USC University of Southern California

EE 105: Introduction to Electrical Engineering

Lab 4.2

Camera classifier with your Lab 4.1 CNN

Yasser Khan

Rehan Kapadia

Objective

Classify handwritten digits live from the demo board's camera with the CNN you trained in Lab 4.1.

One notebook

It uploads the camera firmware and runs every step; no Arduino IDE, no Colab.

Your Lab 4.1 CNN

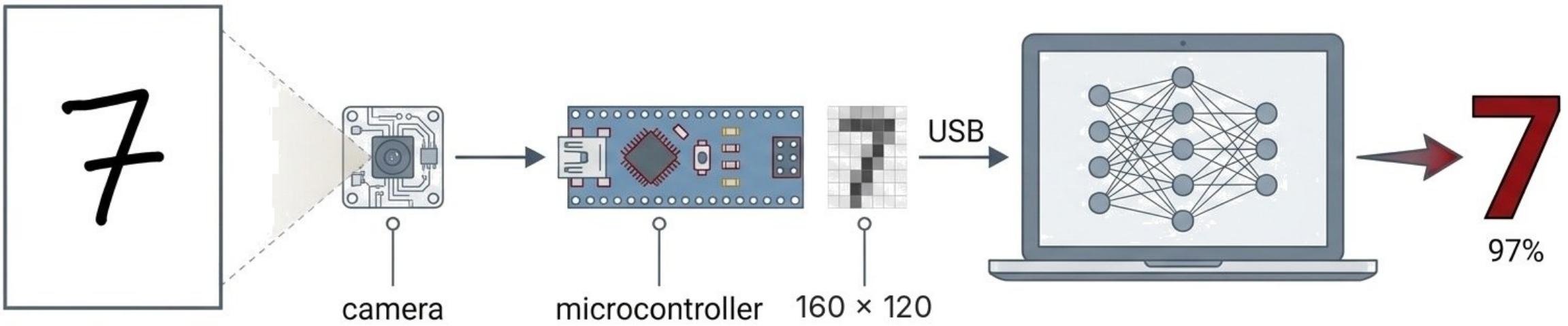
The network saved in Lab 4.1 (mnist_cnn.pt) classifies every picture.

Distribution shift

Measure the accuracy on 20 pictures of your own digits, then improve it.

Real-time classification

The camera streams 160 × 120 gray pictures; the laptop classifies each one.



Files

One notebook runs everything: it loads your Lab 4.1 CNN and uploads the camera firmware itself.

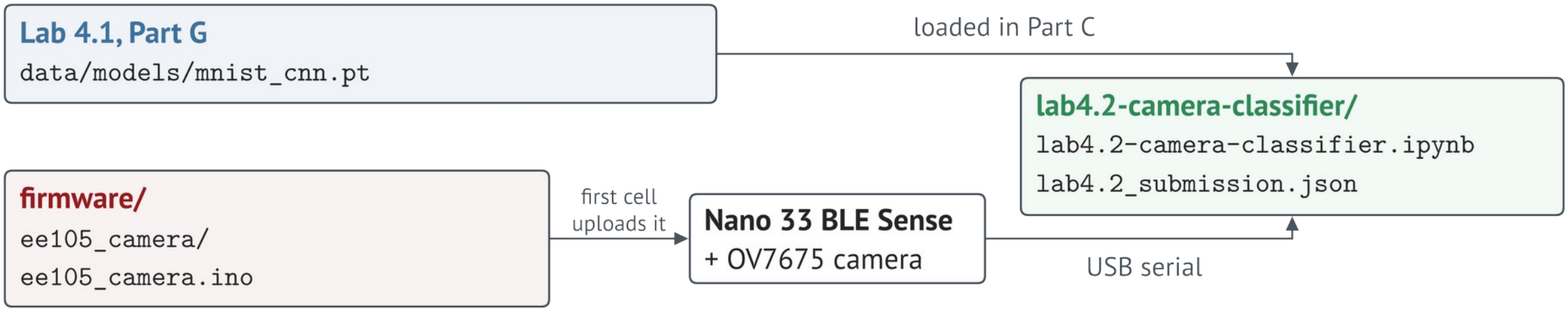
From Lab 4.1

- Trained CNN, saved by step G3:

  data/models/mnist_cnn.pt

Lab 4.2

- Notebook: **lab4.2-camera-classifier.ipynb**
- Camera firmware (uploaded for you): **ee105_camera**
- Results to upload: **lab4.2_submission.json**



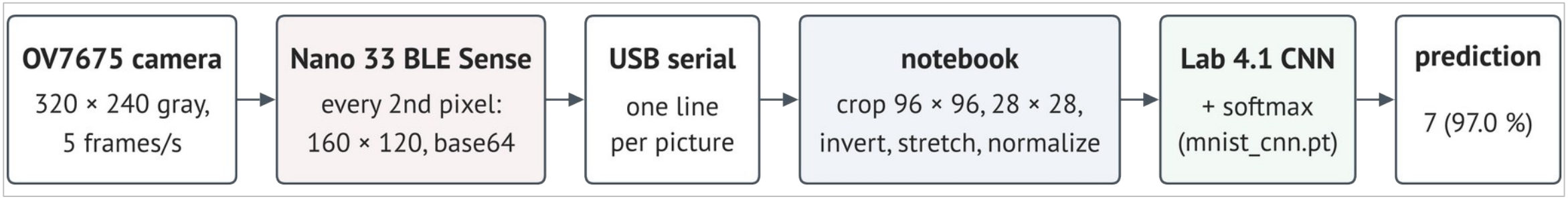
Before the lab

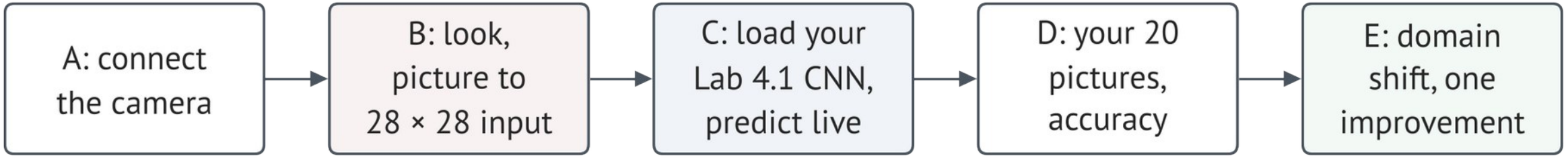
The network comes from Lab 4.1; this lab adds the camera and your own handwriting.

BEFORE THE LAB **Goal:** bring your Lab 4.1 network and ten handwritten digits.

1. Finish Lab 4.1 through step G3 (it saves mnist_cnn.pt)
2. Write 0 to 9 in thick black marker, 4 cm apart

No mnist_cnn.pt? Step C1 trains a stand-in CNN in about 15 s (about 97% on MNIST).





Board setup

Selector on Camera and both jumper caps on the Camera end connect the camera to the Nano.

Setting up the board:

1. Selector **S10** on **Camera**
2. Both caps of **J7** and **J8** on the **Camera** end (pins 1-2)
3. Camera module in **J6**, lens facing out
4. Unplug any other Arduino; plug in the USB cable



Lab 4.2 board setup (book Figure 4.31)

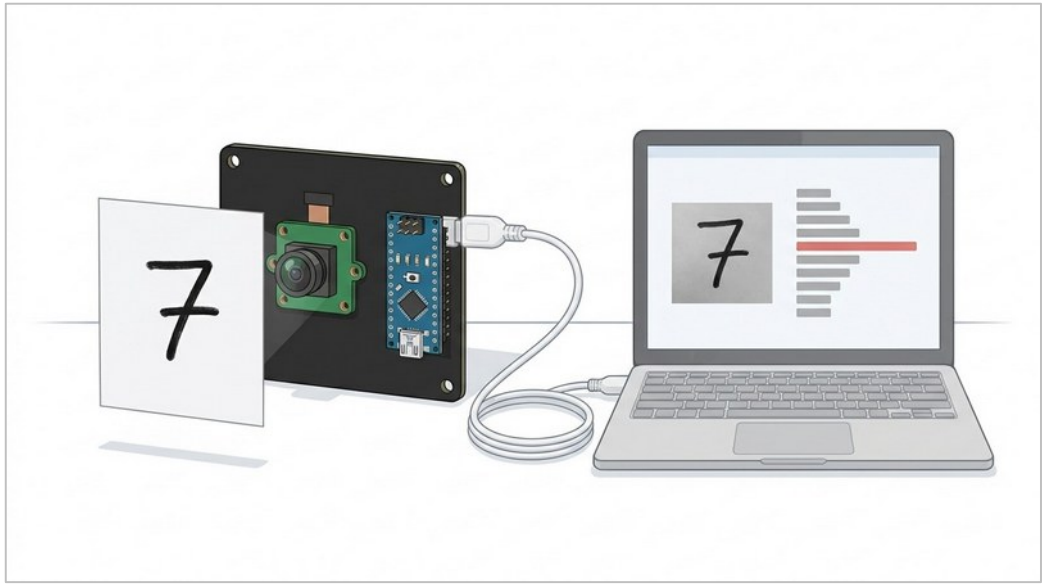
After the lab

Move both caps back to the Module 1 end (pins 2-3). The camera borrows A0 and A1, which the circuits block normally uses.

How a picture travels

The Nano shrinks each camera frame and sends it as text; the laptop turns it into a network input.

1. The camera delivers a **320 × 240** gray frame, 5 frames per second
2. The Nano keeps every 2nd pixel: **160 × 120**
3. It sends one byte per pixel as **base64** text, 4 characters per 3 bytes
4. The notebook cuts a **96 × 96** center square and averages it to 28 × 28
5. Invert, stretch the contrast, then **normalize** exactly as in Lab 4.1



Example (B3): 160 × 120 = 19,200 bytes = 25,600 characters: 2.22 s at 115,200 baud. A picture really takes a few hundred ms; the Nano's USB port ignores the baud rate.

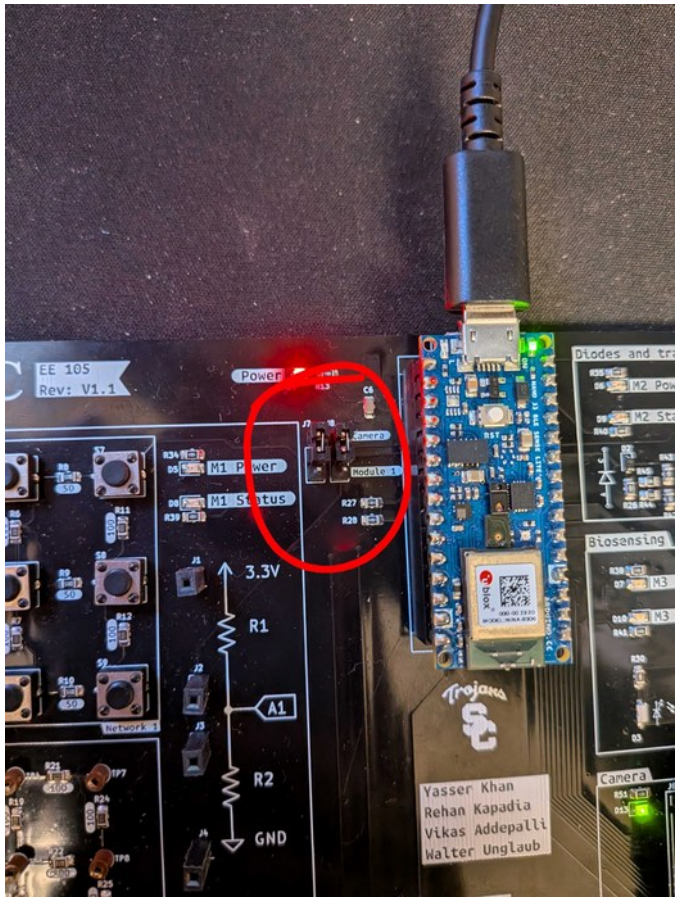
Part A: Connect the camera

Route the camera to the Nano; the first cell uploads the firmware that streams 160 × 120 pictures.

PART A

1. Unplug USB; move both caps of **J7** and **J8** (circled) to the Camera end
2. Turn the selector **S10** to **Camera**; camera in J6
3. Plug in the USB cable; open lab4.2-camera-classifier.ipynb
4. Run the first cell, A1: the first time it **uploads the camera firmware** (about a minute)
5. Both lines it prints start with **[ok]**; the second says S10 is on Camera

```
# A1 · DO  Connect the camera
SIMULATE = False   # True: practice without a board
cam = ee105.connect_camera(simulate=SIMULATE)
cam.check_module("CAM")   # warns if S10 is not on Camera
```



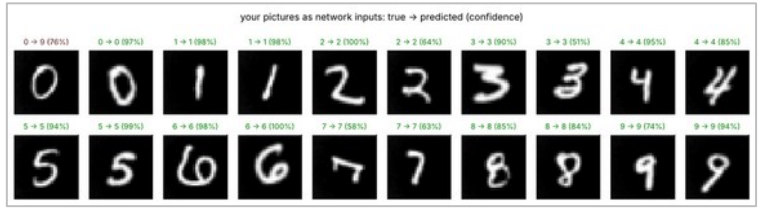
Demo board: J7 and J8 circled

Parts B to E: NN Eval

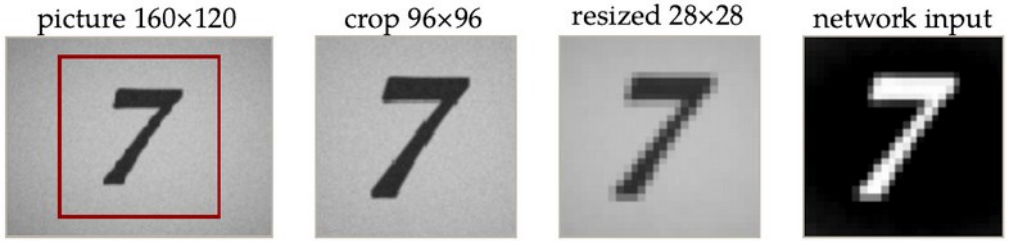
The laptop crops each picture, shrinks it to 28 × 28 and runs the CNN you trained in Lab 4.1.

PARTS B TO E

1. **Look** (B1 to B4): live view; the digit fills the red square
2. **Load** (C1) your Lab 4.1 CNN
3. **Predict** (C2 to C4): live view, three predictions (HW 5, Problem 5)
4. **Test** (D1 to E5): 20 pictures of your digits; accuracy, then improve it



D2 contact sheet: a red title is a wrong prediction



Picture 160 × 120, crop 96 × 96, 28 × 28, then invert and stretch.

Readout **prediction: 7 (97.0 %)**: the confidence is the largest softmax output.

Check: the live view of E5 reads a digit you just wrote. Upload lab4.2_submission.json on ee105.org/portal; move J7 and J8 back to the Module 1 end.