

Deep Generative Models (AI314)

Master of Science in AI



Dr. Nibras Abdullah

*Assistant professor
Faculty of Informatics
Midocean University*



Data for Deep Learning



Deep Neural Networks



Multilayer Perceptron (MLP)



Convolutional Neural Network (CNN)



Data for Deep Learning

Deep learning has revolutionized the field of artificial intelligence, allowing machines to learn and make decisions in ways that mimic the human brain. At the heart of this revolution is the data that powers these deep neural networks. In this presentation, we will explore the key aspects of data for deep learning, from the fundamentals of neural networks to the practical applications of TensorFlow and Keras.

Data for Deep Learning

Many types of machine learning algorithms require structured, tabular data as input, arranged into columns of features that describe each observation. For example, a person's age, income, and number of website visits in the last month are all features that could help to predict if the person will subscribe to a particular online service in the coming month.

Structured data

ID	Age	Gender	Height (cm)	Location
0001	54	M	186	London
0002	35	F	166	New York
0003	62	F	170	Amsterdam
0004	23	M	164	London
0005	25	M	180	Cairo
0006	29	F	181	Beijing
0007	46	M	172	Chicago

Unstructured data



Images



Audio

This service is terrible!

Your website is great!

Text

Unstructured data refers to any data that is not naturally arranged into columns of features, such as images, audio, and text. There is of course spatial structure to an image, temporal structure to a recording or passage of text, and both spatial and temporal structure to video data, but since the data does not arrive in columns of features, it is considered unstructured, as shown in Figure 2-

⊗ Data for Deep Learning



⊗ Deep Neural Networks



Multilayer Perceptron (MLP)



Convolutional Neural Network (CNN)



Deep Neural Network

The majority of deep learning systems are artificial neural networks (ANNs, or just neural networks for short) with multiple stacked hidden layers. For this reason, deep learning has now almost become synonymous with deep neural networks. However, any system that employs many layers to learn high-level representations of the input data is also a form of deep learning (e.g., deep belief networks).

Let's start by breaking down exactly what we mean by a neural network and then see how they can be used to learn high-level features from unstructured data.



What Is a Neural Network?



1

Inspiration from the Brain

Neural networks are inspired by the structure and function of the human brain, with interconnected nodes (neurons) that process and transmit information.

2

Multilayered Architecture

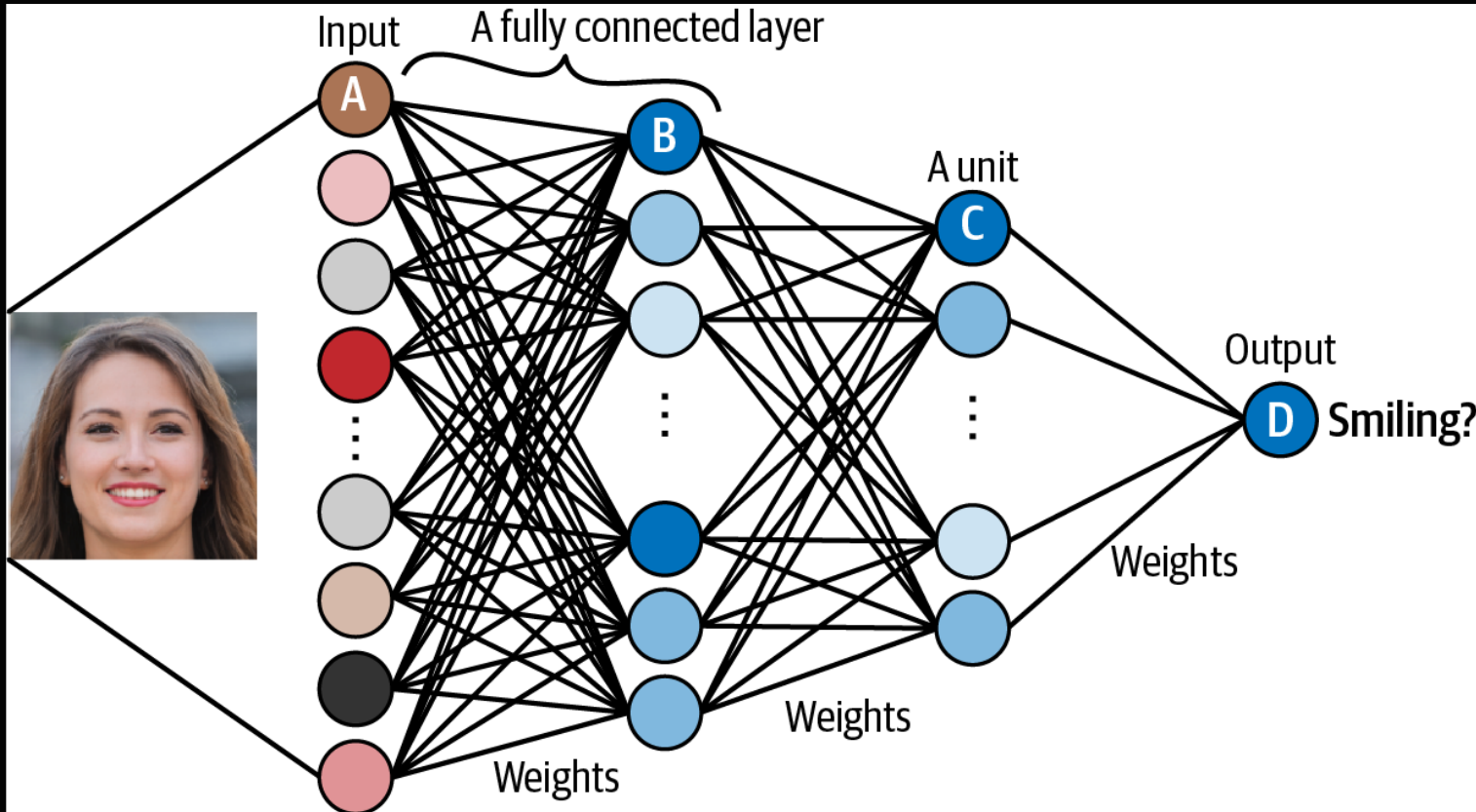
Neural networks typically consist of an input layer, one or more hidden layers, and an output layer, allowing them to learn complex patterns in data.

3

Powerful Learning Capabilities

By adjusting the connections (weights) between neurons, neural networks can learn to recognize patterns and make predictions, making them highly versatile for a wide range of applications.

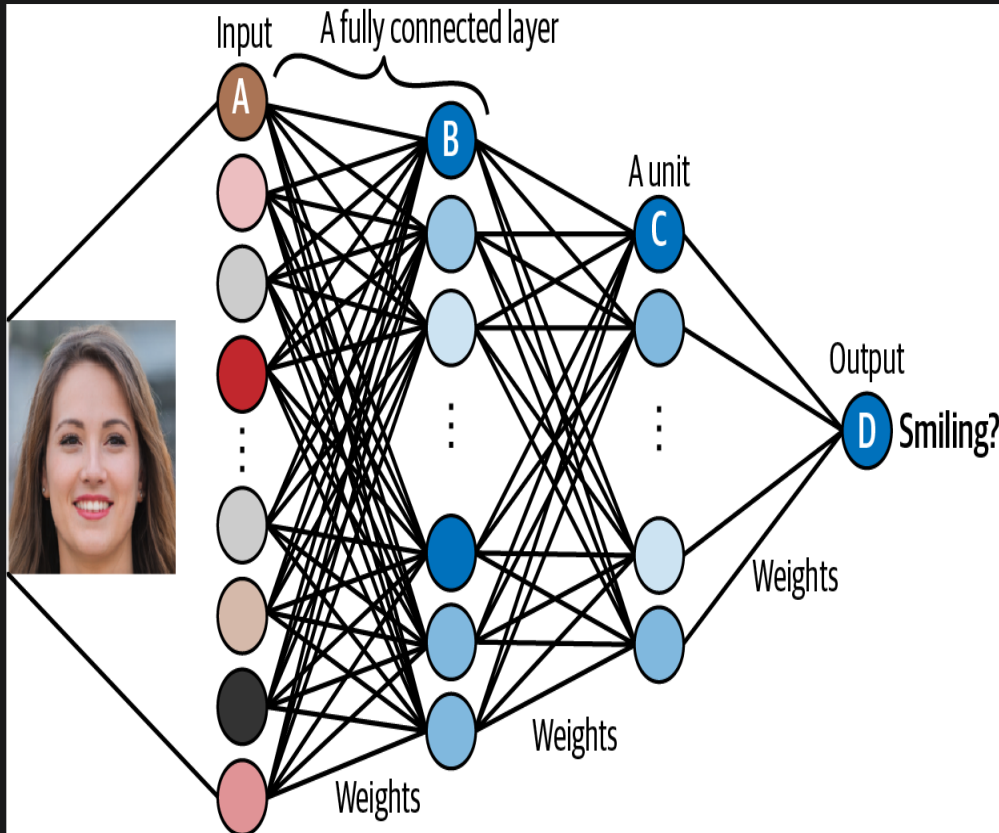
What Is a Neural Network?



A neural network consists of a series of stacked layers. Each layer contains units that are connected to the previous layer's units through a set of weights. As we shall see, there are many different types of layers, but one of the most common is the fully connected (or dense) layer that connects all units in the layer directly to every unit in the previous layer. Neural networks where all adjacent layers are fully connected are called multilayer perceptrons (MLPs). This is the first type of neural network that we will study

Figure 2-2. An example of a multilayer perceptron that predicts if a face is smiling

What Is a Neural Network?



During the training process, batches of images are passed through the network and the predicted outputs are compared to the ground truth. For example, the network might output a probability of 80% for an image of someone who really is smiling and a probability of 23% for an image of someone who really isn't smiling. A perfect prediction would output 100% and 0% for these examples, so there is a small amount of error. The error in the prediction is then propagated backward through the network, adjusting each set of weights a small amount in the direction that improves the prediction most significantly.

This process is appropriately called backpropagation. Gradually, each unit becomes skilled at identifying a particular feature that ultimately helps the network to make better predictions.

Figure 2-2. An example of a multilayer perceptron that predicts if a face is smiling

Learning High-Level Features

Feature Extraction

Neural networks excel at automatically extracting high-level features from raw data, such as edges, shapes, and textures in images, or patterns in time-series data.

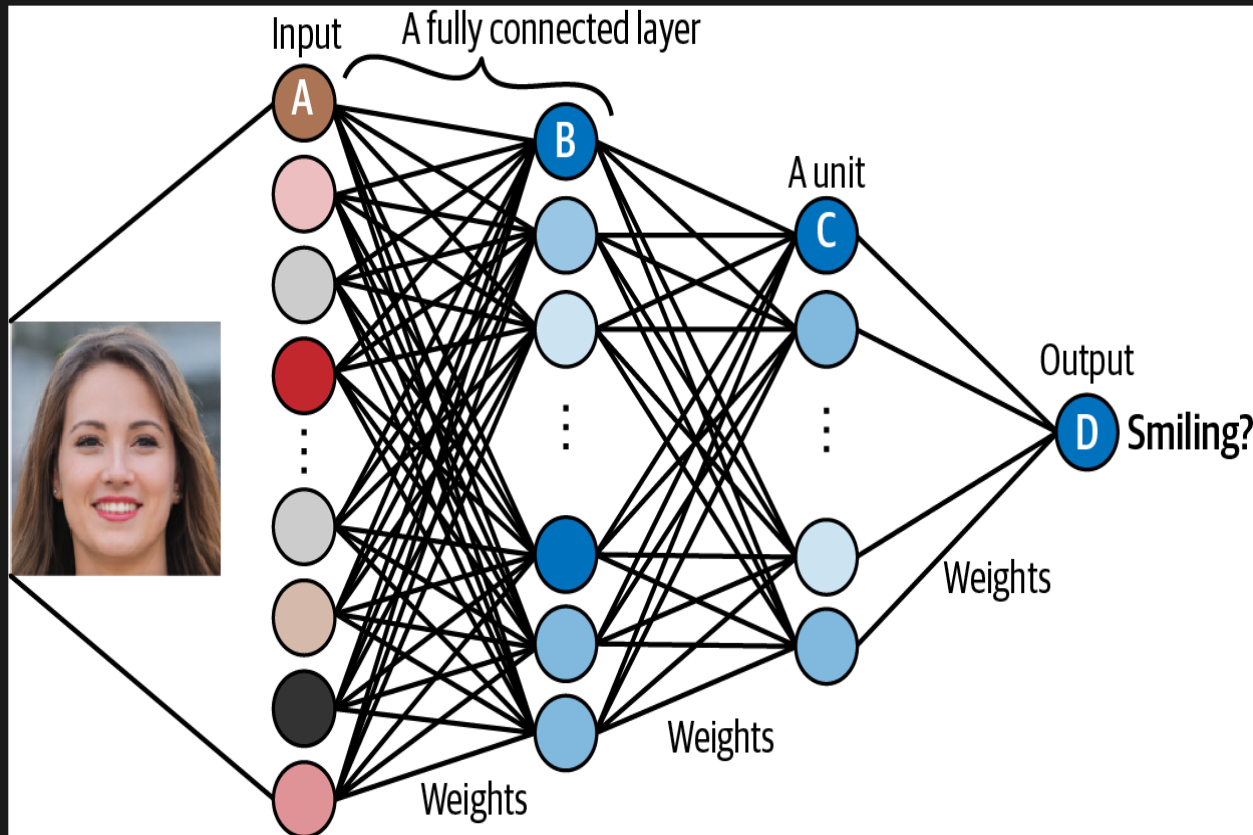
Hierarchical Learning

The deeper the network, the more complex the features it can learn, allowing it to build a hierarchical understanding of the input data.

Unsupervised Pre-training

Techniques like unsupervised pre-training can help neural networks learn useful features even when labeled data is scarce, making them more efficient and robust.

Learning High-Level Features



1. Unit A receives the value for an individual channel of an input pixel.
2. Unit B combines its input values so that it fires strongest when a particular lowlevel feature such as an edge is present.
3. Unit C combines the low-level features so that it fires strongest when a higherlevel feature such as teeth are seen in the image.
4. Unit D combines the high-level features so that it fires strongest when the person in the original image is smiling.

TensorFlow and Keras



1

TensorFlow

TensorFlow is an open-source machine learning framework developed by Google, providing a powerful and flexible environment for building and deploying deep learning models.

TensorFlow and Keras



1

TensorFlow

TensorFlow is an open-source machine learning framework developed by Google, providing a powerful and flexible environment for building and deploying deep learning models.

2

Keras

Keras is a high-level neural networks API that runs on top of TensorFlow, making it easier to quickly prototype and train deep learning models with minimal code.

TensorFlow and Keras



1

TensorFlow

TensorFlow is an open-source machine learning framework developed by Google, providing a powerful and flexible environment for building and deploying deep learning models.

2

Keras

Keras is a high-level neural networks API that runs on top of TensorFlow, making it easier to quickly prototype and train deep learning models with minimal code.

3

Synergistic Integration

The combination of TensorFlow's low-level control and Keras's user-friendly interface allows developers to build and deploy complex deep learning models with greater efficiency and productivity.

Supervised Learning with Neural Networks

1

Input Data

The first step in supervised learning is to provide the neural network with labeled input data, such as images with corresponding class labels.

2

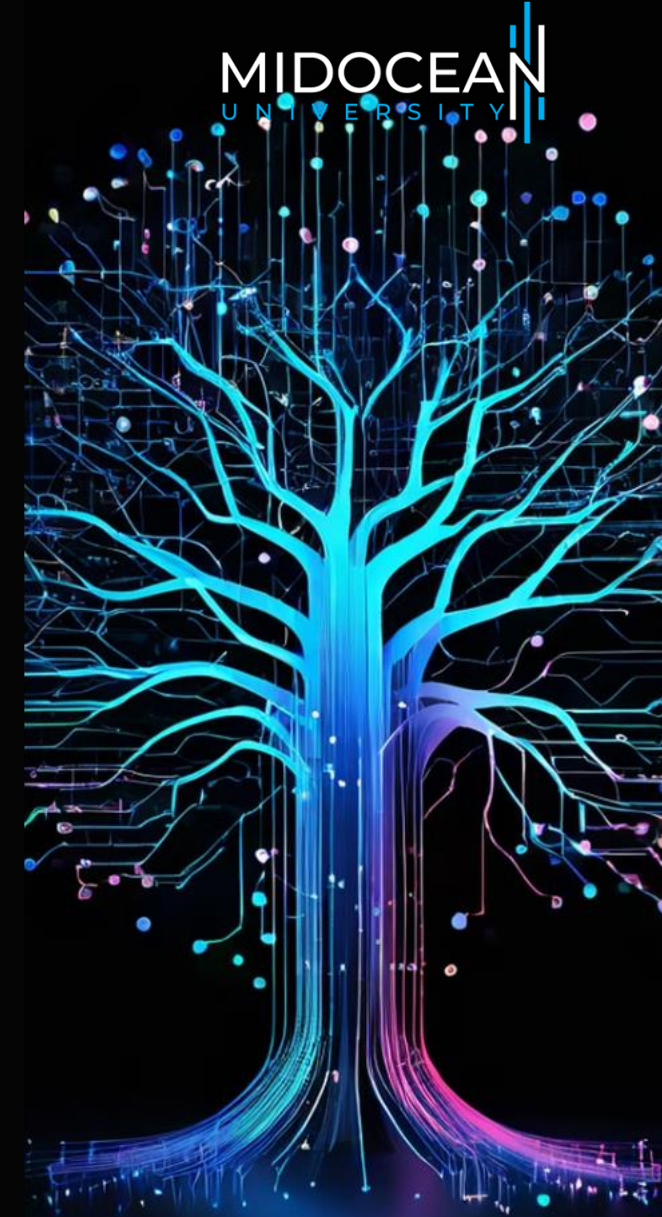
Forward Propagation

The input data is then fed through the neural network, which makes predictions based on the current weights and biases of the network.

3

Backpropagation

The network's predictions are compared to the true labels, and the error is backpropagated through the network to update the weights and biases, enabling the network to learn and improve its performance.



Unsupervised Learning with Neural Networks



Clustering

Unsupervised neural networks can group similar data points together, revealing underlying patterns and structures without the need for labeled data.



Anomaly Detection

Neural networks trained in an unsupervised manner can identify outliers or anomalies in data, which can be useful for tasks like fraud detection or system monitoring.



Dimensionality Reduction

Unsupervised techniques like autoencoders can learn efficient representations of high-dimensional data, allowing for dimensionality reduction and visualization.

Applications of Deep Learning

Computer Vision

Deep learning excels at tasks like image recognition, object detection, and semantic segmentation, with widespread applications in fields like healthcare, autonomous vehicles, and security.

Natural Language Processing

Neural networks can be used for language understanding, translation, text generation, and conversational AI, powering applications such as chatbots, virtual assistants, and machine translation.

Time Series Analysis

Deep learning models can capture complex patterns in time-series data, enabling applications in areas like financial forecasting, predictive maintenance, and weather prediction.

Generative Models

Techniques like Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) allow neural networks to generate new, realistic data, opening up possibilities in areas like creative content generation and synthetic data augmentation.

The Future of Deep Learning

Advances in Hardware	Continued improvements in GPU and TPU hardware will enable more powerful and efficient deep learning models.
Explainable AI	Techniques for making deep learning models more interpretable and transparent will improve trust and adoption.
Reinforcement Learning	Combining deep learning with reinforcement learning will lead to more autonomous and adaptive systems.
Quantum Computing	Breakthroughs in quantum computing have the potential to significantly accelerate deep learning algorithms.

⊗ Data for Deep Learning



⊗ Deep Neural Networks



⊗ Multilayer Perceptron (MLP)



Convolutional Neural Network (CNN)



An abstract visualization of a complex network, likely representing a neural network. It features a dense web of interconnected nodes and edges. The nodes are represented by small, glowing spheres in various colors (blue, green, yellow, orange, red) against a dark background. The edges are thin, glowing lines that connect the nodes, creating a complex, web-like structure. The overall effect is one of dynamic energy and interconnectedness.

Multilayer Perceptron (MLP)

The Multilayer Perceptron (MLP) is a type of artificial neural network that is widely used in machine learning and data analysis. It consists of multiple layers of interconnected nodes, each layer transforming the input data into a more abstract representation. This powerful architecture allows MLPs to learn complex non-linear relationships in data, making them a versatile tool for a variety of applications.

A hand is visible on the left side of the image, using a mouse on a laptop. The background is a dark, futuristic digital space with glowing green and blue lines and patterns, resembling a complex circuit or data flow. The overall color palette is dominated by dark blues, purples, and bright greens.

Compiling the Model

1

Define the Model Architecture

The first step in compiling an MLP model is to define the architecture, including the number of layers, the number of nodes in each layer, and the activation functions to be used.

2

Specify the Loss Function

The loss function, which measures the difference between the model's predictions and the true labels, is a crucial component of the model compilation process.

3

Choose an Optimizer

The optimizer is responsible for updating the model's weights during the training process, with the goal of minimizing the loss function.

Preparing the Data

Data Cleaning

The first step in preparing the data for an MLP model is to clean and preprocess the input data. This may involve handling missing values, removing outliers, and encoding categorical variables into a format that the model can understand.

Feature Engineering

Feature engineering is the process of creating new, more informative features from the raw input data. This can help the model better capture the underlying patterns and relationships in the data, leading to improved performance.

Data Normalization

Normalizing the input data is an important step to ensure that all features are on a similar scale. This can help the model converge more quickly and improve its overall performance.

Preparing the Data

For this example we will be using the CIFAR-10 dataset, a collection of 60,000 32×32 -pixel color images that comes bundled with Keras out of the box. Each image is classified into exactly one of 10 classes, as shown in Figure 2-4.



Figure 2-4. Example images from the CIFAR-10 dataset (source: Krizhevsky, 2009)

By default, the image data consists of integers between 0 and 255 for each pixel channel. We first need to preprocess the images by scaling these values to lie between 0 and 1, as neural networks work best when the absolute value of each input is less than 1.

We also need to change the integer labeling of the images to one-hot encoded vectors, because the neural network output will be a probability that the image belongs to each class. If the class integer label of an image is i , then its one-hot encoding is a vector of length 10 (the number of classes) that has 0s in all but the i th element, which is 1.

Building the Model

1

Define the Architecture

The first step in building an MLP model is to define the architecture, which includes the number of layers, the number of nodes in each layer, and the activation functions to be used.

2

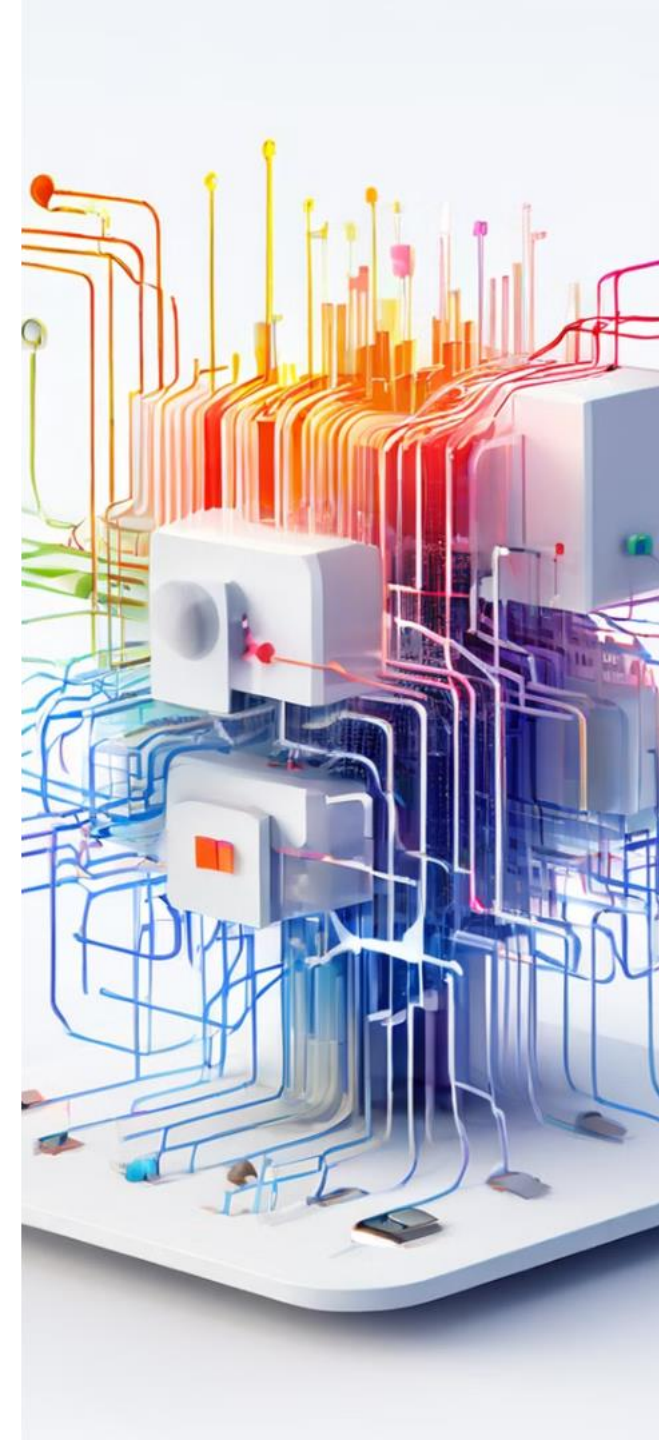
Initialize the Weights

The weights of the connections between the nodes in the MLP model are initialized to small, random values. This helps the model learn the underlying patterns in the data during the training process.

3

Train the Model

The MLP model is trained using a technique called backpropagation, which adjusts the weights of the connections based on the error between the predicted and actual outputs.



Training the Model

1

Data Preprocessing

Before training the model, the input data must be preprocessed, which may involve tasks such as normalization, feature scaling, and handling missing values.

2

Batch Training

The model is trained in batches, where the weights are updated after each batch of data is processed, rather than updating the weights after each individual sample.

3

Monitoring Training Progress

During the training process, it's important to monitor the model's performance on both the training and validation sets to ensure that the model is learning effectively and not overfitting.



Evaluating the Model

Accuracy Metrics

Common accuracy metrics used to evaluate the performance of an MLP model include accuracy, precision, recall, and F1-score.

Loss Metrics

The loss function used during training can also be used as a metric to evaluate the model's performance, with lower loss values indicating better performance.

Generalization

It's important to assess the model's ability to generalize to new, unseen data, which can be done by evaluating the model's performance on a separate test set.

Hyperparameter Tuning

Learning Rate

The learning rate determines the step size taken by the optimizer during the training process, and can have a significant impact on the model's performance.

Number of Layers

The number of layers in the MLP model can affect its ability to learn complex patterns in the data, with deeper models generally being more expressive.

Regularization

Regularization techniques, such as L1 or L2 regularization, can be used to prevent overfitting and improve the model's generalization performance.

Batch Size

The batch size determines the number of samples used in each update of the model's weights, and can affect the model's convergence and stability.

Applications of MLPs



Computer Vision

MLPs are widely used in computer vision tasks, such as image classification, object detection, and image segmentation.



Natural Language Processing

MLPs can be used for various natural language processing tasks, including text classification, sentiment analysis, and language modeling.



Predictive Analytics

MLPs are effective at modeling complex, non-linear relationships in data, making them useful for a variety of predictive analytics tasks, such as forecasting and regression.



Healthcare

MLPs have been applied in the healthcare domain for tasks like disease diagnosis, drug discovery, and patient outcome prediction.



Advantages of MLPs

1

Flexibility

MLPs are highly flexible and can be adapted to a wide range of problem domains and data types, making them a versatile choice for many machine learning tasks.

2

Universal Approximation

MLPs with a sufficient number of hidden layers and nodes can theoretically approximate any continuous function, giving them the ability to learn complex, non-linear relationships in data.

3

Parallel Processing

The structure of MLPs allows for efficient parallel processing, which can significantly speed up the training and inference process, especially on specialized hardware like GPUs.

Limitations of MLPs

Long-Range Dependencies

MLPs can struggle to capture long-range dependencies in data, which can be a limitation for certain types of sequential data, such as natural language or time series.

Hyperparameter Sensitivity

The performance of MLPs can be highly sensitive to the choice of hyperparameters, such as the learning rate, number of layers, and regularization, which can make the model

Overfitting

MLPs have the potential to overfit the training data, especially when the model is too complex or the training dataset is too small, which can lead to poor generalization to new, unseen data.



Limitations of MLP

Vanishing Gradients

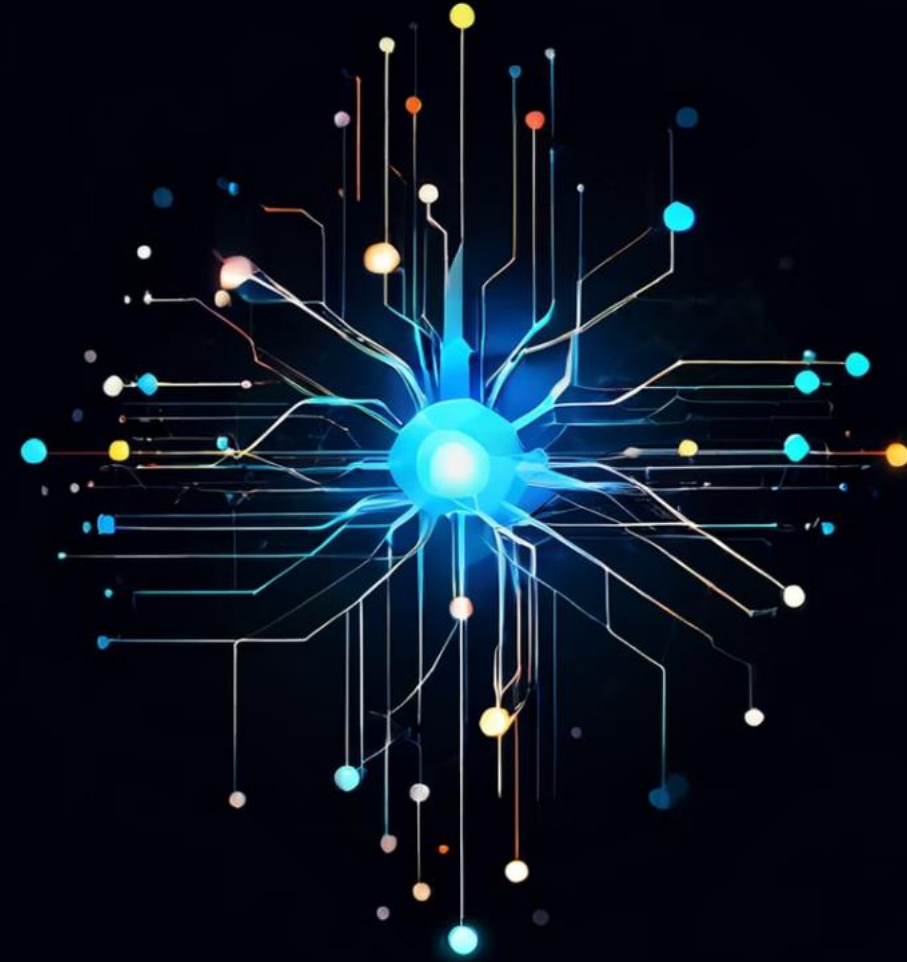
In deep MLPs with many hidden layers, the gradients used in the backpropagation algorithm can become very small, leading to slow convergence or even the model getting stuck in a suboptimal solution.

Interpretability

MLPs, like many other neural networks, can be difficult to interpret, as the internal representations and decision-making process are not easily explainable. This can be a challenge in applications where transparency is important.

Conclusion

The Multilayer Perceptron (MLP) is a powerful and versatile artificial neural network that has found widespread applications in various domains, from image recognition to predictive analytics. While MLPs offer several advantages, such as universal approximation, adaptability, and scalability, they also have some limitations, such as the risk of overfitting, vanishing gradients, and interpretability challenges. Understanding the strengths and weaknesses of MLPs is crucial for effectively leveraging this technology in real-world scenarios and achieving optimal performance.



⊗ Data for Deep Learning



⊗ Deep Neural Networks

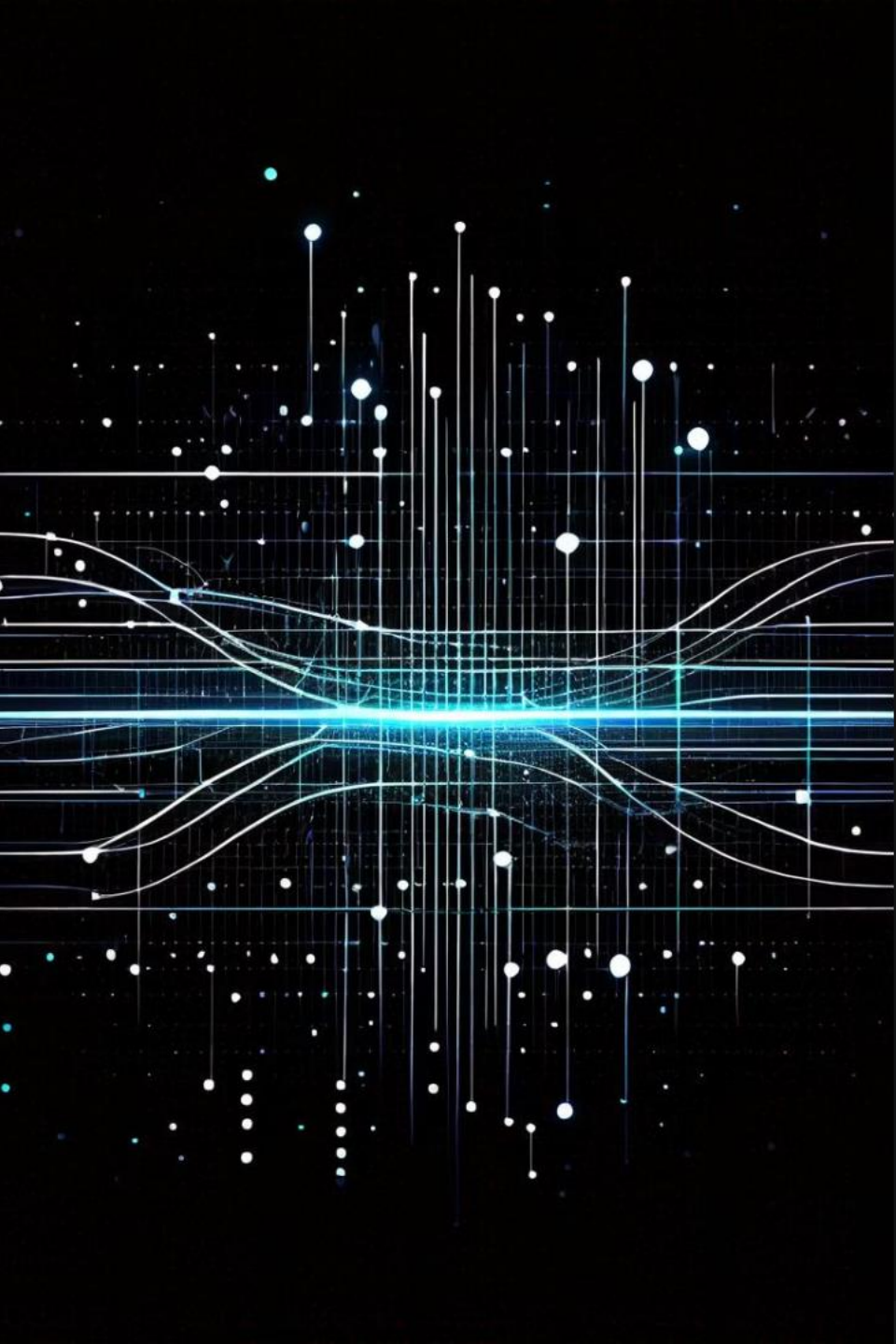


⊗ Multilayer Perceptron (MLP)



⊗ Convolutional Neural Network (CNN)



An abstract digital background on the left side of the slide. It features a dark blue/black field with numerous thin, glowing white and light blue lines that crisscross and curve across the space. Interspersed among these lines are many small, bright white and light blue dots of varying sizes, some appearing as if they are part of a larger network or data stream. The overall effect is one of high-tech connectivity and data flow.

Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a powerful type of deep learning algorithm that have revolutionized the field of computer vision and image recognition. These neural networks are inspired by the human visual cortex and are particularly well-suited for processing and analyzing visual data.

How CNNs Work

Convolutional Layers

The core of a CNN is its convolutional layers, which apply a set of learnable filters to the input image. These filters are designed to detect low-level features like edges and shapes, and as the network goes deeper, the filters become more complex and can detect higher-level features like objects and patterns.

Pooling Layers

Pooling layers are used to reduce the spatial size of the feature maps, which helps to make the network more robust to small translations and distortions in the input image. The most common pooling operation is max pooling, which takes the maximum value from a small region of the feature map.

Fully Connected Layers

At the end of the CNN, there are typically one or more fully connected layers that take the output of the convolutional and pooling layers and use it to make a final classification or prediction. These layers learn to combine the features detected by the earlier layers to make a high-level decision.

Applications of CNNs

1

Image Classification

CNNs excel at classifying images into different categories, such as recognizing objects, faces, or scenes. This has applications in areas like security, self-driving cars, and medical imaging.

2

Object Detection

CNNs can be used to detect and localize multiple objects within an image, which is useful for applications like surveillance, autonomous vehicles, and robotics.

3

Semantic Segmentation

CNNs can be used to assign a class label to every pixel in an image, allowing for detailed understanding of the contents of an image. This is useful for applications like medical image analysis and autonomous driving.

4

Image Generation

Generative Adversarial Networks (GANs), a type of deep learning model that includes CNNs, can be used to generate new images that are similar to a given set of training images. This has applications in art, design, and entertainment.

Advantages of CNNs



Spatial Awareness

CNNs are designed to take advantage of the spatial structure of images, allowing them to learn features that are invariant to translation, scaling, and other transformations.



Efficient Learning

The weight sharing and sparse connectivity of CNNs make them more efficient to train than fully connected neural networks, especially for large-scale image datasets.



End-to-End Learning

CNNs can be trained in an end-to-end fashion, directly mapping from raw image data to the desired output, without the need for manual feature engineering.



High Performance

CNNs have achieved state-of-the-art performance on a wide range of computer vision tasks, making them a powerful tool for real-world applications.



Training CNNs

1

Data Preparation

The first step in training a CNN is to prepare the dataset, which may involve tasks like image preprocessing, data augmentation, and splitting the data into training, validation, and test sets.

2

Model Architecture

The next step is to design the CNN architecture, which involves choosing the number and types of layers, the hyperparameters, and the loss function to be optimized.

3

Training Process

The CNN is then trained on the prepared dataset using an optimization algorithm like stochastic gradient descent, with the goal of minimizing the loss function and learning the optimal weights and biases.

Challenges and Limitations

Interpretability

CNNs are often criticized as being "black boxes" due to the difficulty in interpreting the internal representations and decision-making process of the network. This can make it challenging to understand and explain the model's behavior.

Overfitting

CNNs can be prone to overfitting, especially when the training dataset is small. This can lead to poor generalization performance on new, unseen data. Techniques like regularization and data augmentation are used to mitigate this issue.

Computational Complexity

Training and deploying large-scale CNNs can be computationally intensive, requiring significant hardware resources and energy consumption. This can be a challenge for real-time applications or deployment on resource-constrained devices.

Robustness

CNNs can be vulnerable to adversarial attacks, where small, imperceptible perturbations to the input image can cause the model to make incorrect predictions. Developing robust CNNs is an active area of research.

Future Developments

Efficient Architectures

Developing more efficient CNN architectures that can achieve high performance while using fewer parameters and less computational resources.

Interpretable CNNs

Advancing techniques to make CNNs more interpretable and explainable, allowing for better understanding and trust in the model's decision-making process.

Adversarial Robustness

Improving the robustness of CNNs to adversarial attacks, making them more reliable and secure for real-world

Multimodal Learning

Exploring the integration of CNNs with other deep learning models, such as recurrent neural networks, to enable multimodal learning and understanding.

Conclusion

Powerful Tool

Convolutional Neural Networks have emerged as a powerful tool for a wide range of computer vision tasks, revolutionizing fields like image classification, object detection, and semantic segmentation.

Bright Future

As the field of deep learning continues to evolve, we can expect to see further advancements in CNN architectures, training techniques, and applications, leading to even more powerful and versatile tools for understanding and interacting with the visual world.

1

2

3

Ongoing Advancements

While CNNs have achieved remarkable success, there are still ongoing challenges and limitations that researchers and engineers are working to address, such as interpretability, robustness, and computational efficiency.

Thank You