****

**International Institute of Information Technology Hyderabad**

**Report on Deep Learning**

**By: Ekta Kature [ugmr20230028]**

**(Tech Titans)**

**Deep learning:-**

Deep learning is a subset of machine learning, which is itself a branch of artificial intelligence (AI). It involves training artificial neural networks on large amounts of data to model complex patterns and make predictions.

Let's look at some types of deep learning models and neural networks:

1. Convolutional Neural Networks (CNN)
2. Recurrent Neural Networks (RNN)
3. Transformers

**Convolutional Neural Networks (CNN):-**

CNNs are a family of deep neural networks specifically designed to process network-like information such as images. They automatically and adaptively learn spatial hierarchies of features through convolutional layers.  
**Formula:**

*(I\*K)(i,j)=∑ ∑ I(i+m,j+n).K(m,n)*

where *I* is the input matrix, *K* is the kernel matrix, and (i,j)ndenotes the position in the output matrix.

**Activation function:** To incorporate nonlinearity, an activation function such as Rectified Linear Unit (ReLU) is applied after transformation.  
**Formula (ReLU):**

ReLU(x)=max(0,x)

**Pooling Layer:** This layer uses average or max pooling to achieve downsampling to reduce the spatial scale of the input.  
**Formula (Max Pooling):**

Output(i,j)=max(input values in the pooling window)

### Recurrent Neural Networks (RNNs) RNNs are designed for sequential data where the output depends on previous computations. They have loops that allow information to be carried across timestep.

**Basic RNN:**The core of an RNN is the recurrence relation, which updates the hidden state at each timestep. **Formula:**

*ht=tanh(Whht-1+Wxxt+bh)*

where *ht* ​ is the hidden state at time *t* , ​ *Wh* and *WX* are weight matrices, ​*xt* is the input at time *t* and *bh* is the bias.

**Transformers**:

Transformers are a type of neural network architecture primarily used for natural language processing tasks. They rely on self-attention mechanisms to process input data in parallel, allowing them to handle long-range dependencies more efficiently than RNNs. Transformers have led to significant advancements in language understanding and generation.

**Attention Mechanism**:The attention mechanism enables the model to assess the relative importance of various segments of the input sequence, concentrating on the segments that are more pertinent. It facilitates the capture of dependencies and linkages in data.

**Encoder-Decoder Structure:** In language translation tasks, the input sequence is processed by the encoder, and the output sequence is generated by the decoder.

**Explaination for the code:(using cnn model)**

**Dataset Used:-**The dataset used in this code is the MNIST dataset. The MNIST dataset is a large database of handwritten digits that is commonly used for training various image processing systems. It includes images of digits ranging from 0 to 9.

**Size of Dataset**Training Set: 60,000 imagesTesting Set: 10,000 imagesValidation Set: Not explicitly used in this code, but it can be created by splitting the training set if necessary.

**Training, Testing, Validation Splits**

The dataset is split into two parts:  
**Training Set:** 60,000 images  
**Testing Set:** 10,000 images  
A validation set is not explicitly created in the provided code. However, typically a portion of the training set (e.g., 10-20%) can be reserved for validation.

**Preprocessing Steps for Dataset  
Transformations:** The images are transformed using transforms.Compose which includes: **transforms.ToTensor()**: Converts the images to PyTorch tensors. **transforms.Normalize((0.5,), (0.5,)):** Normalizes the images with a mean of 0.5 and a standard deviation of 0.5. This scales the pixel values to the range [-1, 1].

**Model Type**The model used is a Convolutional Neural Network (CNN), which is suitable for image classification tasks.

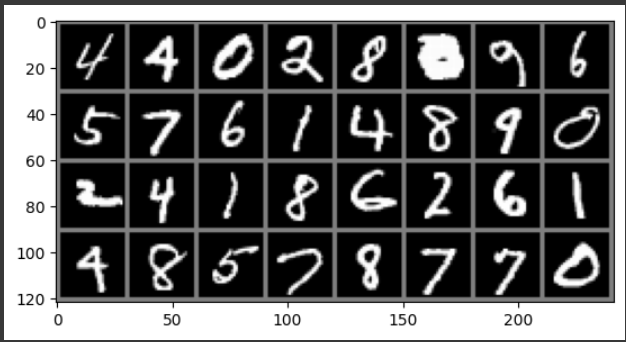
**Model Architecture**The CNN model consists of the following layers:

**Convolutional Layer 1:** Takes an input image with 1 channel (grayscale) and applies 32 filters of size 3x3 with padding of 1.  
**Max-Pooling Layer 1:** Applies a 2x2 max-pooling operation with a stride of 2.  
**Convolutional Layer 2:** Takes the output from the first pooling layer and applies 64 filters of size 3x3 with padding of 1.  
**Max-Pooling Layer 2:** Applies a 2x2 max-pooling operation with a stride of 2.  
**Fully Connected Layer 1**: Takes the flattened output from the second pooling layer (64 \* 7 \* 7) and has 128 neurons.  
**Fully Connected Layer 2:** The final layer with 10 neurons, corresponding to the 10 digit classes (0-9).

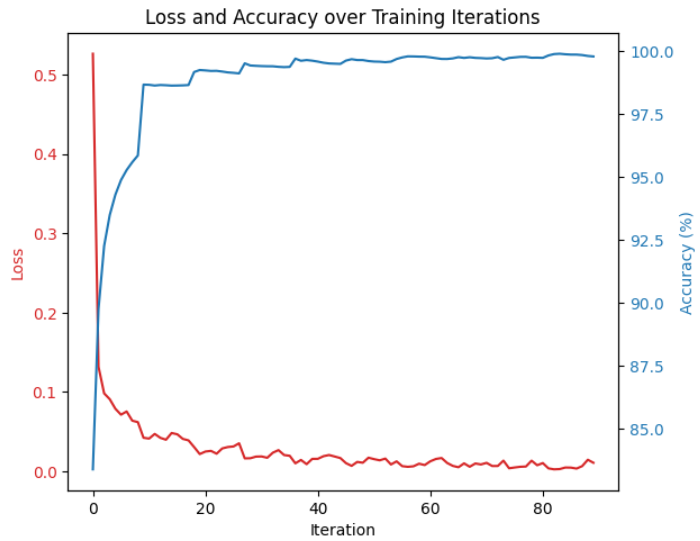
**Optimizer**The optimizer used is Adam (Adaptive Moment Estimation) with a learning rate of 0.001. Adam combines the advantages of two other extensions of stochastic gradient descent: adaptive gradient algorithm (AdaGrad) and root mean square propagation (RMSProp). **Loss Function**The loss function used is Cross-Entropy Loss, which is suitable for multi-class classification problems. It measures the performance of a classification model whose output is a probability value between 0 and 1. **Epochs Used**The model is trained for 10 epochs. An epoch is a full pass through the entire training dataset. **Train Accuracy**The code does not explicitly calculate and print the training accuracy, but it does print the running loss every 200 mini-batches during training. The running loss helps in monitoring the training process. **Test Accuracy**The test accuracy is calculated after the training process. It involves passing the test images through the trained model, comparing the predicted labels with the true labels, and computing the overall accuracy. The code prints the accuracy of the network on the 10,000 test images. **Visualization**

**Sample Images:** The code includes a function to display a batch of training images using matplotlib. It shows a grid of images along with their labels.

**Model Architecture:** The model architecture is visualized using torchviz. A random sample input is passed through the model, and the resulting computation graph is rendered and saved as a PNG file. This helps in understanding the structure and flow of data through the model.  
This explanation covers the main aspects of the code, providing a theoretical overview of the dataset, preprocessing, model, training, and evaluation processes.



**Loss vs accuracy plot:-**

****

**Conclusion:-**

By studying on the models of deep learning and doing its implementation I come across on the conclusion that the CNN model trained on the MNIST dataset demonstrates effective image classification, with test accuracy reflecting its ability to generalize to new data. The training process, visualized through loss and accuracy plots, shows a typical trend of decreasing loss and increasing accuracy. Overall, the model performs well for the task, though incorporating a validation set and fine-tuning hyperparameters could further enhance its performance.

**Github link:-** https://github.com/Ekta-kature/Capstone/blob/main/Basics\_of\_deep\_learning.ipynb