To,

IITD-AIA Foundation on Smart Manufacturing

Subject: Weekly Progress Report for Week-11

Respected sir,

Following is the required progress report to the best of my knowledge considering relevant topics to be covered.

Day 16:

Generative Adversarial Networks .

GANs use a unique adversarial training process involving a generator and a discriminator to achieve this goal.

Steps involved:

- • Initialize generator and discriminator with random weights.
- • Train in alternating steps:
- • Generator creates fake data.
- • Discriminator evaluates real vs. fake data.
- • Adjust weights based on how well each performs.
- • Repeat until generator produces convincing data.
- • Evaluate generated data quality.

Learned by example :

```
// Step 1: Initialization

initialize_generator()

initialize_discriminator()

// Training Iterations

for iteration in range(num_iterations):

    // Step 2a: Generator Update

    fake_data = generate_fake_data()

    generator_loss = compute_generator_loss(fake_data)

update_generator_weights(generator_loss)
```

```
 // Step 2b: Discriminator Update

    real_data, labels_real = get_real_data_and_labels()

    discriminator_loss_real = compute_discriminator_loss(real_data, labels_real)

    fake_data, labels_fake = get_fake_data_and_labels()

    discriminator_loss_fake = compute_discriminator_loss(fake_data, labels_fake)

    total_discriminator_loss = discriminator_loss_real + discriminator_loss_fake

    update_discriminator_weights(total_discriminator_loss)

// Convergence and Evaluation

if is_converged():

    generated_data = generate_data_using_generator()

    evaluate_data_quality(generated_data)
```

Reference: https://machinelearningmastery.com/what-are-generative-adversarial-

networks-gans/

Day 15:

Keras Embedding .

The Keras Embedding layer transforms words into meaningful numbers for computers to understand in tasks like language processing. It helps capture word relationships and is a vital part of many NLP models.

Steps:

- • Import necessary libraries (TensorFlow, Keras).
- • Create a vocabulary by assigning unique integers to words/tokens.
- • Convert text data into integer sequences using the vocabulary.
- • Build a Keras sequential or functional model.
- • Add an Embedding layer as the first layer, specifying input dimension (vocab size) and output dimension (embedding size).
- • Compile the model with loss, optimizer, and metrics.
- • Train the model using integer-encoded sequences and labels.
- • Learned embeddings are updated during training.
- • Use embeddings in downstream layers (LSTM, GRU, etc.) for specific tasks.
- • Optionally, use pre-trained word embeddings for transfer learning.
- • Optionally, fine-tune embeddings and other model parts.
- • Evaluate model performance on validation/test data.
- • Deploy the trained model for inference on new data.

Reference:
https://keras.io/api/layers/core_layers/embedding

Day 14:

Streamlit .

Streamlit is an open-source Python library designed to simplify the process of creating interactive web applications for data science and machine learning projects

Steps involved :

- • Install Streamlit: Use pip to install Streamlit library.
- • Create Python Script: Start a new .py script for your app.
- • Import Libraries: Import necessary libraries, especially Streamlit.
- • Design Your App: Use Streamlit commands to structure your app.
- • Add Widgets: Integrate widgets like sliders, buttons, text inputs.
- • Load Model : Load your ML model into the app.
- • Display Results: Show model predictions or outcomes.
- • Run App: Launch the app using streamlit run script.py.
- • Interact: Access your app via the provided URL.
- • Enhance: Customize further using Streamlit's features.
- • Deploy

Reference:

https://streamlit.io/

https://github.com/streamlit/streamlit

Day13:

Gradio

It allows developers to quickly build and deploy interactive web-based applications without requiring extensive frontend development knowledge. With Gradio, we can easily create interfaces where users can input data, interact with your model, and see the results in real-time.

Steps involved:

- • Install Gradio with pip install gradio.
- • Import necessary libraries: Gradio and your ML model.
- • Define input and output functions for your model.
- • Create an interface using Gradio's Interface class.
- • Specify input and output functions and UI components.
- • Launch the interface with the launch method.
- • Users interact with the UI to provide input.
- • Gradio passes input to your model and displays real-time output.
- • Share the interface by deploying it on a server or sharing a link.
- • Iterate based on feedback: refine layout, handle errors, and improve functionality.
- • Monitor and maintain the interface's performance and user experience

Reference:

https://gradio.app/

https://www.machinelearningnuggets.com/gradio-tutorial

Day 12:

ResNet and Pickle file .

ResNet is a powerful type of neural network for images. It uses shortcuts to build deeper networks, making it easier to train accurate models even with lots of layers. This helps computers recognize objects and patterns better in pictures.

Steps:

- • Take an image as input.
- • Process the image through layers of pattern detectors .
- • Use shortcut connections to help information flow better.
- • Stack shortcut-enhanced blocks to build a deep network.
- • Learn simple features in early blocks and complex patterns in deeper blocks.
- • Make the image smaller using pooling or striding.
- • Decide what's in the image using fully connected layers.
- • Teach the network using labeled images, adjusting internal settings .
- • Test and adjust the network to improve its predictions.
- • Repeat training and testing with many images to refine performance.
- • Use the trained network to recognize objects in new images.

Pickle file - It's a way to save the model's details and state so you can easily store it on disk and reload it later. This helps you avoid retraining and allows you to share or deploy the model with others.

Steps:

- • Serialization: Convert Python object to bytes.
- • Save: Use pickle.dump() to store bytes in a file.
- • File: The saved file is now a pickle file.
- • Load: Employ pickle.load() to read the pickle file.
- • Object: The loaded bytes become a Python object again.

References: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/