Design Document for HIVENet

Enrique Ferndandez, Alex Garcia, William Wodrich, Andrew Davis

CS 461 Oregon State University

Fall 2018

Abstract

Convolutional neural networks require large amounts of computational power, data, and training time to accurately identify images and the subjects therein. Implementations tend to be static and rely on single machines, as it would be cumbersome to transmit all the relevant data over a network. This project aims to prototype a new process by using a distributed design that will integrate various devices with a communal objective. A single device will train an artificial neural network from its subset of data. After training, to reduce the size of the data transmitted, only the coefficients of the artificial neural network are broadcast to other devices. This heightened level of performance reaches across the entire network without dramatic amounts of data traffic.

Contents

1 Introduction

1.1 Scope

Our goal is to find an efficient way to use information from trained inference engines on disjoint edge devices, to train other devices on the shared network. This standard describes the design process which will take place throughout the next several months. It includes process descriptions covering edge device communication, data gathering processes, data transfer, user interaction, and other vital information pertaining to a trained facial recognition network.

1.2 Purpose

The purpose of this Software Specification Document (SDD) is to outline the design process that will be implemented in:

- configuring/setting up facial recognition NN on edge devices (OpenFace),
- establishing the means of communication between edge devices, and
- establishing how to transfer the trained NN, or components of the NN between edge devices.

1.3 Intended Audience

Our project aims to demonstrate HIVENet to our issuing organization, Lonnie Mandigo, Kirsten Winters and Kevin McGrath, and participants at Oregon State University's 2019 Engineering Exposition.

1.4 Issuing Organization

Our project comes from Lonnie Mandigo, of Hewlett-Packard, as well as Oregon State University.

1.5 Conformance

This document shall conform to standards laid out by the issuing authority, should sections **3** and **4** outline the ideas of a learning NN given separate edge devices working together.

1.6 Reference Material

- Eyal Reingold, "Artificial Neural Networks Technology", *University of Toronto Mississauga*, [Online]. Available: *http://www.psych.utoronto.ca/users/reingold/courses/ai/cache/neural_ToC.html*. [Accessed October 31 2018]
- Marcel van Gerven, "Artificial Neural Networks as models of Neural Information Processing", *Frontiers in Computational Neuroscience*, [Online]. Available: *https://www.frontiersin.org/research-topics/4817/artificial-neural-networks-as-models-of-neural-information-processing#authors*. [Accessed October 31 2018]
- "Definition - Data Packet", *techopedia*, [Online]. Available: *https://www.techopedia.com/definition/6751/data-packet*. [Accessed November 1 2018]
- "What is a Neural Network", *DeepAI.org*, [Online]. Available: *https://deepai.org/machine-learning-glossary-and-terms/neural-network* [Accessed November 1 2018]

1.7 Key Terms and Acronyms

- **HIVENet:** ”Hierarchical Information Variant Exchanging Network.” Used throughout the document to refer to the project application.
- **Edge-Device:** A computer wirelessly connected to other edge devices. Equipped with a camera to receive video input.
- **Edge-Process:** The tandem structure of the artificial neural network and inference engine housed on an edge device.
- **Packet:** A unit of data packaged to be transmitted over a network. Edge devices send packets requesting routing data along with their bid value, as well as packets containing neural network coefficients and the unique key for which device will begin training the neural network.
- **Neural Network:** ”NN.” A computational framework that employs several learning algorithms to process complex data input. This framework is generally implemented as directed weighted graphs of artificial neurons.
- **Artificial Neurons:** The most basic component of a neural network. They mirror neurons in the brain, acting as aids in the overall decision process of NNs
- **Coefficients/Weights:** The values inside of a NN that define and modify its behavior. They are associated with the artificial neurons and edges that connect them.
- **Training:** The act of providing a NN with a dataset to operate on and apply its learning algorithms to.
- **Inference Engine:** A component of the edge-process that applies logical functions to a knowledge base to deduce information. In our implementation the knowledge base is provided by the weights of the neural network.
- **Fitness Level:** An enumeration of an edge-process’ competency. As the NNs train themselves on a dataset, it will modify itself to increase accuracy, raising the fitness level.
- **Dataset:** A collection of information for a neural network to process,labeled such that the NN can grade its performance. For this project datasets will be comprised of images of individuals, labeled with the measurements of their facial features.
- **Ad Hoc Network:** A decentralized wired network that does not rely on communication via a base station, but rather connects edge devices to each other.
- **ssh:** Linux command that enables a secure connection to a server from a remote device.
- **sftp:** Secure File Transfer Protocol. Allows for secure data transfer between connected devices.

2 System Overview

The HIVEnet facial recognition project is a proof of concept for a form of sharing the utility of machine learning across a number of devices. Current implementations of machine learning emphasize static designs that take advantage of one computer and large sets of data. While powerful, the utility of these projects is often limited to laboratory environments. The purpose of HIVEnet is to explore a new option of a distributed and portable network of machines cooperating to train a communal network with multiple varied data sets.

To create a portable network, HIVEnet Edge-Devices connect through a dedicated wired ad-hoc network, or a Physical communication line that HIVEnet can dynamically network nodes within range. Once networked together nodes communicate with one another to Train the Neural Network that is shared across all HIVEnet Edge-Devices. With multiple devices in network large amounts of data can be collected and processed for training. Thanks to the eager synchronization system implemented in the HIVEnet architecture, Edge-Devices data is gathered and trained in order to maximize performance and functionality.

Essentially, HIVEnet exists to explore a new form of machine learning and to provide possible design spaces. The design aims to be scalable such that a similar framework can offer utility to a wide variety of projects that want to implement distributed Machine Learning.

3 System Architecture

This section will cover the conceptual model for the HIVEnet project. It will explain the structure of the components and how each piece contributes to the design. Different structures will be decomposed and illustrated on a conceptual level. At the end of the section the design rationale will be provided, explaining some of the decisions that were made when determining the design.

3.1 Architectural Design

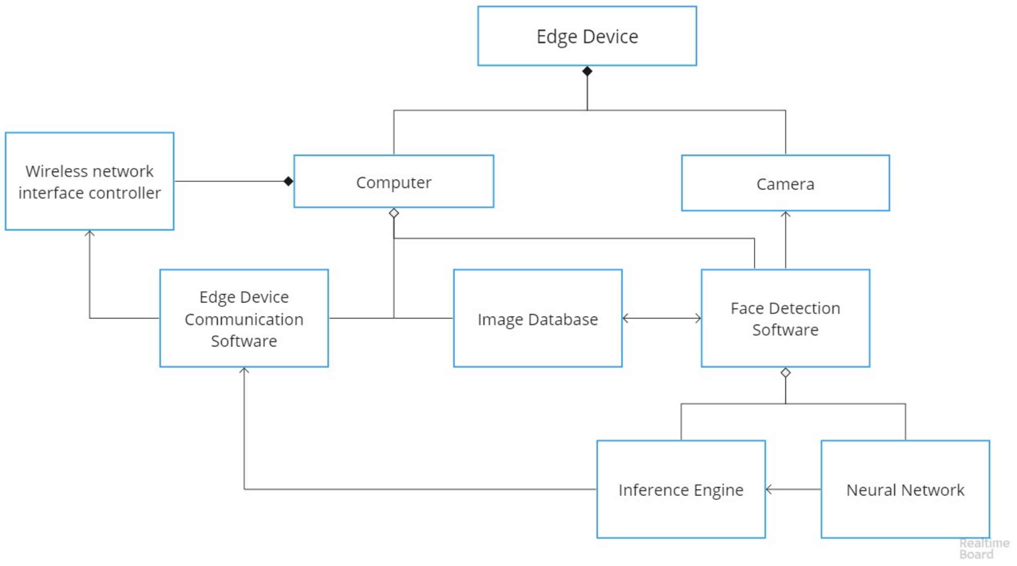
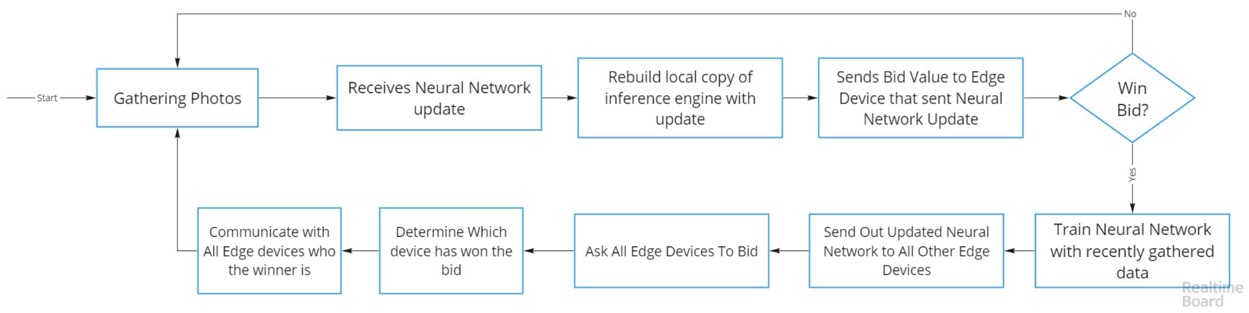


Fig. 1. The hardware components and software components that make up an edge device.

The basic design revolves around the functionality of the Edge-Device (Fig 1). Each node in the HIVEnet must be able to capture image data, recognize faces from that data using an inference engine generated from the communal NN, and train the NN using its local image database, all while maintaining a connection to a dedicated network.

3.2 Node Communication Cycle



Start
Gathering Photos
Receives Neural Network update
Rebuild local copy of inference engine with update
Sends Bid Value to Edge Device that sent Neural Network Update
Win Bid?
No
Yes
Train Neural Network with recently gathered data
Send Out Updated Neural Network to All Other Edge Devices
Ask All Edge Devices To Bid
Determine Which device has won the bid
Communicate with All Edge devices who the winner is
Realtime Board

Fig. 2. A process diagram that describes the neural network sharing, training, and data gathering on an edge device

Each Edge-Device processes image data while not engaged in training the communal neural network. Once The Images are captured, and the Edge-Device completes its training on its localized NN, it utilizes the HIVEnet network to share the new values of the NN coefficients with each online Edge-Device. (Fig 2).
Once the Edge-Device sends its weights generated by the new dataset, the remaining Edge-Devices receive the new weights and begin to update its localized NN. (lower branch of Fig 2). The System then begins to recognize faces that pass by the camera whilst continuing to gather image data and applying the facial recognition via the inference engine rebuilt by the neural network update.

3.3 Facial Recognition Functionality

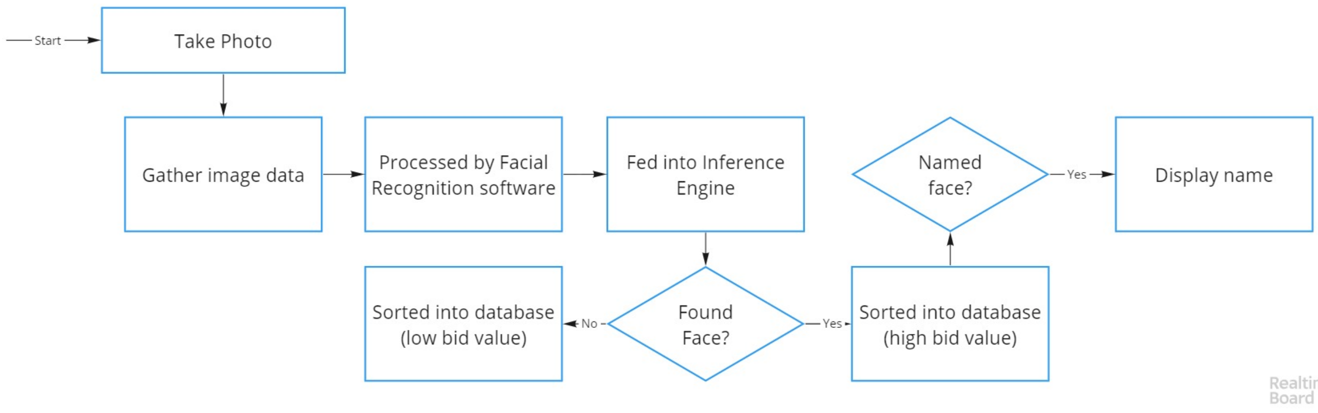

Start
Take Photo
Gather image data
Processed by Facial Recognition software
Fed into Inference Engine
Named face?
Yes
Display name
Found Face?
No
Yes
Sorted into database (low bid value)
Sorted into database (high bid value)
Realtime Board

Fig. 3. A process diagram that describes the states of an edge device processor when running face detection of an image

As Edge-Devices gather image data from their camera they do not passively record it. Instead, they continuously run facial recognition software enabled by the inference engine on their processor that captures a single image, processes the image throught its inference engine, then finally modifying the image with the name of the recognied person iff the person can be successfully identified with a 85 percent confidence interval. The inference engine is generated by the communal NN and updated after each training cycle. It operates on image data from the camera to determine whether the image contains a face or not and attempts to identify the faces present. The inference engine sorts image frames into categories of low or high value based on whether or not they contain faces, so that the NN is trained by devices with valuable data (Fig 3).

3.4 Interconnected Network

To maintain a proper level of inter connectivity all edge devices that are a part of HIVEnet are linked through a reactive ad-hoc wireless network, or a physically connected network (Fig 4). This offers a level of adaptability to the design that permits it to be scalable and resistant to loss of connectivity. As a backup, connecting the edge

3.5 Design Rationale

It's paramount to the project that the design is portable and modular, without overhead costs for administration and oversight. HIVEnet exists to demonstrate a form of deploying machine learning functionality in dynamic environments

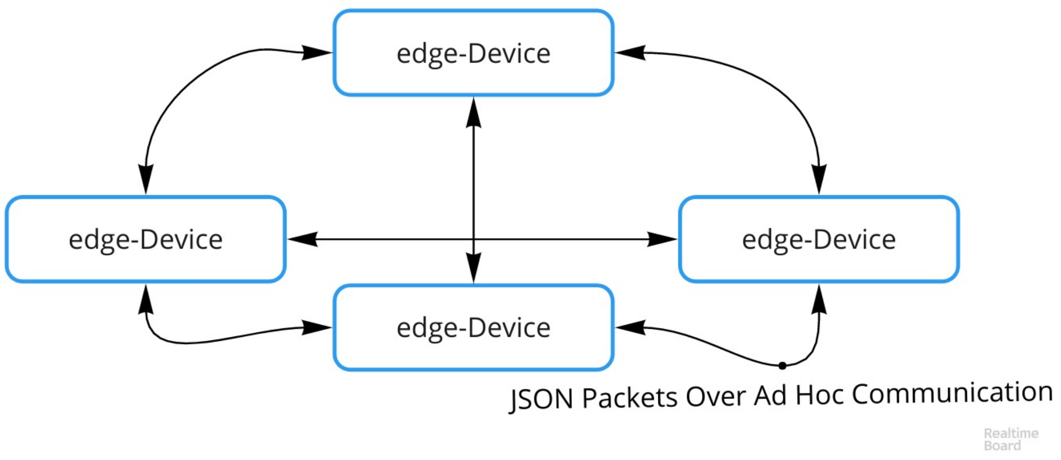


Fig. 4. A model of HIVENet devices interconnected

while still retaining efficiency and efficacy. Additionally, creating a design that revolves around many facsimiles of the same device allows for rapid prototyping for the design team, and eases the development process for iterations of the design.

A reactive ad-hoc network permits the cluster of Edge-Devices to remain interconnected without continuous network usage, as each device's main functions can be performed without communication to other nodes. When the devices do communicate the network is designed to transmit data efficiently. Each node receives a copy of the NN coefficients, a compressed form of the neural network data to avoid sending the data utilized in training. After receiving the data each device can then generate a new inference engine based on the data to update its functionality.

4 HIVENet Design in Perspective

4.1 Design Stakeholders and their Concerns

Our stakeholders include those invested in the project, as well as those providing insight into it. These members include:

- Lonnie Mandigo from Hewlett-Packard, and
- Kirsten Winters, Kevin McGrath, and Benhman Saeedi from Oregon State University.

Taking into account the objectives of the design and the development team, the design should be modular such that the team can rapidly prototype the design before moving to expand the project to its target scope. The project should be implemented in a language that can take advantage of existing platforms such as TensorFlow.
For the concerns of the client, the system implemented must be portable, data efficient and decentralized. The aim of the project is to create a framework which will allow for more effective implementations of neural networks in enabled devices to be developed and deployed.

4.2 Design Viewpoints

This section illustrates some of the viewpoints that are applicable to the HIVENet design. Relevant viewpoints are context, structure, dependency, and interaction.

4.2.1 Context viewpoint

Large amount of users will be interacting with the HIVENet application. Each user should be able to be uniquely identified on any edge device at 85 percent accuracy. Because HIVENet only exchanges the coefficients of the shared neural network, the expertise of an NN trained on large amounts of data can be shared efficiently across a wireless network.

4.2.2 Structure viewpoint

In order to make the prototyping process rapid and efficient, and to ensure that the design is properly scalable, each edge device should be identical but independent such that they can be dynamically added or removed from a network without disrupting the function of other devices. Each device can contribute to the network without the need of a central network authority.

4.2.3 Dependency viewpoint

Edge devices are designed to be as independent as possible so that they will not be affected if other devices are added or removed from the network. The network as a whole should remain functional and efficient in a dynamic environment without a user or developer intervening to monitor or maintain its function.

4.2.4 Interaction viewpoint

The HIVENet is at its core a framework for multiple devices to collaborate in the training of an artificial neural network, requiring them to communicate and remain sin constant connection with one another. The design must be scalable such that the entire network can efficiently share a single NN without supervision or centralized networking.

5 Component Design

HIVENet is made up of edge devices, each running similar processes in parallel. Edge-Device weights aew shared between all devices whilst storing datasets locally, thus requiring a network to be established. The components involved, from the device itself, to the data set used in each device, is described in the section.

5.1 Edge Device Network

5.1.1 Overview

The Edge Device Network is a collection of edge devices, each edge device being able to communicate with all other edge devices. The Edge Device Network is decentralized and connected via LAN, comprised entirely of edge nodes.

5.1.2 Structure

The network's structure is inherent to its definition. This network connects edge devices in a fully connected peer-to-peer topology.

5.1.3 Design Rational

There are several reasons the HIVENet project is implementing a network of edge devices. Firstly, this project has an emphasis on partitioning the training of a neural network across multiple devices. By the neural network training occurring on multiple smaller devices as opposed to a computationally powerful single device as is convention, less training information is required to be sent across the network as explained in Edge Device Communication. Additionally, the facilitation of edge devices to gather neural network training data and train neural networks, the possibility of training a neural network in parallel across multiple devices arises. If possible, this would allow for more effective training of a neural network on smaller devices.

Another reason to implement a network of edge devices is to allow for multiple devices to be able to implement the same copy of a neural inference engine which is being continually improved.

5.2 Edge Device

The atomic unit of the design, edge nodes are designed to gather a data-set to improve the performance of a partially trained NN and share it across a network of local edge devices.

5.2.1 Structural Viewpoint

The device is composite of a computer and video camera. The computer computer can access to the Edge Device Network. Additionally, the computer can be accessed from the edge device terminal. The computer houses a local image training set as well as a program for training a face recognition neural network and a program for implementing face recognition with an inference engine. The edge device has access to a display which it uses to output the result of the facial recognition inference engine.

5.3 Composition Viewpoint

An edge device is a part of an edge device network which connects each edge device to all other edge devices.

5.3.1 Context Viewpoint

To a user of the edge device, they will be able to interact with the device in two ways. One way is to append the neural network training set, another is to use the neural inference engine on the device to recognize a face in an image. Appending the neural network data set is accomplished by taking a video recording of their face and recording their name in the Edge-Device they are interacting with. A user can use the facial recognition inference engine by walking around the other Edge-Devices and where the devices will be in face detection mode and be taking a photo of their face. The device will then process that photo in the neural inference engine which will attempt to identify a face in the image. The result of the face detection will be viewable from an external display.

5.3.2 Design Rationale

Each edge device needs the minimal functionality to gather image data and store it as a set for the purpose of training a partially trained neural network, as well as the ability to dynamically network with other edge devices in a wired ad hoc network. This allows a HIVENet operate with any number of nodes in a portable and scalable manner.

5.4 Edge Device Processor

The computing engine of an edge device, the processor threads between training a local neural network and gathering and sorting image data based on an on-board inference engine. Additionally, the edge device can be used to implement the inference engine to perform face detection in an image. The face detection

5.4.1 Composition Viewpoint

The processor is the brain of each edge device, allowing it to train the stored NN, apply facial recognition to the camera feed, and communicate with other devices.

5.4.2 State Dynamics Viewpoint

The Processor has two Primary states: Registering/training, and recognizing. Each HIVEnet Edge-Device is focuses on training its localized Neural network and with its unique datasets its gathered from the connected camera. Once the training is complete the device queries the network for routing its weights and prepares the itself for Recognizing. Registering and training is threated every time a new user is being appended to the dataset of the Edge-Device.
While in the data gathering state the processor applies the facial recognition software utilizing the neural network to analyze, sort, and react to image input from the integrated camera.

5.4.3 Design Rationale

Because the HIVENet shares a communal NN, there is a distinct amount of time for each node where it will not be focused on training a neural network. During this time the processor applies the functionality of its local image of the neural network (updated across the network whenever the training node completes its process) to recognize faces captured by the integrated camera. Additional functionality may be included based on facial identification, where certain actions are prescribed for when the device recognizes an individual.

5.5 Edge Device Inference engine

The inference engine of each edge device enables the facial recognition software to identify features of images captured by the camera. This allows the device to sort image data as ”interesting” or ”uninteresting” based on the features found, or in other words how clearly the image represented is a face. The inference engine will also be able to identify specific faces and can then execute additional processes based on the subject's identity.The foundation of the inference engine is based on the coefficients of the local neural network dataset. Each inference engine on each Edge-Device is generated through the collection of the weights from each Edge-Device and partitioned by the number of devices on the HIVEnet network. Each partition is rebuilt after each training cycle that is triggered by a new user to reflect the learning nature of HIVENet.

5.5.1 Context Viewpoint

Whereas the neural network of a HIVENet can be thought of as a mobile entity shared across the collective of devices, the inference engine is a reflection of the NN, and drives the functionality of the facial recognition software on each device.

5.5.2 Composition Viewpoint

Inference engines are stored on the processors of each edge device. They are built using the values of the local HIVENet NN dataset and are integrated with the facial recognition software for the camera.

5.5.3 Design Rationale

The facial recognition application of each device relies on the inference engine to identify facial features from image data the camera captures. The engine uses the learning from the HIVENet NN to improve its performance.

The functional element of the design, each HIVENet Edge-Device has a shared neural network which enables the facial recognition software loaded onto each edge device via the local inference engine.

5.5.4 Composition Viewpoint

Each edge device has a local image of the shared HIVENet NN, the image is updated across the network after every training cycle so that the aggregate network maintains the same level of performance on all devices.

5.5.5 Design Rationale

The network is designed to migrate from device to device in order to be exposed to new data, as the network itself -represented by the NN coefficients- is much more portable than the set of data used to train it. After each training cycle the training device determines where the NN should migrate to based on a bidding value determined based off of how useful the data might be for improving the NN. This mobility drives the utility of the design.

5.6 Edge Device Communication

Each edge device will have communication channels that allows for communication between an edge device and all other edge devices. Communication between edge devices is asynchronous. The main purpose of the communication is to transfer data describing the node relationships of a neural network. This data is sufficient for the receiving edge device to be able to replicate the neural network that the transmitting edge device has on it's machine.

5.6.1 Composition Viewpoint

Each edge device has the functionality to handle its own edge device communication. Edge device communication is facilitated on the Edge Device Network, a decentralized network with a fully connected topology.

5.6.2 Structural Viewpoint

Edge device communication will facilitate the sending and receiving of neural network transmission packets.

5.6.3 Design Rational

because HIVENet is a distributed system that consist of several edge devices, we are required to create a method of communication that will allow for information to be sent across all edge devices. using channels that allow for transfer of data between edge devices will allow for transfer of information between the devices.

5.7 Neural Network Transmission Packet

The neural network transmission packet is a JSON packet which possess all necessary information to represent the nodes and the weights that make a neural network.

5.7.1 Composition Viewpoint

A neural network transmission packet will be created by an edge device processor whenever an edge device needs to share a copy of a neural network. A neural network transmission packet can be seen in an edge device network in figure 4

5.7.2 Structure Viewpoint

An neural network transmission packet will posses at the minimum a description of each node in the neural network. The description of a node will include all other nodes the node is connected to by an edge, and the weight of that edge.

5.7.3 Design Rational

The neural network transmission packet will allow edge devices to share neural networks to other edge devices. The functionality of sharing neural networks across edge devices is important to allow edge devices to be able to train the same neural network locally.

5.8 Edge Device Image Set

the edge device image set is a neural network training data set. each data point in this data set is an image of a person's face and the name of the person.

5.8.1 composition viewpoint

The edge device image set is stored on the edge device. It is used by the HIVENet NN for neural net training.

5.8.2 Dependency Viewpoint

This component is exclusive to each individual edge device. Once used to train inference engines, separate data that is not the image set is shared along the network.

5.8.3 Design Rational

The purpose of the edge device image set is to provide data to train the face recognition NN on the edge device. Without the image set, there would be no data for training.

5.9 Edge Device Terminal

An edge device terminal gives access to a shell on the edge device.

5.9.1 Interaction Viewpoint

The terminal will allow a developer's personal machine to access an edge device. This interaction will be facilitated by a peer-to-peer client-server http protocol, meaning that both the edge device and the developers personal machine will need a network connection.

5.9.2 Design Rational

The edge device terminal allows the developers to interact with all the edge devices from their own machine. This simplifies the development of each edge device since they will not each require keyboards, mouses and displays for development. Additionally, with a terminal, some of the development on each edge device can be automated ensuring consistency across all edge devices.

5.10 Display

A display will be used to display the result of the face detection from an inference engine on an edge device. The display will be a self contained machine and not require aid from an edge device to operate.

5.10.1 Interaction Viewpoint

The display will be on a machine that will be accessible to each edge device via TCP/IP. Each edge device can connect to the Display to transmit a resulting image file from a face detection inference engine employment.

5.10.2 Design Rational

A display is required so that each edge device is able to demonstrate the effectiveness of the inference engine on its machine. This will be useful for demonstration purposes with the audience as well as for testing purposes during development.

6 UI Design

The system will consist of two main components that users will interact with: edge devices, and a central display. Edge devices will require minimal interaction to thread between a training mode, and a working mode. The central display will show the results of the user recorded, displaying a fully recognized face with their associated name. This portion will consist of a terminal.

6.1 Edge Device Physical UI

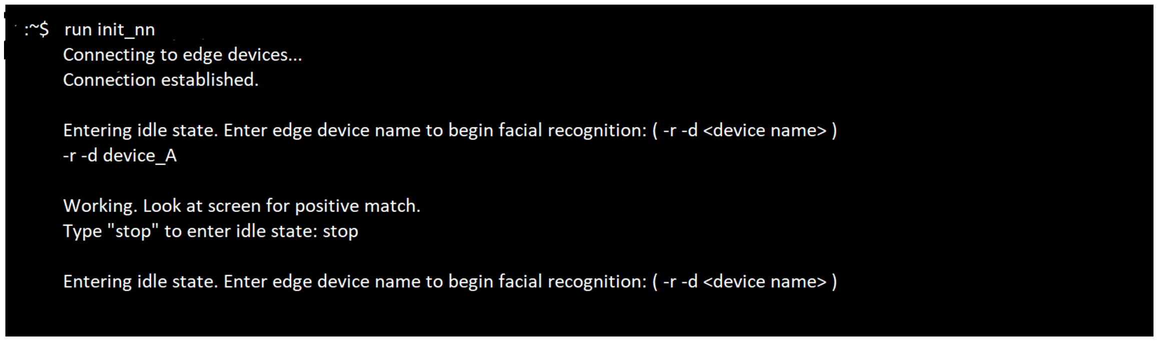
Each Edge-Device requires the user to indicate when to threat the training mode (video recording for data collection, used to train the NN). When complete the Edge-Device will automatically switch to recognition mode, in which the user presents their face to the camera and is identified by HIVENet. This will be done simply with a submittion button of the name of the user that triggers the training on that device.

6.2 Edge Device Graphic User Interface

This UI will be limited to the display unit of the network. The Graphic User Interface will be have limited inputs to:

- select the user to recognize (recognizing phase),
- display the most recent pictures from recognized users,
- register user with web form followed by image capturing of the new users face.
- enter an idle mode to wait to enter recognizing phase again.

This process is shown in the following image:



```
:~$  run init_nn
     Connecting to edge devices...
     Connection established.

     Entering idle state. Enter edge device name to begin facial recognition: ( -r -d <device name> )
     -r -d device_A

     Working. Look at screen for positive match.
     Type "stop" to enter idle state: stop

     Entering idle state. Enter edge device name to begin facial recognition: ( -r -d <device name> )
```

Fig. 5. Mockup of terminal view for display unit

Note, this terminal view outlines the order of events that are described above, but may not accurately represent the explicit command syntax.

When in the recognizing phase, the persons face will appear with a green box around it should the system recognize them correctly, as shown below.

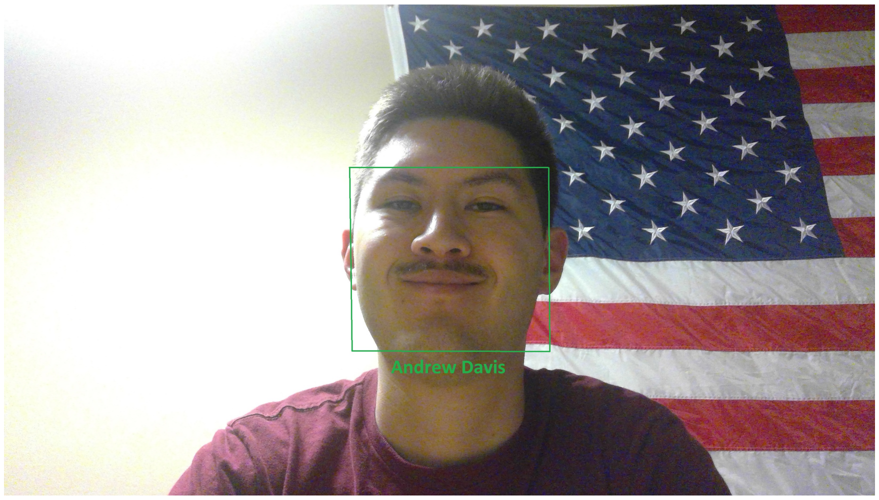


Fig. 6. Positive match on user