**Name:-tejaswini badpaiya**

**Rollno:-** **ugmr20230024**

**Table of Contents**

1. **Introduction**
2. **Historical Background**
3. **Key Concepts in Deep Learning**
4. **Convolutional Neural Networks (CNNs)**
5. **Recurrent Neural Networks (RNNs)**
6. **Transformers**
7. **PyTorch**
8. **Conclusion**

**Comprehensive Report on Deep Learning**

**Introduction to Deep Learning**

**Deep learning, a subset of machine learning, involves using neural networks with multiple layers to understand complex data patterns. Inspired by the human brain's neural networks, it has revolutionized artificial intelligence (AI).**

**Types of Neural Networks**

**Convolutional Neural Networks (CNNs): Ideal for image processing.**

**Recurrent Neural Networks (RNNs): Designed for sequential data, like text.**

**Generative Adversarial Networks (GANs): Generate realistic data by training two networks adversarially.**

**Applications**

**Computer Vision: Image classification, object detection.**

**Natural Language Processing (NLP): Language translation, chatbots.**

**Speech Recognition: Converting speech to text.**

**Autonomous Vehicles: Self-driving car technologies.**

**Healthcare: Disease prediction, medical imaging.**

**2. Key Concepts in Deep Learning**

**Artificial Neural Networks**

* **Artificial Neurons:** The basic units of neural networks that mimic the behavior of biological neurons. Each neuron receives input, processes it, and passes the output to the next layer.
* **Layers:** Neural networks consist of an input layer, one or more hidden layers, and an output layer. The depth (number of layers) of the network is what makes it "deep."
* **Weights and Biases:** Parameters that the network adjusts during training to minimize error and improve performance.

**Activation Functions**

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

* **ReLU (Rectified Linear Unit):** Outputs zero for negative inputs and the input itself for positive inputs.
* **Sigmoid:** Squashes input values to a range between 0 and 1.
* **Tanh:** Squashes input values to a range between -1 and 1.

**Training Process**

Training a neural network involves:

* **Forward Propagation:** Passing inputs through the network to generate outputs.
* **Loss Function:** Calculating the error between the network's output and the actual target values.
* **Backpropagation:** Adjusting the network's weights and biases based on the error gradient to minimize the loss.

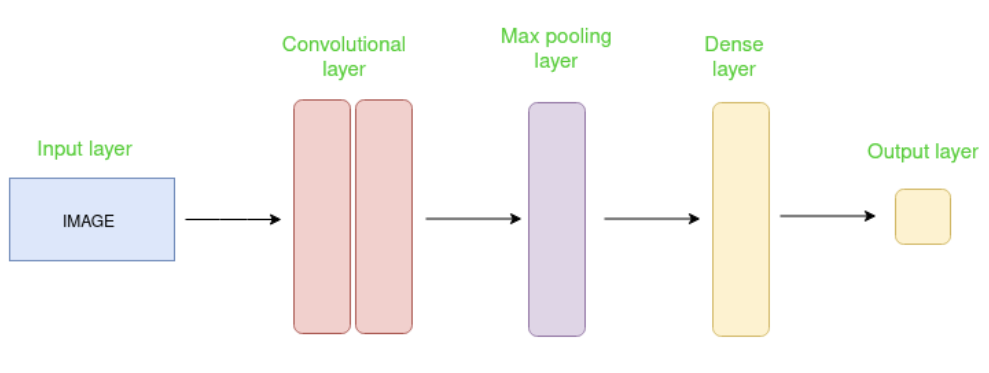
**Optimization Techniques in Deep Learning**

1. **Gradient Descent Variants**
   * **Stochastic Gradient Descent (SGD): Updates parameters using one training example at a time.**
   * **Batch Gradient Descent: Uses the entire dataset for each update.**
2. **Advanced Gradient Descent Algorithms**
   * **Momentum: Accelerates gradients vectors for faster convergence.**
   * **Adagrad: Adapts learning rates for each parameter.**
   * **RMSprop: adjust learning rates.**
   * **Adam: Combines Adagrad and RMSprop benefits.**
3. **Learning Rate Schedulers**
   * **Step Decay: Reduces learning rate periodically.**
   * **Exponential Decay: Reduces learning rate exponentially.**
   * **Cyclic Learning Rates: Oscillates learning rate between bounds.**
   * **Warm Restarts: Periodically resets the learning rate.**
4. **Regularization Techniques**
   * **L2 Regularization: Adds penalty proportional to the square of parameter**
5. **Gradient Clipping**
   * **Limits gradients to prevent exploding gradients.**
6. **Early Stopping**
   * **Stops training when validation performance degrades to prevent overfitting.**

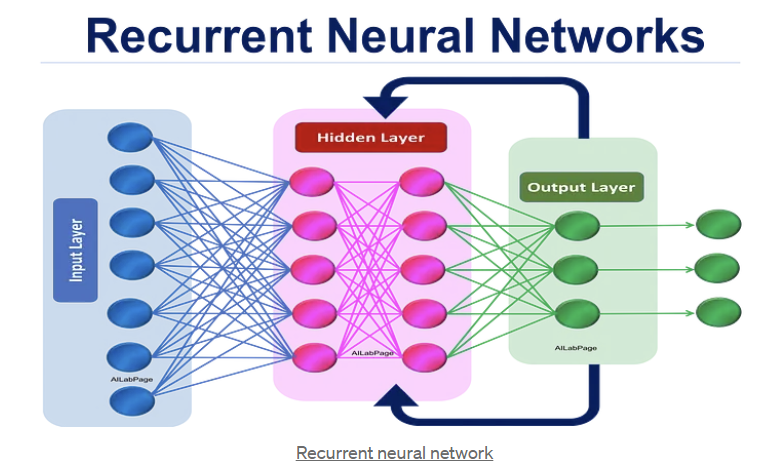
**Convolutional Neural Networks (CNNs):**

Convolutional Neural Networks (CNNs) are a class of deep neural networks specifically designed to process data with a grid-like topology, such as images. They have proven to be highly effective in various computer vision tasks due to their ability to automatically and adaptively learn spatial hierarchies of features from input images.

**Key Components of CNNs**

1. **Convolutional Layers:**
2. 
   * **Convolution Operation:** **Convolution Operation in CNNs**
   * **Purpose:** Extracts features from input images by preserving spatial relationships.
   * **How It Works:**
   * **Filter (Kernel):** A small matrix slides over the input image.
   * **Element-wise Multiplication:** Each element of the filter multiplies corresponding elements in the input.
   * **Sum:** The results are summed to get a single value.
   * **Stride:** Controls the step size of the filter movement.
   * **Padding:** Adds borders to the input for output size control.
   * **Output (Feature Map):** A new image representing the presence of features.
   * **Example:**
   * **Input Image:** 5x5 matrix.
   * **Filter:** 3x3 matrix.
   * **Stride:** 1.
   * **Padding:** 0 (valid padding).
   * **Operation:**
   * Filter slides over the input image.
   * Multiplies element-wise and sums to produce a single value at each position.
   * **Benefits:**
   * **Reduces Dimensionality:** Without losing important information.
   * **Detects Patterns:** Like edges, textures, and objects.
3. **Activation Functions:**
   * **ReLU (Rectified Linear Unit):** The most commonly used activation function in CNNs, which introduces non-linearity by setting all negative values to zero and leaving positive values unchanged.
4. **Pooling Layers:**
   * **Purpose:** To reduce the spatial dimensions (width and height) of the feature maps, thereby decreasing the computational load and memory usage.
   * **Max Pooling:** The most common pooling operation, which takes the maximum value from each patch of the feature map.
   * **Average Pooling:** Computes the average value of each patch in the feature map.
5. **Fully Connected (Dense) Layers:**
   * **Purpose:** To perform high-level reasoning and classification based on the features extracted by the convolutional and pooling layers.
   * **Structure:** Each neuron in a fully connected layer is connected to every neuron in the previous layer.
6. **Output Layer:**
   * **Purpose:** To produce the final classification or prediction.
   * **Activation Function:** Typically uses the Softmax activation function for multi-class classification tasks.

**Recurrent Neural Network (RNN):**

****

**Recurrent Neural Networks (RNNs) are a type of neural network designed to handle sequential data, such as time series or language. Unlike traditional neural networks, RNNs have loops in their architecture, allowing information to be passed from one step to the next. This "memory" helps RNNs understand context and sequence in data, making them effective for tasks like speech recognition, language translation, and predicting stock prices. They process data one item at a time and maintain a hidden state to capture past information, making them powerful for tasks where order and context are crucial.**

**Key Components of RNNs**

there are the key points about Recurrent Neural Networks (RNNs):

1. **Sequential Data:** Designed for tasks involving sequences like time series or text.
2. **Recurrent Connections:** Have loops to pass information from previous steps.
3. **Hidden State:** Maintains a memory of past inputs to capture context.
4. **Backpropagation Through Time (BPTT):** Extends backpropagation for training on sequences.
5. **Vanishing/Exploding Gradients:** Common challenges in training RNNs.
6. **Variants:** Includes LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) for improved performance.
7. **Applications:** Used in language modeling, speech recognition, and time series forecasting.

4o mini

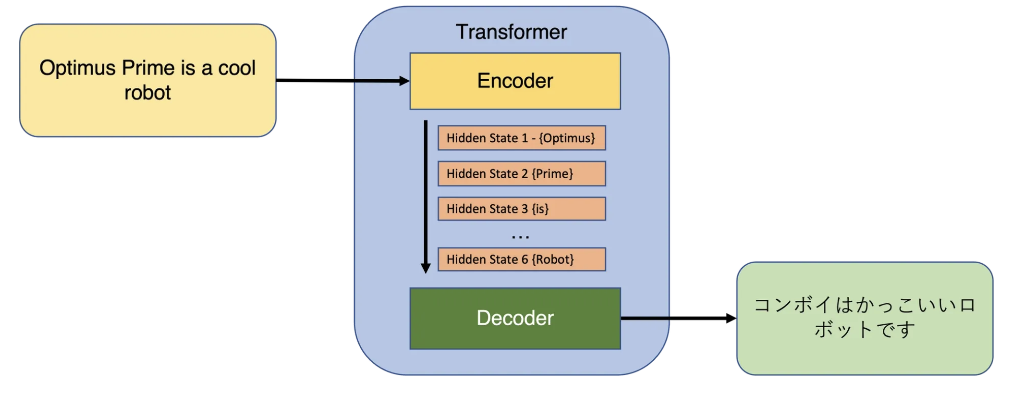
1. **Bidirectional RNNs**:

 **Idea**: Enhances traditional RNNs by processing sequences in both forward and backward directions.

 **Application**: Useful for tasks where context from both past and future inputs is beneficial, such as speech recognition and named entity recognition.

**Transformers:**

Transformers are a type of deep learning model that has revolutionized natural language processing (NLP) tasks by leveraging self-attention mechanisms. They were introduced in the paper "Attention is All You Need" by Vaswani et al. (2017).



**Key Components:**

**Key Features:**

**Here are the key points of Transformers:**

1. **Self-Attention: Captures relationships between words.**
2. **Parallel Processing: Processes all tokens at once.**
3. **Encoder-Decoder: For transforming input to output.**
4. **Positional Encoding: Maintains token order.**
5. **Multi-Head Attention: Captures diverse relationships.**
6. **Feedforward Layers: Additional processing per token.**
7. **Residual Connections: Helps in training deep networks.**
8. **Transfer Learning: Pre-trained models can be fine-tuned.**

**Transformer architecture:**

** Encoder:**

* **Self-Attention Mechanism: Weighs the importance of each word in relation to others.**
* **Feedforward Network: Processes each position separately.**
* **Positional Encoding: Adds positional information to tokens.**
* **Residual Connections: Helps in training deep networks.**
* **Layer Normalization: Stabilizes and accelerates training.**

** Decoder:**

* **Masked Self-Attention: Prevents attending to future tokens in the sequence.**
* **Encoder-Decoder Attention: Attends to the encoder's output.**
* **Feedforward Network: Similar to the encoder's feedforward network.**
* **Residual Connections and Layer Normalization: Similar to the encoder.**

** Key Components:**

* **Multi-Head Attention: Uses multiple attention heads to capture different relationships.**
* **Position-wise Feedforward Networks: Applies independently to each token.**

1. **Tensor Computation**:
   * At its core, PyTorch provides powerful multi-dimensional array operations, similar to NumPy arrays but optimized for GPU acceleration.
   * Tensors in PyTorch can be used to represent and manipulate data at various stages of a machine learning pipeline, from input data to model predictions.
2. **Automatic Differentiation**:
   * One of PyTorch's standout features is its automatic differentiation capability through the autograd package.
   * This feature allows gradients to be computed automatically for tensors, facilitating efficient implementation of gradient-based optimization algorithms like backpropagation.
3. **Dynamic Computation Graphs**:
   * PyTorch uses a dynamic computational graph approach, where the graph is built on-the-fly during runtime.
   * This dynamic nature enables more flexible and intuitive model construction compared to static graph frameworks.
4. **Modular and Extensible**:
   * PyTorch offers a modular and extensible architecture, allowing developers to build complex neural network architectures with ease.
   * It provides a rich set of built-in modules and utilities for defining layers, activation functions, loss functions, and more.
5. **Support for GPU Acceleration**:
   * PyTorch seamlessly integrates with CUDA-capable GPUs to leverage their parallel processing capabilities.
   * This GPU acceleration significantly speeds up computations, making it ideal for training deep neural networks on large datasets.

**Conclusion:**

Deep learning represents a pivotal advancement in machine learning, particularly through the evolution of artificial neural networks and specialized architectures like CNNs, RNNs, and Transformers. PyTorch, with its powerful tensor computation, automatic differentiation capabilities, and GPU acceleration support, has emerged as a leading framework for developing and deploying deep learning models efficiently.

**GitHub path of Implementations of CNN, RNN and Transformers:**

**Repository:-https://github.com/tejaswinibadpaiya/FMML\_Projects\_and-Labs/commit/9d46daf0e34c7aa44ec79cb02182bb2d9e599774**

* 1. [**https://github.com/tejaswinibadpaiya/FMML\_Projects\_and-Labs/blob/f741cdb19c2b9a5fa3397cdbc1af33be77637dbe/cnn.ipynb**](https://github.com/tejaswinibadpaiya/FMML_Projects_and-Labs/blob/f741cdb19c2b9a5fa3397cdbc1af33be77637dbe/cnn.ipynb)
  2. [**https://github.com/tejaswinibadpaiya/FMML\_Projects\_and-Labs/blob/f741cdb19c2b9a5fa3397cdbc1af33be77637dbe/cnn.ipynb**](https://github.com/tejaswinibadpaiya/FMML_Projects_and-Labs/blob/f741cdb19c2b9a5fa3397cdbc1af33be77637dbe/cnn.ipynb)
  3. **https://github.com/tejaswinibadpaiya/FMML\_Projects\_and-Labs/blob/f741cdb19c2b9a5fa3397cdbc1af33be77637dbe/transformer.ipynb**