2022 Synopsys ARC 盃 AIoT 設計應用競賽 決賽作品

非接觸式控制面板

Contactless Control Panel

指導教授： 張錫嘉 教授

隊員：陳冠瑋、曹家輔、李家毓、鄭紹文

隊名：綠洲熊與他們的窩

July 22th, 2022

Agenda

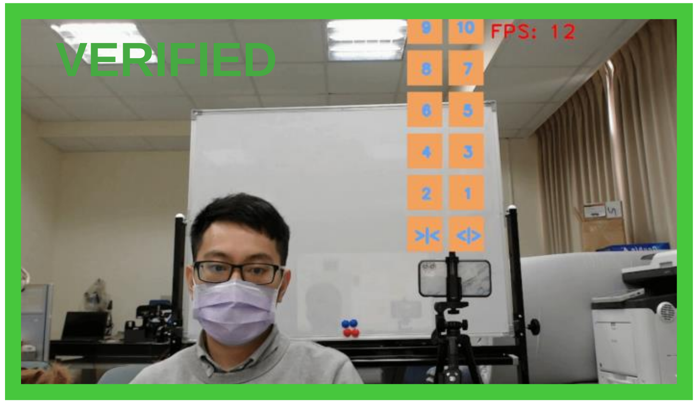
- Introduction
- Difficulties & Innovation
- Design & Implementation
- Result & Demo
- Q&A

Agenda

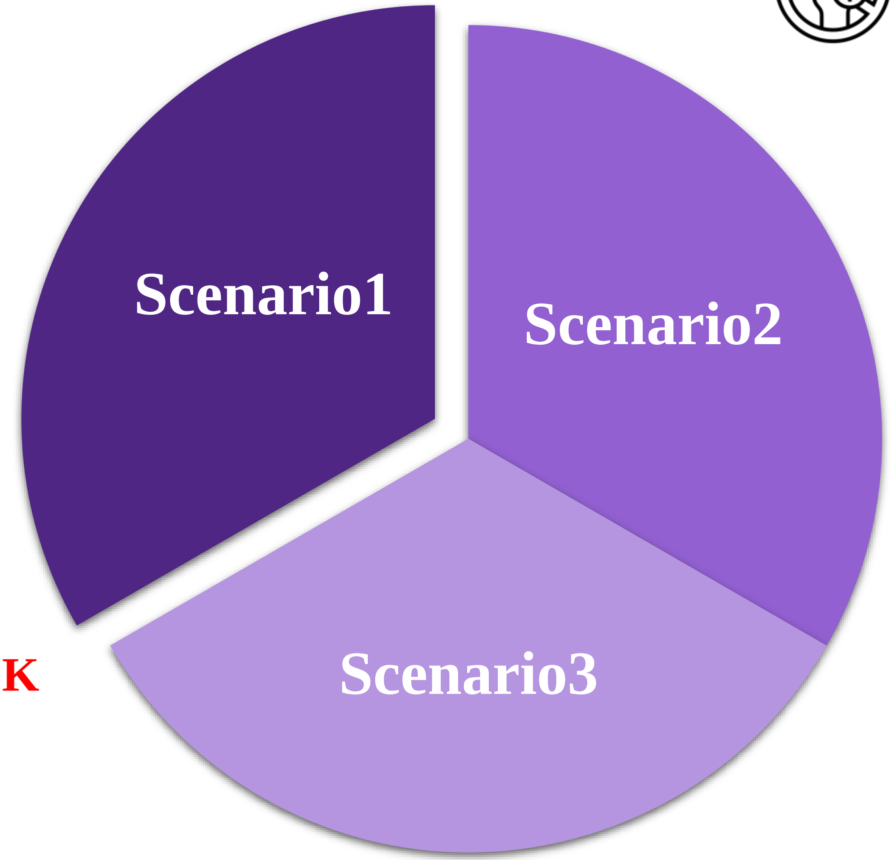
- **Introduction**
- Difficulties & Innovation
- Design & Implementation
- Result & Demo
- Q&A

Introduction

Virtual elevator button panel & Visitor verification



- **Face verification**
  - **Hardware**: **ARC EM9D AIoT DK**
  - **Model**: **Siamese Network**
- **Virtual button panel**
  - **Hardware:** NVIDIA Jetson Nano



Contactless ticket machine screen

Smart door lock

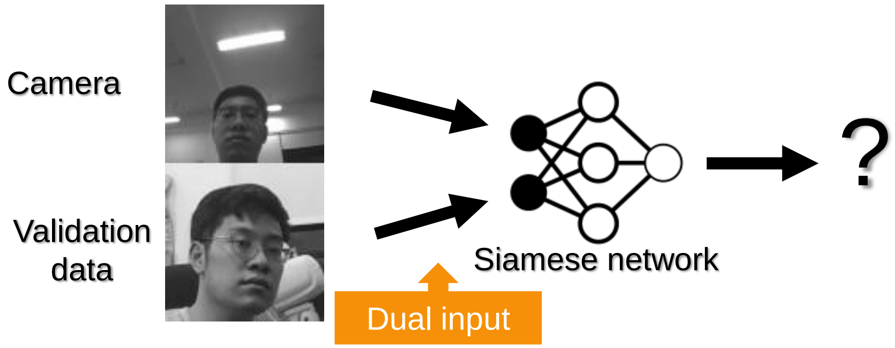
Agenda

- Introduction
- **Difficulties & Innovation**
- Design & Implementation
- Result & Demo
- Q&A

© 2022 Synopsys, Inc.

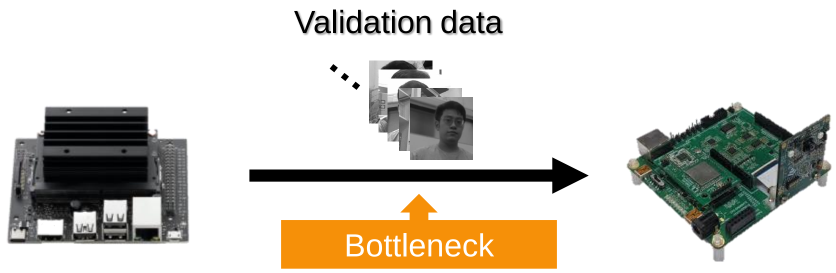
Difficulties - Face Verification (ARC EM9D)

- **Memory size**
  - Available memory size < 2MB
  - Auto-Encoder: Dimension Reduction
  - **Post training quantization: fp32 → int8** ✓



Input images pairs for Siamese network

- **Input images pairs for the Siamese network**
  - Siamese network needs to compare two images at the same time



Transmission time

- **Transmission time**
  - Validation data transmission
  - **Save images for comparison in the system memory** ✓

- **MLI Library does not support some Ops**

sigmoid ✗
tanh ✗ → logistic ✓

abs (int8) ✗ → mul (int8) ✓

Innovation - Face Verification (ARC EM9D)

- **Siamese network architecture on TinyML**
  - Discriminate the similarity between two different input
  - Small number of samples are needed

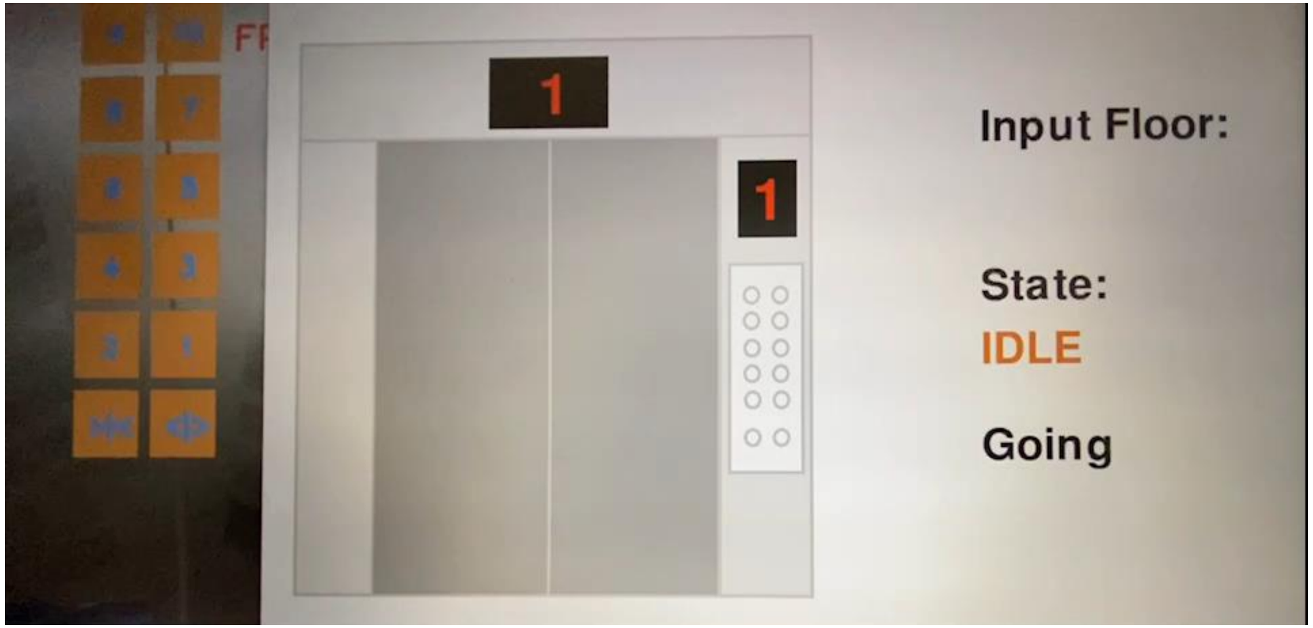
- **Small face verification model**
  - **Low power:** Used in unreliable power supply scenarios
  - **Less computing & memory resources:** Faster inference, faster deployment

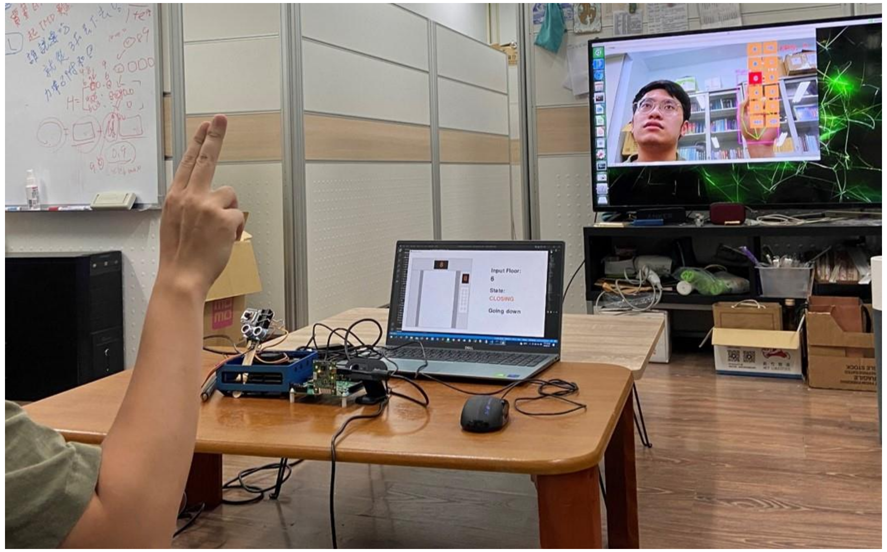
- **Input dimension reduction**
  - Auto-Encoder: Reduce the dimension of the input features
  - Solve the bottleneck of inference time
  - Deploy a more efficient model

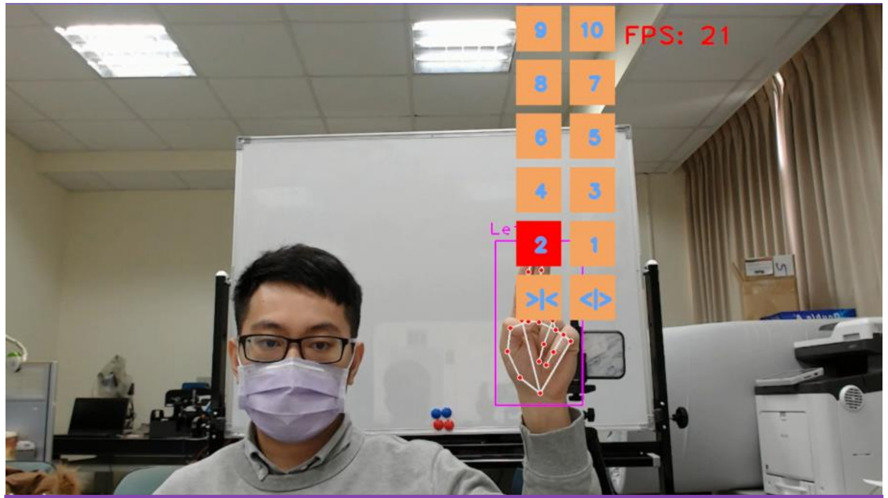
© 2022 Synopsys, Inc.

Innovation - Virtual Button Panel (NVIDIA Jetson Nano)

- **User - Friendly**
  - Click the virtual panel through the gesture
  - **Index Finger**: Select the virtual button
  - **Index Finger + Middle Finger**: Click the virtual button







Click the virtual button

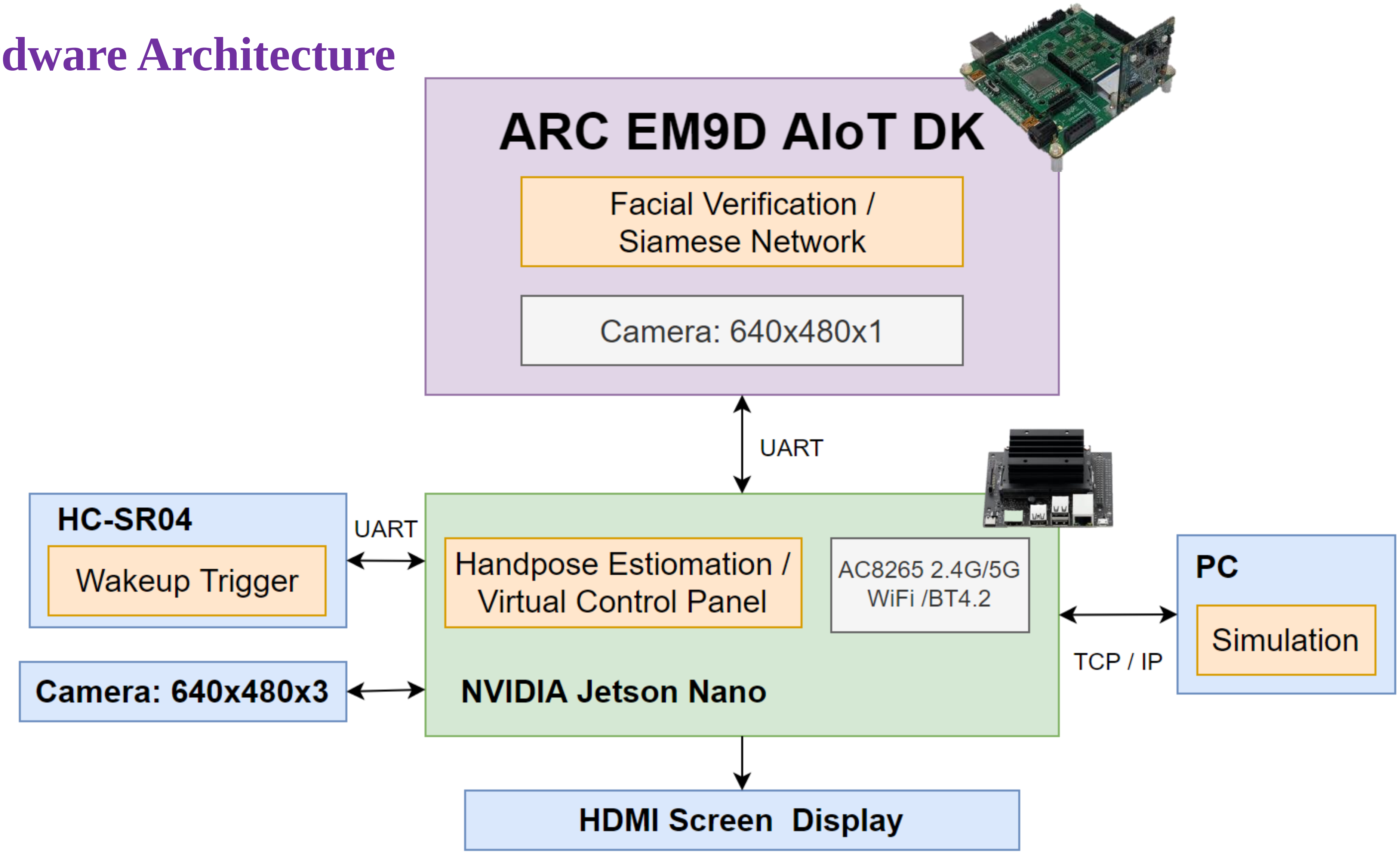
© 2022 Synopsys, Inc.

Agenda

- Introduction
- Difficulties & Innovation
- **Design & Implementation**
- Result & Demo
- Q&A

Design & Implementation

■ Hardware Architecture



ARC EM9D AIoT DK

Facial Verification / Siamese Network

Camera: 640x480x1

UART

HC-SR04

Wakeup Trigger

UART

Handpose Estiomation / Virtual Control Panel

AC8265 2.4G/5G WiFi /BT4.2

NVIDIA Jetson Nano

Camera: 640x480x3

PC

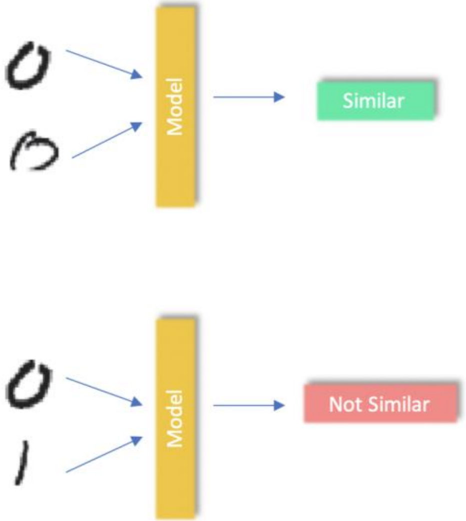
Simulation

TCP / IP

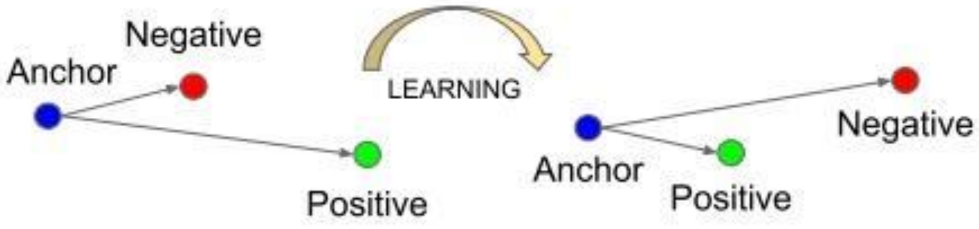
HDMI Screen Display

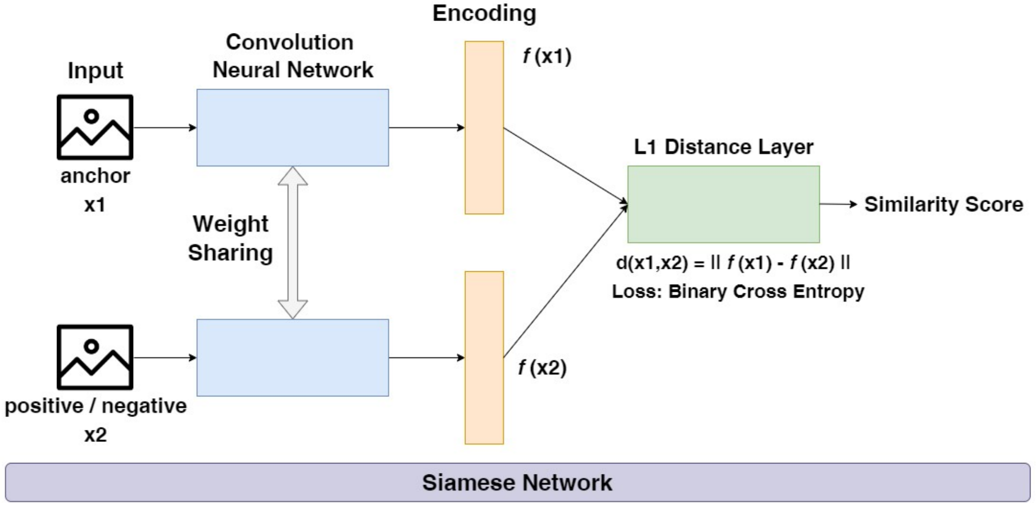
Design & Implementation

- **Siamese Network**
  - Inspired by “Siamese twins”
  - Classify if the two inputs are the same or different
  - Take two different inputs passed through two same subnetworks
  - **Convert the classification problem to a similarity problem**



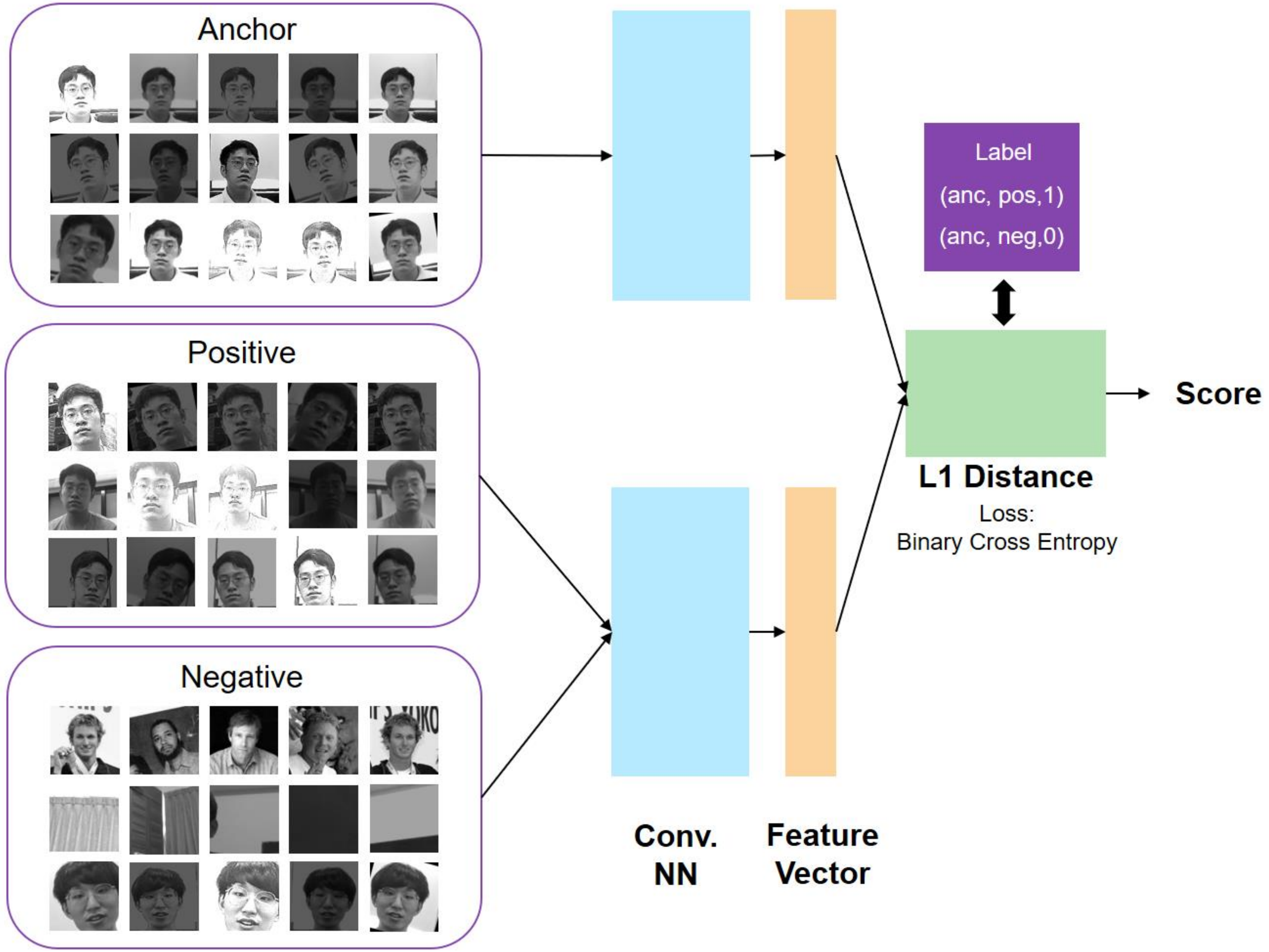
- **Why choose “Siamese Network”?**
  - Humans exhibit a strong ability to recognize new patterns
  - A **few-shot learning** model





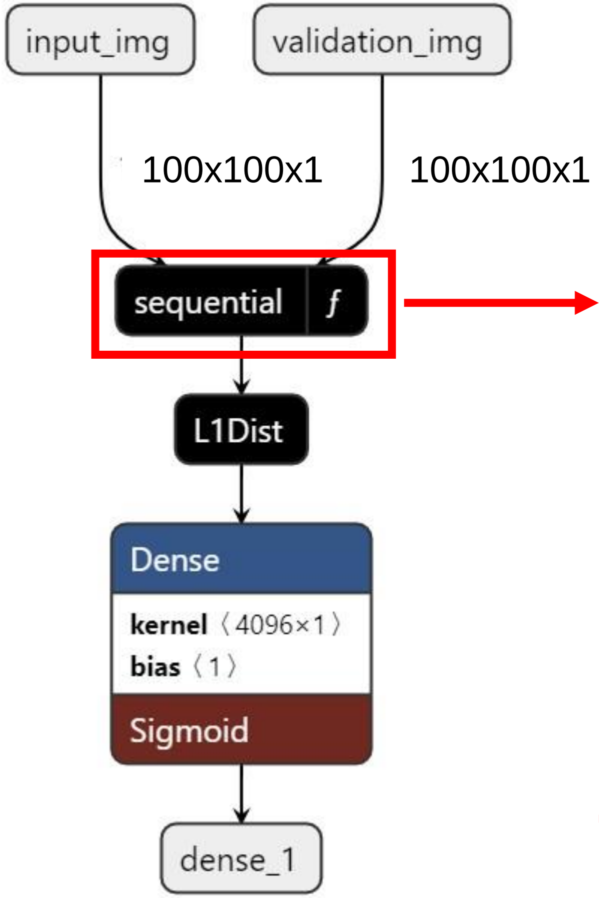
Design & Implementation

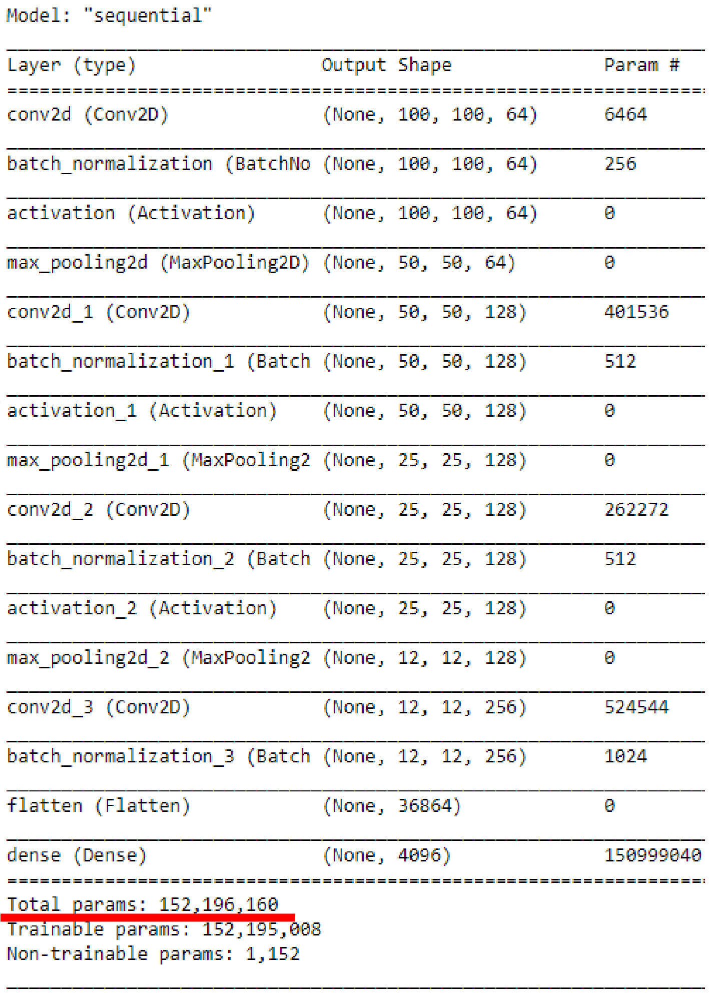
- **Data preprocessing**
  - **Resize: 100x100x1**
  - **Data Augmentation**
  - **Dataset**
    - Anchor: 600
    - Positive: 600
    - Negative: 600



Design & Implementation – Original Siamese Network

■ Inference on PC *(Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz)*



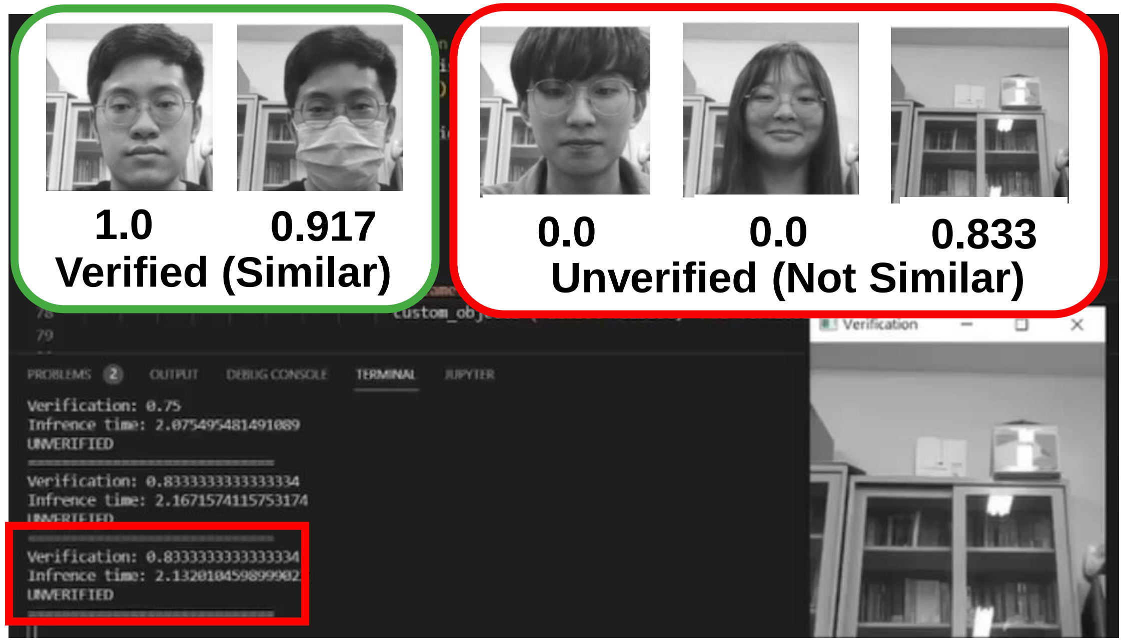


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 100, 100, 64)      6464
_________________________________________________________________
batch_normalization (BatchNo (None, 100, 100, 64)      256
_________________________________________________________________
activation (Activation)      (None, 100, 100, 64)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 50, 50, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 50, 50, 128)       401536
_________________________________________________________________
batch_normalization_1 (Batch (None, 50, 50, 128)       512
_________________________________________________________________
activation_1 (Activation)    (None, 50, 50, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 25, 25, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 25, 25, 128)       262272
_________________________________________________________________
batch_normalization_2 (Batch (None, 25, 25, 128)       512
_________________________________________________________________
activation_2 (Activation)    (None, 25, 25, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 256)       524544
_________________________________________________________________
batch_normalization_3 (Batch (None, 12, 12, 256)       1024
_________________________________________________________________
flatten (Flatten)            (None, 36864)             0
_________________________________________________________________
dense (Dense)                (None, 4096)              150999040
=================================================================
Total params: 152,196,160
Trainable params: 152,195,008
Non-trainable params: 1,152
_________________________________________________________________
```

Data for comparison (10 imgs)

■ Result

- Float32
- **Total parameters: 152.2 M**
- **Accuracy:** 0.967
- **Avg. inference time:** 2 s



1.0 0.917
Verified (Similar)

0.0 0.0 0.833
Unverified (Not Similar)

```
Verification: 0.75
Infrence time: 2.075495481491089
UNVERIFIED
========================================
Verification: 0.8333333333333334
Infrence time: 2.1671574115753174
UNVERIFIED
========================================
Verification: 0.8333333333333334
Infrence time: 2.132010459899902
UNVERIFIED
========================================
```

Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." ICML deep learning workshop. Vol. 2. 2015.

© 2022 Synopsys, Inc.

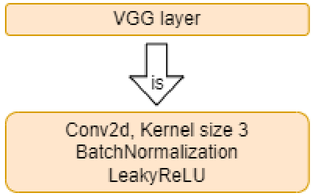
Design & Implementation – AutoEncoder

Dimension Reduction

- Compress input image into latent with Encoder
- Use latent to recover image with Decoder

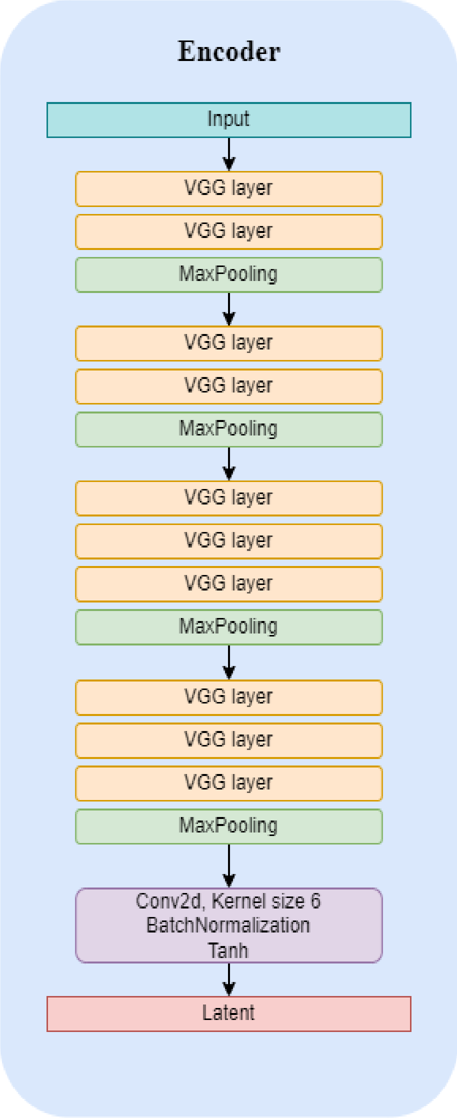
Method

- Use VGG-16 as an architecture baseline

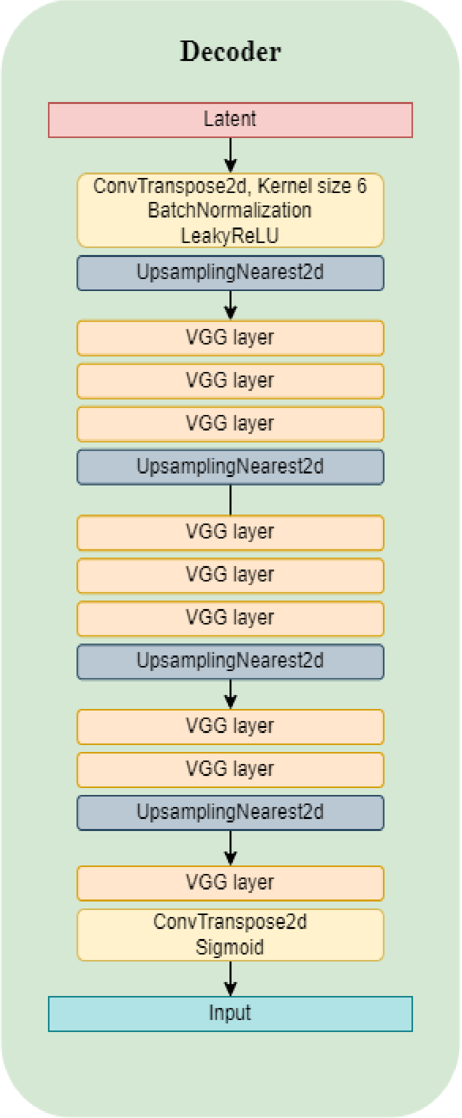

VGG layer

is

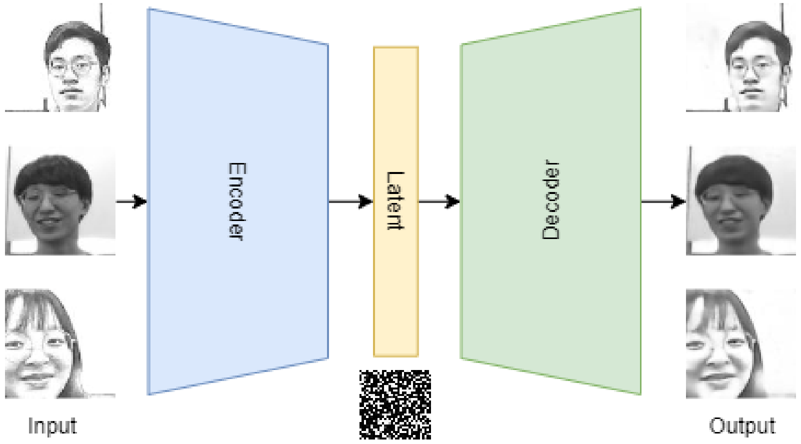
Conv2d, Kernel size 3
BatchNormalization
LeakyReLU


Encoder

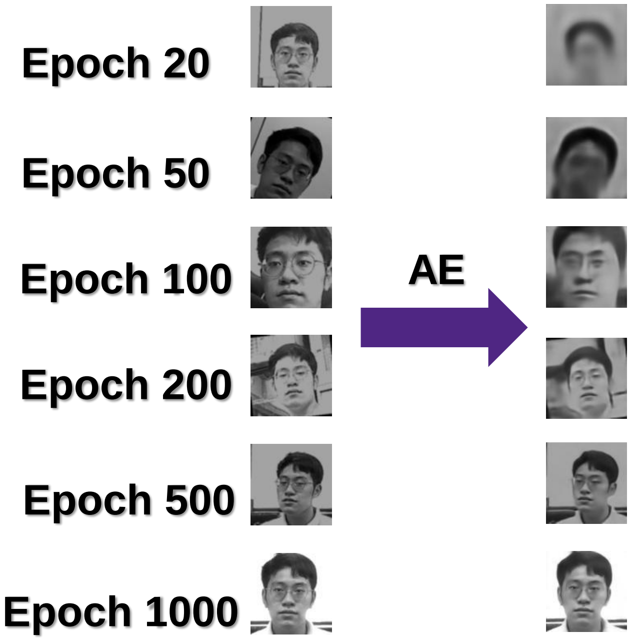
Input
VGG layer
VGG layer
MaxPooling
VGG layer
VGG layer
MaxPooling
VGG layer
VGG layer
VGG layer
MaxPooling
VGG layer
VGG layer
VGG layer
MaxPooling
Conv2d, Kernel size 6
BatchNormalization
Tanh
Latent


Decoder

Latent
ConvTranspose2d, Kernel size 6
BatchNormalization
LeakyReLU
UpsamplingNearest2d
VGG layer
VGG layer
VGG layer
UpsamplingNearest2d
VGG layer
VGG layer
VGG layer
UpsamplingNearest2d
VGG layer
VGG layer
UpsamplingNearest2d
VGG layer
ConvTranspose2d
Sigmoid
Input



AutoEncoder



Ground truth Reconstructed

© 2022 Synopsys, Inc.

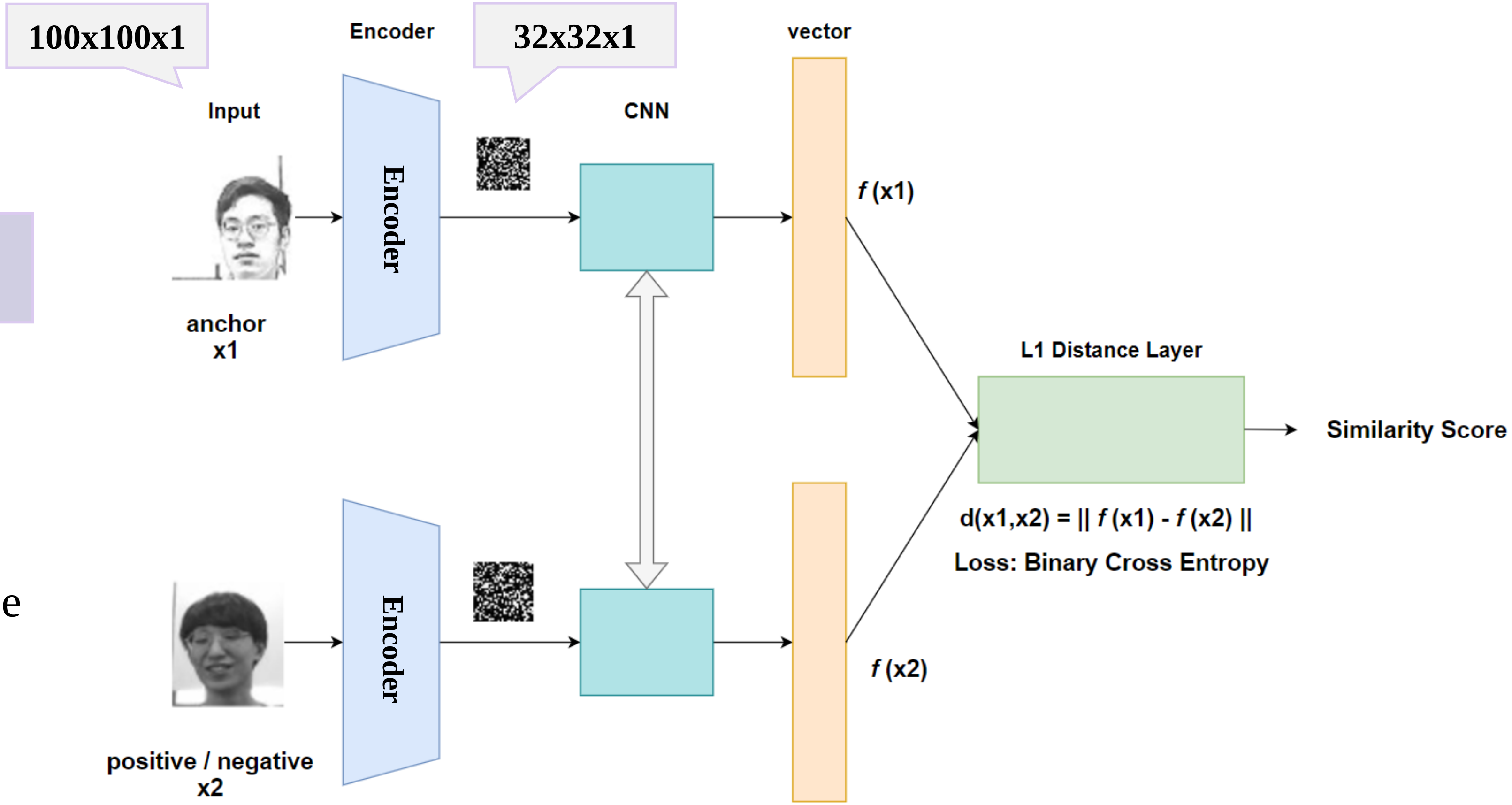
Design & Implementation – AutoEncoder + Siamese Network

■ Inference on PC *(Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz)*

■ Result

- **Float32**
- **Parameters:**
  - Encoder: 26,516,928
  - Siamese Network: 65,675
  - **Total: 26.7 M**
- **Memory Usage:**
  - ≈ 500x ARC EM9D memory size
- **Accuracy:** 0.98
- **Inference time:** 1.2 s

Original 152.2 M

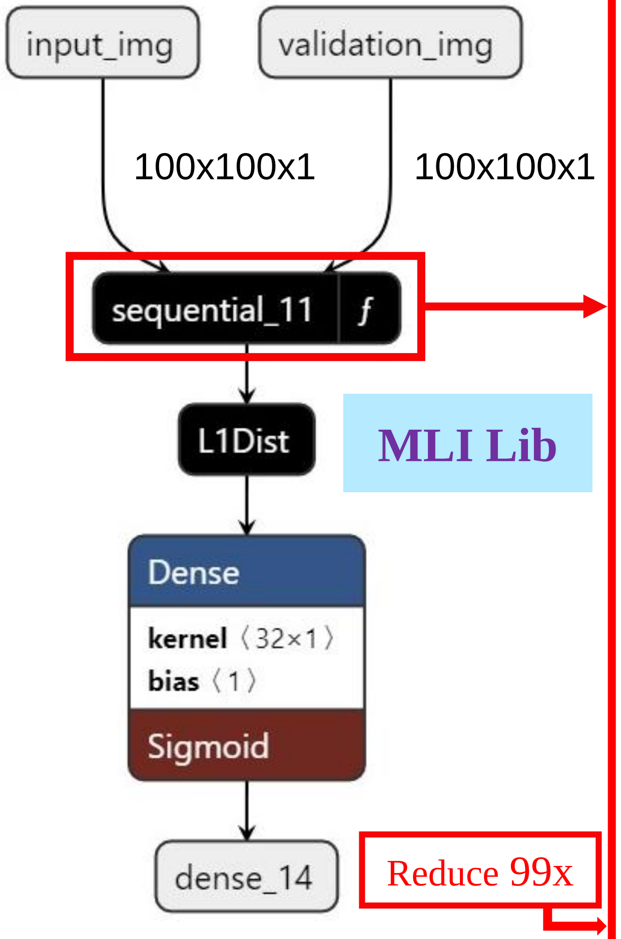


AutoEncoder + Siamese Network

NCTU OASIS Lab

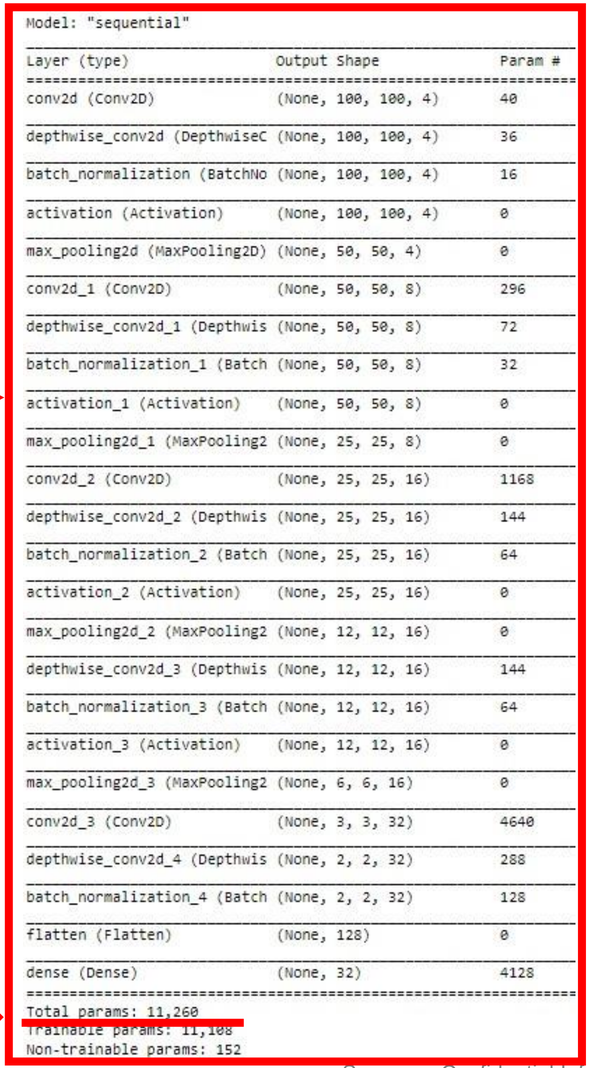
Design & Implementation – Siamese Network (Fine tune)

■ Inference on ARC EM9D



MLI Lib

Reduce 99x



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 100, 100, 4)       40
_________________________________________________________________
depthwise_conv2d (DepthwiseC (None, 100, 100, 4)       36
_________________________________________________________________
batch_normalization (BatchNo (None, 100, 100, 4)       16
_________________________________________________________________
activation (Activation)      (None, 100, 100, 4)       0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 50, 50, 4)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 50, 50, 8)         296
_________________________________________________________________
depthwise_conv2d_1 (Depthwis (None, 50, 50, 8)         72
_________________________________________________________________
batch_normalization_1 (Batch (None, 50, 50, 8)         32
_________________________________________________________________
activation_1 (Activation)    (None, 50, 50, 8)         0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 25, 25, 8)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 25, 25, 16)        1168
_________________________________________________________________
depthwise_conv2d_2 (Depthwis (None, 25, 25, 16)        144
_________________________________________________________________
batch_normalization_2 (Batch (None, 25, 25, 16)        64
_________________________________________________________________
activation_2 (Activation)    (None, 25, 25, 16)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 16)        0
_________________________________________________________________
depthwise_conv2d_3 (Depthwis (None, 12, 12, 16)        144
_________________________________________________________________
batch_normalization_3 (Batch (None, 12, 12, 16)        64
_________________________________________________________________
activation_3 (Activation)    (None, 12, 12, 16)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 16)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 32)          4640
_________________________________________________________________
depthwise_conv2d_4 (Depthwis (None, 2, 2, 32)          288
_________________________________________________________________
batch_normalization_4 (Batch (None, 2, 2, 32)          128
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 32)                4128
=================================================================
Total params: 11,260
Trainable params: 11,108
Non-trainable params: 152
```

■ Description

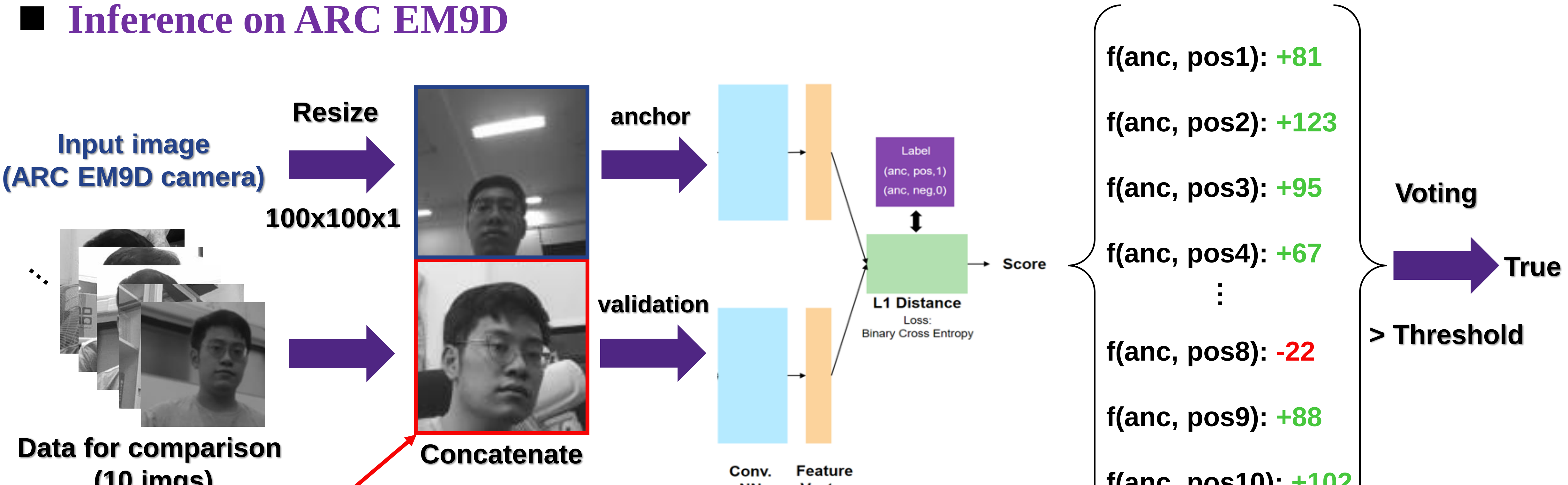
- **Total parameters: 11,260**
- **Int8 quantization**
- **Avg. inference time:** 1 s

■ MLI Library (mircro_op)

- AddConv2D()
- AddDepthwiseConv2D()
- AddMaxPooling2D()
- AddRelu()
- AddLogistic()
- AddFullyConnected()
- AddSub()
- AddMul()

Design & Implementation - Siamese Network (Fine tune)

■ Inference on ARC EM9D



f(anc, pos1): +81

f(anc, pos2): +123

f(anc, pos3): +95

f(anc, pos4): +67

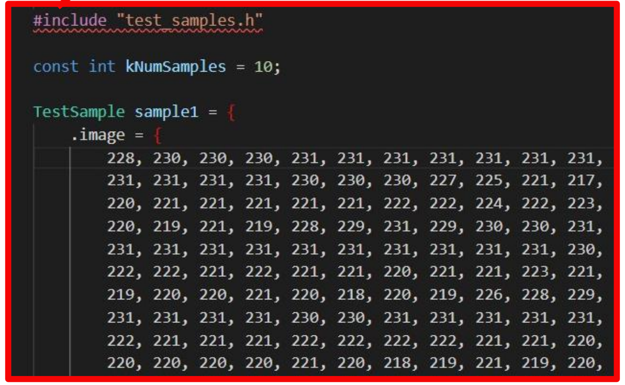
⋮

f(anc, pos8): -22

f(anc, pos9): +88

f(anc, pos10): +102

Verification score



```
#include "test_samples.h"

const int kNumSamples = 10;

TestSample sample1 = {
    .image = {
        228, 230, 230, 230, 231, 231, 231, 231, 231, 231, 231,
        231, 231, 231, 231, 230, 230, 230, 227, 225, 221, 217,
        220, 221, 221, 221, 221, 221, 222, 222, 224, 222, 223,
        220, 219, 221, 219, 228, 229, 231, 229, 230, 230, 231,
        231, 231, 231, 231, 231, 231, 231, 231, 231, 231, 230,
        222, 222, 221, 222, 221, 221, 220, 221, 221, 223, 221,
        219, 220, 220, 221, 220, 218, 220, 219, 226, 228, 229,
        231, 231, 231, 231, 230, 230, 231, 231, 231, 231, 231,
        222, 221, 221, 221, 222, 222, 222, 222, 221, 221, 220,
        220, 220, 220, 220, 221, 220, 218, 219, 221, 219, 220,
```

Write the comparison data on the ARC EM9D (pos samples)

Agenda

- Introduction
- Difficulties & Innovation
- Design & Implementation
- **Result & Demo**
- Q&A

Results & Demo – Siamese Network

	Original Siamese Network	AE + Siamese Network	Fine-tune (float32)	Fine-tune (int8)	Fine-tune (int8)
Processing Platform	*CPU	*CPU	*CPU	*CPU	ARC EM9D
Total Parameter	152.2 M	26.7 M	11,260	11,260	11,260
Can program on ARC EM9D ?	No	No	No	Yes *(98%)*	
Accuracy	0.967	0.98	0.933	0.8	N/A
Inference time	2.1 s	1.5 s	1.0 s	0.8 s	0.1 s

*** use** ***(Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz)***

Agenda

- Introduction
- Difficulties & Innovation
- Design & Implementation
- Result & Demo
- **Q&A**

Q&A

非接觸式控制面板

Contactless Control Panel

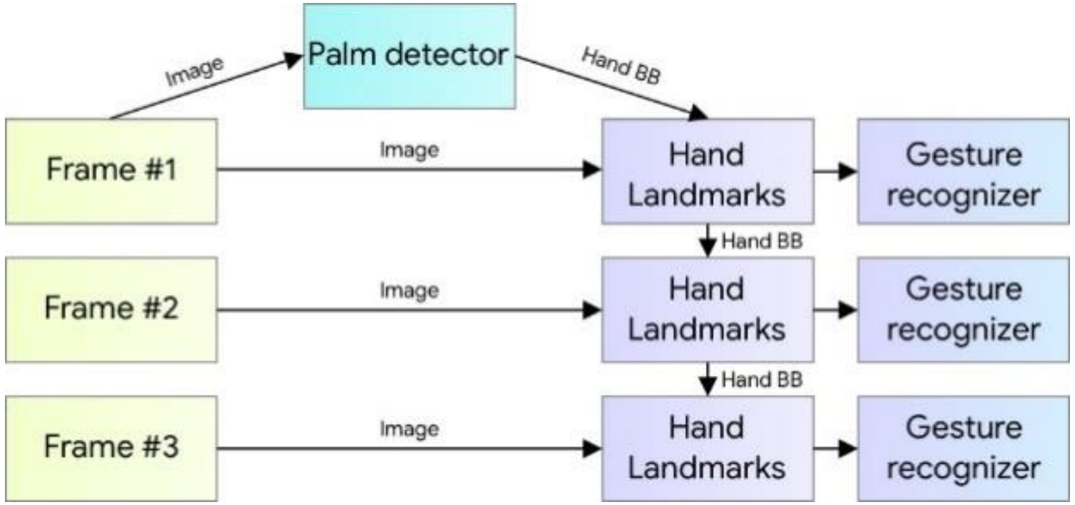
隊名: 綠洲熊與他們的窩

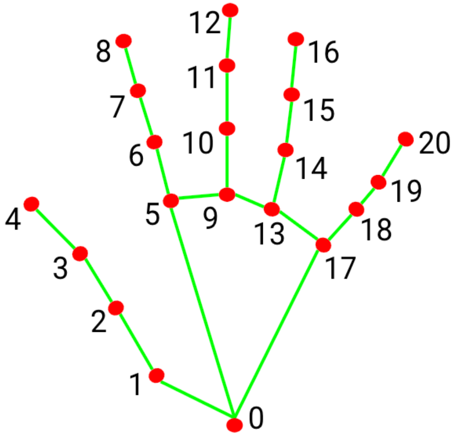
© 2022 Synopsys, Inc.

Appendix - Virtual Button Panel (NVIDIA Jetson Nano)

- **Hand Pose Estimation**
  - – MediaPipe Hand

- **MediaPipe Hand Landmarks**
  - – Palm detector model
  - – Hand landmarks model

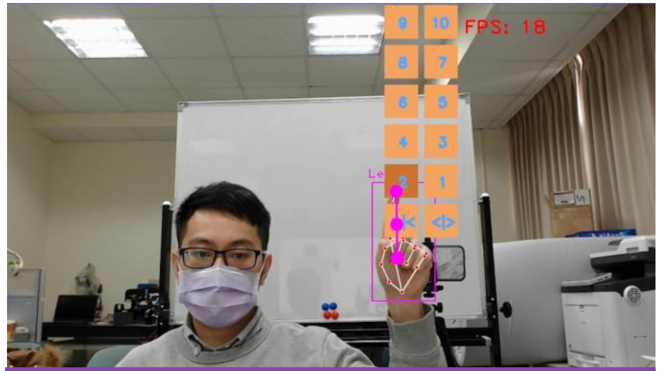




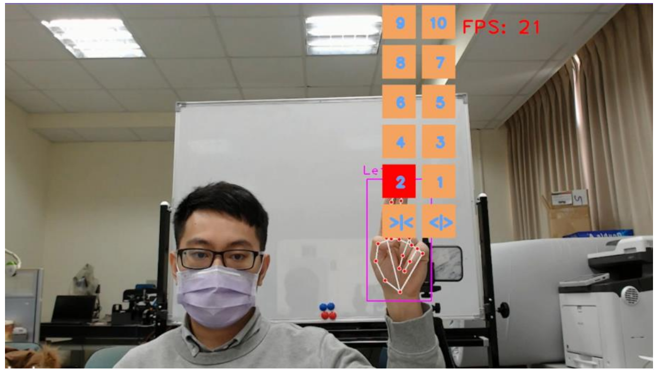
0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Appendix

■ Virtual Button Panel (NVIDIA Jetson Nano)



Select the virtual button

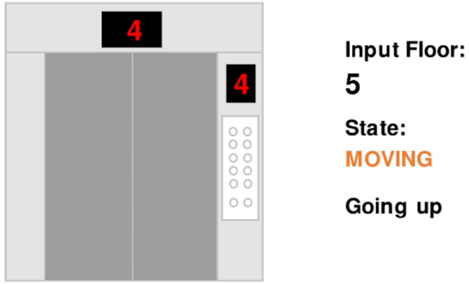


Click the virtual button

NVIDIA Jetson Nano (Developer Kit Version) - Jetpack 4.5.1 [L4T

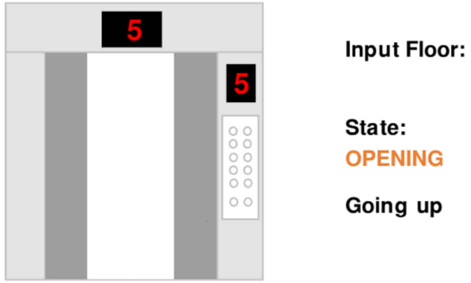
Implement on Jetson Nano

■ Simulation Animation (PC)



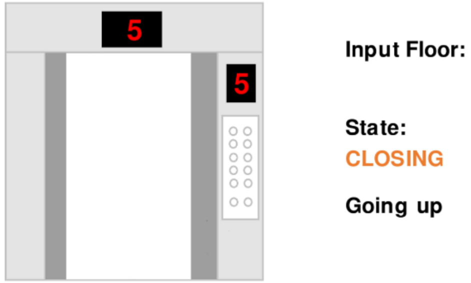
Input Floor:
5
State:
MOVING
Going up

Move



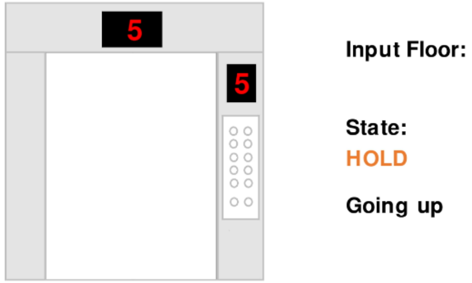
Input Floor:
State:
OPENING
Going up

Open



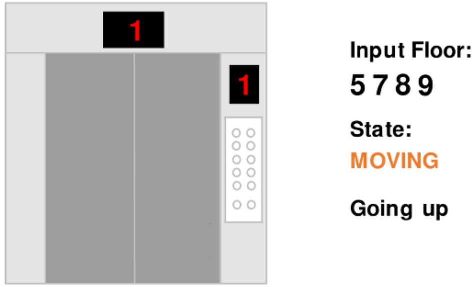
Input Floor:
State:
CLOSING
Going up

Close



Input Floor:
State:
HOLD
Going up

Hold



Input Floor:
5 7 8 9
State:
MOVING
Going up

Multiple floors

Reference

- [1] Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." ICML deep learning workshop. Vol. 2. 2015.
- [2] Sheng, T., Feng, C., Zhuo, S., Zhang, X., Shen, L., & Aleksic, M. (2018, March). A quantization-friendly separable convolution for mobilenets. In *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)* (pp. 14-18). IEEE.

© 2022 Synopsys, Inc.

Thank You