Analyzing and experimenting on novel Oracle-Mnist dataset

Krishna Gandhi
ID : 1170559
M.Sc.Computer Science
Lakehead University
Thunder Bay, ON, CA

Md Ariful Islam
ID : 1165169
M.Sc.Computer Science
Lakehead University
Thunder Bay, ON, CA

Niharika Sojitra
ID : 1170232
M.Sc.Computer Science
Lakehead University
Thunder Bay, ON, CA

***Abstract*—In recent years, most prominent technology to emerge was deep learning. Convolutional Neural Networks (CNN or ConvNet) are the default deep learning model for image classification issues in computer vision and pattern recognition. One of the most recent and complicated benchmark dataset for image classification is Oracle-MNIST. It has unique noises and distortion in images which makes it challenging real world task to undertake. In this study, we analyzed Oracle-MNIST dataset with the help of data visualization, machine learning and deep learning techniques. We also propose a modified CNN architecture, which we explore further by cross validation to find optimal parameters. While machine learning models could achieve only 65% accuracy in classifying the images, CNN achieved 94.97% accuracy exceeding the previous 93.80% accuracy achieved by the author of the Oracle-MNIST dataset.**

***Index Terms*—Oracle-MNIST, PCA, CNN, Hyperparameter Tuning**

I. Introduction

In the modern world, numerous machine learning and deep learning algorithms have produced outcomes in computer vision and pattern recognition that were far beyond what was anticipated. To further facilitated task of image classification, we have many benchmark datasets that have been created till date.

One such dataset is MNIST, introduced in 1998 by LeCun et.al. [1]. It is a widely used collection of grayscale images of handwritten digits from 0 to 9, which are normalized to fit into 28x28 pixel bounding box and used for training various image processing systems. It has 60000 training images and 10000 testing images. This dataset is also historical due to achieving near human level accuracy in classifying all 10 classes. Convolution Neural Networks have become the one most sought-after algorithm for image classification since they have above 99.5 percent accuracy for the MNIST classification.

After succeeding at this, people went on to more challenging tasks in an effort to produce comparable outcomes for noisy, low-resolution photos. Since it was first released, MNIST has undergone changes, have many different version of it, like Fashion-MNIST [2], Oracle-MNIST [3].

Oracle-MNIST is a novel version of an MNIST dataset with similar features, including a 10-class classification system with 27,222 training and 3000 testing samples of grayscale images with dimensions of 28x28. But Oracle-MNIST is composed of scanned photographs of ancient Chinese characters, Oracle - one of oldest hieroglyphs from China, rather than handwritten ones. The Shang Dynasty (about 1600–1046 B.C.) is depicted in these characters, which are inscribed on tortoise shells and animal bones. This history includes details about divination methods, military campaigns, hunting expeditions, medical procedures, and childbirth. As the centuries went by, the legibility and clarity of the oracle characters began to deteriorate since they were written on shells and bones. Out of the 4500 characters uncovered, only 2200 have been decoded so far. Scanned images of these characters are noisy, blurred, and contain significant amount

of variation since different authors have varied writing styles. The authors have not performed any image enhancement techniques on Oracle-MNIST due to reduction in performance of classification algorithms. CNN outperforms better than all other classification algorithms used by the authors, with an accuracy rate of 93.8% [2].

However, it is argued that Oracle-MNIST constitutes the more challenging classification task than MNIST because the images of ancient characters suffer from extremely serious and unique noises caused by three-thousand years of burial and aging and dramatically variant writing styles by ancient Chinese, which all make them realistic for machine learning research.

In this study, we have applied both machine learning and deep learning (CNN) techniques on the novel Oracle-MNIST dataset and compare the classification performance with the the original MNIST dataset where significantly higher classification accuracy has been achieved by many different studies. Along with this, we want to explore the impact of Cross Validation on various Machine Learning and Deep Learning algorithms, to enhance the past findings. We also employed hyperparameter tuning technique, GridSearchCV with deep neural nets on Oracle-MNIST.

II. Literature Review

The Oracle-MNIST dataset is novel dataset and we were not able to find any study conducted specifically on this dataset except [3] where the authors introduced the dataset itself. However, there were numerous studies conducted on similar datasets like MNIST original dataset [4], Fashion-MNIST dataset [2], EMNIST dataset [5], we reviewed studies on them as part of our literature review.

LeCun et al. [4] included linear classifiers (whose error rate ranges from 7.6 to 12%), K-nearest neighbors approaches (K-NN, ranging from 1.1 to 5%), non-linear classifiers (about 3.5%), support vector machines (SVM, from 0.8 to 1.4%), neural networks (NN, from 1.6 to 4.7%) and convolutional neural networks (CNN, from 0.7 to 1.7%). Belongie et al. [8] achieved 0.63% and Keysers et al. [9] achieved 0.54% and 0.52% using K-NN, Kégl and Busa-Fekete [10] achieved 0.87% using boosted stumps on Haar features, LeCun et al. [4] achieved 0.8% and Decoste and Schölkopf [11] attained results from 0.56 to 0.68% using SVM.

When using non-convolutional neural networks, Simard et al. [12] achieved 0.9% and 0.7% respectively using a 2-layer neural network with MSE and cross-entropy loss functions. Deng and Yu [13] achieved 0.83% using a deep convex network without data augmentation or preprocessing. Very interesting results were attained by Meier et al. [14] (0.39%) using a committee of 25 neural networks and Cireşan et al. [15] (0.35%) using a 6-layers neural network (it is worth mentioning that the reproducibility of this result has been put into question by Martin [16] in his blog).

On the other hand, works based on convolutional neural networks attained a much better average performance (in fact, the worst result reported in LeCun's ranking was 1.7%). LeCun et al. [4] combined different convolutional architectures along with data augmentation techniques, obtaining error rates ranging from 0.7 to 0.95%. Lauer et al. [17] attained between 0.54% and 0.83% using a trainable feature extractor along with SVMs, and Labusch et al. [18] reported an error rate of 0.59% using a similar technique consisting on a CNN to extract sparse features and SVMs for classification. Simard et al. [12] obtained error rates between 0.4% and 0.6% using CNN with cross-entropy loss function and data augmentation. Ranzato et al. [19] reported results between 0.39% and 0.6% using a large CNN along with unsupervised pretraining, and some years later Jarrett et al. [20] reported a test error of 0.59% with a similar approach technique and without data augmentation. The best results in this ranking are those obtained by Cireşan et al. [21], which reported an error rate of 0.35% using a large CNN, and 0.27% [22] and 0.23% [23] using committees of 7 and 35 neural networks respectively, using data augmentation in all cases. Wan et al. [24] used CNNs with a generalization of dropout they called DropConnect, and reported an error rate of 0.57% without data augmentation and as low as 0.21% with data augmentation. Zeiler and Fergus [25] proposed the use of stochastic pooling achiev-

ing an error rate of 0.47%. Goodfellow et al. [26] described the maxout model averaging technique, attaining a test error rate of 0.45% without data augmentation. In 2015, Lee et al. [27] described "deeply supervised nets", an approach by which they introduce a classifier (SVM or softmax) at hidden layers, reporting a result of 0.39% without data augmentation.

III. Methodology

A. Dataset Description

Oracle-MNIST - novel, realistic and challenging dataset, comprising of real-world images of ancient Chinese characters is established for bench-marking pattern recognition classification, with particular challenges on image noise and distortion [3]. It contains 30,222 images of oracle characters belonging to 10 classes - all in 28x28 grayscale. Fig. 1 shows the images present in dataset.

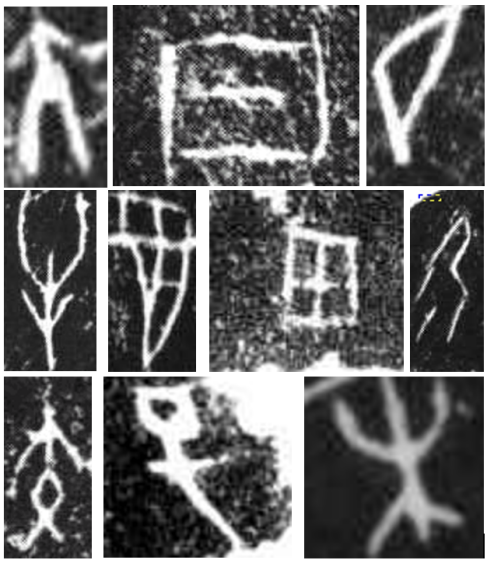


Fig. 1: Examples of images in Dataset

The training set has 27,222 images and test set contains 300 images per class(3000). This split was provided by base paper. Hence, to make the comparison fair we decided to keep the same split. [1]

TABLE I: Dataset distribution for Oracle-MNIST.

Dataset	Oracle-MNIST
Training	27,222
Testing	3000

Fig. 2 presents the class distribution in the training dataset.The dataset is nearly evenly distributed among the 10 classes.

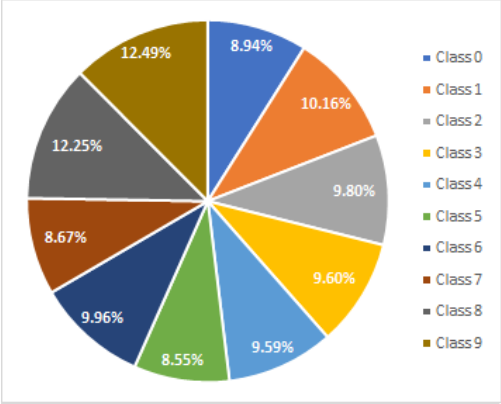


Fig. 2: Class Distribution in Training Dataset

B. Dimensionality Reduction and Data Visualization

In terms of the parameters of the images and classes, Oracle MNIST is strikingly similar to MNIST and Fashion-MNIST. As a result, even though it is a novel dataset that hasn't been explored before, we can visualise it in the same way as we can for the other two. There are various methods that can help in visualizing image data, one of the most basic method is Principle Component Analysis (PCA) which is described further below.

Principal component analysis (PCA): Principal component analysis (PCA) is a technique that transforms high-dimensions data into lower-dimensions while retaining as much information as possible. It is a technique used to emphasize variation and bring

[1]The dataset is freely available on https://github.com/wm-bupt/oracle-mnist

out strong patterns in a dataset. PCA is extremely useful when working with data sets having a large number of features. Common applications such as image processing, genome research etc. always have to deal with thousands, if not tens of thousands of columns. [41]

In our experiment, we have applied Principal Component Analysis (PCA) for both visualization and training of models. For visualization, we have applied PCA to to our dataset to reduce to 2 and 3 dimensions for plotting. We considered 95% variance to obtain the principal components for model training.

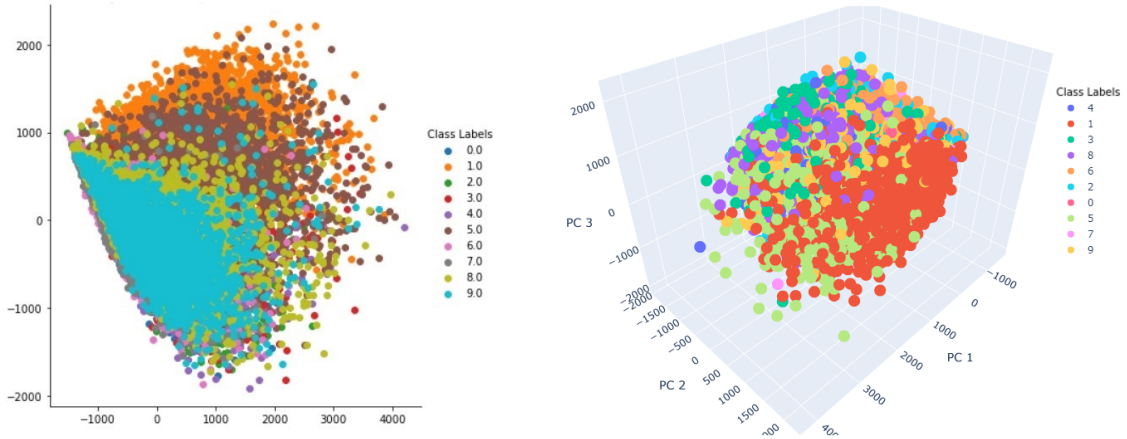
C. *Experimentation models*

1) *Machine Learning Models:*

- K-Nearest Neighbours (KNN): The k-nearest neighbors (KNN) algorithm is a data classification method for estimating the likelihood that a data point will become a member of one group or another based on what group the data points nearest to it belong to. It is a type of supervised machine learning algorithm used to solve classification and regression problems. However, it's mainly used for classification problems. KNN is a lazy learning and non-parametric algorithm. It's called a lazy learning algorithm or lazy learner because it doesn't perform any training when you supply the training data. Instead, it just stores the data during the training time and doesn't perform any calculations. It doesn't build a model until a query is performed on the dataset. KNN is heavily used in identifying patterns, such as in text and digit classification, it is particularly helpful in identifying handwritten numbers. [42]
- Logistic Regression (LR) : Logistic regression is a simple and more efficient and extensively employed machine learning algorithm for binary and linear classification problems and achieves very good performance with linearly separable classes. It is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on but it can be generalized to multi-class classification. Multinomial logistic regression can model scenarios where there are more than two possible discrete outcomes. Logistic regression is a useful analysis method for classification problems, where the model is trying to determine if a new sample fits best into a category. [42]
- Random Forest (RF) : Random forest is a supervised learning algorithm which consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. It can be used both for classification and regression. It is an extension of the bagging method as it utilizes both bagging and feature randomness to create an uncorrelated forest of decision trees - this is a key difference between decision trees and random forests. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting. It also provides a pretty good indicator of the feature importance. Random forest algorithms have a variety of applications, such as recommendation engines, image classification and feature selection. Random forest algorithms have three main hyperparameters, which need to be set before training. These include node size, the number of trees, and the number of features sampled. [42]

2) *Deep Learning Model:*

- Convolutional Neural Network: Convolutional Neural Network (CNN) is a class of Artificial Neural Networks (ANN) which is a current state-of-the-art model commonly used in computer vision problems such as image classification. CNN mainly consist of three types of layers: 1. Convolutional layers 2. Pooling layers and 3. Fully connected layers. Figure 4 describes the Basic Feed-forward ANN architecture.



(a) 2-D (b) 3-D

Fig. 3: 2-D and 3-D visualization of the dataset with PCA

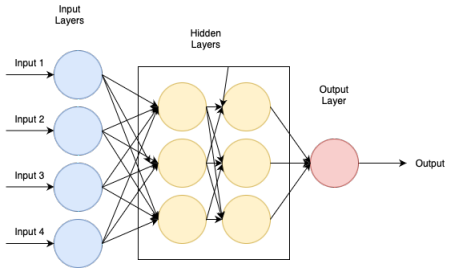


Fig. 4: Basic ANN architecture

It includes all of the inputs, a number of hidden layers, and one output layer that categorises the provided sample. CNN adheres to the same architecture but differs in that it was created primarily to address pattern recognition tasks in images. CNN focuses more on decreasing characteristics and encoding image-specific features that are suitable for creating an image-focused network.

CNN's biggest flaw is that it has trouble handling the computational complexity needed to encode image data. Most ANNs work well with common ML benchmarking datasets like MNIST, which have a small dimensionality [29].

The usual structure of a CNN is a stack of unique layers that convert the input image into the class scores. Convolution layer, pooling layer, fully connected layer, and classification layer are four different types of layers that are frequently utilised. The convolution and pooling layers are typically repeated in multiple pairs, followed by a fully linked layer and a classification layer. The foundational component of the CNN is the convolution layer. With a limited receptive field for the input image, each neuron in the convolution layer computes the output by convolution of the receptive field with a linear filter [31], [35].

There are number of parameters of CNN one can affect the performance, by tuning it according to the use of various tasks and data we can improve the results [39]. First is optimizer, there are many CNN optimizers present nowadays for classification with deep neural net. some of them are Gradient Descent(GD), Stochastic GD, mini-batch GD, Adagrad, AdaDelta, Adam. One of the most commonly used one is SGD. It is a variation of Gradient descent, but instead of updating model parameter once in cycle for whole dataset, it updates for each sample. It takes less

memory and is considerably faster [37].

The weight activation function of any Machine Learning algorithm is one of its most noticeable components. A variety of activation functions are available to aid machine learning in hidden nodes. The Sigmoid function is the most popular activation function employed by artificial neural networks. Here Equation (1) and Equation (2) defines sigmoid function and tangent hyperbolic respectively [30].

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{1}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2}$$

However, there is a significant issue with employing common activation functions like sigmoid or tangent hyperbolic since they are contractive and their gradients frequently reach zero when they are used with large values. It is referred to as the vanishing gradient problem and causes SGD updates to be extremely small. In order to combat this, Nair et. al created the rectified linear units (ReLU) activation function in place of the standard function [38]. The ReLU activation function's gradients at positive values become constant and no longer disappear. This suggests that employing the ReLU activation function can prevent the vanishing gradient problem. This explains why ReLU activation function can accelerate deep neural network learning. The ReLU is now employed as the default activation function for deep neural nets. Equation (3) depicts ReLU [31].

$$Relu(z) = max(0, z) \tag{3}$$

Weight initialization is another parameter that aids in accelerating learning. In the learning process of any activation function, the starting values of the weights are crucial [3]. Learning will be slow if the weights are large or small because they will saturate the gradients. So having initial weights that are optimal for any dataset can help it learn faster [33]. There are numerous built-in weight initializers available right now, including he_normal, He_unifrom, zeros, ones, Glorot_uniform, Glorot_normal, orthogonal, identity, LeCun_uniform, constant, random_normal [34].

Learning rate is learning speed given to each weight. When the cost function is strongly non-spherical, the parameter μ helps to increase learning rate by reducing the size of steps with high curvature, which results in a higher learning rate in the direction of low curvature [].

$$\Delta w(t+1) = \eta \frac{\partial E_{t+1}}{\partial w} + \mu \Delta w(t) \tag{4}$$

Equation (4) describes the learning function for weight, in which η is learning rate while μ stands for momentum.

In this paper, we describe how above mentioned parameters are used on Oracle-MNIST dataset to optimize and increase performance of model.

- Cross Validation for Hyper-parameter Tuning using GridSearchCV:
  1) Weight Initialization: The performance of a neural net highly depends on how its parameters are initialized when it is starting to train. Weights are initialized in a number of different ways while starting the training of neural nets. Sometimes, we initialize it with constant values like 0's and 1's and sometimes with values sampled from some distribution [36].
  2) Kernel Size: Kernel size is a filter used in CNN. The size of kernel is very sensitive, if we choose smaller kernel size then we can expect more details however, it can lead us to overfitting and also increases the computational power. If we choose the bigger size then we tend to overfit. Hence, it becomes optimal to tune this parameter.
  3) Activation Function: An activation function decides whether a neuron should be activated or not. It is added to the output of the neural network.
  4) Batch Size: Mini-batch size in a range of 16 to 128 is usually preferable in the

learning of ConvNets since ConvNet is sensitive to batch size.

5) Optimizers: An optimization algorithms modify each epoch's weights and find the value of the parameters that minimizes the loss function while training the deep learning model. As a result, it helps in reducing the overall loss and improve the accuracy. Thus, it is very important to choose right weight for the model [40].

There were two parts in this experiment for this study: (1) Exploratory Data Analysis and visualization, and (2) Models training. For data visualization, PCA was used and for models training, three Machine Learning models (KNN, LR and RF) and one Deep Learning model (CNN) was used. Only, training accuracy, training loss and test accuracy were considered in this study.

IV. Experiments

All experiments in this study were conducted on Kaggle kernel using GPU P100 and machine learning libraries such as Keras and SKlearn.

A. Exploratory Data Analysis and Visualization

From our Exploratory Data Analysis as depicted in Fig. 2, the dataset is nearly evenly distributed and hence we didn't apply any class imbalance handling mechanism on the training data.

B. Dimensionality Reduction

In the MNIST-Oracle dataset, there are 784 dimensions (features). We have applied PCA to reduce the dimensionality for training our models. We have chosen 95% variance as the input to PCA and the number of features were reduced to 171 from original 784 features as shown in Fig 5. We trained our machine learning models with these 171 features.

Fig. 6 displays the top 25 principal components and the variance explained by each of them. We can see the first 12 principal components can explain 60% variance while 50 PCs can explain 81% variance of the data.

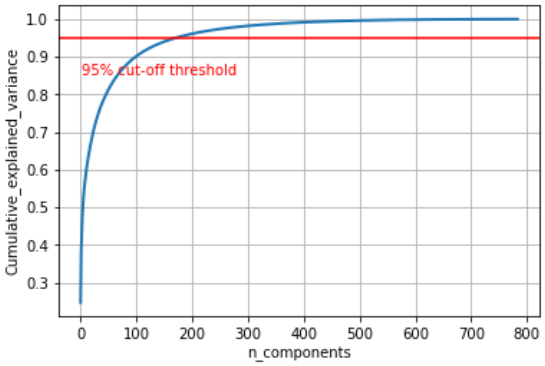


Fig. 5: Data Variance explained by Principal Components

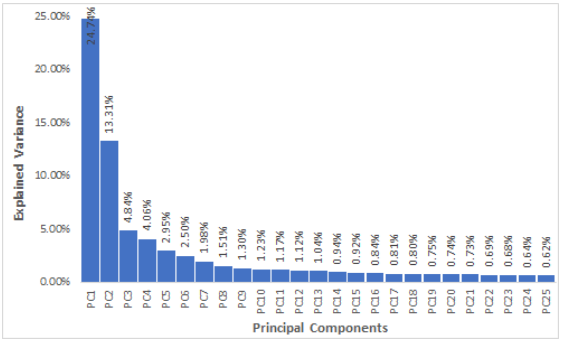


Fig. 6: Data Variance explained by Top 25 Principal Components

C. Machine Learning Algorithms

We used the same parameters of the base paper in order to retain the comparability with the base paper. Along with this, we also experimented with additional parameters. We executed all of the three machine learning algorithms with following 4 scenarios:

Scenario-1: Train the models with all 784 features of training set and test with the test set.

Scenario-2: Train the models with all 784 features of training set with 5-Fold Cross validation and test with the test set.

Scenario-3: Apply PCA and train the models with reduced features of training set and test with the test set.

Scenario-4: Apply PCA and train the models with reduced 171 features of training set with 5-Fold cross validation and test with the test set.

Test dataset was transformed separately to have the same dimensions as training set. Min-Max scalar was used for scaling the training and test data separately.

Table II, III and IV below presents the parameters used in Logistic Regression, K-Nearest Neighbor and Random Forest methods respectively.

TABLE II: Hyper-parameters used for Logistic Regression

Hyper-Parameter	Values
C(Inverse Regularization)	0, 0.001, 0.01, 0.1,1.0, 10, 100
Penalty	L2, None
Multi-Class	OVR, Multinomial, Auto
Solver	lbfgs

TABLE III: Hyper-parameters used for K-Nearest Neighbors

Hyper-Parameter	Values
Weights	uniform, distance
N-Neighbors	9, 5, 1
P(Distance Type)	1, 2

TABLE IV: Hyper-parameters used for Random Forest

Hyper-Parameter	Values
Number of Estimators	100, 50, 10
Max Depth	100, 50, 10
Criterion	Gini, Entropy

We have used following configuration for cross-validation and PCA across all the methods.

TABLE V: Parameters used for Cross Validation and PCA

Parameter	Values
Cross Validation	5-Fold
PCA	95% Variance

D. Proposed CNN

The base CNN architecture had 2xConv-Pool-ReLu, 2xFC, Dropout. Since, from the data visualization, we analyzed that separating this data is very complicated [29]. So, we aimed to add few more Conv-layers to extract as much information it could. Therefore, we added two more Conv-layers and kept rest all parameters same as base CNN. In this paper, we proposed our CNN model with 4xConv-Pool-ReLu, 2xFC, Dropout. The below Fig. 7 represents the proposed CNN architecture.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 28, 28)        1664
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 14, 14)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 14, 14)       204928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 128, 7, 7)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 256, 7, 7)         819456
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 256, 4, 4)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 512, 4, 4)         3277312
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 512, 2, 2)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              2098176
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                10250
=================================================================
Total params: 6,411,786
Trainable params: 6,411,786
Non-trainable params: 0
```

Fig. 7: Model Architecture

The hyper-parameters listed in Table 3 were taken from base paper [29]. After several trials, we observed increase in performance(accuracy). Hence, to find the optimal parameters we performed cross-validation.

TABLE VI: Hyper-parameters used for proposed CNN before Cross Validation

Hyper-parameters	Oracle-MNIST
Weight Initializer	Glorot_Uniform
Kernel Size	5
Activation Function	ReLu
Batch Size	64
Optimizer	SGD
Epochs	15
Learning Rate	0.1

E. Cross-validation using GridSearchCV

For this experiment, the following parameters listed in Table VII were taken into consideration to perform Grid Search. The hyper-parameter search is conducted using scikit-learn's GridsearchCV function. The model training and validation are implemented using 3-fold cross-validation.

TABLE VII: List of hyper-parameters for Grid Search

Hyper-parameter	List
Weight_Initializer	['uniform', 'lecun_uniform', 'normal', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform']
Kernel Size	[3, 4, 5]
Activation Function	['LeakyReLu', 'ReLu']
Batch Size	[64, 128]
Optimizer	['SGD', 'Adam']

This cross-validation experiment took total time of approx. 4 hours. The table VIII illustrates the list of best parameters found by cross validation.

TABLE VIII: List of best hyper-parameters after cross validation

Hyper-parameter	List
Weight_Initializer	Lecun_Uniform
Kernel Size	3
Activation Function	ReLu
Batch Size	64
Optimizer	SGD

V. Results

A. Machine Learning Algorithms

1) Logistic Regression: "Fig 8", "Fig 9", "Fig 10" presents snapshots of our experimental results using Logistic Regression, K-Nearest Neighbors and Random Forest algorithm. Detailed results with model parameters are summarized in Table. XI in the "Appendix" section.

While experimenting with Logistic Regression, We tried with "one-vs rest", multi-nomial and automatic multi-class classification methods from scikit learn with different levels of C, the Inverse of regularization strength values ranging from 0 to 100. We achieved the best results when C was equal to 100. we have used 'l2' as the Penalty for all cases.

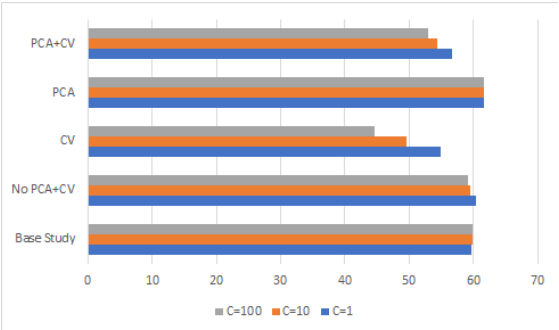


Fig. 8: Experimental Results with Logistic Regression

The base study accuracy was 59.7% without CV and PCA. They have used C=100 and One-vs-Rest multi-class classification. With the same settings, our accuracy was slightly less 58.73% but when we applied PCA, the accuracy was increased by nearly 3% and became 61.6% with reduced 171 features.

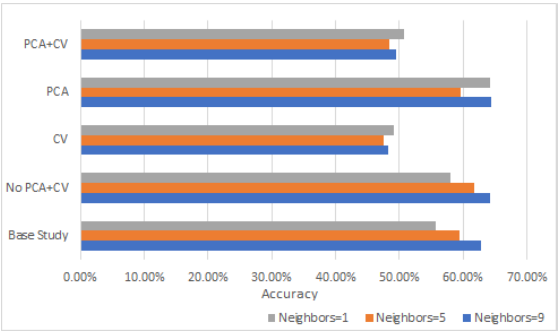


Fig. 9: Experimental Results with K-Nearest Neighbor

In our experiment with KNN, we tried with 1, 5, and 9 neighbors with Manhattan and Euclidean distances. The base study achieved the highest accuracy of 62.7% with 9-neighbors and p=2(Euclidean) distance. With the same settings, our study achieved 64.2% accuracy with all 784 features. When we applied PCA and reduced the dimensions to 171, the accuracy was slightly improved to 64.3%.

We also have applied PCA with and without cross validation and the accuracy had been decreased significantly to 49.53% and 48.27% respectively.

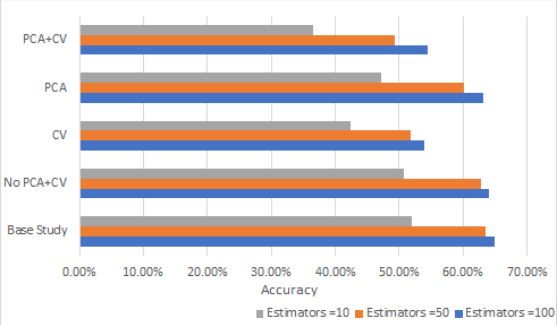


Fig. 10: Experimental Results with Random Forest

In our experiment with Random Forest, we used 100, 50 and 10 tree estimators with both Gini impurity and Entropy for information gain as a measure of the quality of the split. We tried with max-depth of 100, 50 and 10 as well. The base study achieved highest accuracy of 65% with 100 estimators and Gini index along with max-depth=100. With the same settings, our model accuracy was 1% less, 64.07%. When we applied Cross Validation with and without PCA, the accuracy was slightly decreased to 63.1% unlike Logistic Regression and K-nearest Neighbor models. This is due to, By applying PCA before learning decision tree, you will provide only a new space with the maximum variance regarding your original space. Since PCA is not a discriminative method, results can not produce a superior performance.

With the machine learning algorithms, the highest accuracy of 64.70% was achieved with K-Nearest Neighbor algorithm when PCA was applied without cross validation.

B. CNN

After finding the best parameter from cross validation, we applied this parameters on proposed architecture. The results of our experiments are shown below in Table IX.

TABLE IX: Result Comparision between Base and Proposed CNN

Algorithm	Parameters	Test Accuracy
Base CNN	2xConv-Pool-ReLu, 2xFC, Dropout	93.80
Proposed CNN	4xConv-Pool-ReLu, 2xFC, Dropout	**94.97**

VI. Discussion and Conclusion

As an objective of the current study, we analyzed and experimented with a novel dataset called Oracle-MNIST. It is a benchmark dataset with similar technical characteristics to MNIST, except instead of handwritten digits, it consists of scanned images of ancient Chinese characters. In addition, Mei Wang and Weihong Deng examined a variety of machine learning and deep learning algorithms, and they discovered that CNN classifies this dataset the best, whereas Machine Learning techniques fall short. However, the performance of CNN has not yet reached saturation and there was still room for improvement with an error rate of 6.2% .

Therefore, to improve their result, first we performed data analysis and visualization to understand the data distribution in high dimensions. When applying PCA to further analyze the dataset, it was found that The data is exceedingly dense and non-separable.

Further testing revealed that CNN easily outperformed traditional machine learning algorithms, which struggled to learn from the dense data. When applying two more convolution layers, there was a slight but not appreciable improvement because it was discovered through literature that the denseness of the CNN can improve the result for more intricate datasets. As a consequence, we made the decision to use cross validation to determine the optimal parameters that would enable us to enhance the present findings, which are presented in Table VIII and have resulted in more than 1% improvement at 94.97% accuracy.

Limitations and Future Work

As was previously indicated, Oracle-MNIST contains extremely dense and non-separable data, so while there has been some improvement in the results, it is still well short of what we had hoped. There is room for improvement in terms of results because the original MNIST achieved more than 99.5% accuracy for classification using a CNN with precision close to that of a human. In the future, one can employ superior clustering techniques to separate data and produce significantly better outcomes

for both machine learning and deep learning algorithms. Additionally, they can enhance results by using the committee of CNNs and other benchmark models along with transfer learning. To further examine the data, one can also use data augmentation and feature improvement techniques.

ACKNOWLEDGMENT

We would like to express our deepest appreciation to authors of the Oracle-MNIST Dataset for publishing their work, without that we could not have achieved this research. Additionally, we would like to extend our sincere thanks to our course instructor, Dr. Thiago for his valuable guidance in this project. Also we could not have achieved this work without each other. Contribution for each member is mentioned in the Table X below.

TABLE X: Team Contribution

TASK	Krishna	Md. Ariful	Niharika
Reasearch and Topic Selection	✓	✓	✓
Literature Review	✓	✓	✓
Data visualization - PCA	✓	✓	✓
Experimentation on LR and RF		✓	
Experimentation on KNN			✓
Experimentation on CNN	✓		
Results and Discussion	✓	✓	✓
Project Presentation	✓	✓	✓
Report Writing	✓	✓	✓

REFERENCES

[1] Yann LeCun, Corinna Cortes, and Christopher JC Burges. 2010. MNIST hand- written digit database. AT&T Labs [Online]. Available: **http://yann.lecun.com/exdb/mnist/** (2010)

[2] Xiao, H., Rasul, K. & Vollgraf, R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. (2017,8,28)

[3] Wang, M. & Deng, W. Oracle-MNIST: a Realistic Image Dataset for Benchmarking Machine Learning Algorithms. *ArXiv Preprint ArXiv:2205.09442*. pp. 1-7 (2022)

[4] LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings Of The IEEE*. **86**, 2278-2324 (1998)

[5] Cohen, G., Afshar, S., Tapson, J. & Schaik, A. EMNIST: Extending MNIST to handwritten letters. *2017 International Joint Conference On Neural Networks (IJCNN)*. pp. 2921-2926 (2017)

[6] Baldominos, A., Saez, Y. & Isasi, P. A survey of handwritten character recognition with mnist and emnist. *Applied Sciences*. **9**, 3169 (2019)

[7] Kussul, E. & Baidyk, T. Improved method of handwritten digit recognition tested on MNIST database. *Image And Vision Computing*. **22**, 971-981 (2004)

[8] Belongie, S., Malik, J. & Puzicha, J. Shape matching and object recognition using shape contexts. *IEEE Transactions On Pattern Analysis And Machine Intelligence*. **24**, 509-522 (2002)

[9] Keysers, D., Deselaers, T., Gollan, C. & Ney, H. Deformation Models for Image Recognition. *IEEE Transactions On Pattern Analysis And Machine Intelligence*. **29**, 1422-1435 (2007)

[10] Kégl, B. & Busa-Fekete, R. Boosting Products of Base Classifiers. (Association for Computing Machinery,2009), https://doi.org/10.1145/1553374.1553439

[11] Decoste, D. & Schölkopf, B. Training invariant support vector machines. *Machine Learning*. **46**, 161-190 (2002)

[12] Simard, P., Steinkraus, D. & Platt, J. Best practices for convolutional neural networks applied to visual document analysis. *Seventh International Conference On Document Analysis And Recognition, 2003. Proceedings.*. pp. 958-963 (2003)

[13] Deng, L. & Yu, D. Deep Convex Network: A Scalable Architecture for Speech Pattern Classification. *Interspeech*. (2011,8), urlhttps://www.microsoft.com/en-us/research/publication/deep-convex-network-a-scalable-architecture-for-speech-pattern-classification/

[14] Meier, U., Ciresan, D., Gambardella, L. & Schmidhuber, J. Better Digit Recognition with a Committee of Simple Neural Nets. *2011 International Conference On Document Analysis And Recognition*. pp. 1250-1254 (2011)

[15] Cireşan, D., Meier, U., Gambardella, L. & Schmidhuber, J. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. *Neural Computation*. **22**, 3207-3220 (2010,12), 'https://doi.org/10.1162/NECO

[16] Martin, C. Tensorflow reproductions: Big deep simple MNIST. *Calculated—Content*. (2016,6), https://calculatedcontent.com/2016/06/08/tensorflow-reproductions-big-deep-simple-mnist/

[17] Lauer, F., Suen, C. & Bloch, G. A trainable feature extractor for handwritten digit recognition. *Pattern Recognition*. **40**, 1816-1824 (2007), https://www.sciencedirect.com/science/article/pii/S0031320306004250/

[18] Labusch, K., Barth, E. & Martinetz, T. Simple Method for High-Performance Digit Recognition Based on Sparse Coding. *IEEE Transactions On Neural Networks*. **19**, 1985-1989 (2008)

[19] Ranzato, M., Poultney, C., Chopra, S. & Cun, Y. Efficient Learning of Sparse Representations with an Energy-Based Model. *Advances In Neural Information Processing Systems*. **19** (2006), https://proceedings.neurips.cc/paper/2006/file/87f4d79e36d68c3031ccf6c55e9bbd39-Paper.pdf

[20] Jarrett, K., Kavukcuoglu, K., Ranzato, M. & LeCun, Y. What is the best multi-stage architecture for object recognition?. *2009 IEEE 12th International Conference On Computer Vision*. pp. 2146-2153 (2009)

[21] Cireşan, D., Meier, U., Masci, J., Gambardella, L. & Schmidhuber, J. Flexible, High Performance Convolutional Neural Networks for Image Classification. *Proceedings Of The Twenty-Second International Joint Conference On Artificial Intelligence - Volume Volume Two*. pp. 1237-1242 (2011)

[22] Ciresan, D., Meier, U., Gambardella, L. & Schmidhuber, J. Convolutional Neural Network Committees for Handwritten Character Classification. *2011 International Conference On Document Analysis And Recognition*. pp. 1135-1139 (2011)

[23] Ciresan, D., Meier, U. & Schmidhuber, J. Multi-column deep neural networks for image classification. *2012 IEEE Conference On Computer Vision And Pattern Recognition*. pp. 3642-3649 (2012)

[24] Wan, L., Zeiler, M., Zhang, S., Le Cun, Y. & Fergus, R. Regularization of neural networks using dropconnect. *International Conference On Machine Learning*. pp. 1058-1066 (2013)

[25] Zeiler, M. & Fergus, R. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. (arXiv,2013), https://arxiv.org/abs/1301.3557

[26] Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A. & Bengio, Y. Maxout networks. *International Conference On Machine Learning*. pp. 1319-1327 (2013)

[27] Lee, C., Xie, S., Gallagher, P., Zhang, Z. & Tu, Z. Deeply-supervised nets. *Artificial Intelligence And Statistics*. pp. 562-570 (2015)

[28] Kussul, E. & Baidyk, T. Improved method of handwritten digit recognition tested on MNIST database. *Image And Vision Computing*. **22**, 971-981 (2004), https://www.sciencedirect.com/science/article /pii/S0262885604000721, Proceedings from the 15th International Conference on Vision Interface

[29] O'Shea, K. & Nash, R. An Introduction to Convolutional Neural Networks. *CoRR*. **abs/1511.08458** (2015), http://arxiv.org/abs/1511.08458

[30] W. Hao, W. Yizhou, L. Yaqin and S. Zhili, "The Role of Activation Function in CNN," 2020 2nd International Conference on Information Technology and Computer Application (ITCA), 2020, pp. 429-432, doi: 10.1109/ITCA52113.2020.00096.

[31] Ide, H. & Kurita, T. Improvement of learning for CNN with ReLU activation by sparse regularization. *2017 International Joint Conference On Neural Networks (IJCNN)*. pp. 2684-2691 (2017)

[32] Dewa, C. & Afiahayati Suitable CNN Weight Initialization and Activation Function for Javanese Vowels Classification. *Procedia Computer Science*. **144** pp. 124-132 (2018), https://www.sciencedirect.com/science/article/pii/ S187705091832221X, INNS Conference on Big Data and Deep Learning.

[33] Katanforoosh & Kunin, "Initializing neural networks", deeplearning.ai, 2019.

[34] "Usage of initializers," [Online]. Available: https://faroit.com/keras-docs/2.0.6/initializers/.

[35] Albawi, S., Mohammed, T. & Al-Zawi, S. Understanding of a convolutional neural network. *2017 International Conference On Engineering And Technology (ICET)*. pp. 1-6 (2017)

[36] S. Maheshkar, " Neural Network Initialization With Keras," 7 6 2022. [Online]. Available: https://wandb.ai/sauravm/Regularization-LSTM/reports/Neural-Network-Initialization-With-Keras–VmlldzoyMTI5NDYx.

[37] S. Doshi, "Various Optimization Algorithms For Training Neural Network," 13 1 2019. [Online]. Available: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6.

[38] Nair, V. & Hinton, G. Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair. *Proceedings Of ICML*. 27 pp. 807-814 (2010,6)

[39] Breuel, T. The Effects of Hyperparameters on SGD Training of Neural Networks. (2015,8)

[40] A. Gupta, "A Comprehensive Guide on Deep Learning Optimizers," 7 10 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/.

[41] Scikit Learn Documentation. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html/

[42] IBM Cloud Education, IBM Cloud Learn Hub, 07-Dec-2020. [Online]. Available: https://www.ibm.com/cloud/learn/. [Accessed: 08-Dec-2022]."

[43] IBM Cloud Education, "What is Logistic Regression", IBM Cloud Learn Hub. [Online]. Available: https://www.ibm.com/topics/logistic-regression/

[44] IBM Cloud Education, "Random Forest," IBM Cloud Learn Hub, 07-Dec-2020. [Online]. Available: https://www.ibm.com/cloud/learn/random-forest/

APPENDIX

TABLE XI: Experimental Results with Machine Learning Algorithms

2*Algorithm	2*Parameters	Accuracy (Base Study)	Accuracy(This Study)			
		No PCA+CV	No PCA+CV	CV	PCA	PCA+CV
24***Logistic Regression**	C =1, Penalty = none, Multi Class= ovr	N/A	58.73	38.87	61.60	52.13
	C =0, Penalty = l2, Multi Class= ovr	N/A	40.43	35.97	40.43	35.97
	C =0.001, Penalty = l2, Multi Class= ovr	N/A	56.20	44.17	56.23	44.27
	C =0.01, Penalty = l2, Multi Class= ovr	N/A	61.60	56.20	61.40	56.37
	C =0.1, Penalty = l2, Multi Class= ovr	N/A	61.30	58.17	61.67	58.20
	C =1, Penalty = l2, Multi Class= ovr	59.7	60.43	54.90	61.60	56.60
	C =10, Penalty = l2, Multi Class= ovr	59.8	59.47			
	C =100, Penalty = l2, Multi Class= ovr	59.8	59.13			
	C =1, Penalty = none, Multi Class= multinomial	N/A	58.40	45.27	61.63	49.60
	C =0, Penalty = l2, Multi Class= multinomial	N/A	42.80	36.03	42.80	36.10
	C =0.001, Penalty = l2, Multi Class= multinomial	N/A	57.80	45.27	57.73	45.23
	C =0.01, Penalty = l2, Multi Class= multinomial	N/A	61.53	56.87	61.17	57.20
	C =0.1, Penalty = l2, Multi Class= multinomial	N/A	61.50	58.70	61.80	58.73
	C =1, Penalty = l2, Multi Class= multinomial	N/A	60.17	55.00	61.63	56.07
	C =10, Penalty = l2, Multi Class= multinomial		59.33			
	C =100, Penalty = l2, Multi Class= multinomial		58.67			
	C =1, Penalty = none, Multi Class= auto	N/A	58.40	45.27	61.63	49.60
	C =0, Penalty = l2, Multi Class= auto	N/A	42.80	36.03	42.80	36.10
	C =0.001, Penalty = l2, Multi Class= auto	N/A	57.80	45.27	57.73	45.23
	C =0.01, Penalty = l2, Multi Class= auto	N/A	61.53	56.87	61.17	57.20
	C =0.1, Penalty = l2, Multi Class= auto	N/A	61.50	58.70	61.80	58.73
	C =1, Penalty = l2, Multi Class= auto	N/A	60.17	55.00	61.63	56.07
	C =10, Penalty = l2, Multi Class= auto		59.33			
	C =100, Penalty = l2, Multi Class= auto	N/A	58.67			
12***K-Nearest Neighbor**	Neighbors=9, P=1, Weight=distance	61.8	63.17	49.3	64.73	50.63
	Neighbors=9, P=2, Weight=distance	62.7	64.2	48.27	64.3	49.53
	Neighbors=9, P=1, Weight=uniform	61.6	62.73	45.8	59.7	46.2
	Neighbors=9, P=2, Weight=uniform	61.5	63.3	49.77	64.83	51.07
	Neighbors=5, P=1, Weight=distance	60.3	62	49.53	63.7	50.83
	Neighbors=5, P=2, Weight=distance	59.5	61.73	47.47	59.6	48.43
	Neighbors=5, P=1, Weight=uniform	59.6	61.27	47.63	64.4	49.8
	Neighbors=5, P=2, Weight=uniform	59	61.47	47.43	63.03	48
	Neighbors=1, P=1, Weight=distance	55.8	56.93	45.8	59.7	46.2
	Neighbors=1, P=2, Weight=distance	55.7	57.97	49.07	64.2	50.7
	Neighbors=1, P=1, Weight=uniform	55.8	56.93	48.03	62.6	50.1
	Neighbors=1, P=2, Weight=uniform	55.7	57.97	47.47	59.6	48.43
18***Random Forest**	Estimators =100, Max depth =100, Criterion= gini	65.00	64.07	53.87	63.10	54.50
	Estimators =50, Max depth =100, Criterion= gini	63.60	62.83	51.90	60.10	49.33
	Estimators =10, Max depth =100, Criterion= gini	52.00	50.77	42.43	47.20	36.60
	Estimators =100, Max depth =50, Criterion= gini	64.90	64.80	54.20	64.10	54.27
	Estimators =50, Max depth =50, Criterion= gini	62.90	62.23	51.90	60.17	49.30
	Estimators =10, Max depth =50, Criterion= gini	51.30	51.80	42.03	45.53	38.37
	Estimators =100, Max depth =10, Criterion= gini	58.30	58.80	52.63	58.37	54.80
	Estimators =50, Max depth =10, Criterion= gini	57.60	57.43	51.93	57.97	52.07
	Estimators =10, Max depth =10, Criterion= gini	53.20	53.57	42.13	51.90	40.20
	Estimators =100, Max depth =100, Criterion= entropy	64.90	64.53	54.60	62.80	54.30
	Estimators =50, Max depth =100, Criterion= entropy	62.50	63.50	51.93	59.77	49.63
	Estimators =10, Max depth =100, Criterion= entropy	52.10	52.93	43.23	47.20	35.87
	Estimators =100, Max depth =50, Criterion= entropy	65.00	64.83	53.87	63.63	54.27
	Estimators =50, Max depth =50, Criterion= entropy	62.90	62.10	51.87	59.60	49.00
	Estimators =10, Max depth =50, Criterion= entropy	51.80	50.87	41.73	48.27	36.83
	Estimators =100, Max depth =10, Criterion= entropy	58.30	58.07	54.47	58.63	53.87
	Estimators =50, Max depth =10, Criterion= entropy	58.00	58.03	51.77	58.07	50.97
	Estimators =10, Max depth =10, Criterion= entropy	52.80	53.03	43.60	51.13	36.23