Comprehensive Report on Deep Learning

Presented By:
Vaibhav Sakharwade
UGMR ID: 20230012

JULY
2024

Table of Contents

Deep Learning is subset of machine learning based on which artificial neural networks with the representation of learning. It involves all training model to recognize patterns and which make decisions with less human intervention.

Introduction

Deep Learning is a subset of machine learning based on artificial neural networks with representation learning. It involves training models to recognize patterns and make decisions with minimal human intervention. Deep learning has shown remarkable performance in tasks such as image recognition, speech processing, and natural language understanding due to its ability to learn from vast amounts of data.

Basics of Deep Learning

1. Neural Networks

Neural Networks are the foundation of deep learning. They consist of layers of interconnected nodes (neurons) that simulate the workings of the human brain. Each neuron receives input, processes it using an activation function, and passes the output to the next layer.

a. Layers

Input Layer: The layer where data enters the network. It does not perform any computations.

Hidden Layers: Layers between the input and output layers where the network performs most of its computations. These layers learn to extract features and patterns from the data.

Output Layer: The final layer that produces the network's predictions. The number of neurons in the output layer depends on the specific task, such as classification or regression.

b. Neurons

Each neuron in a layer is connected to every neuron in the previous and next layers. These connections have associated weights that are adjusted during training. The output of each neuron is computed as the weighted sum of its inputs, passed through an activation function.

Basics of Deep Learning

2. Activation Functions

Activation functions introduce non-linearity into the network, enabling it to learn complex patterns. Common activation functions include:

Sigmoid: $\sigma(x) = \frac{1}{1 + e^{-x}}$. It outputs values between 0 and 1 and is often used in the output layer of binary classification problems.

ReLU (Rectified Linear Unit): $f(x) = \max(0, x)$. It is widely used in hidden layers due to its simplicity and effectiveness in mitigating the vanishing gradient problem.

Tanh: $\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$. It outputs values between -1 and 1 and is used in hidden layers when the data is centered around zero.

3. Training Neural Networks

Training involves adjusting the weights of the connections between neurons using a process called backpropagation.

a. Forward Pass

In the forward pass, the input data is passed through the network, and the output is computed. This output is then compared to the true labels using a loss function to measure the error

b. Backward Pass (Backpropagation)

In the backward pass, the error is propagated back through the network, and the weights are updated using optimization algorithms such as gradient descent.

Basics of Deep Learning

3. Training Neural Networks

C. Optimization Algorithms

Gradient Descent: Iteratively adjusts the weights to minimize the loss function variants include.

Stochastic Gradient Descent (SGD): Updates weights using one training example at a time.

Mini-batch Gradient Descent: Updates weights using a small batch of training examples.

Adam (Adaptive Moment Estimation): Combines the advantages of both AdaGrad and RMSProp, maintaining an adaptive learning rate for each parameter.

Convolutional Neural Networks {CNNs}

CNNs are specialized neural networks designed to process and analyze visual data, such as images and videos. They leverage spatial hierarchies in data through convolutional layers, pooling layers, and fully connected layers.

1. Convolutional Layers

Convolutional layers apply filters (kernels) to the input image, creating feature maps that highlight various features like edges, textures, and patterns.

a. Filters and Feature Maps

Filters: Small matrices that slide over the input image to detect specific features. Each filter produces a feature map, which represents the presence of the filter's feature at different locations in the input.

Stride and Padding: Stride determines how much the filter moves at each step. Padding involves adding borders to the input image to control the spatial dimensions of the output feature maps.

b. Activation Functions in CNNs

Convolutional layers are typically followed by activation functions (e.g., ReLU) to introduce non-linearity.

Convolutional Neural Networks {CNNs}

2. Pooling Layers

Pooling layers reduce the spatial dimensions of the feature maps, making the network more computationally efficient and reducing overfitting.

a. Types of Pooling

Max Pooling: Takes the maximum value from each patch of the feature map.

Average Pooling: Takes the average value from each patch.

Global Pooling: Reduces each feature map to a single value.

3. Fully Connected Layers

After several convolutional and pooling layers, fully connected layers (dense layers) are used to combine features and make the final prediction. These layers are similar to those in traditional neural networks.

4. Dropout

Dropout is a regularization technique used in CNNs to prevent overfitting. During training, randomly selected neurons are ignored (dropped out), forcing the network to learn redundant representations

Recurrent Neural Networks {RNNs}

RNNs are designed to handle sequential data by maintaining a memory of previous inputs, making them suitable for tasks such as time series prediction and natural language processing.

1. Basic RNN

a. Structure

Hidden State: At each time step, the hidden state is updated on the basis of current input and the previous hidden state.

Output: The output at each time step is on the basis of current hidden state.

b. Limitations

Basic RNNs suffer from the vanishing gradient problem, where gradients become very small, making it difficult to learn long-term dependencies.

Recurrent Neural Networks {RNNs}

2. Long Short-Term Memory (LSTM)

LSTMs are a type of RNN designed to overcome the vanishing gradient problem, capable of learning long-term dependencies.

a. LSTM Cell

Cell State: Carries information across long sequences. It is modified by the input, forget, and output gates.
Gates:
Forget Gate: Responsible deciding what information to discard from the cell state.

Input Gate: Responsible deciding what new information to add to the cell state.

Output Gate: Responsible deciding what information to output based on the cell state.

b. Advantages

LSTMs can capture long-term dependencies in data, making them effective for tasks such as language modeling and machine translation.

Recurrent Neural Networks {RNNs}

3. Gated Recurrent Unit (GRU)

GRUs are a simpler variant of LSTMs that combine the forget and input gates into a single update gate, making them computationally efficient.

a. GRU Cell

Update Gate: Controls the flow of information to the hidden state.

Reset Gate: Determines how much past information to forget.

b. Comparison with LSTM

GRUs are computationally simpler and often perform similarly to LSTMs, but they may not capture dependencies as effectively in some cases.

Transformers

GRUs are computationally simpler and often perform similarly to LSTMs, but they may not capture dependencies as effectively in some cases.

1. Attention Mechanism

The attention mechanism allows the model to focus on relevant parts of the input sequence when making predictions.

a. Self-Attention

Scaled Dot-Product Attention: Computes attention scores using the dot product of query and key vectors, scaled by the square root of the dimension of the key vectors.

Multi-Head Attention: Uses multiple attention heads to capture different aspects of the input sequence.

2. Encoder-Decoder Architecture

Transformers consist of an encoder that processes the input sequence and a decoder that generates the output sequence. Each layer in the encoder and decoder has an attention mechanism and feed-forward neural networks.

Transformers

a. Encoder

Self-Attention Layers: Capture dependencies within the input sequence.

Feed-Forward Layers: Apply non-linear transformations to the output of the self-attention layers.

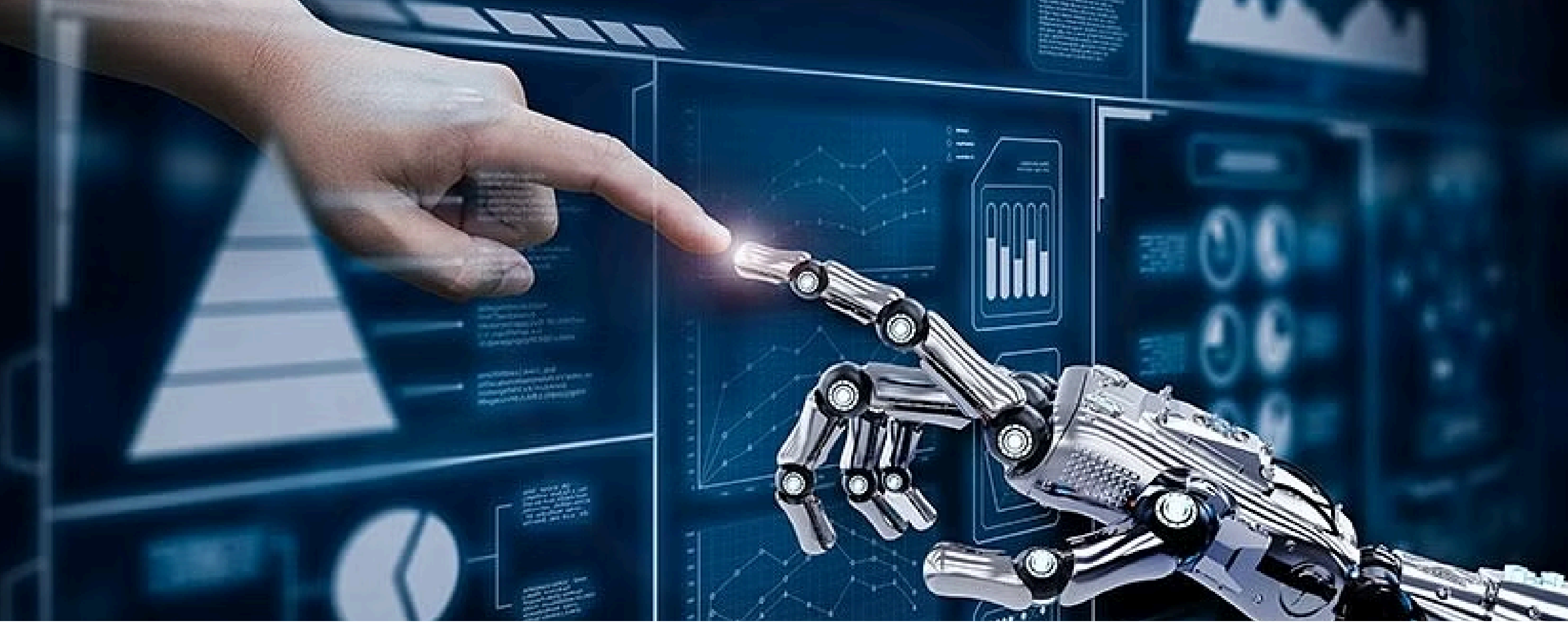
b. Decoder

Masked Self-Attention: Prevents the model from attending to future positions in the output sequence.

Encoder-Decoder Attention: Attends to the encoder’s output to incorporate information from the input sequence.

3. Applications of Transformers

Transformers are widely used in natural language processing tasks such as machine translation, text summarization, and language modeling. They have also been adapted for tasks like image classification and generation.



Conclusion

Deep Learning has revolutionized various fields by enabling the development of models that can learn complex patterns and make accurate predictions. From CNNs for image recognition to RNNs for sequential data and Transformers for language understanding, deep learning continues to advance, pushing the boundaries of what machines can achieve. As research progresses, we can expect even more sophisticated models and applications, further enhancing the capabilities of artificial intelligence.

Link To The Repositories Codebase

Summary

The provided codebase demonstrates how to:

- Load and preprocess the CIFAR-10 dataset.
- Define CNN, RNN, and Transformer models.
- Train each model using a specified training function.
- Evaluate the accuracy of each model on the test dataset.

https://github.com/Vaibhav-S75/IIIT-_Projects_Lab_01.git

GitHub