DRIVING THE FUTURE: GENERATIVE AI FOR NEXT-GENERATION AUTONOMOUS VEHICLES

A REPORT

submitted in partial fulfillment of the requirements

for the award of

Bachelor of Science

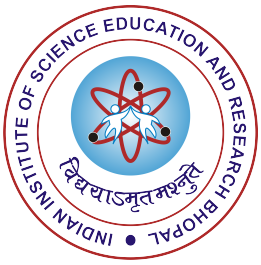
in

ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

by

T VAMSIKRISHNA YADAV

(20287)



DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH BHOPAL
BHOPAL – 462066

April 2024

भारतीय विज्ञान शिक्षा एवं अनुसंधान संस्थान भोपाल
Indian Institute of Science Education and Research
Bhopal
(Estb. By MHRD, Govt. of India)

CERTIFICATE

This is to certify that **T VAMSIKRISHNA YADAV**, BS (Electrical Engineering And Computer Science), has worked on the project entitled **'Driving the Future: Generative AI for Next-Generation Autonomous Vehicles '** under my supervision and guidance. The content of this report is original and has not been submitted elsewhere for the award of any academic or professional degree.

April 2024 **Dr. Kuntal Roy, EECS Department**
IISER Bhopal

ACADEMIC INTEGRITY AND COPYRIGHT DISCLAIMER

I hereby declare that this project is my own work and, to the best of my knowledge, it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at IISER Bhopal or any other educational institution, except where due acknowledgement is made in the document.

I certify that all copyrighted material incorporated into this document is in compliance with the Indian Copyright Act (1957) and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless IISER Bhopal from any and all claims that may be asserted or that may arise from any copyright violation.

April 2024 **T VAMSIKRISHNA YADAV**

IISER Bhopal

LIST OF SYMBOLS OR ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
SVM	Support Vector Machines
NN	Neural Network
DL	Deep Learning
AV	Autonomous Vehicles
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
VAE	Variational Autoencoder
GAN	Generative Adversarial Networks
YOLOv5	You Only Look Once version 5
MNIST	Modified National Institute of Standards and Technology
FMNIST	Fashion MNIST
GAIA	Generative AI for Autonomy

ABSTRACT

Autonomous driving is a promising future for transportation. The realm of transportation is undergoing a revolutionary transformation with the emergence of autonomous vehicles (AVs). This project delves into the intricate world of autonomous driving, exploring its potential to revolutionize the way we travel.

This project explores the Generative Artificial Intelligence (AI) field and its potential applications to the next generation autonomous vehicles. First, the project explores the fundamentals of deep learning by implementing a simple Artificial Neural Network (ANN) for handwritten digit classification using the MNIST dataset and a Convolutional Neural Network (CNN) with the Fashion-MNIST dataset. We review the relevant literature, including studies like "GAIA-1: A Generative World Model for Autonomous Driving", and then it explores autoencoders and Generative Adversarial Networks (GANs) in the context of autonomous vehicles. Building upon the aforesaid foundation, the project delves into practical implementations of deep learning models. A pre-trained model, YOLOv5, is utilised for image segmentation tasks, demonstrating its capabilities in image analysis. Finally, the project leverages a pre-trained U-Net model, showcasing the power of pre-trained networks for specific applications like image segmentation. Improving specific aspects of autonomous vehicle performance related to simulating diverse driving scenarios can be done by generating realistic and varied training data for handling unexpected situations, and enhancing perception can be done by training models to recognize objects better and predict their behaviour. Overall, this project contributes to the exploration of Generative AI for autonomous vehicles, laying the groundwork for future research by demonstrating the capabilities of various deep learning architectures in related tasks.

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

1.1 Machine Learning

The trait that enabled humans to outcompete other animals was our capacity for self-improvement and learning from our failures. We hardly even stop to consider the learning process because it appears so evident. Trying to get a computer to recognise an image highlights the challenge. This is where neural networks and machine learning (ML) in general come into play [20]. Neural networks draw inspiration from the human brain, which gains knowledge via experience and gradually enhances its performance in relation to a particular task. There are many different types of ML algorithms. These include clustering, classification, and generative algorithms [21, 22], which use models like neural networks and support vector machines (SVM) [23].

The following is a high-level description of various paradigms.

- **Supervised learning:** ML techniques that concentrate on inference models, including regression and classification, are referred to as supervised learning. A supervised algorithm's objective is to discover the correlations between an input set (X) and a related output set (Y), which can be utilised to determine the right output for an unknown input.

  Generally supervised learning techniques fall into one of two categories:

  1. **Classification:** The test samples are to be assigned to one or more distinct groups that correspond with the problem. To categorise the incoming data, a classifier can self-train using the training samples.

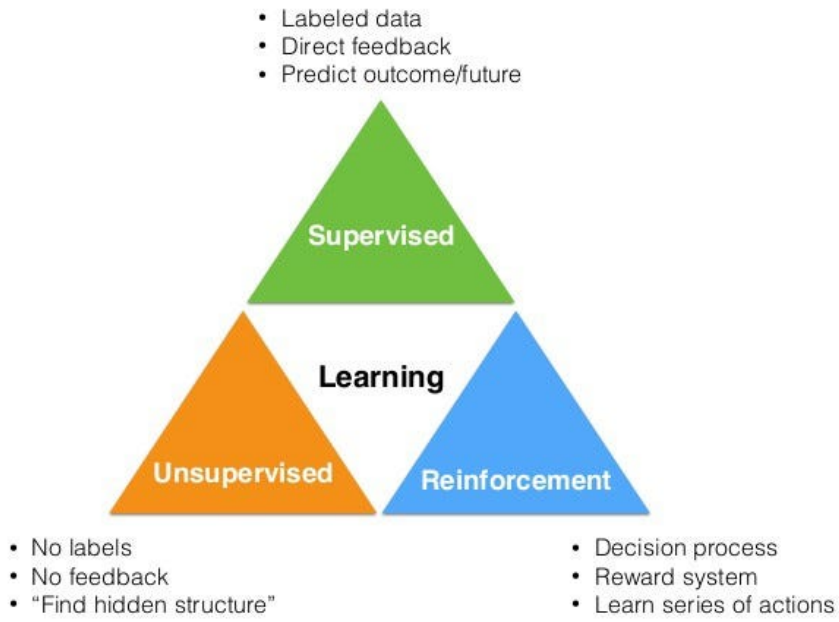


Fig. 1.1: ML Paradigms [1]

2. **Regression:** Regression is the term for an output that is intended to include one or more continuous variables.

Figure 1.1 displays a representation illustrating the main machine learning paradigms.

- **Unsupervised learning:** Algorithms for unsupervised machine learning explore the fundamental characteristics of a given dataset. Among them are dimensionality reduction, which seeks to minimise a dataset's dimension while retaining as much information as possible, generative modelling, which focuses on discovering the underlying probability distribution of a given data set, and clustering, which groups data points based on similarities.

  The two categories of unsupervised learning approaches are as follows:

  1. **Clustering:** Finding things with similar properties and separating objects with different properties is the goal. It eventually

produces various distinct data clusters.

2. **Density Estimation:** To ascertain the data's distribution inside the input space is the goal.

- **Reinforcement learning:** Reinforcement learning is a machine learning paradigm in which an agent attempts to learn the best course of action by investigating feasible actions given an accessible (perhaps action-dependent) state space. Actions that result in a higher score are rewarded, whereas actions that result in a lower score are penalised. The taken actions, respectively, reached states are associated with a given score. One such example is AlphaGo [24], a Go engine developed by DeepMind using reinforcement learning, which beat the world champion 4 to 1.

Another important aspect is **Feature Engineering**. In this process, the primary goal is to identify several noteworthy characteristics within the data. In order to enhance the efficiency of classification and clustering algorithms, feature engineering is utilised to eliminate noise from the data. As stated below, there are two different kinds of feature engineering.

1. **Supervised:** The techniques for supervised feature engineering are sometimes referred to as feature selection. By identifying connections between the attributes and the classes, these techniques award scores in the appropriate ways. Finally, depending on the ratings, the best characteristics are chosen. Various heuristic and information-thetic feature selection strategies exist, such as information gain and χ^2 statistic.

2. **Unsupervised:** The term "feature extraction" refers to unsupervised feature engineering methods. Such techniques include those based on entropy and principal component analysis.

1.2 Neural Networks

Inspired by the structure and operation of the human brain, neural networks, sometimes referred to as artificial neural networks (ANNs), are a potent tool

in the field of machine learning. The salient features are described below.

Synthetic Neurons: Artificial neurons, which roughly resemble the actual neurons in our brains, are the fundamental units of neural networks. Each neuron processes information from other neurons, computes, and sends the result to other neurons.

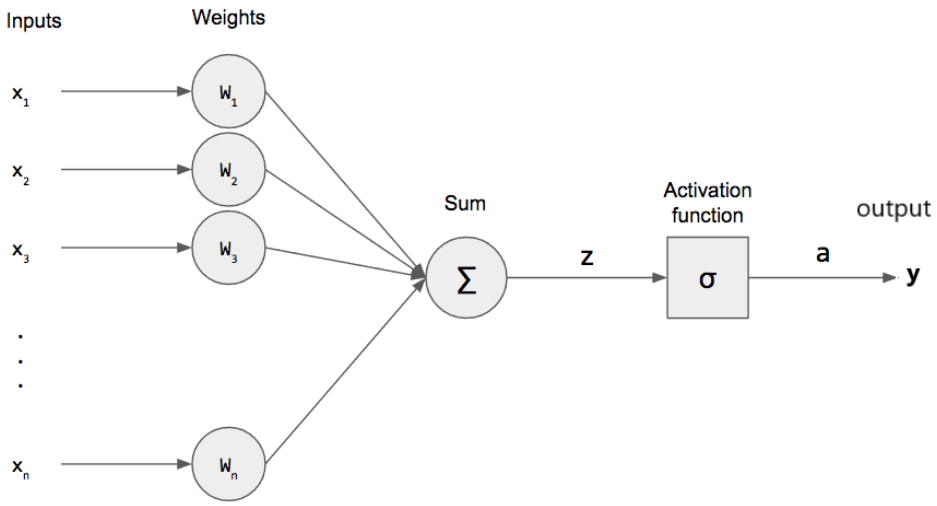


Fig. 1.2: The Basic Perceptron Model [2]

Layers: There are several layers to these artificial neurons as follows.

- The input layer is where the network's first data is fed in.
- Hidden Layers: The primary processing and learning take place in one or more hidden layers.
- The output layer generates the network's ultimate output.

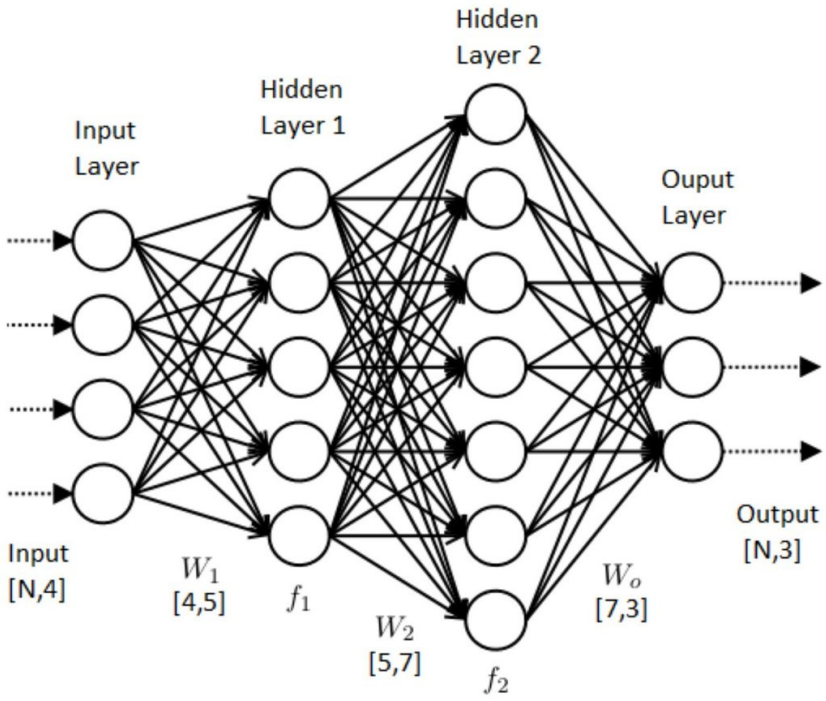


Fig. 1.3: Artificial Neural Network [3]

Weights and Connections: Every neuron has weights that are correlated with its connections to other neurons in various layers. The strength of the signal that is sent between neurons is determined by these weights. There is also a *bias* term in addition to the weights.

Activation Function: Every neuron applies a weighted sum of its inputs to an activation function. Whether a neuron is "activated" and sends its output to the following layer is determined by this function.

Learning: When given training data, neural networks modify the weights of its connections in order to learn. This is called the *training* of the ANNs. The goal of this procedure is to reduce the discrepancy between the intended outputs and the network's predictions to unknown data, which is called the *evaluation* phase.

1.2.1 Loss Function

A *loss function* (also referred to as a *cost function*) is a crucial part of neural network training. It functions as a gauge for how closely the network's predictions resemble the real ground truth values. The network learns to modify its internal parameters (weights and biases) to enhance its performance on the given task by minimising the loss function.

A loss function's main purpose is to measure the difference between the ground truth values found in the training data and the predicted values produced by the neural network.

Loss Function Types: The particular goal and the kind of output the network generates determine which loss function is best. Two typical kinds of loss functions are as follows.

1. **Mean Squared Error (MSE):** MSE is a common option when used in regression issues, which entail predicting continuous numerical values. The average squared difference between the actual and anticipated values is computed.

$$L = \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y}_i)^2 \tag{1.1}$$

2. **Categorical Crossentropy:** This is a technique used in classification situations when the network predicts discrete categories. It calculates the difference between the one-hot encoded representation of the true labels and the probability distribution of the predicted labels.

   Crossentropy in categories $= -\sum(y_i * logp_i)$

1.2.2 Optimizing MSE with Gradient Descent

Let us examine a basic linear regression model that associates an input variable (x) with a predicted output value y. The model has a single weight (w) and bias (b). Using this model, the MSE loss function is as follows

$$L = \frac{1}{N}\sum_{i=1}^{N}(y - (wx + b))^2 \tag{1.2}$$

To minimize MSE using gradient descent, we would calculate the partial derivatives of MSE with respect to w and b:

$$\frac{\partial L}{\partial w} = -\frac{2}{N}\sum_{i=1}^{N}(y - (wx + b))x \tag{1.3}$$

$$\frac{\partial L}{\partial b} = -\frac{2}{N}\sum_{i=1}^{N}(y - (wx + b)) \tag{1.4}$$

These partial derivatives represent the gradients, which indicate the direction and magnitude of change required to reduce the loss. The weights and biases are then updated iteratively using the following update rules:

$$w_{new} = w_{old} - \eta \cdot \frac{\partial L}{\partial w} \tag{1.5}$$

$$b_{new} = b_{old} - \eta \cdot \frac{\partial L}{\partial b} \tag{1.6}$$

where η, the *learning rate* is a hyperparameter that controls the step size of the updates.

By iteratively applying these updates and minimizing the loss function, the neural network gradually learns to improve its predictions and better approximate the underlying relationship between the input and output data.

1.2.3 Backpropagation

An effective technique for training artificial neural networks is backpropagation. By doing so, the network is able to grow from its errors and eventually become more proficient at a particular task. This is a thorough explanation of backpropagation using formulas:

Fundamental Idea

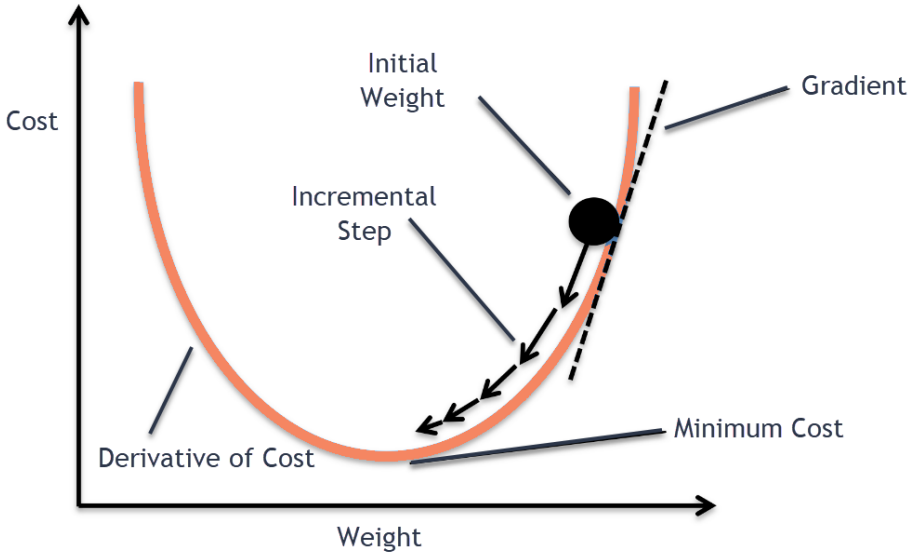


Fig. 1.4: Gradient Descent [4]

In order for backpropagation to function, the network's weights and biases are iteratively adjusted based on the discrepancy between the predicted and actual (ground truth) values. In essence, it works backwards through the network to propagate the error, estimating the contributions of each weight and bias to the total error. Subsequently, the weights and biases are updated using these contributions in a way that minimises error for subsequent predictions.

Two basic steps are involved in backpropagation

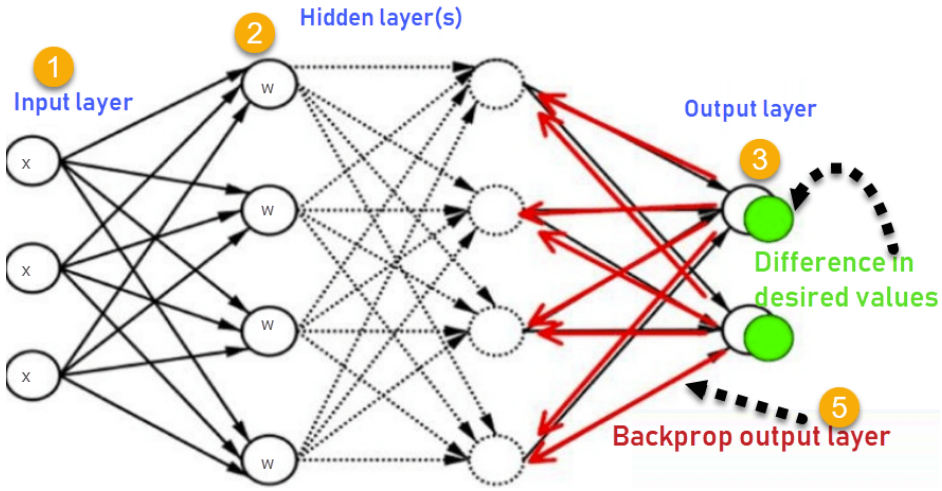


Fig. 1.5: Backpropagation [5]

Forward Pass: Layer after layer, input data is injected into the network. Every neuron generates its output by applying an activation function to the weighted sum of its inputs. The input for the following layer is the output of the previous layer. The output layer generates the network's prediction in the end.

Backward Pass: At the output layer, the error that is, the difference between the ground truth and the prediction is calculated. After then, the network starts to propagate this fault backward. We compute the partial derivative of the error for each neuron in relation to its weights and bias. The amount that each bias and weight contributed to the total error is shown by these partial derivatives. The weights and biases are adjusted in a way that minimises the error using an optimisation process (such as gradient descent). For this update, the current weight or bias value is subtracted from the learning rate (a tiny number that controls the step size) multiplied by the partial derivative.

1.3 Implementing NN with MNIST Database

Our Model Architecture

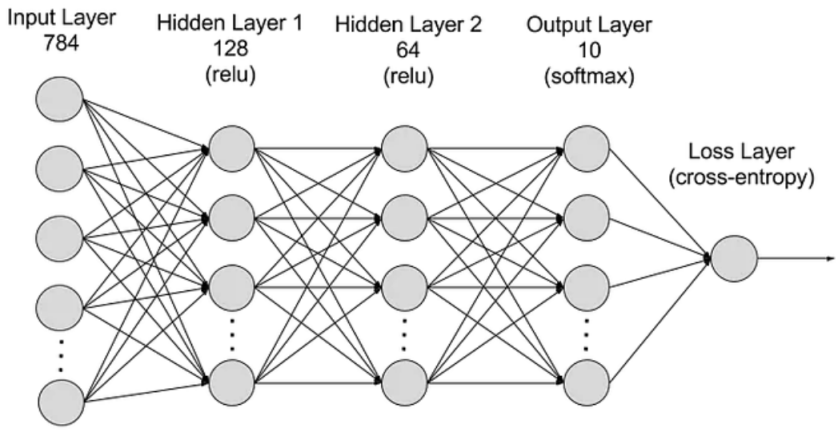


Fig. 1.6: NN Architecture [6]

Model summary: Model "sequential"
Total params: 109386 (427.29 KB)
Trainable params: 109386 (427.29 KB)
Non-trainable params: 0 (0.00 Byte)

Difference between Trainable and Non-Trainable parameters: Parameters are essential to the learning process of Neural Networks. In essence, these parameters are the values that the network modifies throughout training in order to enhance its performance on a particular job. They fall into two main categories: parameters that are trainable and those that are not.

Trainable Parameters: The main movable weights and biases of a neural network are called trainable parameters. During the training phase, the network picks up and refines these values. By varying these parameters

(weights and biases), the network gains the ability to manage information flow and eventually improves prediction quality.

Non-Trainable Parameters: Values in the network that are not updated during training are known as non-trainable parameters. These could be hyperparameters or pre-established variables that govern the entire learning process without being actively learned by the network like Normalization parameters, Learning Rate, Batch Size.

Training, Testing, and Validation Data Split: To prevent overfitting and guarantee that your model performs well on untested data, it is imperative in machine learning, especially when training neural networks, to divide your data into training, testing, and validation sets.

Training Data: The majority of your data typically between 60 and 80 percent is utilised for model training. By modifying its internal parameters (weights and biases), the model "learns" from the patterns and relationships found in the training data.

The quality and quantity of your training data significantly impact the model's performance. More diverse and representative training data leads to a model that can generalize better to unseen situations.

Testing Data: This is a different collection of data (usually 10–20 percent of the total) that the model has never seen before for training purposes. It is employed to assess how well the model performs with unknown data. This aids in evaluating the model's ability to generalise to new cases and how well it has learned from the training set.

Test results offer an objective assessment of the model's effectiveness in actual use. It can be inaccurate to assess performance using the training data since the model may only "memorise" the training examples without actually comprehending the underlying patterns.

Validation Data: This is another separate set of data (typically 10-20

percent) used during the training process to fine-tune the model's hyperparameters that control the learning process itself (e.g., learning rate, number of epochs). Unlike training data, the validation data is not used to directly train the model's weights and biases.

Early Overfitting Detection: Usually, testing and training data are separated out prior to training. Using validation data, you can keep an eye out for overfitting as the model learns from the training set. It is very suggestive of overfitting if the model begins to perform worse on the validation set while doing better on the training set. Then, to avoid memorising and enhance generalisation, you can modify hyperparameters (such as training time and learning rate).

Comparing Validation and Training loss

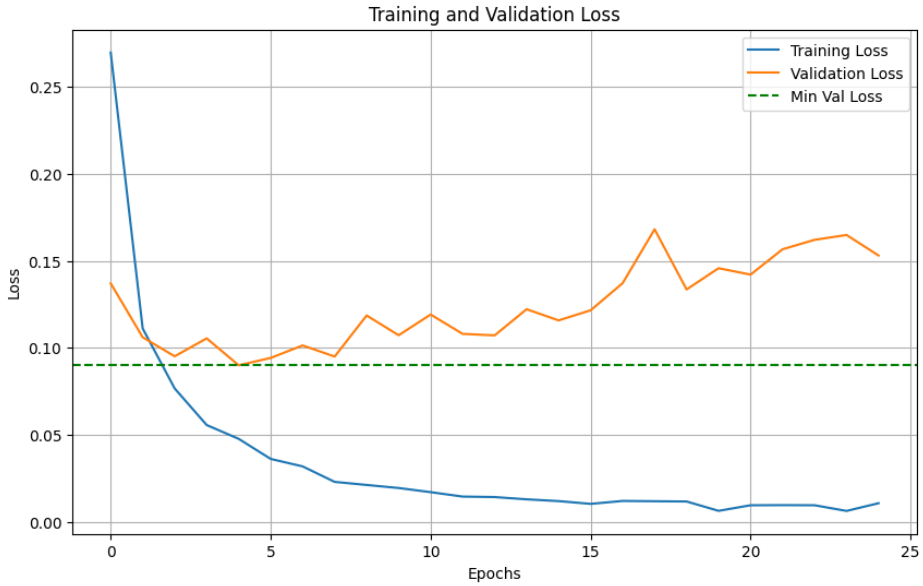


Fig. 1.7: Comparing Validation and Training loss

Comparing Validation and Training Accuracy

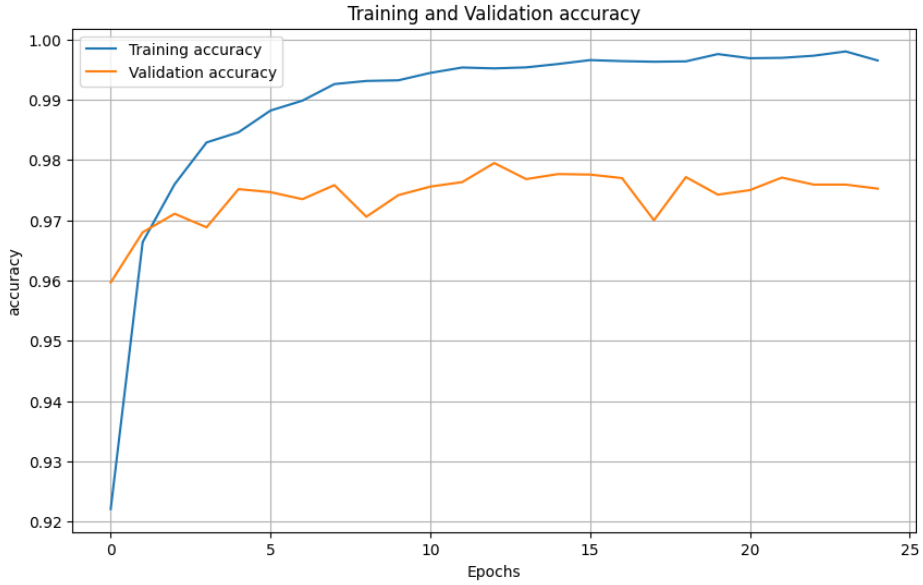
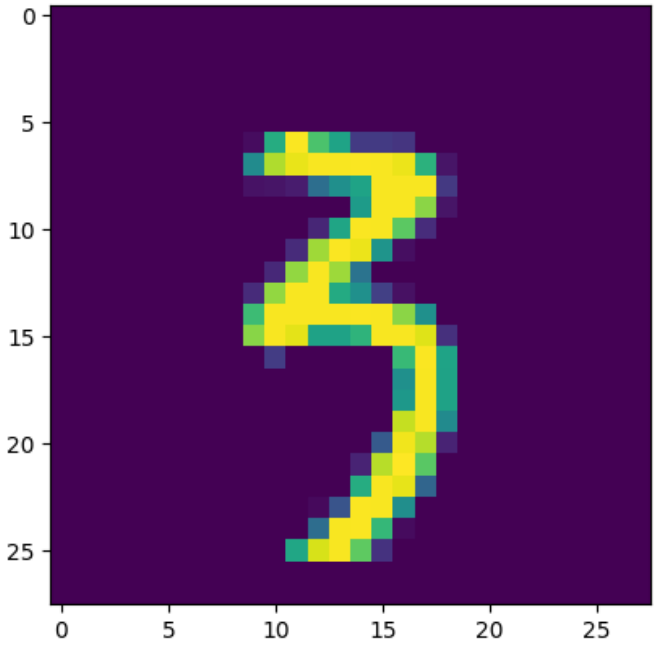


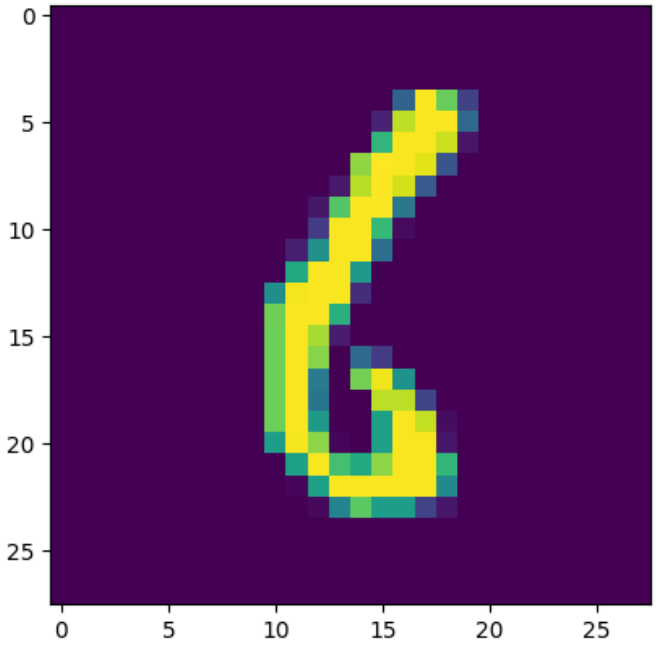
Fig. 1.8: Comparing Validation and Training Accuracy

Failure Points of Neural Networks

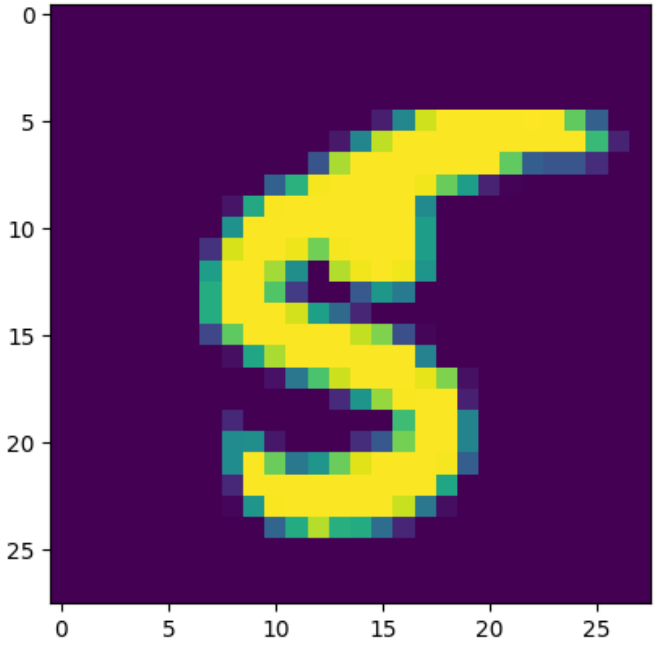
Actual class is 3 but Predicted as 4



Actual class is 6 but Predicted as 5



Actual class is 5 but Predicted as 6



1.4 Convolutional Neural Network

Convolutional neural networks (CNNs): Experts in Perceiving Images Strong artificial neural networks called convolutional neural networks, or CNNs, are made especially for handling grid-like input, most frequently photographs. They are essential to computer vision because they are highly skilled at tasks including object detection, image segmentation, and image classification.

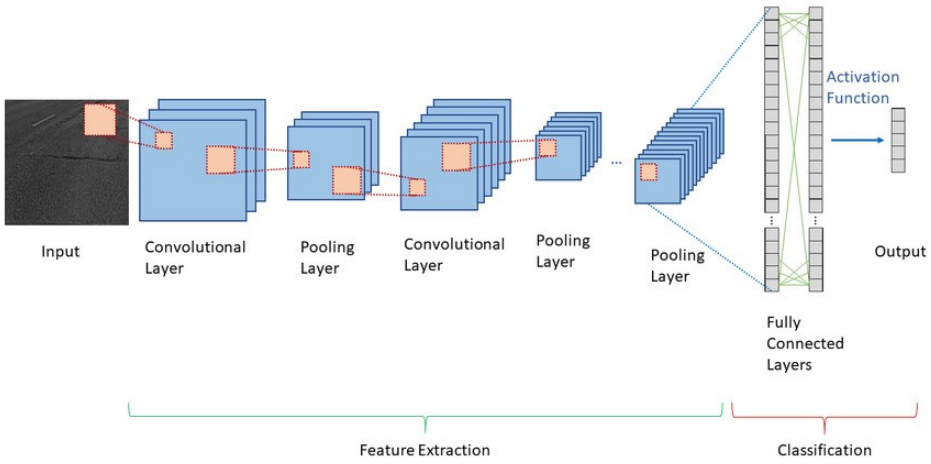


Fig. 1.9: CNN Architecture [7]

Core Idea

CNNs make use of an image's intrinsic qualities, like the spatial relationships among its pixels. They accomplish this by utilising two crucial architectural components:

Convolutional Layers: These layers extract features such as edges, lines, and forms from the input image by applying filters, or kernels, that move across it. A convolutional layer's output is a feature map that highlights the existence of each feature that each filter identifies in the image.

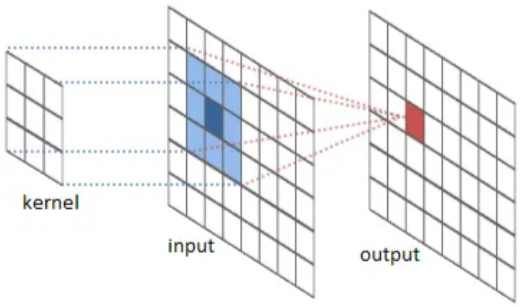


Fig. 1.10: Convolutional Layer [8]

Pooling Layers: These layers downsample the feature maps to lower the dimensionality of the data. Typically, they achieve this by utilising methods like average pooling or max pooling, which take the maximum value. By controlling overfitting, pooling enables the network to concentrate on more salient elements.

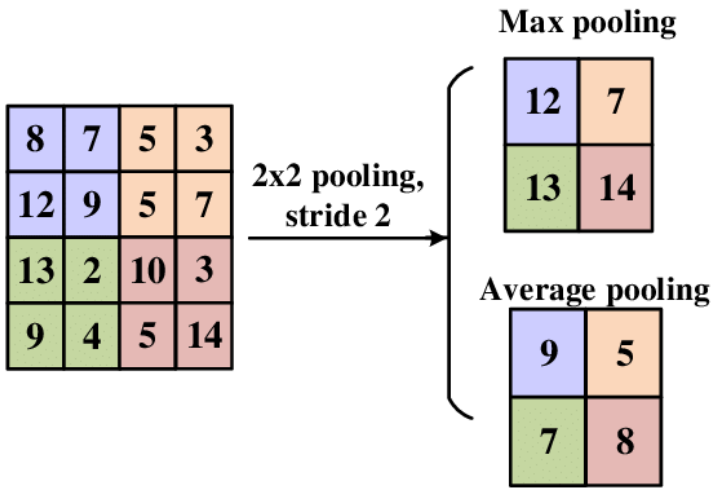


Fig. 1.11: Pooling Layer [9]

Organisation

The layers of a typical CNN architecture are as follows:

The input layer gets the unprocessed image data, which is often displayed

as a three-dimensional tensor with RGB (width, height, and colour) channel dimensions.

Convolutional Layers: A stack of convolutional layers is used, with each layer learning varying degrees of feature complexity. The deeper the network, the more complex the filters get.

Pooling Layers: These layers, which are frequently sandwiched between convolutional layers, minimise the amount of data and add a degree of invariance to slight rotations or shifts in the image.

Activation Layers: To add non-linearity and enhance the network's capacity to learn intricate patterns, non-linear activation functions such as Rectified Linear Unit, or ReLU, are generally added after convolutional layers.

Fully-Connected Layers: Fully-connected layers are utilised later on in the network's development, much like in traditional neural networks, to carry out tasks like picture classification or regression using the features that were retrieved.

Output Layer: Depending on the task, this last layer generates the network's output. It may have many neurons for picture categorization, each expressing a probability for a particular class (e.g., dog, cat).

Many practical applications are powered by CNNs such as:

Classification of Images: Identifying elements of scenes, objects, or actions in pictures. Locating and recognising things in an image is known as object detection. Image segmentation is the process of dividing an image's pixels into meaningful groups, such as foreground and background. Recognition of faces in pictures or videos: Facial Recognition. Medical image analysis: Helping with medical scan tasks such as disease classification or tumour detection. Self-Driving Cars: Autonomous vehicles that can recognise items and navigate their surroundings.

Filters in CNN

In Convolutional Neural Networks (CNNs), filters, also known as kernels, play a crucial role in feature extraction.

The fundamental components of CNN feature extraction are filters. They enable the network to pick up on and recognise pertinent patterns in the incoming data.

The number of filters per layer, their weight combinations, and filter size choices all have a big influence on how well the network performs and can acquire useful features.

By learning appropriate weight configurations, filters become powerful tools for identifying patterns and ultimately leading to accurate image classification, object detection, or other image-based tasks.

Examples of Filters

Edge detection: Vertical or horizontal edges can be identified using filters that have their weights concentrated on one side.

Blob Detection: Certain shapes of blobs can be identified by filters that have positive weights in the centre and negative weights around the edges.

Texture Recognition: Certain textures can be captured by filters whose weight matrix contains recurring patterns.

1.5 Implementing CNN with Fashion MNIST Database

Our CNN Model Architecture

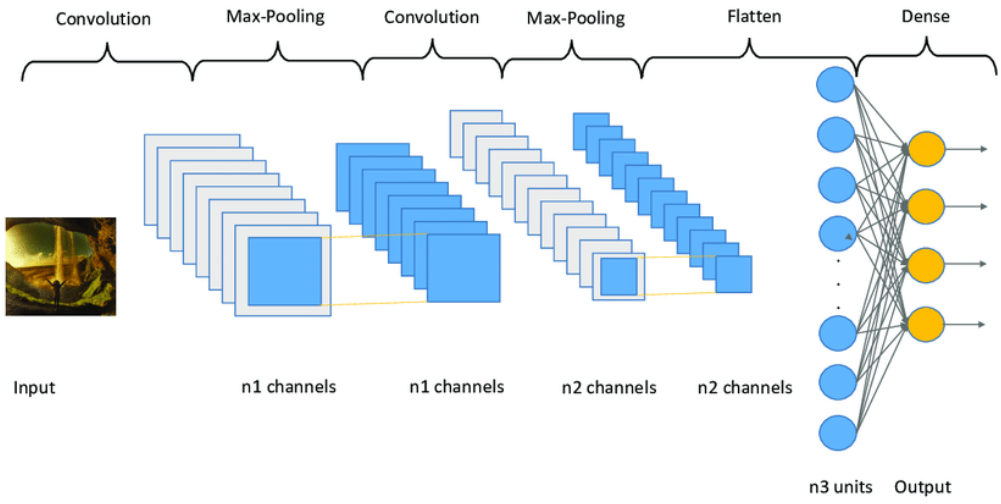


Fig. 1.12: CNN Architecture [10]

Model summary: Model "sequential"
Total params: 113386 (442.91 KB)
Trainable params: 113386 (442.91 KB)
Non-trainable params: 0 (0.00 Byte)

Comparing Validation and Training loss

Fig. 1.13: Comparing Validation and Training loss

Comparing Validation and Training Accuracy

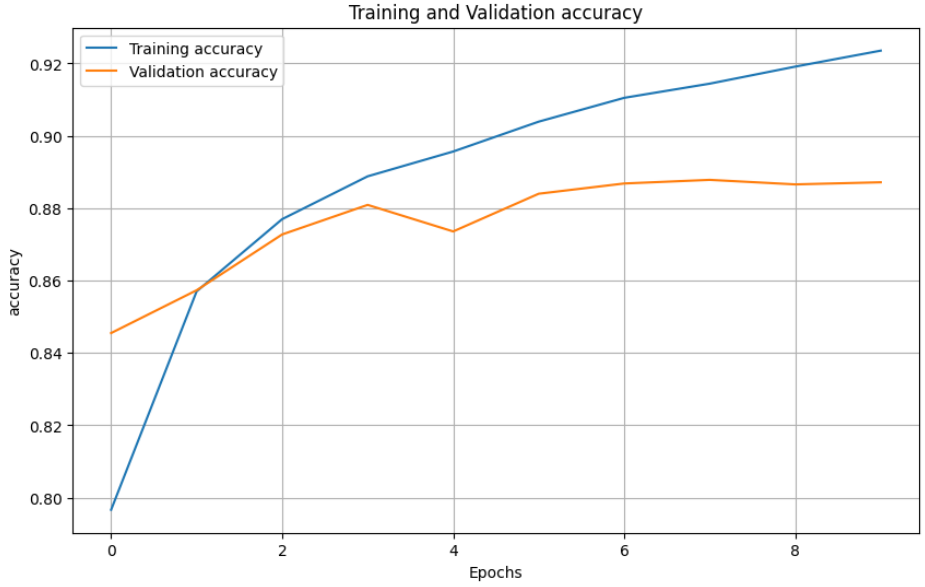
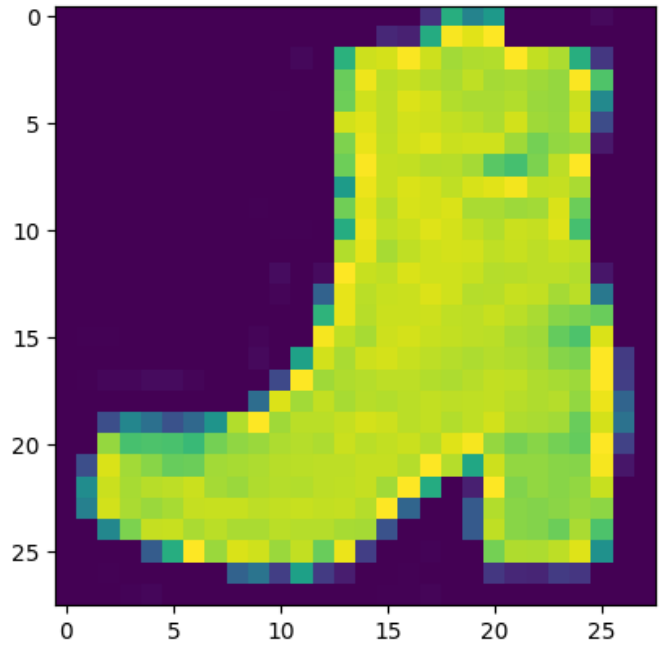
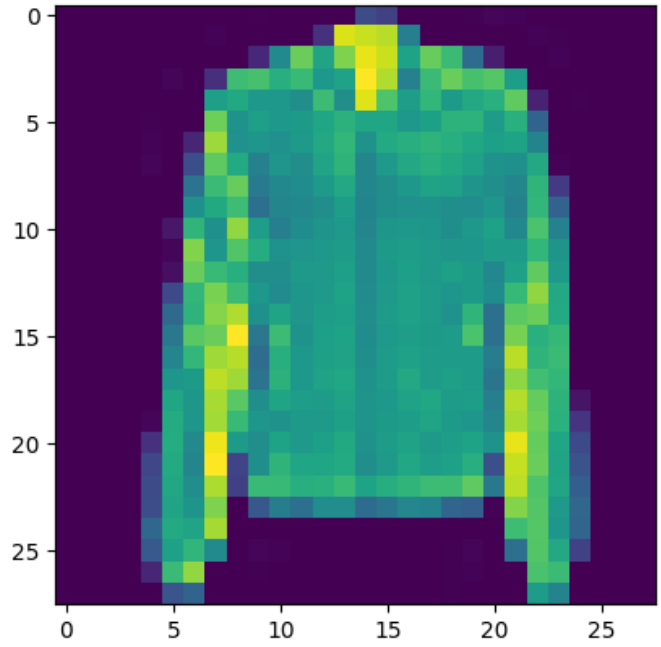


Fig. 1.14: Comparing Validation and Training Accuracy

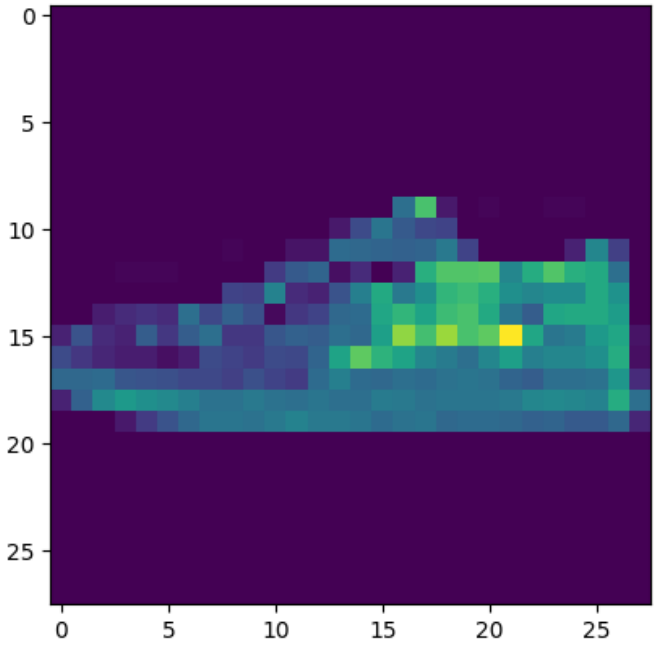
Failure Points of Convolutional Neural Networks
Actual class is Ankle boot but Predicted as Sandal



Actual class is Pullover but Predicted as Bag



Actual class is Sneaker but Predicted as Sandal



Difference in Predictions between SimpleNN and CNN Prediction of Class Sandal

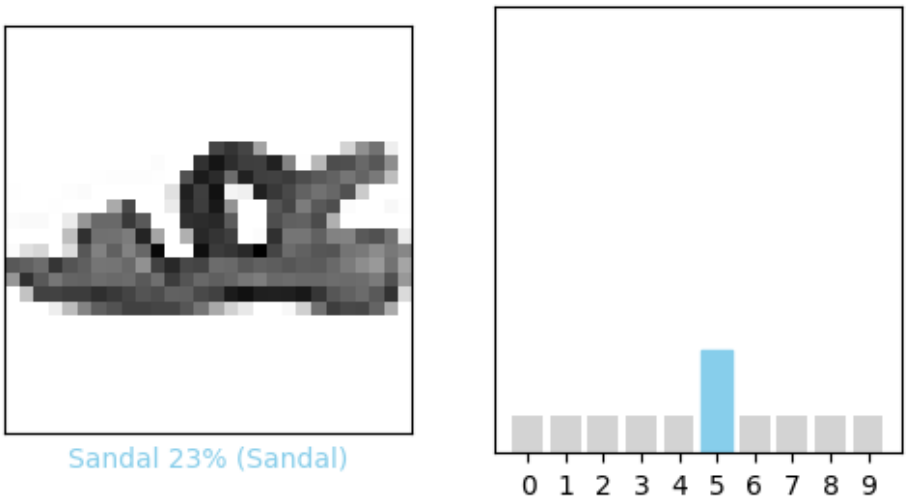


Fig. 1.15: SimpleNN Prediction

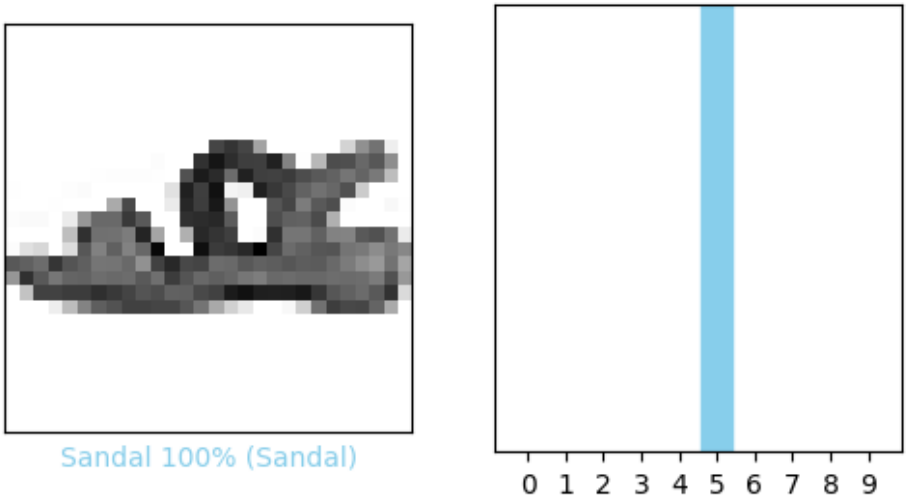


Fig. 1.16: CNN Prediction

Prediction of Class Bag

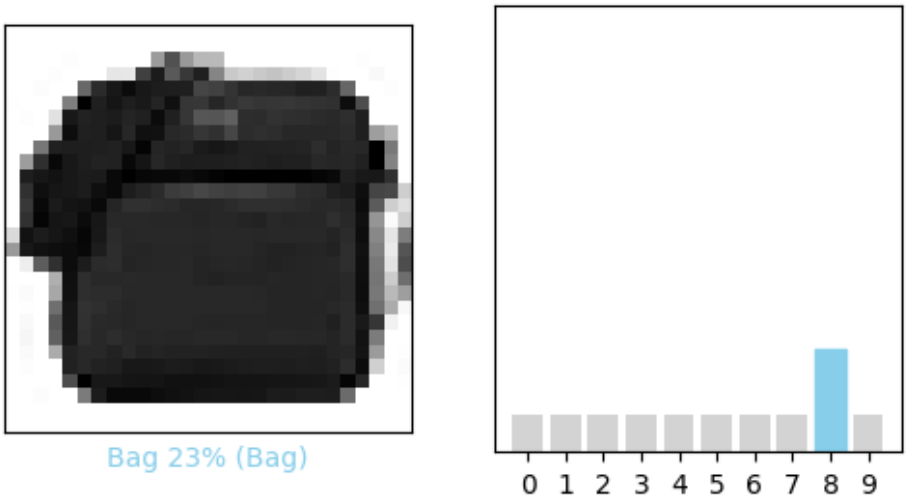


Fig. 1.17: SimpleNN Prediction

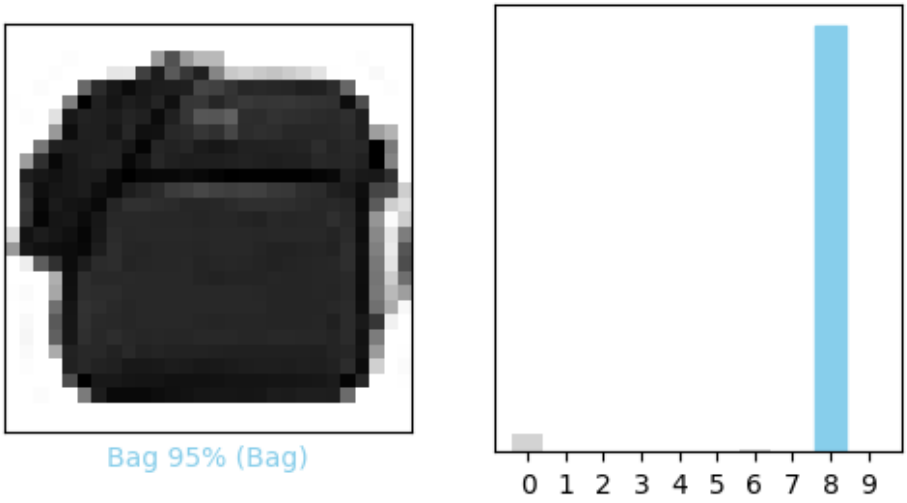


Fig. 1.18: CNN Prediction

Prediction of Class Tshirt

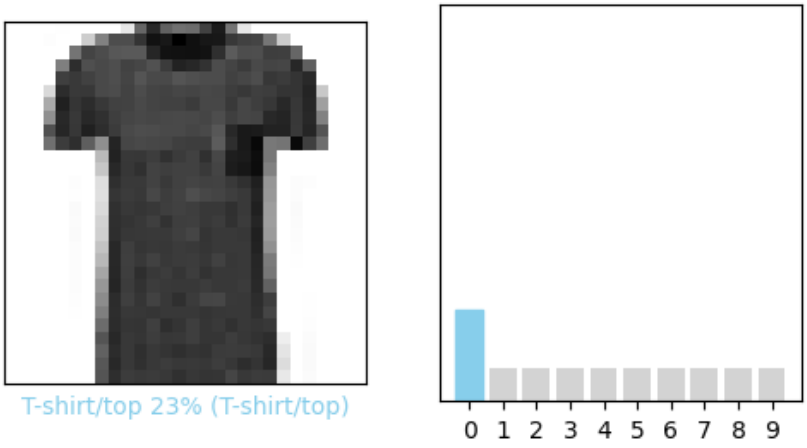


Fig. 1.19: SimpleNN Prediction

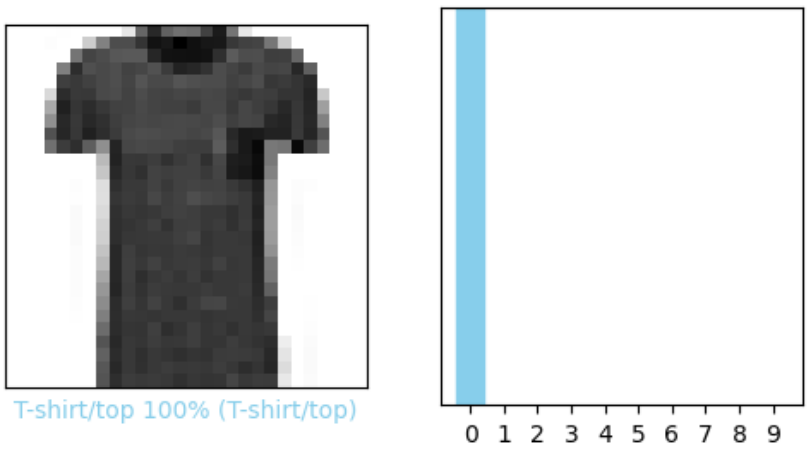


Fig. 1.20: CNN Prediction

The difference in testing accuracy between the simple neural network and the CNN model suggests that the CNN architecture is better suited for extracting meaningful features from image data compared to the simple neural network.

1.6 Semantic Segmentation

Semantic segmentation is a fundamental computer vision task that aims to categorize every single pixel in an image into a specific class or object. It goes beyond traditional image classification, which predicts a single dominant class for the entire image.

Semantic segmentation finds extensive use across multiple domains

Autonomous vehicles: Designed to recognise and comprehend surrounding things for safe movement, such as traffic signs, lanes, and pedestrians.

Medical imaging: By segmenting organs, tumours, and other structures for improved diagnosis, medical imaging can assist physicians in analysing medical images.

Robot Vision: Segmenting things for gripping, manipulating, or object recognition enables robots to see and interact with their surroundings.

Image editing: Based on segmentation results, it allows for automatic background removal or object extraction.

Comparing This Vision Task to Others

Image Classification: Classifies the entire image into a single category (e.g., landscape, dog, or cat). It does not specify the precise locations of things inside the picture.

Object Detection: Drawing bounding boxes around things in an image allows object detection to locate and identify them. It does not, however, discriminate between distinct instances of the same object class.

Semantic Segmentation: Provides a comprehensive pixel-by-pixel labelling for every object in the image, allowing for a more detailed understanding of the scene.

Semantic segmentation can provide difficulties because of things like

Occlusion: It might be challenging to segment things correctly when they are partially obscured.

Small Objects: The model must be able to capture minute details in order to segment small things.

Background Clutter: Accurate segmentation might be hampered by complex backdrops with textures or repeating patterns.

1.7 Deep Learning

Deep Learning: Inspired by the Brain

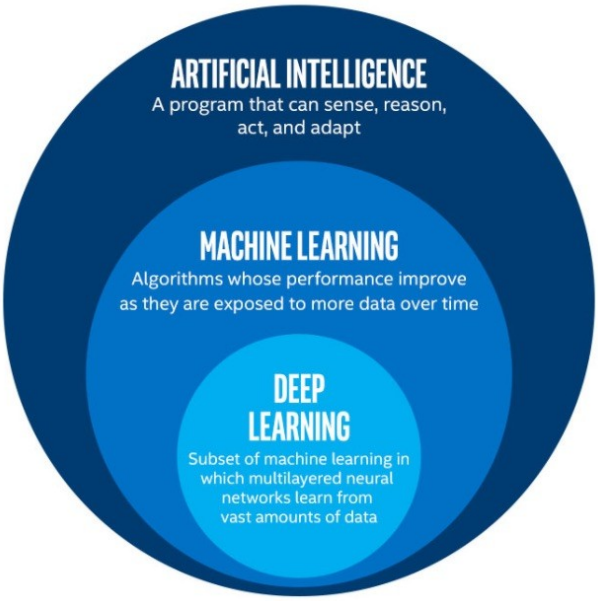


Fig. 1.21: The relationship between DL, ML and AI [11]

A branch of machine learning called "deep learning" is motivated by the composition and operations of the human brain. In order to interpret data and learn from it, it makes use of deep artificial neural networks.

1.7.1 Capabilities of DNNs

Extracting complicated patterns: Deep learning models can find complex correlations and characteristics in the data that simpler models would miss by processing the data through numerous levels of processing.

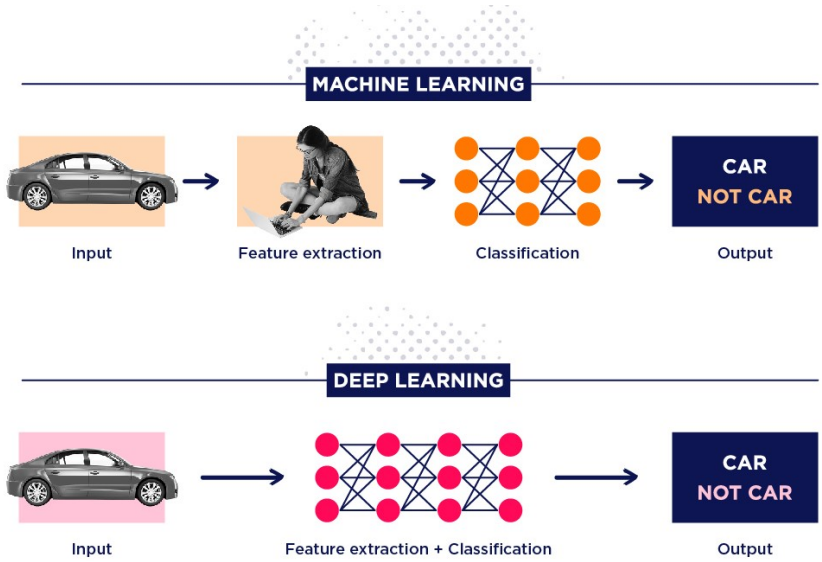


Fig. 1.22: Deep Learning vs Machine Learning [12]

Learning from big datasets: Massive data sets are ideal for deep learning. A deep learning model performs better on unseen data and can generalise more effectively the more data it has been trained on.

Automating feature extraction: Deep learning models have the ability to automatically extract meaningful features from the data, in contrast to standard machine learning techniques that frequently call for human feature engineering.

1.8 Generative Models

The goal of generative models, an interesting area of machine learning, is to produce new instances of data that closely resemble the training set of data. Generative models seek to understand the underlying distribution of the data and utilise that understanding to synthesise completely new samples, in contrast to discriminative models that categorise or forecast already-existing data.

Core Function

Consider a generative model that was developed using a sizable image dataset. After that, the model is able to produce brand-new photos with the same traits and aesthetics as the training set. These produced visuals could be entirely unique works of art or reimaginings of preexisting subjects.

There are a wide range of applications for generative models in many disciplines, such as: Image generation is the process of producing realistic or artistic images for data augmentation, product design, or creative purposes. Text generation is the process of creating logical and realistic text formats, such as scripts, code, poetry, or even conversational discourse for chatbots. Composing new works of music in the style or genre of another person. Drug development involves modelling and finding novel compounds with the right characteristics.

Different Generative Model Types

There are various generative modelling techniques, and each has advantages and disadvantages of its own. These are a few typical kinds:

Variational Autoencoders (VAEs): VAEs learn to decode the latent representation, or lower-dimensional space, created by encoding the input data, and subsequently reconstruct the original data. Through latent space manipulation, VAEs can produce new versions of the data.

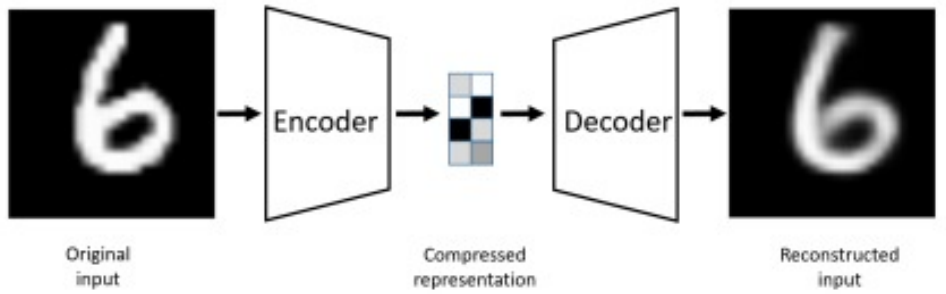


Fig. 1.23: An autoencoder example. The input image is encoded to a compressed representation and then decoded [13]

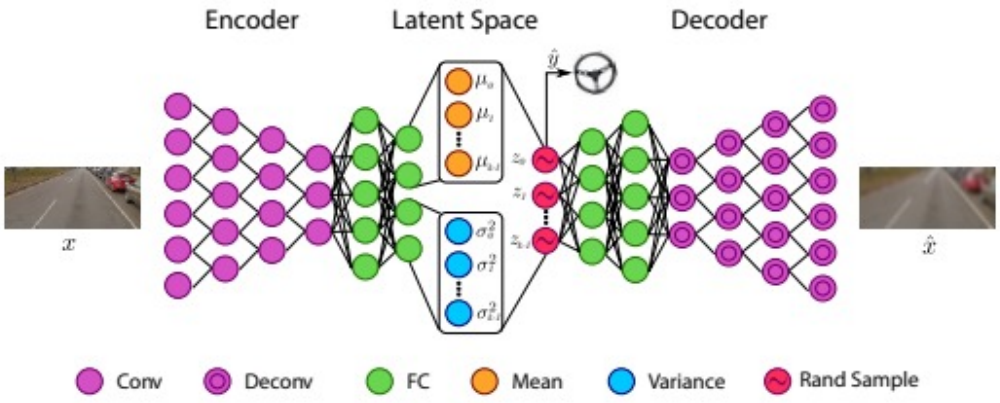


Fig. 1.24: VAE Architecture [14]

Generative Adversarial Networks or (GANs): are neural networks that compete with one another. While the discriminator network attempts to discern between created samples and actual data, the generator network generates new samples of data. The generator is forced by this adversarial training process to provide ever more realistic data that can deceive the discriminator.

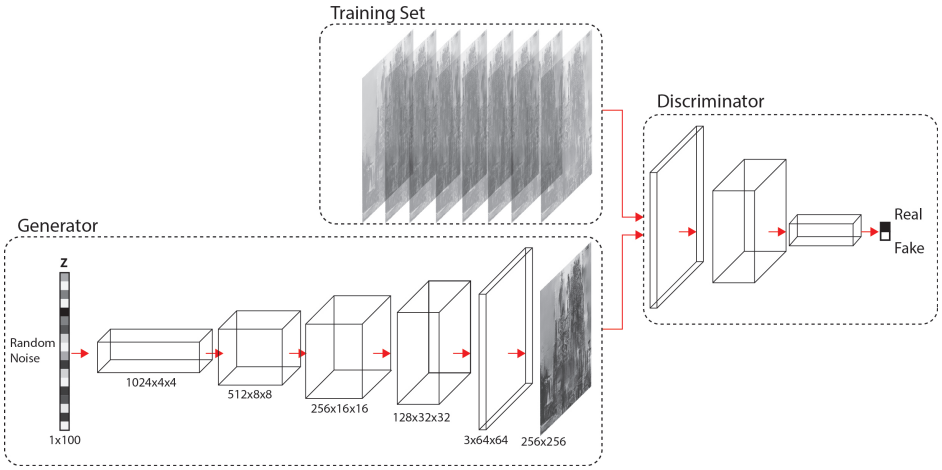


Fig. 1.25: Generative Adversarial Networks Architecture [15]

Autoregressive Models: These models produce data piece by piece, predicting the subsequent element in the sequence based on the previously created items. This method is frequently applied to problems involving text generation.

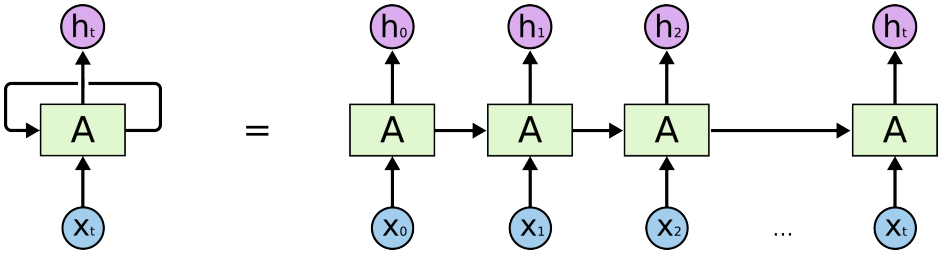


Fig. 1.26: Autoregressive Models in Deep Learning [16]

2. LITERATURE REVIEW

2.1 GAIA-1: A Generative World Model for Autonomous Driving

GAIA-1[25]is a revolutionary generative world model for autonomous driving that is presented in this study. The goal of generative world models is to comprehend environmental dynamics and forecast future occurrences by utilising historical data from observations and activities.

Crucial Elements of GAIA-1

Multimodal Inputs: GAIA-1 can create driving scenarios based on different cues and data by processing text, video, and action inputs.

Fine-Grained Control: By giving users the ability to modify the generated scenarios' ego-vehicle behaviour and scene elements, they can specifically explore a variety of driving scenarios.

Unsupervised Learning: GAIA-1 acquires knowledge through unsupervised learning, which eliminates the need for explicitly labelled training data. This facilitates training and condition adaptation for a variety of driving scenarios.

Advantages of GAIA-1

Enhanced Training: Compared to the limited amount of real-world data, GAIA-1 can produce realistic and varied driving situations that can be utilised to train autonomous driving models more successfully.

Scenario exploration: It makes it easier to identify and address possible flaws or failures by enabling researchers and engineers to investigate and test autonomous driving systems in a variety of simulated scenarios.

Generalisation: By gaining knowledge from a variety of created scenar-

ios, autonomous driving models may be better able to adapt to new circumstances. All things considered, GAIA-1 is a major step forward in the field of autonomous driving since it offers a strong tool for creating a variety of controlled and varied driving scenarios, which makes it easier to design and test trustworthy and durable autonomous vehicles.

2.2 Autoencoders

Neural networks using autoencoder[26] architectures are used for unsupervised learning tasks, especially in data compression, feature learning, and dimensionality reduction. The basic principle of autoencoders is to encode incoming data into a latent space, from which the original data may be reconstructed, thus learning an efficient representation of the data. fundamental architecture

The encoder and decoder networks make up autoencoders. The decoder uses the latent space representation to recreate the original data, whereas the encoder maps input data into a lower-dimensional latent space. Reducing the reconstruction error between the input and the reconstructed output is the goal.

Autoencoder Types

The article covers a range of autoencoder architectures, such as:

Vanilla Autoencoders: Simple autoencoder architecture consisting of layers that are fully coupled.

VAEs or variational autoencoders: VAEs are probabilistic generative models that use variational inference to learn a latent space representation. They were first described in the publication "Auto-Encoding Variational Bayes".

Autoencoder variants trained to add noise to input and reconstruct the original clean data are known as **denoising autoencoders.**

Autoencoders that enforce a sparsity constraint on the latent space representation are known as **sparse autoencoders**, and they help the model acquire sparse representations.

Contractive autoencoders are autoencoders that ensure a smooth and continuous latent space by adding a penalty term to the loss function.

Adversarial Autoencoders: To develop a more structured and meaningful latent space, autoencoders are paired with adversarial training approaches.

Applications

The study examines the numerous uses of autoencoders in a variety of fields, including as generative modelling, text analysis, anomaly detection, image and video processing, and more. Autoencoders have shown effective in a variety of applications, including representation learning, image denoising, picture super-resolution, and image production.

Obstacles and Prospective Paths

The writers talk about the difficulties in training autoencoders, including mode collapse, vanishing gradients, and overfitting. They also point to possible avenues for future research, such as the creation of training algorithms with greater resilience, the investigation of novel architectures, and applications in transfer and reinforcement learning. All things considered, "Autoencoders" offers a thorough review of autoencoder models, including their designs, training methods, applications, and difficulties. Researchers and practitioners interested in learning about and applying autoencoder-based techniques in Machine learning and Artificial intelligence will find it to be a useful resource.

2.3 Variational Autoencoder for End-to-End Control of Autonomous Driving with Novelty Detection and Training De-biasing

[14]This work suggests a novel method for controlling autonomous driving by integrating training de-biasing and novelty detection into a **variational autoencoder (VAE).**

Important Points

End-to-End Control: The suggested approach does away with the requirement for extra modules or hand-crafted features by using a VAE to directly map sensory inputs (like camera images) to control outputs (like steering commands) for autonomous driving.

Novelty Detection: A technique for identifying unique and unexpected driving circumstances is incorporated into the VAE architecture. This enables the system to recognise situations that were not covered in training and maybe implement the necessary safety precautions.

Training De-biasing: The study tackles the problem of skewed training data, which could restrict the model's capacity to be broadly used. The authors suggest a method to remove bias from the training process and enhance the model's performance under various driving scenarios.

Advantages: The suggested method provides an end-to-end control solution, making the design of autonomous driving systems simpler and possibly enhancing their performance. Novelty detection helps make autonomous cars more resilient and safe by allowing them to respond properly to unforeseen circumstances.Training de-biasing produces a more generalizable model that can handle a variety of real world events by reducing the impact of biased training data.

Overall, by using VAE's capabilities for effective representation learning, adding novelty detection for increased safety, and resolving potential biases in training data for better generalizability, this work offers a promising method for autonomous driving control.

2.4 Auto-Encoding Variational Bayes

The **Variational Autoencoder (VAE)** [**27**], a potent generative model and an expansion of the conventional autoencoder architecture, is introduced in this study. Below is a thorough synopsis of the main ideas discussed in the paper:

By learning a mapping from input data to a latent space and back again, traditional autoencoders learn how to reconstruct data. By treating the latent space as a probability distribution, VAEs expand on this idea. To create new data samples and map observed data into a latent space, VAEs employ probabilistic inference.

Objective

In this study, the Evidence Lower Bound (ELBO), a novel objective function for training VAEs, is presented. The two components of ELBO are the KL divergence between the estimated posterior and the prior distribution over the latent variables, and the reconstruction error, which assesses how successfully the model reconstructs the input data.

Model Building

The three components of VAEs are a reparameterization trick, a decoder network, and an encoder network. The encoder network associates input data with a latent space distribution's parameters. Using a differentiable transformation, the reparameterization approach takes samples from the latent space. From latent space representations, the decoder network produces data samples.

Training Procedure

The ELBO objective function is optimised using stochastic gradient descent (SGD) or related methods to train VAEs. Backpropagation is used to maximise the ELBO with regard to the model parameters.

Applications

VAEs can be applied to a number of tasks, such as representation learning, data reduction, and image production. They offer an ethical approach to semi-supervised and unsupervised learning tasks.

Evaluation

The study assesses the effectiveness of VAEs using a range of datasets, showing that they can produce accurate samples and discover significant latent representations.

Comparing This Model with Others

The benefits of VAEs over conventional autoencoders and other generative models, such Generative Adversarial Networks (GANs), are covered in the study.

Extensions and Additional Study

The authors suggest a number of expansions and areas for more study, such as conditional and hierarchical VAEs as well as applications in natural language processing. In conclusion, **"Auto-Encoding Variational Bayes"** offers a unique paradigm for variational inference-based generative model training. The work has greatly influenced probabilistic modelling and deep learning, spurring more study and developments in generative modelling methods.

2.5 Generative Adversarial Networks

In Ref. [22], a novel framework for training generative models Generative Adversarial Networks is introduced. Below is a thorough synopsis of the main ideas covered in the paper:

Overview of Generative Models

In order to produce fresh samples that closely resemble the training set, generative models seek to understand the underlying distribution of the data. The scalability and sample quality of traditional generative models, such Variational Autoencoders (VAEs) and Restricted Boltzmann Machines (RBMs), are limited.

GANs or Generative Adversarial Networks

The generator and discriminator neural networks make up a GAN. In order to provide samples that are identical to actual data, the generator learns how to generate samples from random noise. The discriminator gains the ability to discriminate between produced and genuine samples. In the training process, the generator and discriminator play a minimax game in

which they are trained iteratively, with the generator trying to trick the discriminator and vice versa.

Objective Function: Maximising the likelihood that genuine and generated samples will be correctly labelled is the discriminator's goal. The generator's goal is to reduce the likelihood that produced samples are fabricated in the discriminator's eyes. One way to formulate the training process is as a minimax optimisation problem.

Algorithm for Training: The training procedure for GANs, which alternates between updating the generator and the discriminator in each iteration, is described by the authors. Using binary cross-entropy loss and standard backpropagation, the discriminator is trained. The generator's training goal is to maximise the log chance that one of the discriminator's outputs is incorrect.

Experiments and Results: The study provides experimental findings showing how well GANs generate realistic samples in a variety of domains, such as text, audio, and images. GANs are demonstrated to surpass earlier generative models in terms of sample quality and diversity, yielding high-quality examples that closely mirror the training data.

Discussion and Future Directions: The writers go on a number of GAN extensions and uses, such as style transfer, image-to-image translation, and semi-supervised learning. They also draw attention to issues such mode collapse, instability during training, and assessment measures that are used to gauge how well generated samples are performing. In conclusion, **"Generative Adversarial Nets"** presents the GAN framework as a potent method for generative model training. The article covers the training algorithm, experimental findings, theoretical underpinnings of GANs, and potential future avenues for generative modelling research.

3. RESULTS

3.1 Running a Pre-trained Model

YOLOv5 (You Only Look Once version 5) [28] is a state-of-the-art object detection model known for its following properties.

Speed and Accuracy: It offers a good balance between real-time performance and high detection accuracy, making it suitable for various applications.

Scalability: YOLOv5 comes in different versions (s, m, l, x) with increasing complexity and accuracy, allowing you to choose the one that best suits your resource constraints and performance needs.

Ease of Use: Compared to some other object detection models, YOLOv5 is considered relatively easy to set up, train, and deploy.

Here I tried to run this pre-trained above mentioned model called YOLOv5 which is used for object detection tasks.

Some of the test results

Original Example Image 1

Fig. 3.1: Example 1 tomassereda/istock/getty images

Detected Image

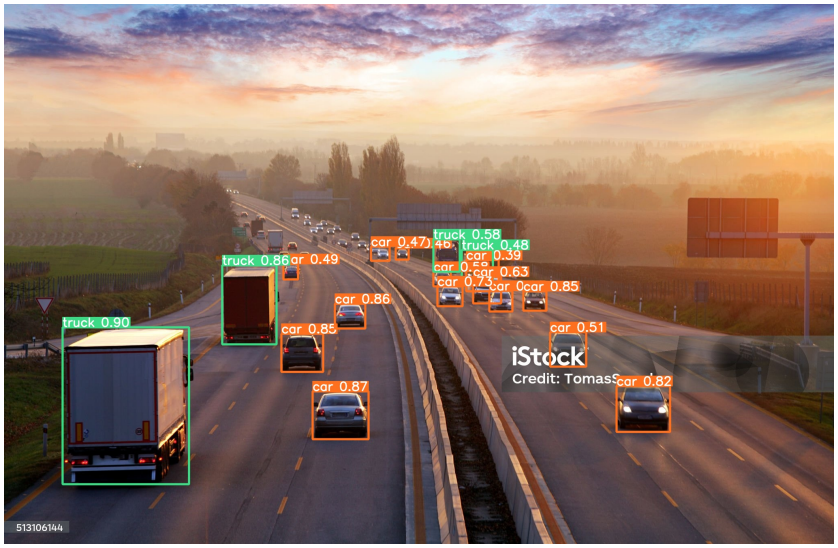


Fig. 3.2: Example 1 - Detected Image

Original Example Image 2

Fig. 3.3: Example 2 [17]

Detected Image

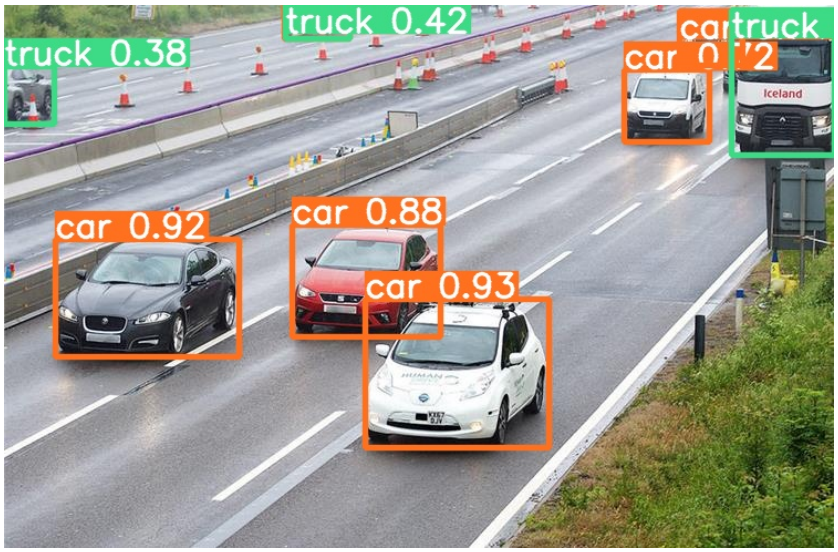


Fig. 3.4: Example 2 - Detected Image

Original Example Image 3

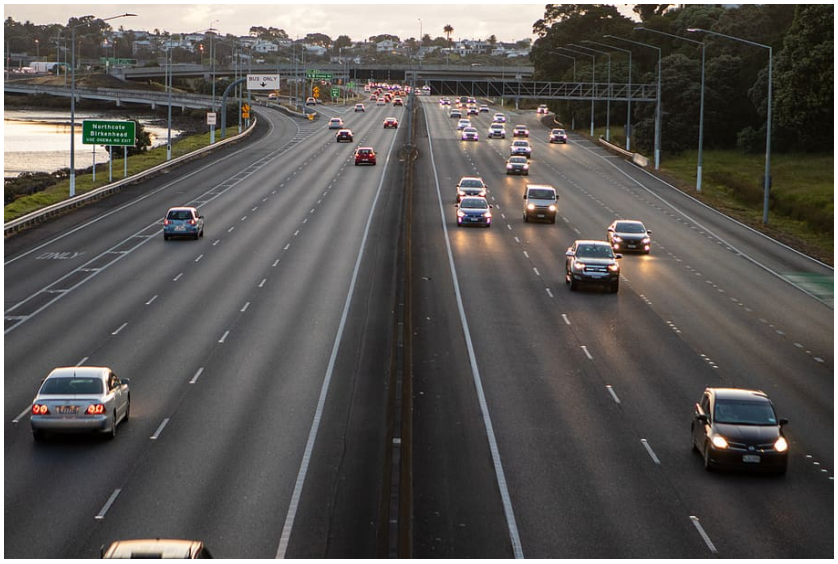


Fig. 3.5: Example 3 [18]

Detected Image

Fig. 3.6: Example 3 - Detected Image

3.2 Training and Testing of U-Net Model

U-Net Architecture

U-NET is a convolutional neural network architecture. It is a fully convolutional neural network that is designed to learn from fewer training samples.

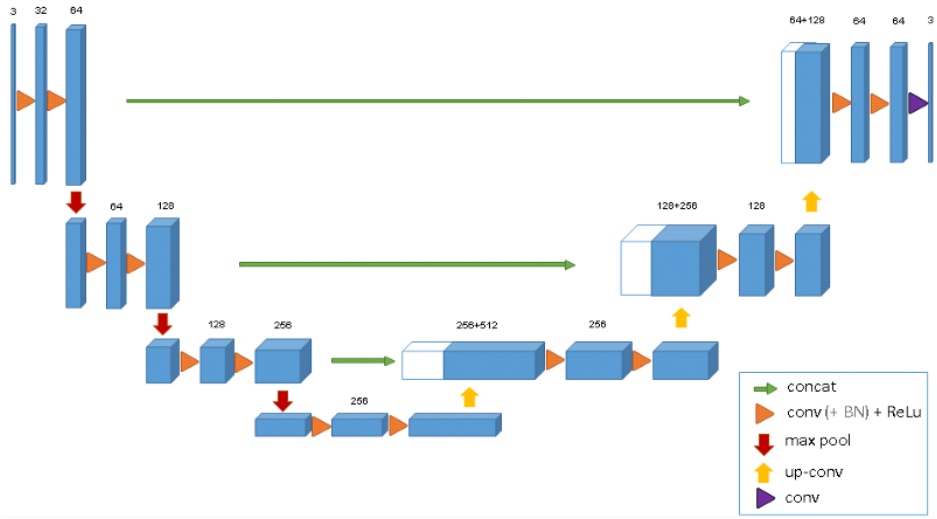


Fig. 3.7: U-Net Architecture [19]

A critical aspect of U-Net is the presence of skip connections between the contracting and expansive paths at corresponding levels. These connections directly copy feature maps from the encoder to the decoder with the same resolution.

Skip connections help the decoder preserve spatial details lost during down sampling in the encoder. This allows the decoder to generate more accurate segmentation maps.

Training loss and Training Accuracy

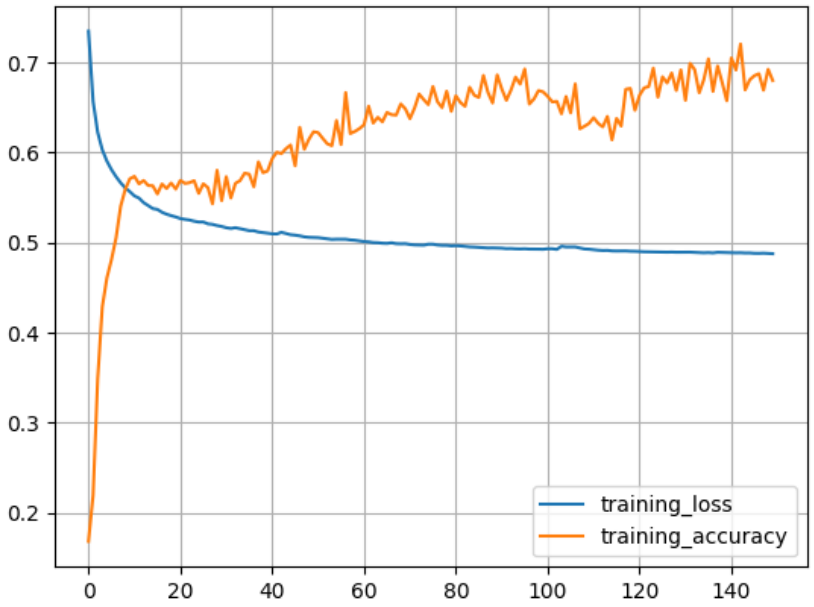


Fig. 3.8: Training loss and Training Accuracy

U-Net Model Predictions

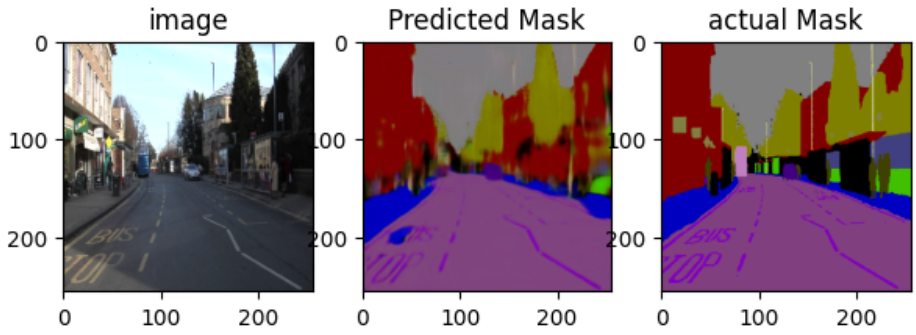


Fig. 3.9: U-Net Model Predictions Example 1

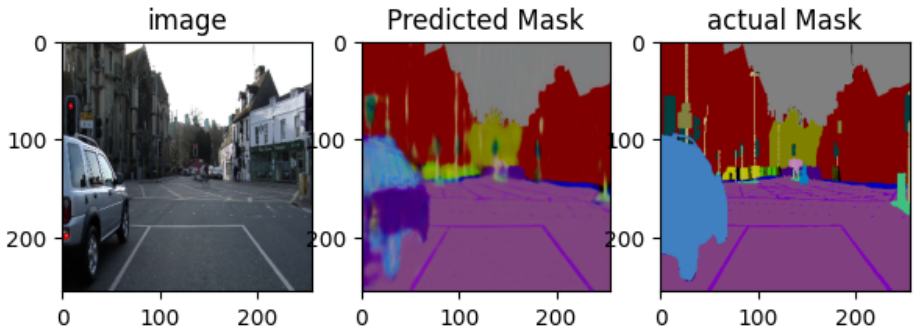


Fig. 3.10: U-Net Model Predictions Example 2

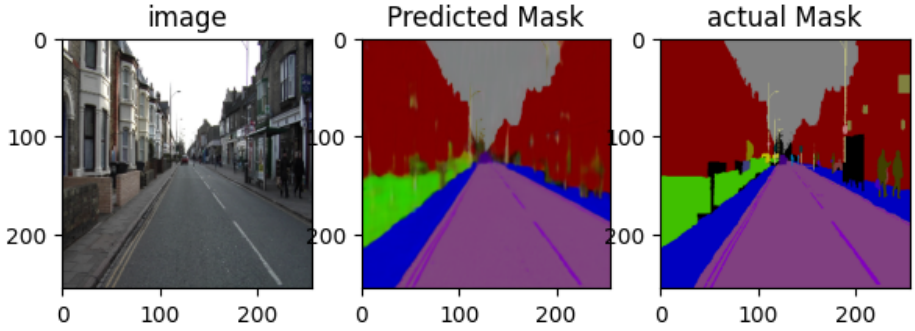


Fig. 3.11: U-Net Model Predictions Example 3

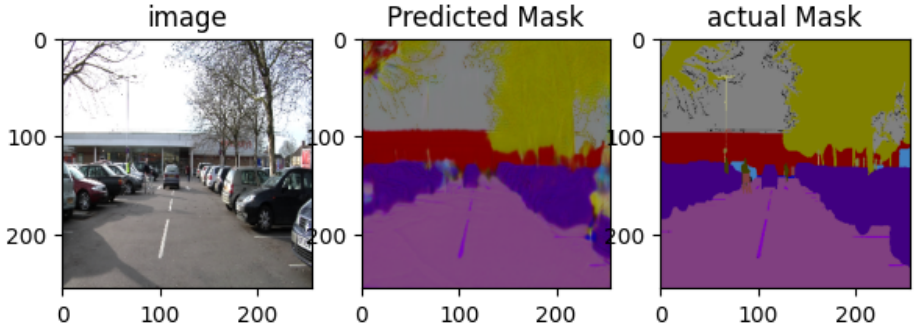


Fig. 3.12: U-Net Model Predictions Example 4

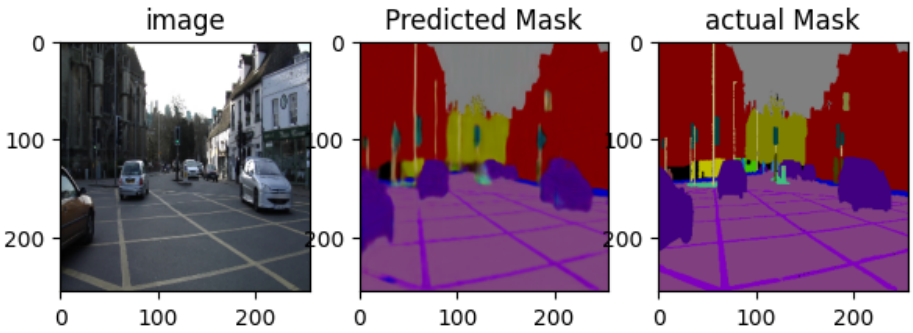


Fig. 3.13: U-Net Model Predictions Example 5

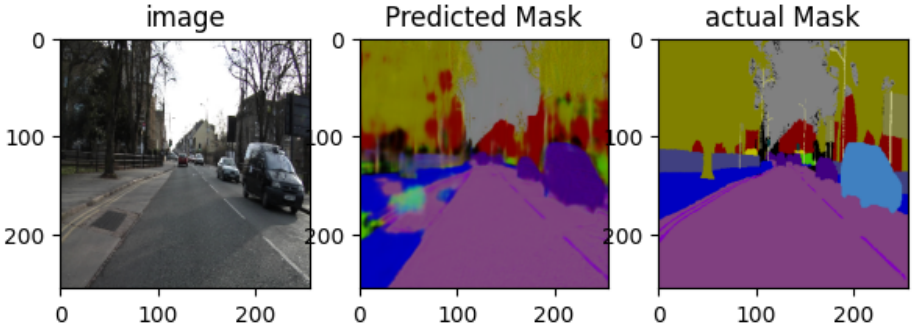


Fig. 3.14: U-Net Model Predictions Example 6

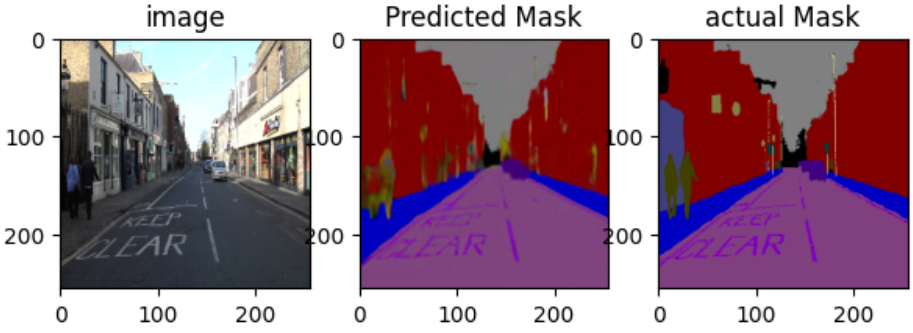


Fig. 3.15: U-Net Model Predictions Example 7

4. DISCUSSIONS AND CONCLUSIONS

This project investigates the potential of Generative Artificial Intelligence (AI) in revolutionizing future autonomous vehicles (AVs).

Focus: Explores how Generative AI can contribute to the advancement of autonomous driving technology.

Literature Review: Analyzes relevant research on Generative AI applications in AVs, including studies on autoencoders, generative adversarial networks (GANs), and the "GAIA-1" model.

Implementations

YOLOv5: Demonstrates its effectiveness in Object detection tasks crucial for AVs.

Simple ANN: Implemented for handwritten digit classification (MNIST dataset) to introduce deep learning fundamentals.

CNN: Implemented with the Fashion-MNIST dataset to showcase convolutional neural networks.

Pre-trained U-Net: Highlights the power of pre-trained models for image segmentation applications.

Applications in Autonomous Driving

Data Generation: Generative AI can create realistic and varied training data for simulating diverse driving scenarios, enabling AVs to handle unexpected situations.

Object Recognition and Behavior Prediction: Model training with Generative AI can improve object recognition and prediction of their behavior, enhancing AV perception capabilities.

Overall: This project lays the groundwork for further exploration of Generative AI applications in autonomous vehicles by showcasing the potential of various deep learning architectures for related tasks.

Future Research Directions: The project paves the way for further research on:

- Integrating generative AI models into AV systems.
- Utilizing generative models for specific AV tasks like data augmentation.
- Exploring advanced deep learning architectures for robust object recognition and behavior prediction in AVs.

By building upon these findings, researchers can advance the development of safe, reliable, and intelligent autonomous vehicles.

BIBLIOGRAPHY

[1] *ML Paradigms, https://krisnamustika.blogspot.com/2018/03/sentiment-analysis.html.*

[2] *The Basic Perceptron Model, https://gamedevacademy.org/perceptrons-the-first-neural-networks/.*

[3] *Artificial Neural Network, https://tecnobits.com/en/que-son-las-redes-neuronales-artificiales/.*

[4] *Gradient Descent, https://www.researchgate.net/figure/Gradient-Descent-Algorithm-26_fig1_352019480.*

[5] *Backpropagation, https://www.linkedin.com/pulse/guided-backpropagation-kodi-jahnavi/.*

[6] *NN Architecture, https://hackaday.com/2017/11/13/tiny-tensor-brings-machine-deep-learning-to-micros/.*

[7] *CNN Architecture, https://www.linkedin.com/pulse/decoding-cnn-architecture-unveiling-power-precision-neural-moustafa/.*

[8] *Convolutional Layer, https://www.mobiquity.com/insights/introduction-to-convolutional-neural-networks.*

[9] *Pooling Layer, https://www.researchgate.net/figure/4-Pooling-layer-operation-Left-a-4x4-matrix-input-Right-2x2-pooling-stride-2-Max_fig4_371987154.*

[10] *Our CNN Architecture, https://www.researchgate.net/figure/A-vanilla-Convolutional-Neural-Network-CNN-representation_fig2_339447623.*

[11] *DL,ML and AI, https://www.semanticscholar.org/paper/Using-Artificial-Intelligence-for-Quantifying-Diab/24799be85fe075e78a2e5ea50d4db80b78ae580a/figure/0.*

[12] *DL vs ML ,https://www.analyticsvidhya.com/blog/2023/04/machine-learning-for-social-media/.*

[13] *Autoencoder, https://tech.skit.ai/feature-disentanglement1/.*

[14] Alexander Amini, Wilko Schwarting, Guy Rosman, Brandon Araki, Sertac Karaman, and Daniela Rus. Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training debiasing. pages 568–575, 10 2018.

[15] *GAN, https://www.computationalarchitecturelab.org/generative-adversarial-networks-and-plan-stylistic-plan-generation.*

[16] *AUTOREG, https://stackoverflow.com/questions/58449353/lstm-deal-with-multiple-rows-in-a-date.*

[17] *EX2, https://www.autocar.co.uk/car-news/features/autonomous-car-trials-are-they-smart-or-reckless.*

[18] *EX3, https://pixabay.com/photos/traffic-highway-freeway-movement-4639317/.*

[19] *U-Net, https://github.com/asimonov/CarND-VehicleDetection-Unet.*

[20] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52:99–115, 1990.

[21] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv: 1312.6114*, 2013.

[22] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv: 1406.2661*, 2014.

[23] Corinna Cortes and Vladimir Naumovich Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[24] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, L. Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[25] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. GAIA-1: A Generative World Model for Autonomous Driving. *arXiv: 2309.17080*, 2023.

[26] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *arXiv: 2003.05991*, 2021.

[27] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv: 1312.6114*, 2022.

[28] *You Only Look Once version 5, https://github.com/ultralytics/yolov5.*