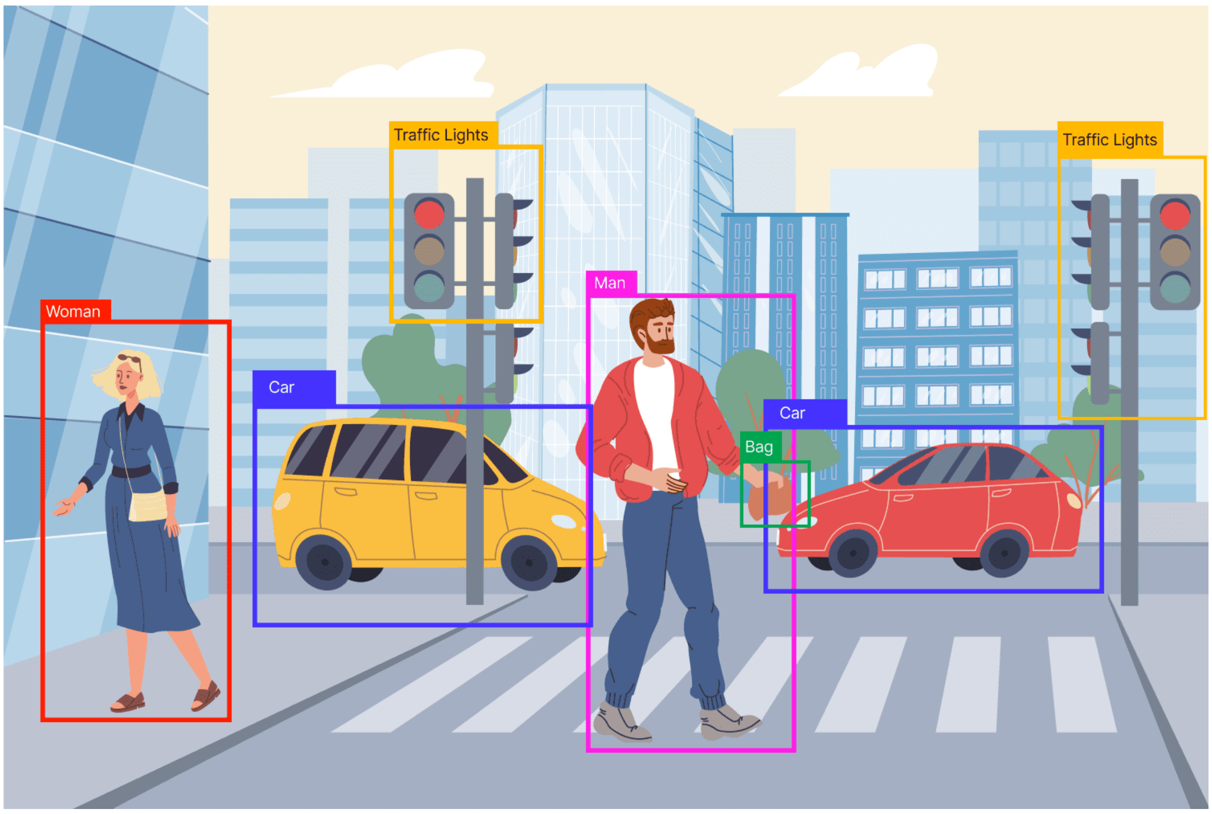
42028: Deep Learning and Convolutional Neural Network

Assignment -2

Student Name: Kiran Shanker Das

Student ID: 24580348



1. Introduction:

In this report, we delve into three separate experiments conducted in the field of computer vision. Each experiment addresses specific challenges by utilizing deep convolutional neural network (CNN) architecture, which is optimized for both performance and accuracy. Our focus areas include classifying bird and dog species, as well as detecting solar panels while identifying anomalies.

Outline of the Report

Experiment 1: Image Classification using ResNet

- Implementation of a Simple ResNet50-based Mini-ResNet CNN architecture for classifying the bird and dog species dataset using the Keras API.
- Train and test the model
- Based on the performance, the aim is to add layers, apply transfer learning and check for improvements.

Experiment 2: Object Detection using Faster R-CNN

- Description of the annotated Solar Panel Dataset and its subdivisions.
- Using Faster R-CNN with ResNet-50 backbone for detecting solar panels within thermal images.
- Training and validation procedures for evaluating the model's detection capabilities.

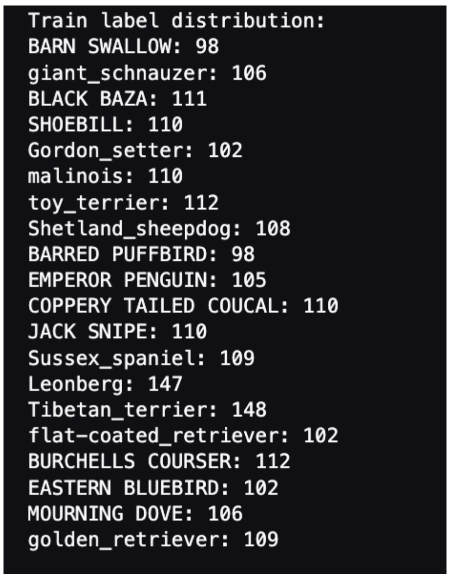
Experiment 3: Object Detection using YOLOv5

- Introduction of YOLOv5 for detecting anomalies in solar panels from thermal images.
- Known for its exceptional balance between accuracy and speed, we will be training the model over several epochs and closely monitoring the various loss functions

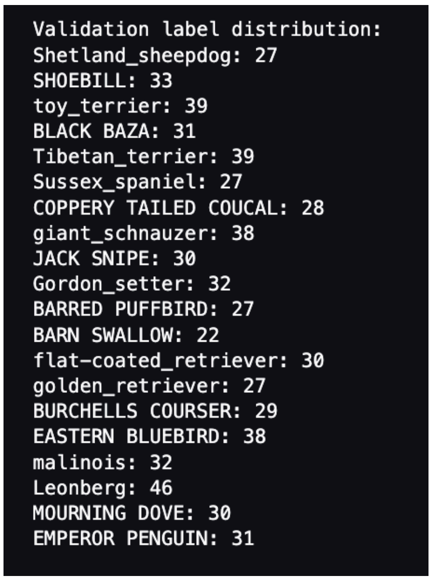
2. Dataset:

Image Classification- Experiment 1:

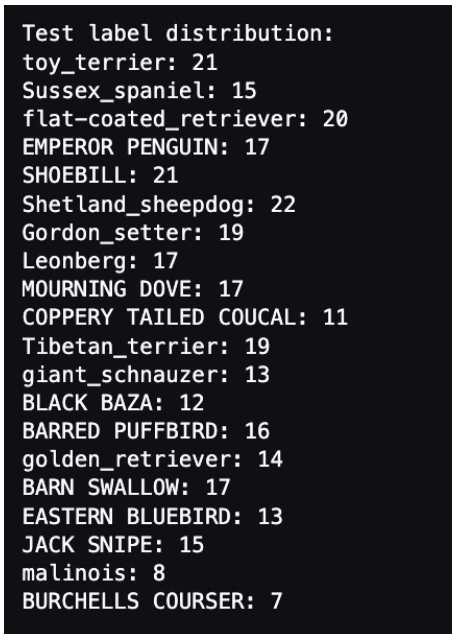
The dataset being used for ResNet is a collection of images featuring various bird and dog species. Each image is associated with a specific label indicating the species depicted in the image. The dataset is divided into three sets: training, validation, and test sets. We have split the dataset into 70, 20, and 10 for training, validation, and test sets respectively. There are a total of 3165 samples across the sets. Not all the image sizes are the same. The objective is to identify an optimal resizing strategy that preserves essential information without significant loss.



```
Train label distribution:
BARN SWALLOW: 98
giant_schnauzer: 106
BLACK BAZA: 111
SHOEBILL: 110
Gordon_setter: 102
malinois: 110
toy_terrier: 112
Shetland_sheepdog: 108
BARRED PUFFBIRD: 98
EMPEROR PENGUIN: 105
COPPERY TAILED COUCAL: 110
JACK SNIPE: 110
Sussex_spaniel: 109
Leonberg: 147
Tibetan_terrier: 148
flat-coated_retriever: 102
BURCHELLS COURSER: 112
EASTERN BLUEBIRD: 102
MOURNING DOVE: 106
golden_retriever: 109
```



```
Validation label distribution:
Shetland_sheepdog: 27
SHOEBILL: 33
toy_terrier: 39
BLACK BAZA: 31
Tibetan_terrier: 39
Sussex_spaniel: 27
COPPERY TAILED COUCAL: 28
giant_schnauzer: 38
JACK SNIPE: 30
Gordon_setter: 32
BARRED PUFFBIRD: 27
BARN SWALLOW: 22
flat-coated_retriever: 30
golden_retriever: 27
BURCHELLS COURSER: 29
EASTERN BLUEBIRD: 38
malinois: 32
Leonberg: 46
MOURNING DOVE: 30
EMPEROR PENGUIN: 31
```



```
Test label distribution:
toy_terrier: 21
Sussex_spaniel: 15
flat-coated_retriever: 20
EMPEROR PENGUIN: 17
SHOEBILL: 21
Shetland_sheepdog: 22
Gordon_setter: 19
Leonberg: 17
MOURNING DOVE: 17
COPPERY TAILED COUCAL: 11
Tibetan_terrier: 19
giant_schnauzer: 13
BLACK BAZA: 12
BARRED PUFFBIRD: 16
golden_retriever: 14
BARN SWALLOW: 17
EASTERN BLUEBIRD: 13
JACK SNIPE: 15
malinois: 8
BURCHELLS COURSER: 7
```

Fig 1- Random_state = 24580348

Object Detection- Experiment 2:

The annotated Solar Panel Dataset used in this experiment comprises over 1500 thermal images of solar panels. These images are labelled for object detection tasks and are divided into five distinct classes:

Cell: Represents individual solar cells within a solar panel.

Cell-Multi: Denotes multiple solar cells within a single panel.

No-Anomaly: Indicates areas of the solar panel without any anomalies or issues.

Shadowing: Refers to shadows cast on the solar panel surface.

Unclassified: Represents regions that cannot be categorized into the above classes.

The annotations follow the Pascal VOC (Visual Object Classes) format, ensuring compatibility with various object detection frameworks and tools.

Dataset Split:

The dataset is split into two subsets ie training and validation. These sets are stored in the AWS workspace with the following directories:

Training Set: Contains images located in the directory ../train/images, along with their corresponding annotations.

Validation Set: Consists of images located in the directory ../valid/images, used for fine-tuning model parameters and evaluating performance during training.

Please note that no test set was given.

A quick glimpse at the dataset:

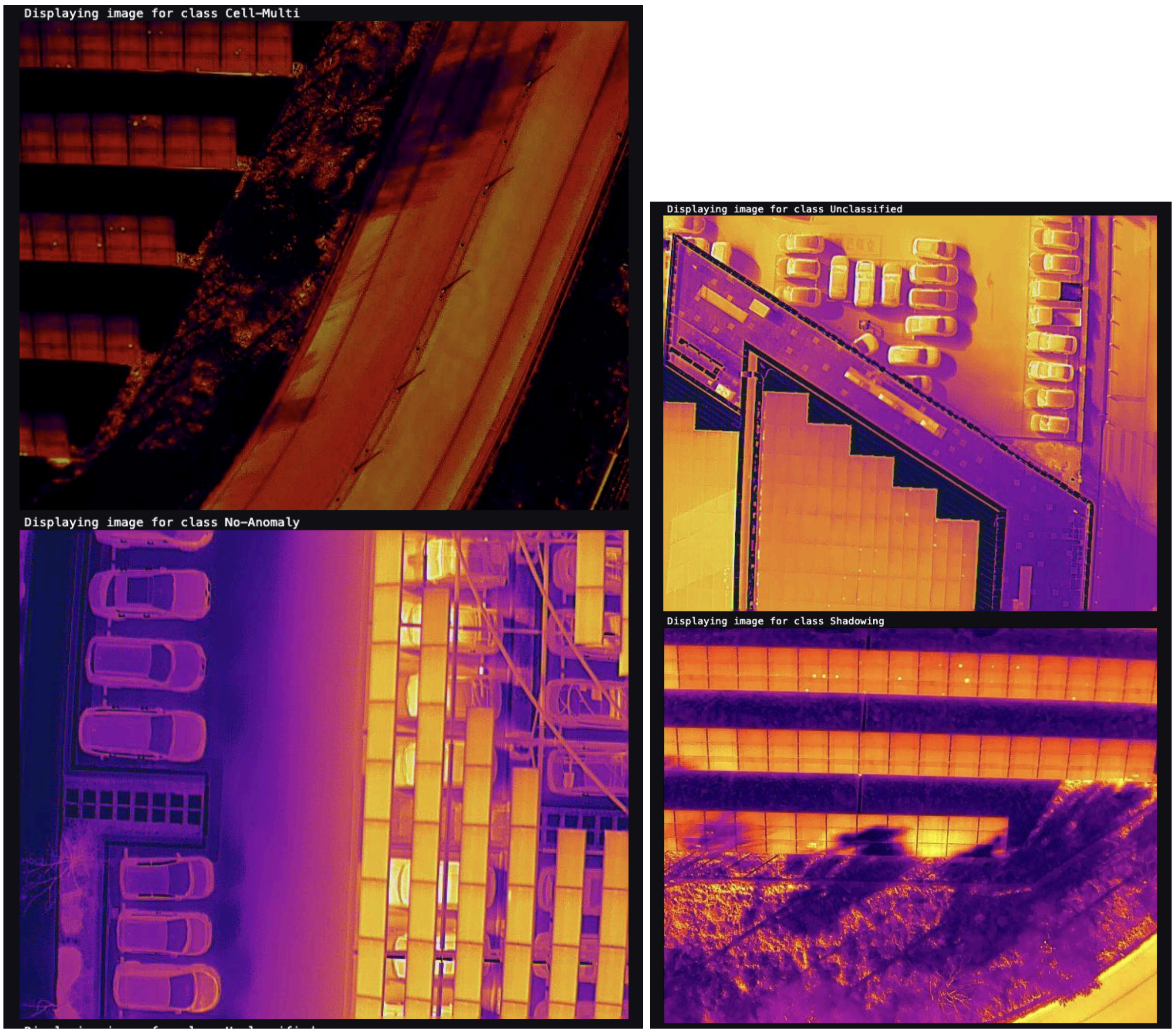

Displaying image for class Cell-Multi
Displaying image for class No-Anomaly
Displaying image for class Unclassified
Displaying image for class Shadowing

Fig 2- Sample thermal images from each class

Looking at the data, it seems that "No-Anomaly" significantly outweighs the other classes. This imbalance can potentially lead to biased models, reduced performance for minority classes, and difficulties in learning discriminative features.

Object Detection- Experiment 3:

This dataset contains over 1600 thermal labelled images of solar panels for object detection in 5 classes. Here annotations/labels are in a txt format so that it can be used for training with Yolov5. This is split into train, validation and test.

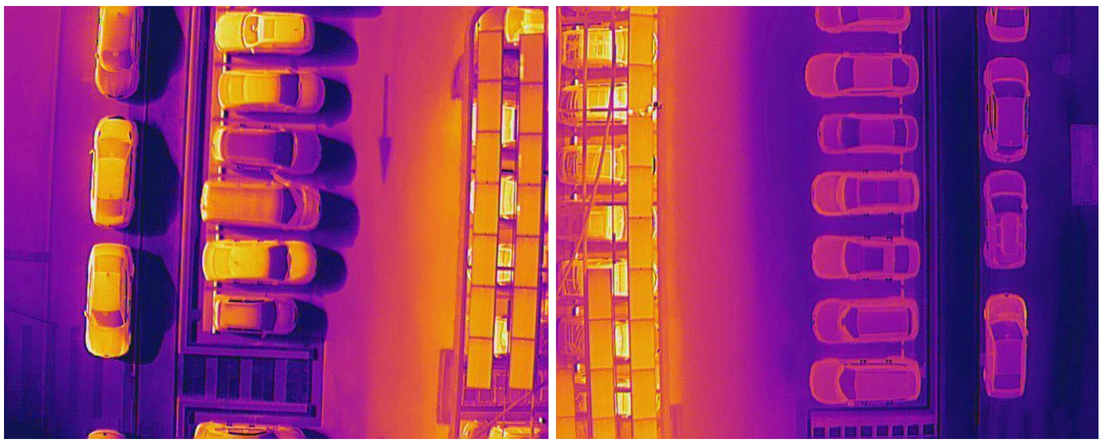


Fig 2- Sample thermal images from the dataset

3. Proposed CNN Architecture for Image Classification:

Baseline Architecture:
The Micro-ResNet architecture leverages residual connections to address the vanishing gradient problem, a common challenge in training deep neural networks. These connections allow the network to learn from both the current layer and its preceding layers, mitigating the vanishing gradient effect.
Layers: (Baseline Model Summary- link)
Our model, named "Micro_ResNet," is designed with a simple res_block with dropout for regularization.
Considerations:
Input Layer: This layer receives images as input, each sized 256x256 pixels with 3 colour channels (RGB).
Convolutional Layers: Our model consists of multiple convolutional layers with varying filter sizes and numbers to capture different levels of abstraction in the images.
Batch Normalization Layers: These layers improve training stability and help network learn more effectively.
Activation Layers: ReLU activation functions are applied to the outputs of convolutional layers.
Pooling Layers: We downsample the feature maps obtained from convolutional layers.
Global Average Pooling Layer: This layer computes the average of each feature map's values, for global representation of the features.
Flatten Layer: The output of the global average pooling layer is flattened into a 1D vector, preparing it for input into the fully connected layers.
Dense Layers: Includes two fc/dense layers: the first with 1000 units and the second with 20 units, required to match the number of output classes (bird and dog species).
Dropout: Deactivates a percentage of units during training, preventing the network from becoming overly dependent on specific features.

Assumptions:
The intuition behind the Micro-ResNet architecture for this image classification task with birds and dogs lies in addressing two key challenges:

Feature Extraction: Extracting informative features from images that allow the model to distinguish between different bird and dog species and see how well the model can perform on the validation set.
Overfitting Prevention: To avoid overfitting, we've included regularizers (l2(1e-5)), set a dropout rate of 0.2, and used GlobalAveragePooling2D. These initial settings help us assess the model's ability to prevent overfitting so we can adjust them later.

Custom Architecture: We have made changes to the baseline model with the below approach:
Custom Model summary- link

1. Adding a pre-trained model (Imagenet weights initialised): The model uses the ResNet50 architecture pre-trained on ImageNet for feature extraction, excluding the top layers to facilitate transfer learning on a custom dataset.
2. Custom ResBlocks: Additional residual blocks are added to the base model to enhance feature learning along with batch normalization and dropout for improved model performance.
3. Classification Layers: Just like the baseline model, Global average pooling is applied for dimensionality reduction followed by dense layers for classification, including dropout regularization to reduce overfitting, followed by a softmax activation layer for multi-class classification.

Assumptions: The baseline model wasn't improving beyond 10 epochs. Using pre-trained initialised weights might help improve the accuracy of the model.

4. CNN Architecture for Object Detection:

fasterrcnn_resnet50_fpn_v2: The Solar Panel Dataset consists of thermal images with multiple classes (solar cells, shadows, etc.) requiring object detection and localization. Faster R-CNN excels at these tasks with its RPN and classification/bounding box regression steps. Additionally, thermal images might lack clear boundaries, making Faster R-CNN's region-based approach advantageous.

The chosen backbone, ResNet-50, offers several advantages:
Powerful Feature Extraction (50 Layers): Its deep architecture with 50 layers can learn complex features from thermal images, crucial for differentiating solar panels from other elements with subtle temperature variations.
Transfer Learning Potential: Pre-trained on a massive dataset like ImageNet, ResNet-50 can leverage this knowledge to adapt to solar panel detection in thermal images more efficiently.

YOLOv5- While Faster R-CNN with ResNet-50 is a strong choice for this task, another suitable model for solar panel detection in thermal images is YOLOv5 (You Only Look Once, version 5).
Here's why YOLOv5 might be a compelling alternative:
Real-Time Detection Potential: YOLOv5 is known for its speed and efficiency, making it a good candidate for scenarios where real-time detection of solar panels is desirable.
Single-Stage Detection: Unlike Faster R-CNN's two-stage approach (RPN and classification/bounding box regression), YOLOv5 performs both object detection and bounding box regression in a single stage. This can potentially lead to faster inference times, crucial for real-time applications.
Unified Framework: YOLOv5 offers an all-in-one framework for object detection, eliminating the need for separate region proposal and classification steps. This can simplify the model architecture and potentially streamline the training process.

5. Experimental results and discussion:

a. Experimental settings:

i. **Image Classification:**

Baseline: Data Augmentation

```
   rotation_range=40,
   width_shift_range=0.2,
   height_shift_range=0.2,
   shear_range=0.2,
   zoom_range=0.2,
   horizontal_flip=True,
   fill_mode='nearest')
   #Image resized to 256 by 256
```

Baseline: Regularisers

```
kernel_regularizer=l2(1e-5))
dropout_rate=0.2 #in the resblock
classifer_Block = Dropout(0.5) #in the Dense layer
```

Custom: Data Augmentation

```
   rotation_range=40,
   width_shift_range=0.2,
   height_shift_range=0.2,
   shear_range=0.2,
   zoom_range=0.2,
   horizontal_flip=True,
   fill_mode='nearest')
   #Image resized to 256 by 256
```

Baseline: Regularisers

```
kernel_regularizer=l2(1e-4)
dropout_rate=0.2 (in the resblock)
classifer_Block = Dropout(0.5)(Dense layer)
lr = 1e-4  # Initial learning rate decreased to 1e-4
   if epoch > 50:
      lr *= 0.1  # Will decrease learning rate after
                    50 epochs
  return lr
```

ii. **Object Detection: fasterrcnn_resnet50_fpn_v2:** Batch size of 2 is chosen for training on PyTorch 3.8 GPU ml.g4dn.xlarge to balance memory utilization and computational efficiency, ensuring optimal performance without exceeding GPU memory constraints.

iii. **Object Detection: YOLOv5:** Batch size of 16 and resized images of 416x416 pixels have been used to train the model.

b. Experimental results:

i. Image classification:

1. Performance on baseline/standard architecture

```
+-------------+--------------------+
|   Metric    |       Value        |
+-------------+--------------------+
| Test Loss   | 1.8151865005493164 |
| Test Acc    | 0.4076433181762695 |
| Val Loss    | 1.9527000188827515 |
| Val Acc     | 0.3679245412349701 |
| Train Loss  | 1.706318736076355  |
| Train Acc   | 0.4465011358261108 |
+-------------+--------------------+
```

2. Performance on customized architecture

```
| Metric        | Value                  |
|---------------|------------------------|
| Accuracy      | 0.9682                 |
| Test Loss     | 0.22683264315128326    |
| Test Accuracy | 0.9681528806686401     |
| Accuracy      | 0.9497                 |
| Val Loss      | 0.2424217313528061     |
| Val Accuracy  | 0.9496855139732361     |
| Accuracy      | 0.9986                 |
| Train Loss    | 0.0025208028964698315  |
| Train Accuracy| 0.9986456036567688     |
```

ii. Object Detection:

1. Performance on Faster-RCNN

```
{'classes': tensor([1, 2, 3, 4, 5], dtype=torch.int32),
 'map': tensor(0.5027),
 'map_50': tensor(0.6226),
 'map_75': tensor(0.6049),
 'map_large': tensor(-1.),
 'map_medium': tensor(0.8092),
 'map_per_class': tensor([0.3886, 0.2899, 0.5584, 0.5148, 0.7619]),
 'map_small': tensor(0.3625),
 'mar_1': tensor(0.0794),
 'mar_10': tensor(0.3340),
 'mar_100': tensor(0.5389),
 'mar_100_per_class': tensor([0.4181, 0.3107, 0.6035, 0.5572, 0.8052]),
 'mar_large': tensor(-1.),
 'mar_medium': tensor(0.8515),
 'mar_small': tensor(0.3926)}
```

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

AP / AR per class

```
---------------------------------------------------------------------------
|    | Class            | AP              | AP               |
---------------------------------------------------------------------------
|1   | Cell             | 0.389           | 0.418            |
|2   | Cell-Multi       | 0.290           | 0.311            |
|3   | No-Anomaly       | 0.558           | 0.604            |
|4   | Shadowing        | 0.515           | 0.557            |
|5   | Unclassified     | 0.762           | 0.805            |
---------------------------------------------------------------------------
|Avg                    | 0.503           | 0.539            |
```

2. **Performance on SDD/YOLO or any object detector**

```
         Class    Images  Instances      P          R        mAP50
           all      1167      63861      0.884      0.802      0.862      0.656
          Cell      1167       3663      0.775      0.681      0.746      0.561
    Cell-Multi      1167       2023      0.798      0.544      0.691      0.507
    No-Anomaly      1167      48825      0.965      0.927      0.956      0.739
     Shadowing      1167       6959      0.941      0.962      0.979      0.69
  Unclassified      1167       2391      0.939      0.895      0.937      0.786
```

iii. Discussion:

Custom Resnet: Overall, the training progress looks promising, with the model achieving a reasonably high validation accuracy of around 0.94 while keeping the validation loss under control. The large gap between training and validation accuracy, especially in the initial epochs, is because weights from imagenet were initialised. This gap appears to be reducing as training progresses with 1e-4 (ideal for fine-tuning and stable training as weights already have a good starting point), which is a positive sign.

fasterrcnn_resnet50_fpn_v2: The model achieved a mean average precision (mAP) of 50.27% across all classes, with mAP@50% intersection over union (IoU) of 62.26% and mAP@75% IoU of 60.49%. It performed best on medium-sized objects with an mAP of 80.92%, while performance on small-sized objects was lower with an mAP of 36.25%. The mean average recall (mAR) at 1, 10, and 100 detections per image was 7.94%, 33.40%, and 53.89% respectively, indicating moderate performance in detecting objects across different thresholds.

YoloV5:
1.No-Anomaly class has the highest precision (96.5%) and recall (92.7%), indicating that the model performs exceptionally well in detecting instances of no anomaly.
2 Shadowing class also performs well with high precision (94.1%) and recall (96.2%).
3. Cell-Multi class has relatively lower precision (79.8%) and recall (54.4%), indicating that the model struggles more with accurately detecting instances of this class.
4. Cell class has similar precision (77.5%) and recall (68.1%) to Cell-Multi, indicating some difficulty in detection.

5. Unclassified class has high precision (93.9%) but slightly lower recall (89.5%), suggesting that while the model is good at identifying instances of this class, it might miss some of them.

6. <u>Conclusion</u>

The experiments conducted in this report showcase the efficacy of deep learning, particularly convolutional neural networks (CNNs), in tackling various challenges in image classification and object recognition. A simple architecture, as seen in Experiment 1 for image classification, with fewer layers and parameters, offers advantages like reduced computational cost, ease of training, and interpretability. In Experiment 1, we gauged if the diversity introduced after data augmentation led to any performance gain, and indeed, it did, although we observed significant fluctuations in val_accuracy, necessitating the exploration of various regularization techniques and learning rates. Whereas, with the backbone, as in Experiment 2 and Experiment 3, using complex architectures like ResNet-50 and YOLOv5, we observed a significant enhancement in performance (although it took a few epochs for train-val accuracy or mAP in the case of YOLOv5 to improve or converge), especially in tasks requiring complex feature extraction and localization. These complex backbones, while computationally expensive, excel in capturing nuanced patterns and representations in the data, ultimately leading to higher accuracy and better generalization.