# LAB EXPERIMENT 2:

## Basics of Python, Jetson Orin Nano and it's Tools

By: Gowtham Kumar Kamuni (A20549435)

Instructor: Dr. Jafar Saniie

ECE 501

Lab Date: 05-24-2024

Due Date: 05-30-2025

Acknowledgment: I acknowledge all of the work (including figures and codes) belongs to me and/or persons who are referenced.

Signature: Gowtham Kumar Kamuni

# I. Introduction

## A. Purpose

The main goal of this lab was to get hands on experience working with the Jetson Orin Nano and to understand how Python and AI models can run directly on edge devices. We started from scratch, flashing the OS, setting up the environment, and working inside the Linux terminal, to get a feel for how real world embedded AI systems are built and tested. Along the way, we explored how Python works on the Jetson, ran some basic programs, and finally used a pre-trained model to classify images in real time. This experiment gave us a full picture of how software interacts with hardware in modern AI-based systems.

## B. Background

Jetson Orin Nano is a powerful little device built for running AI tasks at the edge, meaning it doesn't need a cloud connection to process data. It comes with a strong GPU and CPU combo, and supports CUDA, Python, and a lot of deep learning libraries. The idea behind this lab was to use this board like a developer would in a real project, boot it, program it, and test it live.

We started by flashing the Jetson with Ubuntu and setting it up with a monitor, mouse, keyboard, and internet. After booting in, we wrote and ran simple Python programs directly on the device. We then cloned NVIDIA's jetson-inference repository, built it from source using cmake and make, and used it to classify images using a model called GoogLeNet.

Throughout this lab, we not only learned how to work with embedded AI, but also saw its limits, like how the model struggled with some real-world images. That gave us insight into how models behave outside training environments, and how hardware and software need to be tuned for real accuracy.

## II. Lab Procedure and Equipment List

### A. Equipment

- Jetson Orin Nano Developer Kit
- MicroSD Card (flashed with Ubuntu 20.04 Jetson image)
- External monitor with DisplayPort
- USB Keyboard and Mouse
- Power supply (barrel jack)
- Internet connection (Wi-Fi or Ethernet)
- MacBook Air (for transferring files and final report writing)
- Firefox browser (used on Jetson to download test images)

### B. Procedure

**1. Flashing the OS & Jetson Setup:**
We began by downloading the Jetson Orin Nano SD card image from NVIDIA's website and flashing it to the microSD card using Etcher. After inserting the card and powering on the Jetson, we connected a monitor, keyboard, and mouse to boot into the Ubuntu desktop environment.

**2. Running Basic Python Code:**
A working folder was created on the desktop, and we used the terminal and gedit to write and execute Python files. We started with a simple "Hello World" script and followed up with an object-oriented banking example that involved class definitions and method calls.

**3. Installing and Building jetson-inference:**
We updated the system and installed dependencies like git, cmake, and numpy. Then we cloned NVIDIA's jetson-inference GitHub repository, entered the build directory, and compiled the project using cmake and make.

**4. Running Image Classification:**
After building successfully, we ran the imagenet.py script to classify images using a pre-trained GoogLeNet model. We tested the model on various real-world images including objects, people, and household scenes. The script saved output images with labels and confidence percentages overlayed.

**5. Collecting Results and Screenshots:**
We tested a total of 6 unique images (eg: i1.jpg to i6.jpg) and captured their output classifications. The classification performance varied depending on image content and lighting, which gave us a realistic understanding of the model's strengths and limitations. Terminal screenshots were also taken to capture timing data for CUDA and CPU processing.

## III. Results and Analysis

Test Case 1: Timber Wolf


*Fig: Input and classified output for timber wolf (high accuracy)*

We tested a total of six images, each run through the imagenet.py script on Jetson Orin Nano using the GoogLeNet model. Below is the summary of the first classification test.

| Property | Value |
|---|---|
| Input Image File | image2.jpg |
| Output Image File | output2.jpg |
| Description | A timber wolf walking in snow |
| Predicted Class | Timber wolf |
| Confidence | 91.27% |
| Notes | Prediction was highly accurate |

To explore the limitations of the GoogLeNet model and the ImageNet dataset it was trained on, I decided to test the classifier using real world images captured directly in my apartment. The goal was to see how well the model handles everyday objects in natural indoor lighting conditions. Below are a few selected input and output results. The remaining image tests and classifications are provided in a separate folder for reference.

Test Case 2: Backpack


*Fig: Backpack misclassified as blue jeans (moderate confidence)*

| Property | Value |
|---|---|
| Input Image File | i1.jpg |
| Output Image File | output_i1.jpg |
| Description | A black backpack on a couch |
| Predicted Class | Blue jeans / denim |
| Confidence | 62.57% |
| Notes | Misclassified due to texture and color. Jetson likely mistook the black canvas material and folded fabric as denim. |

Test Case 3: Fruits & Vegetables

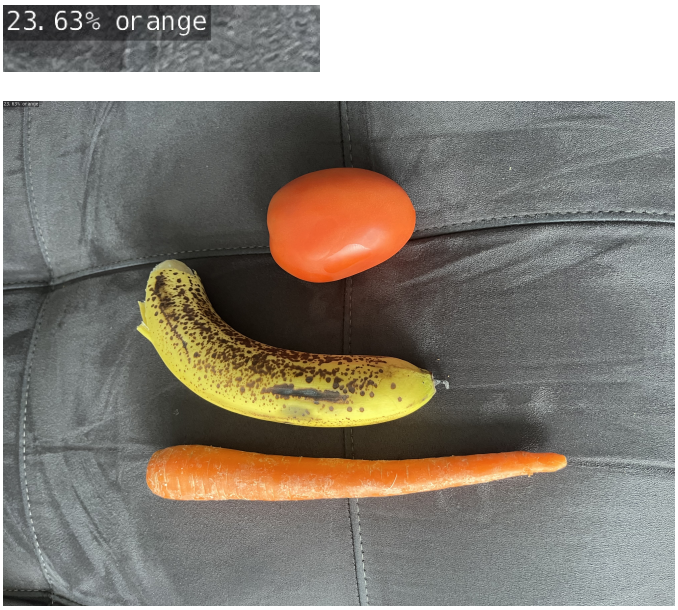



*Fig: Mixed fruits misclassified as orange (low confidence)*

| Property | Value |
|---|---|
| Input Image File | i3.jpg |
| Output Image File | output_i3.jpg |
| Description | A banana, carrot, and tomato placed on a dark surface |
| Predicted Class | Orange |
| Confidence | 23.63% |
| Notes | The model selected "orange" despite none of the objects being oranges. This was likely due to the dominant color in the tomato and carrot blending visually. |

Test Case 4: Electric Guitar


*Fig: Electric guitar correctly classified (very high confidence)*

| Property | Value |
|---|---|
| Input Image File | i5.jpg |
| Output Image File | output_i5.jpg |
| Description | A red electric guitar placed vertically against a wall |
| Predicted Class | Electric guitar |
| Confidence | 98.57% |
| Notes | Excellent recognition. The shape and features of the guitar were clearly identified by the model, showing high reliability for structured object types. |

These tests gave me a better understanding of how pre-trained models like GoogLeNet work in real-world situations. While the model performed really well on clearly defined objects like the guitar and the timber wolf, it struggled with less structured scenes or images containing multiple objects. Overall, this part of the

lab helped me evaluate how deep learning models behave outside training environments and showed how important context, lighting, and dataset relevance are for accurate classification.

## A. Discussion

### 1. Advantages and disadvantages of Jetson Orin Nano compared with other single-board computers

The biggest advantage of the Jetson Orin Nano is its GPU power. It's built to run AI models, and that makes it very different from something like a Raspberry Pi. I was able to run real-time image classification on it using CUDA, and the performance was smooth. The hardware is optimized for deep learning, with support for TensorRT and NVIDIA libraries built-in. It also has a lot of I/O options.

On the downside, the setup is heavier, flashing the OS, installing dependencies, and compiling code takes time. It's not as beginner friendly or "plug and play" as simpler boards. Also, I checked the price, it is high, and the learning curve is steeper if you're not familiar with Linux or AI pipelines.

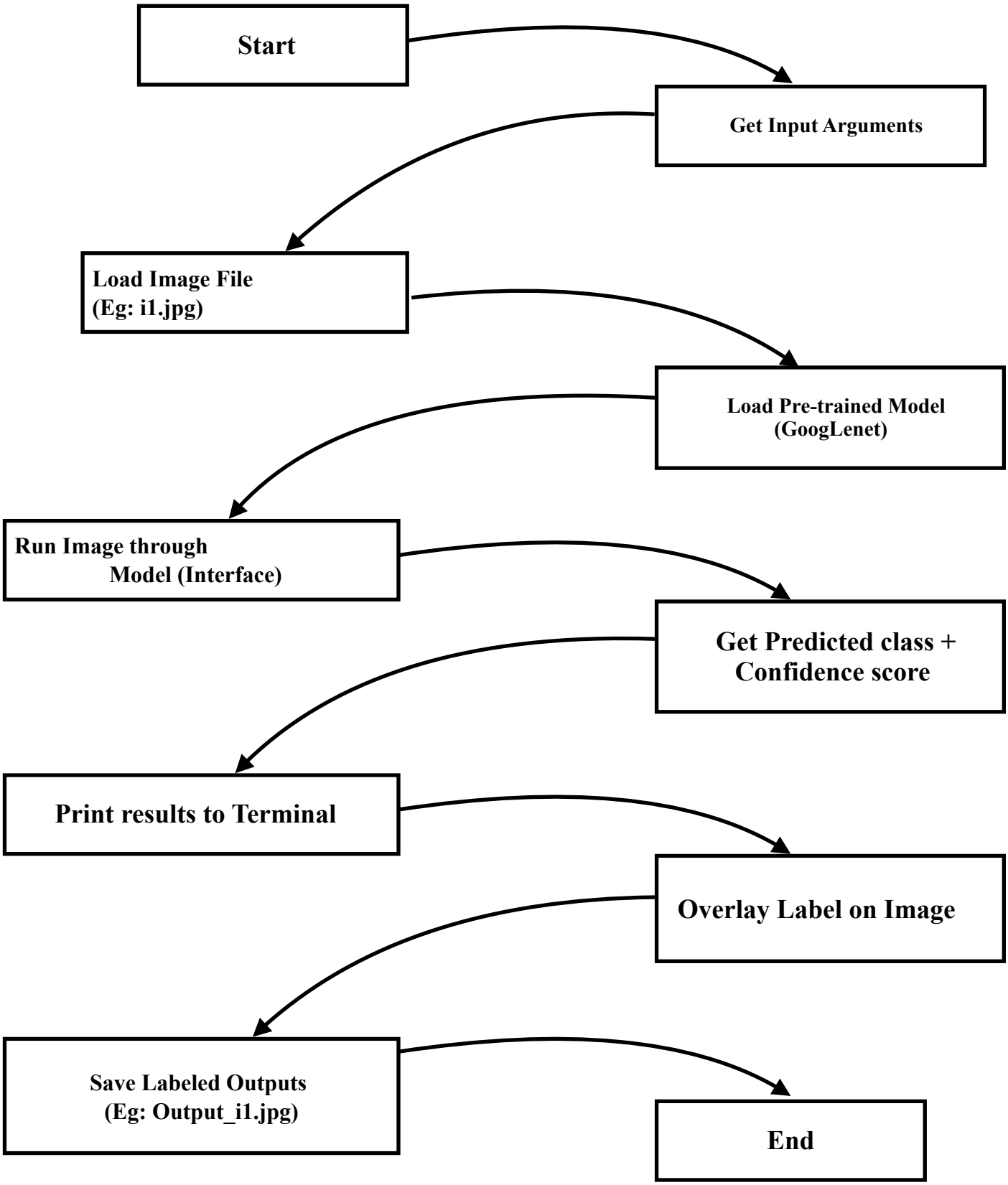
### 3. Why is it advantageous to use the power jack instead of Micro-USB?

Micro-USB just doesn't provide enough power, especially when running GPU based tasks. During the build process and when running inference, the Jetson pulls more power than Micro USB can reliably deliver. Using the power jack made the boot process stable, the system didn't lag, and there were no random shutdowns. For this kind of edge AI application, stable power is a must.

### 4. Why is the first run of the image recognition program slower than the later ones?

The first time I ran the classifier, I noticed it took longer to give a result. This is because Jetson compiles and optimizes the model using TensorRT the first time it runs. That initial optimization gets cached, so future runs are faster. It's like the model is "warming up" and preparing for real-time execution. After that, everything runs more smoothly with minimal delay.

## 2. Flowchart and explanation of the program in Appendix A



## IV. Conclusions

This lab was one of the most practical and insightful experiences I've had with embedded systems and AI. It started with setting up the Jetson Orin Nano completely from scratch, flashing the OS, connecting the hardware, and configuring everything using the Linux terminal. That process alone gave me a solid understanding of what it takes to prepare a real edge AI platform for development.

Once the system was running, I explored basic Python programming directly on the Jetson and slowly moved into more complex tasks like building NVIDIA's jetson-inference library from source. Compiling code with cmake and make, resolving dependencies, and understanding how pre-trained models like GoogLeNet are integrated taught me how software and hardware come together in an AI pipeline.

Running real-time image classification was where things got even more interesting. Some predictions were impressively accurate, like the electric guitar and the timber wolf, while others were clearly off, misclassifying objects based on color or shape. These results weren't failures; they were valuable insights into how AI models behave in uncontrolled, real-world settings. I got to see the strengths of pre-trained models as well as their limits when used outside of their training domain.

Overall, this lab wasn't just about writing code or following steps, it was about building, experimenting, testing, and understanding. It helped me think critically about model performance, dataset limitations, and the importance of choosing the right tools for a task. Most importantly, it showed me that building AI on embedded hardware is absolutely possible, and incredibly powerful, when done right.

## References


[1] Lab Experiment 2 Manual

[2] NVIDIA Jetson Orin Nano Developer Kit –

https://developer.nvidia.com/embedded/jetson-orin-nano-devkit

[3] ImageNet (ILSVRC) Dataset Overview –
http://www.image-net.org/

[4] Personal Experiments and Observations Conducted on Jetson Orin Nano during ECE 501 Lab 2 (May 2024)