**SPROJ\_EECE\_7398\_MXX\_V02\_Milestone\_2**

Welcome back to our fifth tutorial series. In this video, we'll explore the project repository structure, clone it to our local machine, and set up the development environment. Let's get started.

I'm now going to share my screen and show you the GitHub repository we'll be working with. The fidip project is focused on infant pose estimation with small data. And you can see from the readme that it's a well documented research project.

Looking at the commit history, we can see that it's an actively maintained repository with recent updates. Let's first take a look at the repository structure. This is a typical research repository structure. Let's quickly review what each directory is for.

Experiments/coco, configuration files for different experimental setups. Lib for library. Core implementation code and libraries. Syn generation as for synthetic generation. Code for generating synthetic infant poses, a key small data technique. Tools, scripts for training, testing, and other operations. Fidip environment.yml. Environment configuration for conda.

When we run the project, additional directories will be created automatically. Data, where our data set will be stored. Log, where training logs will be saved here. Models, where pre-trained models are stored. And output, where results from model runs.

Take a moment to notice how well structured the repository is. This is common in research codebases. And learning to navigate them is an important skill. The first step is to clone the repository to our local machine.

Before we continue, let me share some tips about working with research repository. Always start with the readme. Before diving into the code, thoroughly, read the readme file.

It usually contains crucial information about the project, its dependencies, and how to set it up. Look for paper references. Research repos often implement methods from academic papers. Reading or skimming the reference paper can give you valuable context.

Identify the core components. Try to identify the main parts of the code base-- data processing, model architecture, training loop, and evaluation methods. Check for configuration files.

Many research projects use configuration files, such as YML, JSON, and other ways to manage experimental settings. Understanding these will help you modify the project for your needs. Don't feel overwhelmed. Research code can be complex, but you don't need to understand every line to use it effectively. Focus on the high level flow first.

Now, let's continue with setting up our environment. The readme mentions we need Python 3.12 with Cuda 12.1 on Ubuntu 18.04 and an NVIDIA GPU. For our tutorial, we'll use the provided conda environment file.

Let's go ahead and create our fidip environment from the provided YML file. Now let's activate our fidip conda environment. Now we need to compile some custom libraries.

We go to the lib directory. We use the make command. Let's go back to our main directory. Setting up environments for deep learning projects can sometimes be tricky. Here are some troubleshooting tips.

Version conflicts. If you encounter version conflicts in the conda environments, try creating a new environment with just the core dependencies first/ then manually install the rest

Cuda compatibility. Make sure your NVIDIA drivers are compatible with Cuda version required. You can check your driver version with NVIDIA SMI.

Here we can see our NVIDIA SMI command working perfectly. We can see the Cuda version. We can see the GPU name and all the processes running.

Verifying Cuda is working. After setting up your environment, it's crucial to verify that PyTorch is actually using your GPU. Here's how to check.

Here, as you can see, there is a GPU test Python script where we Import the torch library and test its availability. When we run it, we can see the basic Cuda information, the PyTorch version, the Cuda version we're using, and the available devices. And we can see the test is complete.

If Cuda is properly configured, you should see your GPU information, library compilation errors. If the make command fails, check for missing dependencies or compile flags. The error message often provide hints about what's missing.

Great. We've successfully cloned the repository and set up our environment. Now, our development environment is ready. In the next lesson, we'll download the pre-trained models and prepare the data needed for our infant pose estimation project. See you there.