## 

**Diploma Thesis**

*Comparison of Artificial Intelligence systems for the detection of objects on UAV-based images.*

**TRIMAS CHRISTOS**

**CHANIA, JUNE 2021**

**Committee**

**Professor Zervakis Michalis(Supervisor)**

**Associate Professor Lagoudakis Michalis**

**Professor Petrakis Euripidis**

**Acknowledgements**

First, I would like to thank my family who supported me all those years and never made a discount in my studies.

Next, I would like to thank Professor Michalis Zervakis and Dr. Marios Antonakakis who as my advisors guided me through the whole process of this thesis.

Last, but certainly not least, my dear friends who supported and helped me, through the difficult years of Electrical and Computers Engineering studies.

**Περίληψη**

**Abstract**

**Contents:**

***Chapter 1: Introduction…………………………………………………………….1***

* 1. **Unmanned Aerial Vehicles………………………………………….....1**
  2. **Object Detection………………………………………………………....1**

***Chapter 2: Artificial Intelligence, Machine Learning, Deep Learning…………..1***

* 1. **Machine Learning………………………………………………………..1**
  2. **Deep Learning……………………………………………………………1**
  3. **Convolutional Neural Network (CNN)……………………………….1**
     1. *Convolution Layer…………………………………………………..1*
     2. *Pooling Layer*
     3. *Fully Connected Layers*
     4. *Activation Functions*
     5. *Loss functions*

***Chapter 3: Convolutional Neural Networks***

* 1. **Visual Geometry Group Network (VGGNet)**
  2. **Residual Network (ResNet)**
  3. **U-Net**
  4. **Feature Pyramid Network(FPN)**

***Chapter 4: Object Detection systems***

* 1. **Two-stage architectures**
     1. *R-CNN*
     2. *Fast R-CNN*
     3. *Faster R-CNN*
  2. **Single Stage architectures**
     1. *Retina Network*
     2. *You Only Look Once (YOLO)*

**Chapter 5: RetinaNet analysis**

* 1. **Anchor Boxes**
  2. **Backbone Network**
  3. **Classification/Regression Networks**
  4. **Loss Functions.**
  5. **Why Retina?**

**Chapter 6: Dataset.**

* 1. **Stanford Drone Dataset (SDD) description.**
  2. **Changes in the Dataset**

***Chapter 7: Evaluation Metrics***

* 1. **Intersection over Union(IoU)**
  2. **Metrics**

***Chapter 8: Experiments***

* 1. **Simple Split**
  2. **k-fold Cross Validation**
  3. **Bottom-up FPN architecture**

***Chapter 9: Comparison***

* 1. **Comparison with other models**
  2. **Comparison with original RetinaNet**

***Chapter 10: Conclusions***

* 1. **Limitations**
  2. **Future work**

***Chapter 11: References***

1. **Introduction**

**1.1) Unmanned Aerial Vehicles**

Unmanned systems are typically known as powered vehicles that do not carry a human operator, can be operated autonomously or remotely and can carry a variety of payloads depending on their type, functionality and mission objectives.

Unmanned Aerial Systems, also known as a drone, have experienced the greatest growth. As of 2020, seventeen countries have armed UAVs, and more than 100 countries use UAVs in a military capacity. The global military UAV market is dominated by companies based in the United States and China. With extensive cost reduction in electronics, the defense forces around the globe are utilizing UAVs for applications such as logistics, communications, attack and combat, while commercial applications include aerial photography and filmmaking, cargo transport and detection of disasters [1].

Whether it comes to the detection of objects of interest (refugee waves, tracking and exterminating target), prison surveillance or information gathering of a battlefield, UAVs have proven their usefulness. A significant contribution to this development, played the evolution of cameras. The cameras on-board UAVs are a rich source of information that can be processed in order to extract meaningful information. Besides the cameras, the development of other advanced hardware and software technologies allow drones to carry out their missions without human intervention, such as computer vision, object detection, machine learning, thermal sensors and deep neural networks.

**1.2) Object Detection**

Object detection is a computer technology related to computer vision and image processing that deals with localization and identification of semantic objects of a certain class, in digital images and videos. In other words, given an image or a video stream, an object detector can identify-classify objects of interest and provide information about their positions within the image.

With the evolution of cameras and the oversimplification of data gathering and processing, object detection can be used in the following military and commercial areas:

* Surveillance.
* Search and Rescue missions.
* Anomaly detection.
* Autonomous driving

The basic idea of object detection, is that every object class has its own special features that helps in classifying the class- for example all circles are round. Object detection models learn those special features and create patterns on the objects properties. Features may be specific structures in the image such as points or edges. More broadly a feature is any piece of information which is relevant for solving the computational task related in computer vision applications. The feature extraction process can be a computational expensive and many times due to time constraints, a higher level algorithm may be used to guide the feature detection stage, so that only certain parts of image are searched for features.

There are two kinds of object detection methods:

1. Neural Network approaches.
2. Non-Neural approaches.

Non-Neural approaches use one of the following techniques for feature extraction and an algorithm such as Support Vector Machines for classification of those features.

* **Viola-Jones object detection framework based on Haar features.**
* **Scale-Invariant feature transform.**
* **Histogram of oriented gradients features.**

Neural Network approaches can be distinguished in to two-stage detectors and single-stage detectors. The first ones use a box proposal algorithm as the first stage, and the second stage classifies those proposals, while the second ones detect objects and classify them in the image in one pass through the network.

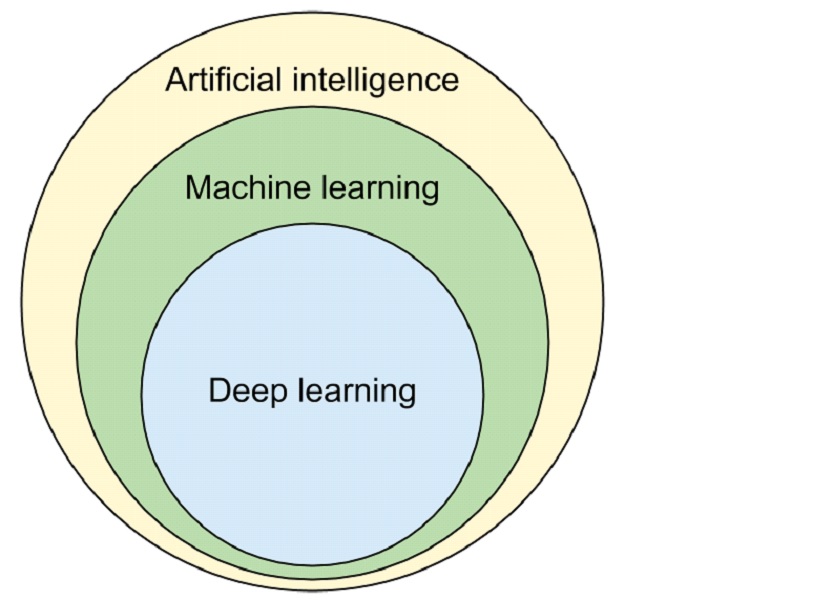
The most known detectors are:

* **Regional Proposal Networks such as R-CNN.**
* **Single Shot MultiBox Detector.**
* **You Only Look Once (YOLO).**
* **Retina-Net.**

**2. Artificial-Intelligence, Machine-Learning, Deep-Learning**

Nowadays, the word Artificial Intelligence or A.I. sounds everywhere and it is used increasingly. A.I. refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

The most common applications of A.I. are: autonomous cars, voice and face recognition, data analysis, virtual assistance and other applications in various industries. Subfields of Artificial Intelligence are machine learning and deep learning.



**Image [1]: A.I., M.L. and D.L.**

**2.1) Machine Learning**

The concept of machine learning dramatically changes the way of how classical programming works. In the classical method, someone provides the data and defines the rules of the program to obtain an answer. In machine learning or ML, someone give the data with the answers and demands from the machine to create the rules. The rules can then be applied to a new data to confirm the results and to generate new answers. In other words, ML consists of algorithms that improve automatically through experience and by the use of data.

A subset of Machine Learning is Deep Learning.

**2.2) Deep Learning**

From Machine Learning Deep Learning was born. D.L. is part of a broader family of machine learning methods based on artificial neural networks with feature learning. Deep-learning architecture such as deep neural networks and convolutional neural networks have been applied to fields including computer vision and image analysis.

A Deep Neural Network (DNN) is an artificial neural network with multiple layers between the input and the output layers. In computer vision the most used class of deep neural networks are Convolutional Neural Networks or CNNs.

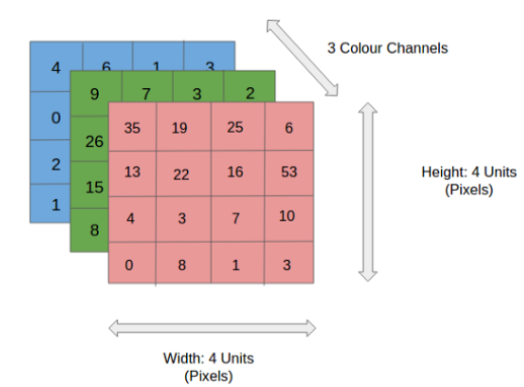
**2.3) Convolutional Neural Networks**

A Convolutional Neural Network is a Deep Learning algorithm which can take in an input image, assign importance to various objects in the image and be able to differentiate one from the other. ConvNets require less pre-processing compared to other classification algorithms.

The architecture of a CNN is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the visual cortex. Individual neurons respond to stimulations only in a restricted region of the visual field known as the Receptive Field. A collection of such overlap cover the entire visual area.

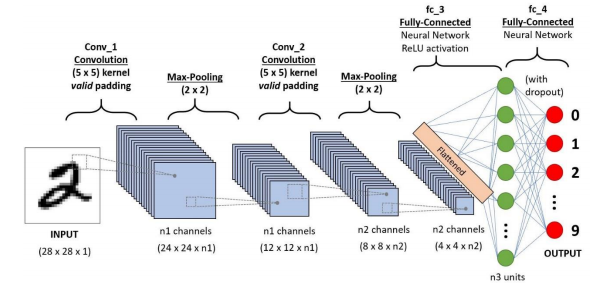
An image is a matrix of pixel values. A lot of times images contain objects that have pixel dependencies throughout the image. A CNN is able to successfully capture spatial dependencies in an image though the application of relevant filters and the network can be trained to understand the sophistication of the image better.

In image [2], an RGB image, which has been separated by its three color planes, is represented. Although this example has small dimensions, real images can reach higher dimensions, for example an 8K image has 7680x4320x3 dimensions, making object detection in such dimensions a computational intensive procedure. The role of CNN is to reduce the image into a form which is easier to process the image, but at the same time without losing features which are critical for getting good predictions.



**Image [2]: An RGB image**

ConvNets, usually, are divided into two parts, the convolutional and the densely connected. The first one applies various layers such as Convolution and Pooling to reduce the dimensions and retain the important features of the image, while the second one is responsible for classification. In the following image [3], an example of a CNN architecture is shown.



**Image [3]: A 4 layer CNN.**

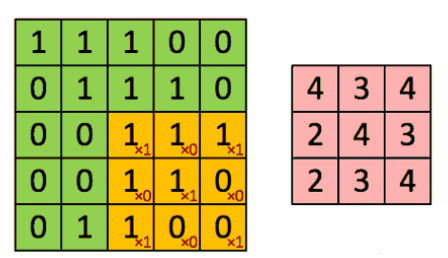
**2.3.1) Convolution Layer**

In a ConvNet, the input is an image (tensor) with a shape: (H) x (W) x (C), representing height, width and number of channels respectively. After passing the input through the convolutional layer, the image becomes abstracted to a feature map, with new shape: (Feature Map Height) x (Feature Map Width) x (Feature Map Channels).

Generally, a convolutional layer has the following attributes/hyperparameters:

* Convolutional filters, also known as kernels.
* The number of input and output channels.
* Padding (augmentation of the kernel) and Stride (size of the step the kernel parses an image).

A convolutional kernel is basically a matrix that is applied throughout the image. Each filter is convolved across the width and height of the input image, computing the dot product between the filter entries and the input, resulting to a feature map of that filter. The network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

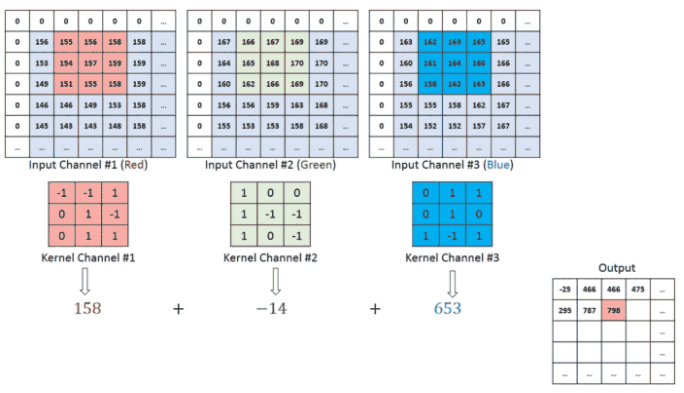


**Image [4]: Original and Convolved Image.**

In image [4], the kernel shifts 9 times in the orginal image, performing every time a matrix multiplication operation between the kernel and the portion of the image over which the kernel is hovering at the time. In this example, the filter parses the image with a stride of 1.

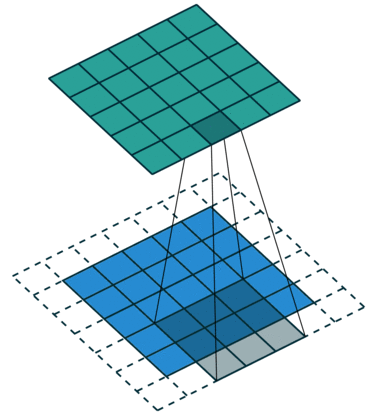
In cases of images with multiple channels such as RGB (Image [5]), the kernel has the same depth as that of the input image, and matrix multiplication is performed between the Kernels and each Channel. The results then are summed to give a squashed one-depth channel Convolved Feature Map.

The goal of Convolution operations is to extract high-level semantically rich features from the input image. One layer is not enough to achieve this, therefore CNNs need not to be limited to only one convolutional layer. The first layers are responsible for Low-Level features such as edges. With more depth in the network, the architecture adapts to the High-Level features as well, providing a network which understands the whole image.



**Image [5]: RGB example of convolution.**

To add more layers (depth) to the network, there are two types of operation. One in which the convolved feature is reduced in dimensionality (Valid padding) compared to the input, and the other in which the dimensionality remains the same or it is increased (Same padding).



**Image [6]: Same padding with zeros.**

The same padding operation that is shown in Image [6] has been achieved by augmenting the input image from 5x5x1 to 6x6x1 and then applying the 3x3x1 kernel over the augmented image. If the valid padding operations was performed, the convolved matrix will have the same dimensions with the kernel.

**2.3.2) Pooling Layer**

**3.Models**

Since the purpose of this diploma thesis is to compare models, the following algorithms were chosen.

**3.1 RetinaNet**

RetinaNet is a single, unified network composed of a backbone network and two task-specific subnetworks. The backbone, is responsible for computing a convolutional feature map over an entire input image and is an off-the-self convolutional network. More specifically, as the backbone of the RetinaNet the Feature Pyramid Network has been adopted. FPN is a standard convolutional network with a top-down pathway and lateral connections so the network efficiently constructs rich, multi scale feature pyramid from a single resolution input image. Each level of pyramid can be used for detecting objects at a different scale. Besides the backbone network, there are also two more subnetworks. The first one is responsible for object classification on the backbone’s output and the second subnet performs convolutional bounding box regression.

**3.1.1 Feature Pyramid Network**

Traditionally, in computer vision, featurized image pyramids have been used to detect objects with varying scales in an image. Featurized image pyramids are feature pyramids built upon image pyramids. This measns one would take an image and subsample it into lower resolution and smaller size images, thus forming a pyramid. Hand-engineered feaures are then extracted from each layer in the pyramid to detect objects. This makes the pyramid scale-invariant and the process is quite intensive in terms of computation and memory.

With the development of deep learning, these hand-engineered features were replaced by CNNs. Later, the pyramid itself was derived from the pyramidical hierarchical structure that CNNs have (for example SSD). In CNNs the output size of feature maps decreases after each successive block of convolutional operation, and forms a pyramidal structure.

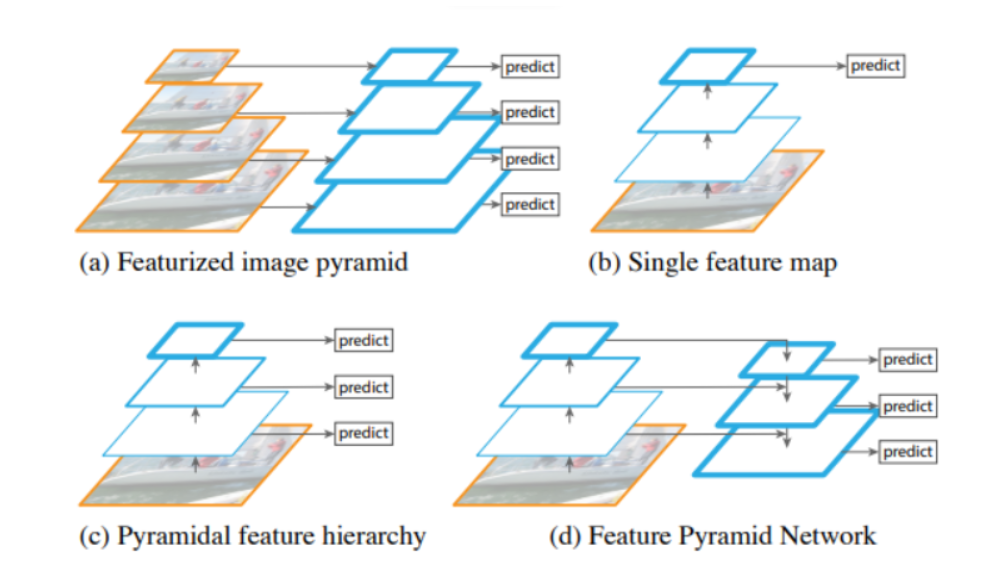


Image ?. Different types of pyramid architectures

Various architectures (Image ?) have been proposed that utilize the pyramid structure. In example (a), the Feature image pyramid that was discussed before is exhibited. As mentioned before, it is computing intensive. In example (b), the Single (scale) feature map is shown. Although it detects faster than (a), pyramids are still needed to get the most accurate results. Pyramidal feature hierarchy (c), utilize only semantically rich information, but it does not reuse the multi-scale feature maps from different layers and therefore small objects cannot be detected. Feature Pyramid Network or FPN (d), covers the flows of all the other architectures. FPN creates an architecture with rich semantics at all levels as it combines low-resolution semantically strong features, with high resolution semantically weak features. To achieve this, a top-down pathway with lateral connections to bottom-up convolutional layers has been created.

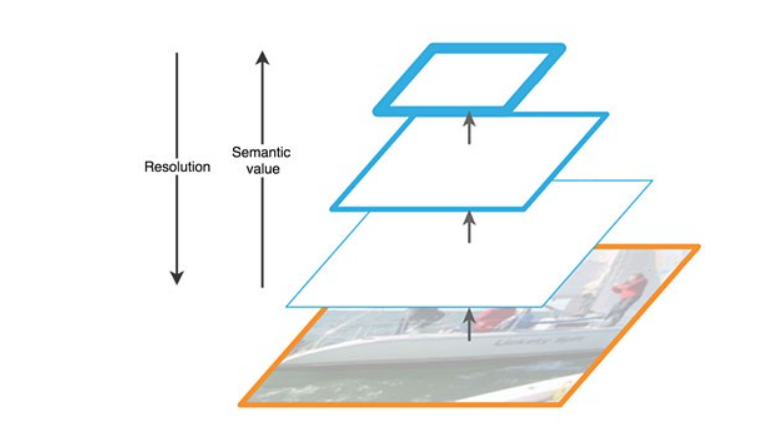


Image ?. Feature extraction in FPN

**3.1.2 Anchors**

When a Neural Network predicts multiple objects in a picture, the network is actually making thousand predictions and only showing the ones with the highest confidence of containing an object. In other words, the detector creates thousands of anchor boxes for each predictor that represent the ideal location, shape and size of the object it specializes in predicting. For each anchor box, it calculates which object’s bounding box has the highest overlap divided by non-overlap. This is called Intersection over Union or IOU. If the highest IOU surpasses a certain threshold, then it signals the anchor box that it should detect the object that gave the highest IOU.

**3.1.3 Classification/Regression subnetworks**

The classification subnet, predicts the probability of object presence at each spatial position for each anchor and for each object class. This subnet is a small Fully Connected Network, attached to each FPN level, and the parameters of this subnet are shared across all pyramid levels. The design is simple. Taking an input feature map with X channels from a given pyramid level, the subnet applies four 3x3 convolutional layers, each with filters equal to the number of the feature map channels, followed by ReLU activations, followed by a 3x3 convolutional layer, with the number of classes times the number of anchors, filters. Finally, sigmoid activations are attached to output the binary predictions per spatial location.

In parallel, a small FCN is attached to each pyramid level for the purpose of regressing the offset from each anchor box, to a nearby ground-truth object, if one exists. The design is identical to the classification subnet, with the exception that it terminates in (4xNumberOfAnchors) linear outputs per spatial location. For each anchor per spatial location, these 4 outputs predict the relative offset between the anchor and the ground-truth box.

**3.1.4 RetinaNet Architecture**

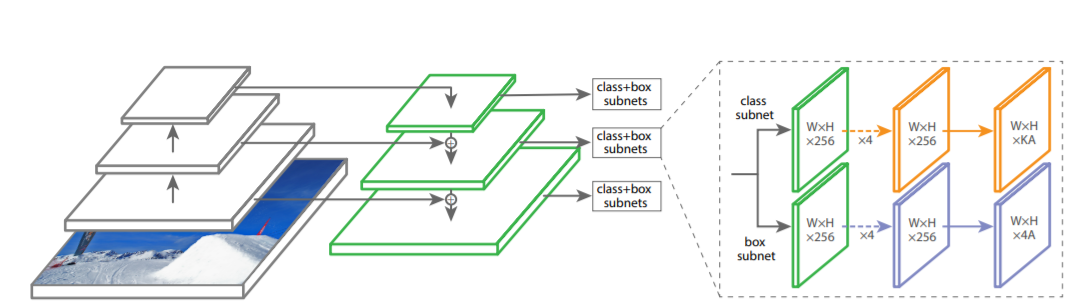
The one stage RetinaNetwork architecture as previously stated, uses a Feature Pyramid Network on top of a feedforward ResNet architecture as backbone, in order to generate a rich, multi-scale convolutional feature pyramid.

Image ?. RetinaNet architecture.

ResNet is used to construct the bottom-up pathway. It composed of many convolution modules, each with many convolution layers. Moving up, the spatial dimension is reduced by 0.5 or the stride is doubled. The output of each convolution module is labeled as Ci and later used in the top-down pathway (FPN).

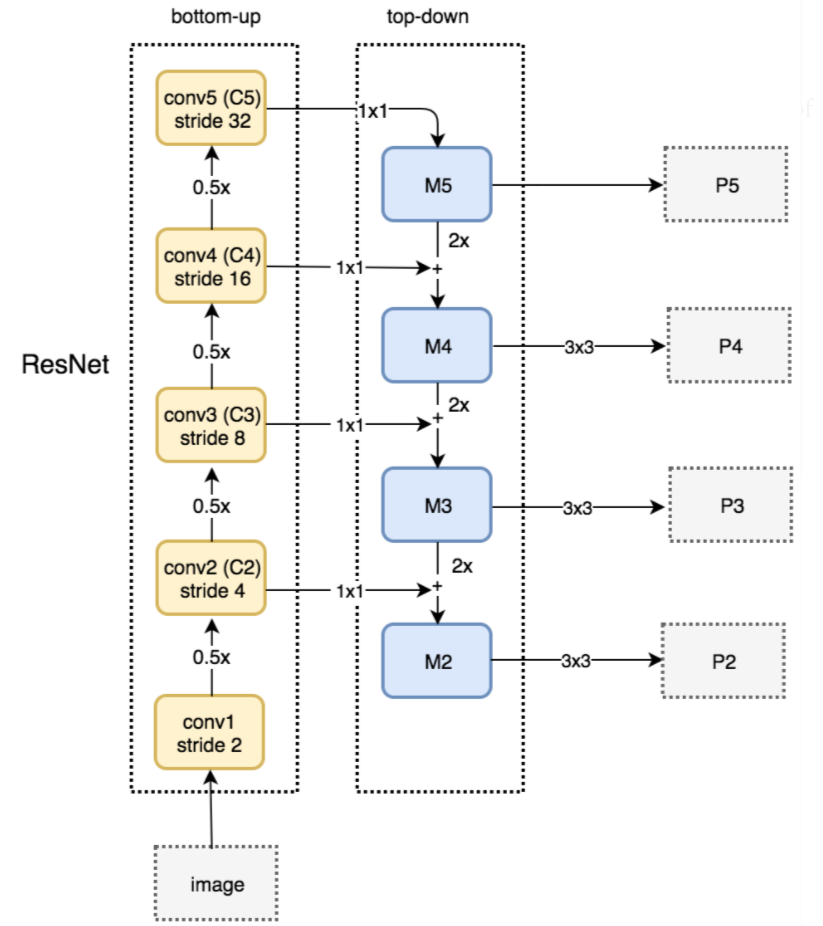


Image ?. Backbone architecture.

For the top-down pathway, a 1x1 convolution filter is applied to reduce the convolutional modules depth and create the feature map layers. As we go down the top-down path, the previous layers are getting upsampled by 2, using nearest neighbors upsampling. Again, a 1x1 convolution is applied to the corresponding feature maps in the bottom-up pathway. Then they are getting added element-wise. Finally, a 3x3 convolution merges all layers. This filter reduces the aliasing effect when merged with the upsampled layer. This process, stops for Conv1 module, because the spatial dimensions are too large and will slow down the process too much. Furthermore, since the classification and box regressor subnets, share the same inputs from the feature maps outputs, all pyramid feature maps have the same size output channels.

To this backbone, the RetinaNet attaches two subnetworks that were described in section 3.1.3, one for classifying anchor boxes, and one for regressing from anchor boxes to ground-truth object boxes.

The design of the Network is simple but effective, because allows to focus more on optimization, using Focal Loss, an enhancement over Cross-Entropy Loss that eliminates the accuracy gap between RetinaNet, and state-of-the-art two-stage detectors like Faster R-CNN with FPN, while maintain a low inference time.

**3.1.5 Loss Function**

Single stage models suffer from an extreme foreground-background class imbalance problem due to dense sampling of anchor boxes. In RetinaNet, at each pyramid layer there can be thousands of anchor boxes, but only a few will be assigned to a ground-truth object, while the vast majority will be background class. These easy examples or detections with high probabilities, although resulting in small loss values can collectively overwhelm the model. Focal loss reduces the loss contribution from easy examples and increases the importance of correcting misclassified examples.

Starting from the cross entropy (CE) loss for binary classification:

In the above specifies the ground-truth class and is the model’s estimated probability for the class with label . Changing the notation for convenience and defining pt:

Now the CE can be rewritten as:

A common method to address the class imbalance is to introduce a weighting factor for class 1 and 1-*α* for class -1. The weighting factor can be treated as a hyperparameter to set by cross validation. Again for notational convenience the *α-balanced* CE loss function is defined as:

As [writters] suggest, the large class imbalance encountered during training of dense detectors, overwhelms the cross-entropy loss. Easily or well classified negatives, comprise the majority of the loss and dominate the gradient. While *α* balances the importance of positive/negative examples, it does not differentiate between easy/hard examples. Therefore, a reshaped form of CE is proposed to down-weight easy examples and thus focus training on hard negatives.

By adding a modulating factor to the cross-entropy loss, the focal loss is defined:

The focal loss is visualized in the next figure for values of gamma between zero and five. It is important to state that if gamma equals to zero, then the focal loss function behaves like cross-entropy. Besides that, when an example is misclassified and pt is small, the modulating factor is near 1 and the loss is unaffected. As pt tends to become 1, the factor goes to 0 and the loss for well-classified examples is down-weighted.

In practice, an *α-balanced* form of the focal loss is preferred.

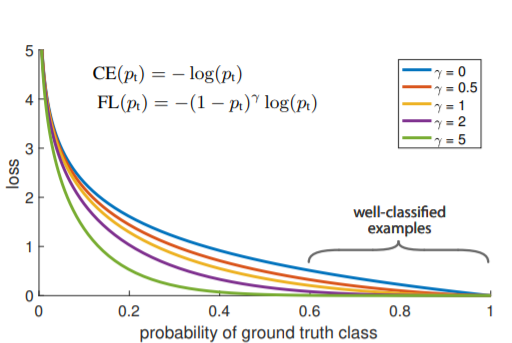


Image ?. Focal Loss.

**4. Metrics**

**4.1 Intersection over Union (IOU)**

Intersection over Union is measured based on Jaccard Index, also known as the Jaccard similarity coefficient (a statistic used for gauging the similarity and diversity of sample sets), that evaluates the overlap between two bounding boxes. It requires a ground truth bounding box and a predicted bounding box. Using IOU the detection can be characterized as valid (True positive) or not (False Positive).

To be more accurate, IOU is given by the overlapping area between the predicted bounding box (Bp) and the ground truth bounding box (Bgt) divided by the area of union between them:

In order to categorize a detection as valid, one can assume the existence of a threshold, that the IOU needs to exceed.

* **True Positive (TP):** A correct detection, with .
* **False Positive (FP):** A wrong detection, with .
* **False Negative (FN):** A ground truth that was not detected.
* **True Negative (TN):** It represents corrected misdetections. In object detection tasks there are many possible bounding boxes that should not be detected within an image; thus, TN would be all those possible bounding boxes. That is why, in object detection TN has no meaning and it is not used as a metric.

In the upcoming images, the visualization of IOU can be seen.

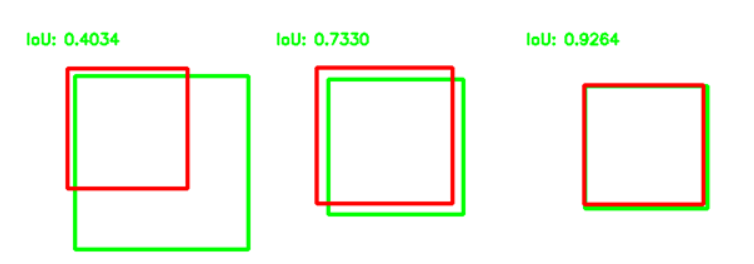


Image ?. IOU visualization.

**4.2 Classification Accuracy**

The simplest metric and is defined as the number of correct predictions divided by the total number of predictions.

Since the TN do not count as a metric in object detection, the above equation transforms into:

**4.3 Average Precision**

There are many cases in which classification accuracy is not enough to indicate if a model’s performance is good. One scenario is when there is an imbalance in the class distribution (one class is more frequent than the other). In this case, even if the model predicts every sample as the most frequent class, it will get a high accuracy rate, which is not learning rather a blind prediction model.

Therefore, a class specific performance metric is vital for the evaluation of a model. Precision or average precision, is one of such metrics, which is defined as:

**4.4 Recall or Sensitivity or True Positive Rate**

Recall measures how good the model finds all the positives, and it is defined as:

**4.5 F1 Score**

F1 score is the harmonic mean (HM) of precision and recall.

**5. Data**

For the purpose of our experiments, the Stanford Drone Dataset, from the Computational Vision and Geometry Lab of the University of Stanford. The dataset was developed to help with the development of target tracking or trajectory forecasting algorithms.

It consists of eight unique scenes with a total of 6 different classes (Bicyclists, Pedestrians, Skaterboarder, Cart, Cars, Bus).



Image ?. Stanford Drone Dataset.

In my experiments, I renamed the Pedestrian class in to Person and also kept the class Car. Therefore, the classes Biker, Skater, Cart and Bus were excluded.

After taking various screenshots and using the label maker *labelImg*, nearly 1500 annotations were taken, out of which 1136 were used for training, and the remaining for evaluation. From those 1136 annotated data the 80% belong to Person class and the remaining to Car class.

Since the deep learning models need only the bounding box coordinates, there was no pre-process phase on the data.

(Changes and additions need to be done here if and when a ML model is added)

**6. Experiments**

**6.1 RetinaNet implementation**

To create from scratch a Neural Network is a very challenging task, therefore a pre-made library in Python by the company Fizyr was used. The library contains all the necessary tools for training, testing and evaluation of a RetinaNet model.

All the training phase, took place in Google Colab. There were three different approaches of the problem.

1. The first one was a random approach. There was no hyper-parameter tuning, no transfer-learning. The model was trained for 100 epochs, with a step size of 1000 and batch-size equal to 1.
2. The second approach was pretty similar to the first one. There was again no hyper-parameter tuning, but weights from a pre-trained, in the COCO dataset, model were used.
3. For the third approach a more technical approach was followed. The hyper-parameters depend from the GPU card and the dataset. Therefore, in order to select the batch size, I randomly started from 8 and depending if the model started training, I increased or decreased that number by power of two. In the Tesla T4 GPU with this particular dataset, the batch-size that can initiate training was 2. For a batch-size of two the step size needed to be 428 (bigger than this number created an interruption to the system). Finally, the RetinaNet converges quick, therefore once the loss of the system started to grow the training was completed. If I left the model to train more, then the system will be overfitted. As result, there would be no predictions, rather blind suggestions. Another addition to this training approach was a configuration file, in order to be able to detect bounding boxes with smaller size than 32x32, which was the default smallest possible anchor box. Finally, the weights of a pre-trained, in the COCO dataset, model were used.

The following matrix summarizes the above. Every row, represents a different approach, while every column represents a hyper-parameter or the existence of a pre-trained model to help with the training.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Batch-size | Steps | Epochs | Transfer-Learning | Configuration file |
| 1st approach | 1 | 1000 | 100 | No | No |
| 2nd approach | 1 | 1000 | 66 | Yes | No |
| 3rd approach | 2 | 428 | 50 | Yes | Yes |

Matrix ?. Parameters of the different approaches.

**6.2 Next implementation**

**7. Results**

**7.1 RetinaNet results**

To conduct the experiments, a confidence threshold has been defined. The confidence threshold represents the probability of a certain class to be detected by the model. If the probability exceeds the confidence threshold, and the IOU is greater than its own threshold, then the possible object gets correctly detected. In other words, IOU gives the location of a possible object and the confidence gives the probability of a certain class to be that detected object.

**7.1.3 3rd approach**

|  |  |
| --- | --- |
|  | **50 epochs, 2 batch-size, 428 steps, transfer learning, smaller anchors** |
| **Confidence threshold** | 0.1 |
| **IOU threshold** | 0.6 |
| **Accuracy** | 0.97 |
| **Precision** | 0.294 |
| **Recall** | 0.972 |
| **F1\_score** | 0.451 |

Matrix ?. Results for confidence threshold = 0.1 and IOU greater than 0.6.

|  |  |
| --- | --- |
|  | **50 epochs, 2 batch-size, 428 steps, transfer learning, smaller anchors** |
| **Confidence threshold** | 0.5 |
| **IOU threshold** | 0.6 |
| **Accuracy** | 0.936 |
| **Precision** | 0.72 |
| **Recall** | 0.94 |
| **F1\_score** | 0.81 |

Matrix ?. Results for confidence threshold = 0.5 and IOU greater than 0.6.

|  |  |
| --- | --- |
|  | **50 epochs, 2 batch-size, 428 steps, transfer learning, smaller anchors** |
| **Confidence threshold** | 0.7 |
| **IOU threshold** | 0.6 |
| **Accuracy** | 0.85 |
| **Precision** | 0.79 |
| **Recall** | 0.85 |
| **F1\_score** | 0.78 |

Matrix ?. Results for confidence threshold greater than 0.7 and IOU greater than 0.6.

|  |  |
| --- | --- |
|  | **50 epochs, 2 batch-size, 428 steps, transfer learning, smaller anchors** |
| **Confidence threshold** | 0.8 |
| **IOU threshold** | 0.6 |
| **Accuracy** | 0.81 |
| **Precision** | 0.73 |
| **Recall** | 0.78 |
| **F1\_score** | 0.75 |

Matrix ?. Results for confidence threshold greater than 0.8 and IOU greater than 0.6.

|  |  |
| --- | --- |
|  | **50 epochs, 2 batch-size, 428 steps, transfer learning, smaller anchors** |
| **Confidence threshold** | 0.8 |
| **IOU threshold** | 0.4 |
| **Accuracy** | 0.8 |
| **Precision** | 0.74 |
| **Recall** | 0.78 |
| **F1\_score** | 0.75 |

Matrix ?. Results for confidence threshold = 0.8 and IOU greater than 0.4.