Software Design Document

for

Custom Image Classifier

Government Engineering College, Wayanad

Guided by Prof. Nikesh P

Group No 2

Jithin T Mathews (WYD16CS030)

Amrutha K V (WYD16CS010)

Jithin K M (WYD16CS029)

Prasila P (WYD16CS045)

September 20, 2020

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to delineate the design of our project titled - ' Custom Image Classifier' in different Design Viewpoints.

1.2 Scope

Our Custom Image Classifier aims to classify objects or images the neural network is not trained to recognize and also provide a confidence score of the system.

1.3 Definitions, Acronyms and Abbreviations

1.3.1 Definition

- **Confidence Score** :- It is a measure of how sure the system is about its prediction, low confidence score implies the system is not sure about the input data its been given.
- **Keras** :- Neural Network library.
- **NumPy** :-NumPy is a python library used for working with arrays. It is used to store a large amount of data set.
- **Tensorflow**:- Open source library for machine learning
- **PyTorch** :- PyTorch is an open source machine learning library for Python.
- **Miniconda** :- Miniconda provides an external python environment so that changes in the python variables will not affect the system python environment.
- **ResNet-18 model** :- It's an eighteen layered neural network which we use in our project for training.

1.3.2 Abbreviations

- **CNN** :- Convolutional Neural Networks
- **IDE** :-Integrated Development Environment
- **AI** :- Artificial Intelligence
- **ResNet** :-Residual neural network
- **ONNX** :-Open Neural Network Exchange

2 References

- Transfer learning https://towardsdatascience.com/transfer-learning-using-mobilenet-and-
- jetson nano- https://courses.nvidia.com/courses/course-v1:DLI+C-RX-02+V1/about
- Basics behind CNNs http://www.subsubroutine.com/sub-subroutine/2016/9/30/cats-and-dogs
- AI learning https://news.developer.nvidia.com/get-started-with-ai-on-the-nvidia-jetson

3 Architectural Description

NVIDIA Jetson Nano is a small, powerful computer that lets us run multiple neural networks in parallel for applications like image classification, object detection. Jetson Nano is a GPU-enabled edge computing platform for AI and deep learning applications. The GPU-powered platform is capable of training models and deploying online learning models and is most suited for deploying pre-trained AI models for real-time high-performance inference. The neural network model supported by NVIDIA Jetson Nano is ONNX (Open Neural Network Exchange) model.ONNX supports interoperability between frameworks, which means we can train a model in one of the many popular machine learning frameworks like PyTorch which we have used in our project, and convert it into ONNX format and consume the ONNX model. The NVIDIA Jetson Nano module features a 1.43 GHz ARM Cortex-A57 quad-core processor and 128-core NVIDIA Maxwell graphics with speeds up to 921 MHz. Jetson Nano delivers 472 GFLOPs for running modern AI algorithms fast..

- BOARD SPECIFICATION

  1. GPU: 128-core NVIDIA Maxwell architecture-based GPU
  2. CPU: Quad-core ARMR©A57
  3. Video: 4K @ 30 fps (H.264/H.265) / 4K @ 60 fps (H.264/H.265)encode and decode
  4. Camera: MIPI CSI-2 DPHY lanes, 12x (Module) and 1x (DeveloperKit)
  5. Memory: 4 GB 64-bit LPDDR4; 25.6 gigabytes/second
  6. Connectivity: Gigabit Ethernet
  7. OS Support: Linux for TegraR
  8. Module Size: 70mm x 45mm
  9. Developer Kit Size: 100mm x 80mm

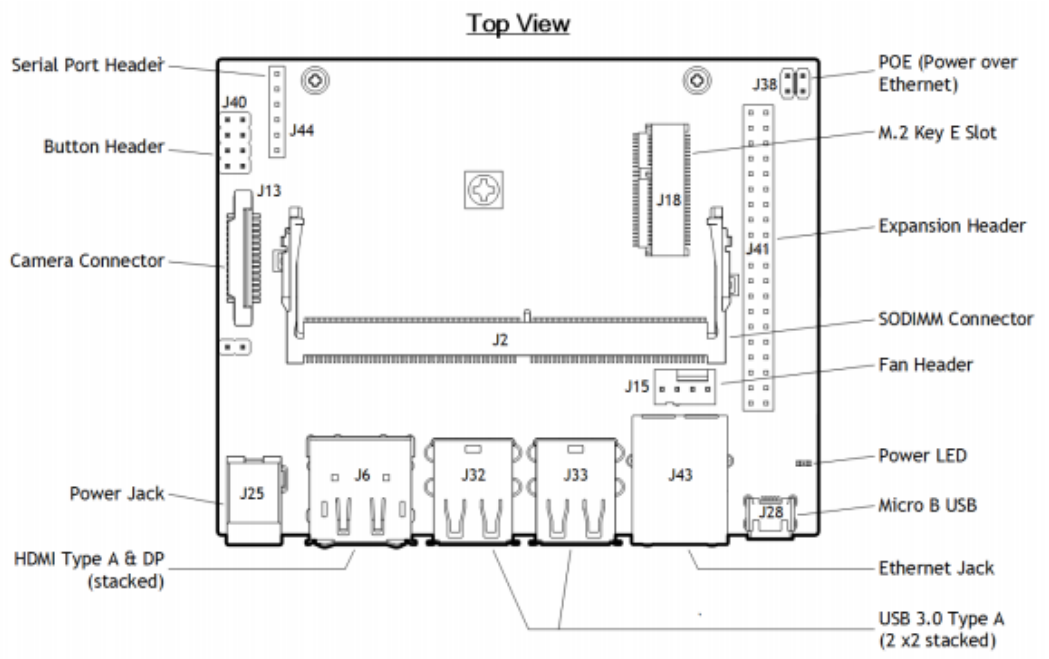

Figure 1: Nano Board

4 Decomposition Description

The System is divided into 3 phases based on the functionality of the system, for each of the functionality, the system would provide. The Modules are :-

- **Environment setup** - This phase consists of setting up a remote environment for the system to do training. This will prevent the system operations from not to change the operating system's environment variable and bring about critical errors. The system will install a python environment and create the environment with packages and libraries that will be needed for the training of the classification model.
- **Training** - This Module would be called after the collection of required training data. The training script will scan the input training data and associate each input data to a class that will be used to decide which classes will be new data which was not in the training set be. The purpose of this module would be to acquire the data for the generation of the model and save it for implementation. The developer would need to assign proper epoch values for the model to be accurate.
- **Real-time Classification** - This module will convert the saved model from the training set into a model that can be run on the jetson nano board. This module will use PyTorch IDE. The user interface will provide a live stream from the board along with the output of the object being read.

4.0.1 Component Decomposition

The System is divided into 3 components, two of which will be running on the same computer, one will include both computer and external hardware.

- The Component running on the computer will set up a remote working environment. This will
  - Isolate the system environment from our working environment.
  - Prevent system variable from changing accidentally.
  - Prevent system variable changes from affecting the working environment variables.
- The training Component will scan the thousands of training input data and form a model after doing chosen amounts of epochs.
- Third Component would be running on the Nvidia Jetson nano board. The real-time stream from the board will be shown in the user's interface and as the output of the classification model along with its confidence score.

5 Dependency Description

5.1 Intermodule Dependency

Consists of three modules which are independent of each other. The environment ensures that the training module variables and other settings do not reflect on the main system settings. It also prevents outside variables and settings from changing the training parameters. The training module uses the input data to make our classification model within the environment provided. The classification model will run the model created by the training model and provide output to the user.

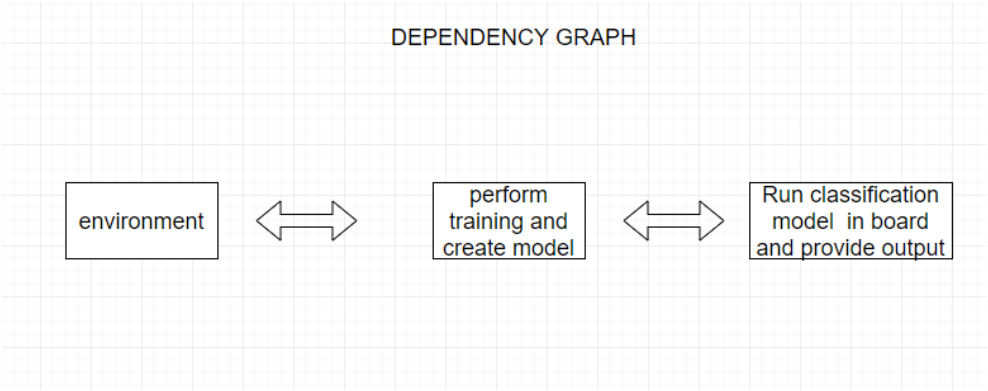

Figure 2: Data Flow Diagram Lvl 0

5.2 Process Model

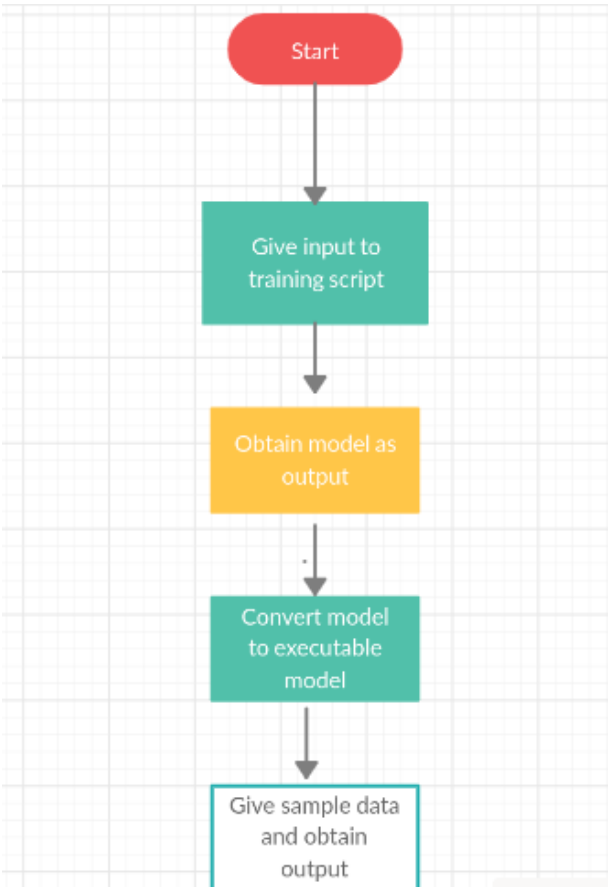

Figure 3: Process Model

The figure above shows an abstract view of the system training cycle. This has to be done until the output is of acceptable performance. A key part of the process is the construction of a model with peak performance. The system has an innate environment for the training of the model as well as an interface to implement the model with the Nvidia Jetson board and provide accurate output to the user.

6 User Interface Description

The UI will be kept minimal. In fact, a minimal interface can contain quite a lot of complexity. The key to minimalism isn't making the interface simple by itself. It's about making it as simple as it needs to be. In other words, it means stripping back unnecessary elements, and only keeping what's strictly necessary for the functioning of the website or app. When the model is executed, the User will be shown a live stream from the camera of the development board and the predicted class of the object in front of the camera. A terminal window also will be shown which depicts the classification computations of the class.

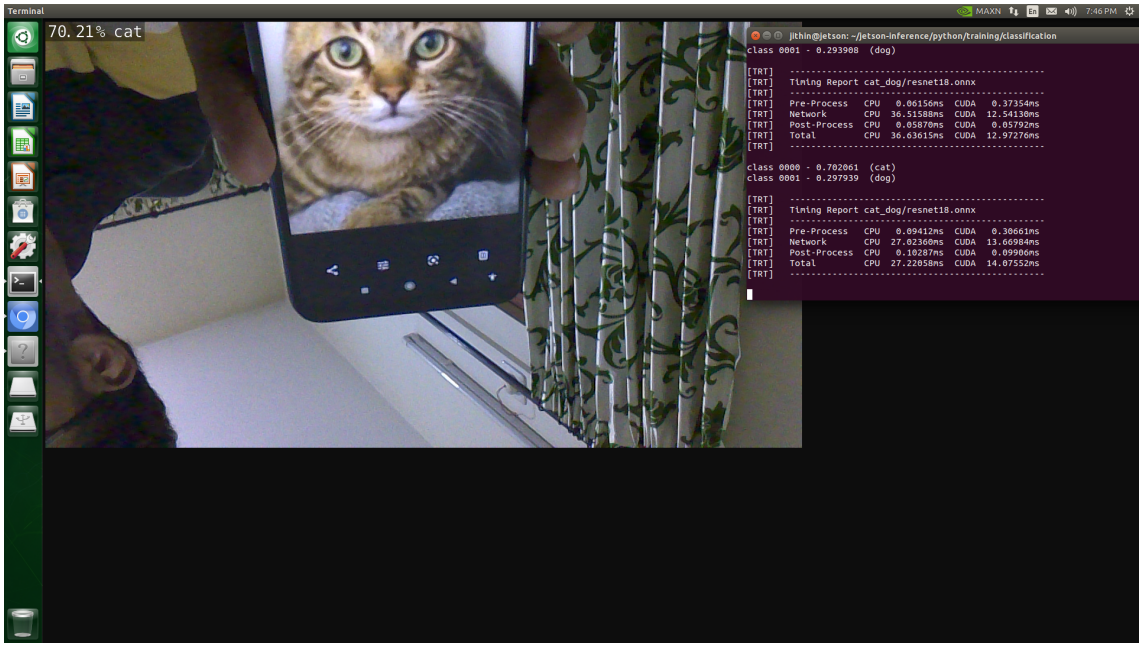

Figure 4: User Interface

7 Detailed Design

7.1 Module Detailed Design

7.1.1 Environment

1. On Startup, Download Miniconda 3 and install it in the system. Create a new isolated environment "ml".

2. Activate the environment and install the necessary libraries and packages
   Your environment is setup for working and training.

7.1.2 Training

1. Run the PyTorch model and enter the run command for the training script.

2. The training script will contain image pre-processing libraries, the number of times the training data will be analyzed (epochs), and the number of classes we are planning to have in the model.

3. Next the formed training model will have to be saved. It will be saved as another PyTorch model, we will convert this model to ONNX format to be used in the jetson nano board.

7.1.3 Running the model

1. Enter the execute command for the model and provide the path to data set with it.

2. The model will be executed and the user interface will pop up.