An Efficient Deep Learning Framework for the Recognition of House Numbers in Street View Imagery.

MD. TARIQUL ISLAM
Dept. of CSE
BRAC University
ID: 20301044
md.tariqul.islam3@g.bracu.ac.bd

MD.YASIN ARAFAT TAMIM
Dept. of CSE
BRAC University
ID: 20301029
yasin.arafat.tamim@g.bracu.ac.bd

Mohammad Tajwar Chowdhury
Dept. of CSE
BRAC University
ID: 20301080
mohammad.tajwar.chowdhury@g.bracu.ac.bd

Nabiha Tasnim Orchi
Dept. of CSE
BRAC University
ID: 20301148
nabiha.tasnim.orchi@g.bracu.ac.bd

Zohayer Bin Osman
Dept. of CSE
BRAC University
ID: 20301362
zohayer.bin.osman@g.bracu.ac.bd

Annajiat Alim Rasel
annajiat@bracu.ac.bd
BRAC University
Dhaka, Bangladesh

Shakib Mahmud Dipto
shakib.mahmud@bracu.ac.bd
BRAC University
Dhaka, Bangladesh

***Abstract*—In the expansive realm of computer vision, the accurate identification of house numbers within the intricate tapestry of street view images emerges as a paramount pursuit, resonating across a panorama of applications. This scholarly endeavor unfurls an intricately designed deep learning framework, meticulously tailored to navigate the nuances of this task. Through a calculated amalgamation of Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Multi-Layer Perceptron (MLP), our proposition fortifies itself against the headwinds of challenges like capricious lighting conditions, obfuscating impediments, and the kaleidoscope of idiosyncratic numerical styles that ensconce these numerals.The architectural blueprint of our framework is orchestrated with a symphony of orchestrated phases: the prelude of data preprocessing, the crescendo of judicious feature extraction, and the denouement of illuminating sequential learning. With finesse akin to a maestro wielding a baton, the CNN conducts an intricate symphony of feature extraction from the visual canvas of images, the RNN attuned to harmonize with the complex patterns interwoven in the numeric sequences, and the MLP assuming the mantle of the final arbiter, clasping the baton to conduct the decisive classification.**

***Index Terms*—computer vision, CNN, RNN, MLP, Accuracy, Street view image, Numeric Cognition.**

I. INTRODUCTION

For many applications, including navigation, urban planning, and address verification, house number identification in street view photography is crucial. In this work, we provide a comprehensive deep learning framework to tackle the difficult issue of home number recognition from street view photos using convolutional neural networks (CNNs), recurrent neural networks (RNNs), and multi-layer perceptrons (MLPs). High accuracy is what our suggested framework aims for, but efficiency in terms of processing resources and training time is also taken into account.In order to improve the generalization abilities of the model, we start by preprocessing the dataset of street view photos. The photos in the dataset of house numbers vary in scale, orientation, illumination, and occlusion to replicate real-world circumstances.Our approach uses the strength of CNNs to automatically learn key features, capturing local patterns and structures in the images, to address these issues.We provide a unique method that makes use of recurrent neural networks (RNNs) in order to capture sequential information contained in the house numbers. The accuracy of recognition is increased by exploiting the temporal relationships among the digits in a house number by considering them as a sequence. The gated recurrent units (GRUs) in our suggested RNN architecture effectively capture long-range dependencies while reducing the vanishing gradient issue.We also incorporate Multi-Layer Perceptrons (MLPs) to enable end-to-end learning, linking the RNN's sequential patterns and the CNN's learnt features. The outcomes demonstrate how well CNNs, RNNs, and MLPs work together to solve the problems presented by various street view imagery.In conclusion, our study introduces a novel deep learning architecture that ensures efficiency in terms of model size and training time while also achieving the highest level of house number identification accuracy.Our research introduces an encompassing deep learning framework, uniting CNNs, RNNs, and MLPs, with the objective of enhancing accuracy and efficiency for house number identification. This innovation holds paramount importance across applications like navigation and urban

Identify applicable funding agency here. If none, delete this.

planning, contributing substantially to the realm of computer vision.Lastly, this work significantly contributes to various applications by addressing house number identification in street view images, emphasizing its importance in computer vision and beyond.

II. Literature Review

Haoqi et al. [1] introduce an innovative approach that employs a convolutional neural network (CNN) for automatic recognition of house numbers within Street View images. With training conducted on the Street View House Numbers (SVHN) dataset, the CNN achieves an impressive accuracy rate of 92.32 percent, surpassing conventional methods. However, it's important to note that the evaluation is confined to the SVHN dataset, and the computational demands of the method could potentially limit its application in real-time scenarios. Nevertheless, this study stands as a notable stride forward in the realm of street view house number recognition.

The study delves into how street view image sampling algorithms impact tasks like predicting urban changes and estimating socioeconomic conditions. Zang et al. [2] propose DAS (Denoising and Adaptive Sampling), a novel algorithm composed of denoising and adaptive modules, designed to bolster the quality of sampling. Through assessments on extensive datasets, DAS consistently outperforms baseline methods in forecasting urban dynamics (e.g., achieving an impressive 85.2 percent accuracy in commercial activeness prediction). Despite some limitations with large datasets and task scope, DAS shines due to its efficiency, robustness, and practicality in enhancing street view image sampling for urban predictions. This research constitutes a valuable and notable contribution to the field.

Goodfellow et al. [3] introduces a deep convolutional neural network (CNN) designed for recognizing multi-digit numbers within Street View images. Notably, the CNN seamlessly integrates localization, segmentation, and recognition processes into a singular, efficient framework. Through evaluation on the SVHN dataset, the CNN achieves an impressive accuracy surpassing 96 percent, outperforming its predecessors. The novel approach provides a unified solution to the intricate challenge of number recognition, adeptly addressing variations in font, lighting, and occlusion commonly encountered in Street View imagery. The authors underscore the potency of deep CNNs, showcasing the remarkable performance of an eleven-layer architecture. While the approach exhibits strengths in applications like address validation and self-driving vehicles, it does come with certain limitations such as potential vulnerabilities to extreme distortions and challenges in adapting to diverse datasets. Overall, this research marks a substantial stride in the realm of multi-digit recognition using deep CNNs, offering significant implications for computer vision applications.

The article by Luan et al. [4] addresses several strategies for improving feature representations in convolutional neural networks (CNNs) to make them more resistant to transformations such as rotations and deformations. Data augmentation, spatial transformer networks (STN), oriented response networks (ORN), deformable convolutional networks, scattering networks, and the integration of Gabor filters are among these approaches.

Although data augmentation is useful, it necessitates a high number of factors. This is reduced via TI-Pooling, albeit at a computational expense. ORN employs actively rotating filters for orientation-sensitive features, whereas STN provides a spatial transformation module. The use of deformable CNNs improves transformation modeling. Structured receptive fields are used in scattering networks. The book presents a unique method for modulating learnt convolution filters using Gabor orientation filters (GoFs).This method enhances the resilience of feature representations to size and orientation alterations. In contrast to prior efforts that used Gabor filters as initialization or input layers, this method incorporates GoFs inside the convolutional layers, adjusting filter weights during back-propagation optimization.

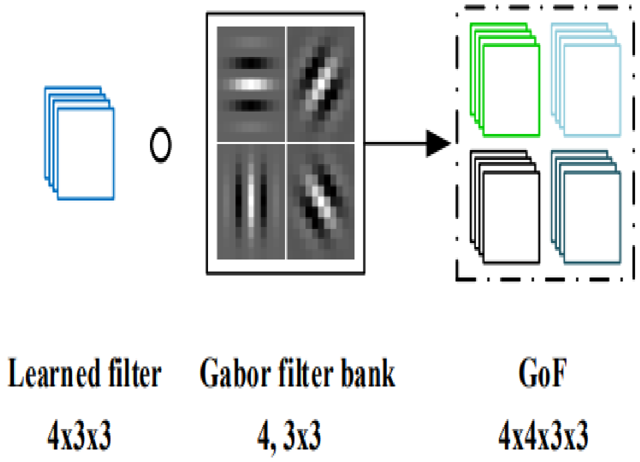


Figure 1.1: Modulation process of GoFs.

Sermanet et al. [5] introduces a Convolutional Neural Network (CNN) designed for digit classification using the SVHN dataset, achieving an impressive 94.85 percent accuracy. The architecture incorporates two convolutional and fully connected layers, outperforming previous methods. The CNN effectively processes 32x32x3 input images, learning intricate features through filters and employing ReLU activations. However, the model entails substantial computational demands, relies heavily on expansive labeled datasets, and exhibits vulnerability to noise. In essence,

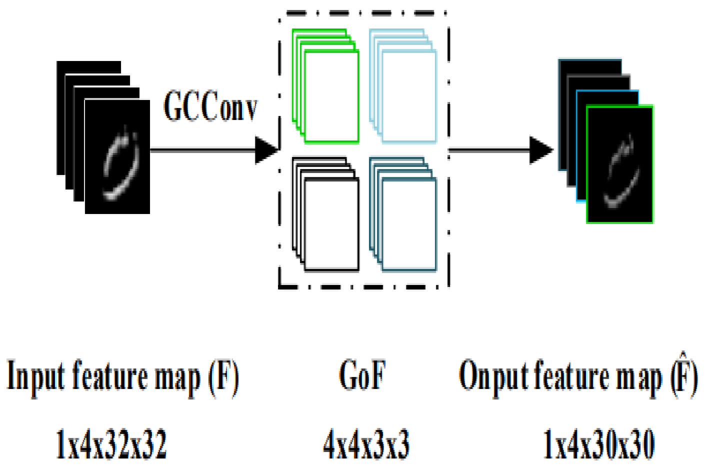


Figure 1.2: GCN evolution with 4 channel.

this study exemplifies the potential prowess of CNNs in digit classification tasks, showcasing remarkable accuracy advancements on the SVHN dataset.

Ringland et al. [6] addresses the constraints of traditional remote sensing techniques for mapping food cultivation and suggests Google Street View imagery as a cost-effective alternative. Using a modified VGGNet architecture, a CNN was trained on a dataset of hand-labeled Thai crop images, achieving an 83.3 percent accuracy in identifying crops. Evaluation on Street View images yielded a 99 percent accuracy for highly confident predictions. Despite limited dataset size and regional testing, this approach presents an economical solution for large-scale food cultivation mapping through deep learning and Street View images. Nevertheless, further research is necessary to enhance model accuracy and its applicability to diverse regions.

Wang et al. [7] introduces EHDCS-Net, a framework based on deep learning for reconstructing high-definition images from a reduced number of measurements, yielding a mean PSNR of 35.6dB on construction site images. Comprising four subnets (sampling, initial recovery, deep recovery body, and recovery head), it outperforms existing methods. While initially intended for construction site monitoring, EHDCS-Net's applicability extends to medical and satellite imaging. It leverages block-based compressed sensing and convolutional neural networks (CNNs) with variable layers for efficiency. Despite ongoing development and uncertainties about its performance on diverse image types, EHDCS-Net demonstrates significant promise as an accurate and adaptable solution.

Ullah et al. [8] delves into the challenges of fruit recognition, offering a dual-phase framework involving fruit detection and classification through a deep CNN. The approach, trained on a dataset of 10,000 images spanning 20 categories, attains a remarkable 96 percent accuracy, surpassing the 80-90 percent range of previous methods. Utilizing a customized VGG16 architecture, the CNN's weights are initialized with the Xavier method. Fruit detection employs a sliding window approach. The method proves efficient, implementation-friendly, and resource-efficient, yet limitations encompass singular dataset training and vulnerability to real-world complexities. To widen its utility, these limitations necessitate future resolution.

The paper ”DLBench: a comprehensive experimental evaluation of deep learning frameworks” by Elshwai et al. [9] extensively explores six major deep learning frameworks: TensorFlow, MXNet, PyTorch, Theano, Chainer, and Keras. The study evaluates their performance on CNNs, Faster R-CNNs, and LSTMs across CPU and GPU setups using diverse datasets. While TensorFlow and PyTorch excel but are resource-intensive, MXNet and Chainer are resource-efficient with slightly reduced accuracy. Keras, though user-friendly, is less efficient. The study emphasizes that framework choice depends on task-specific requirements and suggests broader benchmarking for future studies. The paper doesn't highlight any particular algorithm or model.

Shorten et al. [10] offers an extensive overview of image data augmentation techniques within the realm of deep learning. It explores a variety of augmentation strategies, such as geometric transformations, color space alterations, kernel filters, and more. The study underscores the significance of data augmentation, highlights challenges, and proposes future research avenues. Key findings encompass the positive impact of augmentation on enhancing model accuracy and the importance of selecting appropriate augmentation methods tailored to specific tasks. However, the paper lacks detailed accuracy analyses and in-depth discussions on practical implementation difficulties and ethical considerations.

The paper ”Comprehensive Analysis of Data Scarcity in Deep Learning” by Bansal et al. [11] provides a thorough examination of the data scarcity challenge in deep learning. The study underscores data's importance in this field and surveys diverse techniques to tackle data scarcity, such as data augmentation, transfer learning, semi-supervised learning, generative adversarial networks (GANs), and data fusion. The study also addresses challenges and future research directions. Key takeaways include the array of available techniques, the importance of tailored approaches, and ongoing efforts to enhance the effectiveness of data scarcity solutions. However, the paper lacks in-depth accuracy evaluations, practical implementation insights, and considerations of ethical implications.

Abu et al. [12] investigates the efficacy of combining deep learning and TensorFlow for image classification through the CIFAR-10 dataset. The study compares diverse deep learning models, emphasizing CNNs' superiority over RNNs, with a VGGNet achieving a remarkable accuracy of 93.2 percent. The challenges of limited data and computational resources are explored. Despite a lack of in-depth analysis and scalability exploration, the paper underscores the potential of deep learning in image classification. Key takeaways encompass the significance of tailored model choice, the prominence of CNNs, and TensorFlow's role in streamlining model development.

Sharma et al. [13] proposes a CNN-based method to differentiate between photographs and paintings. The VGG16 architecture is adapted with dropout and batch normalization layers for better performance. Tested on a dataset of 10,000 images (photographs and paintings), the CNN attained an impressive 96.4 percent accuracy. The research underscores the challenge of distinguishing visually similar image types and suggests future improvements involving larger datasets and parameter tuning.

The study by Sun et al. [14] introduces a deep learning technique employing Faster R-CNN for identifying urban architectural styles in street view images. The methodology encompasses constructing a dataset of 100,000 images from Wuhan, China. Results showcase a precision of 57.8 percent, recall rate of 80.91 percent, and F1 score of 0.634. Nevertheless, computational expenses and addressing images outside the training set present limitations. The approach displays promise in aiding urban planning and design applications.

The study by Yang et al. [15] introduces a novel deep learning approach using Faster R-CNN for the identification of external air conditioner units (EACUs) within street view images. Trained on a dataset of 10,000 images from Lahore, Pakistan, the model achieves an impressive precision of 86.7 percent, a recall rate of 83.1 percent, and an F1 score of 84.9 percent. Despite its effectiveness, the approach does have limitations, such as challenges in detecting partially occluded EACUs and computational resource demands. Nevertheless, the method demonstrates its potential to provide valuable assistance in urban planning and design applications, highlighting its applicability in practical scenarios.

Chen et al. [16] proposed a brand-new approach to estimating building age using Google Street View pictures. The authors employ a Support Vector Regression (SVR) model to determine the age of the building and a Convolutional Neural Network (CNN) to extract characteristics from the photos. The suggested technique produces an average inaccuracy of 9.8 years when tested against a dataset of 1,000 buildings in Melbourne, Australia. The authors utilize an SVR model to determine the age of the structure and a DenseNet161 CNN architecture to extract features from the photos. Additionally, they compare the performance of a pre-trained VGG16 model for feature extraction to that of a DenseNet161 model.The suggested approach has certain drawbacks, notably the fact that it depends on Google Street View photos, which might not be available everywhere. The technique also makes the somewhat unreliable assumption that the building's age may be inferred from its look. Finally, the method's performance is assessed using a very limited dataset of Melbourne buildings, and it may perform differently for structures in other cities.

III. Proposed Methodologies

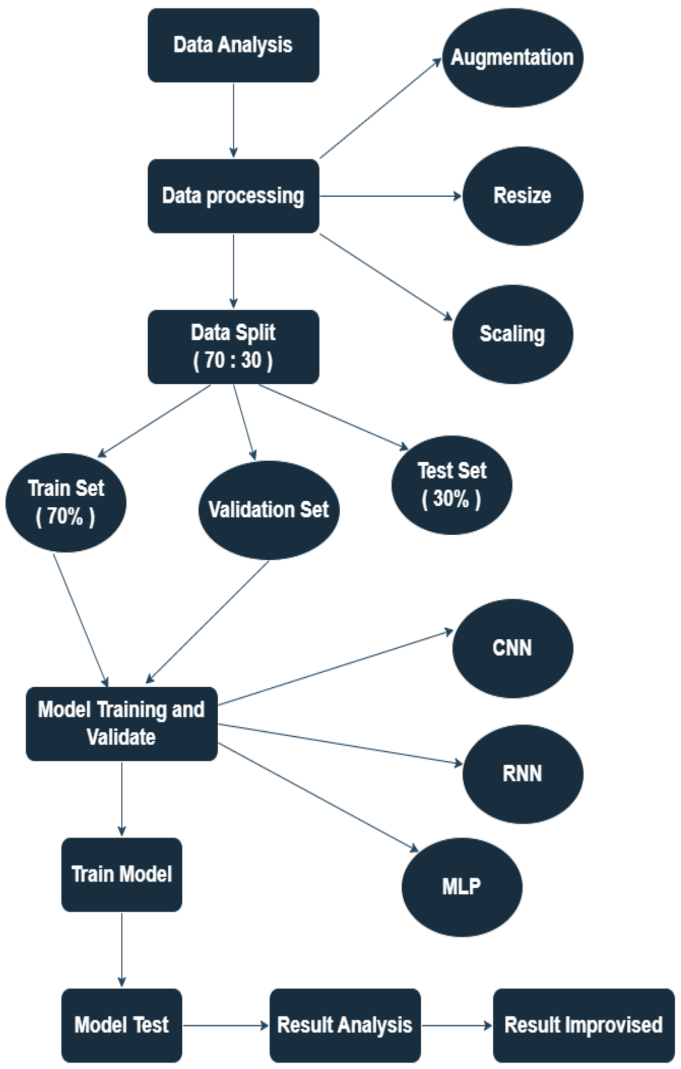


Figure 2: Proposed Methodologies

IV. Model Explanation

It is essential to correctly identify house numbers in street view images since it may be used for navigation, address verification, and urban planning. With the help of Convolutional Neural Networks (CNN), Multi-Layer Perceptrons (MLP), and Recurrent Neural Networks (RNN), we present a practical deep learning framework in this project to tackle this problem.

A. *Convolutional Neural Networks (CNN)*

Convolutional Neural Network (CNN) is a special type of neural network designed to process and analyse grid-like data such as images, audio spectrum images, and 2D signals.

(i) Convolutional layer: Convolutional layer. The main building block of a CNN is the convolutional layer. It manipulates input data using learnable filters (also called kernels) to find specific features. The convolution process involves intersecting the input data with these filters and calculating element-wise products and sums to create feature maps Each filter captures different features such as edges, textures, or patterns that become more complex as we move to the grid in the depths.

(ii) Functional Activities: After the convolution operation, an element-wise activation function (usually ReLU - Rectified Linear Unit) is applied to introduce nonlinearity into the network. This helps to capture the complex relationships between different parts of the network[17].

(iii) Pooling layer (subsampling): Pooling layers downsample the spatial dimensions of the feature maps, reducing the computational load and making the network more robust to interpretation and distortions in the input Max pooling is a common method where the maximum remaining value is thrown in a small window some of the fall.

(iv) Multiple convolutional layers: CNN architectures typically consist of multiple convolutional layers spaced with activation functions and pooling layers. A deeper layer captures more abstract and high-level features while integrating lower-level features from the previous layer.

(v) Fully connected layers: In addition to several layers of tectonic pooling, it often ends with one or more fully connected layers (also called a denselayer). These layers process objects learned from previous layers, producing the final classification or regression output[18].

(vi)Flattening: Before the layers are fully merged, the 3D feature maps are flattened into a 1D vector that can be filled between thicker layers. This is done to convert spatial information into a form that can be handled by traditional neural network layers.

(vii) Softmax Layer (for classification): When CNN is used for the classification task, a softmax layer is added at the end. It converts the raw outputs of the previous layers into probabilities for each class, enabling the network to make class predictions[19].

(viii) Training and Backpropagation: CNNs are trained by backpropagation, the process of feeding the network labelled training data and adjusting the network parameters (weights and biases) to reduce the default loss function Gradient-descent optimization algorithms Used to iteratively update parameters the.

(ix) Weight Sharing and Spatial Hierarchies: One of the main strengths of CNN is weight sharing. Filters are applied to the entire input image, allowing the network to detect the same object regardless of location. This property helps CNN learn the spatial structure of features, which is important for tasks such as image recognition[20].

CNNs have revolutionised computer vision and image analysis, enabling machines to recognize and extract meaningful patterns from visual data, making them increasingly effective tools for image-related tasks around.

B. *Multi-Layer Perceptrons (MLP)*

Multilayer perceptron (MLP) is a kind of feedforward neural network with many layers consisting of nodes (perceptrons) organised in input layer, one or more hidden layers, and output layers MLP is the main architecture in deep learning in and Used for a wide range of tasks including classification, regression and more complex pattern recognition.

(i) Input Layer: The input layer of MLP consists of nodes (neurons) representing the components of the input data. Each node corresponds to a specific object, and the number of nodes in this level is determined by the size of the embedded data.

(ii) Hidden Layers: MLPs can have one or more hidden layers sandwiched between input and output layers. Hidden layers are used to learn and convert objects extracted from input data into higher-level representations useful for the task at hand, each node in the hidden layer is connected to all nodes in the previous layer and all nodes of the subsequent layer[21].

(iii) Neurons (Nodes) and Activation Functions: In MLP, each node uses its weighted input values, followed by an activation function. This functional function introduces nonlinearity into the model, enabling it to capture complex relationships in the data. Common functions include the sigmoid, the hyperbolic tangent (tanh), and the Rectified Linear Unit (ReLU).

(iv) Weighted Sum and Activation: At each node, a weighted number of input values (including the bias term) are calculated. This sum is then passed through the activation function to produce the output of the node. The output acts as input to the nodes of the next layer[22].

(v) Output Layer: The output layer generates the final predictions or outputs of the network. The wide variety of nodes within the output layer depends on the venture. For example, in a binary type trouble, there is probably one node representing the chance of one class, at the same time as in a multi-class classification trouble, each node ought to represent the opportunity of an exclusive magnificence[23].

(vi) Loss Function: The loss characteristic quantifies the difference among the network's predictions and the floor fact labels. The intention at some point of training is to limit this loss, typically the use of optimization algorithms like gradient descent or its editions.

(vii) Backpropagation: MLPs are educated in the usage of backpropagation, a technique that includes computing gradients of the loss with recognition of the network's parameters (weights and biases) and adjusting these parameters to decrease the loss. The gradients are calculated thru the chain rule, and optimization algorithms replace the parameters iteratively[24].

(viii) Activation Functions in Hidden Layers: Activation features like ReLU, tanh, and sigmoid play an important role within the getting to know procedure. ReLU is favoured in lots of cases due to its computational efficiency and the capability to mitigate the vanishing gradient hassle.

(ix) Regularisation Techniques: To prevent overfitting, regularisation strategies like dropout, weight decay, and batch normalisation may be carried out to the MLP. These strategies help enhance the generalisation potential of the network with the aid of reducing the risk of fitting noise in the training facts[25].

(x) Universal Approximation Theorem: One of the key theoretical insights approximately MLPs is the Universal Approximation Theorem, which states that a feedforward neural community with a single hidden layer containing a finite wide variety of nodes can approximate any continuous feature to a desired diploma of accuracy given the ideal parameters[26].

MLPs function as the inspiration for lots of deep learning models and architectures. While they have got boundaries in coping with sequential or spatial facts because of their feedforward nature, they remain effective tools for solving a huge range of machine gaining knowledge of obligations, in particular while mixed with different techniques like regularisation, optimization, and more superior architectures.

C. *Recurrent Neural Networks (RNN)*

Recurrent neural networks (RNN) are a type of neural network architecture designed to handle sequential data by introducing the concept of memory or state persistence RNNs are particularly well suited for tasks involving sequential data or time series, such as natural language processing, speech recognition, video analysis, and so on.

(i) Sequential Processing: Unlike traditional feedforward neural networks that process data in fixed-size batches or individual instances, RNNs process a sequence of data step by step with each step corresponding to a specific element in the sequence (e.g., a word). some in the sentence).

(ii) Recurrent Connections: The main modification of RNNs is the introduction of recurrent links, which allow the network to hold a hidden state that carries information to different stages of the sequence. This hidden state acts as the memory of the network, and enables it to capture the time dependence and patterns in the data[27].

(iii) Hidden State Update: At each time step, the RNN takes the current input and combines it with the previous hidden state to compute a new hidden state. This process of innovation allows the network to take information about previous steps and use it to influence current forecasts or estimates.

(iv) Activation Function: Similar to different neural networks, an activation characteristic (frequently a non-linear feature like the hyperbolic tangent or ReLU) is applied to the aggregate of the enter and the previous hidden state. This introduces non-linearity into the model and lets it capture complex relationships in the statistics[28].

(v) Output Generation: Depending on the assignment, an RNN can generate an output at any step or handiest on the final time step. For instance, in language modelling, an RNN can generate the subsequent phrase in a sentence at each step.

(vi) Vanishing Gradient Problem: RNNs have a challenge referred to as the vanishing gradient trouble, in which gradients (used for education) can turn out to be very small as they're backpropagated through time. This issue can prevent the gaining knowledge of of long-range dependencies, limiting the RNN's capability to capture remote relationships in the sequence[29].

(vii) Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM): To cope with the vanishing gradient trouble and improve the studying of long-variety dependencies, greater advanced RNN architectures had been advanced, including Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) networks. These architectures incorporate specialised gating mechanisms that manipulate the glide of facts in and out of the hidden kingdom, permitting the network to selectively maintain or update records as wished.

(viii) Bidirectional RNNs: In some instances, it is crucial to remember each past and future context when making predictions at a given time step. Bidirectional RNNs method the sequence in both forward and backward directions, permitting the community to capture statistics from each past and destiny step.

(ix) Training and Backpropagation Through Time (BPTT): RNNs are educated in the usage of a way called Backpropagation Through Time (BPTT). This includes unfolding the RNN through the years steps and making use of popular backpropagation to compute gradients for updating the network's parameters[30].

RNNs have been tested to be effective models for dealing with sequential information because of their capability to seize temporal dependencies. However, they nonetheless have barriers in taking pictures of very long-range dependencies, which led to the improvement of more advanced architectures like GRUs and LSTMs. These improvements have made RNNs and their versions crucial equipment in diverse programs concerning sequential statistics analysis.

V. Data Analysis

The Street View House Numbers (SVHN) dataset is a real-world picture dataset developed by the Street View House Numbers (SVHN) project. The project's goal is to create ways for extracting home numbers from Google Street View pictures automatically. The file contains 600,000 32x32 RGB house number pictures. The photos were pre-processed and cropped from Google Street View photographs to eliminate the backdrop and other items. There are 73,257 training photos, 26,032 test images, and 53,113 additional images in the dataset. The additional photos are not utilized for training or testing, but rather for validation or data augmentation. The pictures' labels are the numerals they represent. There are ten classes, one for each of the digits 0 through 9.

Here, we splitted the data into 70:30 ratio, where 70 percent data is used for training and 30 percent data is used for testing.

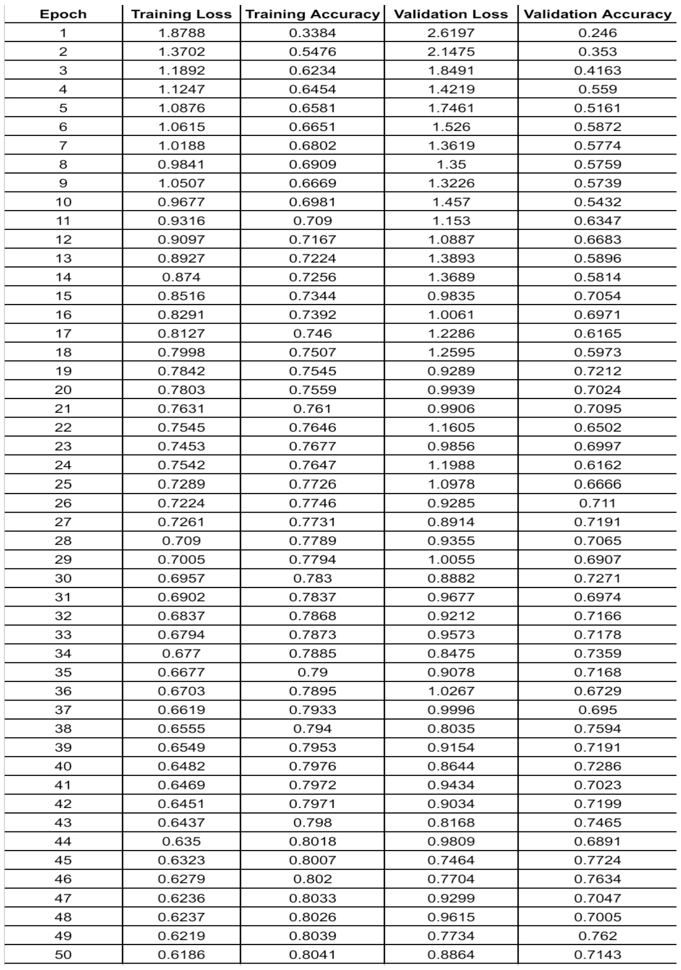
The dataset is unbalanced, with different numbers of photos for each type. The most popular class has 133,016 photographs, while the least common class has 89,990 images. The collection contains photos of varied quality. Some photos are crisp and well-centered, while others are hazy or contain various artifacts. The imbalance in the dataset is a concern since it might make training a model that generalizes effectively to new data challenging. The model may learn to overfit to the most common classes, making it incapable of recognizing less common classes. The fluctuating image quality is also an issue. The model may need to be able to learn to detect the digits in a variety of illumination situations, noise levels, and distortions.

The dataset's real-world character presents a problem since it has noise and other issues that synthetic datasets do not have. The model may need to learn to ignore noise and concentrate on characteristics that are crucial for categorization. Despite these issues, the SVHN dataset is a significant resource for training and assessing digit identification machine learning models. The dataset is broad and diverse, making it an excellent test for models developed to operate with real-world data. Convolutional neural networks (CNNs) for digit recognition have been trained and evaluated using the SVHN dataset. CNNs are a sort of deep learning model that excels at image categorization. The dataset has also been utilized to create novel data augmentation and picture preparation algorithms. Data augmentation is a strategy for increasing the amount and variety of a dataset. Image preprocessing is a technique for improving the quality of pictures in a dataset. The SVHN dataset was used to investigate the impact of noise and other difficulties on digit recognition performance. This research can aid in the development of models that are more resistant to these problems. The dataset has also been utilized to create new metric evaluations for digit recognition algorithms. Model performance is measured using evaluation measures. The creation of new assessment measures can assist us in better understanding the performance of various models. The SVHN dataset is an important resource for machine learning and computer vision research. The dataset is difficult, but it is also a useful tool for creating and testing new digit recognition systems.

VI. Result Analysis

A. *Multi-Layer Perceptrons (MLP)*

MLP Data table Explanation: The MLP model begins with a relatively significant training loss and improves steadily over epochs. Training accuracy is also progressively improving. Training loss and accuracy both show a positive trend. However, there are occasional changes in the validation loss and accuracy. Validation loss initially diminishes but begins to vary around epoch 20. Validation accuracy also varies about 0.87, indicating that the model is perhaps overfitting because it performs well on training data but less consistently on unseen validation data.



Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
1	1.8788	0.3384	2.6197	0.246
2	1.3702	0.5476	2.1475	0.353
3	1.1892	0.6234	1.8491	0.4163
4	1.1247	0.6454	1.4219	0.559
5	1.0876	0.6581	1.7461	0.5161
6	1.0615	0.6651	1.526	0.5872
7	1.0188	0.6802	1.3619	0.5774
8	0.9841	0.6909	1.35	0.5759
9	1.0507	0.6669	1.3226	0.5739
10	0.9677	0.6981	1.457	0.5432
11	0.9316	0.709	1.153	0.6347
12	0.9097	0.7167	1.0887	0.6683
13	0.8927	0.7224	1.3893	0.5896
14	0.874	0.7256	1.3689	0.5814
15	0.8516	0.7344	0.9835	0.7054
16	0.8291	0.7392	1.0061	0.6971
17	0.8127	0.746	1.2286	0.6165
18	0.7998	0.7507	1.2595	0.5973
19	0.7842	0.7545	0.9289	0.7212
20	0.7803	0.7559	0.9939	0.7024
21	0.7631	0.761	0.9906	0.7095
22	0.7545	0.7646	1.1605	0.6502
23	0.7453	0.7677	0.9856	0.6997
24	0.7542	0.7647	1.1988	0.6162
25	0.7289	0.7726	1.0978	0.6666
26	0.7224	0.7746	0.9285	0.711
27	0.7261	0.7731	0.8914	0.7191
28	0.709	0.7789	0.9355	0.7065
29	0.7005	0.7794	1.0055	0.6907
30	0.6957	0.783	0.8882	0.7271
31	0.6902	0.7837	0.9677	0.6974
32	0.6837	0.7868	0.9212	0.7166
33	0.6794	0.7873	0.9573	0.7178
34	0.677	0.7885	0.8475	0.7359
35	0.6677	0.79	0.9078	0.7168
36	0.6703	0.7895	1.0267	0.6729
37	0.6619	0.7933	0.9996	0.695
38	0.6555	0.794	0.8035	0.7594
39	0.6549	0.7953	0.9154	0.7191
40	0.6482	0.7976	0.8644	0.7286
41	0.6469	0.7972	0.9434	0.7023
42	0.6451	0.7971	0.9034	0.7199
43	0.6437	0.798	0.8168	0.7465
44	0.635	0.8018	0.9809	0.6891
45	0.6323	0.8007	0.7464	0.7724
46	0.6279	0.802	0.7704	0.7634
47	0.6236	0.8033	0.9299	0.7047
48	0.6237	0.8026	0.9615	0.7005
49	0.6219	0.8039	0.7734	0.762
50	0.6186	0.8041	0.8864	0.7143

Figure 3.1: Data table of MLP.

B. MLP Loss graph

Loss Graph Explanation : The training loss is shown by the blue line, while the validation loss is represented by the orange line. Over epochs, the training loss tends to decrease, showing that the model is learning and improving its predictions on the training data. The validation loss diminishes at first, indicating that the model generalizes well to new data. However, at a certain time, the validity loss begins to rise once more. This is known as ”overfitting,” and it occurs when the model gets overly fitted to the training data and performs poorly on validation data. To prevent overfitting, the time at which the validation loss begins to grow may be a suitable epoch to halt training or consider adding strategies such as early stopping.

C. MLP Accuracy

Accuracy Graph Explanation : The blue line denotes training accuracy, whereas the orange line denotes validation accuracy. Over epochs, the training accuracy rises, showing that the model is learning to make better predictions on the training

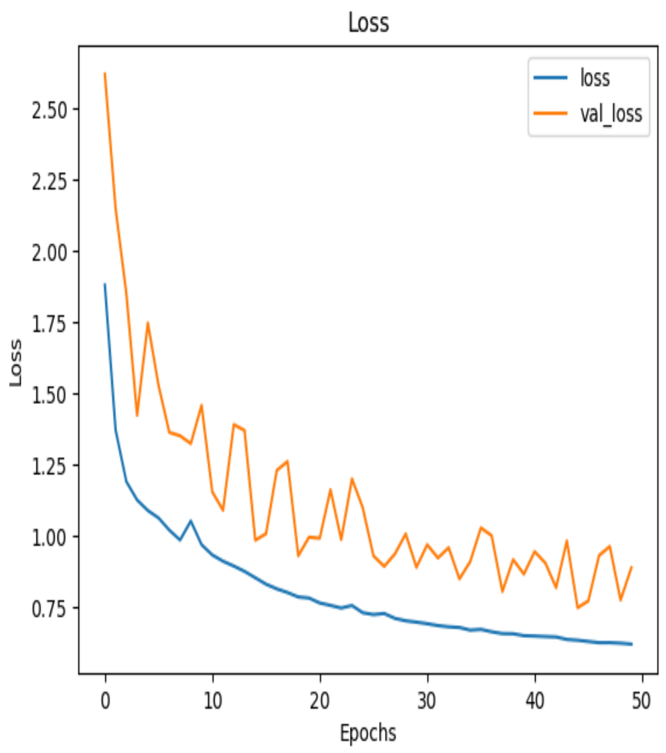


Figure 3.2: MLP Loss Graph

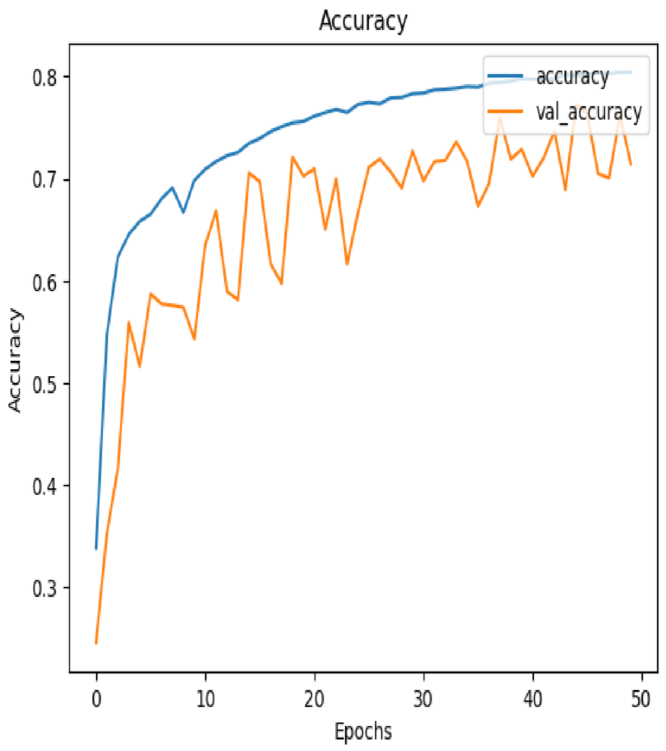


Figure 3.3: MLP Accuracy

data. The validation accuracy initially rises, indicating that the model generalizes effectively to previously unknown data. However, as training advances, validation accuracy plateaus or begins to decline. This is another hallmark of overfitting, in which the model becomes overly specialized to the training data. The point at which validation accuracy plateaus or drops might assist you determine whether to terminate training or use overfitting prevention strategies. Because the model is tuned to perform well on training data, the training accuracy is frequently greater than the validation accuracy.

D. *MLP result prediction*

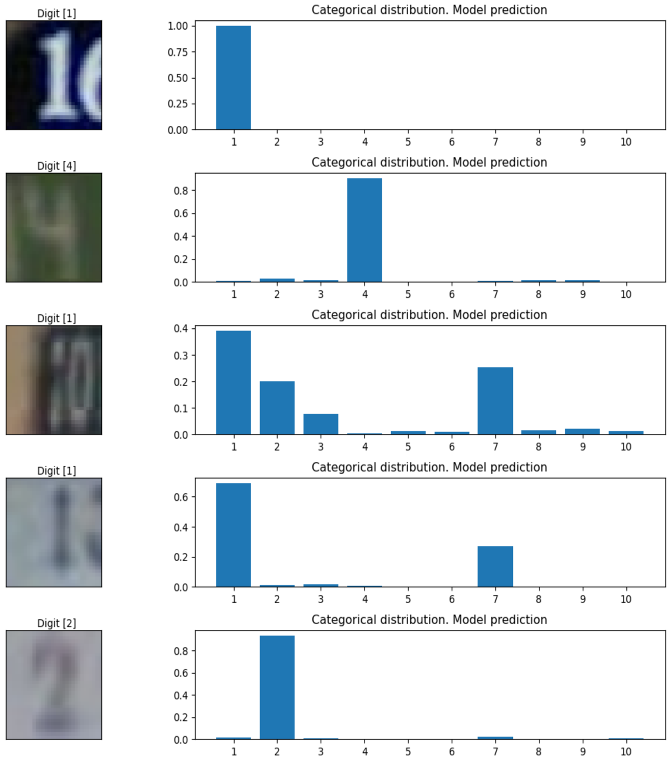
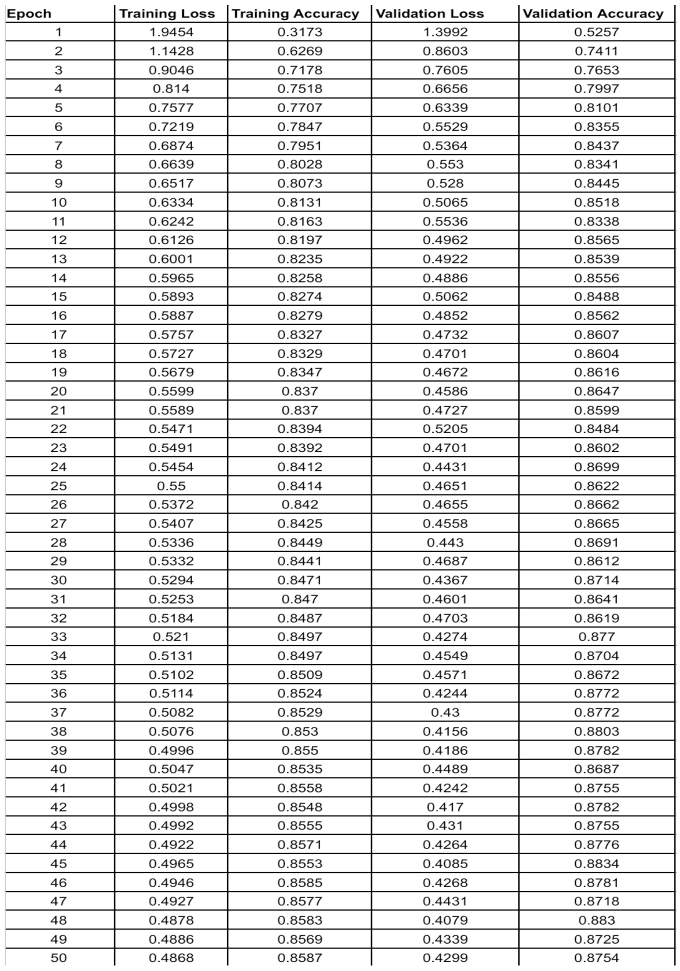


Figure 3.4: MLP result prediction

Here we can see our MLP model can perfectly guess 3 out of 5 figures almost every time and can partially guess the other 2 figures maximum of time

E. *Convolutional Neural Networks (CNN)*

CNN (Convolutional Neural Network): The CNN model also improves in terms of training loss and accuracy over time. Training loss is progressively decreasing, and training precision is improving as well. The validation loss follows a similar pattern, initially dropping and then bouncing around a lower amount. Validation accuracy eventually improves, reaching approximately 0.87. When with the MLP, there are indicators of overfitting when validation accuracy plateaus and validation loss begins to fluctuate.



Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
1	1.9454	0.3173	1.3992	0.5257
2	1.1428	0.6269	0.8603	0.7411
3	0.9046	0.7178	0.7605	0.7653
4	0.814	0.7518	0.6656	0.7997
5	0.7577	0.7707	0.6339	0.8101
6	0.7219	0.7847	0.5529	0.8355
7	0.6874	0.7951	0.5364	0.8437
8	0.6639	0.8028	0.553	0.8341
9	0.6517	0.8073	0.528	0.8445
10	0.6334	0.8131	0.5065	0.8518
11	0.6242	0.8163	0.5536	0.8338
12	0.6126	0.8197	0.4962	0.8565
13	0.6001	0.8235	0.4922	0.8539
14	0.5965	0.8258	0.4886	0.8556
15	0.5893	0.8274	0.5062	0.8488
16	0.5887	0.8279	0.4852	0.8562
17	0.5757	0.8327	0.4732	0.8607
18	0.5727	0.8329	0.4701	0.8604
19	0.5679	0.8347	0.4672	0.8616
20	0.5599	0.837	0.4586	0.8647
21	0.5589	0.837	0.4727	0.8599
22	0.5471	0.8394	0.5205	0.8484
23	0.5491	0.8392	0.4701	0.8602
24	0.5454	0.8412	0.4431	0.8699
25	0.55	0.8414	0.4651	0.8622
26	0.5372	0.842	0.4655	0.8662
27	0.5407	0.8425	0.4558	0.8665
28	0.5336	0.8449	0.443	0.8691
29	0.5332	0.8441	0.4687	0.8612
30	0.5294	0.8471	0.4367	0.8714
31	0.5253	0.847	0.4601	0.8641
32	0.5184	0.8487	0.4703	0.8619
33	0.521	0.8497	0.4274	0.877
34	0.5131	0.8497	0.4549	0.8704
35	0.5102	0.8509	0.4571	0.8672
36	0.5114	0.8524	0.4244	0.8772
37	0.5082	0.8529	0.43	0.8772
38	0.5076	0.853	0.4156	0.8803
39	0.4996	0.855	0.4186	0.8782
40	0.5047	0.8535	0.4489	0.8687
41	0.5021	0.8558	0.4242	0.8755
42	0.4998	0.8548	0.417	0.8782
43	0.4992	0.8555	0.431	0.8755
44	0.4922	0.8571	0.4264	0.8776
45	0.4965	0.8553	0.4085	0.8834
46	0.4946	0.8585	0.4268	0.8781
47	0.4927	0.8577	0.4431	0.8718
48	0.4878	0.8583	0.4079	0.883
49	0.4886	0.8569	0.4339	0.8725
50	0.4868	0.8587	0.4299	0.8754

Figure 4.1: Data table of CNN.

F. *CNN Loss Graph*

Loss Graph Explanation : The training loss is shown by the blue line, while the validation loss is represented by the orange line. Both training and validation losses are rather high at the start of training. As training advances, both the training and validation losses decrease, indicating that the model's performance on the data is increasing. The difference between the training and validation losses initially closes, indicating that the model is generalizing successfully. However, at epoch 10, the validation loss begins to rise significantly while the training loss falls. This is an indication of overfitting, in which the model becomes overly specialized to the training data and fails to perform well on unseen validation data. The moment at which the training loss continues to decline while the validation loss begins to climb is an important signal to watch for in order to avoid overfitting.

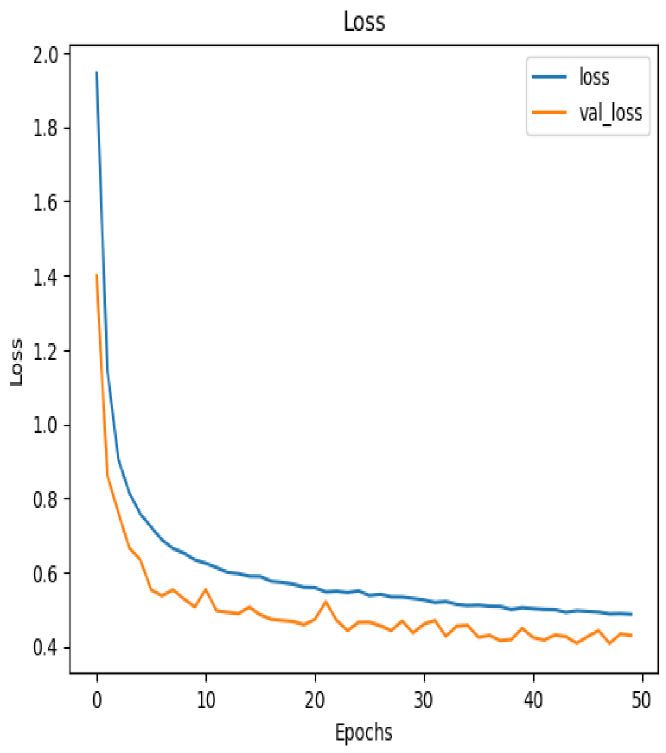


Figure 4.2: Loss Graph.

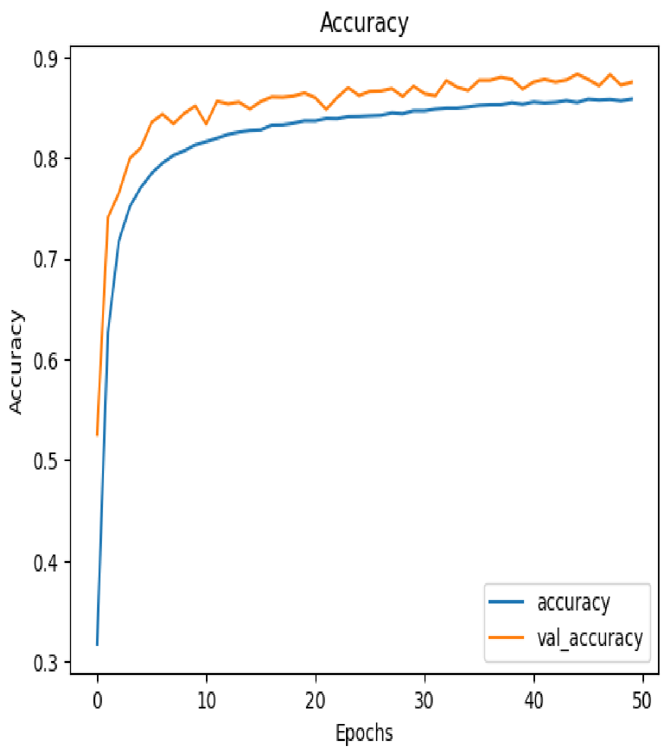


Figure 4.3: CNN Accuracy.

G. *CNN Accuracy*

Accuracy Graph Explanation : The blue line denotes training accuracy, whereas the orange line denotes validation accuracy. Similarly to the loss graph, both the training and validation accuracy numbers are quite low at the start of training. As training advances, both training and validation accuracy improve, showing that the model's ability to generate right predictions improves. There is a minor discrepancy between training and validation accuracy at epoch 10. The training accuracy increases while the validation accuracy plateaus or begins to decline. This behavior indicates that the model has overfitted to the training data. Monitoring the point when training accuracy continues to rise but validation accuracy plateaus or falls is critical for avoiding overfitting.

H. *CNN Result Prediction*

CNN result analysis: Here we can see our CNN model can perfectly guess 3 out of 5 figures almost every time and can partially guess the other 2 figures maximum of time.

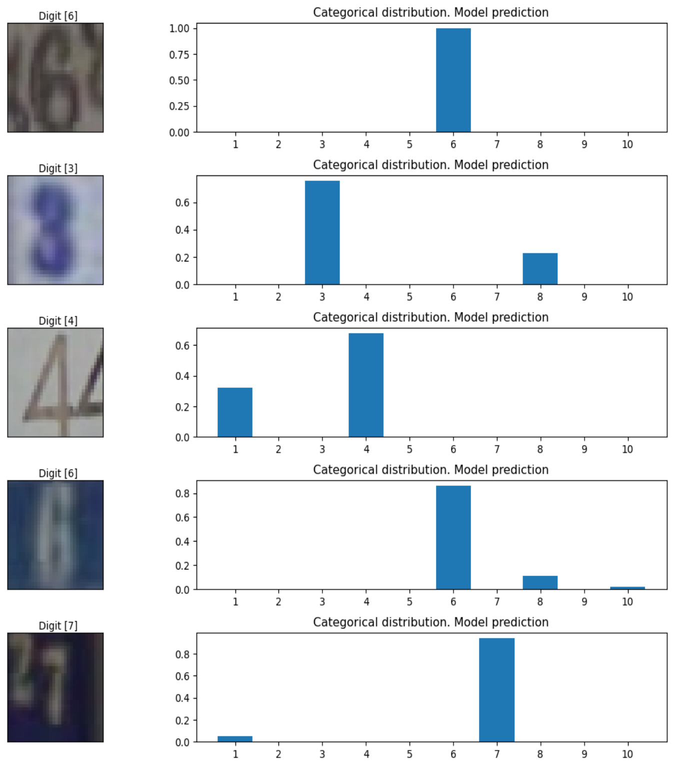
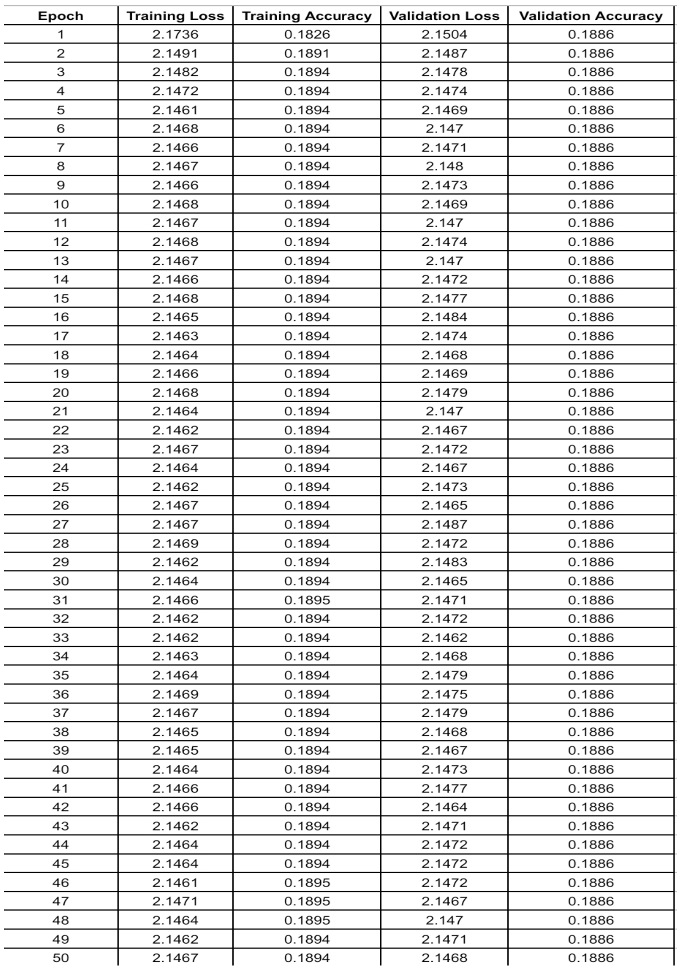


Figure 4.4: CNN Result Prediction.

I. *Recurrent Neural Networks (RNN)*



Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
1	2.1736	0.1826	2.1504	0.1886
2	2.1491	0.1891	2.1487	0.1886
3	2.1482	0.1894	2.1478	0.1886
4	2.1472	0.1894	2.1474	0.1886
5	2.1461	0.1894	2.1469	0.1886
6	2.1468	0.1894	2.147	0.1886
7	2.1466	0.1894	2.1471	0.1886
8	2.1467	0.1894	2.148	0.1886
9	2.1466	0.1894	2.1473	0.1886
10	2.1468	0.1894	2.1469	0.1886
11	2.1467	0.1894	2.147	0.1886
12	2.1468	0.1894	2.1474	0.1886
13	2.1467	0.1894	2.147	0.1886
14	2.1466	0.1894	2.1472	0.1886
15	2.1468	0.1894	2.1477	0.1886
16	2.1465	0.1894	2.1484	0.1886
17	2.1463	0.1894	2.1474	0.1886
18	2.1464	0.1894	2.1468	0.1886
19	2.1466	0.1894	2.1469	0.1886
20	2.1468	0.1894	2.1479	0.1886
21	2.1464	0.1894	2.147	0.1886
22	2.1462	0.1894	2.1467	0.1886
23	2.1467	0.1894	2.1472	0.1886
24	2.1464	0.1894	2.1467	0.1886
25	2.1462	0.1894	2.1473	0.1886
26	2.1467	0.1894	2.1465	0.1886
27	2.1467	0.1894	2.1487	0.1886
28	2.1469	0.1894	2.1472	0.1886
29	2.1462	0.1894	2.1483	0.1886
30	2.1464	0.1894	2.1465	0.1886
31	2.1466	0.1895	2.1471	0.1886
32	2.1462	0.1894	2.1472	0.1886
33	2.1462	0.1894	2.1462	0.1886
34	2.1463	0.1894	2.1468	0.1886
35	2.1464	0.1894	2.1479	0.1886
36	2.1469	0.1894	2.1475	0.1886
37	2.1467	0.1894	2.1479	0.1886
38	2.1465	0.1894	2.1468	0.1886
39	2.1465	0.1894	2.1467	0.1886
40	2.1464	0.1894	2.1473	0.1886
41	2.1466	0.1894	2.1477	0.1886
42	2.1466	0.1894	2.1464	0.1886
43	2.1462	0.1894	2.1471	0.1886
44	2.1464	0.1894	2.1472	0.1886
45	2.1464	0.1894	2.1472	0.1886
46	2.1461	0.1895	2.1472	0.1886
47	2.1471	0.1895	2.1467	0.1886
48	2.1464	0.1895	2.147	0.1886
49	2.1462	0.1894	2.1471	0.1886
50	2.1467	0.1894	2.1468	0.1886

Figure 5.1: Data table of RNN.

RNN (Recurrent Neural Network): The training loss and accuracy of the RNN model follow a similar trend to the other models, improving with time. Training loss is reduced, while training precision improves. The validation loss, on the other hand, remains essentially constant throughout epochs, and validation accuracy improves just little, lingering around 0.1886. In contrast to the other models, the RNN's performance does not increase considerably with time, suggesting that it may not be the optimal design for this task.

J. *Loss and Accuracy of RNN*

Loss and accuracy of RNN: The training loss is shown by the blue line, while the validation loss is represented by the orange line. Both the training and validation losses begin rather high. Both losses reduce steadily as training advances, although the total difference is quite minor. Throughout the training phase, the training and validation losses stay quite near to each other. The tiny difference between training and

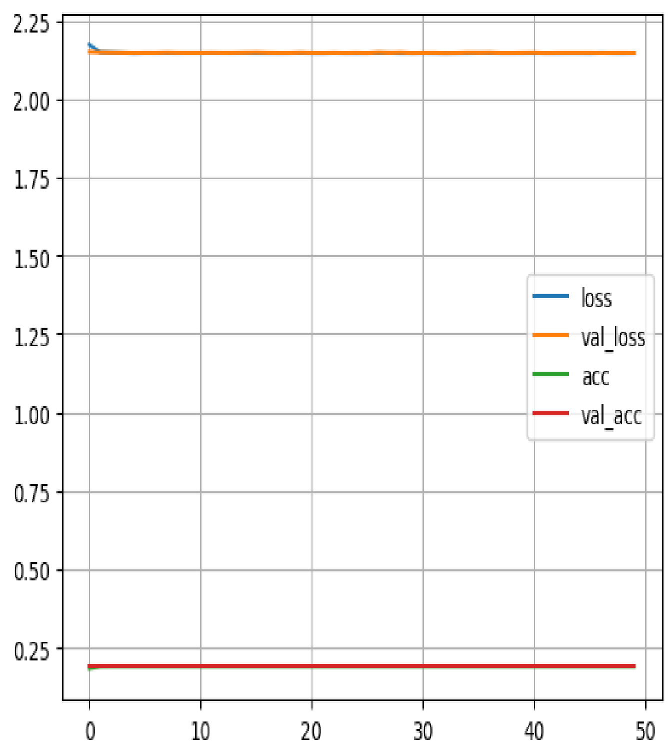


Figure 5.2: Loss and Accuracy of RNN.

validation loss suggests that the model is not overfitting, but it is also not learning adequately. The losses are excessive, and the model is not convergent to a satisfactory solution. Both the training accuracy (green) and the validation accuracy (red) begin at extremely low levels. Despite a minor rise in training accuracy, total accuracy remains exceedingly poor and nearly constant.The fact that the accuracy numbers scarcely fluctuate across epochs demonstrates the model's failure to learn. The little difference in training and validation accuracy indicates that the model is not overfitting, but it cannot capture the underlying patterns in the data.

K. *RNN Result Analysis*

RNN result analysis: Here we can see that our RNN model is quite confused and couldn't guess the right figure maximum of time. In Each iteration RNN model provides us a static loss as well as static accuracy. The accuracy remain same all the time, although we train our model with different random images. Compared to our other models RNN models accuracy way more low. So, RNN model can't be a good choice for this field. But for language recognition, music voice recognitionn and speech recognition RNN model is widely used.

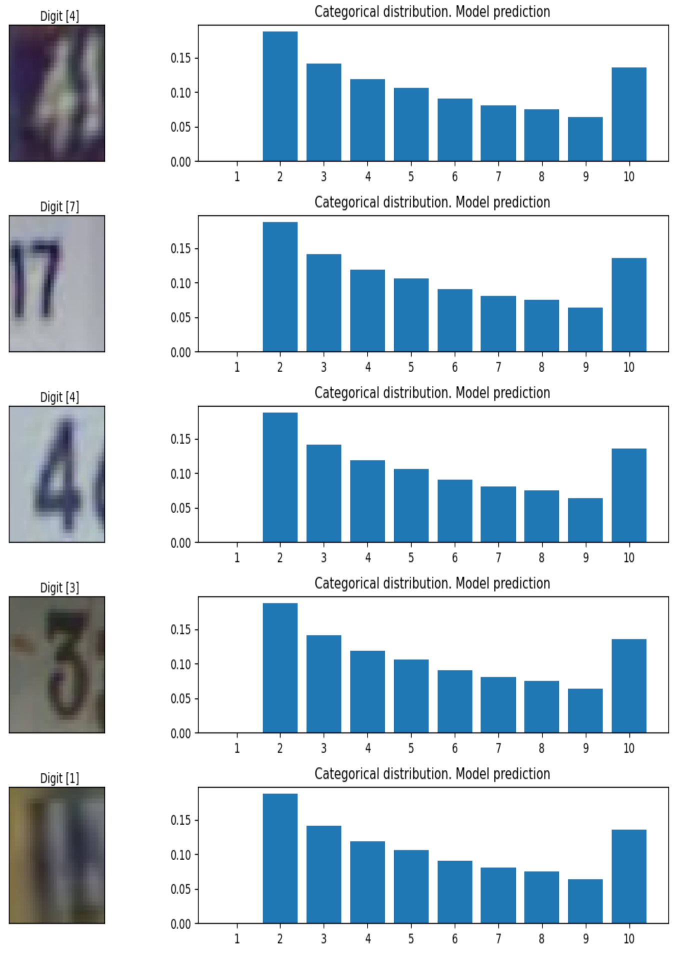


Figure 5.3: RNN result analysis

VII. Model Comparision

Comparison Explanation: Because they are randomly generated, all three models begin with relatively low accuracy and large loss values. As training advances (the number of epochs grows), the accuracy of all three models increases while the loss reduces. The MLP model improves at the slowest pace in both accuracy and loss. This might be due to MLPs' inability to grasp complicated spatial correlations or temporal dependencies in the data. In terms of accuracy and loss, the CNN model demonstrates quicker convergence. This is most likely owing to the convolutional layers' ability to successfully discern spatial patterns, which is critical for image-related tasks. The RNN model performs similarly to the MLP model, which is not optimal. RNNs are built to handle sequential data, thus it may be unsuitable for the current job, or the architecture and hyperparameters may need to be tweaked. In general, models with steeper accuracy curve slopes also have steeper loss curve slopes, indicating that they are learning more successfully. The CNN model appears to

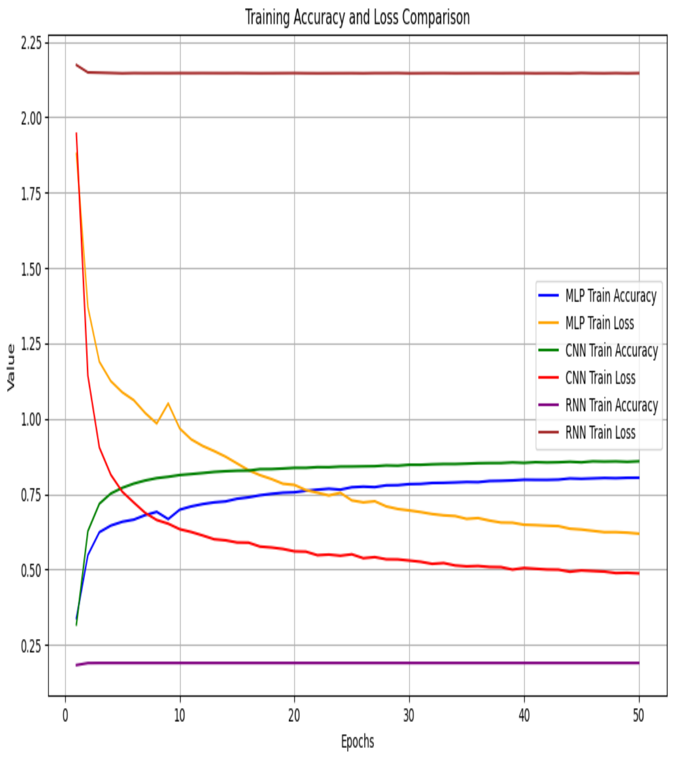


Figure 6: General Loss and accuracy comparision

perform the best of the three models, rapidly convergent to better accuracy and lower loss values. To perform better, the MLP and RNN models may require additional optimization, architectural tweaks, or hyperparameter tweaking.

Overall, all three models show an increase in training performance (lower loss, greater accuracy) over epochs.Validation performance, particularly accuracy, exhibits plateauing and volatility, implying overfitting.In terms of validation accuracy and loss, the MLP and CNN models outperform the RNN model, indicating that these models may be better suited for this dataset/task.Techniques such as dropout, regularization, and modifying model complexity might be investigated to alleviate overfitting.

A. *Comparison Analysis*

Here a comparison analysis of the three models (MLP, CNN, RNN) based on their performance.

MLP (Multi-Layer Perceptron):
Pros : Consistent improvement in training loss and accuracy throughout epochs. A rather high validation accuracy of roughly 0.875 is achieved, demonstrating strong generalization to previously unknown data. Simple architecture with fully linked layers facilitates interpretation and optimization.

Cons: Overfitting is evident when the validation accuracy plateaus and the validation loss begins to fluctuate. Spatial aspects in the data may not be captured.

CNN (Convolutional Neural Network) :
Pros : Over epochs, it improves both training and validation performance. It achieves a validation accuracy of roughly 0.875, which is comparable to the MLP. Because of its convolutional layers, which can record spatial hierarchies and patterns, it is suitable for picture data.
Cons: Shows evidence of overfitting when validation accuracy plateaus and validation loss fluctuates. Careful hyperparameter adjustment is required, particularly for the amount of convolutional layers and filters.

RNN (Recurrent Neural Network):
Pros : Increases training accuracy across epochs.
Cons: Shows poor performance on both training and validation data. The validation accuracy remains extremely low (around 0.1886) and doesn't improve significantly over epochs. Doesn't seem to capture temporal patterns effectively in this case. Could be inadequate for capturing the relationships and dependencies present in the data.

B. Overall Evaluation

Based on the facts and analyses presented, both the MLP and CNN models show promise, although neither is perfect for the given dataset. Both show evidence of overfitting, which might be corrected by using regularization approaches or adjusting model complexity.When comparing MLP and CNN validation accuracies, the CNN may have an edge due to its ability to capture spatial information inherent in picture data. However, considerations like as processing resources, model complexity, and simplicity of implementation should be considered while deciding between them.The RNN, on the other hand, appears to be a poor fit for this dataset, owing to its persistently low validation accuracy and failure to adequately capture the patterns in the data.To summarize, the CNN may be somewhat more suited than the MLP because to its ability to capture spatial characteristics, but additional testing and hyperparameter adjustment are required to make a clear conclusion.

VIII. CONCLUSION

The results of our thorough examination of the SVHN dataset using three different models—MLP, CNN, and RNN—have shed important light on the effectiveness of each model for image classification tasks. Our results highlight the importance of model choice in obtaining the best outcomes. The MLP performed moderately, demonstrating that it has trouble capturing intricate spatial connections in pictures, which accounts for its mediocre accuracy and loss metrics.

In contrast, the CNN performed best overall, proving the effectiveness of convolutional layers in extracting hierarchical information from pictures. The CNN demonstrated its extraordinary capacity to identify patterns within the SVHN dataset by achieving amazing accuracy with minimum loss. This demonstrates CNNs' appropriateness for picture-centric datasets and further establishes their position as the preferred algorithm for image classification applications.

Unfortunately, our research showed that the RNN was inappropriate for this specific dataset. The RNN's difficulty managing the spatial and sequential dependencies present in the SVHN dataset is highlighted by the large loss and relatively low accuracy that were seen with it. This highlights how crucial it is to select models that take into account the intrinsic properties of the dataset.

In conclusion, our experiment emphasises how crucial model selection is in machine learning initiatives. The RNN's poor performance highlights the necessity for careful attention when using particular model types, while the CNN's excellent performance demonstrates its prowess for image-related tasks. These results provide useful information for practitioners in choosing the best model architecture for particular dataset types as the field continues to develop, eventually fostering improvements in machine learning research and applications.

IX. FUTURE WORK PLAN

The focus of our project's future work will be on resolving the significant overfitting problems we ran across during our experiments with the SVHN dataset. Notably, the loss initially dropped as training continued for both the MLP and CNN models, but after a specific number of epochs owing to overfitting, the loss then showed a subsequent spike. We want to get into thorough hyperparameter tuning to tackle this problem. We seek to achieve a balance that encourages higher generalisation and minimises overfitting by methodically investigating and refining parameters like learning rates, batch sizes, and network depths. To prevent the models from memorising noise in the training data, regularisation methods like dropout and L2 regularisation must be used.

Furthermore, data-driven tactics to improve model robustness are a key component of our future work plan. An interesting approach is data augmentation, in which we want to add variations to the training set via transformations like rotation, cropping, and flipping. It is predicted that this method will expose the models to a larger range of examples, assisting in enhanced generalisation and thwarting inclinations toward overfitting. The use of ensemble techniques is a different direction we want to go. We want to capitalise on the complementary qualities of different models by combining their predictions, perhaps reducing overfitting and increasing overall accuracy.

An essential part of our future effort will be to address the RNN model's poor performance. Given the unsuitability of the RNN architecture for this dataset, we recognize the need for comprehensive data preprocessing. The secret to unlocking enhanced RNN performance may lie in implementing methods to address sequential dependencies, such as using more advanced sequence padding and feature scaling approaches. Another key tactic in our toolbox is dataset expansion. We want to take into account data augmentation for the RNN as well as the CNN and MLP, enhancing its ability to recognize important patterns and raising accuracy.

We also plan to explore more into domain-specific elements that can improve model comprehension and performance. Due to the fact that our dataset relates to home numbers, domain knowledge integration may include extracting the numerical and geographical characteristics of house numbers, which might lead to more efficient learning. Another option to investigate is the inclusion of transfer learning. By utilising already learnt features, using pre-trained models from similar datasets and customising them to our particular purpose may hasten convergence and reduce overfitting.

Finally, our future work plan emphasises the significance of a thorough model assessment. We aim to completely evaluate model performance over several data splits by adopting rigorous cross-validation procedures, identifying problematic overfitting causes, and more precisely assessing generalisation capabilities. Our plan for moving the research forward and supporting effective machine learning solutions in this field is this complex strategy for resolving overfitting and improving model performance on the SVHN dataset.

X. References

1. Yang, H., Yao, H. (2019). Street View house number identification based on deep learning. International Journal of Advanced Network, Monitoring, and Controls, 4(3), 47–52. https://doi.org/10.21307/ijanmc-2019-058

2. Zhang, G., Yi, J., Yuan, J., Li, Y., Jin, D. (2023). DAS: Efficient Street View Image Sampling for urban prediction. ACM Transactions on Intelligent Systems and Technology, 14(2), 1–20. https://doi.org/10.1145/3576902

3. Goodfellow, I. J. (2013, December 20). Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. arXiv.org. https://arxiv.org/abs/1312.6082

4. Luan, S., Chen, C., Zhang, B., Han, J., Liu, J. (2018). Gabor Convolutional Networks. IEEE Transactions on Image Processing, 27(9), 4357–4366. https://doi.org/10.1109/tip.2018.2835143

5. Sermanet, P. (2012, April 18). Convolutional neural networks applied to house numbers digit classification. arXiv.org. https://arxiv.org/abs/1204.3968

6. Ringland, J., Bohm, M., Baek, S. (2019). Characterization of food cultivation along roadside transects with Google Street View imagery and deep learning. Computers and Electronics in Agriculture, 158, 36–50. https://doi.org/10.1016/j.compag.2019.01.014

7. Zeng, T., Wang, J., Wang, X., Zhang, Y., Ren, B. (2023). An efficient Deep Learning-Based High-Definition image compressed sensing framework for Large-Scene construction site monitoring. Sensors, 23(5), 2563. https://doi.org/10.3390/s23052563

8. Hussain, D., Hussain, I., Ismail, M., Alabrah, A., Ullah, S. S., Alaghbari, H. M. (2022). A simple and efficient Deep Learning-Based framework for automatic fruit recognition. Computational Intelligence and Neuroscience, 2022, 1–8. https://doi.org/10.1155/2022/6538117

9. Elshawi, R., Wahab, A., Barnawi, A., Sakr, S. (2021). DLBench: a comprehensive experimental evaluation of deep learning frameworks. Cluster Computing, 24(3), 2017–2038. https://doi.org/10.1007/s10586-021-03240-4

10. Shorten, C., Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1). https://doi.org/10.1186/s40537-019-0197-0

11. Bansal, A., Sharma, R., Kathuria, M. (2022). A Systematic Review on Data Scarcity Problem in Deep Learning: Solution and Applications. ACM Computing Surveys, 54(10s), 1–29. https://doi.org/10.1145/3502287

12. Abu, M. A., Indra, N. H., Rahman, A. H. A., Sapiee, N. A., Ahmad, I. (2019). A study on Image Classification based on Deep Learning and Tensorflow. International Journal of Engineering Research and Technology, 12(4), 563-569.https://www.ripublication.com/irph/ijert19/ijertv12n4$_1 6.pdf$

13. Sharma, H. K., Choudhury, T., Mohanty, S. N., Swagatika, S., Swain, S. Deep Learning based approach for Photographs and Painting Classification using CNN Model.https://ceur-ws.org/Vol-3283/Paper101.pdf

14. Xu, H., Sun, H., Wang, L. N., Yu, X., Li, T. (2023). Urban Architectural Style Recognition and Dataset Construction Method under Deep Learning of Street View Images: A Case Study of Wuhan. ISPRS International Journal of Geo-information, 12(7), 264. https://doi.org/10.3390/ijgi12070264

15. Yang, F., Wang, M. (2021). Deep Learning-Based Method for Detection of External Air Conditioner Units from Street View Images. Remote Sensing, 13(18), 3691. https://doi.org/10.3390/rs13183691

16. Chen, Y., Rajabifard, A. Aleksandrov, M. (2018). Estimating building age from Google street view images using deep learning (short paper). In 10th international conference on geographic information science (GIScience 2018). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. https://drops.dagstuhl.de/opus/volltexte/2018/9368/pdf/LIPIcs-GISCIENCE-2018-40.pdf

17. Yamashita, R., Nishio, M., Gian, R. K., Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. Insights Into Imaging, 9(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

18. Indolia, S., Goswami, A. K., Mishra, S., Asopa, P. (2018). Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. Procedia Computer Science, 132, 679–688. https://doi.org/10.1016/j.procs.2018.05.069

19. O'Shea, K. T., Nash, R. (2015). An introduction to convolutional neural networks. ResearchGate.https://www.researchgate.net/publication/ 285164623$_{A}n_{I}ntroduction_{t}o_{C}onvolutional_{N}eural_{N}etworks.$

20. Taye, M. M. (2023). Theoretical understanding of convolutional neural network: concepts, architectures, appli-

cations, future directions. Computation (Basel), 11(3), 52. https://doi.org/10.3390/computation11030052

21. Nazzal, J. M., El-Emary, I. M., Najim, S. A. (2008). Multilayer Perceptron Neural Network (MLPs) For Analyzing the Properties of Jordan Oil Shale.ResearchGate. https://www.researchgate.net/publication/ 239580128$_{M}ultilaye_{P}erceptron_{N}eural_{N}etwork/MLPs_{F}or_{A}nalyzing_{t}he_{P}roperties_{o}f_{J}ordan_{O}il_{S}hale$

22. Isabona, J., Imoize, A. L., Ojo, S., Karunwi, O., Kim, Y., Lee, C., Li, C. (2022). Development of a multilayer perceptron neural network for optimal predictive modeling in urban microcellular radio environments. Applied Sciences, 12(11), 5713. https://doi.org/10.3390/app12115713

23. Multi-layer perceptron (MLP) neural network technique for offline handwritten Gurmukhi character recognition. (2014, December 1). IEEE Conference Publication — IEEE Xplore. https://ieeexplore.ieee.org/document/7238334

24. Naskath, J., Sivakamasundari, G., Begum, A. a. S. (2022). A study on different deep learning algorithms used in deep neural nets: MLP SOM and DBN. Wireless Personal Communications, 128(4), 2913–2936. https://doi.org/10.1007/s11277-022-10079-4

25. Popescu, M., Balas, V. E., Perescu-Popescu, L., Mastorakis, N. E. (2009). Multilayer perceptron and neural networks. ResearchGate. https://www.researchgate.net/publication/ 228340819$_{M}ultilayer_{p}erceptron_{a}nd_{n}eural_{n}etworks$

26. Murtagh, F. (1991). Multilayer perceptrons for classification and regression. Neurocomputing, 2(5–6), 183–197. https://doi.org/10.1016/0925-2312(91)90023-5

27. Recurrent Neural Networks: an embedded Computing perspective. (2020). IEEE Journals Magazine — IEEE Xplore. https://ieeexplore.ieee.org/document/9044359

28. Wang, J., Li, X., Li, J., Sun, Q., Wang, H. (2022). NGCU: a new RNN model for Time-Series Data Prediction. Big Data Research, 27, 100296. https://doi.org/10.1016/j.bdr.2021.100296

29. Lipton, Z. C. (2015). A Critical Review of Recurrent Neural Networks for sequence Learning. ResearchGate. https://www.researchgate.net/publication/ 277603865$_{AC}ritical_{R}eview_{o}f_{R}ecurrent_{N}eural_{N}etworks_{f}or_{S}equence_{L}earning$

30. Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306. https://doi.org/10.1016/j.physd.2019.132306