JAI HOO SATNARYAN.

SIX WEEKS INTERNSHIP AT IIT JAMMU

INTERNSHIP REPORT

2nd WEEK

Day 8: Introduction to TinyML and Machine Learning Basics

Summary:

1. **Understanding Machine Learning:**
   - Machine learning (ML) can seem intimidating, especially with its association with complex mathematics, academic papers, and specialized Python libraries.
   - ML is a tool for solving problems using computers to predict outcomes based on past data.
   - It is accessible to anyone with basic programming knowledge, and you don't need a PhD to work with it.
2. **Key Concepts:**
   - Definition of machine learning and its applications.
   - Key terms and ideas in ML.
   - Workflow for solving problems with deep learning, a popular approach to ML.
3. **What is Machine Learning?**
   - ML is used to make predictions based on past data.
   - Example: Predicting machine breakdowns in a factory by analyzing data like production rate, temperature, and vibration.
   - Unlike traditional programming, ML algorithms learn from data to make predictions.
4. **Deep Learning:**

- A popular approach to ML based on a simplified model of the human brain.
- Uses networks of simulated neurons (arrays of numbers) to model relationships between inputs and outputs.
- Different architectures are suited to different tasks (e.g., image data vs. sequence prediction).

5. **Practical Application:**
   - Focus on deep learning models that can run on microcontrollers with limited memory and processing power.
   - Creating models that perform useful tasks within the constraints of tiny devices.

Notes:

- TinyML combines ML with embedded systems to create low-power, efficient models that can run on small devices.
- Emphasis is on practical, hands-on learning, with less focus on deep theoretical concepts.
- Key ML concepts and workflows are introduced to help understand and implement ML models on tiny devices.
- The ultimate goal is to empower the ability to build and modify ML applications on embedded systems.

Day 9: The Deep Learning Workflow

1. Decide on a Goal

Establish a clear objective for my algorithm. Define what i want to predict. This step involves transforming a business problem into a machine learning problem.

Example Goal: Predict whether a factory machine will break down soon.

2. Collect a Dataset

Identify and gather the data necessary for the model. Select relevant data that will help in solving the problem. Consider the data's availability during prediction time.

Example Dataset: Rate of production, temperature, and vibration readings from the machine.

3. Design a Model Architecture

Choose or design a neural network architecture suited to my problem and data. This involves deciding on the type of neural network (e.g., convolutional neural networks, recurrent neural networks) and its structure.

4. Train the Model

Train the model using my dataset. This involves feeding the data into the model and adjusting its parameters to minimize prediction errors.

5. Convert the Model

Once trained, convert the model into a format suitable for deployment on my target device. For TinyML applications, this might involve optimizing the model to run efficiently on a microcontroller.

6. Run Inference

Deploy the model on the target device and use it to make predictions on new data. Ensure the data fed into the model during inference is in the same format as the training data.

7. Evaluate and Troubleshoot

Assess the model's performance and troubleshoot any issues. Evaluate metrics like accuracy, precision, recall, etc. Iterate through the process to improve the model.

Step-by-Step Example:

1. **Goal**: Classify machine operation as "normal" or "abnormal."
2. **Dataset**: Collect temperature, vibration, and production rate data, ensuring these are available during predictions.
3. **Model Architecture**: Start with a simple neural network and refine iteratively.

4. **Training**: Use collected data to train the model.
5. **Conversion**: Optimize and convert the trained model for the Arduino TinyML kit.
6. **Inference**: Deploy the model on the Arduino and predict machine status in real-time.
7. **Evaluation**: Continuously monitor and adjust the model based on performance metrics.

Day 10: Building and Training a Model

Objective:

- Start building a simple machine learning model to predict sine wave values.
- Prepare the necessary tools and datasets.

Tasks:

1. **Setup Environment**
   - Install Python (if not already installed).
   - Install TensorFlow.
   - Access Google Colab.
2. **Obtain a Simple Dataset**
   - Create a dataset of sine wave values.
   - Use Python and numpy to generate the sine wave data.
3. **Train a Deep Learning Model**
   - Define a simple neural network using TensorFlow.
   - Train the model with the sine wave dataset.

Detailed Steps:

1. Setup Your Environment

- **Install Python:**
  - Download and install Python from the official Python website.
- **Install TensorFlow:**
  - Open terminal or command prompt and run:

```
pip install tensorflow
```

- **Access Google Colab:**
  - Go to Google Colab and sign in with Google account.
  - Create a new notebook.

Day 11: Python and Jupyter Notebooks, Google Colaboratory, TensorFlow, and Keras

Objective: Explore Python, Jupyter Notebooks, Google Colaboratory, TensorFlow, and Keras in the context of machine learning.

Tasks:

1. **Python and Jupyter Notebooks**
   - **Python Overview:** Python is widely used in machine learning due to its simplicity and extensive libraries for data handling and mathematics.
   - **Jupyter Notebooks:** Jupyter Notebooks allow for interactive development, combining code execution, visualization, and documentation in one place.
2. **Google Colaboratory (Colab)**
   - **Overview:** Colab is an online platform provided by Google for running Jupyter notebooks in the cloud.
   - **Advantages:**
     - No need to install dependencies locally; runs on Google's hardware.
     - Easily shareable and collaborative.
     - Supports accelerated hardware for faster training.
3. **TensorFlow and Keras**
   - **TensorFlow Overview:** TensorFlow is a comprehensive framework for building, training, and deploying machine learning models.
   - **Keras:** Keras, integrated with TensorFlow, provides a high-level API for building and training neural networks.

- **Use in TinyML:** TensorFlow Lite enables deploying models on mobile and embedded devices, crucial for TinyML applications.

Day 12: Understanding Deep Learning Concepts

Objective: To grasp essential deep learning concepts for developing TinyML applications.

Reasons for Study: Understanding these concepts will help in building accurate and efficient machine learning models for microcontrollers, which is crucial for your TinyML project.

Topics Covered:

1. **What is Deep Learning?**
   - **Explanation:** Deep learning is a type of machine learning that uses neural networks to learn patterns from data. It's great for tasks like recognizing images, understanding speech, or predicting outcomes based on complex data.
   - **Example:** Think of training a model to identify different breeds of dogs from pictures. The deep learning model learns to recognize unique features like ears, fur patterns, and snout shapes.
2. **Deep Learning vs. Machine Learning**
   - **Explanation:** Deep learning is a subset of machine learning. Unlike traditional machine learning, deep learning models can automatically learn hierarchies of features from raw data, which is useful when the data is complex and unstructured.
   - **Example:** Comparing a basic machine learning model used for predicting weather based on historical data with a deep learning model used for detecting emotions from facial expressions in real-time video.
3. **Types of Neural Networks**
   - **Explanation:** Neural networks are models inspired by the human brain's structure. Different types of neural networks are suited for different tasks, such as recognizing images (Convolutional Neural Networks) or predicting sequences (Recurrent Neural Networks).

- **Example:** Using a Convolutional Neural Network (CNN) to automatically tag photos on social media based on their content, like identifying people, objects, or locations.

4. **History and Evolution of Deep Learning**
    - **Explanation:** Deep learning has evolved significantly over the years with advancements in computing power and algorithms. Early models were basic, but today's deep neural networks can handle vast amounts of data and complex tasks.
    - **Example:** From early experiments with perceptrons in the 1950s to breakthroughs in deep learning with AlphaGo's victory in playing Go, a complex board game.
5. **Applications of Deep Learning**
    - **Explanation:** Deep learning is applied in various fields, from healthcare (diagnosing diseases from medical images) to entertainment (recommendation systems for movies) and robotics (autonomous driving).
    - **Example:** Using deep learning models in self-driving cars to detect pedestrians, traffic signs, and obstacles on the road to make real-time driving decisions.
6. **What is a Perceptron?**
    - **Explanation:** A perceptron is the simplest form of a neural network, mimicking a single neuron's decision-making process. It takes inputs, applies weights, and produces an output based on an activation function.
    - **Example:** Training a perceptron to decide whether to play tennis based on weather conditions like temperature, humidity, and wind speed.
7. **Perceptron Training Techniques**
    - **Explanation:** Techniques like gradient descent help adjust a perceptron's weights iteratively to improve its accuracy in predicting outcomes.
    - **Example:** Adjusting a perceptron's weights to better predict if a customer will buy a product based on their shopping history and demographics.

Day 13 and Day 14: Weekend Break

Given that these days were designated as weekend breaks, no specific technical learning or activities related to the project were conducted during this period.