# Retina Net

**Introduction**

The **Facebook AI Research** (FAIR) team has introduced the RetinaNet model to address the problem of **dense and small object detection**. As a result, it has become a popular object detection model that can be used with both aerial and satellite imagery. RetinaNet was developed by researchers by making two improvements over existing single-stage object detection models – Focal Loss & Feature Pyramid Networks (FPN)

**Why Retina Net Model:**

Both classic one stage detection methods, such as boosted detectors and DPM, and more recent methods, such as SSD, evaluate almost 10^4 to 10^5. There are several candidate locations per image, but only a few of them contain objects (i.e. foreground) and the rest are just background objects. As a result, there is a problem with **class imbalance**.

This turned out to be the primary reason why one stage detectors performed worse than two-stage detectors. As a result, researchers developed the RetinaNet Model with the concept of **Focal Loss** to compensate for the class imbalances and inconsistencies of single-shot object detectors such as YOLO and SSD when dealing with extreme foreground-background classes.

**RetinaNet Model Architecture:**

In essence, we can divide the RetinaNet architecture into three parts:

1. Backbone Network (bottom-up pathway + top-down pathway with lateral connections,

such as ResNet + FPN),

1. Object Classification Sub-Network,
2. Object Regression Sub-Network

**Model Implementation & Results:**

Keras Object Detection API is used here to train, evaluate and train the model. RetinaNet uses a feature pyramid network to efficiently detect objects at multiple scales and introduces a new loss, the Focal loss function, to alleviate the problem of the extreme foreground-background class imbalance. RetinaNet uses a ResNet based backbone, using which a feature pyramid network is constructed. In the project, we use ResNet50 as the backbone and return the feature maps at strides 8, 16 and 32. Here, the entire given dataset has around 8K images. To reduce the image pre-processing functionality and to make it computationally efficient, the 'TFRecord' file format of the dataset is used here which is available on TensorFlow dataset. Below are a few input parameters and hyperparameters used while training the Retinanet model. The best accuracy achieved by these parameters is around 0.43.

|  |  |
| --- | --- |
| **Parameters/Hyperparameters** | **Value** |
| Image size | 256 x 256 |
| Match IOU | 0.8 |
| Ignore IOU | 0.5 |
| Confidence Threshold | 0.8 |
| NMS\_IOU\_Threshold | 0.8 |
| Box variance | [0.1, 0.1, 0.2, 0.2] |
| Batch Size | 1 |
| Learning Rate | 0.001 |
| Momentum | 0.9 |
| Clipvalue | 0.5 |
| Epoch | 10 |

Line chart

Description automatically generated with medium confidence

Above plots shows model accuracy and model loss vs epochs. The predictions made by the retinanet model trained here and the ground truth of the images is as show below.

|  |  |
| --- | --- |
| Ground Truth | Predictions by Retinanet |
| A picture containing text, van, car  Description automatically generated |  |
| A white car on a road  Description automatically generated with medium confidence |  |