リスト1　Pythonパッケージのインストール↓

```
pip3 install protobuf==3.19.6
pip3 install tensorflow==2.9.1
pip3 install torch==1.12.1+cpu torchvision==0.13.1+cpu -f https://download.pytorch.org/whl/torch_stable.html
pip3 install torchsummary==1.5.1
pip3 install onnx==1.12.0
pip3 install progressbar33==2.4
pip3 install prettytable==3.4.0
pip3 install h5py==3.6.0
pip3 install pycryptodome==3.15.0
pip3 install configparser==5.3.0
pip3 install psutil==5.9.1
```

リスト2　モデル実行用のサンプル・プログラム↓

```
// ヘッダ・ファイルのインクルード
:
// C++から呼び出す場合
#ifdef __cplusplus
extern "C" {
#endif
TsOUT* dnn_compute(TsIN* input_img, TsInt* errorcode);
#ifdef __cplusplus
}
#endif
:
int main(void) {
:
    TsIN in = {/* 入力データの一次元配列のポインタ */};
    TsInt err = 0;
    TsOUT* out = dnn_compute(&in, &err);
    /* out: 出力データの一次元の配列のポインタ */
:
}
```

リスト3　異常検知プログラムdetect_anomaly.pyのmain関数↓

```
def main(datatype: str):
    THRESHOLD = load_threshold(datatype)
    interpreter, in_d, out_d, model_path = load_model(datatype)
    in_scale, in_zp = in_d['quantization']
    out_scale, out_zp = out_d['quantization']
: (省略)
    scaler_mean, scaler_std = load_scaler()
    sensor = ADXL345()
    queue = deque(maxlen=WINDOW_SIZE)
: (省略)
    try:
        # 0.1秒ごとの周期実行(処理時間を考慮)
        next_time = time.time()
        while True:
            # メインループ (約 100 ms 周期):
            # 1) Ctrl-C ポーリングで停止指示を確認
            # --- センサー読み取りと推論処理 ---
            # 2) ADXL345 から生データ取得
            x, y, z = sensor.read_axes()
            queue.append([x, y, z])

            if len(queue) == WINDOW_SIZE:
                data = np.array(queue, dtype=np.float32)  # (W, 3)
                # 3) Z スコア正規化 → クリップ
                zscore = (data - scaler_mean) / (scaler_std + 1e-8)
                z_clip = np.clip(zscore, -K_CLIP, K_CLIP).flatten().reshape(1, -1).astype(np.float32)

                if datatype == "int8":
                    # 4) INT8 へ量子化し、ウィンドウバッファへプッシュ
                    q_in = np.round(z_clip / in_scale + in_zp).astype(np.int8)
                    interpreter.set_tensor(in_d['index'], q_in)
                    # 5) DNN 推論 (dnn_compute)
                    interpreter.invoke()
                    q_out = interpreter.get_tensor(out_d['index']).astype(np.int8)
                    # 6) 出力を逆量子化して MSE 算出
                    y_hat = out_scale * (q_out.astype(np.int32) - out_zp)
                    x_deq = in_scale  * (q_in.astype(np.int32) - in_zp)
                    mse = np.mean((x_deq - y_hat) ** 2)
                else:
: (省略)
                # 7) しきい値で NORMAL / ANOMALY 判定 → シリアル出力 & LED 制御
                status = "ANOMALY" if mse > THRESHOLD else "NORMAL"
                print(f"{status} | MSE={mse:.6f}")
            # --- 周期制御: 0.1秒ごとに実行 ---
: (省略)
    except KeyboardInterrupt:
        print("¥n◉ 終了します。")
```

リスト4　AnomalyDetect.pyのloop()関数部分↓

```
// メインループ（約 100 ms 周期）:
void loop(void)
  // メインループ（約 100 ms 周期）
{
  //  1) Ctrl-C ポーリングで停止指示を確認
  poll_control();
  if (!g_run) { delay(10); return; }
  static unsigned long last_loop_ms = 0;
  unsigned long now_ms = millis();
  if ((unsigned long)(now_ms - last_loop_ms) < LOOP_INTERVAL_MS) {
    return;
  }
  last_loop_ms = now_ms;

unsigned long elapsed_ms = millis() - start_time_ms;
  float elapsed_s = (float)elapsed_ms / 1000.0f;
  //  2) ADXL345 から生データ取得
  int16_t rx, ry, rz;
  if (!adxl_read_raw(rx, ry, rz)) {
    return;
  }
  //  3) Z スコア正規化 → クリップ
  const float xn0 = (((float)rx - SCALER_MEAN[0]) * INV_SCALER_STD[0]);
  const float yn0 = (((float)ry - SCALER_MEAN[1]) * INV_SCALER_STD[1]);
  const float zn0 = (((float)rz - SCALER_MEAN[2]) * INV_SCALER_STD[2]);
  float xn = clip3(xn0);
  float yn = clip3(yn0);
  float zn = clip3(zn0);
  //  4) INT8 へ量子化し、ウィンドウバッファへプッシュ
  const int8_t xi8 = trans(xn);
  const int8_t yi8 = trans(yn);
  const int8_t zi8 = trans(zn);
  if (showMeasDebug) {
    Serial.print(F("  "));
    Serial.print(F(" | raw=(")); Serial.print(rx); Serial.print(',');
    Serial.print(ry); Serial.print(','); Serial.print(rz);
    Serial.print(F(") | zn0=(")); Serial.print(xn0, 3); Serial.print(',');
    Serial.print(yn0, 3); Serial.print(','); Serial.print(zn0, 3);
    Serial.print(F(") | zn_clip=(")); Serial.print(xn, 3); Serial.print(',');
    Serial.print(yn, 3); Serial.print(','); Serial.print(zn, 3);
    Serial.print(F(") | qi8=(")); Serial.print(xi8); Serial.print(',');
    Serial.print(yi8); Serial.print(','); Serial.print(zi8);
    Serial.print(')');
  Serial.println();
  }
    // 時系列ウィンドウバッファに最新サンプルをプッシュ(古いデータを後方へシフト)
  push_sample(data_in, xi8, yi8, zi8);
    // ウィンドウが満たされるまでの充填カウント(ウォームアップ表示用)
  if (sample_count < WINDOW_SIZE) sample_count++;
  if (sample_count < WINDOW_SIZE) {
    Serial.print(F("t="));
    Serial.print(elapsed_s, 1);
    Serial.println(F("s"));
  }
    // ウィンドウ(WINDOW_SIZE)に達したら推論と判定を実行
  if (sample_count == WINDOW_SIZE) {
    input_data  = (TsIN*)&data_in[0][0];
    //  5) DNN 推論 (dnn_compute)
    output_data = dnn_compute(input_data, &errorcode);
    //  6) 出力を逆量子化して MSE 算出
    float mse_f_raspi = calc_mse_float_raspi((const int8_t*)input_data, (const int8_t*)output_data);
    //  7) しきい値で NORMAL / ANOMALY 判定 → シリアル出力 & LED 制御
    if (mse_f_raspi > THRESHOLD_MSE_FLOAT_ZN) {
      Serial.print(F("t=")); Serial.print(elapsed_s, 1); Serial.print(F("s | ANOMALY | MSE=")); Serial.print(mse_f_raspi,
4); Serial.println();
      digitalWrite(LED_PIN, LED_ON);
    } else {
      Serial.print(F("t=")); Serial.print(elapsed_s, 1); Serial.print(F("s | NORMAL | MSE=")); Serial.print(mse_f_raspi,
4); Serial.println();
      digitalWrite(LED_PIN, LED_OFF);
    }
  }
}
```

リスト5　AnomalyDetection.inoの実行結果↓

```
Anomaly Detection Start
Press Ctrl-C to pause, 'g' to resume.
SCALE_IN=0.014117, ZEROPOINT_IN=18
SCALE_OUT=0.018190, ZEROPOINT_OUT=4
SCALER_MEAN=-69.868690,16.962446,83.402786
SCALER_STD=77.839569,92.759888,177.568314
K_CLIP=3.000, THRESHOLD=0.141297
t=0.1s
:
```

```
t=0.9s
t=1.0s | NORMAL | MSE=0.0312
:
t=2.1s | NORMAL | MSE=0.0260
t=2.2s | ANOMALY | MSE=0.1990
:
t=3.5s | ANOMALY | MSE=0.1661
t=3.6s | NORMAL | MSE=0.0338
:
```

リスト6　AnomalyDetection.inoのLEDピンとLED ON/OFFの極性を変更↓

```
#define LED_PIN BUILTIN_LED1
#define LED_ON  HIGH
#define LED_OFF LOW
```