**MNIST-DATASET**

**A PROJECT REPORT**

***Submitted By***

**Varanasi Hemasai Reddy (21MIM10056)**

**Shrankhala Srivasthav (21MIM10038)**

**Sheryl Thussu (21MIM10019)**

**Ankit Kumar Singh (21MIM10048)**

**Lavnish Jain(21MIM10065)**

*in partial fulfillment for the award of the degree*

*of*

**INTEGRATED MASTER OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**SPECIALIZATION IN ARTIFICIAL INTELLIGENCE AND**

**MACHINE LEARNING**

****

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**VIT BHOPAL UNIVERSITY**

**KOTHRIKALAN, SEHORE**

**MADHYA PRADESH – 466114**

JAN 2023

**VIT BHOPAL UNIVERSITY, KOTHRIKALAN, SEHORE**

**MADHYA PRADESH – 466114**

**BONAFIDE CERTIFICATE**

Certified that this project report titled **“Cryptocurrency Prediction”** is the bonafide work of **“Shrankhala Srivasthav (21MIM10038), Varanasi Hemasai Reddy(21MIM10056), Sheryl Thussu(21MIM10019), Ankit Kumar(21MIM10077), Lavanish Jain(21MIM00)”** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

**PROGRAM CHAIR PROJECT GUIDE**

Dr. A.V.R Mayuri Dr. G. Prabhukanna

School of Computer Science and Engineering School of Computer Science and Engineering

Integrated MTech CSE (AI and ML) Assistant Professor

VIT BHOPAL UNIVERSITY VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on …......... 2023

**ACKNOWLEDGEMENT**

First and foremost, I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to **Dr. A.V.R Mayuri**, Head of the Department (CDS), School of Computer Science and Engineering for much of his valuable support and encouragement in carrying out this work.

I would like to thank my internal guide **Dr. G. Prabukanna** and **Dr. A.V.R Mayuri**, for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Computer Science and Engineering, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

**LIST OF ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Abbreviation** | **Definition** |
| **1** | LSTM | Long Short-Term Memory |
| **2** | RNN | [Recurrent neural network](https://en.wikipedia.org/wiki/Recurrent_neural_network) |
| **3** | RMSE | Root Mean Square Error |
| **4** | MAE | Mean Absolute Error |
| **5** | MAPE | Mean Absolute Percentage Error |
| **6** | NLANN | Nonlinear Autoregressive Model Neural Network |
| **7** | FTS | Fuzzy Time Series |

**ABSTRACT**

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The dataset includes 60,000 training images and 10,000 test images, where each image is a 28x28 pixel grayscale image of a handwritten digit. The digits have been normalized and centered in a fixed-size image. The goal of the dataset is to correctly classify each digit, with the labels ranging from 0 to 9.

The MNIST dataset is often used as a benchmark for image classification algorithms, and has been widely used for training and testing machine learning models, particularly for image classification using deep learning.

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | List of Abbreviations  Abstract | 4-5 |
| 1 | **CHAPTER-1:**  **PROJECT DESCRIPTION AND OUTLINE**   * 1. Introduction   2. Motivation for the work   3. Problem Statement   4. Literature Review   5. Proposed LSTM Model | 6-19 |
| 2 | **CHAPTER-2: INTRODUCTION TO LSTM MODEL**   * 1. Hardware and Software Requirements   2. Introduction to Dataset   3. Description of LSTM Model   4. Diagram of LSTM Model   5. Predicted Graph   6. Technical Code | 19-28 |
| 3 | **CHAPTER-3: PROGRAM OUTCOME AND**  **APPLICABILITY**   * 1. Code Output   2. Key Points About Code Implementation   3. Significant Project Outcomes | 29-30 |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 4 | **CHAPTER-4: CONCLUSION AND RECOMMENDATIONS**  4.1 Outline  4.2 Existing work with limitations  4.3 Future Enhancements  4.4 Inferences | 31-32 |
|  | References | 33-35 |

**Chapter 1**

**PROJECT DISCIPLINE AND OUTLINE**

**1.1: Introduction**

The MNIST dataset is a widely used collection of hand-written digit images that is commonly used in machine learning research. It consists of 60,000 training images and 10,000 testing images, with each image being a grayscale 28x28 pixel image of a hand-written digit from 0 to 9.

The dataset is often used as a benchmark for machine learning algorithms, especially in the field of image recognition and computer vision. The goal is typically to build a machine learning model that can accurately classify the digits in the images, based on their pixel values.

The MNIST dataset has been used in a wide range of applications, from developing computer vision algorithms for self-driving cars to building handwriting recognition software for digitizing handwritten documents. It has also been used as a benchmark for evaluating the performance of deep learning models.

Overall, the MNIST dataset is a useful resource for anyone interested in exploring the field of machine learning or working on a related project, especially in the area of image recognition.

The MNIST dataset was created by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges, who used it as a benchmark for evaluating the performance of their machine learning algorithms for image recognition. The dataset was originally published in 1998 and has since become a popular resource for researchers and practitioners in the field.

One reason why the MNIST dataset has been so widely used is that it is relatively small and easy to work with. The images are simple and easy to visualize, and the dataset can be downloaded and loaded into a variety of machine learning libraries and frameworks. This makes it an ideal dataset for testing and comparing different machine learning models and algorithms.

Another reason why the MNIST dataset is so useful is that it represents a real-world problem that many machine learning applications must address: the recognition of handwritten characters. By training machine learning models on the MNIST dataset, researchers can develop algorithms that can recognize and classify handwritten digits, which can then be applied to a wide range of other applications, from recognizing handwriting in legal documents to identifying numbers on license plates.

In recent years, more complex and challenging image recognition datasets have been introduced, such as CIFAR-10 and ImageNet. However, the MNIST dataset remains a popular choice for beginners and researchers who want to develop and test machine learning models for image recognition.

The term "gradient descent" refers to an optimization algorithm used to minimize the cost function in machine learning. The cost function represents the difference between the predicted output of a model and the actual output. The goal of gradient descent is to find the values of the model's parameters that minimize the cost function, which in turn results in a more accurate model.

In gradient descent, the model's parameters are updated iteratively by taking small steps in the direction of steepest descent of the cost function. The direction of steepest descent is given by the negative of the gradient of the cost function, which represents the rate of change of the cost function with respect to each of the model's parameters.

There are several variations of gradient descent, including batch gradient descent, stochastic gradient descent, and mini-batch gradient descent. Batch gradient descent updates the model's parameters using the gradient of the cost function evaluated over the entire training dataset, while stochastic gradient descent updates the parameters using the gradient evaluated over a single training example. Mini-batch gradient descent is a compromise between the two, where the gradient is evaluated over a small subset of the training data.

The choice of gradient descent algorithm and its parameters can have a significant impact on the performance and convergence of a machine learning model. In recent years, several extensions to the basic gradient descent algorithm have been proposed, such as momentum, adaptive learning rate, and second-order methods, which aim to improve the convergence speed and stability of the optimization process.

Overall, gradient descent is a fundamental optimization algorithm in machine learning, and understanding its principles and variations is essential for building accurate and efficient machine learning models.

Handwritten digits recognition is a challenging task in a computer vision field, that has led the path to many modern technologies. It has numerous practical applications: automated form reading, postal mail zip-code processing, bank check processing, etc. The complication of a handwritten digit’s recognition task is derived from the individual handwritten style that defines the size, slope, width and other typeface characteristics for the numbers. Although the human error rate for the digits recognition task is low, it is not an easy task for the machines as there could be some ambiguities (e.g., ‘3’ and ‘8’). However, it is crucial for the digit recognition methods to be ultimately accurate as they deal with the numbers. Modified National Institute of Standards and Technology (MNIST) is considered a benchmark dataset for the digit recognition systems. It was introduced in 1998 by LeCun’s et al. in [6] and is publicly available at LeCun’s homepage. MNIST contains 70,000 grey-scale images of a 28 × 28 size. The dataset comprises patterns from two sources: NIST’s Special Database-1 (high-school students’ writings) and NIST’s Special Database-3 (the United States Census Bureau employees’ writings). MNIST is split into train and test parts with 60,000 and 10,000 instances, respectively. The split was chosen in such a manner that the same writer would not be involved in both samples. The significant discrepancies in the typefaces of different writers could be noted based on a sample from MNIST dataset in Figure 1. Generally, the explicit test set, data availability and a comprehensible task have made MNIST applicable for the fast and easy comparison of different classification frameworks.

The recognition algorithms for MNIST have achieved a sufficiently high accuracy, which is commonly measured by the error rate. Solutions based on convolutional neural networks are considered state-of-the-art for MNIST and output the error rate less than 1%. Among the machine learning approaches SVM leads the list for the dataset without data augmentation or preprocessing. However, some issues exist concerning the reproducibility of the proposed solutions. The present paper proposes a new machine learning approach that allows to get a comparable accuracy without any data preprocessing. The key advantage of our algorithm consists in the low computational cost: training time for MNIST amounts to 5 seconds. In section 3 we describe in detail the developed framework. Section 4 compares our approach with the reproducible SVM solution. The scalability of the proposed algorithm is analyzed in section 5.

Gradient descent is a type of iterative optimization algorithm that updates the model's parameters in small increments. The size of these increments is determined by a learning rate hyperparameter, which controls how quickly the model's parameters are updated. If the learning rate is too small, the optimization process may take a long time to converge to the minimum of the cost function. If the learning rate is too large, the optimization process may oscillate and fail to converge.

**1.2: Motivation of Work**

Today in world everything is shifting towards Online, even education. Many students in this age of competition wish to learn more and more new things within the comfort of their home. It gets really difficult to know which online course is helpful in giving robustness to our resume. So, to tackle such complexity in selecting the course, we have come up with Course recommender System. It not only saves the valuable time of student’s life but also providing the popular and genuine courses based on their interests.

**1.3: PROBLEM STATEMENT**

The goal of this project is to build a machine learning model that can accurately classify hand-written digits from the MNIST dataset. The MNIST dataset consists of 70,000 images of hand-written digits from 0 to 9, with 60,000 images used for training and 10,000 images used for testing. The images are grayscale 28x28 pixels in size.

The project should begin with exploratory data analysis to gain insights into the characteristics of the dataset, such as the distribution of digit classes and the pixel intensities. The project should then proceed with data pre-processing steps, such as normalization and data augmentation, to prepare the dataset for machine learning algorithms.

The next step is to implement and train various machine learning algorithms, such as logistic regression, support vector machines, and neural networks, to classify the hand-written digits. The algorithms should be evaluated using appropriate performance metrics, such as accuracy, precision, and recall, and compared to determine which algorithm performs best on the MNIST dataset.

Finally, the best performing machine learning algorithm should be used to classify new hand-written digits from external sources, such as scanned documents or real-time handwriting inputs.

The success of the project will be judged based on the accuracy and generalization performance of the best performing machine learning algorithm, as well as the quality and thoroughness of the data pre-processing and model evaluation steps.

**1.4: LITERATURE SURVEY**

In the past few years, deep [neural networks](https://www.sciencedirect.com/topics/neuroscience/neural-networks) (DNNs) have made tremendous progress in various [machine learning](https://www.sciencedirect.com/topics/computer-science/machine-learning) tasks in [computer vision](https://www.sciencedirect.com/topics/engineering/computervision) ([Krizhevsky, Sutskever, & Hinton, 2017](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b16)), [natural language processing](https://www.sciencedirect.com/topics/engineering/natural-language-processing) ([Collobert & Weston, 2008](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b6)), and speech recognition ([Amodei et al., 2016](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b1)). However, DNNs mimic only the hierarchical [topological structure](https://www.sciencedirect.com/topics/computer-science/topological-structure) of the flow of information in the brain and process data in a real-valued form, but this is far removed from the information processing mechanisms of the brain. Spikes play a crucial role in efficient information processing, and spiking neural networks (SNNs) have been developed to emulate information processing in the brain. SNNs are considered to be third-generation neural networks ([Maass, 1997](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b19)). The discrete spike-driven communication between [spiking neurons](https://www.sciencedirect.com/topics/engineering/spiking-neuron), multiple brain-inspired learning rules, and intricate connections make these networks biologically plausible and energy efficient.

However, owing to the lack of an effective training algorithm, the development of SNNs has stalled for a while, and they have not yet demonstrated performance comparable to that of DNNs. Many researchers have taken inspiration from the process of learning by synapses in the brain to introduce such mechanisms as spike-timing-dependent plasticity ([Bi & Poo, 1998](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b3)) and short-term facilitation or depression to the learning of weights of the SNNs ([Diehl and Cook, 2015](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b8), [Falez et al., 2019](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b10), [Tavanaei and Maida, 2016](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b29), [Tavanaei and Maida, 2017](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b30), [Zhang, Zeng, Zhao and Shi, 2018](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b41), [Zhang, Zeng, Zhao and Xu, 2018](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b42)). Although such an approach is suitable for biological interpretation, most of these mechanisms are local learning rules, and it becomes increasingly difficult to achieve global convergence as the number of network layers increases. GLSNN ([Zhao, Zeng, Zhang, Shi, & Zhao, 2020](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b43)) introduces global feedback layers combined with local learning rules for optimization to alleviate this problem, but still performs poorly on complex datasets.

The success of deep learning comes from the [backpropagation algorithm](https://www.sciencedirect.com/topics/engineering/backpropagation-algorithm); however, the characteristics of discontinuity and [nondifferentiability](https://www.sciencedirect.com/topics/engineering/nondifferentiability) of SNNs have impeded the implementation of [backpropagation](https://www.sciencedirect.com/topics/engineering/backpropagation) based on [gradient descent](https://www.sciencedirect.com/topics/engineering/gradient-descent). It is not feasible to apply the backpropagation algorithm to directly train SNNs. An alternative is to convert well-trained DNNs into SNNs through additional adjustments of their parameters ([Diehl et al., 2015](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b9), [Hu et al., 2018](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b12), [Sengupta et al., 2019](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b26), [Xu et al., 2018](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b37)). Such methods of conversion have achieved accuracy comparable to the state of the art on large datasets such as ImageNet and complex architectures such as VGG and ResNet. However, their impressive performance is due to well-trained DNNs, and this does not solve the problem of adequately training SNNs. Moreover, their DNN-based training does not take full advantage of the temporal information in SNNs. When SNNs are deep, the time window should be large enough to enable the mean firing rates to approach the values obtained by DNNs. With the recent proposal of the approximation of the gradient of spiking threshold functions, the backpropagation algorithm can now be directly applied to train SNNs ([Jin et al., 2018](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b14), [Lee et al., 2016](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b18), [Wu et al., 2018](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b34), [Wu, Deng et al., 2019](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b35)). In this process, the discontinuous derivative of the spiking neurons is approximated by a continuous function. Although they perform well on some simple datasets, their performance on the whole still lags significantly behind that of traditional DNNs.

In addition to typical spiking neurons and feedforward [network structures](https://www.sciencedirect.com/topics/computer-science/network-structure), many complicated mechanisms support the brain’s learning and inference. We draw on some of these to further improve backpropagation-based SNNs. The typical network structure is based on a simple forward structure. When the spiking neuron’s membrane reaches a threshold, it releases a spike to the [postsynaptic neuron](https://www.sciencedirect.com/topics/engineering/postsynaptic-neuron). There are many other complex structures in the brain, such as [feedback connections](https://www.sciencedirect.com/topics/mathematics/feedback-connection) ([Felleman and Van Essen, 1991](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b11), [Sporns and Zwi, 2004](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b28)). Cross-layer feedback connections take the information predicted by the higher cortex to early [cortical areas](https://www.sciencedirect.com/topics/computer-science/cortical-area) to help with inference and learning. In particular, the autapse connected to the soma ([Ikeda and Bekkers, 2006](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b13), [Wang et al., 2017](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b32), [Yin et al., 2018](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b38)) applies time-delayed feedback on the neuron’s membrane potential to regulate the precision of the spike and network activity.

Excitatory spiking neurons are used in most current SNNs. Once the membrane potential of the [presynaptic neuron](https://www.sciencedirect.com/topics/engineering/presynaptic-neuron) reaches a certain threshold, it releases the spike to enhance the membrane potential and render the postsynaptic neurons easier to fire. However, there exist both excitatory and inhibitory neurons in the brain, and the dynamic balance between them is crucial to healthy cognition and behavior ([Dehghani et al., 2016](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b7), [Rubin et al., 2017](https://www.sciencedirect.com/science/article/pii/S0893608022002520" \l "b25)).

By taking inspiration from the two mechanisms described above, we introduce an adaptive time-delayed self-feedback mechanism (SFBM) to help regulate the membrane potential to improve backpropagation-based SNNs. We also introduce inhibitory neurons to backpropagation-based SNNs. The dynamic balance between the excitatory and inhibitory neurons accelerates the convergence of the neural networks and improves their performance. We use the combination of the two mechanisms to propose a deep SNN with adaptive self-feedback and balanced excitatory–inhibitory neurons (BackEISNN). The results of experiments on several commonly used datasets verified the performance of our BackEISNN. It achieved state-of-the-art performance on the MNIST, N-MNIST, and Fashion-MNIST datasets. Our contributions here can be summarized as follows:

* We introduce the adaptive SFBM to apply time-delayed feedback to the membrane potential of spiking neurons to regulate the precision of spikes.
* We introduce a balanced excitatory and inhibitory neuron mechanism (BEIM) to control the balanced firing of spikes, which enables the network to converge more quickly and thus perform better.
* We combined the SFBM and BEIM, and subjected this combination to extensive experiments on the MNIST, Fashion-MNIST, N-MNIST, and CIFAR10 datasets. The results indicate that the proposed mechanisms can significantly improve the performance of backpropagation-based SNNs and accelerate the training of SNNs. It achieved state-of-the-art performance on the MNIST, N-MNIST, and Fashion-MNIST datasets, and delivered impressive performance on the CIFAR10 dataset while using a relatively light structure [1].

Improving existing [neural network](https://www.sciencedirect.com/topics/neuroscience/neural-networks) architectures can involve several design choices such as manipulating the loss functions, employing a diverse learning strategy, exploiting gradient evolution at training time, optimizing the network hyper-parameters, or increasing the architecture depth. The latter approach is a straightforward solution, since it directly enhances the representation capabilities of a network; however, the increased depth generally incurs in the well-known vanishing gradient problem. In this paper, borrowing from different methods addressing this issue, we introduce an interlaced multi-task learning strategy, defined SIRe, to reduce the vanishing gradient in relation to the object [classification task](https://www.sciencedirect.com/topics/computer-science/classification-task). The presented methodology directly improves a convolutional [neural network](https://www.sciencedirect.com/topics/neuroscience/neural-networks) (CNN) by preserving information from the input image through interlaced auto-encoders (AEs), and further refines the base [network architecture](https://www.sciencedirect.com/topics/computer-science/network-architecture) by means of skip and residual connections. To validate the presented methodology, a simple CNN and various implementations of famous networks are extended via the SIRe strategy and extensively tested on five collections, i.e., MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, and Caltech-256; where the SIRe-extended architectures achieve significantly increased performances across all models and datasets, thus confirming the presented approach effectiveness.

A separate but non-negligible aspect of datasets containing a medium or a high number of classes is the time required to train a network. Having more classes, samples, or images with increased width and height (e.g., going from a 28 × 28 to a 224 × 224 shape when using the MNIST and ImageNet datasets, respectively) generally requires the design of more complex architectures to achieve higher performance, which naturally translates in a given model requiring a considerably longer training time before it reaches convergence on such a collection. Regardless, solutions to address these aspects either reduce the number of images simultaneously fed to the model, i.e., the batch size, at the expense of a longer time to complete a training epoch, or use higher-performing hardware with conspicuous amounts of RAM and processing speed, such as tensor processing units (TPUs), to limit eventual training time increases. As a consequence, many approaches are constantly being developed even on the more complex datasets such as ImageNet (Cheng et al., 2020, Wei et al., 2020, Zhao, Wu, et al., 2021) [2].

Classification, which involves classifying data or images into a predefined number of classes, is one of the most fundamental and challenging problems in machine learning (ML). Neural networks (NN), K-nearest neighbor (KNN), logistic regression (LR), and support vector machine (SVM) are standard ML-based classifiers methods commonly used to solve classification tasks. Nevertheless, a classifier inevitably requires a dimensionality-reduction model, such as principal component analysis (PCA), linear discriminant analysis (LDA), and double graph-based discriminant projections (DGDP) [9], to solve real-world problems arising from the complexity and high dimensionality of the data. Hence, traditional ML research typically combines a dimensionality-reduction model, such as PCA, LDA, and AE, with a classification model such as SVM, NN, and LR, respectively. Consequently, double errors tend to occur in both the dimensionality-reduction model and classifier when incorporating more than one independent model. Furthermore, a significant error in the output of a dimensionality-reduction model can lead to a misleading outcome if it is fed into the classifier. In this decade, the emergence of deep learning has made convolutional neural network (CNN) state-of-the-art classifiers. The most impressive breakthrough of CNN is its ability to automatically exploit feature extraction through its convolutional layers. Nevertheless, models with deeper or more complex architecture are more susceptible to overfitting. Thus, given their complex structure, CNNs are still prone to generalization errors. Notably, their many complex features make them difficult to train and handle. Hence, directly incorporating dimensionality reduction into a classifier model is one of the most promising solutions [3].

Guaranteeing stability of inference by neural networks (NNs), in particular deep NNs, is difficult because they involve a large number of computational processes. This problem is known as a black box problem. Consequently, the use of NNs in applications that require high stability, such as automated driving (Akai, Morales, Takeuchi et al., 2017, Akai, Morales, Yamaguchi et al., 2017), is challenging. The NNs are expected to have major role in various autonomous systems; however, it is necessary to guarantee the stability of their inference. In this work, we focus on classification problems with NNs and investigate their performance from topological insight to analyze the stability. We define stable NNs to mean that they do not generate outliers as their inference because the inference around the outliers changes drastically. In this investigation, we use persistent homology (PH) (Edelsbrunner & Zomorodian, 2002), which is widely used in topological data analysis (TDA), in combination with a method to estimate the confidence set for persistence diagrams (PDs) (Fasy et al., 2014). The PH is a method for computing topological features of a space at different spatial resolutions and the PD is a representation method of the PH (these are detailed in Section 3). The investigation is able to show that the NNs do not acquire unstable mapping for forcible classification, that is, it could help to guarantee stability of applications using NNs [4].

The [predictive probabilities](https://www.sciencedirect.com/topics/mathematics/predictive-probability) at the output layer of [neural network](https://www.sciencedirect.com/topics/neuroscience/neural-networks) [classifiers](https://www.sciencedirect.com/topics/computer-science/classification-machine-learning) are often misinterpreted as model (epistemic) uncertainty ([Gal & Ghahramani, 2016](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b6)). [Bayesian statistics](https://www.sciencedirect.com/topics/mathematics/bayesian-statistics) provides a coherent framework for representing uncertainty in neural networks ([Goodfellow et al., 2016](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b8), [MacKay, 1992](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b26)), but has not so far gained widespread use in deep learning — presumably due to the high computational cost that traditionally comes with second-order methods. Recently, [Gal and Ghahramani (2016)](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b6) developed a theoretical framework which casts dropout at test time in deep neural networks as approximate [Bayesian inference](https://www.sciencedirect.com/topics/mathematics/bayesian-inference). Due to its mathematical elegance and negligible computational cost, this work has caught great interest in a variety of different fields ([Litjens et al., 2017](https://www.sciencedirect.com/science/article/pii/S0893608021004056" \l "b24), [Loquercio et al., 2020](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b25), [Yan et al., 2020](https://www.sciencedirect.com/science/article/pii/S0893608021004056" \l "b45), [Zhu and Laptev, 2017](https://www.sciencedirect.com/science/article/pii/S0893608021004056" \l "b47)), but has also generated questions as to what types of uncertainty these approximations actually lead ([Osband, 2016](https://www.sciencedirect.com/science/article/pii/S0893608021004056" \l "b33), [Osband et al., 2016](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b34)) and what types are relevant ([Kendall & Gal, 2017](https://www.sciencedirect.com/science/article/pii/S0893608021004056" \l "b17)). For a general treatment of uncertainty in [machine learning](https://www.sciencedirect.com/topics/computer-science/machine-learning), we refer to [Hüllermeier and Waegeman (2020)](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b14).

Epistemic uncertainty is commonly understood as the reducible component of uncertainty — the uncertainty of the model itself, or its parameters. In our context this amounts to the uncertainty in the estimated class probabilities due to limited amount of training data. While the [epistemic uncertainty](https://www.sciencedirect.com/topics/engineering/epistemic-uncertainty) can be reduced by increasing the amount of training data, the other component of uncertainty known as [aleatoric uncertainty](https://www.sciencedirect.com/topics/engineering/aleatoric-uncertainty), is irreducible and stems from the uncertainty in the label assignment process ([Song, Kim, Park, & Lee, 2020](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b42)). However, in this paper we only address the epistemic part, and treat the labels as constant when estimating uncertainty.

Our approach goes back to the work of [MacKay (1992)](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b26), and we show that the above reasoning leads to the method known as the [Delta method](https://www.sciencedirect.com/topics/mathematics/delta-method)[1](https://www.sciencedirect.com/science/article/pii/S0893608021004056" \l "fn1) ([Hoef, 2012](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b13), [Khosravi and Creighton, 2011](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b18), [Newey and McFadden, 1994](https://www.sciencedirect.com/science/article/pii/S0893608021004056#b30)) in statistics. However, as the Delta method depends on the empirical [Fisher information](https://www.sciencedirect.com/topics/engineering/fisher-information) matrix which grows quadratically with the number of neural network parameters P – its direct application in modern deep learning is prohibitively expensive. We therefore propose a low cost variant of the Delta method applicable to L2-regularized deep neural networks based on the top K [eigenpairs](https://www.sciencedirect.com/topics/engineering/eigenpairs) of the [Fisher information matrix](https://www.sciencedirect.com/topics/engineering/fisher-information-matrix). We address efficient computation of full-rank approximate eigendecompositions in terms of either the exact inverse Hessian, the inverse outer-products of gradients (OPG) approximation or the so-called Sandwich estimator. Further, we exhibit the fact that deep learning classifiers tend to be heavily over-parameterized. This leads to flat Fisher information eigenvalue spectra which we show can be exploited in terms of a simple [linearization](https://www.sciencedirect.com/topics/engineering/linearization) [5].

Convolutional [neural networks](https://www.sciencedirect.com/topics/neuroscience/neural-networks) (CNNs) ([Krizhevsky, Sutskever, & Hinton, 2012](https://www.sciencedirect.com/science/article/pii/S0893608021003403#b19)) and backpropagation (BP) learning ([Rumelhart, Hinton, & Williams, 1986](https://www.sciencedirect.com/science/article/pii/S0893608021003403#b33)) form the most powerful combination of recent [machine learning methods](https://www.sciencedirect.com/topics/engineering/machine-learning-method). However, when these learning methods are likened to information processing in the brain, several objections, mainly directed toward the [BP](https://www.sciencedirect.com/topics/engineering/backpropagation) of error information, are raised ([Bartunov et al., 2018](https://www.sciencedirect.com/science/article/pii/S0893608021003403" \l "b2)). Conversely, the brain is a more powerful [learning machine](https://www.sciencedirect.com/topics/computer-science/machine-learning) than any current deep learning system, and it simultaneously realizes the ability to scale computations to a very large network and strong semi-supervised learning. Therefore, it can be said that clarifying the [operational principles](https://www.sciencedirect.com/topics/computer-science/operational-principle) of the brain is important for realizing a superior learning machine.

In recent years, biologically motivated methods have predominantly been studied in cases in which learning can be performed by estimating the BP errors obtained from other feedback signals without the actual use of BP errors ([Bengio, 2014](https://www.sciencedirect.com/science/article/pii/S0893608021003403#b3), [Hinton, 2007](https://www.sciencedirect.com/science/article/pii/S0893608021003403#b14), [LeCun, 1986](https://www.sciencedirect.com/science/article/pii/S0893608021003403" \l "b22), [Lee et al., 2015](https://www.sciencedirect.com/science/article/pii/S0893608021003403#b26), [Lillicrap et al., 2016](https://www.sciencedirect.com/science/article/pii/S0893608021003403" \l "b27), [Nø kland and Eidnes, 2019](https://www.sciencedirect.com/science/article/pii/S0893608021003403#b29), [Nøkland, 2016](https://www.sciencedirect.com/science/article/pii/S0893608021003403#b30), [Samadi et al., 2017](https://www.sciencedirect.com/science/article/pii/S0893608021003403" \l "b36)). However, because these methods also represent supervised learning using labeled data, in which considerable amounts of labeled data are required for learning, brain-like computing cannot be realized. To solve this problem, a version of [unsupervised learning](https://www.sciencedirect.com/topics/computer-science/unsupervised-learning) that does not use any BP information is required.

Numerous studies on the unsupervised learning of visual representation have been conducted, and it was initially studied using analytical methods. Some previous studies applied [independent component analysis](https://www.sciencedirect.com/topics/engineering/independent-component-analysis) to obtain visual bases from natural scene images ([Hyvärinen and Hoyer, 2001](https://www.sciencedirect.com/science/article/pii/S0893608021003403#b16), [Olshausen and Field, 1996](https://www.sciencedirect.com/science/article/pii/S0893608021003403" \l "b31)). This method was subsequently expanded to a neural network (NN)-like regime ([Le, 2013](https://www.sciencedirect.com/science/article/pii/S0893608021003403" \l "b21)). Because these methods were vastly different from conventional NNs that were trained using BP learning, the unification of neural and nonneural mechanisms has been challenging [6].

Neural networks and deep learning have been increasingly finding successful application in many fields, such as in speech recognition, image understanding, natural language processing and autonomous vehicles (Goodfellow et al., 2016, LeCun et al., 2015, Mandic and Chambers, 2001, Schmidhuber, 2015). Gradient-based optimization is of core practical importance in this context (Mandic and Chambers, 2000, Mandic et al., 2001, Xu et al., 2015), with stochastic gradient descent (SGD) (Robbins & Monro, 1951) being a dominant method. While variants of SGD for deep learning do exist, the ADAM (Kingma & Ba, 2014), ADAGrad (Duchi, Hazan, & Singer, 2011), ADADelta (Zeiler, 2012) and RMSProp (Tieleman & Hinton, 2012) algorithms are most frequently used in the optimization of various objectives (see e.g., the pytorch,1 mxnet,2 tensorflow,3 and keras4 documentation) [7].

Spiking neural networks (SNNs) are brain-inspired models that transmit spikes between neurons for event-driven energy-efficient computation. SNNs can be implemented with less energy on neuromorphic hardware (Akopyan et al., 2015, Davies et al., 2018, Pei et al., 2019, Roy et al., 2019), which is regarded as the third generation of neural network models (Maass, 1997) and is gaining increasing attention as an alternative to artificial neural networks (ANNs).

Different from ANNs, however, directly supervised training of SNNs is a hard problem due to the complex spiking neuron model which is discontinuous. To handle this problem, converting ANNs to SNNs (Deng and Gu, 2021, Hunsberger and Eliasmith, 2015, Rathi et al., 2020, Rueckauer et al., 2017, Sengupta et al., 2019, Yan et al., 2021), or many direct SNN training methods (Bellec et al., 2018, Bohte et al., 2002, Jin et al., 2018, Kim, Kim et al., 2020, Meng, Xiao et al., 2022, Neftci et al., 2019, Shrestha and Orchard, 2018, Wu et al., 2018, Wu et al., 2019, Xiao et al., 2021, Zhang and Li, 2019, Zhang and Li, 2020, Zheng et al., 2021) have been proposed to incorporate deep learning into SNNs (Tavanaei, Ghodrati, Kheradpisheh, Masquelier, & Maida, 2019). While these methods can partly solve the problems of unsatisfactory performance or high latency, they require complex computation for gradient calculation or approximation rather than the same spike-based computation as the inference process. They aim at training SNNs with general computational operations and deploying trained models for energy-efficient inference with spiking operations [8].

In the past few years, deep neural networks (DNNs) (LeCun et al., 2015) have achieved great success in machine learning, and also hastened many real artificial intelligence applications (Bai et al., 2021, Woźniak, Siłka et al., 2021, Woźniak, Wieczorek et al., 2021). For gaining stronger expressive ability, the scale of DNNs has been growing continuously, whereupon the inherent redundancy of DNNs (Denil et al., 2013) also increases heavily. Therefore, the compression of DNNs has become a hot research topic (Deng et al., 2020), and promotes some of the latest practical applications, such as efficient graph convolutional network (Dong et al., 2021), tensor decomposed visual question answering system (Bai et al., 2021), etc. In general, among amounts of approaches to compress DNNs, tensor decomposition is promising due to several special advantages over other methods, e.g., foundation of explicable mathematics, easy implementation with training from scratch, etc. (Deng et al., 2020). These positive characteristics have hastened many researchers to develop **two ways** to shrink DNNs in the view of tensors (Kossaifi, Toisoul et al., 2020). As shown in Fig. 1, one way is just tensorizing the original weight matrices or convolutional kernels into higher-order tensors to decompose them to produce small factor tensors as the new weights, the other is directly mapping the data structure of tensor decompositions to the local neural architectures (extra nonlinear activation functions are necessary to introduce new neurons) to invent new compact designs. Here we term these two ways as tensorizing way and mapping way, respectively [9].

Convolutional neural networks (CNNs) achieve the best performance on various image classification tasks. CNNs learn to classify images from a regular pixel-grid representation of the image. Two limitations of this approach are: 1) Although not all pixels provide an equal amount of new information, by design, the filters in the first layer of a CNN operate on each pixel from top-left to bottom-right in the same way; 2) CNNs require images to be of the same size. Therefore, images are typically resized to a prescribed size before feeding into a CNN. In applications that use standard CNN architectures or pre-trained models on a new image classification dataset, the images are typically uniformly downsampled to meet the input size requirements of the architecture being used. Uniform downsampling may be suboptimal as real data naturally exhibits spatial and multiscale heterogeneity. Few studies have explored the impact of input image resolution on model performance [1], despite its recognized importance [2].

In contrast to CNNs, graph neural networks (GNN) learn from a graph representation of the image. While CNNs are still the state-of-the-art, several studies have shown promise in classifying images from graphs using GNNs [3], [4], [5], [6], [7], [8], [9]. Unlike CNNs, the GNNs used in these studies do not require input graphs to have the same size/structure (e.g., number of nodes and edges can be different). However, these studies have been restricted to graphs that represent either a regular grid of pixels or similar-sized superpixels. In the latter, a single target number of superpixels is defined for an entire dataset irrespective of differences across images and their intrinsic multiscale structure [10].

[Ensemble methods](https://www.sciencedirect.com/topics/computer-science/ensemble-method) have been widely used for improving the results of the best single [classification model](https://www.sciencedirect.com/topics/computer-science/classification-models). A large body of works have achieved better performance mainly by applying one specific ensemble method. However, very few works have explored complex fusion schemes using heterogeneous ensembles with new aggregation strategies. This paper is three-fold: 1) It provides an overview of the most popular ensemble methods, 2) analyzes several fusion schemes using MNIST as guiding thread and 3) introduces MNIST-NET10, a complex heterogeneous fusion architecture based on a degree of certainty aggregation approach; it combines two heterogeneous schemes from the perspective of data, model and fusion strategy. MNIST-NET10 reaches a new record in MNIST with only 10 misclassified images. Our analysis shows that such complex heterogeneous fusion architectures based on the degree of certainty can be considered as a way of taking benefit from diversity [11].

Metamodels, or models of models, map defined model inputs to defined model outputs. Typically, metamodels are constructed by generating a dataset through sampling a direct model and training a [machine learning algorithm](https://www.sciencedirect.com/topics/engineering/machine-learning-algorithm) to predict a limited number of model outputs from varying model inputs. When metamodels are constructed to be computationally cheap, they are an invaluable tool for applications ranging from topology optimization, to [uncertainty quantification](https://www.sciencedirect.com/topics/engineering/uncertainty-quantification), to multi-scale simulation. By nature, a given metamodel will be tailored to a specific dataset. However, the most pragmatic metamodel type and structure will often be general to larger classes of problems. At present, the most pragmatic metamodel selection for dealing with mechanical data has not been thoroughly explored. Drawing inspiration from the benchmark datasets available to the [computer vision](https://www.sciencedirect.com/topics/engineering/computervision) research community, we introduce a benchmark data set (Mechanical MNIST) for constructing metamodels of [heterogeneous material](https://www.sciencedirect.com/topics/materials-science/heterogeneous-material) undergoing [large deformation](https://www.sciencedirect.com/topics/engineering/large-deformation). We then show examples of how our benchmark dataset can be used, and establish baseline metamodel performance. Because our dataset is readily available, it will enable the direct quantitative comparison between different metamodeling approaches in a pragmatic manner. We anticipate that it will enable the broader community of researchers to develop improved metamodeling techniques for mechanical data that will surpass the baseline performance that we show here [12].

Originally inspired by the biological structure of networks of interacting neurons (Fukushima, 1980, Hubel and Wiesel, 1962, McCulloch and Pitts, 1943, Rosenblatt, 1958), neural networks (NNs) have since developed a suite of algorithmic tools (such as backprop, convolutional kernels, etc.) which, combined into complex and deep NN architectures, have achieved unprecedented success in a wide array of machine learning (ML) tasks (Goodfellow et al., 2016, LeCun et al., 2015, Schmidhuber, 2015). However, they have trouble learning from few samples. We seek to improve NN performance on such tasks by revisiting the well of biological example to characterize key biological structures involved in learning, for transfer to the NN context.

In this work, we apply MothNet, a NN architecture closely based on the Moth Olfactory Network, to the task of learning vectorized MNIST digits. MothNet successfully learns to read given very few training samples (1–10 samples per class) and in fact outperforms standard machine learning methods such as Nearest-Neighbors, support-vector machines (SVMs), Neural Nets (i.e. standard fully-connected, feed-forward, non-convolutional nets with softmax loss functions), and convolutional neural networks (CNNs), in the few-samples regime (Fig. 5). In addition, it performs comparably to specialized one-shot learning NNs, but without their need for pre-training to initialize NN weights [13].

In this note, we follow two directions to improve the performance of CNN classifiers. The first is to apply to CNN units the same improvement techniques that we have successfully used with Stacked Denoising Auto-Encoder classifiers. This leads to obtain a new performance record when classifying MNIST digits. The second consists of applying a Stacked Denoising Auto-Encoder classifier to the output of the best of the previous designs, trying to take advantage of the limitations of CNN architectures. An even better classification record is obtained for MNIST.

The above results permit to conclude that combining improvement techniques and stacking deep machines of different nature can be useful to better solve other real-world problems.

Increasing the number of layers leads to less accurate results (He et al., 2016). Hence, researchers have developed methods to improve CNN performance without changing the architecture. Hyperparameters include the size of kernels, number of kernels, length of strides, and pooling size, which affect the performance and training speed of CNNs. Moreover, the impact of hyperparameters increases with the complexity of the network. Hence, the most popular method of improving CNN performance is to optimize the hyperparameters (Hazan et al., 2017, Zhang et al., 2019).

CNN hyperparameter optimization is an integer programming problem. In early studies, manual designs were often used for hyperparameters; that is, scholars or experts adjust the hyperparameters according to experience and expertise, which is time-consuming during testing. Several heuristic approaches, such as grid search, randomized search (Bergstra and Bengio, 2012), Bayesian optimization, and gradient-based optimization, have been developed for hyperparameter optimization. The drawback of these methods is that they are inefficient in a high-dimensional space because the number of evaluations increases exponentially as the number of hyperparameters increases [14].

The rise of artificial intelligence algorithms represented by deep neural networks (DNNs) has promoted the development of many technical fields: autonomous driving, smart home, and medical diagnosis. As Moore's Law slows, electronic chips will struggle to keep up with the growing power of artificial intelligence. The optical neural network has gradually attracted the attention of the industry and academia because it is expected to solve the problem of insufficient computing power caused by the failure of Moore's Law. Recently, optical neural networks based on silicon photonic technology have demonstrated the advantages of integrated photonic chips over traditional electronic chips in terms of cost and energy consumption, in addition to the ultra-high computing speed with photonic devices having great temporal and spatial bandwidth. On the other hand, the optical neural network technology based on the fundamental theory of Fourier optics has also attracted extensive research interest.

In Ref, an optoelectronic hybrid convolutional neural network for image classification is proposed. The system achieves convolutional operations and image feature pre-extraction at the optical end through a simple 4-f system and a phase mask located in the Fourier plane, thus improving the performance of image classification tasks. The all-optical diffractive deep neural network architecture (D2NN) has been proposed in the Ref to realize data inference and lens-free imaging using light propagation and complex amplitude modulation of light by multilayer 3D printing masks. The introduction of D2NN has provided a set of implementation paths for the research and development of free-space all-optical information processing systems. Many researchers have made efforts to expand the applications of optical networks and the optimization of structures and algorithms [15].

Artificial neural networks, as one of the hottest research directions in the field of deep learning, can effectively represent complex relationships and data in the real world. Since artificial neural networks can better simulate the human brain to reflect the real world, more and more artificial neural networks have been proposed subsequently, such as convolutional neural networks, long-term and short-term memory networks, adversarial neural networks, spike neural networks etc. After decades of development, neural network theory has been widely successful in many research areas like pattern recognition, automatic control, signal processing, assisted decision making, and artificial intelligence.

Artificial neural networks have been used on a large scale in the imaging fields of face recognition, image semantic segmentation, image classification, target recognition and image style transformation. From the earliest time-delay networks and LeNet-5 networks to the current fast YOLO networks for image processing [8], neural networks have penetrated into modern life after half a century of development. As for instance, Elakkiya R. et al. used hybrid object detection adversarial network for cancer image recognition, making cancer diagnosis faster and more accurate [9]; Elakkiya R. et al. proposed a hybrid deep neural network that can effectively classify unbalanced MicroRNA data, providing a powerful tool for accurate determination of cancer factors [10]; Deepak K. et al. proposed a deep neural learning technique with long-term and short-term memory can enable computers to recognize human gestures more accurately, which will further facilitate human–computer communication [5]; Hossein C. et al. used MASKRCNN for weed estimation to promote automation in agriculture ; Norman Y. used neural networks for museum visitation model predicts visitor visitation patterns, allowing museums to better improve the visitor experience .

Even though the algorithm of convolutional neural network has been developed to the stage where it can be detected in real time the current neural network designed using software algorithm still receives the constraints of computational efficiency and power consumption. Software-based neural networks involve a large number of sliding convolution operations in their operations, which are extremely dependent on the computational efficiency of the GPU. These neural networks require a large number of GPUs to run and also lead to a large amount of energy consumption. Therefore, there is a great need for higher performance and lower power consumption network designs in the development of neural networks. Hardware-based neural networks have very obvious advantages. The memristor proposed by Chua L. in 1971 and successfully fabricated by Hewlett-Packard Labs in 2008 is well suited for the development of convolutional neural networks in terms of higher performance computing and lower power consumption. Unlike the traditional Von Neumann-based architecture, the memristor-based neural network implements data storage and computation in the circuit, which can achieve more efficient performance. At the same time, memristors also have lower energy consumption compared to traditional transistors, which can save a lot of energy. Therefore, more and more people are now studying neural networks based on memristors [16].

In recent years, significant improvement has been achieved in the field of computer vision. Convolutional Neural Networks (CNN) based supervised learning approaches have been widely used for hand-writing recognition and computer vision tasks. CNN networks have also found wide range of applications in risk sensitive applications like drone-based crowd modelling and medical image analysis applications. Recently, convolutional and deconvolution operations have been combined with a RNN based negative feedback operation to enhance images in real-time. Application of CNN based deep learning models for sensitive tasks across a wide range of application necessitates the accurate modelling of the deep network architecture. Although, the CNN based research works have showed promising results, design of suitable network architecture has been a challenging task and requires expert knowledge. The network performance of the CNN algorithms are highly influenced by the data distribution, class separability, number of layers (depth), hyperparameters and the network organization. Errors in determination of the CNN architecture leads to loss of performance. For example, a shallow network is prone to underfitting, while networks with excessive number of layers lead to overfitting and suffer from the vanishing gradient problem. The stochastic approach to network determination is imprecise and results in CNNs with sub-optimal network architecture.

Major approaches to construct deep network architecture have either focused on modifying existing networks or automatically determining novel networks to maximise network performance over a given data distribution. Preliminary works on evolving neural networks used genetic algorithms to mimic the natural selection process in genetic mutations. The genetic algorithm based evolving neural networks used mutation and cross-overs to determine the best set of hyperparameters. Using a similar biologically inspired approach, has proposed a high throughput random sampling approach to automatically determine the network architecture of neural networks. Inspired by the success of such biologically inspired approaches, statistical learning theories have also been widely explored to automatically determine the architecture of neural networks. Among them have used Bayesian statistics to simultaneously optimize the network architecture of neural networks to maximise their performance. Although such approaches have been successful in improving the network performance by optimizing the network architecture, they cannot be directly adopted for CNN. The heterogeneous nature of the various layers of a CNN necessitates tailored evolution approaches able to handle the unique characteristics of each of the layers. Besides, such approaches are meta-heuristic in nature and closely mimic the stochastic network architecture determination process [17].

The Particle Swam Optimization(PSO) approach has been popular among researchers to automatically determine the network architecture of deep CNN networks [18]. The PSO approach has been greatly improved in the recent years, [19] proposed a variable length particle swam optimization approach to encode the CNN architecture of varying lengths into a common particle, thus enabling the evaluation of network architectures of various depths using a common fitness evaluation method. The evolutionary computing based approaches to deep network architecture determination have also developed AutoML algorithms to collectively determine the network architecture and optimize the model hyperparameters in a combined learning approach. These evolutionary approaches to network architecture determination are great for networks with less complex architectures and well defined search spaces. But for CNN architectures with large number of deep layers and large number of network parameters, the evolutionary algorithms are not scalable. These evolutionary algorithms are not suitable for complex CNN architectures with large number of convolutional layers and fully connected layers with large number of filters and nodes respectively.

A large body has been work has been done in addressing the shortcomings of evolutionary algorithms for deep network architecture determination. With the advances in computational infrastructure and size of the databases, reinforcement learning-based approaches have been explored to autonomously determine complex CNN architectures. Among them, Neural Architecture Search (NAS) [20] proposed a reinforcement learning based common learning strategy to concurrently determine both the deep network architecture and its parameters. A fitness function based reward mechanism is proposed in NAS to optimize the network architecture of the CNN, whereas the error in network performance is backpropagated through the deep network to update the network parameters. In comparison with the previous evolutionary algorithms, the NAS improves the classification performance for large datasets with millions of parameters. However, the iterative learning approach of NAS is one of the major drawbacks, the reward based learning algorithm often requiring thousands of models to be trained to determine the optimal deep network architecture. For the CIFAR dataset, NAS proposed 12,000 CNN network architectures, whose performance were used as the reward for the reinforcement based policy to optimize the network architecture [20]. Therefore, there is a need to develop automatic architecture determination methodologies that are reliable and scalable across various classification problems with different feature distributions, number of classes and small sample sizes [18].

We propose a model for image [classification](https://www.sciencedirect.com/topics/computer-science/classification) by attentional search. Analogous to how humans scan an image by a sequence of saccades, in this model, an attentional window of size much smaller than the target size scans the target by a sequence of “saccades”, integrates the information acquired, and makes a [classification](https://www.sciencedirect.com/topics/computer-science/classification) decision. In order to process a sequence of attended image segments, the network must have memory, which is incorporated through 3 kinds of recurrent elements in the [network architecture](https://www.sciencedirect.com/topics/computer-science/network-architecture): Elman connections, Jordan connections, and Flip-flop neurons. The architecture of the model is designed as three separate channels labeled as – [classifier](https://www.sciencedirect.com/topics/computer-science/classification-machine-learning) network, eye-position network, and saccade network. Multiple attentional windows with different resolutions and a common center are given as input to the [classifier](https://www.sciencedirect.com/topics/computer-science/classification-machine-learning) network and the saccade network. The heat-map representation of the location of the attentional windows is given as input to the eye-position network. The saccade network predicts the next jump of the attention windows with the help of reward signals received by the classifier network. The output features of all the three channels are concatenated, before finally terminating in two output layers representing class prediction and next saccade prediction. The model is trained using deep Q-learning algorithm. Attentional search model is evaluated on MNIST handwritten digit, Kannada MNIST, Medical-MNIST, OCTMNIST, and QuickDraw datasets. Translated and Cluttered Translated versions of each dataset are generated to perform the task of classification based on local target search. Original datasets are used to show the task of classification based on search with global target integration. We also evaluate the saccade performance on Extended Yale Face B database. In various problem cases, the model exhibits comparable or superior performance to a state-of-the-art recurrent attention model [19].

In view of low detection accuracy, high false alarm rate and lack of labeled data in traditional network anomaly detection models, this paper proposes an anomaly detection classification model that incorporates federated learning and MGVN (FL-MGVN). The experiments are conducted on the benchmark datasets NSL-KDD, MNIST and Fashion-MNIST, which show that the proposed FL-MGVN classification model has higher recognition performance and classification accuracy in both binary classification and multi-classification problems than other algorithms. The average AUC on MNIST and Fashion-MNIST reached 0.954 and 0.937, respectively.

The contributions of this paper are summarized as follows:

We propose an anomaly detection classification model that incorporates federated learning and mixed Gaussian variational self-encoding networks which can effectively address network attack and sample dissimilarity.

The proposed MGVN network model first constructs a variational self-encoder using a mixed Gaussian prior to extracting features from the input data, and then constructs a deep support vector network with a mixed Gaussian variational self-encoder.

We have conducted a series of experiments on benchmark datasets such as NSL-KDD, MNIST and Fashion-MNIST to verify the anomaly detection performance in both binary classification and multi-classification problems [20].

Increasing the number of layers leads to less accurate results (He et al., 2016). Hence, researchers have developed methods to improve CNN performance without changing the architecture. Hyperparameters include the size of kernels, number of kernels, length of strides, and pooling size, which affect the performance and training speed of CNNs. Moreover, the impact of hyperparameters increases with the complexity of the network. Hence, the most popular method of improving CNN performance is to optimize the hyperparameters (Hazan et al., 2017, Zhang et al., 2019).

CNN hyperparameter optimization is an integer programming problem. In early studies, manual designs were often used for hyperparameters; that is, scholars or experts adjust the hyperparameters according to experience and expertise, which is time-consuming during testing. Several heuristic approaches, such as grid search, randomized search (Bergstra and Bengio, 2012), Bayesian optimization, and gradient-based optimization, have been developed for hyperparameter optimization. The drawback of these methods is that they are inefficient in a high-dimensional space because the number of evaluations increases exponentially as the number of hyperparameters increases (Injadat et al., 2020, Hinton, 2012, Hsu et al., 2003) [21].

To solve this problem, artificial intelligence techniques, including genetic algorithms (Aszemi and Dominic, 2019, Johnson et al., 2020, Loussaief and Abdelkrim, 2018, Xiao et al., 2020), particle swarm optimization (PSO) (Huang, 2015, Lorenzo et al., 2017, Yamasaki et al., 2017), and the artificial bee colony algorithm (Zhu et al., 2019), have been proposed. However, most of these methods change the CNN structure or combine different algorithms to optimize the performance and are complex and difficult for users to understand. Thus, there is a need for a simple algorithm that does not change the CNN structure to optimize for hyperparameters.

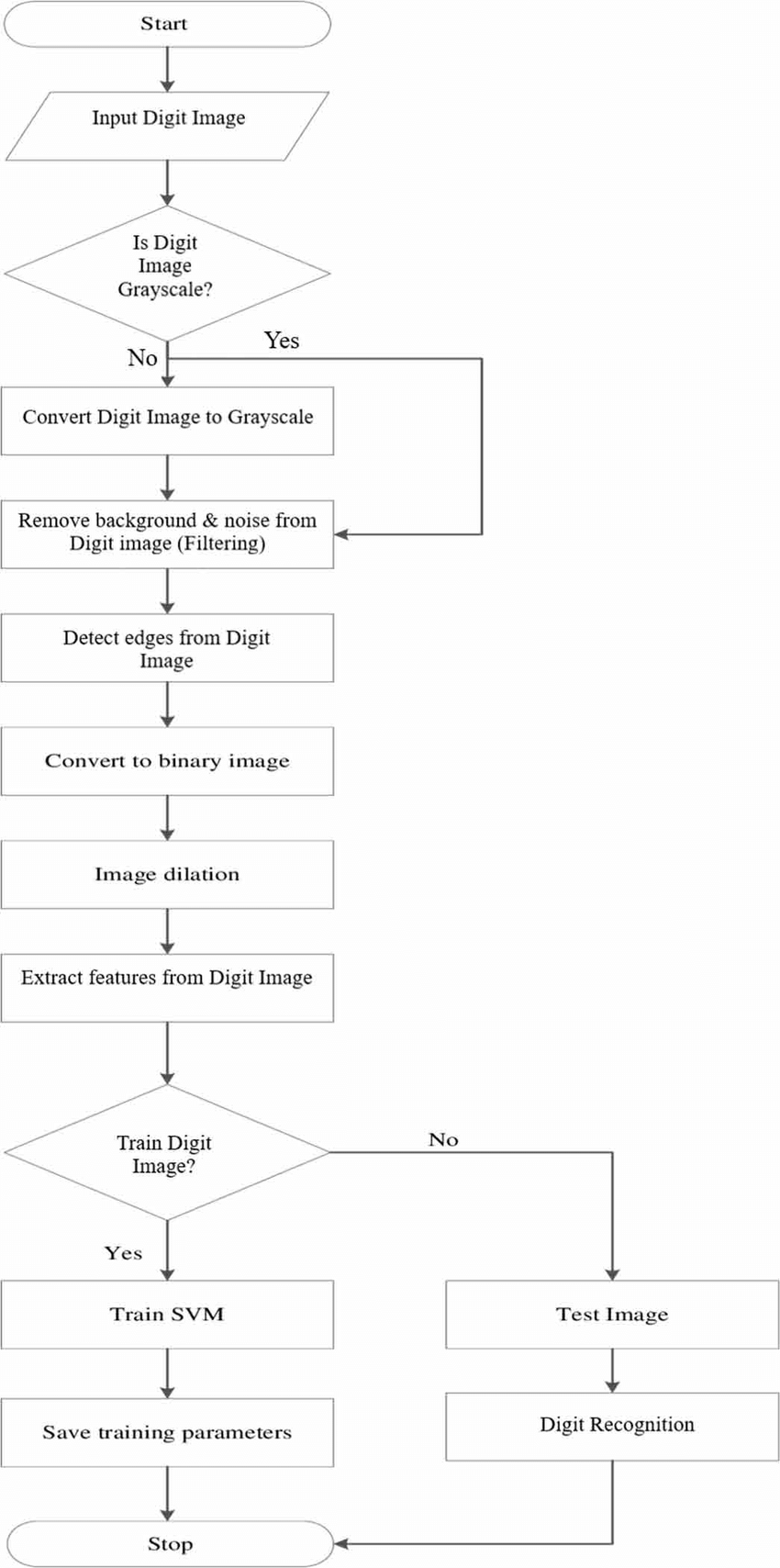
The objective of this study was to use the simplified swarm optimization (SSO) algorithm proposed by Yeh to tune the CNN hyperparameters (Goodfellow et al., 2016). SSO is not only simple and easy to understand but also efficient. Many studies have demonstrated the effectiveness of SSO for solving optimization problems (Yeh, 2012a, Yeh, 2012b, Yeh, 2011, Yeh, 2014, Yeh et al., 2012); however, SSO has not been applied to the hyperparameter optimization problem. Therefore, in this study, a new algorithm called SSO-LeNet was developed to apply SSO to the original LeNet architecture without changing the layers and was validated with different datasets for automated hyperparameter optimization.

The main contributions of this study are summarized as follows.

1. A hyperparameter optimization algorithm called SSO-LeNet is proposed to improve LeNet.
2. SSO-LeNet is the first SSO-based algorithm to improve LeNet.
3. SSO-LeNet outperforms LeNet in solution quality with a significant improvement in test time.
4. SSO-LeNet outperforms PSO-LeNet in solution quality.
5. SSO-LeNet has considerable potential for improving AlexNet, GoogLeNet, etc. efficiently [22].

Most capsule network designs rely on traditional matrix multiplication between capsule layers and computationally expensive routing mechanisms to deal with the capsule dimensional entanglement that the matrix multiplication introduces. By using Homogeneous Vector Capsules (HVCs), which use element-wise multiplication rather than matrix multiplication, the dimensions of the capsules remain unentangled. In this work, we study HVCs as applied to the highly structured MNIST dataset in order to produce a direct comparison to the capsule research direction of Geoffrey Hinton, et al. In our study, we show that a simple convolutional [neural network](https://www.sciencedirect.com/topics/neuroscience/neural-networks) using HVCs performs as well as the prior best performing capsule network on MNIST using 5.5× fewer parameters, 4× fewer training epochs, no reconstruction sub-network, and requiring no routing mechanism. The addition of multiple [classification](https://www.sciencedirect.com/topics/computer-science/classification) branches to the network establishes a new state of the art for the MNIST dataset with an accuracy of 99.87% for an ensemble of these models, as well as establishing a new state of the art for a single model (99.83% accurate) [23].

**1.5: Proposed work**



**Chapter 2**

**INTRODUCTION OF LSTM MODEL**

**2.1 Hardware/Software requirements:**

1. Google Collab

2. Python Language Libraries- NumPy, Pandas, Matplotlib, Sklearn etc.

3. Libraries- NumPy, Pandas, Matplotlib, Sklearn etc.

**2.2 Introduction to Dataset:**

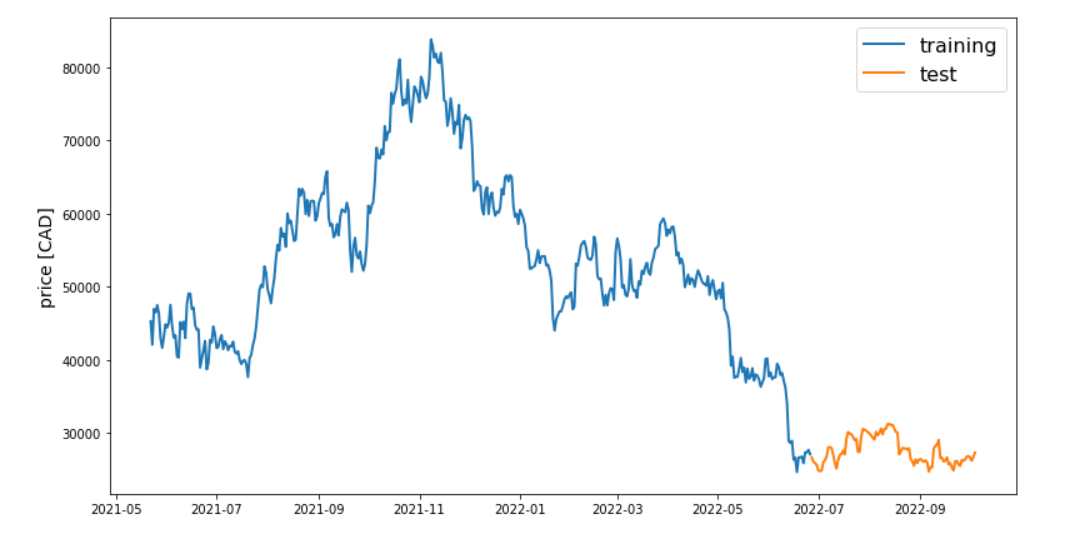
Cryptocurrencies are gaining popularity day by day, and their analysis is a fascinating and demanding research topic. The average daily trading volume of Bitcoin was $67 billion in May 2021. A peculiar feature of cryptocurrencies is that WE are not generally issued by a central authority, making them insusceptible to any governmental impedance. Cryptocurrency rates are closely related to news and influenced by tweets. However, no available dataset can analyze the crypto market adequately. We present a benchmark dataset for Cryptocurrency Price Movement Prediction based on historical prices. We also demonstrate use-cases by providing adapted baseline methods and a quantitative results analysis on our dataset.

We Have taken the data set from crypto compare.com it is a famous website for machine learning engineer to take data set for making machine learning models it has various range of data set of crypto currencies for example bitcoin, Litecoin, Ethereum, shibainu, dogecoin etc. we have taken bitcoin data set of 1 year 5 months approx. it consists of various parameters time, high, low, open, volume from, volume to, close here the time is showing date of the bitcoin prices. High is showing the highest price of bitcoin at that day, low is showing lowest price of bitcoin that day, and opening show where bitcoin last closed and that time, volume from means where the volume of start in bitcoin trade, volume to means where the end time of bitcoin were trade for which how much buyers and sellers is there. And for close means the last time of close the market of crypto. used api key to connect our data set to lstm model. In that way we can use dataset of any cryptocurrency using API keys from cryptocompare.com

All models we’ve built so far do not allow for operating on sequence data. Fortunately, we can use a special class of Neural Network models known as recurrent neural networks (RNNs) just for this purpose. *RNNs* allow using the output from the model as a new input for the same model. The process can be repeated indefinitely.

One serious limitation of *RNNs* is the in a sequence (e.g. Is there a dependency between today`s price and that 2 weeks ago?). One way to handle the situation is by using an **Long short-term memory (LSTM)** variant of *RNN*.

The default LSTM behavior is remembering information for prolonged periods of time. Let’s see how you can use LSTM in Keras.



****

**2.3 Description of LSTM Model:**

Long short-term memory (LSTM) is an [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) used in the fields of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) and [deep learning](https://en.wikipedia.org/wiki/Deep_learning). Unlike standard [feedforward neural networks](https://en.wikipedia.org/wiki/Feedforward_neural_network), LSTM has feedback connections. Such a [recurrent neural network](https://en.wikipedia.org/wiki/Recurrent_neural_network) (RNN) can process not only single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected [handwriting recognition](https://en.wikipedia.org/wiki/Handwriting_recognition), [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition), [machine translation](https://en.wikipedia.org/wiki/Machine_translation), robot control, video games, and healthcare. LSTM has become the most cited neural network of the 20th century.

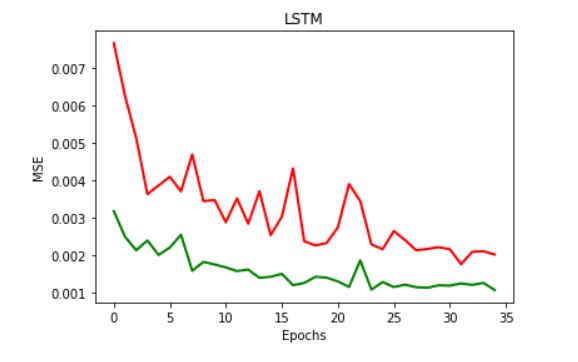
The name of LSTM refers to the analogy that a standard RNN has both "long-term memory" and "short-term memory". The connection weights and biases in the network change once per episode of training, analogous to how physiological changes in synaptic strengths store long-term memories; the activation patterns in the network change once per time-step, analogous to how the moment-to-moment change in electric firing patterns in the brain store short-term memories. The LSTM architecture aims to provide a short-term memory for RNN that can last thousands of timesteps, thus "long short-term memory".

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to [classifying](https://en.wikipedia.org/wiki/Classification_in_machine_learning), [processing](https://en.wikipedia.org/wiki/Computer_data_processing) and [making predictions](https://en.wikipedia.org/wiki/Predict) based on [time series](https://en.wikipedia.org/wiki/Time_series) data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the [vanishing gradient problem](https://en.wikipedia.org/wiki/Vanishing_gradient_problem) that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, [hidden Markov models](https://en.wikipedia.org/wiki/Hidden_Markov_models) and other sequence learning methods in numerous applications

**2.4 Diagram of LSTM Model**

**2.5 Predicted Graph**

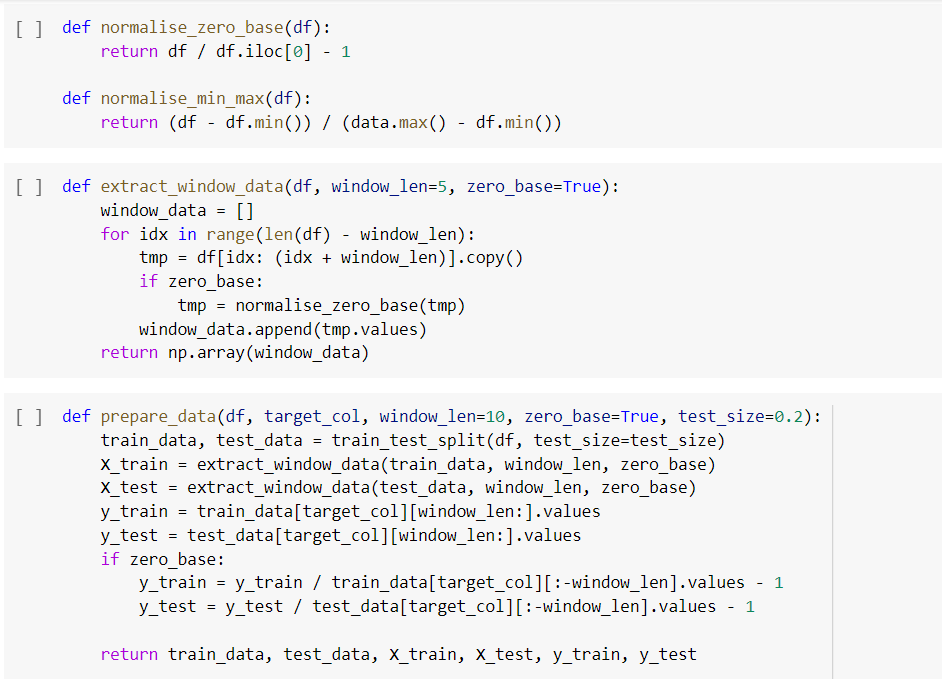


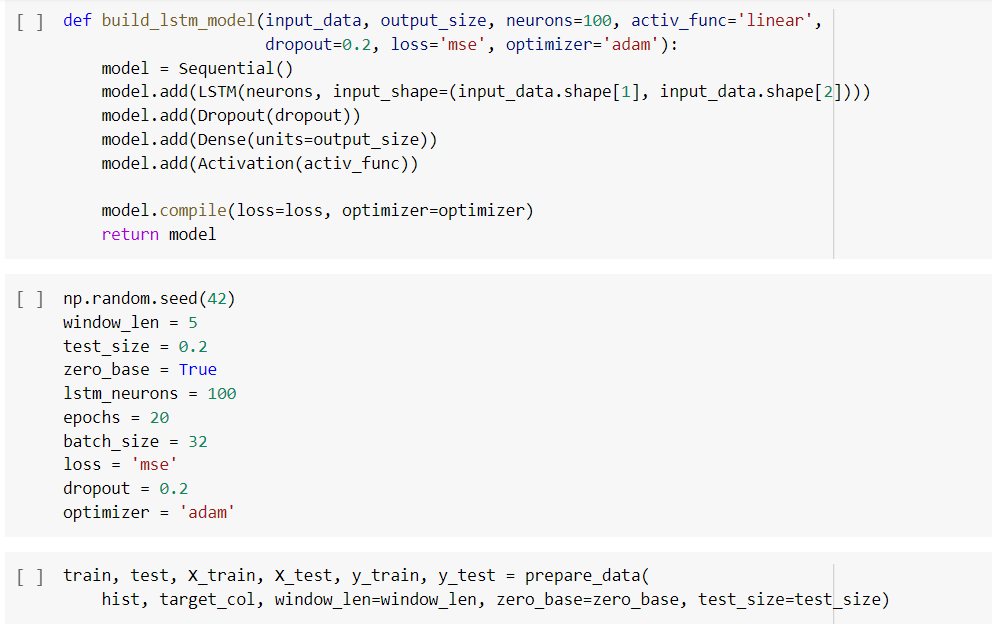
**2.6 Technical Code**

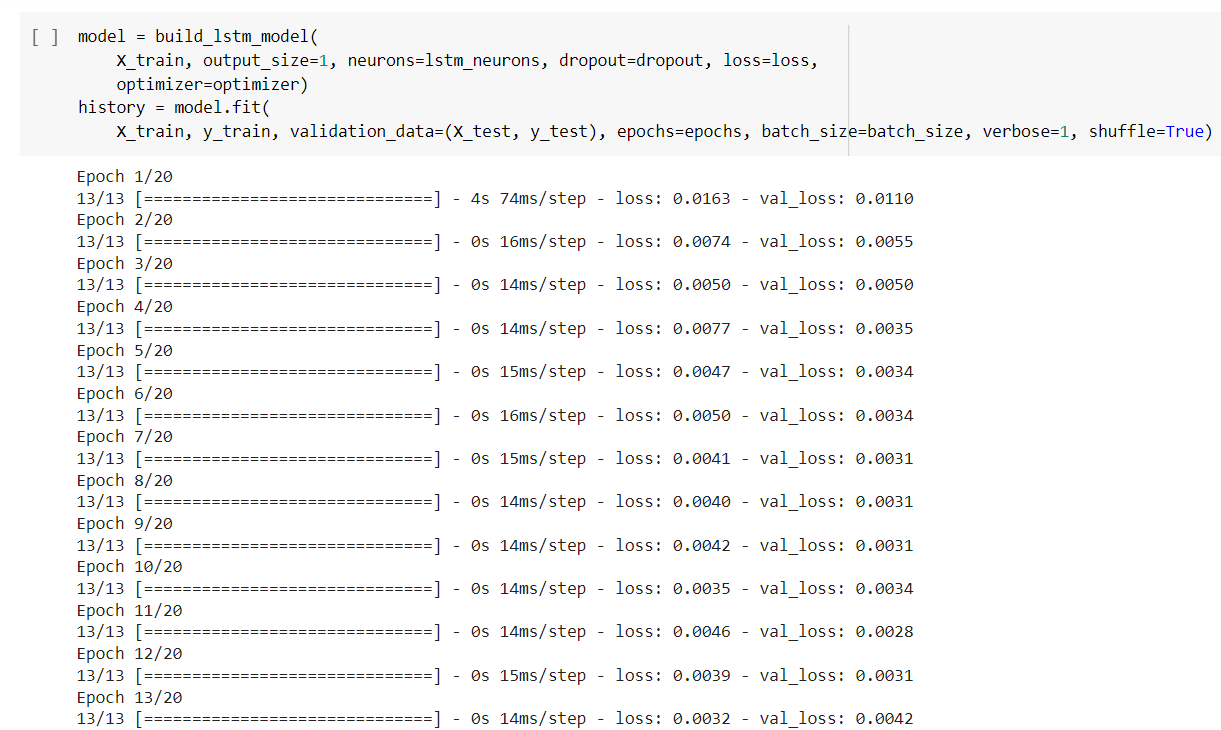
****

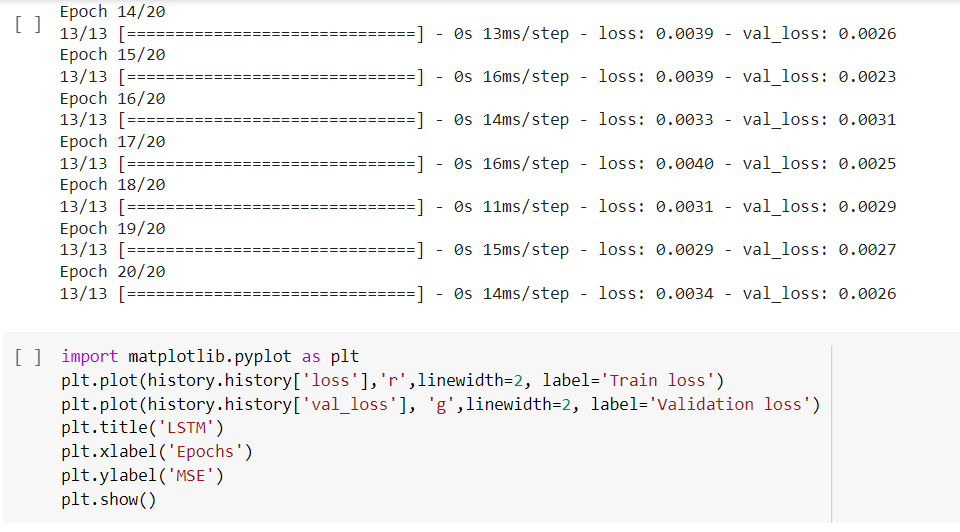
****

****

****

****

****

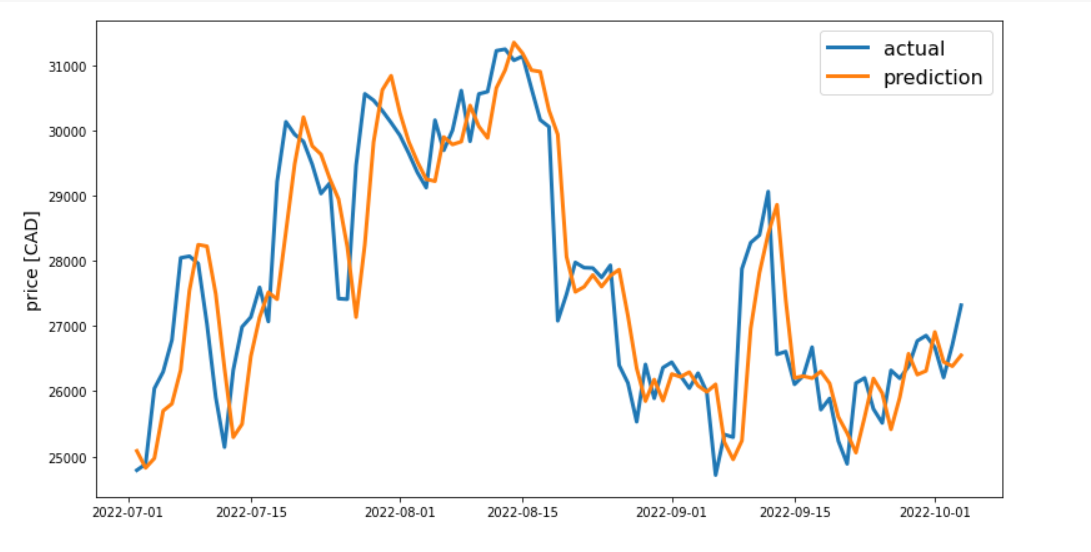
****

****

**Chapter 3**

**PROJECT OUTCOME AND APPLICABILITY**

**3.1 Code Outcomes**

****

**Result Tables:**

|  |  |
| --- | --- |
| **Parameters** | **Values** |
| **Seed** | **42** |
| **Window\_len** | **5** |
| **Test\_size** | **0.2** |
| **Zero\_base** | **True** |
| **Lstm\_neurons** | **100** |
| **epochs** | **35** |
| **Batch\_size** | **40** |
| **loss** | **‘mse’** |
| **dropout** | **0.2** |
| **optimizer** | **‘adam’** |

|  |  |
| --- | --- |
|  | **Value** |
| **MAE** | **0.02554982940364087** |
| **MSE** | **0.0011294702449779806** |
| **ACCURACY** | **0.7135908938100886** |

**3.2 Key Points about Code implementation**

Firstly, we have imported various libraries like NumPy and pandas for data preprocessing, matplotlib for expressing data in graphical format, sklearn for calculating accuracy of machine learning model. The we have taken the dataset from Cryptocompare.com which provide API keys. We generated and connected the API key to access the dataset in our code. Our Dataset is of approximately 1 Year 4 months. After that we divided the dataset into two parts training- 80 % and testing- 20 % expressing it in a graph.

we have tested and trained our data on LSTM (Long Short-Term Memory) model because it is most suitable for cryptocurrency prediction compared to other mode like GRU. Long short-term memory (LSTM) is a type of recurrent neural network (RNN) and powerful to model sequence data because it maintains an internal state to keep track of the data it has already seen. We have taken various parameters like batch size, activation function, drop loss, no of epochs, seeds etc. We have taken values of parameter which gives us the best accuracy for LSTM model i.e., seed-42, test size-0. 2, LSTM neurons -100, Epoches-35, batch size-40, dropout-0. 2, Optimizer-'adam'

We got our values Means absolute error -It measures the average magnitude of the errors in a set of predictions, without considering their direction. It’s the average over the test sample of the absolute differences between actual and predicted observations where all individual differences have equal weight.

I used Mean Absolute Error (MAE) as the evaluation metric. The reason behind choosing MAE over Root Mean Squared Error (RMSE) is that MAE is more interpretable. RMSE does not describe average error alone and hence is much more difficult to understand.

Mean square error-The Mean Squared Error measures how close a regression line is to a set of data points. It is a risk function corresponding to the expected value of the squared error loss. Mean square error is calculated by taking the average, specifically the mean, of errors squared from data as it relates to a function. The result we got is

**MAE- 0.02554982940364087**

**MSE- 0.0011294702449779806**

**Accuracy- 0.7135908938100886**

When we compared predicted prices from actual price using our testing dataset. We found it is showing less fluctuation means our prediction going in right direction.

We can improve our model accuracy as There is always space for improvement. The fuel of a neural network is data so we can build a more robust and accurate model by collecting more data. We can also try to adjust number of nodes in a layer or add additional LSTM layers. The things is that too deep in mind that it is not always good to increase the model accuracy because we may end up having an overfit model.

**Chapter 4**

**CONCLUSIONS AND RECOMMENDATION**

**4.1: OUTLINE**

This chapter of the report basically tells the reader that the report has come to an end. It also breaks down everything the report has discussed into more digestible chunks. It discusses the essential features and significant outcomes of the research. It sums up the key points of the discussion in the report, the essential features of the design or the significant outcomes of the investigation. Hence this chapter functions to round off the story of the project.

**4.2: EXISTING WORK WITH LIMITATIONS**

* There are some platforms available for this particular domain but their coin’s list is not dynamically kept up to date.
* Different features are available across various platforms which makes it uneasy for users to access the perfect choice to peruse.
* Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.
* UX / UI of those platforms are not user understandable.
* Collaborative filtering totally depends on human ratings. Sparsity (Insufficient data), Scalability, Cold start problem (low accuracy)

**4.4: INFERENCES**

The application has wide options of further enhancements which will make it more useful and efficient, without compromising the usability and the interactive GUI.

**CONCLUSION**

E-learning, also known as online learning, is expanding in the market during the past few years and has attracted great attentions due to the lockdowns caused by COVID-19. E-learning has become an important way to acquire knowledge not only for students in schools and universities but also for life-long learners who seek improvement in their social life and workplace. E-learning provides online courses and learning resources under various categories such as computer science, mathematics and business. For example, Udemy provides users with courses, degrees, programs and certificates from top universities such as Stanford and Harvard and companies such as Google and IBM. Faced with various courses and resources, users need to figure out which course to take to help develop their own career and keep competitive in the global market. Therefore, it is crucial to develop recommender systems to support users in choosing courses, resources, or learning materials in E-learning.

E-learning systems fit into the ever-challenging situation and provide learners with remote learning opportunities and abundant learning resources. Facing with the numerous resources online, users need support in deciding which course to take, thus recommender systems are applied in E-learning to provide learners with personalized services by automatically identifying their preferences. This report had systematically discussed the main recommendation techniques employed in in E-learning and identifies new research directions. Three main recommendation techniques are reviewed: content-based, collaborative filtering-based and knowledge-based recommendations. The basic mechanism of these technique together with how they are used to fulfill the specific requirements in the context of E-learning had been highlighted and presented.

The observations in this paper could support researchers and practitioners to better understand the current development and future directions of recommender systems in E-learning.

**REFERENCE**

1. R.K. Mohapatra, B. Majhi, S.K. Jena

Classification performance analysis of mnist dataset utilizing a multi-resolution technique

2015 International Conference on Computing, Communication and Security (ICCCS), IEEE (2015), pp. 1-5

1. B. Gope, S. Pande, N. Karale, S. Dharmale, P. Umekar

Handwritten digits identification using MNIST database via machine learning models

IOP Conference Series: Materials Science and Engineering, 1022, IOP Publishing (2021), Article 012108

1. R. Ebrahimzadeh, M. Jampour

Efficient handwritten digit recognition based on histogram of oriented gradients and SVM

International Journal of Computer Applications, 104 (9) (2014)

Keysers, D. (2007). Comparison and combination of state-of-the-art techniques for handwritten character recognition: topping the mnist benchmark. arXiv preprint arXiv:0710.2231.

1. A. Baldominos, Y. Saez, P. Isasi

A survey of handwritten character recognition with mnist and emnist

Applied Sciences, 9 (15) (2019), p. 3169

1. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner

Gradient-based learning applied to document recognition

Proceedings of the IEEE, 86 (11) (1998), pp. 2278-2324

MNIST dataset [Online] Available:

<http://yann.lecun.com/exdb/mnist/>.

1. S. Albawi, T.A. Mohammed, S. Al-Zawi

Understanding of a convolutional neural network

2017 international conference on engineering and technology (ICET), Ieee (2017)

1. P.Y. Simard, D. Steinkraus, J.C. Platt

Best practices for convolutional neural networks applied to visual document analysis

Icdar, 3 (2003)

1. D.C. Ciresan, U. Meier, J. Masci, L.M. Gambardella, J. Schmidhuber

Flexible, high performance convolutional neural networks for image classification

Twenty-second international joint conference on artificial intelligence (2011)

1. E. Bochinski, T. Senst, T. Sikora

Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms

2017 IEEE international conference on image processing (ICIP), IEEE (2017), pp. 3924-3928

1. L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, R. Fergus

Regularization of neural networks using dropconnect

International conference on machine learning, PMLR (2013), pp. 1058-1066

1. S.C. Wong, A. Gatt, V. Stamatescu, M.D. McDonnell

Understanding data augmentation for classification: when to warp?

2016 international conference on digital image computing: techniques and applications (DICTA), IEEE (2016)

1. Grim, J., & Somol, P. A Statistical Review of the MNIST Benchmark Data Problem.

<http://library.utia.cas.cz/separaty/2018/RO/grim-0497831.pdf>

1. D. Keysers, T. Deselaers, C. Gollan, H. Ney

Deformation models for image recognition

IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (8) (2007), pp. 1422-1435

1. W.C. Yeh et al.

### [**Simplified Swarm Optimization for Bi-Objection Active Reliability Redundancy Allocation Problems**](https://www.sciencedirect.com/science/article/pii/S1568494621002441)

### **Applied Soft Computing**

(2021)

15. W.C. Yeh

### [**A new harmonic continuous simplified swarm optimization**](https://www.sciencedirect.com/science/article/pii/S1568494619303242)

### **Applied Soft Computing**

(2019)

16. W.C. Yeh

### [**Solving cold-standby reliability redundancy allocation problems using a new swarm intelligence algorithm**](https://www.sciencedirect.com/science/article/pii/S156849461930362X)

### **Applied Soft Computing**

(2019)

17. W.C. Yeh

### [**A novel boundary swarm optimization method for reliability redundancy allocation problems**](https://www.sciencedirect.com/science/article/pii/S0951832016306767)

### **Reliability Engineering & System Safety**

(2019)

18. W.C. Yeh

### [**A novel node-based sequential implicit enumeration method for finding all d-MPs in a multistate flow network**](https://www.sciencedirect.com/science/article/pii/S0020025514010779)

### **Information Sciences**

(2015)

19. W.C. Yeh et al.

### [**Uniform parallel machine scheduling with resource consumption constraint**](https://www.sciencedirect.com/science/article/pii/S0307904X14004843)

### **Applied Mathematical Modelling**

(2015)

20. W.C. Yeh

### [**An improved simplified swarm optimization**](https://www.sciencedirect.com/science/article/pii/S0950705115000763)

### **Knowledge-Based Systems**

(2015)

21. W.C. Yeh

### [**Orthogonal simplified swarm optimization for the series–parallel redundancy allocation problem with a mix of components**](https://www.sciencedirect.com/science/article/pii/S095070511400094X)

### **Knowledge-Based Systems**

(2014)

22. W.C. Yeh et al.

### [**Economic-based resource allocation for reliable Grid-computing service based on Grid Bank**](https://www.sciencedirect.com/science/article/pii/S0167739X12000581)

### **Future Generation Computer Systems**

(2012)

23. W.C. Yeh

### [**Simplified swarm optimization in disassembly sequencing problems with learning effects**](https://www.sciencedirect.com/science/article/pii/S0305054811003157)

### **Computers & Operations Research**

(2012)