# 伍、智慧醫療2-跌倒偵測

胡國聖

2025/02/12

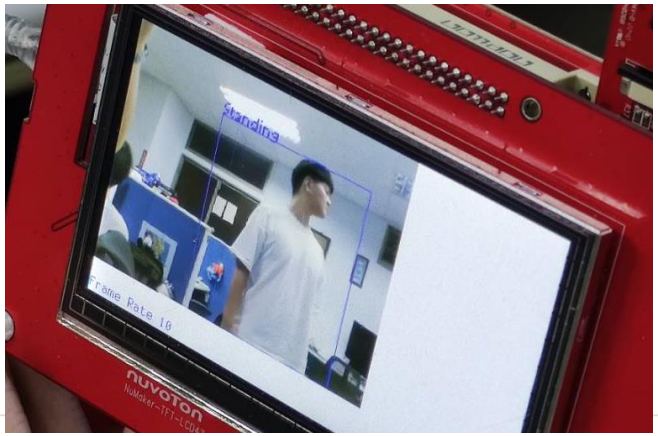
# 大綱

- 伍、智慧醫療2-跌倒偵測
  - 一、案例介紹
  - 二、跌倒偵測技術種類介紹
  - 三、使用Roboflow的開源資料集
  - 四、在PC上使用Anaconda環境進行模型訓練
  - 五、開發版上的推論程式系統流程步驟
  - 六、結論與未來發展

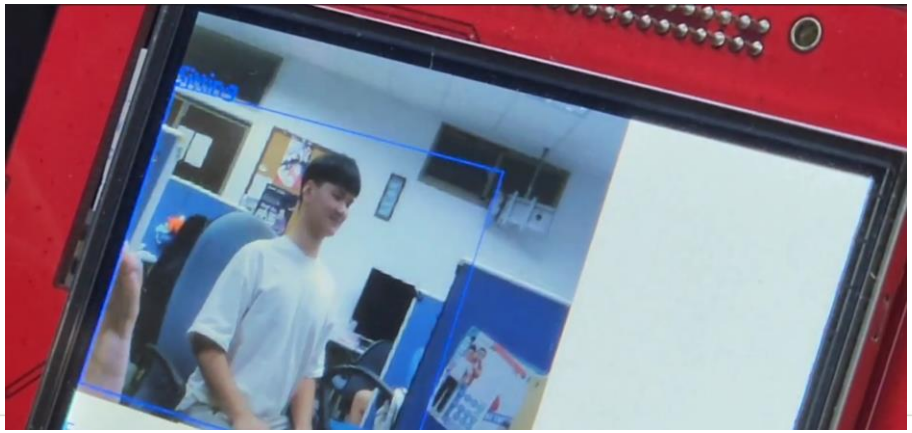
# 一、 案例介紹-跌倒偵測

- (一) 專案摘要
  - 本系統由新唐M55M1開發板判斷人的3種姿勢Fall-down、sitting、standing，資料集來源為Roboflow取得，在Pytorch框架訓練Yolox-nano模型，後續經過模型框架轉換和輕量化後轉成Tensorflow-lite框架，再經過Vela編譯處理即可部屬至開發板上，最後將C語言軟體和模型經由KEIL燒錄至開發板。使用機器學習的方式判斷偵測範圍內是否有人跌倒，不僅能夠減少醫療人力需求且能更快偵測到緊急情況。

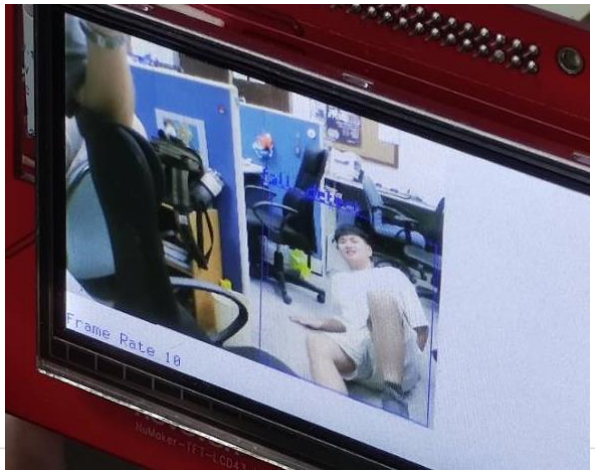
**Standing**




**Sitting**



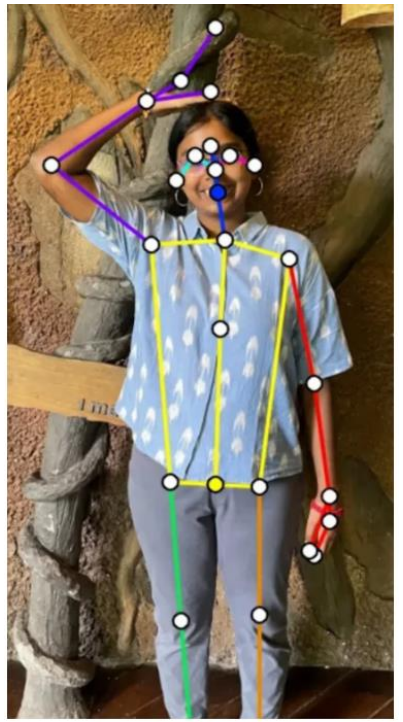
**Fall-down**



# 二、跌倒偵測技術種類介紹

**Keypoint-detection**

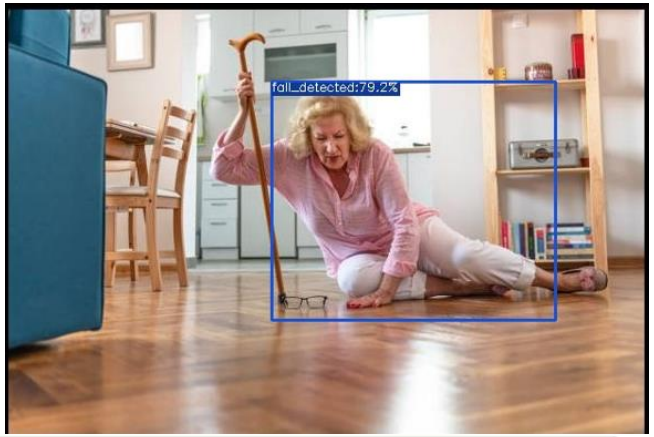
https://blog.roboflow.com/pose-estimation-algorithms-history/



- Keypoint-detection
  - 將圖片中人的軀幹和肢體狀態做標註，對姿勢的辨識準確率更高，但標註過程更為複雜且需要更大量的資料集才能準確判斷，適用於辨識姿勢種類變化差異小的應用(如辨識坐姿好壞)

- Object-detection (本次案例採用)
  - 和一般物件偵測原理相同，標記過程只需框出圖片中的目標並標記姿勢種類即可，相對較簡單，且以本次應用的3種姿勢(Fall-detected、sitting、standing )姿勢變化大的情況下，準確率也相當良好

**Object-detection**



# 三、使用Roboflow的開源資料集

- 資料集來源
  - 使用Roboflow Universe上找尋適用跌倒偵測訓練的資料，以COCO JSON格式下載即可提供給Pytorch做訓練
  - 資料集:Fall Detection Dataset
  - 將目標姿勢分為Fall-detected、sitting、standing 3種狀態判斷
  - Train set: 731 images
  - Valid set: 208 images
  - Test set: 104 images


# 四、在PC上使用Anaconda環境進行模型訓練

- (一) 訓練環境安裝
  - 1. 建立python環境
    - $ conda create --name yolox_nano  python=3.12
    - $ conda activate yolox_nano
      - 透過上述指令在anaconda上建立環境
  - 2. upgrade pip
    - $ python -m pip install --upgrade pip setuptools
  - 3. 至pytorch官網安裝pytorch
    - $ python -m pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118

# 四、在PC上使用Anaconda環境進行模型訓練

- (一) 訓練環境安裝
  - 4. 安裝 mmcv
    - $ python -m pip install mmcv==2.0.1 -f https://download.openmmlab.com/mmcv/dist/cu118/torch2.0/index.html
  - 5. 安裝其它需求套件:
    - 開啟安裝好的檔案目錄並透過下面指令下載
    - $ python -m pip install --no-input -r requirements.txt
  - 6. Installing the YOLOX
    - $ python setup.py develop

# 四、在PC上使用Anaconda環境進行模型訓練

- (二) 模型訓練設定
  - 將資料集整理成以下形式放在Dataset目錄中
  - Datasets/
    - <your_datasets_name>/
      - annotatios/
        - <train_annotation_json_file>
        - <val_annotation_json_file>
      - train2017/
        - <train_img>
      - val2017/
        - <validation_img>

# 四、在PC上使用Anaconda環境進行模型訓練

- (二) 模型訓練設定
  - 在訓練檔yolox_nano_ti_lite_nu.py中指定訓練和驗證資料集位置和訓練設定
    - 設定適量迭代次數(500次需3小時)
  - 類別檔coco_classes.py中定義資料集種類數
    - (none, fall-detected, sitting, standing)

```
COCO_CLASSES = (
    "none",
    "fall_detected",
    "sitting",
    "standing"
)
```

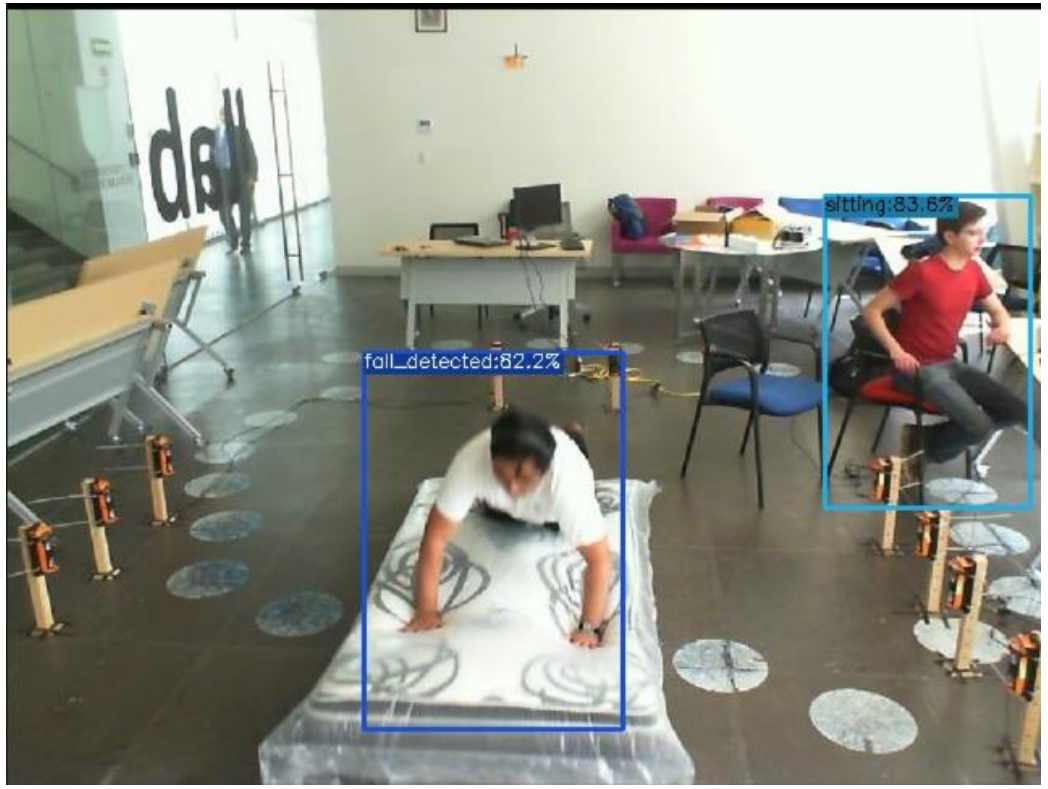
```
# Define yourself dataset path
self.data_dir = "datasets/COCO"
self.train_ann = "train_annotations.coco.json"
self.val_ann = "val_annotations.coco.json"

# --------------  training config --------------------- #
self.num_classes = 4
self.warmup_epochs = 5
self.max_epoch = 500
```

  - 輸入訓練指令
    - $ python tools/train.py -f exps/default/yolox_nano_ti_lite_nu.py -d 1 -b 64 --fp16 -o -c pretrain/tflite_yolox_nano_ti/320_DW/yolox_nano_320_DW_ti_lite.pth

# 四、在PC上使用Anaconda環境進行模型訓練

- Pytorch框架訓練yolox-nano模型結果
  - 訓練測試集104張圖片僅10張圖片有誤(準確率90.3%)

# 四、在PC上使用Anaconda環境進行模型訓練

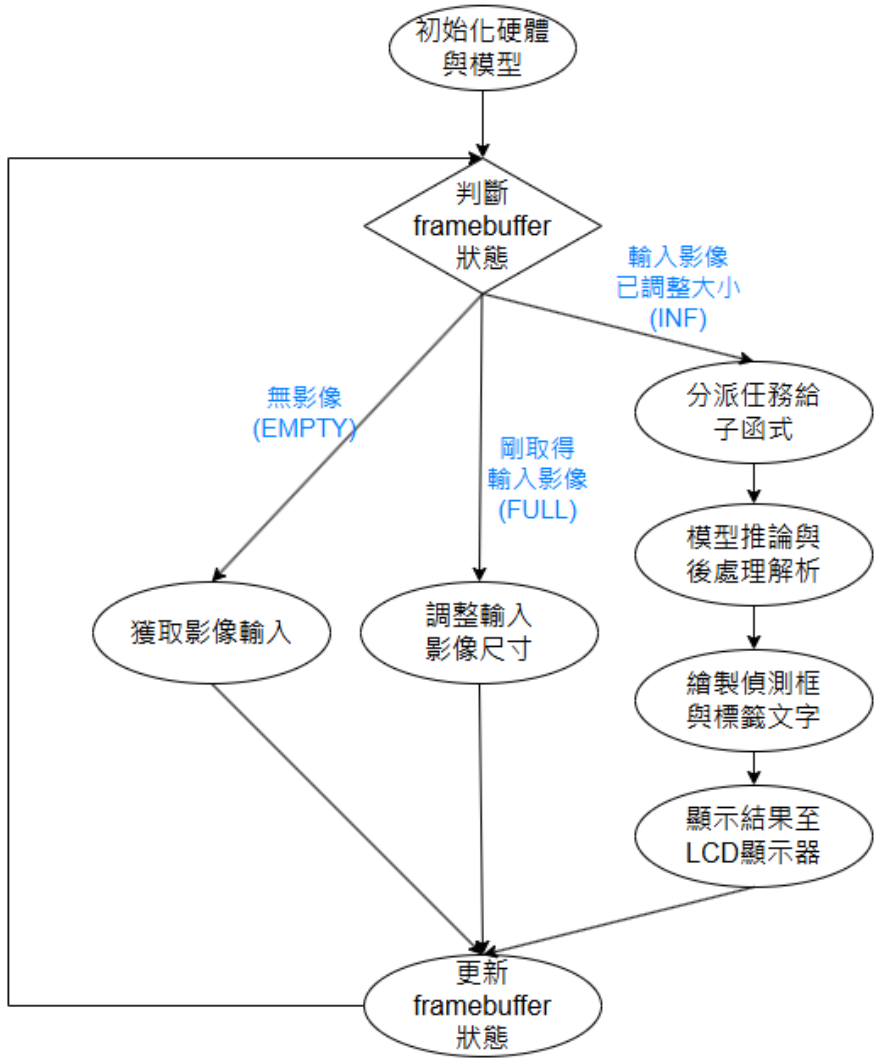
- (三) 模型框架轉換
  - 將訓練完成的checkpoint檔(.ckpt)轉成onnx檔(.onnx)
    - $ python tools/export_onnx.py -f exps/default/yolox_nano_ti_lite_nu.py -c YOLOX_outputs/yolox_nano_ti_lite_nu/latest_ckpt.pth --output-name YOLOX_outputs/yolox_nano_ti_lite_nu/yolox_nano_nu_fall.onnx
  - 將onnx檔轉成tensorflow框架並輕量化轉成tflite檔(.tflite)
    - $ python demo/TFLite/generate_calib_data.py --img-size 320 320 --n-img 200 -o YOLOX_outputs\yolox_nano_ti_lite_nu\calib_data_320x320_n200.npy --img-dir datasets\COCO\train2017
    - $ onnx2tf -i YOLOX_outputs/yolox_nano_ti_lite_nu/yolox_nano_nu_fall2.onnx -oiqt -qcind images YOLOX_outputs\yolox_nano_ti_lite_nu\calib_data_320x320_n200.npy "[[[[0,0,0]]]]" "[[[[1,1,1]]]]"

# 四、在PC上使用Anaconda環境進行模型訓練

- (四) Vela編譯步驟:
  - 將輕量化完的模型放到 vela\generated\
  - 並且修改vela目錄下的 variables.bat
    - $ set MODEL_SRC_FILE=yolox_nano_nu_fall_full_integer_quant.tflite
    - $ set MODEL_OPTIMISE_FILE=yolox_nano_nu_fall_full_integer_quant_vela.tflite
  - 執行gen_model_cpp檔案
    - 執行結果會出現在 vela\generated\yolox_nano_ti_lite_nu_fall_full_integer_quant_vela.tflite.cc

# 五、開發版上的推論程式系統流程步驟

- **C語言程式系統流程圖:**
  - 初始化硬體與模型
  - 根據Frame buffer不同狀態進行不同作業
  - 更新Frame buffer狀態



# 五、開發版上的推論程式系統流程步驟

- **(一) 初始化硬體與模型:**
  - 初始化硬體資源、模型，包括：設定緩衝區的尺寸與格式，將緩衝區的狀態設為 EMPTY，表示可用於存儲新影像。

```
static void omv_init()
{
    image_t frameBuffer;
    int i;

    frameBuffer.w = GLCD_WIDTH;
    frameBuffer.h = GLCD_HEIGHT;
    frameBuffer.size = GLCD_WIDTH * GLCD_HEIGHT * 2;
    frameBuffer.pixfmt = PIXFORMAT_RGB565;

    _fb_base = fb_array;
    _fb_end =  fb_array + OMV_FB_SIZE - 1;
    _fballoc = _fb_base + OMV_FB_SIZE + OMV_FB_ALLOC_SIZE;
    _jpeg_buf = jpeg_array;

    fb_alloc_init0();

    framebuffer_init0();
    framebuffer_init_from_image(&frameBuffer);

    for (i = 0 ; i < NUM_FRAMEBUF; i++)
    {
        s_asFramebuf[i].eState = eFRAMEBUF_EMPTY;
    }

    framebuffer_init_image(&s_asFramebuf[0].frameImage);
```

# 五、開發版上的推論程式系統流程步驟

- **(二) 獲取影像輸入:**
  - 從相機捕捉影像或從內嵌影像載入圖片
  - 更新Framebuffer狀態至FULL

```
545  #if defined (__USE_CCAP__)
546              //capture frame from CCAP
547  #if defined(__PROFILE__)
548              u64CCAPStartCycle = pmu_get_systick_Count();
549  #endif
550
551              ImageSensor_Capture((uint32_t)(emptyFramebuf->frameImage.data));
552
553  #if defined(__PROFILE__)
554              u64CCAPEndCycle = pmu_get_systick_Count();
555              info("ccap capture cycles %llu \n", (u64CCAPEndCycle - u64CCAPStartCycle));
556  #endif
557
558  #else
559              //copy source image to frame buffer
560              image_t srcImg;
561
562              srcImg.w = IMAGE_WIDTH;
563              srcImg.h = IMAGE_HEIGHT;
564              srcImg.data = (uint8_t *)pu8ImgSrc;
565              srcImg.pixfmt = PIXFORMAT_RGB888;
566
567              roi.x = 0;
568              roi.y = 0;
569              roi.w = IMAGE_WIDTH;
570              roi.h = IMAGE_HEIGHT;
571
572              imlib_nvt_scale(&srcImg, &emptyFramebuf->frameImage, &roi);
573  #endif
```

# 五、開發版上的推論程式系統流程步驟

- **(三) 調整輸入影像尺寸:**
  - 將輸入影像調整為模型尺寸大小
  - 更新Framebuffer狀態至INF

```
395         if (fullFramebuf)
396         {
397             //resize full image to input tensor
398             image_t resizeImg;
399 
400             roi.x = 0;
401             roi.y = 0;
402             roi.w = fullFramebuf->frameImage.w;
403             roi.h = fullFramebuf->frameImage.h;
404 
405             resizeImg.w = inputImgCols;
406             resizeImg.h = inputImgRows;
407             resizeImg.data = (uint8_t *)inputTensor->data.data; //direct resize to input tensor buffer
408             resizeImg.pixfmt = PIXFORMAT_RGB888;
409 
410 #if defined(__PROFILE__)
411             u64StartCycle = pmu_get_systick_Count();
412 #endif
413             imlib_nvt_scale(&fullFramebuf->frameImage, &resizeImg, &roi);
```

# 五、開發版上的推論程式系統流程步驟

- **(四) 處理待推論影像:**
  - 分派任務給相關子函式，需要進行推論及後處理解析、繪製物件偵測框、顯示結果至LCD顯示器、更新Framebuffer狀態至EMPTY

```
378         if (infFramebuf)
379         {
380             /* Detector post-processing*/
381 
382             inferenceJob->responseQueue = inferenceResponseQueue;
383             inferenceJob->pPostProc = &postProcess;
384             inferenceJob->modelCols = inputImgCols;
385             inferenceJob->mode1Rows = inputImgRows;
386             inferenceJob->srcImgWidth = infFramebuf->frameImage.w;
387             inferenceJob->srcImgHeight = infFramebuf->frameImage.h;
388             inferenceJob->results = &infFramebuf->results; //&results;
389 
390             xQueueReceive(inferenceResponseQueue, &inferenceJob, portMAX_DELAY);
391         }
```

# 五、開發版上的推論程式系統流程步驟

- (四) 處理待推論影像:

```
            if ((uint64_t) pmu_get_systick_Count() > u64PerfCycle)
            {
                info("Total inference rate: %llu\n", u64PerfFrames / EACH_PERF_SEC);
#if defined (__USE_DISPLAY__)
                sprintf(szDisplayText, "Frame Rate %llu", u64PerfFrames / EACH_PERF_SEC);
                //              sprintf(szDisplayText,"Time %llu",(uint64_t) pmu_get_systick_Count() / (uint64_t)SystemCoreClock);

                sDispRect.u32TopLeftX = 0;
                sDispRect.u32TopLeftY = frameBuffer.h + FONT_HTIGHT;
                sDispRect.u32BottonRightX = (frameBuffer.w);
                sDispRect.u32BottonRightY = (frameBuffer.h + (2 * FONT_HTIGHT) - 1);

                Display_ClearRect(C_WHITE, &sDispRect);
                Display_PutText(
                    szDisplayText,
                    strlen(szDisplayText),
                    0,
                    frameBuffer.h + FONT_HTIGHT,
                    C_BLUE,
                    C_WHITE,
                    false
                );
#endif
                u64PerfCycle = (uint64_t)pmu_get_systick_Count() + (uint64_t)(SystemCoreClock * EACH_PERF_SEC);
                u64PerfFrames = 0;
            }

            PresentInferenceResult(infFramebuf->results, labels);
            infFramebuf->eState = eFRAMEBUF_EMPTY;
        }
```

```
        if (infFramebuf)
        {
            //draw bbox and render
            /* Draw boxes. */
            DrawImageDetectionBoxes(infFramebuf->results, &infFramebuf->frameImage, labels);

            //display result image
#if defined (__USE_DISPLAY__)
            //Display image on LCD
            sDispRect.u32TopLeftX = 0;
            sDispRect.u32TopLeftY = 0;
            sDispRect.u32BottonRightX = (infFramebuf->frameImage.w - 1);
            sDispRect.u32BottonRightY = (infFramebuf->frameImage.h - 1);

#if defined(__PROFILE__)
            u64StartCycle = pmu_get_systick_Count();
#endif

            Display_FillRect((uint16_t *)infFramebuf->frameImage.data, &sDispRect);

#if defined(__PROFILE__)
            u64EndCycle = pmu_get_systick_Count();
            info("display image cycles %llu \n", (u64EndCycle - u64StartCycle));
#endif

#endif

            u64PerfFrames ++;
```

# 五、開發版上的推論程式系統流程步驟

- **(五) 模型推論及後處理解析:**
  - 獲取模型輸入張量
  - 執行推論
  - 獲取模型輸出張量
  - 使用後處理解析推論結果

```
300     TfLiteTensor *inputTensor   = model.GetInputTensor(0);
301     TfLiteTensor *outputTensor = model.GetOutputTensor(0);
302
303     if (!inputTensor->dims)
304     {
305         printf_err("Invalid input tensor dims\n");
306         vTaskDelete(nullptr);
307         return;
308     }
309     else if (inputTensor->dims->size < 3)
310     {
311         printf_err("Input tensor dimension should be >= 3\n");
312         vTaskDelete(nullptr);
313         return;
314     }
315
316     TfLiteIntArray *inputShape = model.GetInputShape(0);
317
318     const int inputImgCols = inputShape->data[arm::app::YoloXnanoNu::ms_inputColsIdx];
319     const int inputImgRows = inputShape->data[arm::app::YoloXnanoNu::ms_inputRowsIdx];
320
321     // postProcess
322     arm::app::object_detection::DetectorPostprocessing postProcess(0.6, 0.65, numClasses, 0);
```

# 五、開發版上的推論程式系統流程步驟

- **(六) 繪製偵測框與標籤文字:**
  - 從推論結果獲取相關資訊並繪製相關資訊到影像上

```
211         imlib_draw_rectangle(drawImg, result.m_x0, result.m_y0, result.m_w, result.m_h, COLOR_B5_MAX, 1, false);
```

```
212         imlib_draw_string(drawImg, result.m_x0, result.m_y0 - 16, labels[result.m_cls].c_str(), COLOR_B5_MAX, 2, 0, 0, false,
213                           false, false, false, 0, false, false);
```

# 五、開發版上的推論程式系統流程步驟

- **(七) 顯示結果至LCD顯示器:**
  - 將繪製完成的影像輸出至LCD顯示器上

```
453             //display result image
454 #if defined (__USE_DISPLAY__)
455             //Display image on LCD
456             sDispRect.u32TopLeftX = 0;
457             sDispRect.u32TopLeftY = 0;
458             sDispRect.u32BottonRightX = (infFramebuf->frameImage.w - 1);
459             sDispRect.u32BottonRightY = (infFramebuf->frameImage.h - 1);
460
461 #if defined(__PROFILE__)
462             u64StartCycle = pmu_get_systick_Count();
463 #endif
464
465             Display_FillRect((uint16_t *)infFramebuf->frameImage.data, &sDispRect);
466
467 #if defined(__PROFILE__)
468             u64EndCycle = pmu_get_systick_Count();
469             info("display image cycles %llu \n", (u64EndCycle - u64StartCycle));
470 #endif
471
472 #endif
473
474             u64PerfFrames ++;
475
476             if ((uint64_t) pmu_get_systick_Count() > u64PerfCycle)
477             {
478                 info("Total inference rate: %llu\n", u64PerfFrames / EACH_PERF_SEC);
```

# 六、結論與未來發展

- **(一) 結論:**
  - 本次跌倒偵測系統能有效利用 Roboflow 資料集結合 YOLOX-Nano 模型，在新唐科技 M55M1 開發板上創造出更快速且節省人力資源的方式來偵測跌倒達到提高醫療品質的效果。透過 Full-INT8 模型量化技術和 Vela 編譯器的最佳化，來提升了模型的推論速度和準確率，在測試中達到高達 90%以上的偵測準確率。
- **(二) 未來發展:**
  - 本次的開發方式僅透過最簡單的攝像鏡頭來捕捉影像，未來可以配合更多功能的鏡頭(如3D攝影機)或使用Keypoint-Detection的做法來增加姿勢的種類與辨識率，做到更專業及精確的應用。

# DEMO影片

- Link

Joy of innovation

NUVOTON

谢谢

謝謝

Děkuji

Bedankt

Thank you

Kiitos

Merci

Danke

Grazie

ありがとう

감사합니다

Dziękujemy

Obrigado

Спасибо

Gracias

Teşekkür ederim

Cảm ơn