPNN). BPNN has three layers, which consists of an input layer, a hidden layer and an output layer [20]. The layers are made up of interconnected nodes by a weighted association and the number of nodes in the layers depends on the problem domain [21]. The input layer will accept the data for training or testing and pass the weights to the connected hidden layer. Hidden layer can be one or more and it will continue calculating the weights it received and send it to the output layer where the result is produced. BPNN compares its real output and target output and adjusts its weight according to the error and propagates back to its network.

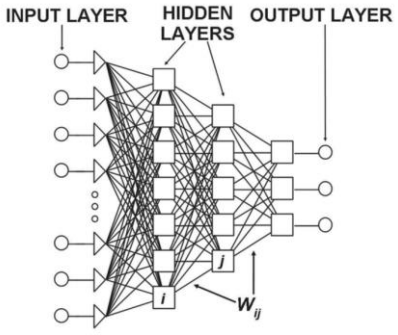

Fig. 4. An example of a neural network with two hidden layers [21].

Arora [23] implemented back propagation neural network classifier to categorize Devnagari handwritten classes and compared its performance with SVM using the same handwritten dataset. The experiment result for BPNN performance is 90.44% for testing dataset accuracy. Another work [24] proposed a diagonal based feature extraction and used a "feed forward back propagation" neural network to classify the data based on the new feature extraction and achieved 96.52% with 54 features and 97.84% with 69 features accuracy rate.

In tackling imbalanced data, Cao and et al. [25] presented a cost sensitive back propagation neural network for a multiclass imbalanced data, as opposed to the "limited" binary class imbalanced data [20].

D. Support Vector Machine

According to Arora [23], SVM can be defined as a 'binary classifier', where the outcome will be divided into two groups based on the optimum hyperplane. Fig. 5 depicts the definition of SVM in pictorial form.

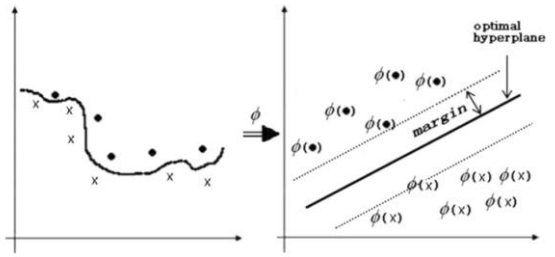

Fig. 5. SVM mapping nonlinear problem to linear using optimum hyperplane [46].

Niu & Suen [26] implemented a hybrid of SVM and CNN for classifying MNIST handwritten digits dataset. Feature extraction is done using CNN and SVM acts as a 'recognizer'. Arora [23] compared the performance of SVM and ANN using the Devnagari handwritten recognition problem. SVM performance in the experiment achieved 92.38% for testing accuracy.

In countering the imbalanced data classification problem using SVM, its weight and activation function are manipulated in order to increase the classification accuracy [27]. Tang and et al. [28] stated that SVM outperforms other conventional classifiers when a moderate imbalanced data is used. Even so, when a high imbalanced data is used instead, SVM classifier can still produce a biased result. Most works using SVM to counter imbalanced data only focused on the performance and not efficiency, hence, SVM can be a slow classifier [28]. However, Zou and et al. [29] stated that SVM could not perform imbalanced data classification successfully based on the works of Cristianini & Shawe-Taylor [30].

In Big Data domain, Koturwar and et al. [31] stated that SVM has the ability to balance massive data correctly. Feature extraction using SVM is good as it can be done promptly using SVM kernel instead of a feature extraction process that results to data lost [31].

One of the disadvantages of SVM classifier is its training and execution is very complex caused it to be implemented in mostly small category set problem[23]. According to Koturwar and et al. [31], large training data makes SVM inefficient and costly, as SVM is not scalable to huge size data. When the training data is noisy and imbalanced, it can affect the outcome of SVM due to its high training execution and low generalization error [31].

E. Naïve Bayesian

Naive Bayes (NB) is a supervised probabilistic classifier that is based on the Bayes' theorem with the assumption the attributes of the data are discrete [32]-[34]. NB calculates the conditional probability of the features and choose the class with the highest value [34].

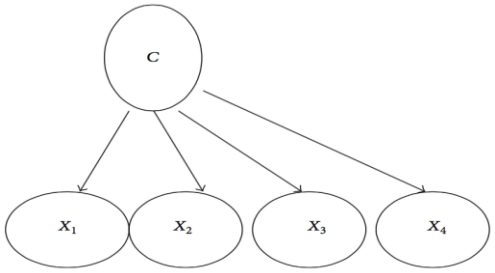

Fig. 6. An example of Naive Bayes structure [35].

Bal and et al. [35] suggests that the NB is made up of one classification node that acts as the parent nodes for all the rest of the nodes as shown in Fig. 6. According to Kumar and et al. [33], Bayes theorem suggests that a problem case to be classified is represented by vector x = (x_1,..., x_n) with n independent features. It brings to the instance probability, p(C_k|x_1,..., x_n) for each K possible outcomes. The equation is summarised as below:

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)} \quad (1)$$

where,

p(C_k) = probability of class *k*,

$p(x|C_k)$ = probability of query *x* given class *k*,

$p(x)$ = probability of query *x*.

This allows the supervised learning to be implemented solely on logical and statistical calculation [36]. Therefore, NB is suitable as a solution for predictive and diagnostic problem [36]. Due to its ability to determine hypothesis by calculating probabilities, NB is robust to input data noises [36]. NB provides stable performance for a bank dataset [32] with an accuracy rate of 89%. Dey et al. (2016) stated that the performance comparison in sentiment analysis of movie and hotel reviews datasets, NB algorithm outperform k-NN with over 80% accuracy rate. Ahmed and et al. [37] proposed a hybrid of NB and Apriori algorithm to detect SMS spam and achieved the accuracy of 98.7% as compared to 97.4% accuracy using traditional NB. In another classification task, Sapkale and Nair [36] used NB as a method to improve domain classification of Google search results. The experiment resulted shorter performance time with the same domain classification rate.

In imbalanced class dataset problem, Imran and et al. [38] applied NB on an imbalanced educational data by using Weka tool and achieved accuracy rate of 68.2432%. Sharma and et al. [39] reviewed the recognition performance of NB algorithm on handwritten Gujarati character data and acquired 96.43% of classification accuracy and F-measure. In another work, a comparison of performance using NB for writers' identification through their handwriting in English language was done [40]. The accuracy result based on aggregated feature attained by NB is 85%. Sarangi and et al. [41] recorded the experiment involving handwritten Odia numerals by performing LU factorization as feature extraction and then classify the dataset using NB. Although the experiment focuses on feature extraction instead of the classifier, overall accuracy result from number 0 until 9 are between 74.39% and 85%.

F. Decision Trees

Decision trees (DT) produce an output based on the series of binary decision in the model called in the form of dendritic graph [42], [43]. It presents all possible output with the path leading to the output [35] as shown in Fig. 7. Tree pruning is a method of downsizing tree size by eliminating nodes that does not give accuracy in result [42].

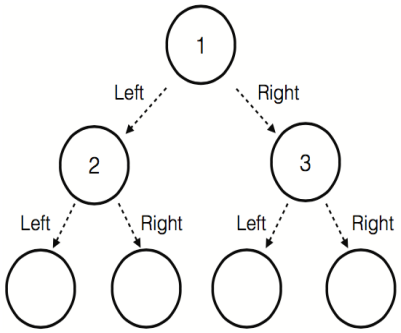

Fig. 7. An example of a decision tree structure deciding output path [44].

According to Y. Zhang and et al. [42], DT is suitable for decision analysis as it can show the strategy to achieve a solution. Analysis using DT is simple because the connection between the input and output is clear [43], [45]. Besides that, DT is able to operate on both numerical and categorical data [43].

One of the few disadvantages of DT in Big Data domain is that the large sets of data will cause more time take to construct a traditional DT [31]. Menickelly et al. [43] mentions that DT is not robust to different training data, which can result low accuracy performance.

One application of DT is in the medicine field where the algorithm is implemented to classify Parkinsonian Syndromes using FDG-PET brain dataset. The algorithm correctly classifies with ranges from 47.4% to 80.0% of accuracy [45].

III. Experimental Setup

For this experiment, an imbalanced dataset that is suitable for classification task is selected. Then, the source code of CNN and DBN is modified to suit the dataset, which is extended from [48]. Then, the preliminary results CNN and DBN are recorded and further evaluated. The experimental dataset used in this experiment is MNIST handwritten digit dataset as many experiments have used the dataset as a benchmark [1], [12], [47]. The dataset is preprocessed and consists of 4 files, 2 training files, and 2 testing files.

The training set contains 60000 examples, and the test set 10000 examples. However, since the main aim of this paper is to review the data disparity and the algorithm's performance, the data has been modified to a smaller size but imbalanced. The labels values are 0 to 9. Pixels are organized row wise and the values are between 0 and 255. 0 means background (white), 255 means foreground (black). The images were centered in a 28x28 image. Data distribution is described in Table I below together with their percentages.

TABLE I. DATA DISTRIBUTION OF MNIST DATASET

Labels	Number of data	Imbalance Percentage (%)
0	500	100
1	45	9
2	150	30
3	250	50
4	150	30
5	35	7
6	25	5
7	200	40
8	350	70
9	15	3

IV. ANALYSIS OF RESULTS

Accuracy rate calculates the number of correct predictions out of the number of all predictions. Classification error shows is the number of wrongly predicted number out of all predictions. Kappa statistics takes into account the correct predictions made by chance and is between 0 and 1. Weighted mean recall calculates class recall or sensitivity for each class. Weighted mean precision calculates through class precisions for individual classes. Absolute error presents the average absolute deviation of the prediction from the actual value. Relative error is the average of the absolute deviation of the prediction value.

Table II presents the results of DBN, CNN and DNN. The accuracy rate of DBN is 92.5% and is the highest accuracy among the three deep learning algorithms. The classification error is 7.5%, which is a promising result. CNN has 10% accuracy rate and 90% classification error rate. DNN achieved accuracy rate of 27.91% and classification error of 71.57%.

For DBN, the kappa statistics result is 0.893, which is very high. CNN has kappa statistics of 0.0. Kappa statistics for DNN is -0.001, which is below than 0. Hence, it means that the two observers are agreeing even less.

The weighted mean recall for DBN is 90.4% or 0.9 and is agreeable since it is more than 0.5. For CNN, its weighted mean recall is 0.1% or 0.1, which is low. The weighted mean recall for DNN is 9.91% or 0.0991, which is not good as it is less than 0.5.

For DBN, the weighted mean precision is 91.5% or 0.915, which is good since it values more than 0.5. CNN achieved 1.0% or 0.01 for its weighted mean precision. The result is not good as it is less than 0.5 compared to DBN. The weighted mean precision for DNN is 5.79% or 0.0579, which is not good as it is also less than 0.5.

TABLE II. RESULTS FOR DEEP LEARNING ALGORITHMS

	DBN	CNN	DNN
Accuracy rate	92.5%	10%	27.91%
Classification error	7.5%	90%	71.57%
Kappa	0.893	0.0	-0.001
Weighted mean recall	90.4%	0.1%	9.91%
Weighted mean precision	91.5%	1.0%	5.79%
Absolute error	0.301	2.9	0.721
Relative error	30.1%	290%	72.13%
Root mean squared error	1.14	3.52	0.84
Squared error	1.3	12.39	0.706
Processing time	3 hours 47 minutes	37 minutes 26 seconds	2 hours 2 minutes 35 seconds

The absolute error for DBN is 0.301. The result is quite low. CNN has the highest absolute error, which are 2.9. DNN achieved 0.721 in absolute error, which are high although not as high as CNN.DBN achieved relative error of 30.1% and is the lowest among the three deep learning algorithms. CNN has 290% relative error and is the highest. Relative error for DNN is 72.13% or 0.7213 and it is quite high as well.

Root mean squared for DBN is 1.14 and its squared error achieved 1.3. CNN has root mean squared value of 3.52 and squared error of 12.39. Root mean squared error of DNN is 0.84 and its squared error achieved 0.706. All of the values are high as they are more than 0.5. However, DNN has the least root mean squared and squared error as compared to DBN and CNN.

The processing time for DBN is 3 hours and 47 minutes. It is the longest processing time compared to CNN and DNN. CNN has the shortest processing time at 37 minutes 26 seconds. The processing time for Deep Learning to compute the dataset is 2 hours 2 minutes and 35 seconds.

Table III presents the results of BPNN, SVM, Decision tree and Naïve Bayes. The accuracy rate of BPNN is 23.9% with 77.03% classification error. SVM has 23.43% accuracy rate and 77.21% classification error. Decision tree has 29.07% accuracy rate which is the highest among four algorithms with the lowest classification error, which is 70.93%. Naïve Bayes has the lowest accuracy rate at 12.32% and the highest classification error at 87.69%.

TABLE III. RESULTS FOR MACHINE LEARNING ALGORITHMS

	BPNN	SVM	Decision tree	Naïve Bayes
Accuracy rate	23.9%	23.43%	29.07%	12.31%
Classification error	77.03%	77.21%	70.93%	87.69%
Kappa	0.032	0.010	0.000	0.017
Weighted mean recall	10.97%	10.18%	10.00%	15.43%
Weighted mean precision	11.17%	10.07%	2.91%	9.42%
Absolute error	0.826	0.772	0.823	0.877
Relative error	82.59%	77.21%	82.28%	87.67%
Root mean squared error	0.847	0.879	0.827	0.936
Squared error	0.717	0.772	0.685	0.876
Processing time	8 hours 16 minutes 2 seconds	3 minutes 14 seconds	28 seconds	5 seconds

Kappa statistics for BPNN is 0.032. SVM has kappa value at 0.010 while decision tree has 0 for its kappa statistics value. Lastly, Naïve Bayes achieved 0.017 for its kappa value. All four algorithms have very low kappa value, but the highest kappa statistics value is attained by BPNN.

Weighted mean recall for sensitivity of BPNN achieved 10.97% or 0.1097 and 10.18% or 0.1018 for SVM. Both algorithms have low sensitivity classifying the imbalanced class handwritten dataset. Decision tree has 10.00% or 0.1 value for its weighted mean recall. Naïve Bayes attained the highest weighted mean recall at 15.43% or 0.1543.

The weighted mean precision for all the algorithms are very weak as they are less than 0.5. BPNN achieved 11.17% or 0.1117 for its weighted mean precision, which is the highest among the four algorithms. SVM attained 10.07% or 0.1007 for its weighted mean precision. Decision tree has 2.91% or 0.0291, which is the lowest weighted mean precision obtained by the rest of the algorithms. The weighted mean precision for Naïve Bayes is at 9.42% or 0.0942.

The results of absolute error for all four algorithms are high because they are more than 50% rate. BPNN has absolute error of 0.826. SVM has the lowest absolute error out of four algorithms, which is 0.772. Decision tree has 0.823 absolute error values and Naïve Bayes achieved 0.877 for absolute error and is the highest.

Relative errors for all the algorithms are high as well since they achieved more than 50% rate. The relative errors for all the algorithms are similar to their respective absolute error. BPNN relative error rate is at 82.59% or 0.8259. SVM has the lowest relative error at 77.21% or 0.7721. Decision tree has relative error rate at 82.28% or 0.828 and Naïve Bayes has relative error rate of 87.67% or 0.8767.

Root mean squared error for all four algorithms is inflated as they almost achieve 100% or 1.0 rate. BPNN attained 0.847 root mean squared error rate while SVM has 0.879 for its root mean squared error rate. Decision tree has the lowest root mean squared error among the four algorithms, which is 0.827. Meanwhile, Naïve Bayes has the highest root squared mean error among the four algorithms at 0.936, which is near 1.0.

The squared error for BPNN is 0.717 and SVM is at 0.772. Decision tree has the lowest squared error out of the four algorithms at 0.685. Naïve Bayes has the highest squared error at 0.876.

The processing time varies for all the algorithms. BPNN has the most expensive processing time at 8 hours 16 minutes and 2 seconds. SVM took 3 minutes and 14 seconds to classify the data accordingly while decision tree took 28 seconds. Naïve Bayes is the least expensive out of the four algorithms as it took 5 seconds to compute.

V. CONCLUSIONS

Imbalanced class dataset affects the outcome despite the stability of the algorithm. The complexity of handwritten form of data also influenced the results of the algorithms. Therefore, all the algorithms have really low accuracy rate, which is below 50% and high classification error with poor performance. However, DBN managed to achieve high accuracy rate and low error rate according to the performance metrics as compared to the other algorithms. As a conclusion, DBN algorithm is stable and robust when an imbalanced handwritten dataset is utilized as an input.

ACKNOWLEDGMENT

This research is supported by the International Islamic University Malaysia under the Research Initiative Grants Scheme (RIGS): RIGS16-346-0510.

REFERENCES

[1] Wang, T., W, D. J., Coates, A., & Ng, A. Y. (2012). End-to-end text recognition with convolutional neural networks. ICPR.

[2] Dieleman, S., mon Brakel, P., & Schrauwen, B. (2011). Audio-based music classification with a pretrained convolutional network. ISMIR.

[3] Le, D., & Provost, E. M. (2013). Emotion recognition from spontaneous speech using hidden markov models with deep belief networks. IEEE.

[4] Hensman, P., & Masko, D. (2015). The impact of imbalanced training data for convolutional neural networks (Unpublished doctoral dissertation). KTH Royal Institute of Technology.

[5] Yan, Y., Chen, M., Shyu, M.-L., & Chen, S.-C. (2015). Deep learning for imbalanced multimedia data classification. IEEE International Symposium on Multimedia.

[6] Liu, Y., Yu, X., Huang, J. X., & An, A. (2010). Combining integrated sampling with SVM ensembles for learning from imbalanced datasets. Elsevier Ltd.

[7] Fernandez, A., Garcıa, S., & Herrera, F. (2011). Addressing the classification with imbalanced data: Open problems and new challenges on class distribution. Springer-Verlag Berlin Heidelberg.

[8] Chawla, N. V., Japkowicz, N., & Kolcz, A. (2004). Special issue on learning from imbalanced data sets. ACM Sigkdd Explorations Newsletter

[9] Berry, J., Fasel, I., Fadiga, L., & Archangeli, D. (2012). Training deep nets with imbalanced and unlabeled data. Interspeech.

[10] Weiss, G. M., & Provost, F. (2001). The effect of class distribution on classifier learning: An empirical study. Technical Report ML-TR-44.

[11] Liu, W., & Chaw, S. (2011). Class confidence weighted knn algorithms for imbalanced datasets. Springer.

[12] Lopes, N., Ribeiro, B., & Goncalves, J. (2012). Restricted boltzmann machines and deep belief networks on multi-core processors. IEEE World Congress on Computational Intelligence.

[13] Mohamed, A., Yu, D., & Deng, L. (2010). Investigation of full-sequence training of deep belief networks for speech recognition. Interspeech.

[14] Hinton, G. E. (2007). Learning multiple layers of representation. Elsevier Ltd.

[15] Sun, Y., Wang, X., & Tang, X. (2013). Hybrid deep learning for face verification. IEEE International Conference on Computer Vision.

[16] Aurelio, M., Krizhevsky, R. A., & Hinton, G. E. (2010). Factored 3-way restricted boltzmann machines for modeling natural images. Proceedings of the 13th In- ternational Conference on Artificial Intelligence and Statistics.

[17] Abdel-Hamid, O., Deng, L., & Yu, D. (2013). Exploring convolutional neural network structures and optimization techniques for speech recognition. Interspeech.

[18] Matsugu, M., Mori, K., Mitari, Y., & Kaneda, Y. (2003). Subject independent facial expression recognition with robust face detection using a convolutional neural network. Elsevier Science Ltd.

[19] Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011). Flexible high performance convolutional neural networks for image classification.

[20] Zhang, D., & Xu, W. (2014). A data-distribution based imbalanced data classification method for credit scoring using neural networks. IEEE.

[21] Amato, F., López, A., Peña-Méndez, E. M., Vanˇhara, P., Hampl, A., & Havel, J. (2013). Artificial neural networks in medical diagnosis. Journal of Applied Biomedicine.

[22] Drew, P. J., & Monson, J. R. T. (2000, January). Artificial neural networks. Surgery. Fan, J., Han, F., & Liu, H. (2014). Challenges of big data analysis. National Science Review.

[23] Arora, S., Bhattacharjee, D., Nasipuri, M., Malik, L., Kundu, M., & Basu, D. K. (2010, May). Performance comparison of SVM and ANN for handwritten devnagari character recognition. IJCSI International Journal of Computer Science Issues, 7.

[24] J.Pradeep, E.Srinivasan, & S.Himavathi. (2011). Diagonal based feature extraction for handwritten character recognition system using neural network. IEEE.

[25] Cao, P., Li, B., Zhao, D., & Zaiane, O. (2013). A novel cost sensitive neural network ensemble for multiclass imbalance data learning. IEEE.

[26] Niu, X. X., & Suen, C. Y. (2011, October). A novel hybrid CNN–SVM classifier for recognizing handwritten digits. Elsevier Ltd, 1318–1325.

[27] Hwang, J. P., Park, S., & Kim, E. (2011). A new weighted approach to imbalanced data classification problem via support vector machine with quadratic cost function. Elsevier Ltd, 8580–8585.

[28] Tang, Y., Zhang, Y.-Q., Chawla, N. V., & Krasser, S. (2009). SVMs modeling for highly imbalanced classification. IEEE Transactions on Systems.

[29] Zou, S., Huang, Y., Wang, Y., Wang, J., & Zhou, C. (2008). Svm learning from imbalanced data by GA sampling for protein domain prediction. IEEE Computer Society.

[30] Cristianini, N., & Shawe-Taylor, J. (2000). An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press.

[31] Koturwar, P., Girase, S., & Mukhopadhyay, D. (2015). A survey of classification techniques in the area of big data. arXiv preprint arXiv:1503.07477.

[32] Patil, T. R., & Sherekar, M. S. S. (2013). Performance analysis of naive bayes and j48 classification algorithm for data classification. International Journal Of Com- puter Science And Applications.

[33] Kumar, J., Prasad, S. S., & Pal, S. (2016). Irism @ ntcir-12 temporalia task: Experiments with maxent, naive bayes and decision tree classifiers. Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies.

[34] T, T., MS, V., & S, K. (2013). Supervised learning approach for tamil writer identity prediction using global and local features. International Journal of Research in Engineering and Technology, 2(5).

[35] Bal, M., Amasyali, M. F., Sever, H., Kose, G., & Demirhan, A. (2014). Performance evaluation of the machine learning algorithms used in inference mechanism of a medical decision support system. The Scientific World Journal.

[36] Sapkale, D., & Nair, D. P. S. (2016). An improved domain classification of Google search results using abstract: naive Bayes classifier. International Journal of Engineering Science and Computing.

[37] Ahmed, I., Guan, D., & Chung, T. C. (2014). Sms classification based on naïve bayes classifier and apriori algorithm frequent itemset. International Journal of Machine Learning and Computing.

[38] Imran, M., Afroze, M., Sanampudi, D. S. K., & Qyser, D. A. A. M. (2016, June). Data mining of imbalanced dataset in educational data using weka tool. International Journal of Engineering Science and Computing.

[39] Sharma, A., Thakkar, P., Adhyaru, D. M., & Zaveri, T. H. (2016). Features fusion based approach for handwritten gujarati character recognition. Nirma Univeristy Journal Of Engineering And Technology.

[40] V, R. S., & S, V. M. (2015). Predicting the identity of a person using aggregated features of handwriting. Advances in Image and Video processing.

[41] Sarangi, P. K., Ahmed, P., & Ravulakollu, K. K. (2014). Naïve bayes classifier with lu factorization for recognition of handwritten Odia numerals. Indian Journal of Science and Technology.

[42] Zhang, Y., Wang, S., Phillips, P., & Ji, G. (2014). Binary pso with mutation operator for feature selection using decision tree applied to spam detection. Elsevier Ltd..

[43] Menickelly, M., Gunluk, O., Kalagnanam, J., & Scheinberg, K. (2016). Optimal gener- alized decision trees via integer programming. Industrial and Systems Engineering.

[44] Baumann, F., Vogt, K., Ehlers, A., & Rosenhahn, B. (2015). Probabilistic nodes for modelling classification uncertainty for random forest. 14th IAPR International Conference on Machine Vision Applications (MVA).

[45] Mudali, D., Teune, L. K., Renken, R. J., Leenders, K. L., & Roerdink, J. B. T. M. (2015). Classification of parkinsonian syndromes from fdg-pet brain data using decision trees with ssm/pca features. Computational and Mathematical Methods in Medicine.

[46] Ren, J. (2012). Ann vs. svm: Which one performs better in classification of mccs in mammogram imaging. Elsevier Ltd.

[47] Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. IEEE.

[48] Amri, A.A, Ismail, A. R, Zarir, A. A. (2017). Convolutional Neural Networks and Deep Belief Networks for Analysing Imbalanced Class Issue in Handwritten Dataset. International Journal on Advanced Science, Engineering and Information Technology (IJASEIT), 7(6), pp 2302-2307.

[49] Mrazova, I, & Kukacka, M. (2012). Can Deep Neural Networks Discover Meaningful Pattern Features? Procedia Computer Science 12, pp 194-199.