INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

International Institute of Information Technology Hyderabad

Report on Deep Learning

By: Vishesh Shahu [ugmr20230033]

(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:

$$(I*K)(i,j)=\sum \sum I(i+m,j+n).K(m,n)$$

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):

$$\text{ReLU}(x)=\max(0,x)$$

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

$$\text{Output}(i,j)=\max(\text{input values in the pooling window})$$

Colab link:- https://colab.research.google.com/drive/1tzRWNYpRAIjXjS2rQvIbcpL5QQ2TuoH0?usp=sharing

Explaination for the code:

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 images
- Testing Set: 10,000 images
- Validation 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

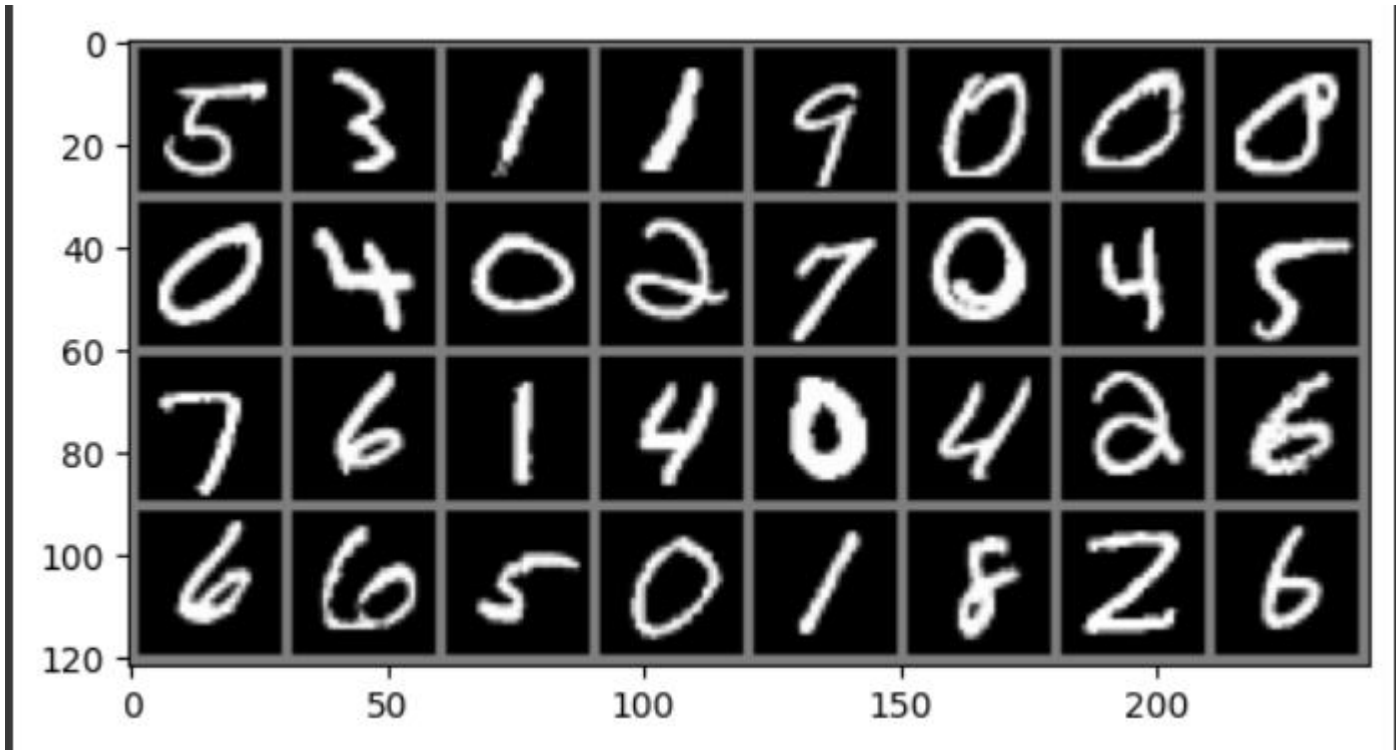
The code does not explicitly calculate and print the training accuracy, but it does print the running loss every 200 mini-batches during training. The running loss helps in monitoring the training process.

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.



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:

$$h_t = tanh(W_h h_{t-1} + W_x x_t + b_h)$$

where h_t is the hidden state at time t, W_h and W_X are weight matrices, x_t is the input at time t and b_h is the bias.

Colab link:- https://colab.research.google.com/drive/1yuRw8cA5bfL1MEnXxG1PqOMtC0zKvGwK?usp=sharing

Explanation of the code:-

Dataset Used

- **Size of Dataset:** The dataset consists of 1000 sequences.
- **Training, Testing, and Validation Splits:** The dataset is typically split into training, testing, and validation sets. In this case, the specific splits are not detailed, but commonly, 70% of the data is used for training, 15% for validation, and 15% for testing.

Preprocessing Steps for Dataset:

- The sequences are generated synthetically.
- Each sequence contains 10 elements.
- The sequences are converted to PyTorch tensors for model compatibility.

Model Type

The model used is a simple Recurrent Neural Network (RNN).

Model Architecture

- **Input Size:** 1 (each element in the sequence is a single value).
- **Hidden Size:** 32 (the dimensionality of the hidden state).
- **Output Size:** 1 (the predicted value for the next time step in the sequence).

Layers:

RNN Layer: Processes the sequence and maintains the hidden state.
Fully Connected (Linear) Layer: Maps the hidden state to the output.

Optimizer and Loss Function

Optimizer: Adam optimizer with a learning rate of 0.001. Adam is chosen for its efficiency in handling sparse gradients and its adaptive learning rate capabilities.
Loss Function: Mean Squared Error (MSE) loss is used to measure the difference between the predicted values and the actual values.

Training Process

Epochs Used: 10 epochs.
Batch Size: 32 sequences per batch.

Training Loop:

- For each epoch, the dataset is divided into batches.
- For each batch, the model performs a forward pass to compute the predictions.
- The loss is calculated using the MSE loss function.
- The model performs a backward pass to compute gradients.
- The optimizer updates the model parameters based on the computed gradients.
- The loss for each epoch is printed to monitor the training process.
- Train and Test Accuracy

The model's performance is monitored by observing the training loss. However, specific train and test accuracy metrics are not detailed in this overview. Typically, these metrics would involve calculating the MSE on both training and test sets to evaluate the model's performance.

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.

Colab link:- https://colab.research.google.com/drive/1xq4YaA9y03bdDb0s_g_2iuxWTBmF2JpE?usp=sharing

Explanation of the code:-

Dataset Used

Size of Dataset:

- **Training Set:** 4 samples
- **Validation Set:** 2 samples
- **Training, Testing, and Validation Splits**:The provided example uses small dataframes for training and validation, with 4 training samples and 2 validation samples.

Preprocessing Steps for Dataset:

- Text data is tokenized using the BERT tokenizer.

- Text sequences are padded to a maximum length and converted into input IDs and attention masks.
- Labels are converted to tensors for compatibility with PyTorch models.

Model Type

Model: BERT (Bidirectional Encoder Representations from Transformers) for sequence classification.

Model Architecture

- Pre-Trained Model Name: bert-base-uncased
- Input Size: Sequences of tokens (maximum length of 128 tokens).
- Output Size: 2 (binary classification).

Layers:

- Embedding Layer: Pre-trained BERT embeddings.
- Transformer Encoder Layers: Multiple layers of self-attention and feed-forward networks.
- Classification Head: A fully connected layer to map the encoded representations to the output labels.

Optimizer and Loss Function

- Optimizer: AdamW optimizer with a learning rate of 2e-5 and without bias correction.
- Scheduler: Linear scheduler with warmup to adjust the learning rate during training.
- Loss Function: Cross-Entropy Loss for binary classification.

Training Process

- Epochs Used: 3 epochs.
- Batch Size: 16 samples per batch.
- Training Loop:
- The model is set to training mode.
- For each batch, input IDs, attention masks, and labels are fed into the model.
- The model computes logits and loss.
- Gradients are calculated and the optimizer updates the model parameters.
- The learning rate scheduler adjusts the learning rate at each step.
- Training accuracy and loss are recorded for each epoch.

Evaluation Process

- The model is set to evaluation mode.
- For each batch in the validation set, input IDs, attention masks, and labels are fed into the model.
- The model computes logits and loss.
- Validation accuracy and loss are calculated and recorded.
- Train and Test Accuracy
- Train Accuracy: The accuracy of the model on the training data after each epoch.
- Validation Accuracy: The accuracy of the model on the validation data after each epoch.
- Best Model: The model state with the highest validation accuracy is saved**.**

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/Visheshshahu-06/Capstone-Project