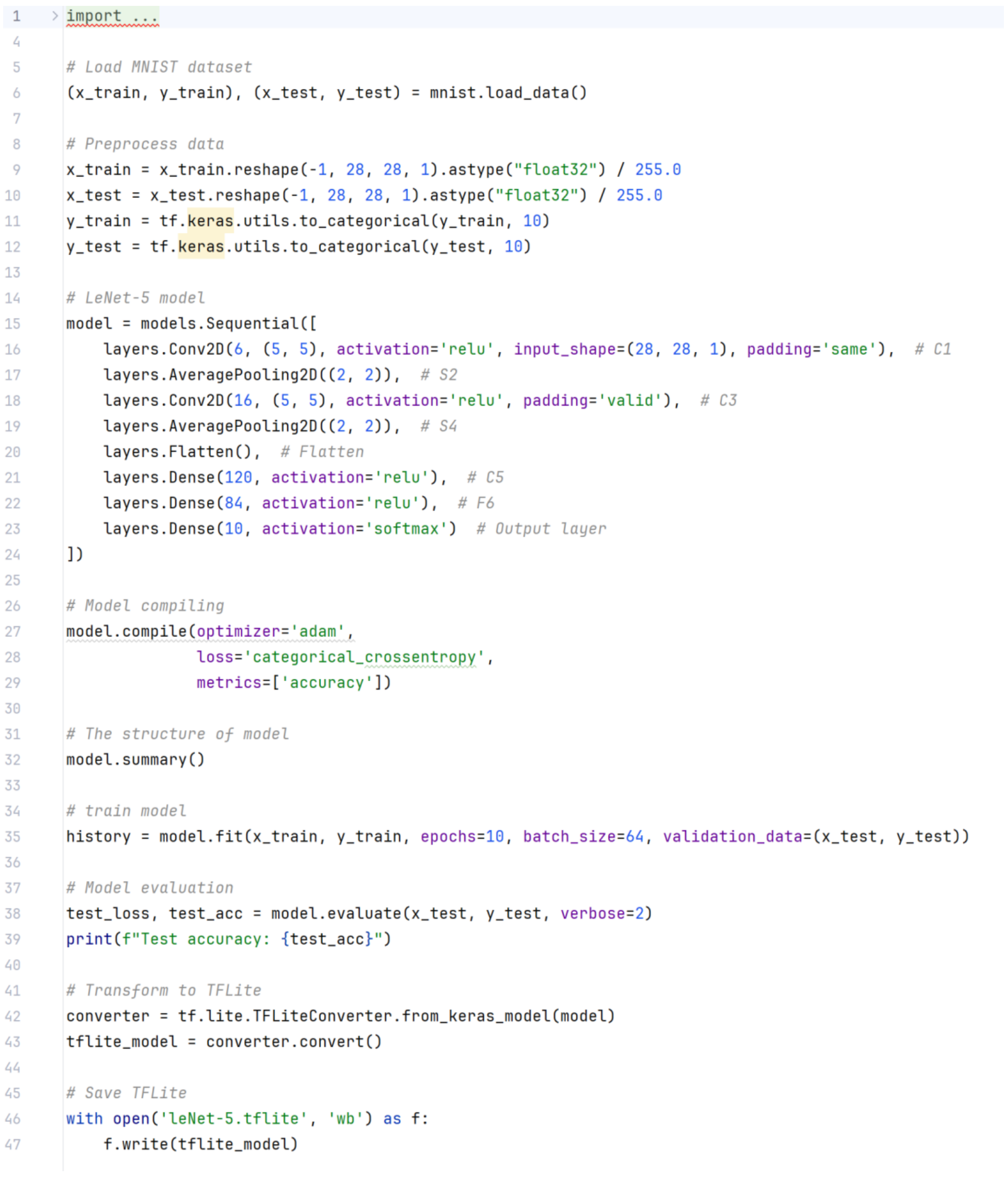
MileStone

1. Model Training

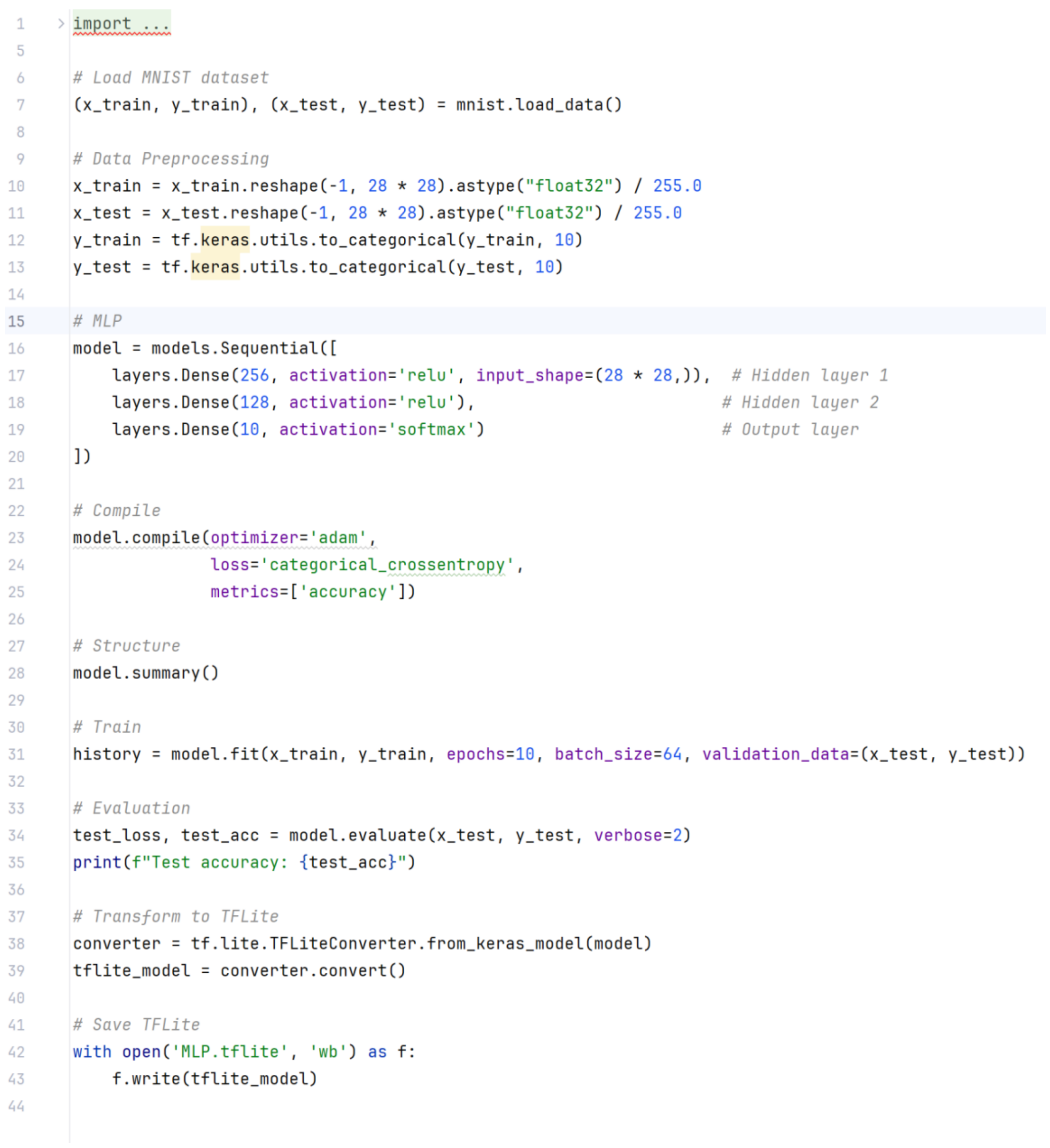
Our group chooses LeNet-5 and MLP to deploy on PYNQ Z1. MNIST dataset is used to train the two models. The trained models are saved as the format of tensorflow lite which is suitable for embedded devices like PYNQ Z1. The training codes are shown below.

LeNet-5



```python
import ...

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Preprocess data
x_train = x_train.reshape(-1, 28, 28, 1).astype("float32") / 255.0
x_test = x_test.reshape(-1, 28, 28, 1).astype("float32") / 255.0
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)

# LeNet-5 model
model = models.Sequential([
    layers.Conv2D(6, (5, 5), activation='relu', input_shape=(28, 28, 1), padding='same'),  # C1
    layers.AveragePooling2D((2, 2)),  # S2
    layers.Conv2D(16, (5, 5), activation='relu', padding='valid'),  # C3
    layers.AveragePooling2D((2, 2)),  # S4
    layers.Flatten(),  # Flatten
    layers.Dense(120, activation='relu'),  # C5
    layers.Dense(84, activation='relu'),  # F6
    layers.Dense(10, activation='softmax')  # Output layer
])

# Model compiling
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# The structure of model
model.summary()

# train model
history = model.fit(x_train, y_train, epochs=10, batch_size=64, validation_data=(x_test, y_test))

# Model evaluation
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"Test accuracy: {test_acc}")

# Transform to TFLite
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save TFLite
with open('leNet-5.tflite', 'wb') as f:
    f.write(tflite_model)
```

MLP

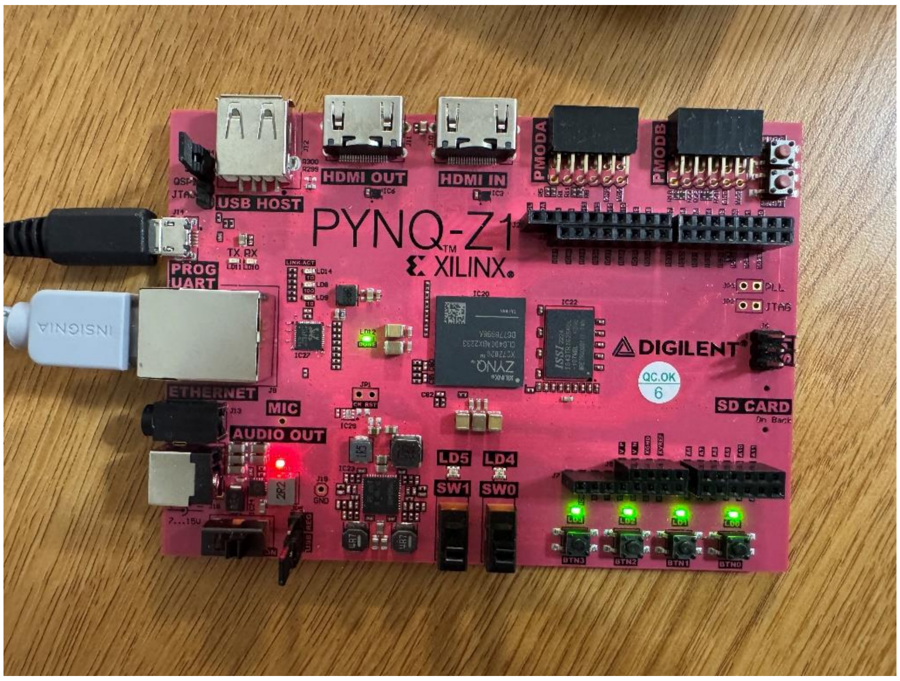


```
import ...

# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Data Preprocessing
x_train = x_train.reshape(-1, 28 * 28).astype("float32") / 255.0
x_test = x_test.reshape(-1, 28 * 28).astype("float32") / 255.0
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)

# MLP
model = models.Sequential([
    layers.Dense(256, activation='relu', input_shape=(28 * 28,)),  # Hidden layer 1
    layers.Dense(128, activation='relu'),                         # Hidden layer 2
    layers.Dense(10, activation='softmax')                        # Output layer
])

# Compile
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Structure
model.summary()

# Train
history = model.fit(x_train, y_train, epochs=10, batch_size=64, validation_data=(x_test, y_test))

# Evaluation
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"Test accuracy: {test_acc}")

# Transform to TFLite
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save TFLite
with open('MLP.tflite', 'wb') as f:
    f.write(tflite_model)
```

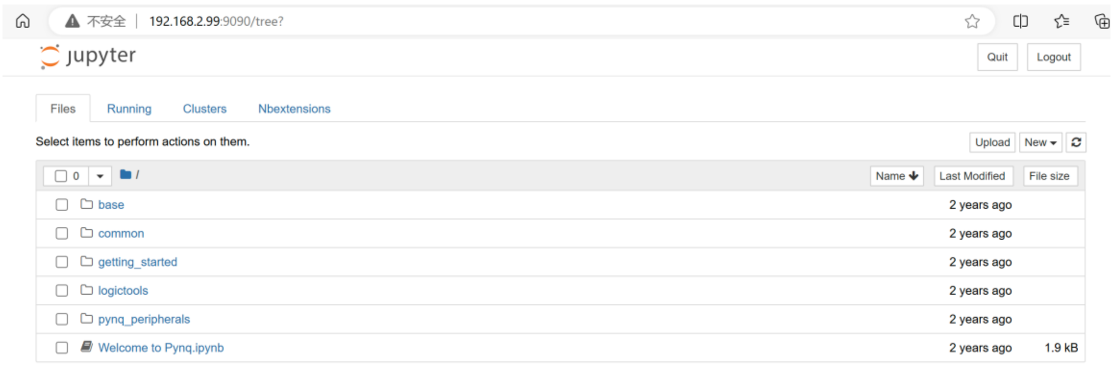
2. Try Connecting PYNQ Z1 to Computer

To connect PYNQ Z1, we followed the instructions on the official site. First, card reader and Win32DiskImager were used to flash the PYNQ-Z1 image onto a Micro SD card. Then inserted it into the board and connected the ethernet and USB cable. After assigning a static IP address, the board successfully built direct connection to the computer. To connect to Jupyter Notebooks, opened a web browser and navigated to http://192.168.2.99 where we can view and run the notebook documentation interactively. Our work is shown below.

Connected PYNQ Z1 Board



Jupyter Notebook



不安全 | 192.168.2.99:9090/tree?

jupyter

Quit Logout

Files Running Clusters Nbextensions

Select items to perform actions on them.

Upload New

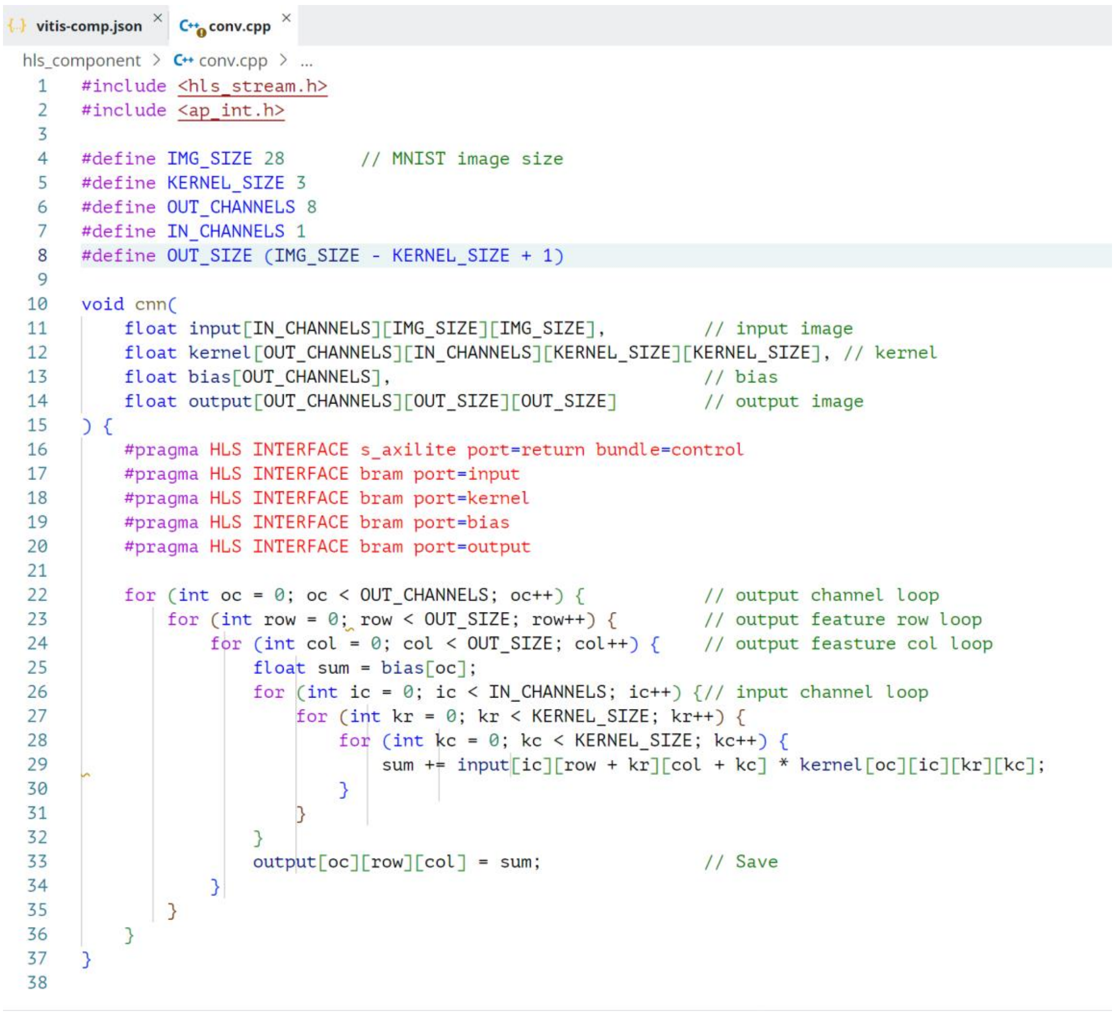
0 /

Name	Last Modified	File size
base	2 years ago	
common	2 years ago	
getting_started	2 years ago	
logictools	2 years ago	
pynq_peripherals	2 years ago	
Welcome to Pynq.ipynb	2 years ago	1.9 kB

3. Program Hardware (in progress)

In this part, visit 2024 is used to design the FPGA accelerator. First, we created an HLS project, and added cnn top-level function to the project, which can increase the processing speed of convolutional layer. Then clicked synthesis to create RTL files. Once succeeded, exported the IP core. All of these I mentioned above are what we have achieved so far. In the following work, we will use Vivado to add and connect modules, and generate hardware design, and output it as bitstream file which will enable the hardware accelerator to function. The speedup may not be obvious since we have only designed cnn accelerator and MLP model doesn't contain convolutional layer. Therefore, we will design more accelerators to speed up PYNQ Z1's inference.

Top-level Function cnn Design



```cpp
#include <hls_stream.h>
#include <ap_int.h>

#define IMG_SIZE 28       // MNIST image size
#define KERNEL_SIZE 3
#define OUT_CHANNELS 8
#define IN_CHANNELS 1
#define OUT_SIZE (IMG_SIZE - KERNEL_SIZE + 1)

void cnn(
    float input[IN_CHANNELS][IMG_SIZE][IMG_SIZE],         // input image
    float kernel[OUT_CHANNELS][IN_CHANNELS][KERNEL_SIZE][KERNEL_SIZE], // kernel
    float bias[OUT_CHANNELS],                             // bias
    float output[OUT_CHANNELS][OUT_SIZE][OUT_SIZE]        // output image
) {
    #pragma HLS INTERFACE s_axilite port=return bundle=control
    #pragma HLS INTERFACE bram port=input
    #pragma HLS INTERFACE bram port=kernel
    #pragma HLS INTERFACE bram port=bias
    #pragma HLS INTERFACE bram port=output

    for (int oc = 0; oc < OUT_CHANNELS; oc++) {           // output channel loop
        for (int row = 0; row < OUT_SIZE; row++) {        // output feature row loop
            for (int col = 0; col < OUT_SIZE; col++) {    // output feasture col loop
                float sum = bias[oc];
                for (int ic = 0; ic < IN_CHANNELS; ic++) {// input channel loop
                    for (int kr = 0; kr < KERNEL_SIZE; kr++) {
                        for (int kc = 0; kc < KERNEL_SIZE; kc++) {
                            sum += input[ic][row + kr][col + kc] * kernel[oc][ic][kr][kc];
                        }
                    }
                }
                output[oc][row][col] = sum;               // Save
            }
        }
    }
}

```

4. Respond to feedback

We have improved our project based on feedback from the project proposal.

a. MNIST dataset has been added to Github.
b. For programing fpga fabric as an accelerator, we have shown our work in Program Hardware part above.
c. Our main target is to compare the inference time of the two models on the board and on the computer. We will also try connecting the board to a camera to make it recognize the handwriting digits in real world.