Texas A&M University - Commerce

Department of Computer Science

Comparative Analysis of Denoising Autoencoder and Convolutional Neural Networks for MNIST Classification

Bharath Muthuswamy Paran

Supervisor: Derek Harter, Ph.D.

A report submitted in partial fulfilment of the requirements of
Texas A&M University - Commerce for the degree of
Master of Science in *Computer Science*

February 27, 2024

Declaration

I, Bharath Muthuswamy Paran, of the Department of Computer Science, Texas A&M University - Commerce, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of TAMUC and public with interest in teaching, learning and research.

Bharath Muthuswamy Paran
February 27, 2024

Abstract

In order to enchance MNIST classification this research conducts an in-depth comparative analysis between two powerful architectures: the Denoising Autoencoder (DAE) and the Convolutional Neural Network (CNN). Both models are well-regarded for their distinct capabilities — DAE excels in unsupervised learning and feature extraction, while CNN is tailored for image-based tasks, particularly classification. The purpose of this study is to enhance the performance of MNIST digit classification, this study navigates through the architectures of DAE and CNN, analyzing their impact on classification accuracy and noise resilience. The research extends beyond traditional performance metrics, delving into the interpretability of learned representations and assessing the models' ability to handle noisy input data. The research problem centers on deciphering how these architectures influence MNIST classification under varying levels of noise, providing insights into their respective strengths and limitations. By addressing this research problem, the study aims to contribute to the advancement of image classification techniques, offering nuanced guidance on selecting models for MNIST classification tasks, especially in scenarios with noisy input data. The significance of this research lies in its potential to improve the resilience of classification models to real-world noise, thereby enhancing their applicability in practical settings. The findings are expected to benefit practitioners and researchers alike, guiding them in the selection and optimization of models for noise-affected MNIST classification scenarios.

Keywords: Deep Learning, Denoising Autoencoder, Convolutional Neural Networks, Classification, Accuracy.

Acknowledgements

An acknowledgements section is optional. You may like to acknowledge the support and help of your supervisor(s), friends, or any other person(s), department(s), institute(s), etc. If you have been provided specific facility from department/school acknowledged so.

Contents

List of Figures

List of Tables

List of Abbreviations

SMPCS	School of Mathematical, Physical and Computational Sciences

Chapter 1

Introduction

The realm of image classification, a complex perceptual task involving the categorization of objects from images, has witnessed significant advancements over the past decade. Referred to as the process of categorizing objects, image classification relies on the sophisticated analysis of multispectral data, utilizing the underlying multispectral pattern of each pixel as a quantitative basis for classification (Lillesand et al., 2015). Notably, there has been a remarkable improvement in classification accuracy, reflecting the evolution of image classification models. In recent times, these models are increasingly applied across diverse fields, showcasing their versatility. Applications range from tasks such as handwritten digit recognition (Ahlawat et al., 2020) and Vehicle detection and classification (Tsai et al., 2018), deep learning approach to pneumonia classification (Stephen et al., 2019), and military object detection (Janakiramaiah et al., 2023). The existing models are broadly categorized into unsupervised and supervised modes, reflecting the diverse approaches employed in addressing the intricate challenges of image classification.

1.1 Background

This research project is focused on enhancing the classification accuracy of the MNIST digit dataset, a widely used benchmark in the field of machine learning. The motivation behind this study is to address the importance of improving the robustness of MNIST digit classification, particularly in scenarios with noisy input data.

The project conducts an in-depth comparative analysis between two powerful neural network architectures: the Denoising Autoencoder (DAE) and the Convolutional Neural Network (CNN). Both models are chosen for their distinct capabilities, where the DAE excels in unsupervised learning and feature extraction, and the CNN is specifically tailored for image-based tasks, including classification.

The research delves into the impact of these architectures on classification accuracy and noise resilience, going beyond traditional performance metrics. It explores the interpretability of learned representations and evaluates the models' ability to handle noisy input data. Algorithms such as Denoising Autoencoders and Convolutional Neural Networks are the core components under investigation.

The project's significance lies in its potential to advance image classification techniques, providing nuanced guidance for selecting models in MNIST classification tasks, especially when dealing with noisy input data. The findings are expected to benefit both practitioners and re-

searchers, offering insights that can guide the selection and optimization of models for real- world, noise-affected MNIST classification scenarios.

1.2 Research Question

The research questions are:

1. What differences exist between denoising autoencoder (DAE) models' performance indicators when it comes to handwritten digit prediction?
2. What are the contributing elements to these variations, especially with respect to the hyperparameters?
3. What role do different searches for meta parameters play in maximising the performance of these two neural networks?

The central research problem revolves around understanding how the DAE and CNN architectures influence MNIST digit classification under varying levels of noise, providing valuable insights into their respective strengths and limitations. The main goal is to identify the aspects that affect accuracy, with a particular emphasis on the effects of noise levels.

1.3 Aims and objectives

Aims: This study's main goal is determine how well a CNN—which is optimised for image- related tasks—can classify handwritten digits in the MNIST dataset when compared to a DAE—which is specifically made for denoising and feature extraction. Also, provide insightful information on the advantages and disadvantages of both architectures on real world noisy images. The main objectives of this study include:

- Gather the MNIST dataset to make it ready for training models.
- Train a convolutional Neural Network (CNN) on the MNIST data, focusing on improving accuracy, precision, recall, F1-score, and understanding confusion between numbers.
- Train a denoising autoencoder (DAE) on the same MNIST data, emphasizing its ability to learn features without labeled information.
- Adjust various parameters of both CNN and DAE to get the best performance.
- Compare how well CNN and DAE perform after adjusting parameters, understanding which one is better for predicting numbers in the MNIST dataset.
- Explore how well both models understand and handle noisy data.
- Test the models' ability to work well when there is some noise in the MNIST dataset.

1.4 Solution approach

The study uses a step-by-step approach, including setting up models, preparing data, splitting the data, adjusting key parameters, and evaluating performance. The research achieves its goals by thoroughly exploring different ways to find the best model hyperparameters.

1.4.1 Data Description:

The MNIST dataset (Tensorflow, 1994) is a widely used collection of handwritten digit images commonly employed in the field of machine learning and computer vision. The dataset consists of digits in the images range from 0 to 9, and each image is labeled with its corresponding digit. This dataset serves as a fundamental benchmark for developing and testing image classification algorithms, particularly those focused on recognizing and classifying handwritten digits.

1.4.2 Data Preprocessing:

Data preprocessing plays a crucial role in optimizing the MNIST dataset for effective deep learning model training. Here are the few steps involved in preparing the MNIST dataset:

1.4.3 Handling Missing or Noisy Data:

Checking for any missing or corrupted data points within the dataset. Addressing any noise or outliers that might affect the model's performance.

1.4.4 Reshaping and Normalization:

Reshaping the images to a standard format. For MNIST, this often involves converting 28x28 pixel images into a flattened array of 784 pixels. Normalize the pixel values to a range between 0 and 1. This ensures consistent scaling and aids in faster convergence during model training.

1.4.5 Model Implementation:

Convolutional Neural Network (CNN) Training:

Implementing and training a standard CNN on the labeled MNIST dataset. Focused on optimizing key performance metrics: accuracy, precision, recall, F1-score, and confusion matrix.

Denoising Autoencoder (DAE) Training:

Training a competitive Denoising Autoencoder on the same MNIST dataset. Leverage DAE's unsupervised learning and feature extraction capabilities.

1.4.6 Hyperparameter Tuning:

Exploring and fine-tune meta-parameters for both CNN and DAE models. Consider a comprehensive range of hyperparameters to maximize predictive performance.

1.4.7 Performance Evaluation:

Evaluating and compare the performance metrics of CNN and DAE models. Assess strengths and limitations, providing insights into model suitability for MNIST digit classification.

1.4.8 Interpretability and Noisy Data Handling:

Investigate the interpretability of learned representations from both models. Assess models' resilience in handling noisy input data.

1.5 Summary of contributions and achievements

Describe clearly what you have done/created/achieved and what the major results and their implications are.

Chapter 2

Literature Review

This section explores the existing existing body of knowledge surrounding the Comparative Analysis of Denoising Autoencoder (DAE) and Convolutional Neural Networks (CNN) for MNIST Classification. Recent advancements in deep machine learning have significantly improved feature learning in image processing tasks, particularly in image classification, making it imperative to explore and understand the capabilities and limitations of different models. Notably, Convolutional Neural Networks (CNNs) have emerged as powerful tools, showcasing superior performance in image classification tasks. However, the inherent challenge of noise in real-world data poses a significant constraint on the performance of deep neural networks, prompting the exploration of denoising techniques. The literature review will examine prior studies focusing on image denoising methods, including traditional approaches and those leveraging deep learning, with a specific emphasis on Denoising Autoencoders. Understanding the existing landscape is crucial for contextualizing the current research and identifying gaps that the Comparative Analysis seeks to address.

2.1 Review of state-of-the-art

The (Guo et al., 2017) explores the application of deep learning, specifically Convolutional Neural Networks (CNNs), in image classification. The authors discuss various models commonly used in deep learning, such as Auto Encoder, sparse coding, Restricted Boltzmann Machine, Deep Belief Networks, and CNNs. They emphasize the effectiveness of CNNs in image classification, particularly showcasing their high performance. The study involves building a simple CNN for image classification, with experiments conducted on benchmark datasets like minist and cifar-10. The authors delve into the fundamental components of CNNs, including Convolutional layers, pooling layers, and fully-connected layers. The study focuses on learning rate strategies and optimization algorithms for solving optimal parameters in image classification. Different strategies are compared, demonstrating the impact on recognition rates during training and testing. The experimental results and

The (Bajaj et al., 2020) proposes an efficient image denoising technique using autoencoders based on deep learning models. The motivation behind image denoising is to eliminate noise from images, which can be caused by various factors such as defects in camera sensors, transmission in noisy channels, or faulty memory locations in hardware. The proposed model utilizes autoencoders, a type of artificial neural network, for image denoising. Autoencoders learn noise

patterns from training images and attempt to eliminate noise from novel images. The proposed network architecture consists of convolutional denoising autoencoder (CDA) blocks, each containing internal layers like convolution, pooling, deconvolution, and upsampling. Skip connections are employed to enhance the model's ability to capture finer image details and facilitate back-propagation during training. The paper discusses related work, highlighting methods such as non-local means, Block Matching and 3D Matching (BM3D), and denoising autoencoders. STL-10 dataset for training and the SET5 standard image dataset for testing. Performance metrics, such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM), are used to evaluate the proposed model's effectiveness in comparison to baseline models. The results show that the proposed model outperforms Convolutional Denoising Deep Neural Network (CDDNN) and Residual Encoder Decoder with 30 layers (RED30) in terms of PSNR, indicating superior denoising capabilities.

2.2 Critique of the review

The literature review provides a comprehensive overview of the existing body of knowledge on the Comparative Analysis of Denoising Autoencoder (DAE) and Convolutional Neural Networks (CNN) for MNIST Classification. The review is structured logically, beginning with an introduction to the significance of the topic and progressing to a review of state-of-the-art research.

- Clear Problem Statement: The literature review effectively establishes the problem by highlighting the challenges posed by noise in real-world data for deep neural networks. The motivation to explore denoising techniques is well justified in the context of advancements in deep machine learning and the superior performance of CNNs in image classification.
- Thorough State-of-the-Art Review: The review of state-of-the-art research is well-detailed, discussing relevant models in deep learning for image classification, such as Auto Encoder, sparse coding, Restricted Boltzmann Machine, Deep Belief Networks, and CNNs. The exploration of fundamental components of CNNs, learning rate strategies, and optimization algorithms adds depth to the understanding of the field.
- Comprehensive Denoising Technique Overview: The literature appropriately includes a discussion on denoising techniques, with a specific emphasis on the proposed model using autoencoders. The incorporation of related work, including traditional methods like non-local means and modern approaches like denoising autoencoders, enriches the background information.
- Methodology Explanation: The detailed description of the proposed image denoising technique using convolutional denoising autoencoder (CDA) blocks, skip connections, and performance metrics (PSNR and SSIM) adds transparency to the methodology.
- Effective Comparison of Models: The comparative analysis between the proposed model and existing models like Convolutional Denoising Deep Neural Network (CDDNN) and Residual Encoder Decoder with 30 layers (RED30) is valuable. The use of performance metrics provides a quantitative basis for comparison.

2.3 Summary

In this literature review, the Comparative Analysis of Denoising Autoencoder (DAE) and Convolutional Neural Networks (CNN) for MNIST Classification is explored within the context of recent advancements in deep machine learning. Convolutional Neural Networks (CNNs) have shown superior performance in image classification tasks, but the challenge of noise in real-world data necessitates the investigation of denoising techniques. The review critically examines prior studies on image denoising, with a specific focus on Denoising Autoencoders.

The state-of-the-art review discusses the application of CNNs in image classification, emphasizing their effectiveness. It covers various deep learning models, including Auto Encoder, sparse coding, Restricted Boltzmann Machine, and Deep Belief Networks, with experiments conducted on benchmark datasets. Another study proposes an image denoising technique using autoencoders, demonstrating its effectiveness through metrics like Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). The proposed model outperforms existing models, showcasing superior denoising capabilities. The comprehensive overview of denoising techniques, methodology transparency, and effective comparison of models contribute to the overall strength of the literature review. This groundwork sets the stage for the Comparative Analysis, aiming to address gaps in understanding and provide insights into the capabilities and limitations of DAEs and CNNs for MNIST Classification.

Chapter 3

Methodology

3.1 Convolutional Neural Networks

CNNs are a variant of multi-layered neural networks designed for processing two-dimensional data, particularly well-suited for classifying visual patterns like pixel images with minimal pre-processing. Feature extraction from observed data occurs at each layer by applying digital filtering techniques, allowing information to propagate through multiple layers.

THe CNN architecture, introduced by LeCun et al. in 1998, are applied to classify images and perceive visual patterns directly from pixel images. Information propagation throughout multiple layers enables feature extraction using digital filtering techniques. CNNs perform two main processes: convolution and subsampling. The convolution involves applying a small-sized kernel over the input feature map (IFM) to produce a convolved feature map (CFM). CFMs are computed by applying the convolutional operation over the original input image using a kernel, which represents weights and biases. Weights are shared among positions, preserving spatial locality during the convolution process. The subsampling simplifies the feature map gained from convolution by selecting significant features and discarding the rest. Max-pooling, a subsampling method used in experiments, involves taking the maximum value over non-overlapping sub-regions.

3.2 Denoising Auto-Encoder

Denoising Autoencoder (DAE) is a specialized type of artificial neural network designed to clean up noisy or corrupted data. It belongs to the family of autoencoders, which are neural networks trained to learn efficient representations of input data. DAEs, in particular, excel in removing noise and capturing essential features from corrupted input.

The encoder is the first part of the DAE. Its role is to transform the input data into a compressed representation. This compressed representation should ideally retain the essential features of the input while filtering out noise. One distinctive feature of DAE is the intentional introduction of noise into the input data. This could involve adding random variations or distortions to simulate real-world noise or corruption.The decoder is the second

part of the DAE. It takes the compressed representation from the encoder and reconstructs the clean version of the original input data. The decoder's task is to recover the essential features and remove the injected noise.

3.3 Examples of the sections of a methodology chapter

A general report structure is summarised (suggested) in Table 3.1. Table 3.1 describes that, in general, a typical report structure has three main parts: (1) front matter, (2) main text, and (3) end matter. The structure of the front matter and end matter will remain the same for all the undergraduate final year project report. However, the main text varies as per the project's needs.

Table 3.1: Undergraduate report template structure

Frontmatter		Title Page Abstract Acknowledgements Table of Contents List of Figures List of Tables List of Abbreviations
Main text	Chapter 1	Introduction
	Chapter 2	Literature Review
	Chapter 3	Methodology
	Chapter 4	Results
	Chapter 5	Discussion and Analysis
	Chapter 6	Conclusions and Future Work
	Chapter 7	Refection
End matter		References Appendices (Optional) Index (Optional)

3.3.1 Example of a software/Web development main text structure

Notice that the "methodology" Chapter of Software/Web development in Table 3.2 takes a standard software engineering paradigm (approach). Alternatively, these suggested sections can be the chapters of their own. Also, notice that "Chapter 5" in Table 3.2 is "Testing and Validation" which is different from the general report template mentioned in Table 3.1. Check with your supervisor if in doubt.

Table 3.2: Example of a software engineering-type report structure

Chapter 1	Introduction	
Chapter 2	Literature Review	
Chapter 3	Methodology	
		Requirements specifications
		Analysis
		Design
		Implementations
Chapter 4	Testing and Validation	
Chapter 5	Results and Discussion	
Chapter 6	Conclusions and Future Work	
Chapter 7	Reflection	

3.3.2 Example of an algorithm analysis main text structure

Some project might involve the implementation of a state-of-the-art algorithm and its performance analysis and comparison with other algorithms. In that case, the suggestion in Table 3.3 may suit you the best.

Table 3.3: Example of an algorithm analysis type report structure

Chapter 1	Introduction	
Chapter 2	Literature Review	
Chapter 3	Methodology	
		Algorithms descriptions
		Implementations
		Experiments design
Chapter 4	Results	
Chapter 5	Discussion and Analysis	
Chapter 6	Conclusion and Future Work	
Chapter 7	Reflection	

3.3.3 Example of an application type main text structure

If you are applying some algorithms/tools/technologies on some problems/datasets/etc., you may use the methodology section prescribed in Table 3.4.

3.3.4 Example of a science lab-type main text structure

If you are doing a science lab experiment type of project, you may use the methodology section suggested in Table 3.5. In this kind of project, you may refer to the "Methodology" section as "Materials and Methods."

Table 3.4: Example of an application type report structure

Chapter 1	Introduction	
Chapter 2	Literature Review	
Chapter 3	Methodology	
		Problems (tasks) descriptions
		Algorithms/tools/technologies/etc. descriptions
		Implementations
		Experiments design and setup
Chapter 4	Results	
Chapter 5	Discussion and Analysis	
Chapter 6	Conclusion and Future Work	
Chapter 7	Reflection	

Table 3.5: Example of a science lab experiment-type report structure

Chapter 1	Introduction	
Chapter 2	Literature Review	
Chapter 3	Materials and Methods	
		Problems (tasks) description
		Materials
		Procedures
		Implementations
		Experiment set-up
Chapter 4	Results	
Chapter 5	Discussion and Analysis	
Chapter 6	Conclusion and Future Work	
Chapter 7	Reflection	

3.4 Example of an Equation in LaTeX

Eq. 3.1 [note that this is an example of an equation's in-text citation] is an example of an equation in LaTeX. In Eq. (3.1), s is the mean of elements $x_i \in \mathbf{x}$:

$$s = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{3.1}$$

Have you noticed that all the variables of the equation are defined using the **in-text** maths command $.$, and Eq. (3.1) is treated as a part of the sentence with proper punctuation? Always treat an equation or expression as a part of the sentence.

3.5 Example of a Figure in LaTeX

Figure 3.1 is an example of a figure in LaTeX. For more details, check the link:

wikibooks.org/wiki/LaTeX/Floats,_Figures_and_Captions.

Keep your artwork (graphics, figures, illustrations) clean and readable. At least 300dpi is a good resolution of a PNG format artwork. However, an SVG format artwork saved as a PDF will produce the best quality graphics. There are numerous tools out there that can produce vector graphics and let you save that as an SVG file and/or as a PDF file. One example of such a tool is the "Flow algorithm software". Here is the link for that: flowgorithm.org.

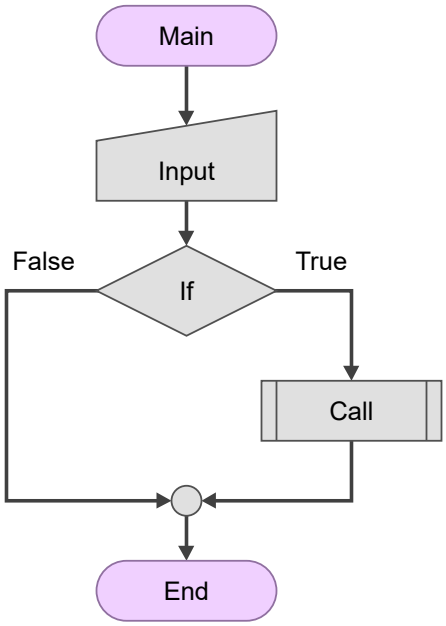

Figure 3.1: Example figure in LaTeX.

3.6 Example of an algorithm in LaTeX

Algorithm 1 is a good example of an algorithm in LaTeX.

Algorithm 1 Example caption: sum of all even numbers

Input: $\mathbf{x} = x_1, x_2, \ldots, x_N$
Output: $EvenSum$ (Sum of even numbers in $\mathbf{x}$)

function EvenSummation($\mathbf{x}$)
  $EvenSum \leftarrow 0$
  $N \leftarrow length(\mathbf{x})$
  for $i \leftarrow 1$ to N **do**
    if $x_i \mod 2 == 0$ **then** ▷ check if a number is even?
      $EvenSum \leftarrow EvenSum + x_i$
    end if
  end for
  return $EvenSum$
end function

3.7 Example of code snippet in LaTeX

Code Listing 3.1 is a good example of including a code snippet in a report. While using code snippets, take care of the following:

- do not paste your entire code (implementation) or everything you have coded. Add code snippets only.
- The algorithm shown in Algorithm 1 is usually preferred over code snippets in a technical/scientific report.
- Make sure the entire code snippet or algorithm stays on a single page and does not overflow to another page(s).

Here are three examples of code snippets for three different languages (Python, Java, and CPP) illustrated in Listings 3.1, 3.2, and 3.3 respectively.

```python
import numpy as np

x  = [0, 1, 2, 3, 4, 5] # assign values to an array
evenSum = evenSummation(x) # call a function

def evenSummation(x):
    evenSum = 0
    n = len(x)
    for i in range(n):
        if np.mod(x[i],2) == 0: # check if a number is even?
            evenSum = evenSum + x[i]
    return evenSum
```

Listing 3.1: Code snippet in LaTeX and this is a Python code example

Here we used the "\clearpage" command and forced-out the second listing example onto the next page.

```java
public class EvenSum{
    public static int evenSummation(int[] x){
        int evenSum = 0;
        int n = x.length;
        for(int i = 0; i < n; i++){
            if(x[i]%2 == 0){ // check if a number is even?
                evenSum = evenSum + x[i];
            }
        }
        return evenSum;
    }
    public static void main(String[] args){
        int[] x  = {0, 1, 2, 3, 4, 5}; // assign values to an array
        int evenSum = evenSummation(x);
        System.out.println(evenSum);
    }
}
```

Listing 3.2: Code snippet in LaTeX and this is a Java code example

```cpp
int evenSummation(int x[]){
    int evenSum = 0;
    int n = sizeof(x);
    for(int i = 0; i < n; i++){
        if(x[i]%2 == 0){ // check if a number is even?
            evenSum = evenSum + x[i];
      }
    }
    return evenSum;
}

int main(){
    int x[]  = {0, 1, 2, 3, 4, 5}; // assign values to an array
    int evenSum = evenSummation(x);
    cout<<evenSum;
    return 0;
}
```

Listing 3.3: Code snippet in LaTeX and this is a C/C++ code example

3.8 Example of in-text citation style

3.8.1 Example of the equations and illustrations placement and reference in the text

Make sure whenever you refer to the equations, tables, figures, algorithms, and listings for the first time, they also appear (placed) somewhere on the same page or in the following page(s). Always make sure to refer to the equations, tables and figures used in the report. Do not leave them without an **in-text citation**. You can refer to equations, tables and figures more them once.

3.8.2 Example of the equations and illustrations style

Write **Eq.** with an uppercase "Eq" for an equation before using an equation number with (\eqref{.}). Use "Table" to refer to a table, "Figure" to refer to a figure, "Algorithm" to refer to an algorithm and "Listing" to refer to listings (code snippets). Note that, we do not use the articles "a," "an," and "the" before the words Eq., Figure, Table, and Listing, but you may use an article for referring the words figure, table, etc. in general.

For example, the sentence "A report structure is shown in **the** Table 3.1" should be written as "A report structure is shown **in** Table 3.1."

3.9 Summary

Write a summary of this chapter.

Note: In the case of **software engineering** project a Chapter "**Testing and Validation**" should precede the "Results" chapter. See Section 3.3.1 for report organization of such project.

Chapter 4

Results

The results chapter tells a reader about your findings based on the methodology you have used to solve the investigated problem. For example:

- If your project aims to develop a software/web application, the results may be the developed software/system/performance of the system, etc., obtained using a relevant methodological approach in software engineering.
- If your project aims to implement an algorithm for its analysis, the results may be the performance of the algorithm obtained using a relevant experiment design.
- If your project aims to solve some problems/research questions over a collected dataset, the results may be the findings obtained using the applied tools/algorithms/etc.

Arrange your results and findings in a logical sequence.

4.1 A section

...

4.2 Example of a Table in LaTeX

Table 4.1 is an example of a table created using the package LaTeX "booktabs." do check the link: wikibooks.org/wiki/LaTeX/Tables for more details. A table should be clean and readable. Unnecessary horizontal lines and vertical lines in tables make them unreadable and messy. The example in Table 4.1 uses a minimum number of liens (only necessary ones). Make sure that the top rule and bottom rule (top and bottom horizontal lines) of a table are present.

Table 4.1: Example of a table in LaTeX

Bike		
Type	Color	Price (£)
Electric	black	700
Hybrid	blue	500
Road	blue	300
Mountain	red	300
Folding	black	500

4.3 Example of captions style

- The **caption of a Figure (artwork) goes below** the artwork (Figure/Graphics/illustration). See example artwork in Figure 3.1.
- The **caption of a Table goes above** the table. See the example in Table 4.1.
- The **caption of an Algorithm goes above** the algorithm. See the example in Algorithm 1.
- The **caption of a Listing goes below** the Listing (Code snippet). See example listing in Listing 3.1.

4.4 Summary

Write a summary of this chapter.

Chapter 5

Discussion and Analysis

Depending on the type of project you are doing, this chapter can be merged with "Results" Chapter as " Results and Discussion" as suggested by your supervisor.

In the case of software development and the standalone applications, describe the significance of the obtained results/performance of the system.

5.1 A section

Discussion and analysis chapter evaluates and analyses the results. It interprets the obtained results.

5.2 Significance of the findings

In this chapter, you should also try to discuss the significance of the results and key findings, in order to enhance the reader's understanding of the investigated problem

5.3 Limitations

Discuss the key limitations and potential implications or improvements of the findings.

5.4 Summary

Write a summary of this chapter.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Typically a conclusions chapter first summarizes the investigated problem and its aims and objectives. It summaries the critical/significant/major findings/results about the aims and objectives that have been obtained by applying the key methods/implementations/experiment set-ups. A conclusions chapter draws a picture/outline of your project's central and the most signification contributions and achievements.

A good conclusions summary could be approximately 300–500 words long, but this is just a recommendation.

A conclusions chapter followed by an abstract is the last things you write in your project report.

6.2 Future work

This section should refer to Chapter 4 where the author has reflected their criticality about their own solution. The future work is then sensibly proposed in this section.

Guidance on writing future work: While working on a project, you gain experience and learn the potential of your project and its future works. Discuss the future work of the project in technical terms. This has to be based on what has not been yet achieved in comparison to what you had initially planned and what you have learned from the project. Describe to a reader what future work(s) can be started from the things you have completed. This includes identifying what has not been achieved and what could be achieved.

A good future work summary could be approximately 300–500 words long, but this is just a recommendation.

Chapter 7

Reflection

Write a short paragraph on the substantial learning experience. This can include your decision-making approach in problem-solving.

Some hints: You obviously learned how to use different programming languages, write reports in LaTeXand use other technical tools. In this section, we are more interested in what you thought about the experience. Take some time to think and reflect on your individual project as an experience, rather than just a list of technical skills and knowledge. You may describe things you have learned from the research approach and strategy, the process of identifying and solving a problem, the process research inquiry, and the understanding of the impact of the project on your learning experience and future work.

Also think in terms of:

- what knowledge and skills you have developed
- what challenges you faced, but was not able to overcome
- what you could do this project differently if the same or similar problem would come
- rationalize the divisions from your initial planed aims and objectives.

A good reflective summary could be approximately 300–500 words long, but this is just a recommendation.

Note: The next chapter is "**References**," which will be automatically generated if you are using BibTeX referencing method. This template uses BibTeX referencing. Also, note that there is difference between "References" and "Bibliography." The list of "References" strictly only contain the list of articles, paper, and content you have cited (i.e., refereed) in the report. Whereas Bibliography is a list that contains the list of articles, paper, and content you have cited in the report plus the list of articles, paper, and content you have read in order to gain knowledge from. We recommend to use only the list of "References."

References

Ahlawat, S., Choudhary, A., Nayyar, A., Singh, S. and Yoon, B. (2020), 'Improved handwritten digit recognition using convolutional neural networks (cnn)', *Sensors* **20**(12), 3344.

Bajaj, K., Singh, D. K. and Ansari, M. A. (2020), 'Autoencoders based deep learner for image denoising', *Procedia Computer Science* **171**, 1535–1541.

Guo, T., Dong, J., Li, H. and Gao, Y. (2017), Simple convolutional neural network on image classification, *in* '2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)', pp. 721–724.

Janakiramaiah, B., Kalyani, G., Karuna, A., Prasad, L. N. and Krishna, M. (2023), 'Military object detection in defense using multi-level capsule networks', *Soft Computing* **27**(2), 1045–1059.

Lillesand, T., Kiefer, R. W. and Chipman, J. (2015), *Remote sensing and image interpretation*, John Wiley & Sons.

Stephen, O., Sain, M., Maduh, U. J., Jeong, D.-U. et al. (2019), 'An efficient deep learning approach to pneumonia classification in healthcare', *Journal of healthcare engineering* **2019**.

Tensorflow (1994), 'Mnist dataset'. (accessed January 25, 2024).
URL: *https://www.tensorflow.org/datasets/catalog/mnist*

Tsai, C.-C., Tseng, C.-K., Tang, H.-C. and Guo, J.-I. (2018), Vehicle detection and classification based on deep neural network for intelligent transportation applications, *in* '2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)', IEEE, pp. 1605–1608.

Appendix A

An Appendix Chapter (Optional)

Some lengthy tables, codes, raw data, length proofs, etc. which are **very important but not essential part** of the project report goes into an Appendix. An appendix is something a reader would consult if he/she needs extra information and a more comprehensive understating of the report. Also, note that you should use one appendix for one idea.

An appendix is optional. If you feel you do not need to include an appendix in your report, avoid including it. Sometime including irrelevant and unnecessary materials in the Appendices may unreasonably increase the total number of pages in your report and distract the reader.

Appendix B

An Appendix Chapter (Optional)

...