

# NYCU IEE Deep Learning

## Lab 2 Report: Crowd Counting

陳柏翔 313510156

### 1 Introduction

本作業旨在實現人數計數的影像處理任務，使用的資料集為 UCSD Pedestrian Dataset [1]，要在一個半俯視人行道的畫面中分別偵測出向左走、向右走、以及總共的人數。

我在本次作業中，於 Task 1 使用題目要求的 Compressed Dense Net (CDenseNet) [2] 以直接對人數進行回歸估計 (Regression) 的方式達到 MAE 約小於 2.0 的準確率；而在 Task 2 中，使用小幅度修改的 CDenseNet 模型達到優於 Task 1 的準確率，且 MACs 運算量是 Task 1 的 0.27 倍，提高了模型的訓練與推論速度。實作成果截圖詳見 4.1 章節。

此外，我在 Task 2 中仍然嘗試了類似於 Density Map Estimation 的方法，也能夠在 Test Set 上達到 MAE 為 1.55 的準確率，但是我認為此結果非常不穩定，因此不當作 Task 2 的最終作法，而是額外做討論。

### 2 Dataset

#### 2.1 UCSD Pedestrian Dataset

在 UCSD Pedestrian Dataset [1] 中，如同 Figure 1 所示，每張圖片為  $158 \times 238$  像素的灰階圖片，而我們所拿到的 Train / Val / Test Sets 數量分別有 2,500 / 700 / 200 張圖片。



Figure 1: Example Images in UCSD Pedestrian Dataset

#### 2.2 Difficulties

每張圖片都會對應到 3 個標籤值，分別是向左 (L)、向右 (R)、總共 (T) 的人數，除此之外沒有更多對於圖中人物的詳細資訊 (e.g. 人物位置、方向)，而我認為這是最大的一個困難點，因為缺少人物位置就難以用 Density Map Estimation 的方式來實作，除非借用其他 Pre-trained 模型來偵測人物位置，否則只能用直接對三個數值進行回歸的方法，在一張複雜影像中直接估計出人數。然而除此原因以外，還有很大一部分的困難是出自 Dataset 本身，主要有以下兩點：

##### 2.2.1 Imperfect Labeling

如同 Figure 2 所示，左圖為 YOLOv8 偵測到類別為 person 的物件，總計 25 個人；右圖為 Label (Ground Truth) 的 T (Total Count) 項，只計到了 23 個人。然而實際上圖片中的總人數超過 25 個

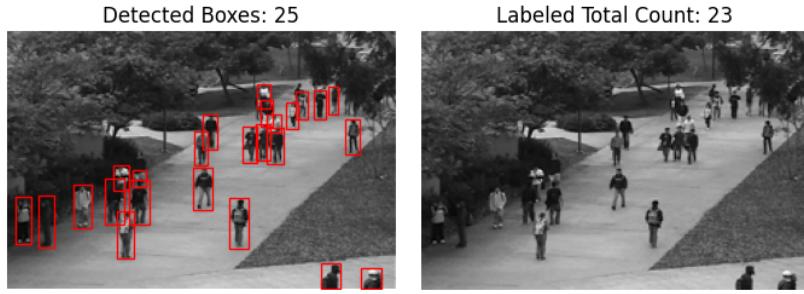


Figure 2: Imperfect Labeling.

人，以此幀畫面而言，YOLOv8 [3] 的偵測結果甚至較 Ground Truth 更為正確。

藉此我發現資料集中有多張圖片都有相同情況，假若模型能夠很精準的找出圖中每個人物，也會因為此資料集中的 Ground Truth 較真實人物數量來得更少，而無法正確判斷是要依照哪些人物來計算向左或是向右的人數。

### 2.2.2 Limitation of Single-Frame Input

此資料集其實是由連續的數幀畫面所構成的影片，因此原本是可以藉由連續的前後幀畫面，分析其中改變的像素值來判斷出人物位置以及人物的移動方向，但是在此次任務中，一筆輸入僅含一張影像就需要估計出人數與方向，我認為這是很大的限制。如果要更好的估計出人數與方向，就需要讓一筆輸入包含複數張連續幀畫面（此處並非指訓練時的 Batch 設置為多張連續幀圖片，而是一筆輸入本身就是一個短片，包含了複數張連續幀畫面）。

## 3 Architecture

### 3.1 Task 1 Network

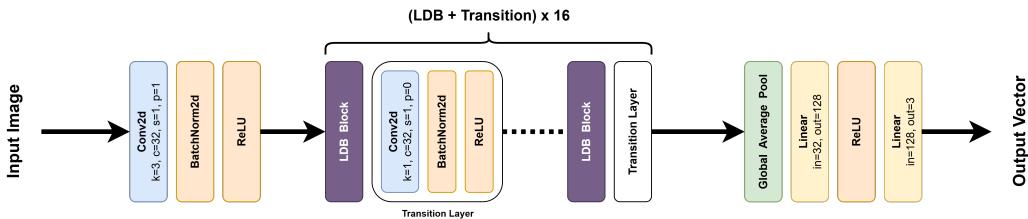


Figure 3: CDenseNet [2] with 16 LDB Blocks

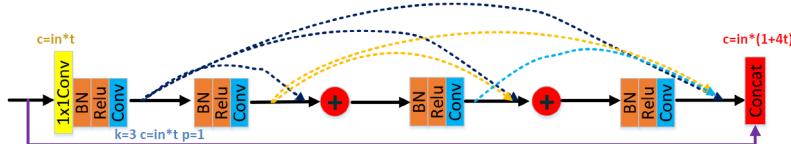


Figure 4: Lightweight Dense Block (LDB)

根據題目要求，在此作業 Task 1 中所使用的模型如 Figure 3 所示，其中的 LDB Block 為 Figure 4，實作程式碼詳見 `CDenseNet.py` 檔案。

### 3.2 Task 2 Network

在 Task 2 中，我測試過非常多種方法來提高準確率，卻都沒有明顯提升效果，因此最終回頭採用與 Task 1 相同的模型來做修改，模型如 Figure 5 所示。

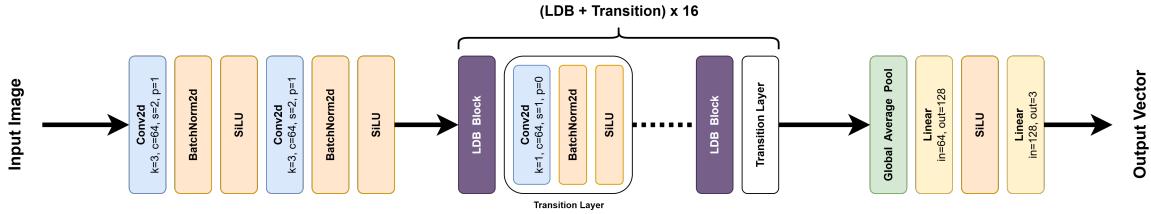


Figure 5: Modified CDenseNet for Task 2

首先，我將模型中所有的 Activation Functions 都替換成了 SiLU (Swish) Function，採用的原因與我在 Lab 1 中使用的理由相同，最主要還是為了讓特徵值為負的區域也能夠有梯度反向傳播。

此外，我將 Kernels (Channels) 數量從原本的 32 設置改為 64，以學習出更多複雜的特徵，然後我發現此任務並不需要那麼龐大的特徵圖也能夠有很好、甚至更好的成果，因此我在模型的輸入 Convolution 層改為兩次 Stride 設為 2 的 Convolutions。

### 3.3 Models Comparison

Table 1 展示對 Task 1 與 Task 2 模型所做的簡單比較，最右邊一欄的 Better One 顯示的是哪個模型比較輕量 (跑得更快、運用較少的資源) 或者更精準。

	Task 1	Task 2	Better One
Total Params	214,659	875,459	Task 1
GMACs	7.78	2.07	Task 2
Total Size (MB)	1021.43	143.53	Task 2
Val MAE	2.207	1.977	Task 2
Test MAE	1.519	1.401	Task 2

Table 1: Comparison between Task 1 and Task 2 Models

對於 Task 2 的模型，因為 Kernels 數量增加而使得參數量從 214,659 提升為 875,459，但是也因此提高了準確率，使 MAE 下降了一些。此外，由於在一開始就對輸入做了兩次 Stride 為 2 的 Convolutions，使得特徵圖的維度下降了非常多，MACs (Mult-adds) 運算量就從原本的 7.78 G 下降至 2.07 G，而總記憶體消耗估算量也從原本的 1021.43 MB 下降為 143.53 MB。

總體來說，Task 2 的模型雖然增加了參數量，但執行起來比 Task 1 快非常多、佔用空間也少非常多，同時還更加精準。

## 4 Experiments

此章節將分別討論我在 Task 1 與 Task 2，還有我使用類似於 Density Map Estimation 的方法實作 Task 2 所得到的成果。

### 4.1 Implementation Results

#### 4.1.1 Task 1 Results

在 Task 1 中，我使用 Smooth L1 作為 Loss Function，並且用 SGD Optimizer 搭配 Learning Rate Scheduler 以及 Gradient Clipping 進行訓練，Epoch 數設置為 30。結果如 Figure 6, 7, 8 所示。

```
[Epoch 26] Train Loss: 0.6267 | MAE[r,l,t] = [ 1.7958, 2.7988, 2.1317] | Avg MAE: 2.2421
          Val Loss: 1.7869 | RMSE[r,l,t] = [ 2.1417, 3.3848, 2.5756] | Avg RMSE: 2.7007
[Epoch 27] Train Loss: 0.6842 | MAE[r,l,t] = [ 4.0603, 2.8597, 3.1086] | Avg MAE: 3.3429
          Val Loss: 2.8673 | RMSE[r,l,t] = [ 4.8194, 3.4762, 3.6501] | Avg RMSE: 3.9819
model saved
[Epoch 28] Train Loss: 0.6920 | MAE[r,l,t] = [ 1.9458, 2.6383, 2.0369] | Avg MAE: 2.2070
          Val Loss: 1.7566 | RMSE[r,l,t] = [ 2.3378, 3.3779, 2.4524] | Avg RMSE: 2.7227
```

Figure 6: Task 1 Screenshot (Validation MAE 2.21)

```
[INFO] Loaded model state dict from checkpoint: model.pth
[TEST] MAE[r,l,t] = [1.4874, 2.0497, 1.0197] | Avg MAE = 1.5189
[TEST] RMSE[r,l,t] = [1.8049, 2.2815, 1.2689] | Avg RMSE = 1.7851
```

Figure 7: Task 1 Screenshot (Test MAE 1.52)

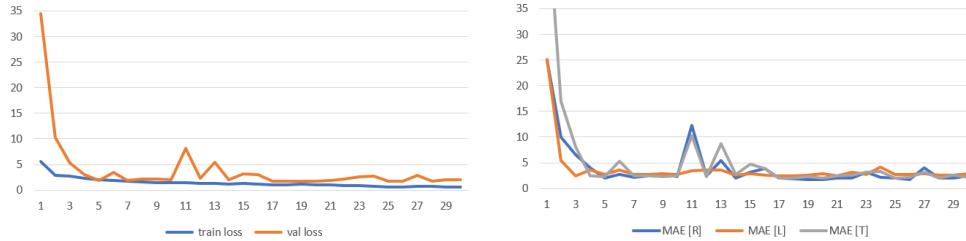


Figure 8: Task 1 (Train/Val) Loss and (Val) Accuracy during Training Process

#### 4.1.2 Task 2 Results

在 Task 2 中，訓練的設置都與 Task 1 相同，只有 Epoch 數設置改為 40。得到的結果如 Figure 9, 10, 11 所示。

```
[Epoch 29] Train Loss: 0.4578 | MAE[r,l,t] = [ 1.9250, 2.4740, 1.6204] | Avg MAE: 2.0065
          Val Loss: 1.5516 | RMSE[r,l,t] = [ 2.2271, 3.0382, 1.9908] | Avg RMSE: 2.4187
[Epoch 30] Train Loss: 0.4008 | MAE[r,l,t] = [ 1.9479, 2.4981, 1.8398] | Avg MAE: 2.0952
          Val Loss: 1.6382 | RMSE[r,l,t] = [ 2.2587, 3.0272, 2.2305] | Avg RMSE: 2.5055
model saved
[Epoch 31] Train Loss: 0.4112 | MAE[r,l,t] = [ 1.6743, 2.3989, 1.8567] | Avg MAE: 1.9766
          Val Loss: 1.5371 | RMSE[r,l,t] = [ 2.1674, 3.0040, 2.2976] | Avg RMSE: 2.4897
```

Figure 9: Task 2 Screenshot (Validation MAE 1.98)

```
[INFO] Loaded model state dict from checkpoint: task2_model.pth
[TEST] MAE[r,l,t] = [1.0376, 1.8761, 1.2882] | Avg MAE = 1.4007
[TEST] RMSE[r,l,t] = [1.3227, 2.3184, 1.6862] | Avg RMSE = 1.7758
```

Figure 10: Task 2 Screenshot (Test MAE 1.40)

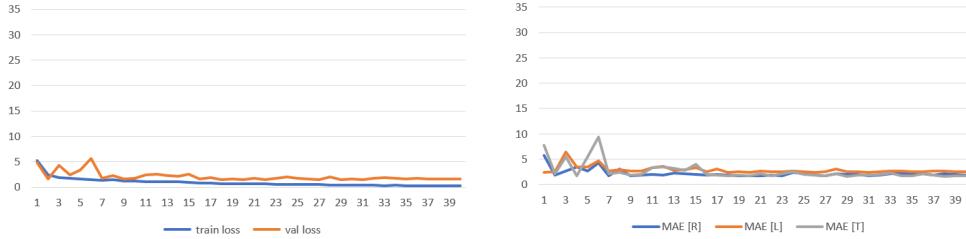


Figure 11: Task 2 (Train/Val) Loss and (Val) Accuracy during Training Process

## 4.2 Density Map Based Method

除了以上在 Task 1, 2 中直接估計出人數的方法以外，在 Crowd Counting 的任務中，有一類常見的做法稱為 Density Map Estimation，細節將於後續章節 5 中討論，而我為了讓模型更好的辨識出影像中的每個人物，我嘗試以這種方式來進行預測，雖然結果不如我在 Task 2 的作法，但我認為仍然非常值得討論。

**Prediction.** 為了分別估計出向左、向右、總共的人數，我將輸出 Heatmaps (在此即代表 Density Maps) 設置為三個 Channels，並期望這三個 Channels 的 Heatmaps 分別辨識出影像中向左、向右、總共的人物。因為一個像素不會存在超過一個完整的人、也不會有負的人數，所以我在輸出 Heatmaps 的最後一層加上 Sigmoid 函數，使得輸出的每個像素值都在  $[0, 1]$  範圍內，而 Heatmaps 上的像素數值就代表了該像素上有多少人。

**Training.** 訓練時，由於我們沒有人物位置的 Ground Truth 資訊，我只能選擇將模型輸出的 Heatmaps 直接加總得到的數值作為估計人數，所以三個 Channels 各自加總就會得到估計的向左、向右、總共人數，也就是如同 Task 1 的輸出三維向量 (三個數值)。

**Model.** 因為輸出的 Heatmaps 是對應到原本輸入影像的密度分布圖，解析度較高，所以為了精確的偵測出每個人物，就需要使用能夠融合不同尺度特徵圖的模型，在此我選擇使用 UNet [4] 作為 Density Map Estimation 的模型。而在最初我嘗試讓輸出的 Heatmaps 解析度與原輸入影像相同，但後來發現我們不需要如此精確的人物位置，所以後面加了兩個Stride為2的Convolution層降低特徵圖解析度。

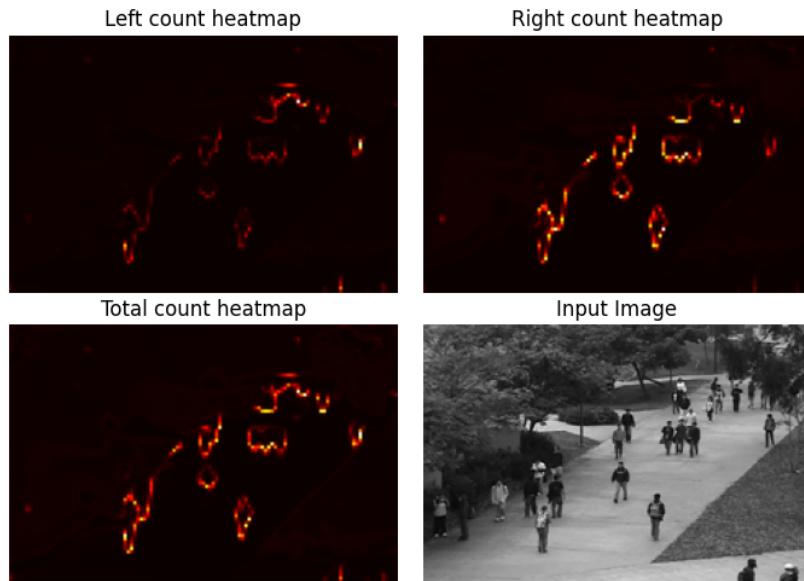


Figure 12: Predicted Heatmaps

藉由這樣的訓練方式所得到的成果如 Figure 12 所示。從輸出結果來看，能夠發現即使只透過人數的資訊進行訓練，也能夠讓模型學習並辨識出影像中的主要物體在哪裡，但是模型只能夠偵測出人物的邊緣，而不是理想中的以人物為中心分布一個 Gaussian Kernel 在周圍。

而除了只能找到邊緣的這個問題之外，捕捉到人的準確率也很低，圖中許多有人的地方偵測出來卻沒有任何亮點，表示模型估計附近沒有任何人，但這也有一部分是如同前面章節 2 所討論的，出自 Dataset 本身的問題。此外，在我多次的實驗結果中，我發現在 Validation Set 上僅有總人數估計能到 MAE 為 2.0 以下，而向左與向右的誤差卻永遠大於 2，我認為這一方面是因為沒有向左與向右人物的 Ground Truth 資訊，難以判斷誰是向左還有誰是向右；另一方面是因為對於每個人物的解析度其實並不高，衣著相關的變化因素又太多，難以學習並利用到人臉這類的細部特徵來判斷朝向。

```
[INFO] Loaded model state dict from checkpoint: task2_model.pth
[TEST] MAE[r,l,t] = [1.4709, 2.0042, 1.1864] | Avg MAE = 1.5538
[TEST] RMSE[r,l,t] = [1.7456, 2.3306, 1.5291] | Avg RMSE = 1.8685
```

Figure 13: Test Result using Density Map Based Method

即使在訓練過程中不如 Task 2 的結果，但是仍然在 Test Set 上得到了不錯的準確率，如 Figure 13 所示。我另外還有嘗試以 YOLOv8 [3] 或者 BASNet [5] 來輔助生成 Ground Truth Heatmaps 進行訓練 (在偵測出的人物位置上放置 Gaussian Kernels)，但是訓練結果都較以上來得差。

## 5 Required Discussion

此章節討論題目要求回答的問題，包含 MAE、RMSE 的差異以及 Density Map Estimation 的說明。

### 5.1 Characteristics of MAE and RMSE

關於 MAE (Mean Absolute Error) 與 RMSE (Root Mean Square Error) 這兩種評估誤差方法的特性差異以及使用時機，我決定以表格的方式呈現，如 Table 2 所示。

	Mean Absolute Error	Root Mean Square Error
<b>Formula</b>	$\frac{1}{n} \sum_{i=1}^n  y_i - \hat{y}_i $	$\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$
<b>Sensitivity</b>	線性 (Linear) 考量所有誤差項	平方 (Quadratic) 考量所有誤差項
<b>Robustness to outliers</b>	較不容易受到離群值 (即大誤差項) 影響，因此較為堅固穩定。	容易受到離群值影響，因此較不堅固穩定。
<b>Optimization</b>	作為 Loss Function 會無法在 0 點微分	作為 Loss Function 在任一處皆可微分
<b>Preferred timing</b>	離群值存在，但不希望受到離群值影響我們對平均誤差的評估。	希望考量離群值，或者希望對較大的誤差項有更大的懲罰。

Table 2: Comparison between MAE and RMSE

另外，在章節 6 中也有討論到一部分關於 MAE 與 MSE 作為 Loss Function 可能會發生的問題，以及我後來選擇的 Loss Function。

## 5.2 Density Map Estimation

在 Density Map Estimation 的方法中，不同於我在 Task 1, 2 中以回歸 (Regression) 的方式直接估計出三個人數數值，而是讓模型對輸入影像產生一張「Density Map (密度圖)」，再透過 Density Map 來計算出總人數。

而 Density Map 上的每一個像素值就代表在該位置上的人數密度。例如在一個人頭的位置，放置一個以該點為中心的 Gaussian Kernel，使得積分後（對整張密度圖做加總）就等於該圖中實際的人數。這樣的方式不僅能輸出總人數，還能提供空間分布的資訊，即告訴我們人群主要聚集在影像中的哪裡。

使用 Density Map Estimation 的方法時，用來衡量輸出好壞的方法 (Metric) 通常可以用 MAE 也可以用 RMSE 來評估，但是如果想要作為優化模型輸出的 Loss Function，我會選用 MSE Loss 或是 Binary Cross Entropy Loss 來計算 Density Map 的 Loss。

## 6 Accuracy Improvement

### 6.1 Loss Function

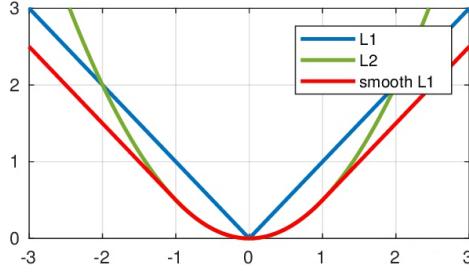


Figure 14: Loss Functions

如 Figure 14 所示，使用 L1 Loss 的話即使誤差很小，更新的梯度 (斜率) 也不會變小，可能會使收斂幅度過大而難以收斂；而使用 L2 Loss (i.e. MSE Loss) 的話則是在誤差稍大的區域都可能得到太大的梯度，同樣因為更新的幅度太大而難以收斂。因此我改用 Smooth L1 Loss 作為模型優化的 Loss Function。

### 6.2 Optimizer

在更新模型參數時，我選用 SGD Optimizer 來進行優化，並搭配 Learning Rate Scheduler 調整更新幅度，參數設置如下：

- Learning Rate: 0.01
- Number of Epochs: 30 (Task 1) or 40 (Task 2)
- SGD Momentum: 0.9
- SGD Weight Decay:  $10^{-5}$
- Scheduler Step Size: 30
- Scheduler Gamma Factor: 0.9

以上的設置我發現比起使用 Adam Optimizer 來得更好，此外，我也更改了儲存模型的條件限制，不一定要是 Validation MAE 最小，如果比最小的 Validation MAE 還大 0.1 以內的範圍都會儲存模型。

### 6.3 Gradient Clipping

由於訓練時總是很不穩定 (Validation MAE 起伏不定)，我認為這是因為梯度太大導致的收斂不穩定，因此我使用了 `torch.nn.utils.clip_grad_norm_()` 函式，並將 `max_norm` 設置為 2.0，來限制住過大的梯度大小。經過多次實驗，發現這樣確實有讓模型更容易收斂，也更容易得到較好的結果。

## References

- [1] Antoni B. Chan and Nuno Vasconcelos. Counting people with low-level features and bayesian regression. *IEEE Transactions on Image Processing*, 21(4):2160–2177, 2012.
- [2] Zhao Zhang, Zemin Tang, Yang Wang, Haijun Zhang, Shuicheng Yan, and Meng Wang. Compressed densenet for lightweight character recognition, 2020.
- [3] Ultralytics. YOLOv8: The ultimate guide to object detection and beyond. <https://docs.ultralytics.com/models/yolov8/>, 2023.
- [4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.
- [5] Xuebin Qin, Deng-Ping Fan, Chenyang Huang, Cyril Diagne, Zichen Zhang, Adrià Cabeza Sant'Anna, Albert Suàrez, Martin Jagersand, and Ling Shao. Boundary-aware segmentation network for mobile and web applications, 2021.

# A Test Results

## A.1 Task 1 Samples

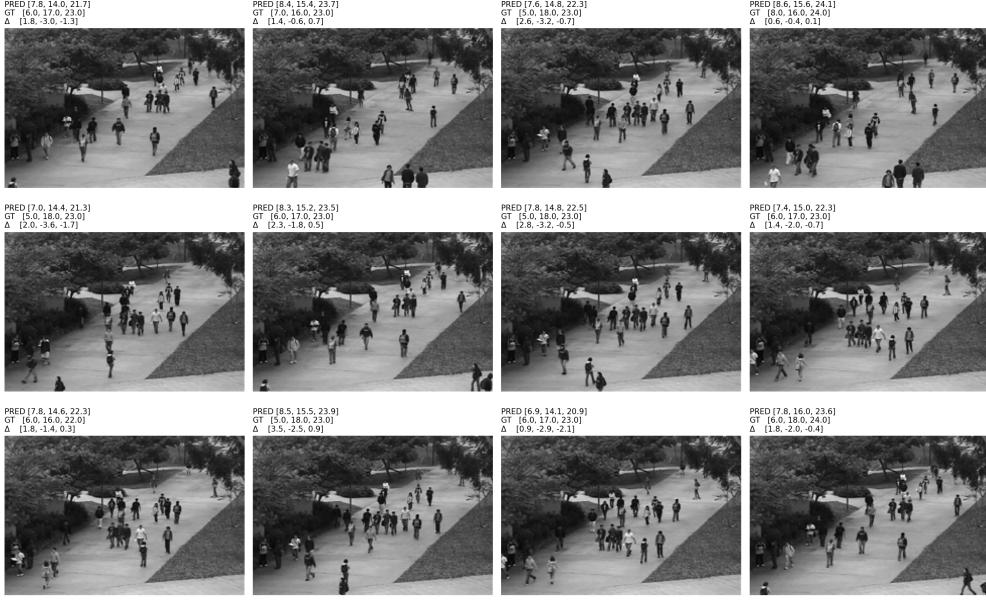


Figure 15: Test Results in Task 1

## A.2 Task 2 Samples

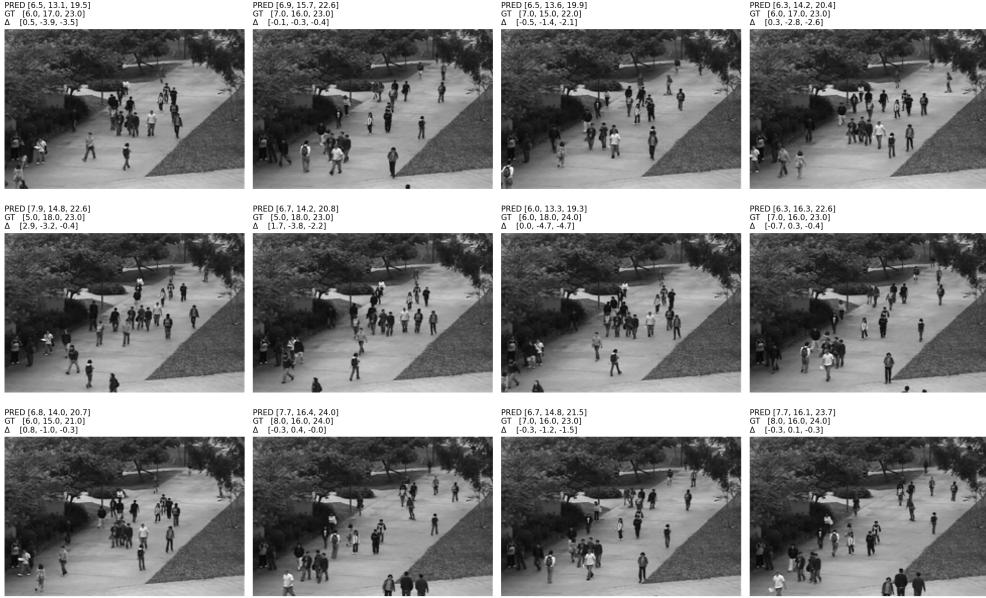


Figure 16: Test Results in Task 2