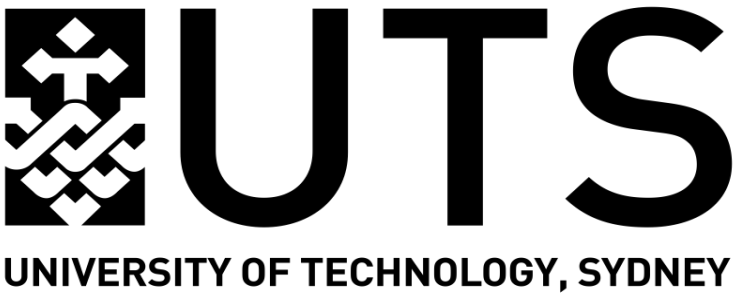
42028: Deep Learning and Convolutional Neural Network

Assignment -2



Student Name: Saumya Bhutani

1. Introduction

This report provides a comprehensive overview of image classification and object detection experiments.

For Image Classification, the chosen architecture is ResNet50. ResNet50 is a deep convolutional neural network known for its deep structure and residual learning framework, effectively addressing the vanishing gradient problem commonly encountered in training deep networks.

The initial experiment involves implementing a baseline model utilising the ResNet50 architecture. This model is initialised with ImageNet weights and trained end-to-end to classify 20 classes.

Further, ResNet50 architecture was customised to better align with the dataset requirements. This involved fine-tuning the architecture by eliminating certain convolutional layers and utilising regularisation techniques to enhance performance. By initialising the model with ImageNet weights and training it without freezing any layers, the model's learned features were adapted to the specific characteristics of the dataset.

For Object Detection, the experiments focused on implementing the below two methodologies:

1. **Faster R-CNN** is a two-stage object detection framework distinguished by its Region Proposal Network (RPN), which efficiently generates region proposals for potential object locations. This methodology involves generating region proposals in the first stage and performing object classification and bounding box refinement using regression techniques in the second stage. The two-stage approach enables accurate localisation and classification of objects within images.

2. **YOLOv5** is a single-shot object detection model known for its real-time performance and simplicity. This architecture is well-suited for scenarios requiring rapid inference times without compromising detection accuracy. YOLOv5 directly predicts bounding boxes and class probabilities from a single convolutional neural network, facilitating efficient detection of objects, including instances of solar panel damage in thermal images.

3. Dataset

Image Classification Dataset

The dataset consists of 20 classes representing various bird and dog breeds. The dataset is divided into training, validation, and testing sets in a ratio of 70:15:15. This split ensures adequate data for model training, validation to tune hyperparameters and unbiased evaluation of unseen data.

Below is the sample image from each of the 20 classes from the training set.

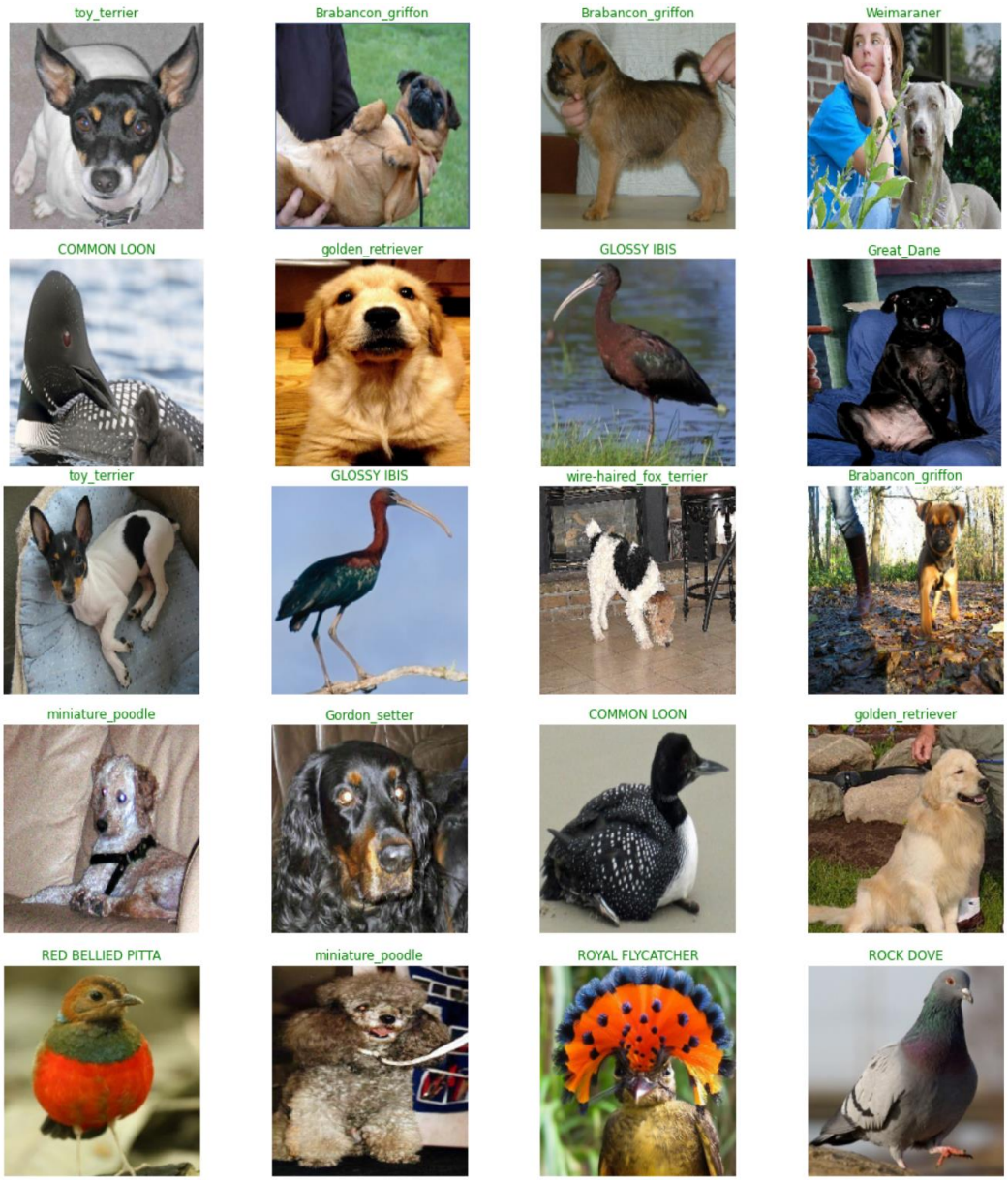


Fig.1 Image Classification Sample Images

Object Detection Dataset

The dataset consists of thermal images depicting solar panel damage. It has five classes: ' Cell', 'Cell-Multi', 'No-Anomaly', 'Shadowing', and 'Unclassified'. The dataset is pre-divided into training, validation, and testing sets, comprising 1167, 250, and 250 images, respectively, in the ratio of 70:15:15.

Below are the sample images from the training set.

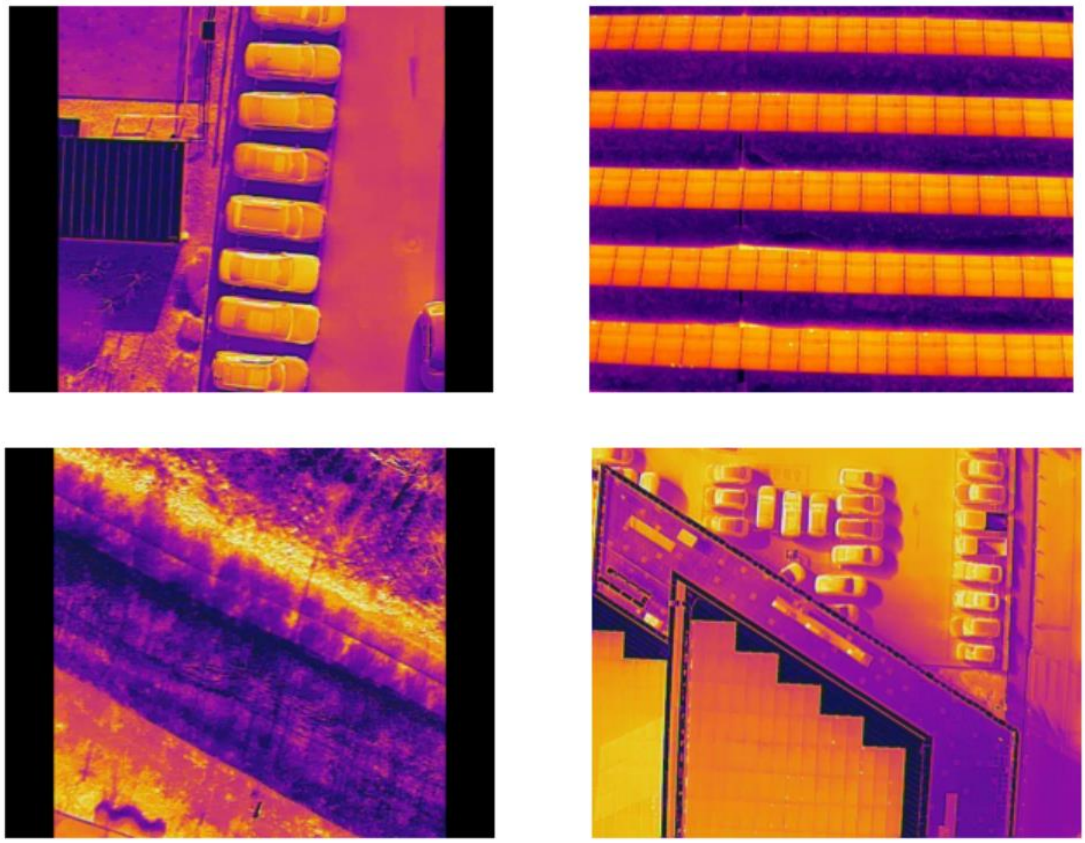


Fig.2 Object Detection Sample Images

4. Proposed CNN Architecture for Image Classification

4.1 Baseline architecture used

Component	Description
Baseline Architecture	ResNet50
Explanation	It consists of 5 stages with multiple blocks in each stage. Each block includes convolutional layers, batch normalisation and ReLU activation functions. Max-Pooling layers are used to reduce spatial dimensions.
Weight Initialization	The model is initialised with weights pre-trained on ImageNet for better generalisation.
Head for classification	Global Average Pooling converts the 3D feature maps to a 1D vector, capturing the global spatial information. A dense layer with 1024 units with ReLU activation is used to capture complex patterns in the data. A softmax activation function is applied to a dense layer to perform classification into 20 classes, producing probabilities for each class.
Model Compilation	Loss function: Categorical Cross-entropy. (for multi-class classification) Optimiser: Adam with learning rate 1e-4 as Adam optimiser with a low learning rate facilitates stable convergence and efficient gradient updates. Metrics: Accuracy.

4.2 Customized architecture

Component	Description
Customised Architecture	ResNet50
Explanation	In the customised ResNet50 model, Stage 5 is entirely removed. This stage typically comprises three convolutional blocks: Conv5_block1, Conv5_block2, and Conv5_block3. All these blocks and their respective convolutional layers are removed from the model. In the customised ResNet50 model, only layers till Block 2-2 from Stage 4 are retained. All convolutional layers and blocks after Block 2-2 in Stage 4 are removed from the model.
Weight Initialization	The model is initialised with weights pre-trained on the ImageNet dataset, allowing it to leverage pre-learned knowledge for improved generalisation.
Head for Classification	The custom head comprises a Global Average Pooling layer and a dense layer with ReLU activation function. This dense layer has 1024 parameters with L2 regularisation (0.45) and dropout (0.65), followed by a batch normalisation layer. Then, a dense layer with a softmax activation function performs classification into 20 classes, producing probabilities for each class.
Model Compilation	The model is compiled using: Optimizer: Adam. (1e-4) Loss function: Categorical cross-entropy. Metrics: Accuracy.

4.3 Assumptions/intuitions

The selection of ResNet50 is based on its deep structure and residual networks, which offer a robust foundation for feature extraction.

Enhancements	Details	Reasons
Data Augmentation	Rotation: 20° Zoom: 20% Horizontal flipping	It introduces variability, allowing the model to learn from diverse perspectives and scales, which enhances its robustness.
ResNet50 Simplification	Eliminating Stage 5. Retaining layers till Stage 4, Block 2-2.	The complexity of the model led to fluctuations in the validation set. Reducing model complexity to mitigate overfitting during training and to enhance training efficiency.
Dense Layers with Regularization	L2 regularisation with a lambda of 0.45.	Promotes simpler decision boundaries to reduce overfitting.
Dropout	Dropout rate of 0.65	Prevents overreliance on specific neurons, enhancing generalisation.

Batch Normalization Layer	-	To adjust scaling activations from the previous layer.

4.4 Model Summary

The link below contains the summaries of the Baseline Model and the Customized ResNet50 Model.

Image Classification Model Summary

5 CNN Architecture for Object Detection:

5.1 Faster RCNN

Architecture	
Backbone Network	ResNet50 comprises multiple residual blocks, allowing for effective feature extraction at different scales.
Region Proposal Network	Generates region proposals, which are areas in the image likely to contain objects. It slides a small window across the backbone network's feature map, predicting each window's bounding box proposals and objectness scores.
Region of Interest (RoI) Pooling	RoI pooling extracts fixed-size feature maps from the backbone feature map for each proposal. Subsequent layers receive inputs of consistent dimensions regardless of the size or aspect ratio of the proposed regions.
RoI Head	The RoI head inputs the RoI-pooled features and performs classification and bounding box regression.
Output	The final output of the Faster R-CNN model consists of bounding box coordinates and class probabilities for each detected object in the input image.

5.2 YOLO

Architecture	
Backbone Network: YOLOv5 small	The backbone consists of a series of convolutional layers, including a combination of convolutional, focus, bottleneck, and spatial pyramid pooling (SPP) layers. These layers extract hierarchical features from the input image at different scales to facilitate object detection.
Head Network	It processes backbone features and predicts bounding boxes and class probabilities with convolutional layers, upsampling, and concatenation. A detection layer then utilises anchor boxes to make predictions for detected objects.

5.3 Assumptions/intuitions

Utilising Faster R-CNN, a state-of-the-art object detection framework, enables accurate identification and localisation of solar panel damage within thermal images. Given the intricate and diverse nature of such damage patterns, Faster R-CNN offers robustness and precision in its detection capabilities. The backbone ResNet50 is a deep neural network which offers robust performance due to its residual network.

YOLOv5 (You Only Look Once) was chosen for its real-time performance, simplicity, accuracy, and unified framework, allowing it to detect multiple objects in a single pass through the network. A YOLOv5small backbone is chosen due to its suitability as per the dataset.

5.4 Model Summary

The link below contains the summaries of the Faster RCNN Model and YOLOv5 Model.

Object Detection Model Summary

6 Experimental results and discussion:

6.1. Experimental settings

Image Classification

Settings	Baseline Model	Customised Model
Data Augmentation	Not performed	▪ Rotation: 20° ▪ Zoom: 20% ▪ Horizontal flipping
Transfer Learning	▪ Pre-trained model: ResNet50 pre-trained on ImageNet. ▪ Top layers: Excluded from the pre-trained ResNet50 model. ▪ Custom head: Added a Global Average Pooling layer, followed by Dense layer (1024 parameters with ReLU). ▪ Trainable layers: All layers in the baseline ResNet50 model are set to trainable. ▪ Number of output classes:20.	▪ Pre-trained model: ResNet50 pre-trained on ImageNet. ▪ Eliminating Stage 5. ▪ Retaining Stage 4, Block 2-2 ▪ Custom head: Added a Global Average Pooling layer, followed by Dense layer. (1024 parameters with ReLU). ▪ Dense having Regularization L2 regularisation with a strength of 0.45 ▪ Dropout rate of 0.65 ▪ Followed by Batch Normalization layer

Hyper-parameter settings	▪ Optimizer: Adam (1e-4) ▪ Loss function: Categorical cross-entropy. ▪ Metrics: Accuracy.	▪ Optimizer: Adam (1e-4) ▪ Loss function: Categorical cross-entropy. Metrics: Accuracy.
Training configurations	▪ Epochs: 30 epochs ▪ Image size: 224 X 224 ▪ Batch size: 32 ▪ Steps per epoch: 68 (total train_generator samples/train_generator batch size)	▪ Epochs: 100 epochs ▪ Image size: 224 X 224 ▪ Batch size: 32 Steps per epoch: 68 (total train_generator samples/train_generator batch size)
Validation Configurations	▪ Batch size: 32 Steps per epoch: 15 (total train_generator samples/train_generator batch size)	▪ Batch size: 32 Steps per epoch: 15 (total train_generator samples/train_generator batch size)
Callbacks	Checkpoint to save the best model by monitoring validation loss	Checkpoint to save the best model by monitoring validation loss
Verbose	Set to 2 for a moderate level of verbosity during training.	Set to 2

Object Detection

Settings	**Faster RCNN**	**YOLOv5**
Backbone	ResNet50	Small
Image size	-	416
Batch size	3	8
Epochs	100	600
Classes	6	5

6.2. Experimental results

Image classification

- **Performance on baseline/standard architecture**

Dataset	**Accuracy**	**Loss**
Training	1.0000	2.9683e-04
Validation	0.9385	0.2511
Testing	0.9215	0.2667

- **Performance on customised architecture**

Dataset	**Accuracy**	**Loss**
Training	0.9977	0.0886
Validation	0.9033	0.3948
Testing	0.9091	0.5121

Object Detection

Performance on Faster-RCNN

```
{'classes': tensor([1, 2, 3, 4, 5], dtype=torch.int32),
 'map': tensor(0.4645),
 'map_50': tensor(0.5813),
 'map_75': tensor(0.5586),
 'map_large': tensor(-1.),
 'map_medium': tensor(0.7893),
 'map_per_class': tensor([0.3414, 0.2699, 0.5578, 0.4117, 0.7416]),
 'map_small': tensor(0.3266),
 'mar_1': tensor(0.0711),
 'mar_10': tensor(0.3063),
 'mar_100': tensor(0.4959),
 'mar_100_per_class': tensor([0.3693, 0.2907, 0.6007, 0.4461, 0.7725]),
 'mar_large': tensor(-1.),
 'mar_medium': tensor(0.8251),
 'mar_small': tensor(0.3534)}


("Classes: ['__background__', 'Cell', 'Cell-Multi', 'No-Anomaly', 'Shadowing', "
 "'Unclassified']")
```

Fig.3

```
AP / AR per class
-------------------------------------------------------------------------
|    | Class               | AP                | AR                |
-------------------------------------------------------------------------
|1   | Cell                | 0.341             | 0.369             |
|2   | Cell-Multi          | 0.270             | 0.291             |
|3   | No-Anomaly          | 0.558             | 0.601             |
|4   | Shadowing           | 0.412             | 0.446             |
|5   | Unclassified        | 0.742             | 0.772             |
-------------------------------------------------------------------------
|Avg                       | 0.464             | 0.496             |
```

Fig.4

Performance on YOLO

MAP on Training Set

```
new_train_yaml summary: 182 layers, 7257306 parameters, 0 gradients
                 Class     Images  Instances          P          R      mAP50
                   all        250      14765      0.876      0.805      0.869      0.651
                  Cell        250        860      0.757      0.697      0.778       0.57
            Cell-Multi        250        485      0.772        0.6      0.725      0.526
            No-Anomaly        250      11437      0.969      0.964      0.985      0.744
             Shadowing        250       1463      0.953      0.923      0.953      0.659
          Unclassified        250        520       0.93       0.84      0.902      0.757
Results saved to ../../../../runs/train/exp
```

Fig.5

MAP on Validation Set

Class	Images	Instances	P	R	mAP50	
all	250	14765	0.872	0.809	0.86	0.67
Cell	250	860	0.753	0.702	0.748	0.574
Cell-Multi	250	485	0.769	0.612	0.71	0.545
No-Anomaly	250	11437	0.964	0.966	0.978	0.755
Shadowing	250	1463	0.945	0.924	0.953	0.686
Unclassified	250	520	0.93	0.84	0.912	0.792

Speed: 0.2ms pre-process, 30.5ms inference, 7.8ms NMS per image at shape (32, 3, 416, 416)
Results saved to runs/val/exp

Fig.6

6.3 Discussion

For Image Classification

The comparison between the baseline and customised ResNet50 models reveals distinct patterns in training and validation performance. While both models achieve high training accuracies, indicating effective learning from the training data, the validation accuracies vary significantly. For the baseline model, the validation loss increased while the training loss decreased towards the end, suggesting overfitting. However, the model shows robust performance on the test set regardless due to residual networks.

Fig.7 Baseline ResNet50

I tried to reduce this overfitting in the customised model by employing techniques like data augmentation, L2 regularisation and dropouts. The model's overfitting is reduced as there are more fluctuations in the validation set. Furthermore, the gap between the validation and test accuracy was reduced compared to the baseline model, which indicates better learning of the model. Leveraging ImageNet weights significantly boosted model performance using pre-trained knowledge from a vast dataset. Given the similarity between my dataset and ImageNet, these weights effectively captured relevant features, expediting model convergence and enhancing overall performance.

Fig.8 Customized ResNet50

For object detection

In the case of Faster R-CNN, mean average precision (mAP) is 46.45%, However, the performance across classes varies significantly. For instance, classes such as 'Cell-Multi' and 'Cell' exhibit relatively lower average precision values of 27% and 34.1%, respectively. This discrepancy suggests that the model struggles with accurately detecting certain object categories due to class imbalance.

Faster R-CNN does face challenges in achieving higher mAP50 values for certain classes, indicating potential difficulties in accurately localising objects with a reasonable overlap threshold. This suggests that while Faster R-CNN may excel in detection, its localisation accuracy varies across different object classes.

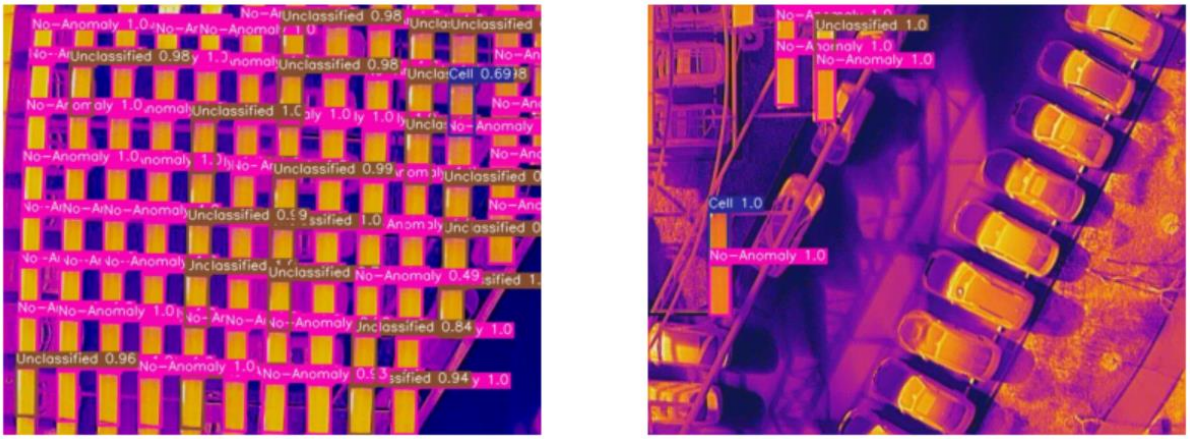


Fig.9 Faster RCNN

Similarly, YOLOv5 demonstrates competitive overall performance with a higher mAP of 65% on the training and 67% on the validation sets. However, despite the higher overall performance, It still faces challenges in accurately detecting certain object classes. For example, while achieving high precision and recall for 'No-Anomaly' objects, the model struggles with classes such as 'Cell' and 'Cell-Multi, as indicated by their lower precision and recall values. This also indicates class imbalance, occlusion, or variability in object appearance.

The lower performance of the "Cell" and "Cell-Multi" classes in Faster R-CNN and YOLOv5 can be primarily attributed to a significant class imbalance within the dataset. The limited number of data samples available for these classes makes it inherently challenging for the models to learn and generalise patterns associated with cell structures effectively. Collecting more diverse and representative data specifically for these classes would be crucial to address this issue. Thus, employing techniques such as data augmentation would help improve the model's ability to detect and classify these underrepresented classes accurately.

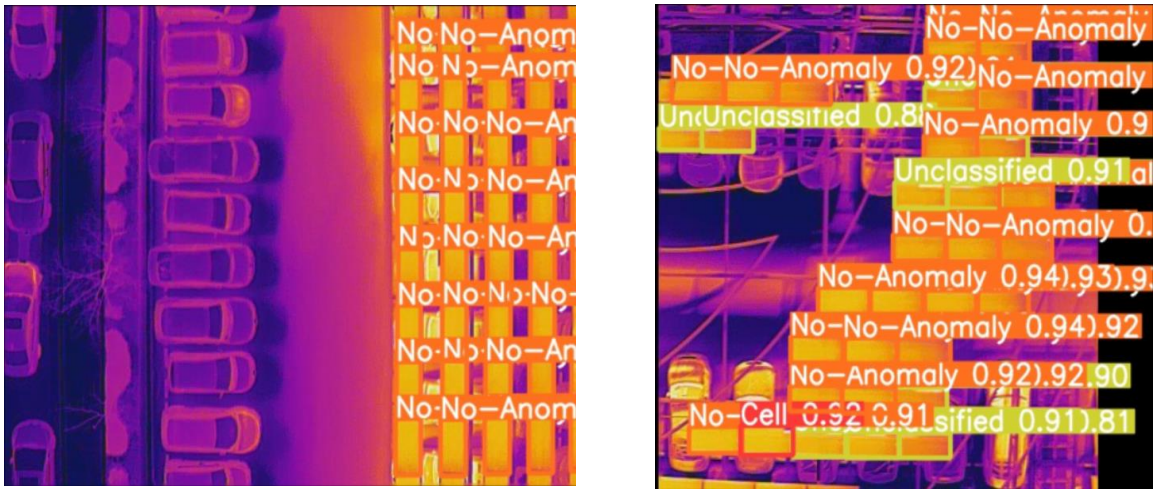


Fig.10 YOLOv5

7 Conclusion

Despite extensive efforts to enhance the customised ResNet50 model through techniques like data augmentation, L2 regularisation, and dropout implementation, the baseline model retains its performance. While both models achieve high training accuracies, the baseline model's perfect accuracy suggests superior fitting to the training data. Although showing signs of overfitting, the baseline model's lower validation loss implies better generalisation than the customised model. Additionally, the baseline model demonstrates robust performance on the test set, indicating better generalisation to unseen data. Despite attempts to simplify the architecture and mitigate overfitting in the customised model, it cannot surpass the baseline's performance. This outcome underscores the importance of careful model design.

The longer training time for Faster R-CNN compared to YOLOv5 boils down to architectural complexity and the additional step of region proposal generation. Faster R-CNN's multi-stage architecture and fine-grained feature extraction demand more computational resources and optimisation, increasing training duration. Notably, YOLOv5 showcases significant improvements in precision and recall for classes like "Cell" and "Cell-Multi" in spite of class imbalance, suggesting enhanced detection capabilities for underrepresented classes compared to Faster R-CNN.

Additionally, YOLOv5 maintains consistency in performance across classes, exhibiting fewer fluctuations in precision, recall, and mAP50 values compared to Faster R-CNN. This indicates that YOLOv5 offers more stable and reliable performance across various object classes.