EC-ENG 535/635 (Fall 2024)

Course Projects

Fatima Anwar

fanwar@umass.edu

UMass Amherst

Implementation + Research

1. Data Collection in Mixed Reality for Privacy Analysis

- **Background**: Head-mounted mixed-reality devices can render and overlay realistic 3D content onto the physical environment, enabling a wealth of interactive possibilities. These technologies are also used to track human pose. These devices are equipped with a rich set of sensors that collect personal and sensitive information (e.g., body motion, eye gaze, hand joints, and facial expression), which can be used to uniquely identify a user, even without explicit identifiers
- **Goal**: Gather sensor data from headset, preprocess it, and perform privacy analysis on it
- **Deliverables**:
  - Configure the setup for data collection
  - Identify the type of applications for experiments
  - Design experiments for the subject wearing headset and the person in the scene
  - Manage and label the collected data
  - Perform privacy analysis on the data
- **Hardware/Software:** Windows Laptop with Universal Windows Platform support, Hololens 2, Python, Unity, MRTK
- **References**:
  - BehaVR: User Identification Based on VR Sensor Data, 2023
  - Personal Identifiability of User Tracking Data During Observation of 360-degree VR Video, 2020
  - Understanding User Identification in Virtual Reality Through Behavioral Biometrics and the Effect of Body Normalization, 2021
- **Team Size: 2-3**

Implementation focus

2. Teleportation into Mixed Reality World

- **Background**: Mixed Reality systems can teleport a participant to a remote physical location enhanced with virtual overlays to provide spatial context. The goal is to enhance content sharing and ease of collaboration. To enable this, 360 cameras capture the physical environment and overlay virtual content on the captured video stream. The high fidelity 360-video provides better collaboration and a high sense of telepresence compared to traditional display systems. View independence between the local and remote user improves the speed of collaboration. 360-video only teleports physical setup and co-located users to remote users. The goal is to overlay virtual objects precisely on the 360-videos, and track those overlays as the scene changes.
- **Goal**: Stream the panoramic view of a physical space from one 360-camera to a VR headset. Also overlay virtual objects on the camera stream.
- **Deliverables**:
  - Configure the 360-camera to live stream a physical space
  - Attach a tracker to a physical object and attach a virtual augmentation to its corresponding view
  - Track the virtual object when its physical counterpart moves
- **Hardware:** Ricoh Theta Z1, HTC vive cosmos elite VR headset, trackers
- **References**:
  - Augmented Virtual Teleportation for High-Fidelity Telecollaboration (IEEE Transactions 2020)
  - Exploring interaction techniques for 360 panoramas inside a 3D reconstructed scene for mixed reality remote collaboration (Journal on Multimodal User Interfaces-2020)
  - SCeVE: A Component-based Framework to Author Mixed Reality Tours
- **Team Size:** 3

Implementation + Research + Creativity

3. Home safety system using LLM Agents

- **Background**: Large Language Models (LLMs) have become extremely popular due to their reasoning capabilities. LLM agents are autonomous systems that combine the power of standalone LLMs and tools such as search, weather, calculator, etc to handle more complex tasks and queries.
- **Goal**: Use LLM agents to design a home safety system.
- **Deliverables**:
  - Understand LLM agents. (tutorial provided)
  - Implement basic LLM agents using the tutorials.
  - Use LLM agents to design a home safety system. It could be anything. E.g., intruder detection, smoke/fire in the kitchen, stray animal detection etc. Be creative. The point here is that you should let the LLM do the reasoning and designing of the system. You should just equip the LLM with the necessary tools to be able to reason. The references and tutorials will help you build such a system.
  - The final output of the LLM should be some code snippet. E.g., in the case of fire detection, it should output code to train some object detection model on kitchen fire images.
- **Expected Hardware / Software:** Python, Laptop with CUDA-enabled GPU. Otherwise: Google Colab
- **References**:
  - Langchain tutorial(langchain is a popular framework to build agents. You can use any other framework of your choice)
  - A paper to understand how agents work and how different tools/models can be connected with them (https://arxiv.org/pdf/2303.17580)
- **Team Size**: 2
- **Prerequisites**: Basic understanding of applied Machine Learning
- **Project Complexity:** Medium
- Note: Since this project involves LLMs, it has a very open-ended nature. There could be hundreds of solutions for the above problem statement. This open-endedness should not be mistaken for lack of instructions. If you feel like you need more concrete instructions, then please select a non-LLM project.

Implementation + Research

4. Federated Learning on Multimodal Sensor Data

- **Background**: The goal of federated learning is to collaboratively train a machine learning model where the participating parties wish to keep their data private. Data can be of different modalities, i.e., can be images, audio, text, sensor data etc. Recently, there have been works in the field of multimodal federated learning. A variety of heterogeneous sensor deployments with multiple sensing modalities are emerging that serve a single learning application. The sensor data is private in nature and requires federated learning approach, hence the demand for multimodal FL in research community is increasing.
- **Goal**: Understand and benchmark different multimodal datasets in a federated setting
- **Deliverables**:
  - Understand multimodal federated learning (code:https://github.com/yuchenzhao/iotdi22-mmfl)
  - Use given datasets to reproduce the results in the paper
    - data: https://drive.google.com/drive/folders/1rWJYkfMavGs1F-H0jykJ5V0fIiwrQdJV
  - Perform a per-class accuracy analysis of the results and observe the effect of skewed data distribution on the per-class accuracy
  - Evaluate the system on a multimodal dataset that is relatively balanced in class distribution
- **Software:** Python, Laptop with CUDA-enabled GPU (optional)
- **References**:
  - Multimodal Federated Learning on IOT Data (https://pure-research.york.ac.uk/ws/portalfiles/portal/79047763/2109.04833v2.pdf)
  - Communication-Efficient learning of deep networks from decentralized data (http://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf)
- **Team Size**: 2
- **Prerequisites**: Basic understanding of applied Machine Learning

5. Federated Learning on Heterogeneous Sensor Data

- **Background**: Federated Learning is a popular machine learning paradigm that allows multiple parties to collaboratively train a joint(global) neural network without exposing their private data. However, the performance of the global model is impacted by the heterogeneity, i.e., skewness of the data of each individual participant.
- **Goal**: Characterize the impact of heterogeneity on FL. Then do the same characterization under adversarial conditions.
- **Deliverables**:
    - Understand Federated Learning and heterogeneity. (paper provided)
    - Implement FL with different datasets and different levels of heterogeneity.
    - Now include some adversarial attacks and perform the same characterization.
    - What do you observe in adversarial vs non-adversarial settings? Why do you think the impact of heterogeneity is different in both settings?
- **Expected Hardware / Software:** Python, Laptop with CUDA-enabled GPU. Otherwise Google Colab should work fine.
- **References**:
    - Communication-Efficient learning of deep networks from decentralized data (http://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf)
    - Example codes for different data partionings https://github.com/SMILELab-FL/FedLab/tree/master/tutorials/Datasets-DataPartitioner-tutorials (https://arxiv.org/pdf/2303.17580)
- **Team Size**: 2-3
- **Prerequisites**: Basic understanding of applied Machine Learning
- **Project Complexity:** Medium

6. Bias Analysis in Federated Learning for Heterogeneous Sensors

- **Background**: Federated learning (FL) is a ML framework in which a single model is trained using data collected from multiple IoT devices. To ensure privacy, these devices avoid sharing their data with a central server. Instead, each device independently performs computations or training on its own data and shares only the model parameters. These shared parameters are then aggregated by the server to form a global model. Due to heterogeneity in data distribution across distributed clients, the global model performance varies for different clients. This discrepancy in accuracy across clients is called bias in FL.
- **Goal**: Focus on model bias towards different groups due to variations in feature distribution
- **Deliverables**:
  - Understand and implement different FL techniques – Federated averaging (FedAvg), Tilted Empirical Risk Minimization (TERM), and Agnostic FL (AFL) (code provided)
  - Utilize a dataset for training and assessment of two FL techniques (data provided)
  - Examine the enhancement in the variance of accuracy among individual client groups when employing various federated learning techniques.
- **Software:** Python, Laptop with CUDA-enabled GPU / Google Colab
- **References**:
  - Papers - Communication-Efficient Learning of Deep Networks from Decentralized Data (https://arxiv.org/abs/1602.05629); Tilted Empirical Risk Minimization (TERM) (https://openreview.net/pdf?id=K5YasWXZT3O); Agnostic Federated Learning (AFL) (https://arxiv.org/pdf/1902.00146.pdf)
    - Code: FedAvg (https://github.com/alexbie98/fedavg); TERM (https://github.com/litian96/TERM); AFL (https://github.com/YuichiNAGAO/agnostic_federated_learning)
    - Data: CIFAR-10 (https://www.kaggle.com/c/cifar-10/data), FashionMNIST (https://github.com/zalandoresearch/fashion-mnist)
- **Team Size**: 2-3
- **Prerequisites**: Basic understanding of applied Machine Learning

Implementation + Research

7. Bias Mitigation in Federated Learning

- **Background**: Federated learning (FL) is a ML framework in which a single model is trained using data collected from multiple IoT devices. To ensure privacy, these devices avoid sharing their data with a central server. Instead, each device independently performs computations or training on its own data and shares only the model parameters. These shared parameters are then aggregated by the server to form a global model. Due to heterogeneity in data distribution across distributed clients, the global model performance varies for different clients. This discrepancy in accuracy across clients is called bias.
- **Goal**: Understand various local model aggregation approaches in FL with different datasets across devices, particularly focusing on their bias towards different groups due to variations in feature distribution.
- **Deliverables**:
- Understand and implement different FL techniques in FL – Federated averaging (FedAvg), Agnostic Federated Learning, and FedNTD (code provided)
- Utilize dataset for the training and assessment of the suggested FL techniques. (data provided)
- Analyze the dynamics of group bias when employing various techniques in different FL settings.
- **Expected Hardware / Software:** Python, Laptop with CUDA-enabled GPU
- **References**:
  - Papers - Communication-Efficient Learning of Deep Networks from Decentralized Data
  - (https://arxiv.org/abs/1602.05629); Agnostic Federated Learning (https://arxiv.org/pdf/1902.00146.pdf); FedNTD (https://arxiv.org/pdf/2106.03097)
  - Code: FedAvg (https://github.com/alexbie98/fedavg); AFL (https://github.com/YuichiNAGAO/agnostic_federated_learning)
  - Data: CIFAR-10, FashionMNIST (https://github.com/SMILELab-FL/FedLab)
- **Team Size**: 2
- **Prerequisites**: Basic understanding of applied Machine Learning
- **Project Complexity:** Medium

Machine Learning Short Courses

- **Machine Learning** short courses:
  - https://www.youtube.com/watch?v=gZmobeGL0Yg&list=PLZbbT5o_s2xq7LwI2y8_QtvuXZedL6tQU
  - https://www.youtube.com/watch?v=v5cngxo4mIg&list=PLZbbT5o_s2xrfNyHZsM6ufI0iZENK9xgG
- **Federated Learning:**
  - https://www.youtube.com/watch?v=X8YYWunttOY
- **LLM Agents:**
  - Langchain tutorial

Implementation + Research

8. Time Synchronization Via Sensing

- **Background**: Time synchronization is an essential part of networked embedded devices. It enables various distributed applications like smart homes, automated agriculture and autonomous vehicles. However, many devices at the edge are resource constrained and existing timing services are a burden on their resources. On the other hand most of these devices are equipped with sensors and send periodic sensing data to gateway servers. We want to leverage this time-stamped sensor data received at the gateway to synchronize the sensing devices
- **Goal**: Develop a time sync protocol using timestamped sensor data from two embedded devices. The data will be received by a raspberry pi device, and the synchronization protocol will run on the raspberry pi device
- **Deliverables**:
  - Characterize the network delay between raspberry pi and the edge device
  - Estimate the relative clock drift between the participating devices
- **Hardware:** ESP32 Things, Raspberry Pi
- **References**:
  - HAEST: Harvesting Ambient Events to Synchronize Time across Heterogeneous IoT Devices
  - Automated Synchronization of Driving Data Using Vibration and Steering Events
  - Exploiting Smartphone Peripherals for Precise Time Synchronization
- **Team Size:** 2
- **Prerequisite:** Experience with programming embedded microcontrollers (e.g. 8051, AVR), Bluetooth communication, Basic Machine Learning Algorithms

9. Emulate ARM TrustZone on a Virtual Machine

- **Background**: Hypervisor allows multiple operating systems in parallel on the same hardware. This technology is increasingly adopted on embedded platforms. However, the hypervisors can be compromised by malicious agents that would lead to attacks on all the operating systems. To mitigate such scenarios, modern hardware platforms provide trusted execution environments (TEEs) that are protected against attacks by privileged adversary like hypervisors. However, modern hypervisors have limited support for Trustzone (ARM's TEE) and the potential for Trustzone to provide secure services to virtualized systems (systems that run hypervisor) is yet untapped. The aim of this project is to build a development environment which runs **Xen** hypervisor with support for ARM Trustzone. Such an environment would enable research that enhances security of virtualized embedded systems
- **Goal**: Run Xen hypervisor with support for OPTEE (Trustzone Software) on QEMU emulator and run performance benchmarks
- **Deliverables**:
  - Understand how a virtualized system works, specifically, how does hypervisor (Xen) run multiple operating systems on the same hardware platform such as ARM
  - Demonstrate a functioning environment running Xen on Qemu emulator for ARM based platforms. (Tutorials provided, the goal is to identify and fix errors in them)
  - Modify the above environment to run OPTEE using ARM Trustzone
  - xtest is a benchmark that evaluates the performance of ARM Trustzone software (i.e. OPTEE). Build and run xtest on a single and multiple guest operating systems
- **Software:** QEMU, Linux, Xen, OPTEE
- **References**:
  - Xen on ARM with virtualization Extension: White Paper (https://wiki.xenproject.org/wiki/Xen_ARM_with_Virtualization_Extensions_whitepaper)
  - Xen support for Trustzone (https://optee.readthedocs.io/en/latest/architecture/virtualization.html#implementation-details, https://wiki.xenproject.org/wiki/Xen_Project_4.13_Feature_List)
  - Tutorials: https://wiki.xenproject.org/wiki/Xen_ARM_with_Virtualization_Extensions/qemu-system-aarch64, https://medium.com/@denisobrezkov/xen-on-arm-and-qemu-1654f24dea75, https://gist.github.com/stewdk/110f43e0cc1d905fc6ed4c7e10d8d35e
- **Team Size: 3**
- **Prerequisites**: Basic understanding of how operating systems work, knowledge of building linux kernel from source code and familiarity with use of linux terminal

Logistics

Course TAs

- Two TAs for this course:
  - **Primary TA:** Momin Khan, makhan@umass.edu
  - **TA:** Khotso Selialia, kselialia@umass.edu

Course Project Logistics

- You will choose your projects from this list,
  - If you have a project idea that is relevant to the course, discuss it with me
- Only one team can choose hardware-dependent projects
- Make use of mine/TA office hours and appointments to talk about the project
  - We will have three check-in sessions in Oct/Nov/Dec
- Come in groups with your team members and come prepared!
- Ask questions in the class about the projects and do your research before talking to me
- I encourage critical thinking and not spoon feeding
- Visualizations are important; demo certain aspect of the project at the end

- **Group Formation**: Many ways to form groups,
  - Talk to people directly you want to work with - easier to do this with two person teams
  - Email me and primary TA about your project interest and your strong skill; we will match people into groups
- **Hardware Components Delivery**:
  - One student per group will pick up hardware from my Lab after signing checkout sheet

Deliverables & Timeline

- Email me your project choice as a group
  - Explain how your team is suitable for taking on that project
- Then Create GitHub Repository, and add the following:
    - Motivation, design goals, deliverables, system blocks, hw/sw requirements, team members responsibilities, project timeline, references
    - Mention team member's lead roles**:** setup, software, networking, writing, research, algorithm design
  - Email me your project name, group members names, their roles, and Github repository by Sep 29th
- Details on final project demonstration and report will be posted later

Questions?