FPGA: 深度學習應用於低光環境之自駕車輔助系統

作者：陳慶瑾、陳欽安、詹書愷

指導教授/實驗負責教授：楊家驤

摘要(Abstract)：

曾經在新聞看到一起 UBER 自駕車交通事故，該事故的發生背景是在低光環境下，系統在撞擊前 1.3 秒才對駕駛人提出示警，因為駕駛人在如此短的時間內反應不及，最後釀成死亡車禍之悲劇。在 AI 技術蓬勃發展以及晶片運算量能不斷突破的今日，我們希望透過能夠在低光環境即時辨識的演算法，搭配 FPGA 硬體加速的方式，來達成低光環境自家車輔助系統之設計。

一、簡介(Introduction)

以 Enlighten GAN 結合 tiny-YOLOv3 來形成運算流程，並透過 OpenVINO Toolkit 將設計好的運算流程轉置成.IR 檔，最後放到 FPGA(Openvino starter kit)執行影像增強與辨識的計算，出現危及駕駛人安全時給予警示。

二、分項概述

1. OpenVINO Toolkits

   為 Intel's 支援 AI 以及 Computer Vision 的套件。在本實驗中主要運用當中的 Deep Learning Deployment Toolkit:

   a. 推論引擎 (Inference Engine)：可以支援選擇 CPU, GPU, VPU, FPGA 作為深度神經網路的運算引擎。

   b. 模型優化器 (Model Optimizer)：將 Tensorflow, Onnx 等神經網路的模型轉成 Intermediate Representation (IR)檔，供推論引擎做推論。

2. Learning to See in the Dark Model

a. 論文目的：在極低光影像修復處理中，由於低光影像的低光子、低 SNR 值，會使短曝光影像中會有嚴重的噪點問題；使在長曝光影像中會有影像模糊的問題。因此本篇論文提出了引入了低光數據資料集，以短曝光影像作為輸入影像，並以長曝光影像作為參考標的訓練端對端神經網路來處理低光影像修復。

b. See-in-the-Dark 數據資料集

資料集中具有長短曝光、室內室外的影像：在室外的影像照度分佈範圍為 0.2 到 0.5(勒克斯)；室內的影像照度分佈範圍為 0.03 到 0.3(勒克斯)。

此外，由於長曝光影像的蒐集需要長時間的曝光，因此此資料集內的圖像均為靜態。

- 5094 張短曝光影像
    - 作為神經訓練網路的輸入
    - 曝光時間分佈於 1/30 到 1/10 秒

- 424 張長曝光影像
    - 作為神經訓練網路的參考標的
    - 曝光時間為輸入影像的 100 到 300 倍：分佈於 10 到 30 秒

- 資料集配置

本實驗採取 SID 數據集中由 Sony a7SII 所蒐集的 2697 張短曝光以及 424 張長曝光影像作為訓練資料集（Training Dataset）。

Sony α7S II	Filter array	Exposure time (s)	# images
x300	Bayer	1/10, 1/30	1190
x250	Bayer	1/25	699
x100	Bayer	1/10	808

• 驗證資料集 (Validation Dataset)：

SID 中隨機取百分之十的影像作為驗證資料集，並從訓練資料集中`移出。

• 測試資料集 (Testing Dataset)：

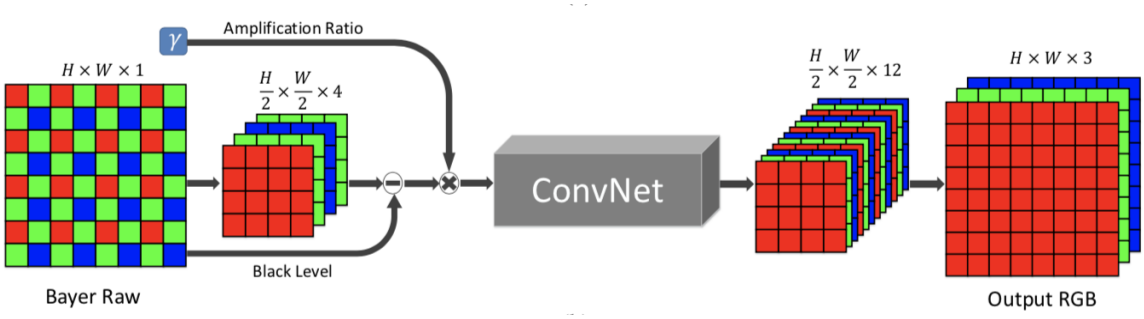
從剩下 SID 中隨機取百分之 20 的影像作為測試資料集，並從訓練資料集中移出

• 訓練資料集 (Training Dataset)：

SID 資料集中屏除驗證以及測試資料集後，就是訓練用的短曝光訓練資料集

• 低光影像訓練流水線

本實驗以 Bayer raw 格式作為輸入影像，訓練資料流如下：



- 將拜爾影像各個濾色鏡的資料分類蒐集成四個通道，同時也將空間解析度下降 2 倍
- 扣除黑階影像數值 （黑階影像：本實驗中，就是 Sony as7II 在全黑情況下拍攝照片的拜爾影像色值)
- 乘上放大係數：可調為 100x, 250x, 300x，作用為將輸入影像與參考標的影像的像素色值強度相當。
- 經過 U-Net 得到 12 個通道的色值張量，再經過子像素卷積層(sub-pixel layer)將影像的解析度還原得到最終輸出。

- 由於輸入影像為 Bayer raw，而輸出影像為 sRGB 形式
- 訓練超參數設置：
  - 損失函數： L1 loss
  - 優化器：Adam optimizer
  - 學習率：10e-4, 在 2000 epochs 後更新為 10e-5
  - Epochs: 4000 epochs

3. Enlighten GAN

EnlightenGAN 的設計，使其可以使用非配對的資料(暗圖片不用對應到亮圖片)，進行訓練；相反的，Discriminator 只需判斷圖片是暗圖片從 Generator 增強的圖片(fake)，或是本來就是亮的圖片(real)。暗圖片作為 Generator 的 input，生成亮圖片(fake)後與真正的亮圖片(real)進行對抗生成訓練。EnlightenGAN 對於原本 GAN 的架構做了幾點優化：

a. Global / Local Discriminator

若直接對 real 跟 fake 的整張圖片(global)進行 adversarial loss training，很多 local 的光線變化是無法捕捉到的。故 EnlightenGAN 除了 global discriminator 以外，另外加了 local 的 discriminator，隨機從 real 跟 fake 中取一些 patch 進行 adversarial loss 的訓練。

b. Self Feature-Preserving Loss

原本的暗圖片與 Generator 對應生成的亮圖片，在 feature 上應保存相似之處，故 EnlightenGAN 將 generator 的 input 與 output 取 VGG feature 計算相似程度，也加入 loss 進行訓練。

c. Self-Regularized Attention

在增強暗圖片時，我們會希望 Generator 將焦點放在較為黑暗的部分，而本來已經明亮的區域則不需去增強。EnlightenGAN 將暗圖片 Illumination

Map(I)抓出，並將(1 - I)作為 Attention Map，以引導 Generator 生成明亮圖片。

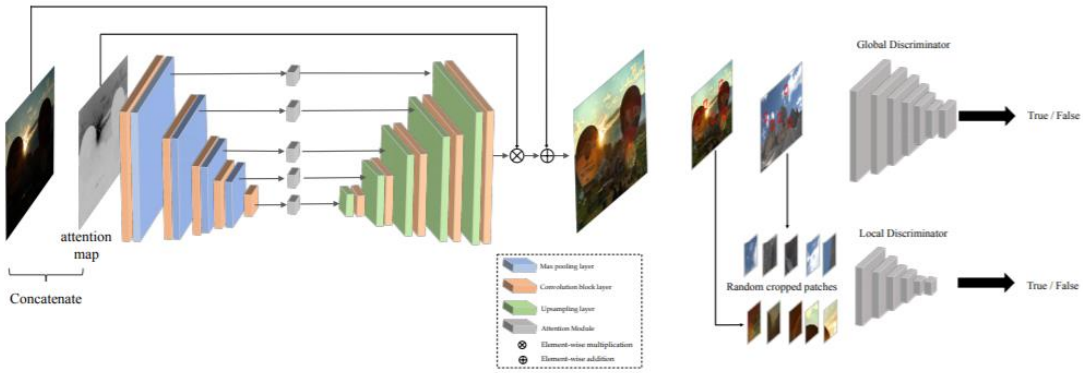


Figure 2: The overall architecture of EnlightenGAN. In the generator, each convolutional block consists of two 3 × 3 convolutional layers followed by batch normalization and LeakyRelu. Each attention module has the feature map multiply with a (resized) attention map.

4. tiny-YOLOv3

YOLO 為物件辨識中知名的模型，其在不同的尺度下檢視 bounding box 可能出現的位置與長寬，其獨特的架構可以減少預測的參數，以更快的運算速度得到更高的準確率。考慮 FPGA 的負荷，我們使用 tiny-YOLOv3，節省許多特徵層的抽取，犧牲些微的正確率，以獲得更低的記憶體用量與更短的運算時間。

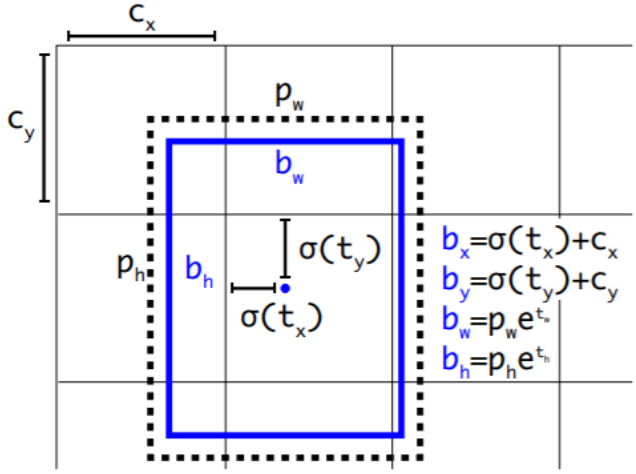


Figure 2. **Bounding boxes with dimension priors and location prediction.** We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function. This figure blatantly self-plagiarized from [15].

5. Dataflow

依據輸入影像的性質分為兩個資料處理步驟進行

a. 輸入影像為拜爾圖像

將輸入影像經由 Learning to See in the Dark 模型，得出增亮圖像；再經過 Yolo-v3 模型進行物件辨識。

b. 輸入影像為 sRGB 圖像（.jpg）

將輸入影像經由 Enlighter Gan 模型，得出增量圖像；在經過 Yolo-v3 模型進行物件辨識。

三、 實驗（或理論）(Experimental or Theoretical)

1. PyTorch to Onnx

由於 OpenVINO Toolkits 目前不支援 PyTorch 模型直接轉換為 Intermediate Result 檔，因此需要先將我們的 PyTorch 模型轉為 Onnx 模型，才能運用 OpenVINO Toolkits 中的 Model Optimizer 進行 IR 轉換。

2. Onnx to IR

經過 OpenVINO Toolkits 的 Model Optimizer，可以將我們的 Onnx 模型，轉換成.xml 檔以及.bin 檔：

a. xml：紀錄神經網路的架構，例如輸入、輸出維度、神經網路層種類、激活函數等

b. bin：紀錄神經網路各層的權重與偏差值

3. IR inference result

通過 OpenVINO Toolkits，我們可以將上步驟得出的 IR 檔案經過各種推論引擎(CPU, GPU, VPU, FPGA 等) 進行推論。

四、結果與討論(Result and Discussion)

1. Learning to See in the Dark

原圖：

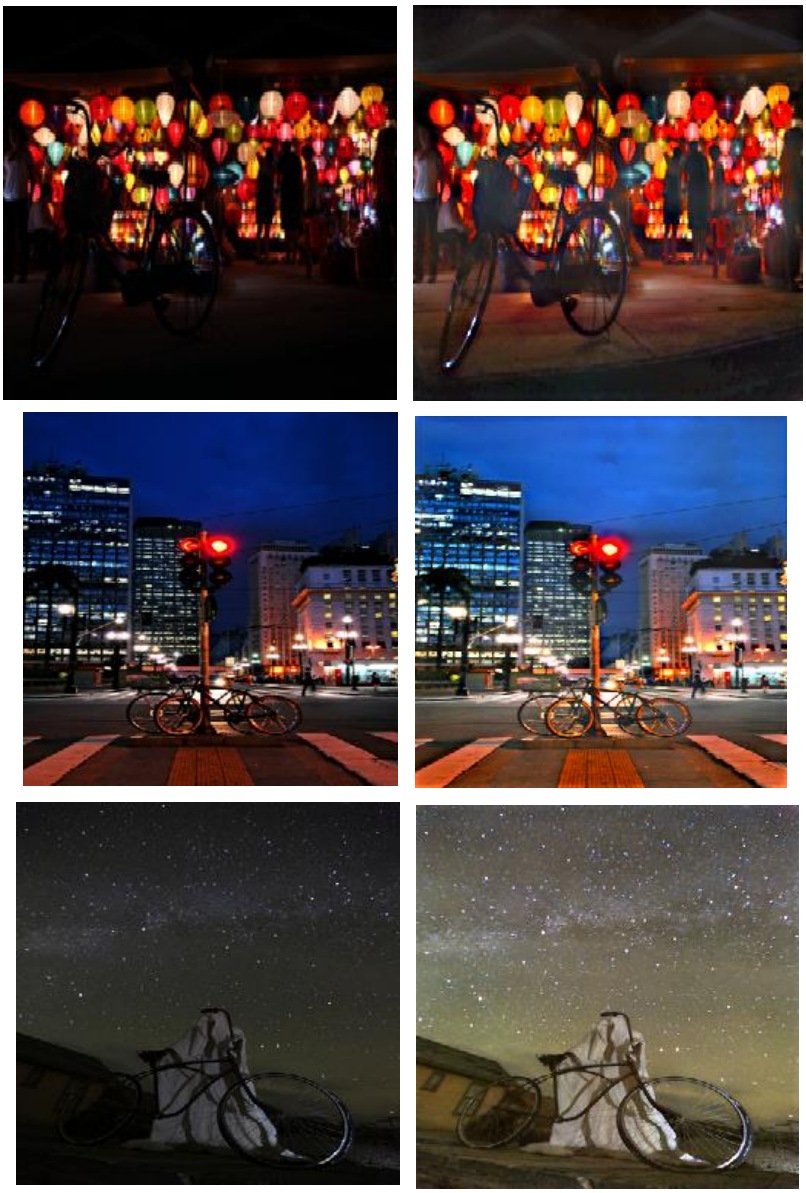
PyTorch 輸出：

IR (運用 CPU 作為推論引擎)：

可以看出經過 model optimizer 後的模型與原先模型的推論能力相當。

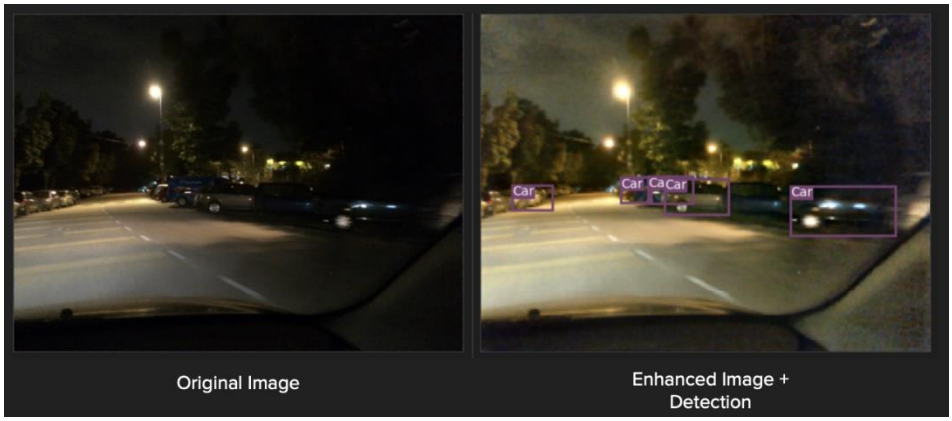
2. Enlighten GAN

輸入暗圖片：　　輸出增亮圖片：



輸入暗光圖像經過 Enlighten GAN 模型後，有效增量圖像。

3. YOLOv3 物件辨識應用於 Enlighten GAN



由上推論結果可以看出原始圖片經過 Enlighten GAN 增亮後，Yolo 便能更準確的進行圖片中的物件辨識。

五、結論(Conclusion)

未經過處理的昏暗環境照片，在 YOLO 架構底下不容易偵測出照片中的物件，即使偵測出物件，辨識率也在 10%之下。但經過適當的流程安排與處理，我們可以將辨識率提升至 40%，約莫是正常環境光線照片在 YOLO 架構下的辨識效率的 50%，大大的提升低光物件辨識率。畢竟我們最終目標是希望透過即早辨識、即早警示的系統來降低車禍率，因此未來會考慮將辨識物件的 dataset 改成辨識道路阻礙物件，來降低辨識種類以增加危險因子的辨識率，並且透過 Openvino Starter kit 來進行加速，以達成即時之效果。希望最後的成果能為自駕車的未來以及道路的安全提升作出貢獻。

六、參考文獻(Reference)

1. Learning to See in the Dark *Chen Chen*
2. EnlightenGAN: Deep Light Enhancement without Paired Supervision
3. Openvino starter kit user manual