ML & DL Based Classification Models for MNIST Data Set

Raksha Vishwanath *dept. of CSE, Suny Buffalo email: rakshavi@buffalo.edu*

Rohan Kunchakuri *dept. of CSE, Suny Buffalo email: rkunchak@buffalo.edu*

Sachin George *dept. of CSE, Suny Buffalo email: sachinge@buffalo.edu*

Rick Dasgupta *dept. of CSE, Suny Buffalo email: somjeetd@buffalo.edu*

***Abstract*- Training accurate machine learning and deep learning algorithms in the presence of imbalanced and noisy labels is an important and demanding task. Handwritten digit recognition has been a problem in the field of pattern classification. Handwriting recognition is one of the compelling research ideas because each individual has their own style of writing. It depends on the capability of the model to determine what digit is written. In this project, we look into two machine learning models, two deep learning models. The main objective of this project is to propose a machine learning and deep learning algorithm that provides better accuracy. We make use of the MNIST dataset which is widely used for this recognition process and it has 70000 handwritten digits. This project performs the analysis of accuracies, precision, and comparison of performance measures of algorithms such as Logistic Regression Model, Support Vector Classification Model, Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss model, and Symmetric Cross-Entropy for Robust Learning with Noisy Labels Model.**

I. INTRODUCTION

Handwritten digit classification is an important problem in optical character recognition. The major difficulty of handwritten numeral recognition is the variation in size, stroke thickness, rotation and deformation of the numeral images because it is written by different users. But this recognition is required as it has useful applications like bank check processing, converting handwritten information to digital format and postal code recognition. In this project we create 6 combinations of balanced, imbalanced and noisy dataset in order to find and compare the accuracies of the different models on this dataset. Imbalanced dataset exists in many real-world domains and getting a higher accuracy for the provided dataset is a challenging task. This project aims to recognize handwritten digits using machine learning models and deep learning models. The MNIST dataset is widely used for the recognition process, and it has 70,000 handwritten digits. Each image in this dataset is represented by an array of size 28*28. The array has 784 pixels having values ranging from 0 to 255. If the pixel value is
'0' it indicates that the background is black and if it is '1' the background is white. We focus on data preprocessing, data classification and their accuracies obtained by different training models. The Deep learning model gives the better accuracy than the Machine learning models.

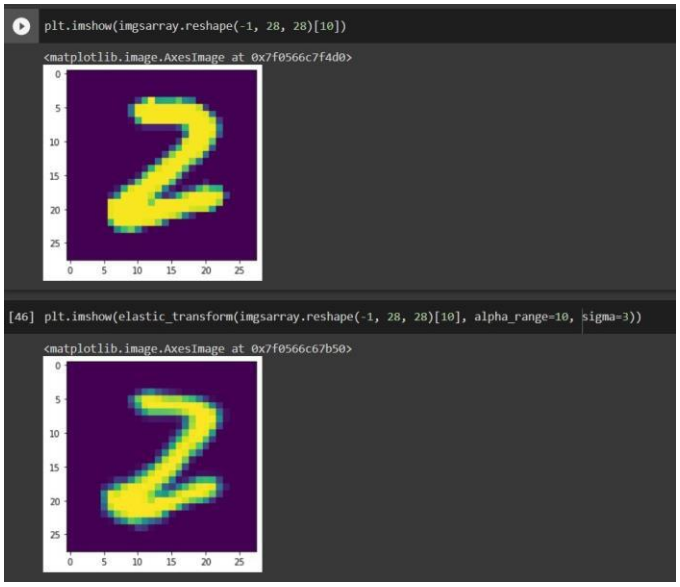
II. METHODOLOGY

A. Description of Dataset

The MNIST dataset is a database of 70,000 handwritten digits, divided into 60,000 training examples and 10,000 testing samples. The images in the MNIST dataset are present in form of an array consisting of 28*28 values representing an image along with their labels. The gray values of each pixel are coded in the [0,255] interval, using a o value for white pixels and 255 for black ones.

B. Data Preprocessing

We have created 6 types of datasets: balanced, imbalanced with symmetric, and asymmetric noise labels. To create different datasets we have used dataset distortion. The MNIST is pretty small so we wanted to avoid data loss because of which we have made use of data augmentation. It is richer in information than the affine transformations that keras image generators use. Elastic distortion is a method of data augmentation, as opposed to affine distortion which is the method Keras uses. Elastic distortion does a good job of mimicking variations in human hand writing. In elastic distortion, we create random displacement fields for height and width with values randomly sampled from unif(-1,1). A displacement field defines a direction and magnitude to move a pixel. Smooth the fields with a gaussian filter. Since $\mu=0$ for unif$(-1,1)$, most values will be close to 0 after the gaussian filter is applied. Thus most of the changes made by the fields will be small. Multiply the fields by a scaling factor to control intensity of the deformations. Then use interpolation to apply the displacement fields to the image.



III. IMPLEMENTATION

A. Logistic Regression

Logistic regression is called after the logistic characteristic, a characteristic used on the coronary heart of the method. The logistic characteristic, additionally referred to as the sigmoid characteristic, became evolved with the aid of using statisticians to explain the traits of populace growth, fast growth, and maximizing the sporting ability of surroundings in ecology. It is a sigmoidal curve that can take any actual range and convert it to a fee among zero and 1, however it isn't precisely inside those limits.

1 / (1 + e^-value)

Where e is the base of natural logarithms and value is the actual numerical value that you want to transform.

Logistic regression uses equations as expressions very similar to linear regression. Input values (x) are linearly combined using weights or coefficient values to predict the output value (y). The main difference from linear regression is that the simulated output values are binary values (0 or 1) rather than numeric values.

Logistic regression models the probability of the base class. Logistic regression is a linear method, but predictions are transformed using a logistic function. The effect of this is that the prediction can no longer be understood as a linear combination of inputs, as in linear regression. The coefficients of the logistic regression algorithm should be estimated based on the training data. This is done using the maximum likelihood estimate. The maximum likelihood score is a common learning algorithm used by various machine learning algorithms, but it makes assumptions about data distribution (more on this when we talk about data preparation).

The best coefficient will be a model that predicts values very close to 1 for the base class and very close to 0 for other classes. The intuition about the maximum likelihood of logistic regression is that the search routine determines the probability of the data.

B. SVM

A Support Vector Machine (SVM) is a supervised machine learning model that uses a classification algorithm to classify problems into two groups. After you provide the SVM with a labeled training dataset for each category, you can classify the new text. Compared to modern algorithms like neural networks, it has two main advantages: That is, faster speed and better performance with a limited number of samples (thousands). Therefore, the algorithm is well suited for text classification problems if you have access to data sets with typically a few thousand or fewer tagged samples.

The goal of SVM is to attract a line that separates the 2 instructions of fact factors. SVM generates a line that may cleanly separate the 2 instructions. How clean, you can ask. There are many feasible approaches of drawing a line that separates the 2 instructions, however, in SVM, it's far decided through the margins and the assist vectors.

The margin is the location setting apart the 2 dotted inexperienced traces as proven withinside the photograph above. The greater the margin the higher the instructions are separated. The assist vectors are the factors through which every of the inexperienced traces passes through. These factors are referred to as assist vectors as they make a contribution to the margins and as a result the classifier itself. These assist vectors are virtually the factors mendacity closest to the border of both of the instructions which have an opportunity of being in both one.

The SVM then generates a hyperplane that has the most margin, in this situation the black formidable line that separates the 2 instructions that's at a premier distance among each the instructions. In case of greater than 2 functions and a couple of dimensions, the road is changed through a hyperplane that separates multidimensional spaces.

C. LDAM

Label Distribution Aware Margin (LDAM) is the loss incurred by minimizing margin-based generalization. This loss overrides the standard cross-entropy target during training and can be applied to previous class imbalance training strategies such as resampling or resampling. We propose a simple yet effective training schedule that defers reweighting until the initial phase is complete, allowing the model to learn the initial representation while avoiding the complexity of reweighting or resampling.

Inspired by the trade-off between class fields for two classes, I propose to apply class-specific margins for multiple classes. Design the soft margin loss function so that the network has a higher margin. The most natural choice is a multi-class extension of hinge loss. Moderate mitigation of hinge loss is the next loss of cross-entropy with a forced margin. More recent studies have shown that under separable assumptions, regularization is weak, or without regularization logistic loss results in a maximum margin solution, which has been shown to be unaffected. Reweighting by definition. This further suggests that LLDAM loss and rebalancing may complement each other, as seen in the experiments.

D. SL

Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial intelligence. It is defined by training algorithms that use labeled datasets to classify data and accurately predict results. When input data is entered into the model, its weights are adjusted until the model is adjusted accordingly. This is done as part of the cross-validation process. Supervised learning helps businesses solve a variety of real-world problems on a large scale, including B. Classify spam in a folder separate from your inbox.

Supervised learning uses a training set to teach the model to achieve the desired output. This training dataset contains inputs and correct outputs that allow the model to train over time. The algorithm measures its accuracy via a loss function and adapts until the error is sufficiently minimized.

Supervised learning can be divided into two types of problems in data mining (classification and regression):
Classification uses algorithms to assign test data to specific categories. It recognizes specific entities in the dataset and tries to draw some conclusions on how to label and define those entities. Common classification algorithms are linear classifiers, support vector machines (SVMs), decision trees, nearest neighbors, and random forests. These are described in detail below.
Regression is used to understand the relationship between dependent and independent variables. It is widely used to make predictions such as: B. Sales of a particular company. Linear regression, logistic regression, and polynomial regression are common regression algorithms.
Supervised learning models are a valuable solution for avoiding manual classification tasks and making future predictions based on labeled data. However, formatting machine learning algorithms requires human knowledge and expertise to avoid overfitting the data model.

E. Proposed ML

We are improving the performance of traditional linear SVM as part of our proposed ML algorithm improvement. For that, we are taking two approaches. Namely, hyperparameter tuning combined with a non-linear RBF kernel. We have used the grid-search method to find and tune values hyperparameters for the non-linear kernel SVM for this dataset. We have focused on choosing hyperparameters based on maximizing the accuracy metric. We used all this on the non-linear Radial Basis Function (RBF) kernel for the SVM. We also used k-fold cross-validation to check which settings yield the most consistent results. Then we recorded the best hyperparameters and ran them finally on the entire training data, which yielded significantly better results than the linear SVM.

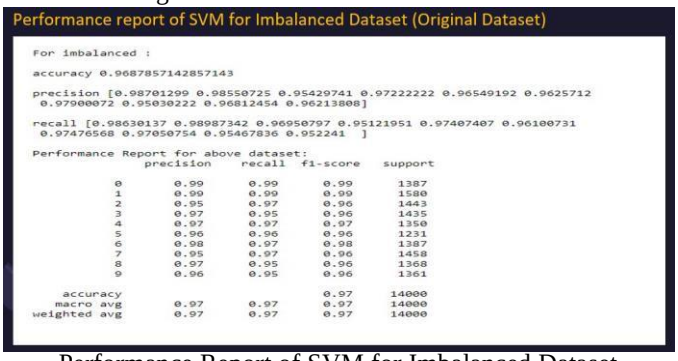
F. Proposed DL

The algorithm uses loss function as a way of directing the classifier to not overfit on easy-to-identify classes while simultaneously learning enough to classify the hard classes well. To address under learning of hard classes issue, they proposed the Symmetric cross-entropy Learning (SL), boosting CE symmetrically with the noise-robust Reverse Cross-Entropy (RCE), to simultaneously addresses it's under learning and overfitting problems. SL is a promising loss function for training robust DNNs against noisy labels.
The only bottleneck when it comes to applying this model over a wider variety of domains is the challenge to find good value for its hyperparameters alpha and beta, which decide the extent to which reverse cross-entropy loss is added in the total loss. We propose a general method to achieve that, that is learning the best set of hyperparameters through learning for the parameters over each epoch of learning.
Roughly, the sequence of our hyperparameter tuning algorithm will be:
1. Given a neural network structure N, hyperparameter set {S}, Strictness parameter Lambda, and Evaluation metric E.
2. Another value given is something we call "Revision factor" R, which decides how many times the values in S are updated per epoch.
3. Train N for 1 epoch (out of at least 20 epochs)
4. Measure the performance of N on E, calculate the gradient of the performance indicated by E.
5. Update the S with the gradient found (gradient descent) multiplied by the revision factor R.
6. IF gradient is less than half of the previous one, replace R with R/2. R ensures we update the H quickly within a few epochs.
7. Repeat from 3-6 until the gradient is smaller than Lambda. Our method if implemented makes the model using SL more generalizable over a wider variety of datasets.

IV. Experiments

A. Performance Metrics

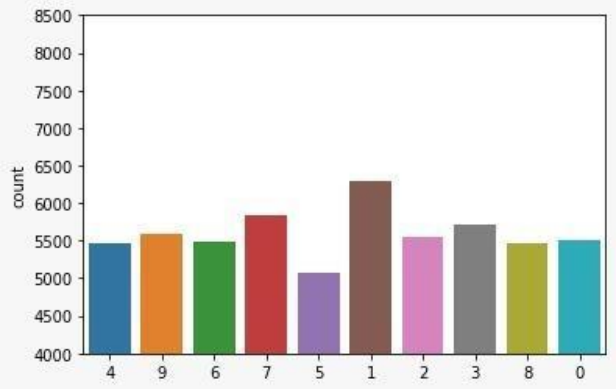
We used various performance metrics for our different models and datasets.
Accuracy – (TP + TN)/(TP+TN+FP+FN)
Accuracy is the number of true predictions from the total number of predictions in the model
Precision – (TP)/(TP+FP)
Precision is the total number of true predictions from our predicted true predictions
Recall – (TP)/(TP+FN)
Recall is the total number of true predictions from the actual true predictions.
F1 Score – 2. Precision. recall/(precision+recall)
The measure of the test accuracy
Confusion Matrix – Distribution of TP, FP, TN, FN
TP – True Positive
FP – False Positive
TN – True Negative
FN – False Negative



Performance Report of SVM for Imbalanced Dataset

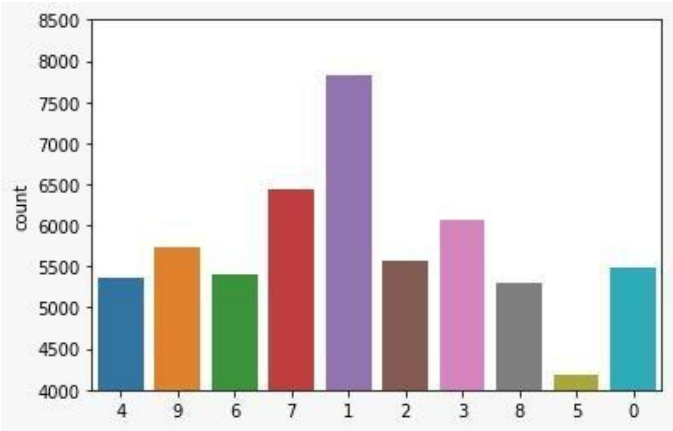
B. Results

Dataset Generation:



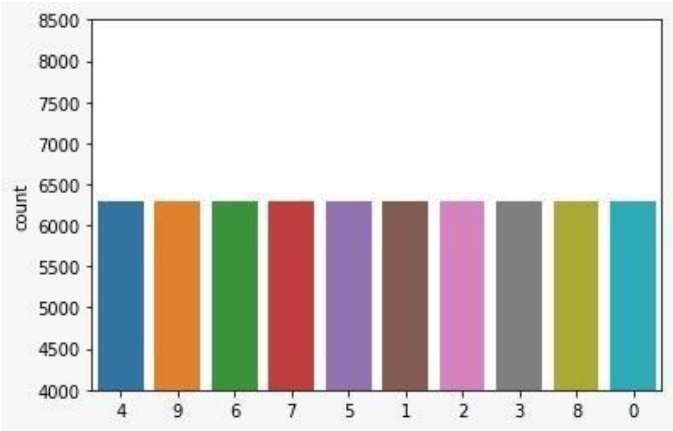
Original MNIST Dataset

As explained above we used Elastic Distortion and Data Augmentation to imbalance and balance the original MNIST Dataset. The counts of the original MNIST were very close to each other and hence we needed to amplify the variation in the counts. We not only wanted to maintain the degree of variation between classes but also increase them. Below are the counts of the Imbalanced Dataset.



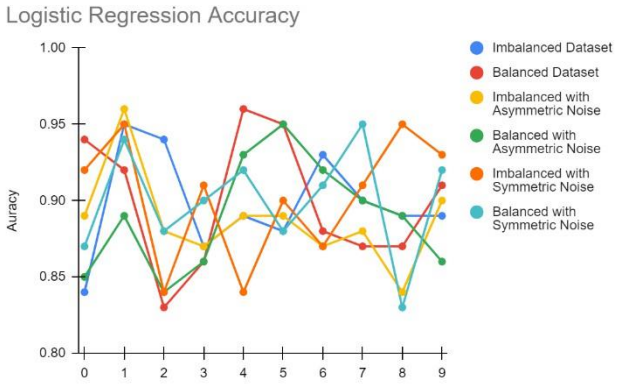
Imbalanced MNIST Dataset

After that, we converted the imbalanced dataset to a balanced dataset by using the same methods



Balanced MNIST Dataset

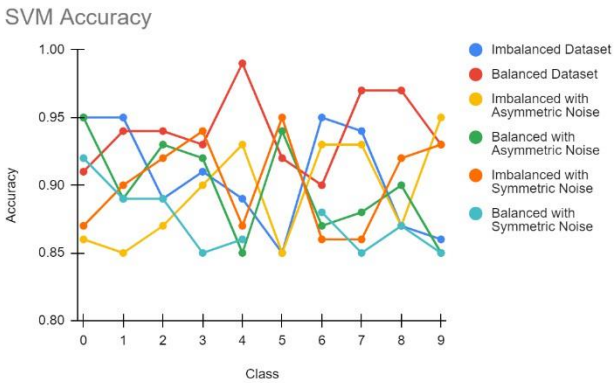
Logistic Regression:



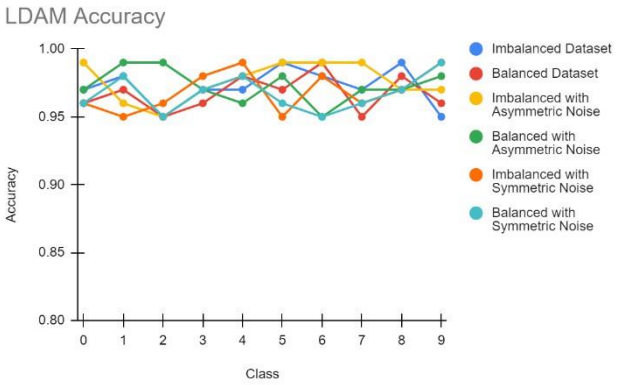
- The above graph represents the accuracy results of distinct datasets in LR for different sets of classes.
- The accuracy results of the datasets for each class range between 0.83 to 0.96.
- The common observation from the above graph is that the variation between the accuracy of each dataset for each class is quite noticeable as it is unsystematic.

SVM:

- The above graph represents the accuracy results of distinct datasets in SVM for different sets of classes.
- The accuracy results of the datasets for each class range between 0.83 to 0.97.
- The general observation from the graph above is that the deviations between the accuracy of each dataset
  are not systematic and are noticeable in each class.

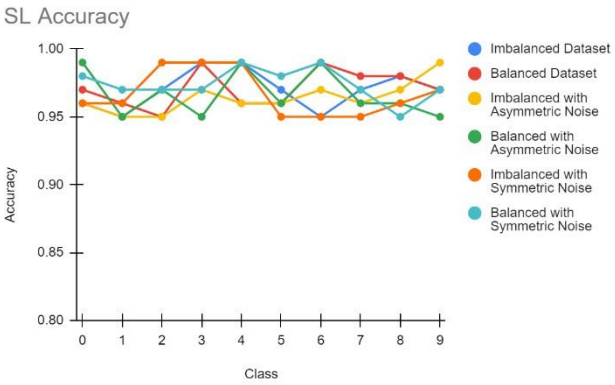


LDAM:



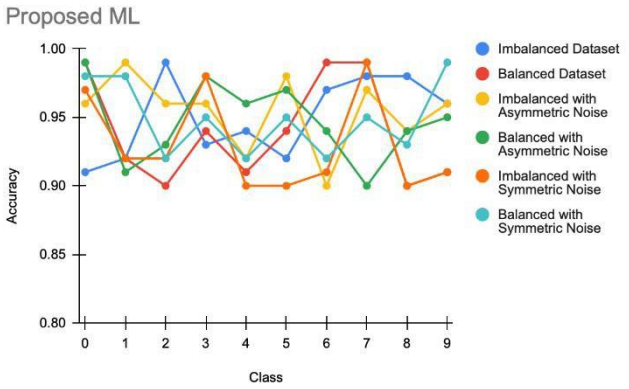
- The above graph represents the accuracy results of distinct datasets in LDAM for different sets of classes.
- The accuracy results of the datasets for each class range between 0.95 to 0.99.
- In comparison to LR & SVM, the accuracy level of the datasets has increased in LDAM due to which the variance amongst the results has reduced, the graph is more clustered now.

SL:



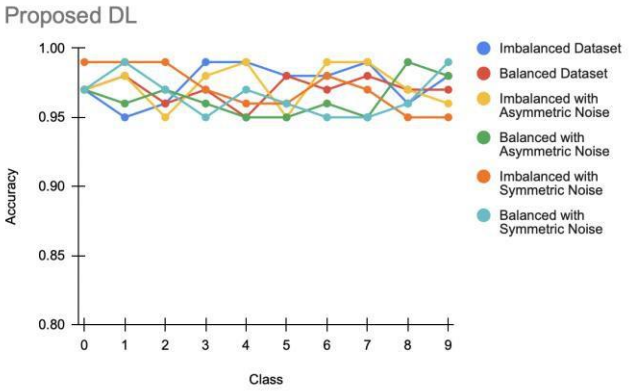
- The above graph represents the accuracy results of distinct datasets in SL for different sets of classes.
- The accuracy results of the datasets for each class range between 0.95 to 0.99.
- Compared to LR and SVM, the graph is more clustered due to the increased level of dataset precision in SL so there is decreased variance between results.

Proposed ML:



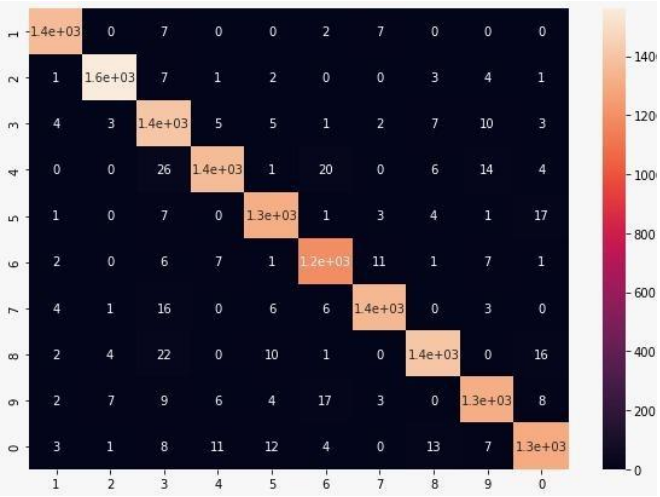
- The above graph represents the accuracy results of distinct datasets in Proposed ML for different sets of classes.
- The accuracy results of the datasets for each class range between 0.90 to 0.99.
- A common observation in the graph above is that the variance between the accuracies of each data set is not systematic and stands out for each class.

Proposed DL:

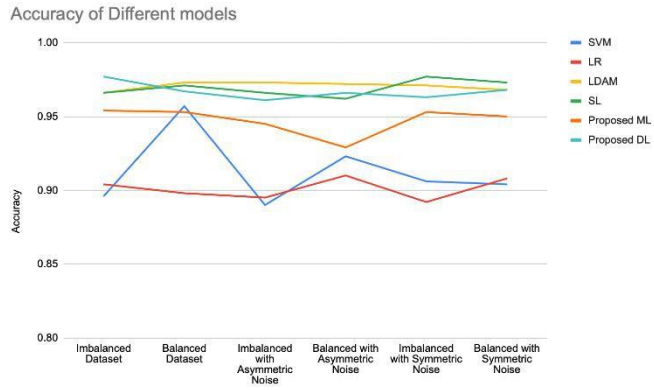


- The above graph represents the accuracy results of distinct datasets in Proposed DL for different sets of classes.
- The accuracy results of the datasets for each class range between 0.95 to 0.99.
- The graph that we get is more congregated as the precision for each class is improved which successively reduces the variation in the accuracy between the classes.

Comparison Among Models:



Above is the confusion matrix for the SVM model. We can see that results are good for the Kernel SVM.



- The above graph shows the accuracy of all the Models for all the datasets.
- The accuracy for LR shows minimum variance for all the datasets.
- For SVM, the accuracy rises for the balanced dataset and there is a sudden fall in accuracy with the imbalanced asymmetric noise dataset.
- For LDAM, the accuracy slightly increases for the balanced dataset and remains constant for the other datasets.
- For SL, the imbalanced dataset has good accuracy and it falls slightly for the imbalanced asymmetric noise dataset.
- The accuracy for Proposed ML falls suddenly for the balanced asymmetric noise dataset and increases for the imbalanced symmetric noise dataset.
- The accuracy for Proposed DL shows minimum variance for all the datasets.

C. OUTPUT TABLE FOR ML & DL

Dataset Type / Models	Balanced	Imbalanced (Original MNIST)	Balanced & Symmetric Noise	Balanced & Asymmetric Noise	Imbalanced & Symmetric Noise	Imbalanced & Asymmetric Noise
ML-1 (SVM)	94.9	91.11	91.3	88.7	90.8	88.9
ML-2 (LR)	90	89.3	89.6	88.3	89.4	90.1
LDAM-DRW	96.6	97.2	96.7	98	96.6	96.5
SL	96.9	96.9	97.9	97.1	96	96.4
Proposed ML(SVM)	94.4	94.3	94.9	95.4	92.7	94.5
Proposed DL	97.5	97.3	96.4	96.6	97.2	97.6

IV. CONCLUSION

As we have seen in the paper, we have performed various comparisons on the different models on variations of MNIST datasets. We applied various techniques to generate Balanced and Imbalanced datasets. We also added symmetric and asymmetric noise to the data. This was done to observe the performance of the models on the different datasets. We observed that the deep learning models perform better than the machine learning models. Although, by tuning the hyperparameters we can achieve high accuracy even with the machine learning models.

CONTRIBUTION

- Raksha Vishwanath – LDAM, LR Code Implementation, Graphs, SL Code implementation, Documentation
- Rick Dasgupta – SVM, Proposed ML, Proposed DL, Graphs
- Sachin George – LDAM, SL Code implementation, Conclusion, Evaluation metrics, Graphs
- Rohan Kunchakuri – LDAM, LR Code Implementation, Graphs, Documentation – Abstract, Introduction, Methodology, Experiments

REFERENCES

We have mentioned the references used in the code and the report below for our paper.

[1] Code 1. https://github.com/kaidic/LDAM-DRW.

*[2]*https://github.com/YisenWang/symmetric_cross_entropy_for_noisy_labels

[3] A method for applying elastic distortion to the MNIST data set is described by Simard, Steinkraus, and Plattin (2003) in "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis".

[4] https://www.irjet.net/archives/V6/i11/IRJET-V6I11328.pdf