Technical University of Crete

School of Electrical and Computers Engineering

Diploma Thesis

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

TRIMAS CHRISTOS

CHANIA, SEPTEMBER 2022

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 my studies.

Abstract

Unmanned Aerial Vehicles (UAVs) have experienced great growth and as of 2020 at least 100 countries use UAVs in tactical missions, while at the same time even more commercial applications deploy drones, for example photography and filmmaking, smart crops, smart cities, emergency handling, drug delivery, traffic management, etc. The big success of UAVs came due to the huge growth of electronics and the revolution of data. One of the most popular application of drones is object detection before designing the planned operation, e.g. differentiate pedestrians from cars or bikes in cross-road management systems. Deep Learning algorithms have been proven to be the best solution in such kind of problems. This diploma thesis collects and studies some of the most well-known detection systems, it analyzes in theory and in practice an object detector, the famous single-stage detector RetinaNet. Furthermore, a modified model is proposed that utilizes more Convolutional Blocks and combines features from different levels of the Neural Network. The extra convolution block is a mirror of the Feature Pyramid Network; therefore, the new model is called "Two-Phase Feature Pyramid Network Retina". Since the goal is to compare those models, the classic RetinaNet and the modified model, were trained and tested using the Stanford Drone Dataset, a dataset designed to train object detectors for UAVs. The modified model achieves an accuracy score 6% higher than the baseline model, and it seems to outperform the original model in every metric, such as Precision, Sensitivity and F1 score. Finally, both the original and the modified Retina, were compared with other well-known object detectors such as YOLO, Faster RCNN, SSD, etc. The proposed architecture seems to outperform almost every object detector from the literature in terms of mean Average Precision. In conclusion, the modified model can be used to detect small objects in applications where accuracy is a critical factor.

Σύγκριση Συστημάτων Τεχνητής Νοημοσύνης για τον Εντοπισμό Αντικειμένων σε Εικόνες από μη-Επανδρωμένα.

ΤΡΙΜΑΣ ΧΡΗΣΤΟΣ

Περίληψη

Τα μη επανδρωμένα ιπτάμενα οχήματα έχουν παρουσιάσει τεράστια εξέλιξη και από το 2020 τουλάχιστον 100 χώρες χρησιμοποιούν μη επανδρωμένα σε τακτικές αποστολές. Ταυτόχρονα χρησιμοποιούνται ολοένα και περισσότερο σε εμπορικές εφαρμογές, όπως η φωτογραφία, οι έξυπνες καλλιέργειες, έξυπνες πόλεις, επιχειρήσεις διάσωσης και διαχείρισης κρίσεων, αποσυμφόρηση οδικών αρτηριών κλπ.. Η μεγάλη επιτυχία των μη επανδρωμένων οχημάτων ήρθε με την εξέλιξη των ηλεκτρονικών και την επανάσταση των δεδομένων που χρησιμοποιούνται σε έξυπνα συστήματα αποφάσεων. Μια από τις πιο διαδεδομένες εφαρμογές των drones είναι η αναγνώριση αντικειμένων για τον ακριβή σχεδιασμό της ολικής λειτουργίας του συστήματος, π.χ. διαχωρισμός των πεζών από τα αυτοκίνητα και τις μηχανές σε συστήματα διαχείρισης της κυκλοφορίας. Οι αλγόριθμοι βαθιάς μηχανικής μάθησης έχουν αποδειχθεί ως η καλύτερη λύση σε προβλήματα τέτοιου τύπου. Η παρούσα διπλωματική εργασία συλλέγει και μελετά μερικά από τα πιο γνωστά μοντέλα αναγνώρισης αντικειμένων, γίνεται μια πλήρης ανάλυση τόσο σε θεωρητικό, όσο και σε πρακτικό επίπεδο, ενός ανιχνευτή αντικειμένων, του γνωστού RetinaNet. Επιπλέον, προτείνεται μια τροποποίηση του RetinaNet με προσθήκη επιπλέον συνελικτικών μπλοκ και συνδυασμό features από διαφορετικά επίπεδα του Νευρωνικού Δικτύου. Καθώς ο στόχος είναι η σύγκριση αυτών των μοντέλων, τόσο το κλασσικό RetinaNet, όσο και το τροποποιημένο μοντέλο, εκπαιδεύονται και ελέγχονται πάνω στο Stanford Drone Dataset, ένα dataset για την εκπαίδευση μοντέλων αναγνώρισης αντικειμένων από μη επανδρωμένα. Το τροποποιημένο μοντέλο πετυχαίνει αύξηση ακρίβειας από το κλασσικό μοντέλο της τάξης του 6%, ενώ αποδίδει καλύτερα σε όλα τα επίπεδα σύγκρισης ως προς την ακρίβεια, την ευαισθησία και το F1 score. Τέλος, γίνεται μια σύγκριση τόσο του RetinaNet, όσο και του modified RetinaNet με άλλους γνωστούς ανιχνευτές αντικειμένων, όπως τα YOLO, Faster RCNN, SSD, κλπ και γενικά η προτεινόμενη αρχιτεκτονική φαίνεται να ξεπερνά σχεδόν όλα τα άλλα μοντέλα ως προς τον μέσο όρο ακρίβειας. Συμπερασματικά, το προτεινόμενο μοντέλο φαίνεται να είναι κατάλληλο για ανίχνευση μικρών αντικειμένων, όπου η ακρίβεια αποτελεί το κυριότερο κριτήριο επιλογής ανιχνευτή.

Contents

List of Tables

List of Algorithms

List of External Resources

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 referred to as drones, have experienced the greatest growth. As of 2020, at least seventeen countries have armed UAVs, and more than 100 countries will be using UAVs in a military capacity by 2021. 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].

Figure 1: RQ-4 Global Hawk.

Whether it comes to the detection of objects of interest (refugee waves, tracking target), prison surveillance or information gathering of a battlefield, UAVs have proven their usefulness. For example, the US Air Force uses large UAVs for strategic reconnaissance, such as the RQ-4 Global Hawk (Figure 1), a 13-meter-long jet that carries a variety of sensors, radars and photographic sensors.

A significant contribution to the development of UAVs, 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 and Algorithmic Solutions

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.

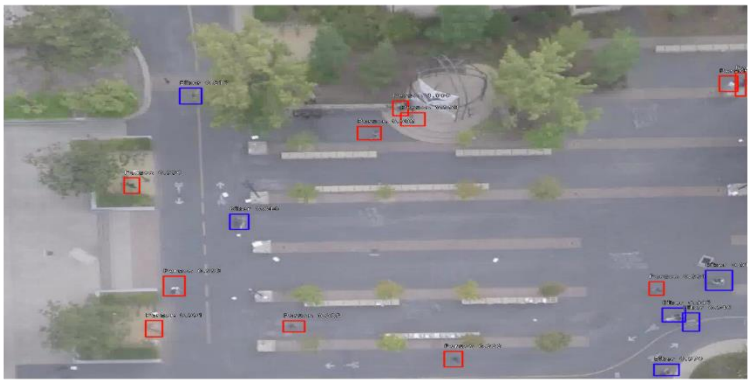


Figure 2 Object Detection Example

With the evolution of cameras and the oversimplification of data gathering and processing, object detection can be used in the following 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 object's 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 detector using Haar features.**
- **Scale-Invariant feature transformation.**
- **Histogram of oriented gradients.**

In the last few years, due to the revolution of Data, most reliable object detectors fall in the category of Deep Learning systems. 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:

1) Regional Proposal Networks like R-CNN or Faster R-CNN: Regional proposal algorithms are a family of deep learning algorithms. They are two-stage detectors, meaning in the first stage of detection, an algorithm like Selective Search or a neural network locates the proposing areas of a detection. The second stage is responsible for feature learning and classification/regression. R-CNN was the first two-stage architecture that was introduced for object detection. Fast R-CNN and Faster R-CNN are modern architectures that rely on the R-CNN model, but they perform faster and with significant more accuracy. Region-based networks have been used for tracking objects from a UAV-mounted camera, locating text within an image and enabling object detection in Google Lens.

2) Single Shot MultiBox Detector: While Faster R-CNN is considered the state of the art in terms of accuracy in object detection, the whole process of detection is very slow when it comes to real-time applications. Therefore, SSD was introduced to speed up the process by eliminating the need for region proposal networks. The SSD is a single-stage architecture, which means the tasks of object localization and classification are done in a single pass of the network. The SSD has two components: a backbone model and SSD head. Backbone model usually is a pre-trained network like VGG that works as a feature extractor. The SSD head are convolutional layers added to the backbone to interpret bounding boxes and classes of objects. SSD is used in a variety of applications such as smart crops, crowd counting and many more.

3) You Only Look Once (YOLO): Prior work on object detection repurposes classifiers and localizers to perform detection, instead the YOLO architecture frames object detection as a regression problem to spatially separated bounding boxes and corresponded class probabilities. YOLO network is the most famous architecture so far and large companies, such as Tesla, are utilizing the model for a variety of applications. YOLO can outperform R-CNN by 1000x.

4) Retina-Net: RetinaNet is one of the best single-stage object detectors that has proven to work well with dense and small-scale objects. Its aim is to improve the Feature Pyramid Network by simply applying a sophisticated loss function called Focal Loss. Like SSD it utilized pre-trained backbone networks for feature learning and has two smaller fully Convolutional networks for classification and box regression. RetinaNet is widely used for detection of objects in aerial or satellite imagery.

1.3) Motivation and Contribution

Unlike fixed position cameras, UAV surveillance has higher mobility and can cover larger areas for surveillance. Contrary to traditional object detection systems, drones have to deal with a huge amount of data, imbalanced datasets and many tiny object instances. Apart from the data and their sizes that may vary, depending the application of the UAV the spatial resolution might vary, making the detection an even more challenging problem. As previously mentioned, considerable work has been done in object detection by large companies and big academic institutions. RetinaNet is a Deep Learning architecture designed by Facebook A.I. Research team, and introduces a new Loss Function to train a system purely designed by simpler Convolutional Models. The Focal loss as they named it, had a huge success especially in imbalanced datasets, as it down-weighted the importance of easy examples and gave more attention in hard examples. The simplicity of the architecture and the introduction of Focal Loss, made RetinaNet the perfect "real-time" object detection system to deploy in a UAV, due to the small resources it required compared to other models. In this diploma thesis the Retina model is being tested in real world data from the Stanford Drone Dataset. After analyzing and testing the architecture, this work aims to further explore the potential of the model by introducing more complexity and depth to a rather simple architecture. Therefore, an expansion of the base model is introduced by using a two-phase mirror Feature Pyramid Network. The new model is also trained and tested in the Stanford Drone Dataset with the same parameters as the base-model in order to compare the two in equal terms. To further evaluate the modified model, a 5-fold cross validation is performed in both the original and the modified RetinaNet. The introduction of the two-phase FPN seems to provide significant higher accuracy than the base-model, a crucial factor for applications like surveillance or traffic management.

2. Theoretical Background

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 computer systems, that were designed to think and act like humans. The term may additionally be applied to any machine that exhibits traits related with the 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.

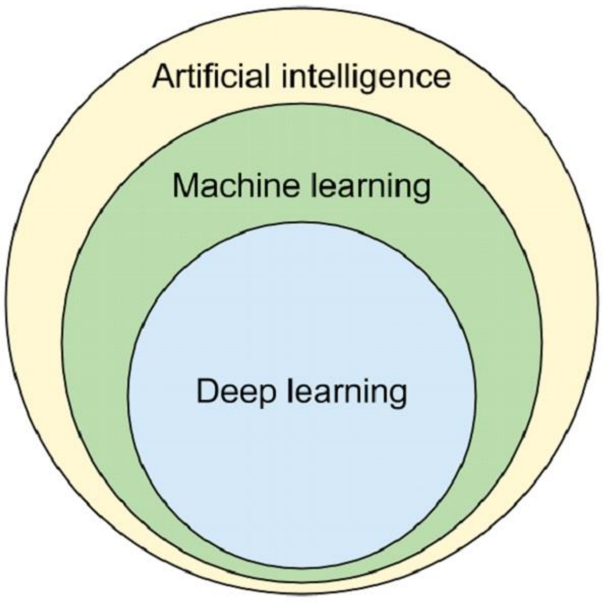


Figure 3: A.I., M.L. and D.L.

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 gives the data with the answers and demands from the machine to create the rules. The rules can then be applied to 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 an element of a broader family of machine learning algorithms supported by 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 proportional to that of the connectivity pattern of Neurons in the Human Brain and the inspiration came from 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 through the application of relevant filters and the network can be trained to understand the sophistication of the image better.

In image [4], 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.

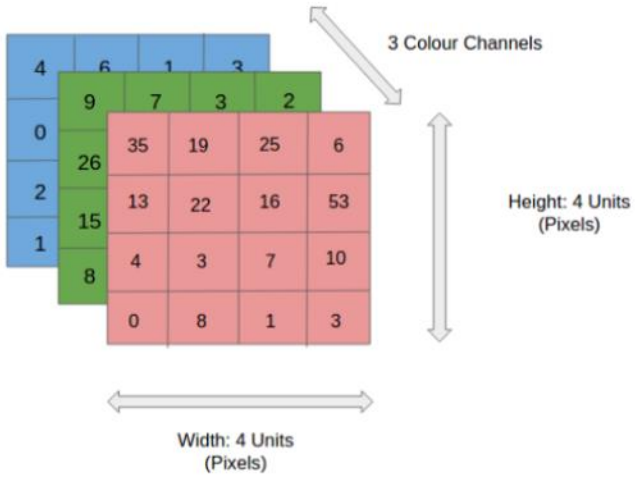
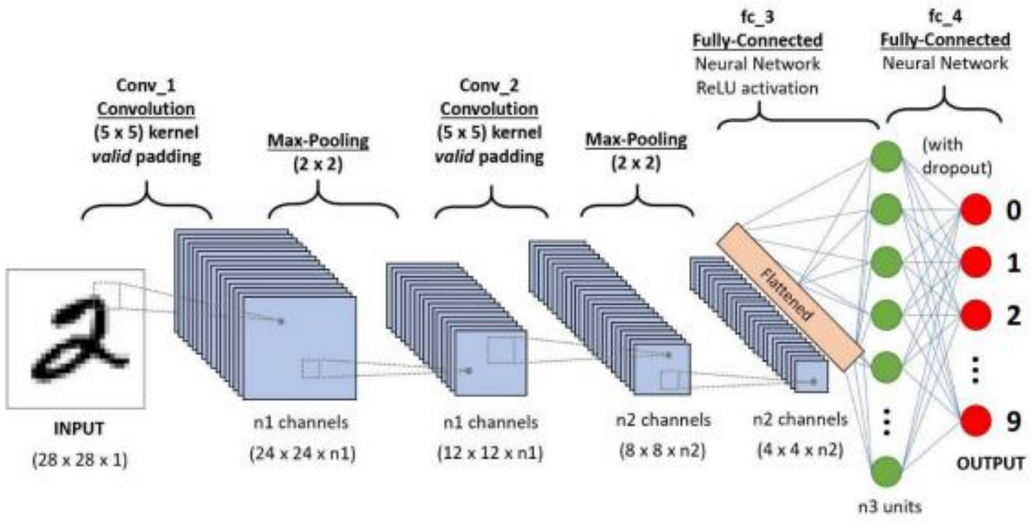


Figure 4: An RGB Image

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 [5], an example of a CNN architecture is shown.



Conv_1 Convolution (5 x 5) kernel valid padding
Max-Pooling (2 x 2)
Conv_2 Convolution (5 x 5) kernel valid padding
Max-Pooling (2 x 2)
fc_3 Fully-Connected Neural Network ReLU activation
fc_4 Fully-Connected Neural Network
(with dropout)
Flattened
INPUT (28 x 28 x 1)
n1 channels (24 x 24 x n1)
n1 channels (12 x 12 x n1)
n2 channels (8 x 8 x n2)
n2 channels (4 x 4 x n2)
n3 units
0
1
2
9
OUTPUT

Figure 5: A 4 layer CNN

2.3.a) 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 particular types of features at some spatial position of the input.

Given a two-dimensional Image **I** as input and a two-dimensional kernel **K** the convolution operation can be described [2]:

$$\boldsymbol{S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(m,n)K(i-m, j-n)}$$

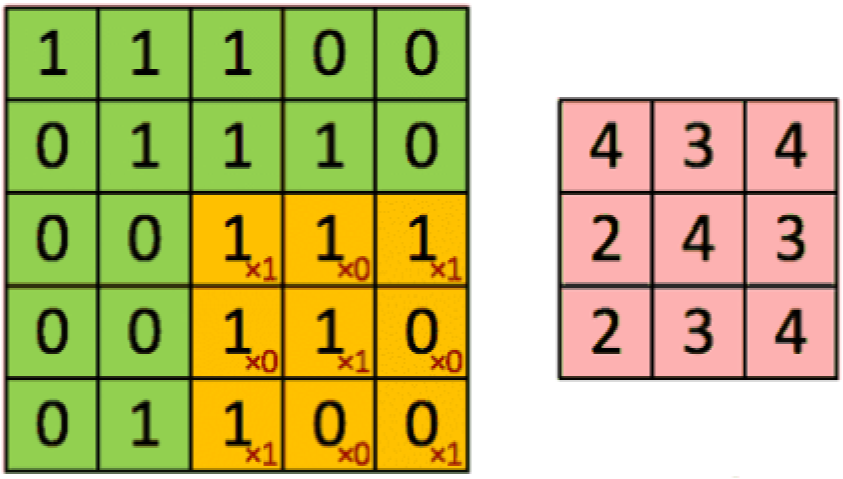


Figure 6: Original and Convolved Image

In image [6], 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 [7]), the kernel has the same depth as that of the input image, and matrix multiplication is perfomed 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.

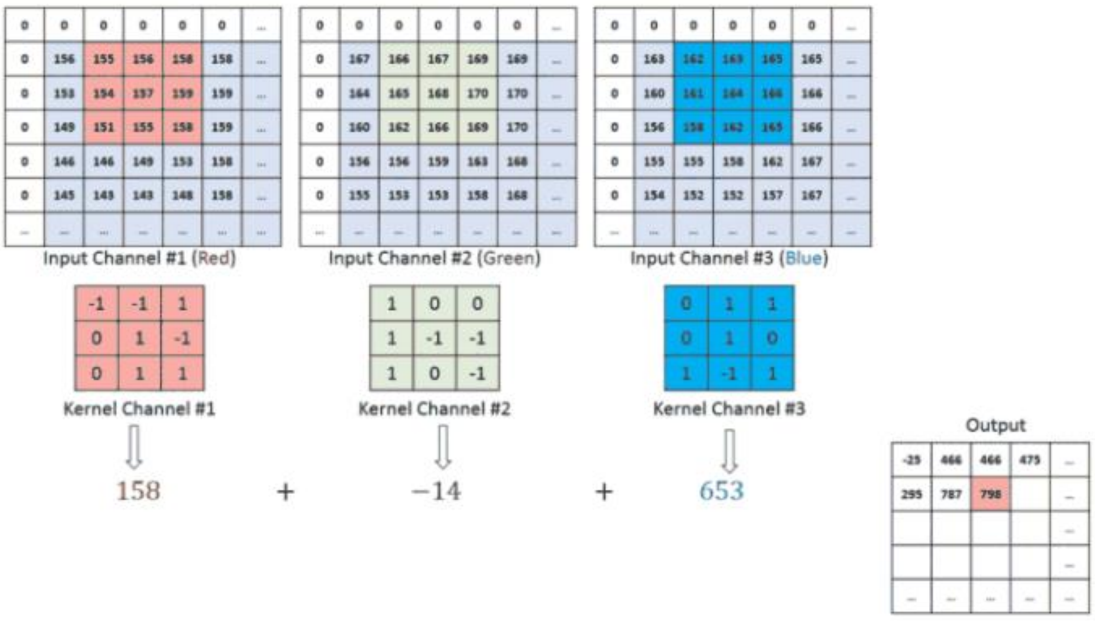


Figure 7: RGB example of convolution

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).

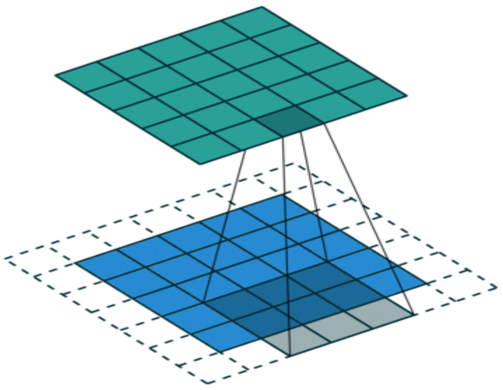


Figure 8: Same padding with zeros

The same padding operation that is shown in Image [8] 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 were performed, the convolved matrix will have the same dimensions with the kernel.

2.3.b) Pooling Layer

Similar to the convolutional layer, the Pooling layer is responsible for reducing the spatial size of the convolved feature. The goal of pooling layer is to decrease the computational power required to process the data and to extract dominant features through dimensionality reduction.

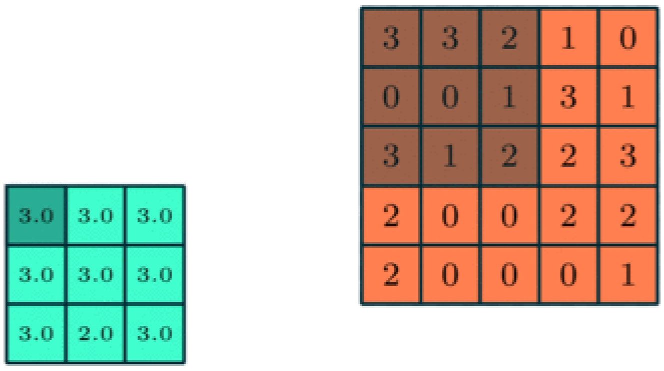


Figure 9: An example of pooling

Most common types of Pooling operations: Max Pooling and Average Pooling.

Max pooling returns the maximum value from a portion of the image I covered by a kernel K. It can be used as a Noise Suppressant, discarding the boisterous activations altogether and hence performing de-noising and dimensionality reduction at the same time.

Average pooling returns the average of all the values from the portion of the image I covered by a kernel K and as result performs dimensionality reduction.

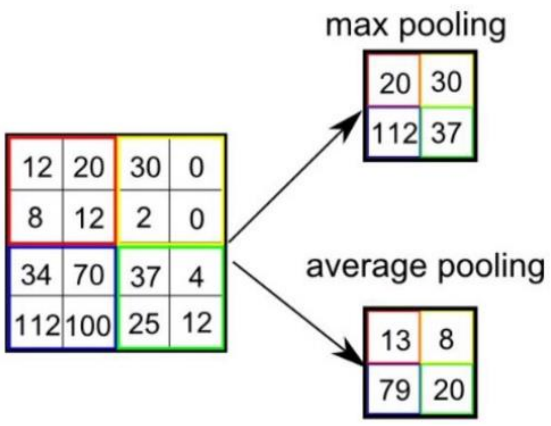


Figure 10: Avg and Max pooling

After repeating convolution and pooling layers for several times, the model will successfully understand low and high level features. The final step is to feed those features to either an Artificial Neural Network or use another technique to perform classification.

2.3.c) Fully Connected Layer

Adding a Fully-Connected (FC) layer is a cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolution and pooling layers. In more details, the input to the FC layer is the output from the final convolutional or pooling layer, which is flattened and then fed into the fully connected layer.

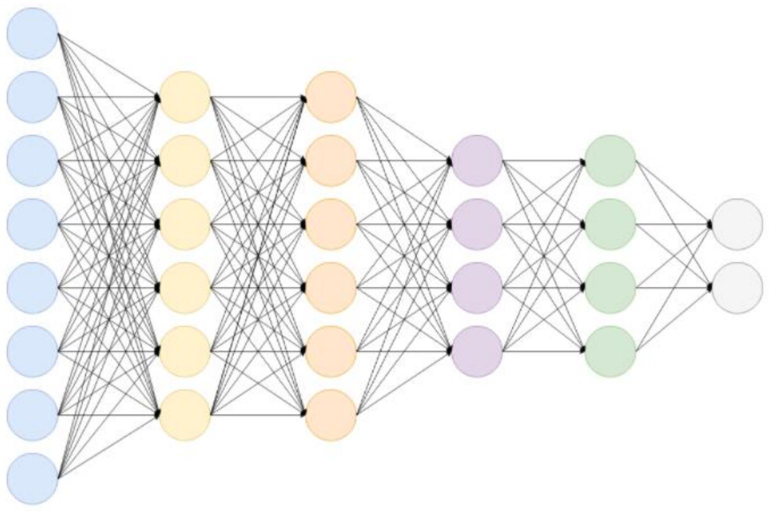


Figure 11: A Fully Connected Network

The output after performing convolutions and pooling layers is a 3-d matrix. To flatten the output, each value of the matrix is stacked and the result is huge vector. The flattened vector is then connected to fully connected layers which are Artificial Neural Nets. Each layer of the ANN applies the following function:

$$\boldsymbol{g(Wx + b)}$$

Where,

x is the input vector with dimension: ***d1 = (number of neurons, 1)***.

W is the weight matrix with dimensions:

d2 = (number of neurons in previous layer, number of neurons in the current layer).

b is the bias vector with dimensions: ***d3 = (number of neurons in current layer, 1)***.

g is the activation function.

After passing through the FC layers, the final layer uses an activation function (see next subsection) to get the probabilities of the input and classify them into a particular class.

2.3.d) Activation Functions

Also known as Transfer Function, is a way to extract the output of a node in an Artificial Neural Network. It maps the resulting values in a well-defined space, depending the function.

Activation functions can be basically divided into 2 types:

1. Linear Activation Functions.
2. Non-Linear Activation Functions.

Linear of Identity Activation Function:

The function is a line ranging between (-∞, ∞).

Equation: $f(x) = x.$

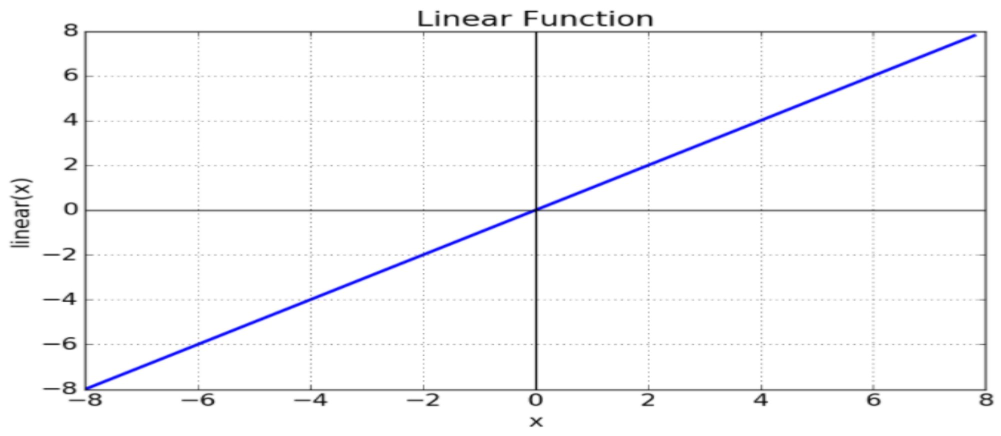


Figure 12: Linear Activation Function

Sigmoid or Logistic Activation Function:

A sigmoid function ranges between (0, 1), therefore making it useful in models that predict probabilities as an output. The function is differentiable and monotonic.

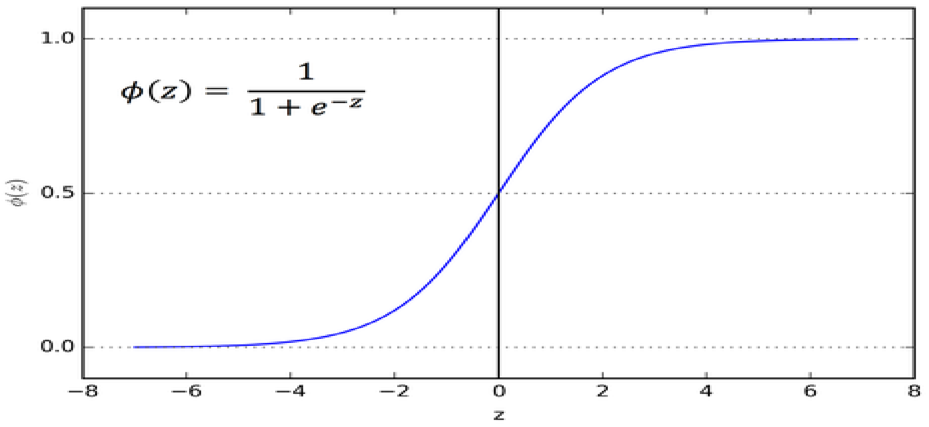


Figure 13: Sigmoid Function

A huge disadvantage is that the logistic function can cause a neural network to get stuck during the training phase. This is because, if a strongly-negative input is provided, it outputs the value very near to zero. This behavior is slowing down the update of the learnable parameters, such as weights and bias.

Equation: $\boldsymbol{\varphi(z) = 1/(e^{-z} + 1)}$

Rectified Linear Unit (ReLU) Activation Function:

This function maps every negative value immediately to zero. It ranges from zero to infinity, and the function and its derivative are monotonic.

Equation: $\boldsymbol{R(z) = max(0, z)}$

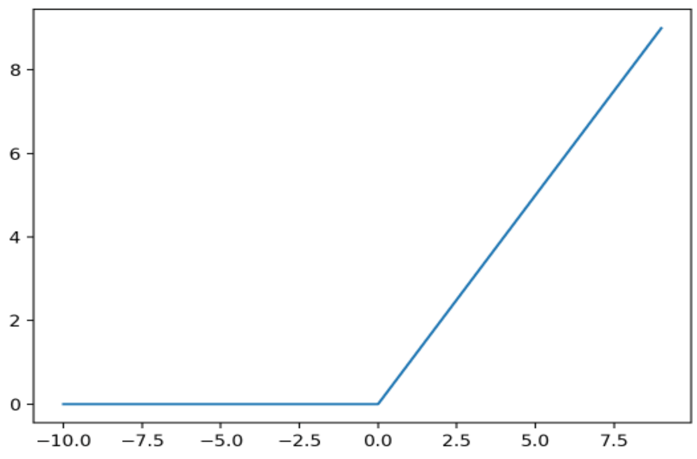


Figure 14: ReLU Activation Function

Softmax Activation Function:

Softmax maps the output in range between [0, 1]. Furthermore, the total sum of the mapped output is 1. Therefore, the output of Softmax is a probability distribution.

Equation:

$$\boldsymbol{\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}}$$

For j = 1,…,K.

2.3.e) Loss Functions

In Neural Networks, the term Loss refers to the prediction error of the Neural Network. The calculation of the loss with the use of a function is called Loss Function. Loss function is responsible for the update of weights of the Neural Network.

There are various Loss functions and the selection of the one that fits the best to a model depends on various factors, such as the type of problem (Regression or Classification), the model architecture and many more.

Few of the most known Loss Functions are:

Cross Entropy: One of the most known loss functions, it measures the performance of a classification model whose output is a probability value.

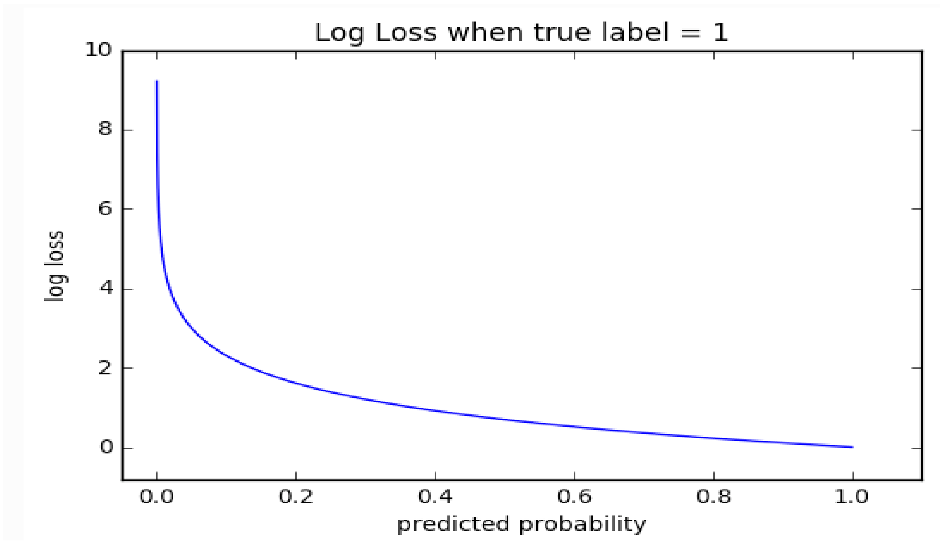


Figure 15: Cross-entropy loss

Cross-entropy can be described by the following equation

$$\boldsymbol{L(\Theta) = -\sum_{i=1}^{M} y_i \log(\hat{y}_i)}$$

where, M is the number of the classes, log the natural logarithm, y the binary indicator (0 or 1) if class label i is the correct classification and y_hat is the predicted label.

Kullback-Leibler divergence: Also called *relative entropy*, it is the gain or loss of entropy when switching from distribution one to distribution two – and it allows to compare the two distributions.

KL divergence is primarily used in Variational Autoencoders or in multiclass classification scenarios. Mathematically can be described by the following equation:

$$KL(P||Q) = \sum p(X) log\,(\frac{p(X)}{q(X)})$$

Smooth L1: Mostly used in object detectors for bounding box regression.

After defining L1 or Manhattan distance:

$$L_1 := (a, b, N) \to \sum_{i=1}^{N} |a[i] - b[i]| \Rightarrow (a, b, N) \to \sum_{i=1}^{N} |a_i - b_i|$$

The smooth L1 can be defined:

$$smooth_{L1} := (x) \to piecewise(|x| < 1, 0.5x^2, |x| - 0.5) \Rightarrow$$

$$x \to piecewise(|x| < 1, 0.5x^2, |x| - 0.5)$$

where x is the Manhattan distance between 2 vectors.

From the above equation the smooth L1 can be re-written:

$$sL1 = \begin{cases} |x| - 0.5, & if\ \ |x| > 1 \\ 0.5x^2, & else\,if\ \ |x| \leq 1 \end{cases}$$

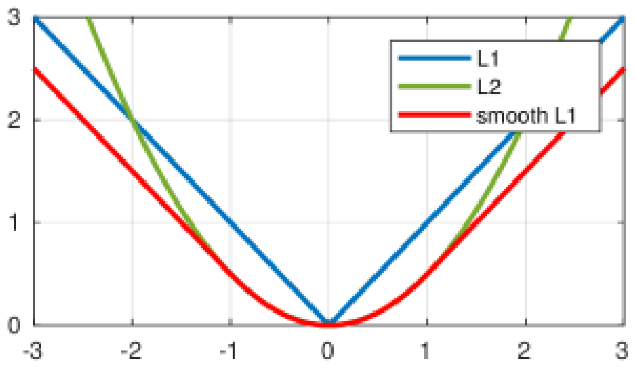


Figure 16: Smooth L1 Loss

In a more general way, the smooth L1 loss can be re-written:

$$sL1 = \begin{cases} |x - y|, & if\ \ |x - y| > a \\ \frac{1}{|a|}(x - y)^2, & else\,if\ \ |x - y| \leq a \end{cases}$$

In other words, the goal of smooth L1 is to minimize the absolute difference between the target and the estimated value.

Focal Loss [3]: Focal loss is a modification of the cross-entropy function, that reduces the contribution from easy examples and increases the importance of correcting misclassified examples.

From the Cross-Entropy function:

$$\boldsymbol{CE(p,y)} = \begin{cases} -\log(p), & y = 1 \\ -\log(1-p), & otherwise \end{cases} \qquad (1)$$

Modifying the above loss function in more simplistic terms:

$$\boldsymbol{p_t} = \begin{cases} p, & y = 1 \\ 1-p, & otherwise \end{cases} \qquad (2)$$

By applying (2), in equation (1):

$$\boldsymbol{CE(p,y) = CE(p_t) = -\log(p_t)} \qquad (3)$$

At this point, Cross-Entropy handles only the weight of positive and negative examples. Positive examples are the target class and negative examples are non-target class or background information. However, in object detection there are samples that were correctly classified as positive or negative example, and there are samples misclassified as negative or positive examples. Those are easy and hard positives/negatives respectively and Cross-Entropy does not handle them at all. Apart from that, usually dataset suffer from class imbalance, making the network biased towards the dominant class.

To solve those problems, Focal Loss adds a modulating factor to the cross-entropy loss, with a tunable hyper-parameter.

$$\boldsymbol{FL(p_t) = (1-p_t)^{\gamma} log\,(p_t)}$$

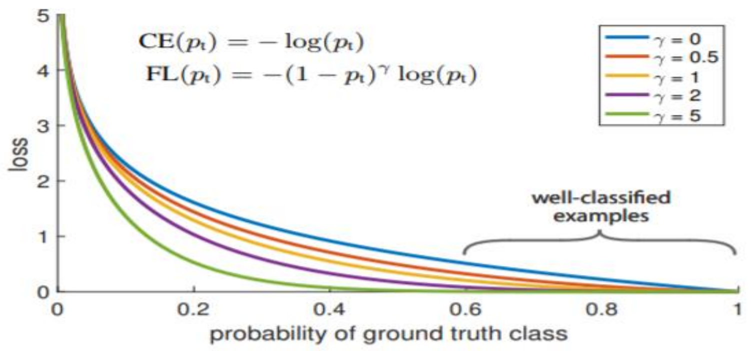


Figure 17: Focal Loss

If the gamma parameter gets the value zero, then the focal loss becomes cross-entropy.

There are other loss functions, such as categorical cross-entropy, hinge loss, Huber loss, Mean Square Error and many more.

With the combination of the layers described previously, various ConvNets architectures can be built and deployed, depending on the task. Traditionally, CNNs are a good option for classification problems, but modern object detection pipelines utilize CNNs as feature extractors as well. This chapter, describes some of the most well established ConvNets and two Fully Convolutional Networks that have make a significant contribution in Computer Vision.

2.4) Visual Geometry Group (VGGNet)

Designed by the Department of Science and Engineering of Oxford University, VGGNets [4] are a series of convolutional neural network models. The original purpose of VGG's research on the depth of ConvNets, was to understand how the depth of NN affects the accuracy of image classification and recognition. Beginning with VGG, two more upgraded models were designed VGG16 and VGG19, with the number representing the depth of the model.

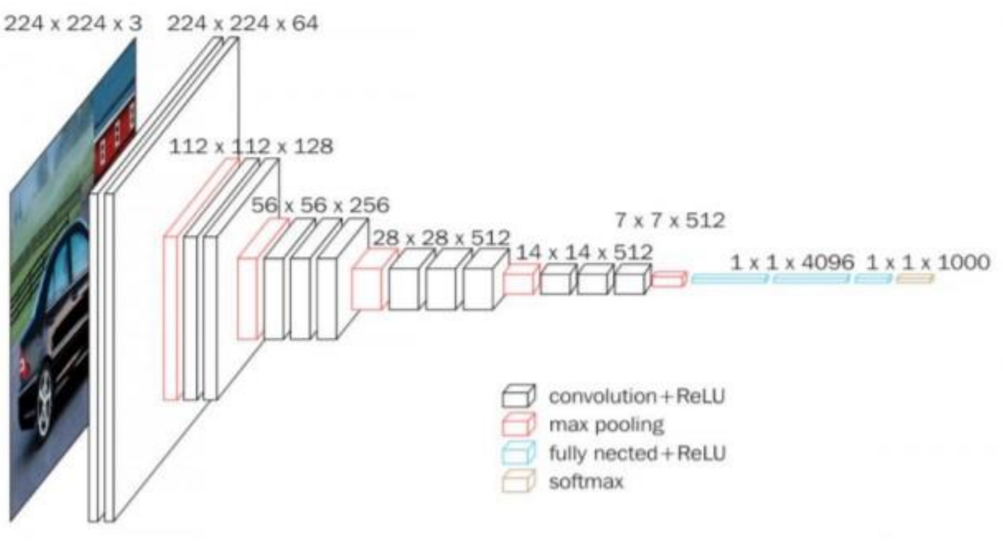


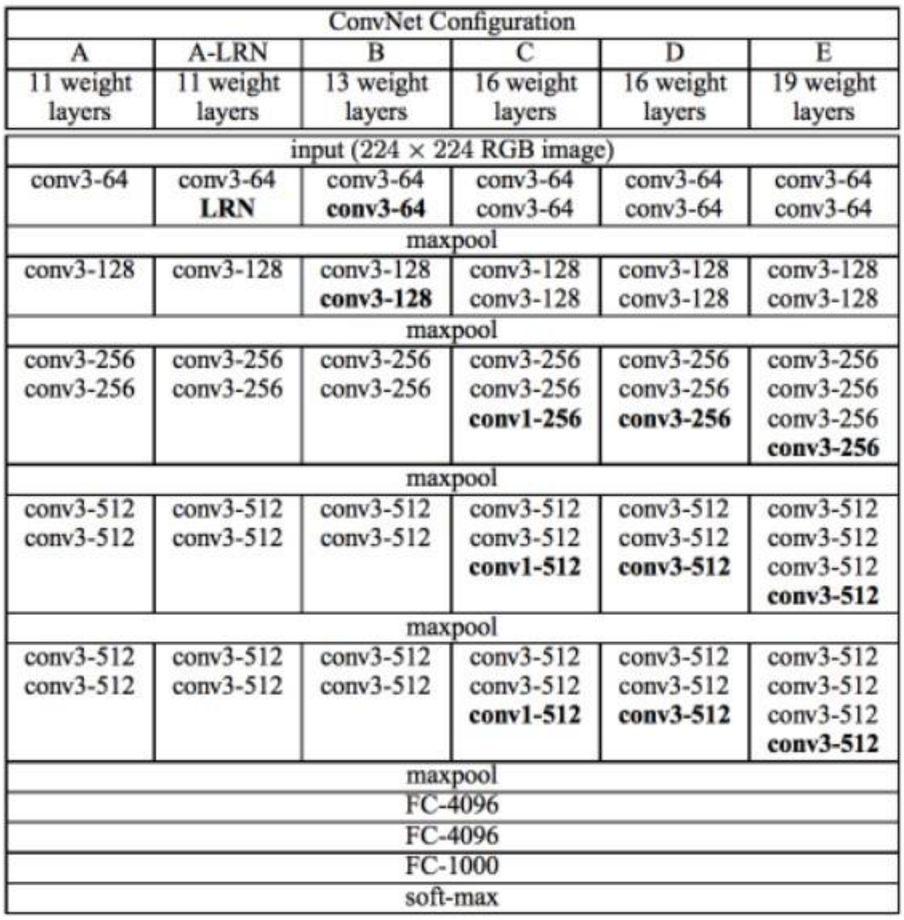
Figure 18: VGG16 architecture

Oxford University proposed the idea of seeing the design of a neural network architecture more abstract and first introduced the idea of blocks and repeating patterns. Visual Geometry Group Networks can be split into six blocks. As can be seen in Image [16], the input image is passed through five blocks. Each block is a sequence of Convolution, Rectified Linear Unit and max pooling layers. The final block, flats the output and uses SoftMax for classification.

One important feature of VGG Net is that it uses convolutional blocks based on 3x3 modules. For example, in the first block each of the output characteristics depends on a 3x3 region of the original image, in the second conv block it depends on 5x5 of the original image and so on. Therefore, each block has a dependency from the original image, that follows this rule:

$$\boldsymbol{current_{module} = (3 + 2n)x(3 + 2n)}$$

Where n = 0, 1, ,.., number of blocks.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 19: VGGNets

VGG shows a simple structure where lower blocks can extract global, semantically rich features and the higher blocks can process the higher resolution pixels of the image.

2.5) Residual Networks (ResNet)

Designing deep neural networks can be a very challenging task to complete. As more layers were stacked in a CNN, more problems were emerging (exploding or vanishing gradient problem). It is important that the addition of layers makes the network strictly more accurate, more expressive rather than just different.

Considering Ƒ is the class of functions that a specific network architecture can reach, then $\forall \boldsymbol{f} \in$ Ƒ exists some set of parameters (biases, weights) that can be obtained through training on a suitable dataset. If ***f**** is the target function and it belongs in Ƒ, then the network is in a good “shape”. Unfortunately, this is rather unlucky, so instead the network tries to find some f_F^* which best fits within Ƒ. Given a dataset with features **X** and labels **y**, the network tries to solve the following optimization problem:

$$f_F^* \overset{\text{def}}{=} \underset{f}{\operatorname{argmax}}\, L(X, y, f) \text{ subject to } f \in F$$

It is only reasonable to assume that designing a different and more powerful architecture **Ƒ’** it should arrive at a better outcome. However, if **Ƒ** is not a subsample of **Ƒ’** then it is not guaranteed that the architecture will perform better. In other words, non-nested function classes do not always move closer to the target function, as illustrated in Image [20]. However, if each architecture is as good as the previous one with the addition of extra complexity (nested functions), then each new model will move towards the target function.

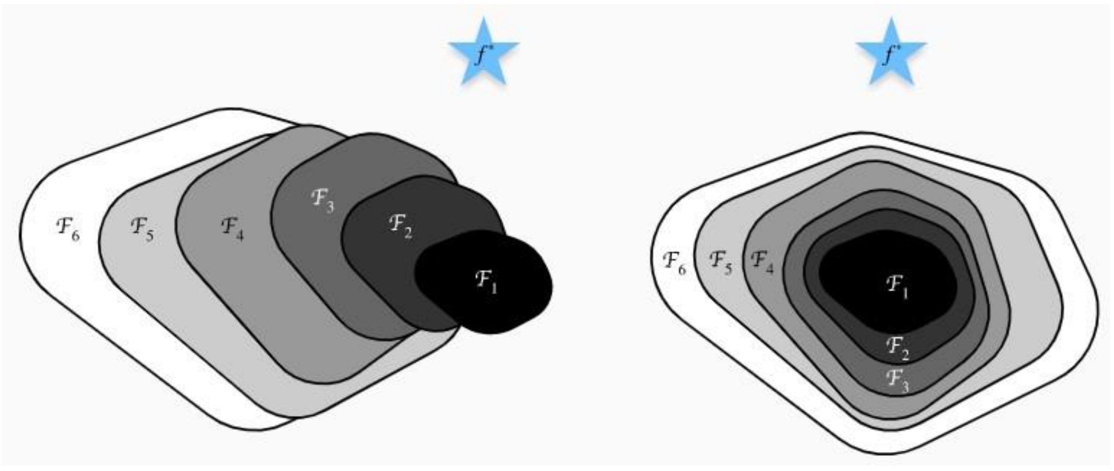


Figure 20: Non-nested and Nested function classes

For deep neural networks, the newly-added layer can be trained into an identity function ***f(x) = x,*** the new model will be as effective as the original model. As the new model may get a finer solution to fit the training dataset, the extra layer might make it easier to minimize training errors.

Assuming a classical convolutional neural network exists then it maps the input **x** to the output **y = f(x)**. A residual network will use a replica of the input **x** to the output of the network and the learning algorithm will only learn the differences between the input and the output. Therefore, the output of the network is ***f(x)+x***. The advantage of this method (residual block), is that it creates layers that they are at least efficient as the previous ones. Furthermore, the architecture of the model is relatively simple, since the same topology is repeated through the entire network.

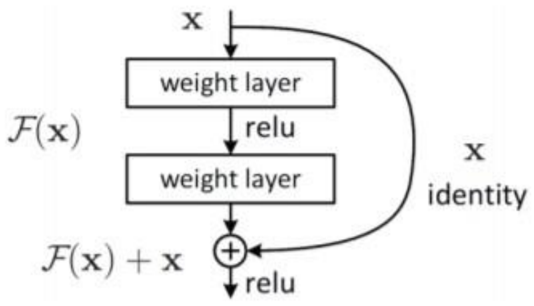
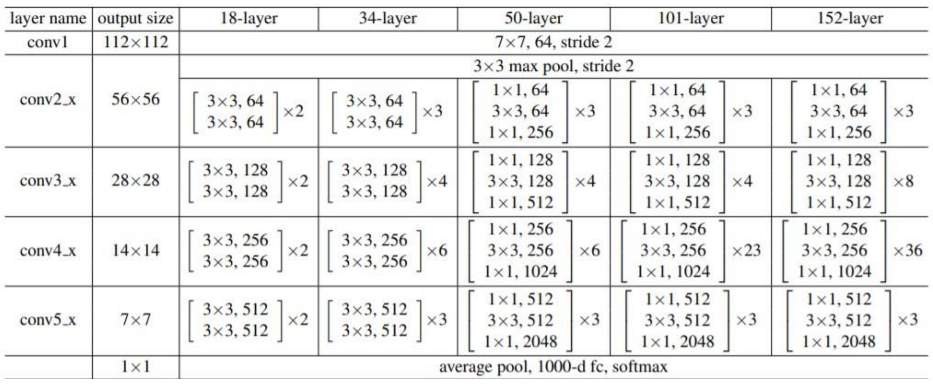

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 21: Residual Block

ResNet or Residual Networks were invented by Microsoft to solve the problems that were stated in the beginning. There are many architectures such as ResNet 18, ResNet 34, ResNet50, Resnet101 and ResNet152, with the number representing the number of layers in each architecture.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				

Figure 22: Residual Networks architecture

ResNet34[5] for example has five convolutional blocks and one block for classification. In each convolutional block the spatial dimensions are reduced by a factor of 0.5, while the number of filters doubles in each block. In the last block, average pooling is applied and a fully connected layer, to flatten the network in the appropriate number of classes. To classify the objects, a softmax layer is applied at the end.

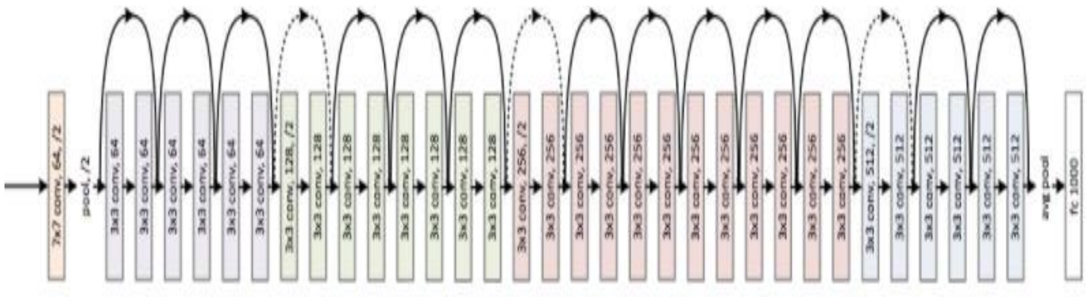


Figure 23: ResNet34 architecture

2.6) U-Net

Convolutional Networks are powerful visual models that yield hierarches of features. All models described in previous sections utilized fully connected layers for classification. Unlike classic CNNs, a Fully Convolutional Network (FCN) does not have fully connected layers. The neural network can only perform convolution and pooling operations (for FCN pooling is either up-sampling or down-sampling). The CNNs layers typically reduce or down-sample the spatial dimensions of the input (height and width), or keep it unchanged. For a Fully Convolutional Network architecture, since they perform pixel-wise classification most of the times, it will be convenient if the spatial dimensions of the input and output are the same. To achieve this, especially after the spatial dimensions are reduced after each convolution step, another type of layer called transposed convolution can be used to increase (up-sample) the spatial dimensions of the intermediate feature maps.

Given a $n_h \times n_w$input tensor and $k_h \times k_w$kernel. Sliding the kernel window with stride of one for $n_w$times in each row and $n_h$times in each column yields a total of $n_h n_w$intermediate results. Each intermediate result is a $(n_h + k_h - 1) \times (n_w + k_w - 1)$tensor that are initialized as zeros. To compute each intermediate tensor, each element in the input tensor is multiplied by the kernel so that resulting $k_h \times k_w$tensor replaces a portion in each intermediate tensor. In the end, all intermediate results are summed together to produce the output of the transposed convolution. An example can be seen in the following Image.

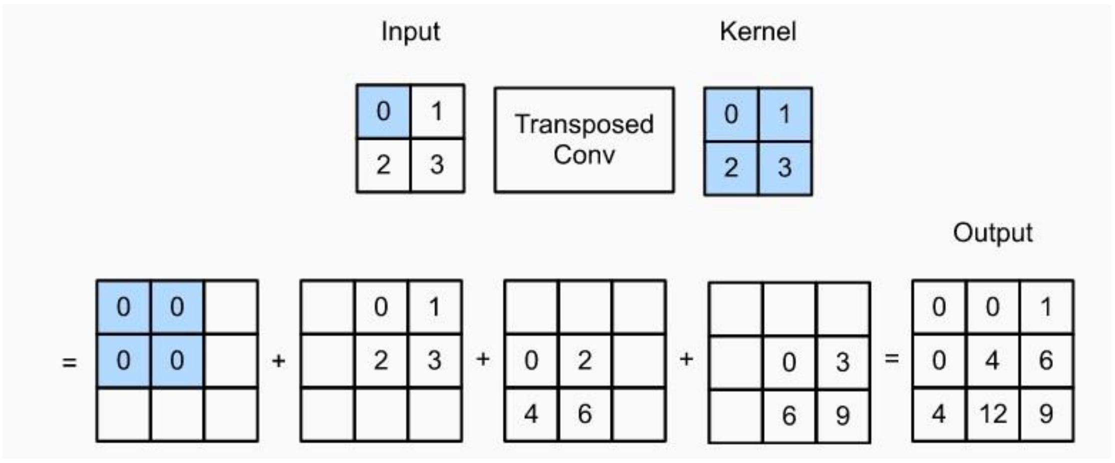


Figure 24: Transpose Convolution example

An example of fully convolutional network is the U-net [6]. It inherited its name because of its distinct U shape, which can be seen in Image [24]. The network consists of a contracting path (top-down pathway) and an expansive path (bottom-up pathway).

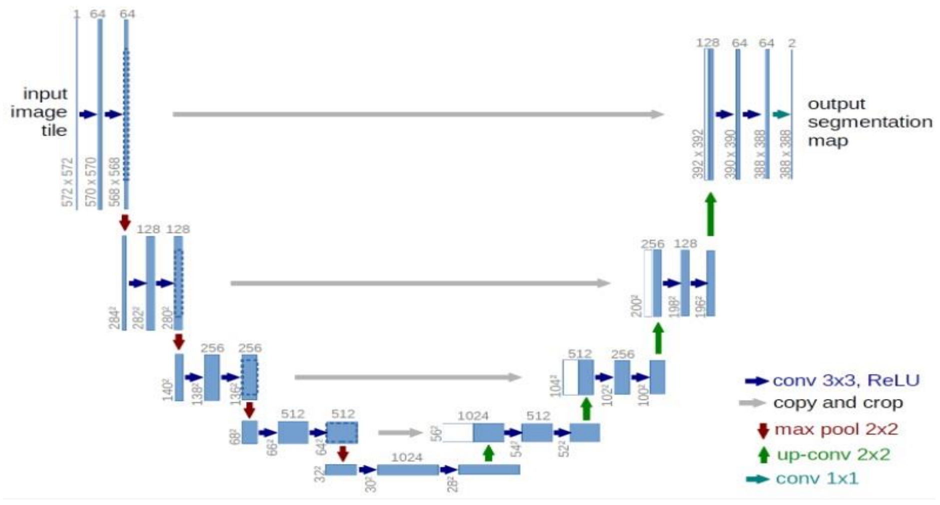


Figure 25: U-net architecture

The top-down path follows the architecture of a simple ConvNet, consisting of convolutions and max pooling operations. At first, two 3x3 zero-padded convolutions are applied, each followed by a ReLU and a 2x2 max polling operation with stride of 2 channels. At each down-sampling step the spatial dimensions are reduced by a factor of 0.5 and the number of feature channels are doubled.

Every step in the bottom-up pathway consists of an up-sampling operation of the feature map, followed by a 2x2 "up-convolution" that halves the number of feature channels, followed by a concatenation with the corresponding cropped feature map from the dilated path and two 3x3 convolutions, followed by a ReLU. The cropping operation is necessary to match the size of the new feature map that is produced after each convolution step. As a final layer a 1x1 convolution is used to map each component feature vector to the desire number of classes.

U-net was initially developed for biomedical-segmentation, but it is widely used in semantic segmentation problems (pixel-wise classification).

2.7) Feature Pyramid Network (FPN)

Detecting objects in different scales is a rather challenging task, especially when it comes to small objects. Feature pyramids on top of image pyramids form the root of a standard solution to this challenge. Those pyramids are scale-invariant meaning that an object's scale changes the offset by shifting its level in the pyramid. This property, allows a model to identify and detect objects across a large range of scales by scanning the model over both positions and pyramid levels.

Featurized image pyramids [7] were heavily used in the era of hand-engineered features. In the modern days, tasks like recognition and object detection use features that have been computed by an algorithm or by another deep learning model and not by hand.

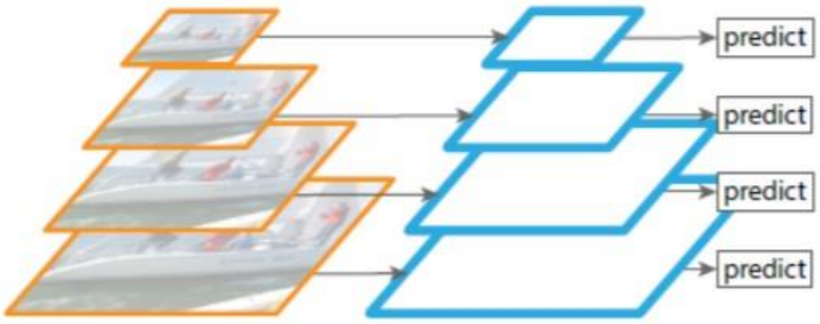


Figure 26: Featurized Image Pyramids

Furthermore, featurizing each level of an image pyramid has various limitations. Inference time increases considerably, making this approach impractical for real applications. Also, training a deep neural network end-to-end on an image pyramid is memory consuming.

Another way to compute a multi-scale feature representation is by using a ConvNet. A CNN computes a feature "hierarchy" layer by layer and with sub-sampling layers the hierarchy gets a multiscale pyramidal shape.

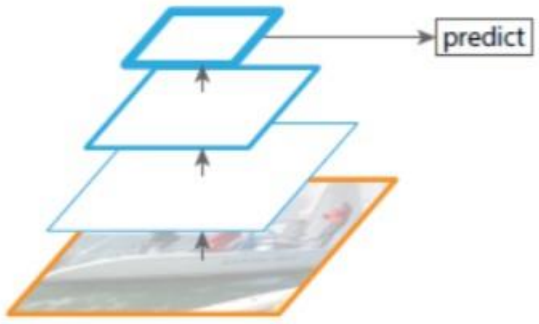


Figure 27: Single Feature Map

This hierarchy produces feature maps of different spatial resolutions, but introduces large semantic gaps caused by different depths. As the network goes deeper, the resolution of the image reduces, but semantically strong features are being extracted. The single feature map model misses the opportunity to reuse higher-resolution maps, consequently misses the detection of object of different sizes.

Feature Pyramid Network or FPN combines low-resolution but semantically strong features with high resolution but semantically weak features. To achieve this, it utilizes a bottom-up and a top-down pathway with lateral or skip connections.

The bottom-up pathway uses a ConvNet like ResNet or VGG. From one convolution module to the next, the spatial dimensions are reduced by ½. The output of each convolution module is later used in the top-down pathway.

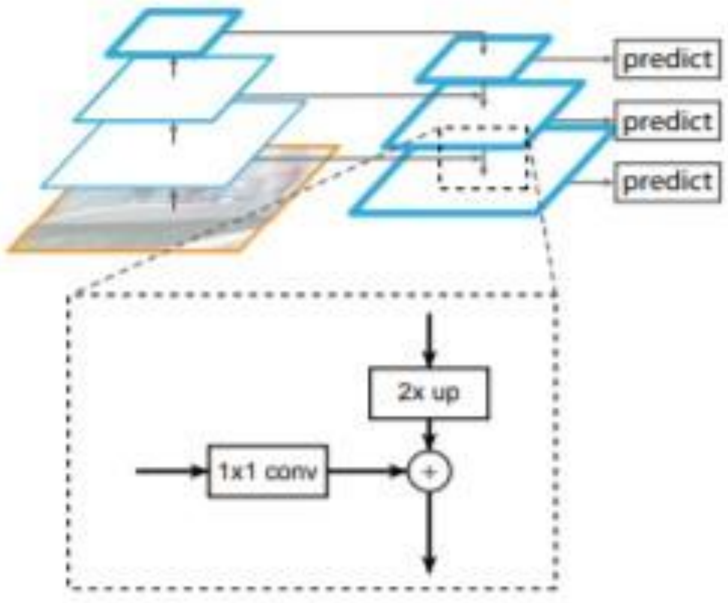


Figure 28: FPN architecture

Each output of the bottom-up pathway is used as input in the top-down pathway. To fit the dimensions, the FPN uses lateral connections, which is simple a 1x1 convolution filter to reduce the output channel depth. Going down the path, each previous layer is up-sampled by 2, using nearest neighbors up-sampling (Figure 29).

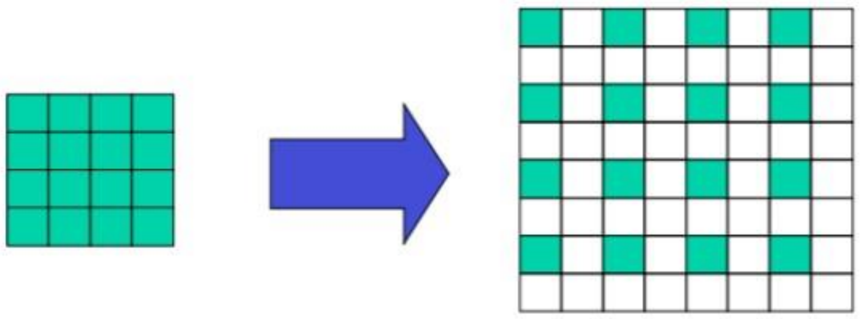


Figure 29: Up-sampling

The result is an image with double the size of the spatial dimensions. Again, a 1x1 convolution is applied to the corresponding feature maps in the bottom-up pathway. Then the results are added element-wise. Finally, a 3x3 convolution filter is applied to all merged layers. This final filter is applied to reduce the aliasing effect of the up/down-sampling that takes place.

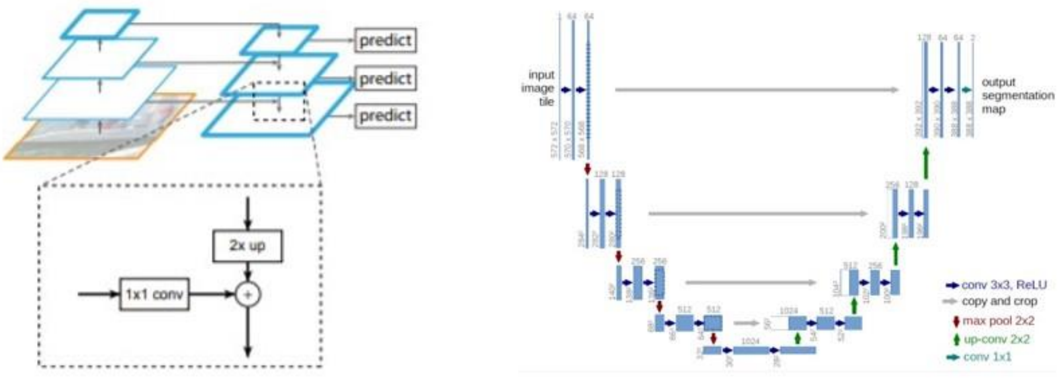


Figure 30: U-net vs FPN

Both U-net and Feature Pyramid network are Fully Convolutional Networks. While they seem to have big similarities, the main difference is that FPN utilizes lateral connections, while U-net only copies and concatenates the cropped areas.

Object detection systems, based on the architecture, can be classified in two detector categories:

- Single-stage detectors.
- Two-stage detectors.

The difference between the two pipelines is that two-stage detectors use in the first stage an algorithm or even an entire Neural Network to create possible locations of objects in an image and the second stage is responsible for classification and box location correction, while single-stage detectors localize an object and classify it in a single pass.

Two-stage detectors were designed for accuracy, while single-stage detectors for speed. As deep neural networks advanced over the years, so did object detector systems, to the point were two-stage and single-stage detectors perform at the same accuracy and speed level.

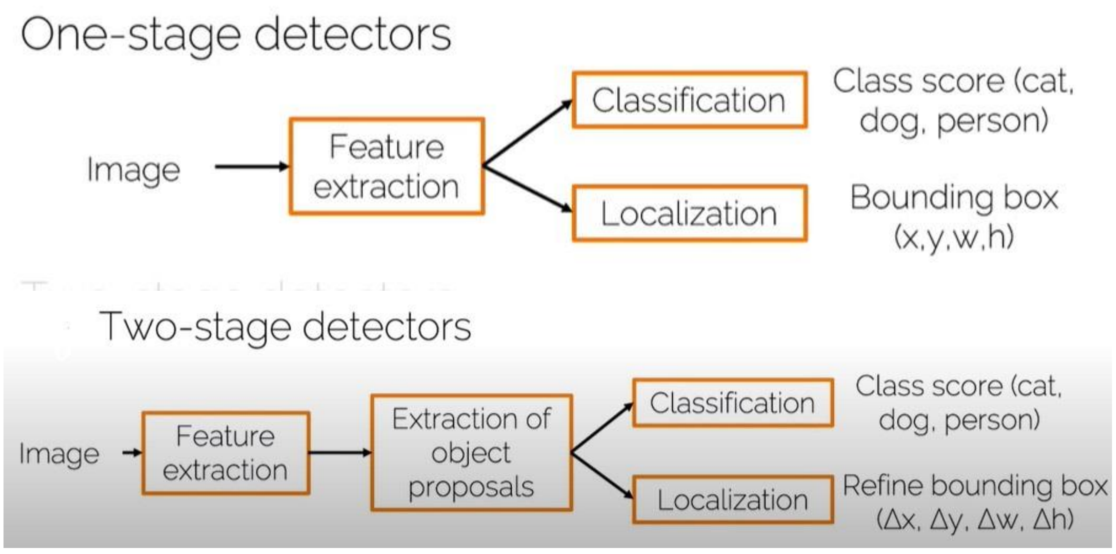


Figure 31: Object Detection Pipelines

2.8) Two-stage architectures

The problem of object detection can be separated into two sub-problems. Box localization and classification of the proposed location. Due to unknown number of instances of an object in an image, a simple ConvNet architecture is not enough. As an alternative, the image can be divided in a fixed number of regions, and the ConvNet can classify if the image contains a certain class of objects. The problem with this approach is that different objects have different spatial locations within an image and different aspect ratios. Hence, a huge number of regions must be selected, making it a huge computational problem.

To bypass this problem various methods were proposed aiming the reduction of the proposed locations. The Networks that utilize that kind of techniques were known as Region-based Convolutional Neural Networks (RCNNs) or two-stage detectors.

2.8.a) Regional CNN (R-CNN)

The first Region Proposal Network [8] (R-CNN) took an image as input and produce a set of bounding boxes as output, where each bounding box contains an object and also the category of the object. In order to keep the number of proposed locations small, R-CNN utilizes in the first stage of the pipeline a mechanism called Selective Search to extract regions of interest (ROI). Each ROI is a rectangle that may represent the boundary of an object in an image.

SELECTIVE SEARCH USING HIERARCHICAL GROUPING

```
    Input: Image.
    Output: Set of object location hypotheses L.
1   Obtain initial regions $R = \{r_1, \ldots, r_n\}$ using F&H method.
2   Initialize similarity set $S = \emptyset$.
3   foreach neighboring region pair $(r_i, r_j)$ do
4       Calculate similarity $s(r_i, r_j)$.
5       $S = S \cup s(r_i, r_j)$
6   while $S \neq \emptyset$ do
7       Get highest similarity $s(r_i, r_j) = \max(S)$
8       Merge corresponding regions $r_i = r_i \cup r_j$
9       Remove similarities regarding $r_i$: $S = S\ s(r_i, r_*)$
10      Remove similarities regarding $r_j$: $S = S\ s(r_*, r_j)$
11      Calculate similarity set $S_t$ between $r_t$ and its neighbors.
12      $S = S \cup S_t$
13      $R = R \cup r_t$
14  Extract object location boxes L from regions in R.
```

Usually, selective search [9] calculates approximately 2.000 possible object locations. After that, each ROI is warped into a square and fed through a convolutional neural network that produces an output feature vector. Each ROI's feature output is fed into a Support Vector Machine to classify the presence of the object within that candidate region proposal. In addition to predicting the presence of an object within the region proposals, the algorithm also predicts four values, which are offset values to increase the precision of the bounding box. In other words, the algorithm uses these 4 values to adjust the coordinates of the region proposal, as close as possible to the ground truth.

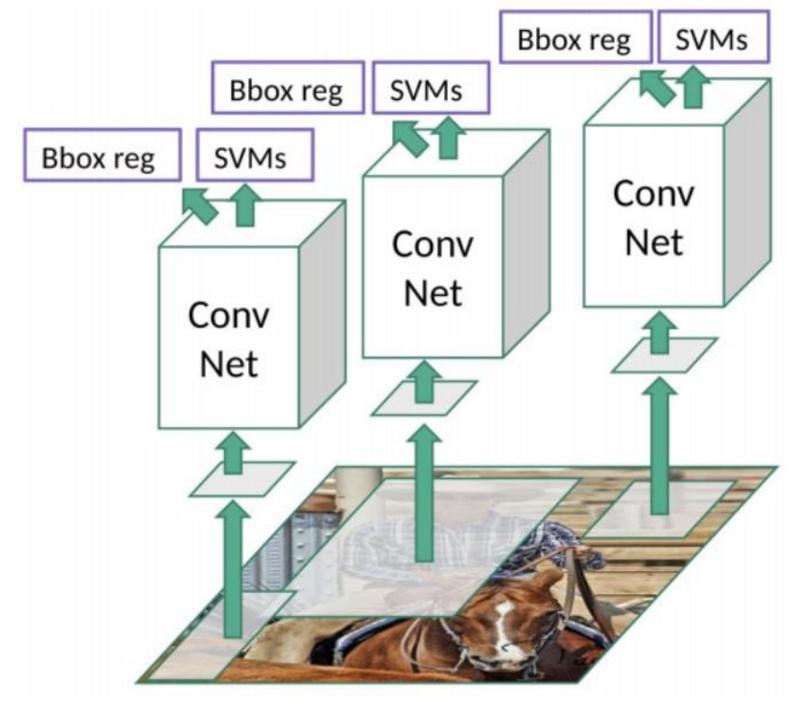


Figure 32: RCNN architecture

In Figure 32, the architecture of RCNN can be seen. In this particular image example, there are three region proposals that are warped into a fix size rectangle and they are fed into a ConvNet. Each feature vector is then used to adjust the bounding box and classify the object within the region. Although the example looks simple, in a real dataset they would be around 2.000 proposals for each image, making the whole process a difficult computational problem. Furthermore, the selective search algorithm is a fixed algorithm and no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

2.8.b) Fast R-CNN

The main performance bottleneck of an R-CNN lies in the independent CNN forward propagation for each region proposal. Fast R-CNN [10] proposed a different mechanism to the region proposal extraction. The algorithm, but instead of feeding the region proposal to the CNN, the input image is fed to the ConvNet to generate a convolutional feature map. Then, from the convolutional feature map, the region proposals are identified through selective search and warped into squares of fixed size. To achieve that, Fast R-CNN introduces Region of Interest pooling layer, which is a similar operation to max-pooling.approach is similar to the R-CNN.

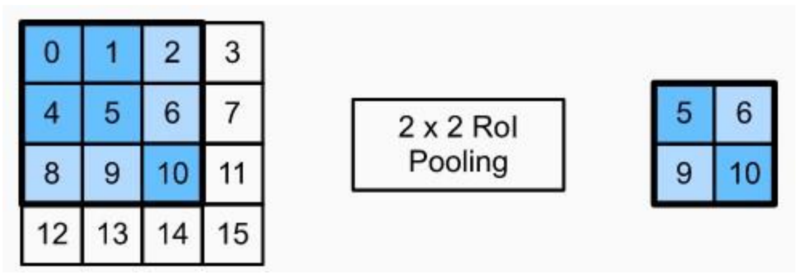


Figure 33: A 2x2 ROI pooling layer example

The feature output from the ConvNet and the region proposals from the selective search algorithm are input into the ROI pooling layer, outputting concatenated features that are further extracted for all the region proposals. The output of the ROI pooling layer is called ROI feature vector. Each of these feature vectors are of fixed size, and they can be fed into a fully connected layer to flatten down the dimensions, and later used as input into a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.

Unlike R-CNN, Fast R-CNN does not have to feed 2.000 region proposals to the ConvNet every time, instead the convolution operation is done only once per image and a feature map is generated from it.

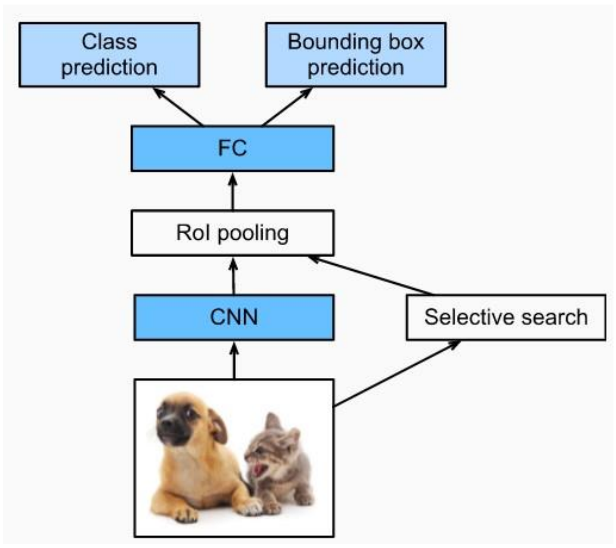


Figure 34: Fast R-CNN architecture

2.8.c) Faster R-CNN

Both R-CNN and Fast R-CNN use the selective search algorithm to extract the region proposals. Selective search is a rather slow and time-consuming algorithmic process that affects critically the performance of the network. Instead of using Selective

Search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals, called Region Proposal Network (RPN).

Region Proposal Networks follow these three steps:

- Generate Anchor boxes.
- Classify each anchor box whether it is foreground of background.
- Learn the shape offsets for anchor boxes to them for objects.

Anchor boxes[11] "*are a set of predefined bounding boxes of a certain height and width*". They are defined to capture the scale and aspect ratio of specific object classes that need to be detected and they are typically chosen based on object sizes in the training dataset.

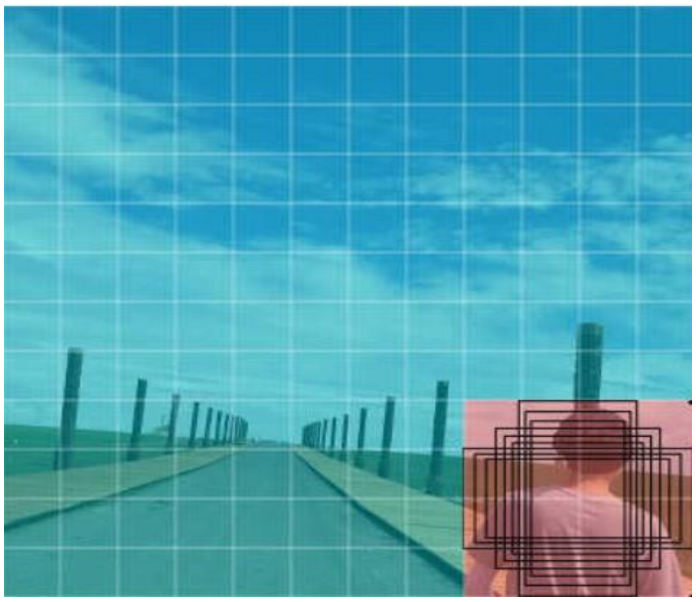


Figure 35: Anchor boxes example

After producing the anchor boxes, the RPN predicts the binary class (background or object) and bounding box for each anchor box. Sometimes, due to the large number of boxes produced, more than one bounding box predicts the same object. To remove overlapped results, non-maximum suppression [11] is applied. The remaining predicted bounding boxes for objects are the region proposals required by the ROI pooling layer.

Non-Max suppression

```
1  Procedure NMS(B,c)
2  $B_{nms} \leftarrow \emptyset$
3  for $b_i \in B$ do
4      Discard ← False
5      for $b_j \in B$ do
6          If same $(b_i,b_j) > \lambda_{nms}$ then
7              If score $(c,b_j) >$ score$(c,b_i)$ then
8                  Discard ← True
9      If not discard then
10         $B_{nms} \leftarrow B_{nms} \cup b_i$
11 Return $B_{nms}$
```

Faster R-CNN [12] is an upgrade of the Fast R-CNN. They follow the same architecture, but Faster R-CNN utilizes RPN instead of Selective Search.

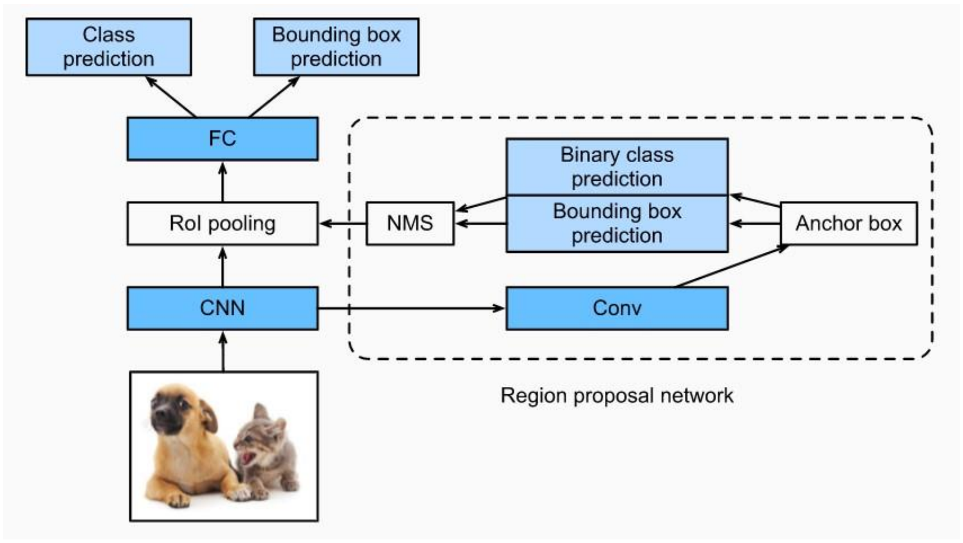


Figure 36: Faster R-CNN architecture

As part of the whole model, the region proposal network is jointly trained with the rest of the model. In other words, the loss or objective function of the Faster R-CNN includes not only the class and bounding box prediction in object detection, but also the binary class and bounding box prediction of anchor boxes in the RPN. The result is an end-to-end training method, where the RPN learns how to generate high-quality region proposals, so as to stay accurate in object detection with a reduced number of region proposals that are learned from data.

2.9) Single-stage architectures

Single stage model architectures, unlike Regional Neural Networks, directly predict object bounding boxes and classification score for an image. There is no intermediate task like detecting region proposals through a neural network or with an algorithm like Selective Search. The result is a much simpler and faster architecture.

Usually, Single Stage architectures utilize a ConvNet such as VGG or ResNet as a backbone network. The backbone network is responsible for feature extraction. Then, usually, two more Fully Convolutional Networks are deployed for box regression and object classification.

2.9.a) You Only Look Once (YOLO)

YOLO [13] is a family of deep learning models, designed for fast object Detection. At the moment, there are three YOLO versions. YOLOv1 (Figure 37) proposed the general architecture, where the second version introduced anchor boxes to improve the bounding box proposal. Version number three proposed changes in the training process of the model.

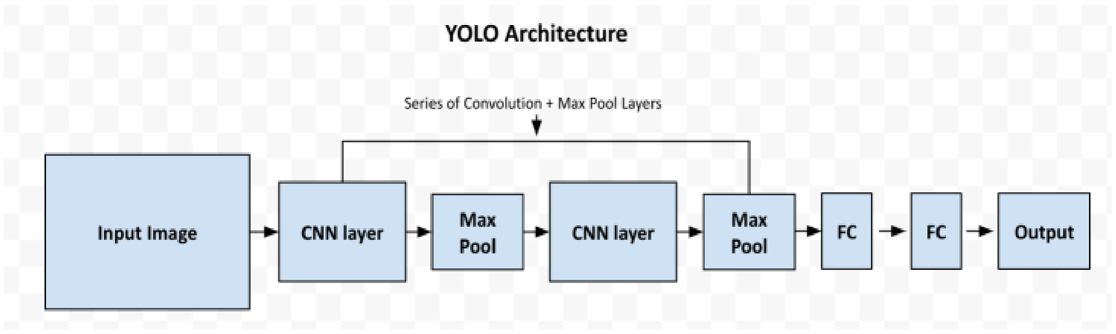


Figure 37: YOLO architecture

As seen in Figure 37, the basic architecture of YOLO relied on a series of Convolutions and Max Pooling layers. Finally, for YOLOv1, two fully connected layers are responsible for predicting the bounding boxes. In v2, the fully connected layers were removed and instead, the anchor boxes are responsible for box regression.

In general, YOLO is a very successful object detector. The simplicity of the architecture, allows YOLO to be trained quite fast and version three is as accurate as the best two-stage object detector.

2.9.b) Retina Network

Focal loss was introduced with RetinaNet. The goal was that a simple Fully Convolutional architecture with the appropriate loss function could fix the extreme foreground-background class imbalance problem that came with single-stage architectures.

RetinaNet [12] consists of a backbone network (ConvNet +FPN) and two task specific fully convolutional networks, one for box regression and one for object classification.

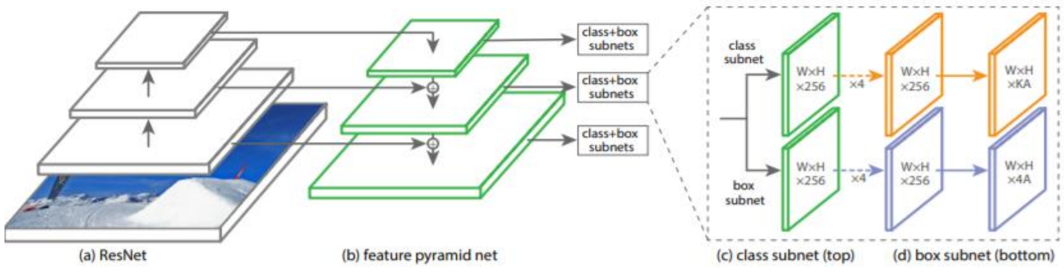


Figure 38: RetinaNet architecture

The backbone network is responsible for computing a convolutional feature map over an entire image and is off-the-self convolutional network.

As previously stated, networks that utilize anchor boxes suffer from extreme foreground-background class imbalance due to dense sampling of anchor boxes. In RetinaNet, each Feature Pyramid level can contain thousands of anchor boxes. Only a few will be assigned to a ground-truth object, while the rest will be background class. These easy examples or detections with high probability, although resulting in small loss values can collectively overwhelm the model. For that reason, Focal Loss was chosen as loss function in the model training phase. FL reduces the loss contribution from easy examples and increases the importance of correcting misclassified examples (see also section 2).

There are many more single-stage detectors such as Single Shot Detector or SSD, SqueezeDet, DetectNet. Also, there are variations of the well-known detectors such as Tiny or Fast YOLO that have been developed for more speed or less computing power.

The RetinaNet can be separated in three major components:

1) The backbone Network, which consists of a bottom-up and a top-down pathway.
2) Two subnetworks for classification and regression.
3) The loss function to train the model.

3.1) Backbone Network

As mentioned in 2.9.b the backbone net is a fully convolutional network with lateral connection. It consists of an encoder and a decoder. In the original paper the encoder was a residual network such as ResNet50 and the decoder was a Feature Pyramid Network. The original image gets fed into the encoder, which processes the image through convolutional kernels and generates deep features. The bottom-up pathway as it is also called can be any convolutional network, as long as it does not contain any fully connected layers.

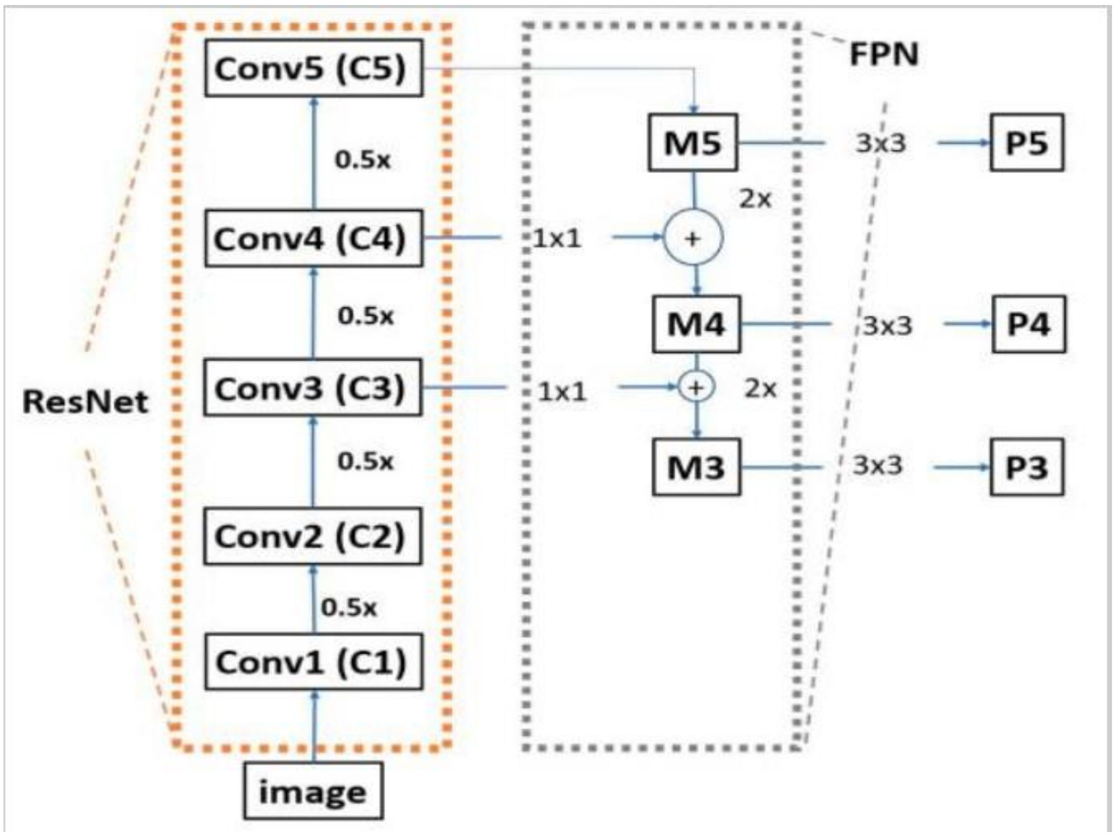


Figure 39: Backbone Network

As seen in Figure 39, moving from the bottom all the way to the top, the spatial dimensions are reduced at each convolutional block by a factor of 0.5.

For each feature map produced by the ResNet, the FPN does the exactly opposite operation. It up-samples the spatial resolution of each feature map input by a factor of two. The result is then merged with the corresponding bottom-up map, which undergoes a 1x1 convolution to reduce channel dimension for element-wise addition.

3.2) Classification/Regression Networks

The classification and regression networks are two Fully Convolutional Networks with simple architecture. The classification network predicts the probability of an object to present at each spatial position for each of the A anchors and K object classes, while the regression net predicts the relative offset between anchor and the ground truth box.

The classification network design is simple. As input takes a feature map with C (256) channels from a pyramid level. Then it applies four 3x3 convolutional layers, each with C filters, followed by ReLU activation functions, followed by a 3x3 convolutional layers with KA filters. As last layer, sigmoid function is attached to output the KA predictions per spatial location.

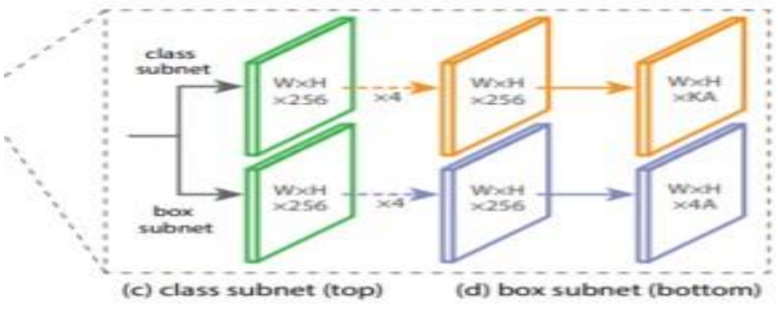


Figure 40: Classification/Regression Networks

The bounding box regression network has an identical design, with the difference of having 4A linear outputs per spatial location.

These two small FCNs are attached to each FPN level. Meaning if the FPN has four levels, 8 subnetworks are needed to perform object detection. Furthermore, although the two networks share identical architecture, they use separate parameters (weights, biases).

3.3) Loss Functions

For classification, RetinaNet uses a modification of the Cross-Entropy loss function, called Focal Loss (see 2.3.e) or FL. Focal Loss reduces the contribution from easy examples and increases the importance of correcting misclassified examples.

For box regression uses smooth L1 (see 2.3.e) or Least absolute deviation or Least Absolute Error, as a loss function. The goal is to minimize the absolute difference between the target value and the estimated value.

3.4) Why Retina?

Choosing a deep learning model to complete a certain task is quite challenging. The evaluation process depends on many factors such as the challenge, the data, the application of the model and many more. In order to have a general idea how a network performs organizations and companies release from time-to-time huge datasets with many classes and evaluate their models with them. One of those datasets is Microsoft's Common Objects in Context or the COCO dataset.

The COCO image dataset was created with the goal of advancing image recognition. The dataset contains demanding, image or video datasets for computer vision or object detection, mostly state-of-the-art neural networks. It is often used to compare the performance of object detection algorithms. It contains 80 classes of objects and most of the well-known object detectors come with a pre-trained model on the COCO dataset.

Having all the above in mind, the following chart, showcases the Mean Average Precision of different Object Detectors using the COCO dataset.

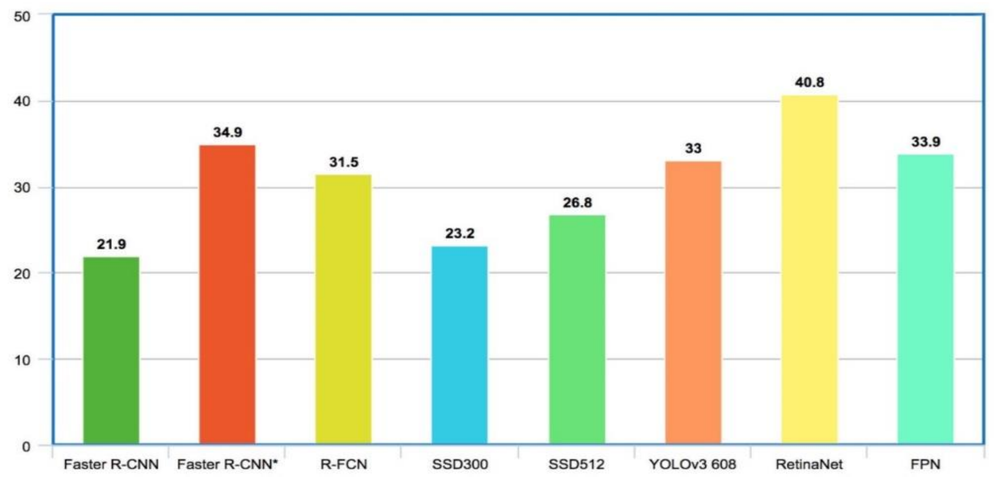


Figure 41: COCO MAP results

At the particular time, comparing to the other famous object detectors, RetinaNet really stands out. Taking in to consideration the fact that it is a Single Stage Detector, therefore it is fast, it was only logical to select this deep learning architecture as our base model for comparison.

4.1) Stanford Drone Dataset (SDD) description

The SDD[14] is a massive data set of aerial images collected by drones over the Stanford campus. The particular dataset is ideal for computer vision task such as object detection or target tracking. It contains more than 60 aerial videos or 69GB of data. The dataset consists of eight unique scenes. For each video, a model can detect 6 different agents – “Pedestrians”, “Bikers”, “Skaters”, “Carts”, “Cars” and “Bus”. Unfortunately, the dataset is biased, since the classes of Pedestrians and Bikers are covering more than 80% of the annotations.

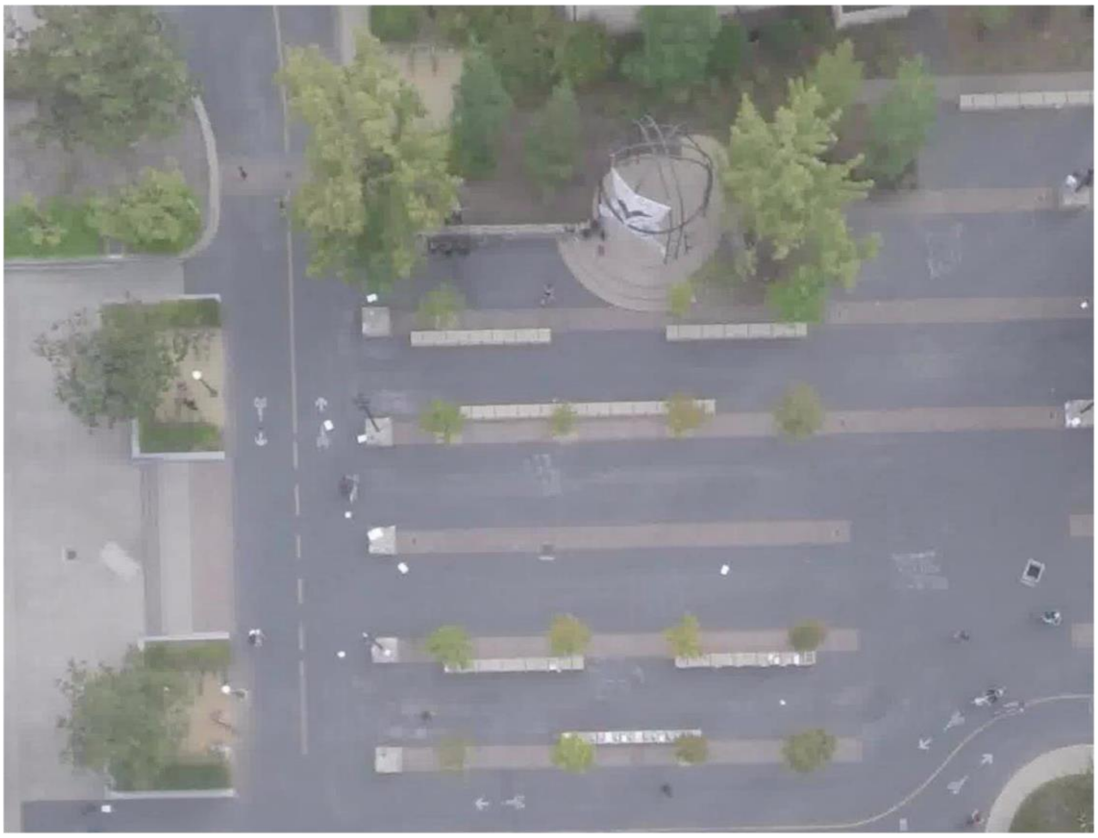


Figure 42: SDD image example

4.2) Changes in the Dataset

For the experiments that were conducted, some of the agents were merged into new classes. To be more precise, the "Pedestrian" and "Skater" classes merged into a new class called "Person". The "Biker" class remained as it is. The "Car" and "Cart" agents were combined and the class kept the name "Car". The "Bus" class remained also as it is.

After all the editing, the Training and Testing annotations for the experiments were:

Classes (Training Annotations)	**Number**
Person	**22.673**
Biker	**11.479**
Car	**1.512**
Bus	**101**
Total	**35.765**

Classes (Testing Annotations)	**Number**
Person	**5.558**
Biker	**1.204**
Car	**23**
Bus	**31**
Total	**6.816**

The dataset is stored in a csv file in the Pascal VOC format:

(image, x1, y1, x2, y2, class_name)

Where x1, x2, y1, y2 are the four coordinates needed to locate an object.

5. Evaluation Metrics

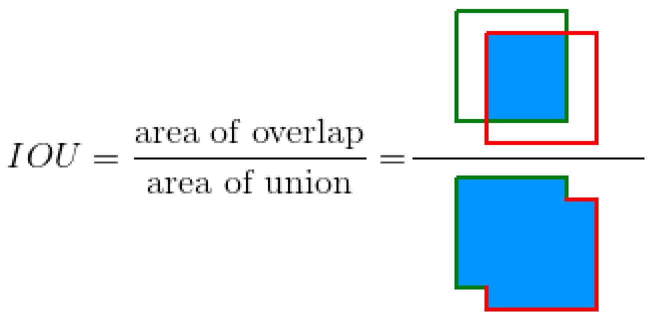
5.1) Intersection Over Union (IoU)

Intersection Over Union[15] is a metric that evaluates the overlap between two bounding boxes. It requires the true coordinates of the object that needs to be detected (the ground truth box B_{gt}) and a bounding box B_p that was predicted. By applying the IoU we can evaluate whether a detection is valid (True Positive) or not (False Positive).

IoU is given by the overlapping area between the prognosticated bounding box B_p and the B_{gt}, divided by the area of union between them:

$$\boldsymbol{IoU} = \frac{\boldsymbol{area}\ (\boldsymbol{B_p} \cap \boldsymbol{B_{gt}})}{\boldsymbol{area}(\boldsymbol{B_p} \cup \boldsymbol{B_{gt}})}$$

To represent the above equation in terms of detection, the image below illustrates the IoU between a ground truth bounding box in green and a detected bounding box in red.



$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$

Figure 43: IoU representation

Using IoU the following concepts can be defined:

- **True Positive (TP)**: A correct detection. IoU $\geq$ threshold.
- **False Positive (FP)**: A wrong detection. IoU $<$ threshold.
- **False Negative (FN)**: A ground truth object not detected.
- **True Negative (TN)**: In object detection this metric has no use, since it represents a corrected misdetection and in object detection there are many possible bounding boxes that should not be detected within an image. Therefore, True Negative would be all possible bounding boxes that were correctly not detected.

The threshold is a constant that is metric dependent. It is usually set to values above 0.5 or 50%.

5.2) Metrics

Using the concepts that were defined in section 5.1, the following metrics can also be extracted:

$$Accuracy = \frac{TP}{TP + FP + FN}$$

Classification Accuracy is the simplest metric to use and it is defined as the number of correct predictions, divided by the total number of predictions.

$$Precision = \frac{TP}{TP + FP}$$

Precision is the ratio of true positives divided by the total positives that were predicted.

Many times, accuracy is not enough to determine whether a model behaves ideally. Therefore, precision needs to be considered. A precision score towards 1 will signify that the model did not miss any true positives and is able to classify well between correct and incorrect labeling. A low precision score on the other hand, means that the classifier has a high number of false positives which can be an outcome of imbalanced class or untuned model hyperparameters.

$$* Recall = Sensitivity = \frac{TP}{TP + FN}$$

Sensitivity, is essentially the ratio of true positives to all the positives in ground truth. Recall towards 1 will signify that the model did not miss any true positives and it is able to classify well between correctly and incorrectly labeling. A low recall score, means that the classifier has a high number of false negatives which can be an outcome of imbalanced class or again untuned model hyperparameters.

$$F1score = \frac{2 * precision * recall}{precision + recall}$$

The F1-score metric, is a combination of precision and recall. To be more accurate, F1 is the harmonic mean of the two metrics.

A high F1-score, translates as a high precision and high recall scores. It presents a good balance between precision and recall and gives good results on imbalanced classification datasets.

A low F1 score is difficult to explain. Both of the metrics can be responsible for a low F1-score. Low sensitivity means that the model did not do well on very much of the entire test set. Low precision means that among the cases that were identified as positive, the model did not get many of them right.

Precision measures how the accurate the predictions of the model are, while Recall measures how well the model finds all the positives. The combination of those two metrics for various thresholds allow to produce another metric called **Precision-Recall Curve** or **PR-curve**. For each class, the area below the PR-curve represents the Average Precision of that class. Weight summing each AP over each class, we get the **mean Average Precision** of the model.

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

Where AP_k the Average Precision of class k and n the number of classes [16].

*For this diploma thesis the Sensitivity or Recall score is defined, but in the equation the None class (see confusion matrix in the experiments section) is not taken into account, therefore the results tend to be significantly higher, but that is the result of the proposed Recall metric. If the None class is taken into consideration the Sensitivity score is closer to 0.75 for confidence threshold of 0.6 in the first experiment.

6. Base model Experiments

To create a base case, the original RetinaNet model was trained using a simple split of the dataset. Eighty percent of the data were used in the training process and twenty percent for testing.

For convenience, the training process took place in Google Colab. For the training process, NVIDIA Tesla P100/V100 GPU accelerators were deployed.

Tesla P100 is powered by NVIDIA Pascal architecture [17]. There are 3.584 CUDA cores and the accelerator can perform at 4.7 TeraFLOPS. The memory of the GPU is 16GB CoWoS HBM2 at 732 GB/s, with maximum power consumption at 250W.

Tesla V100 is powered by the Volta architecture [18]. It combines Tensor and Cuda Cores within a unified architecture. To be more specific, it is equipped with 640 Tensor cores and 5.120 CUDA cores. It performs at 7 TeraFlops and the GPU memory is 32GB HBM2. With a combination of improved raw bandwidth of 900GB/s and higher DRAM utilization efficiency, the V100 delivers 1.5X higher memory bandwidth over the P100. The maximum Power consumption is at 250W like the P100.

Both the P100 and V100 support every major deep learning framework such as Tensorflow, PyTorch, Caffe2, etc.

6.1) Simple Split

For the first experiment, the process was relatively simple. First the data were split into training and testing data, 80/20 split. Then the training annotations were fed into the network as input images. The model was trained for 100 epochs using the NVIDIA Tesla P100 GPU accelerator. To make the training process faster, a pre-trained ResNet in the COCO dataset was utilized. Therefore, only the classification and regression networks needed to be trained and readjust their weights to detect the objects of the SDD.

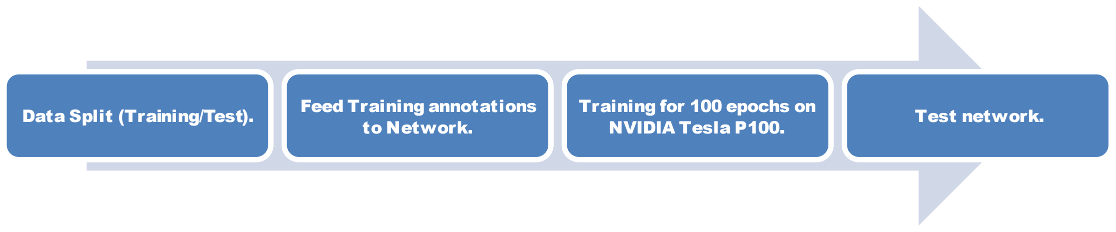

Data Split (Training/Test).
Feed Training annotations to Network.
Training for 100 epochs on NVIDIA Tesla P100.
Test network.

Figure 43: Training/Testing process

In the following matrix the metrics of the experiment are represented. For a prediction to be classified as True, a 0.6 threshold of confidence was chosen. That means that if a detection has an IoU score confidence over the IoU threshold, then to be classified as a TP, the confidence score of the detected object needs to exceed the prediction threshold.

To evaluate how well the model performs over each class, a confusion matrix was constructed. Furthermore, a None class was added to the CM, representing class objects the model found, but they were not in the set of the testing annotations.

Metric	**Simple split (0.6 threshold)**
Accuracy	0.800
Precision	0.847
Recall	0.934
F_1 score	0.889
mAP/wmAP	0.611/0.870

Table[1]: Simple split results

From the results in table No.2 the precision for each class can be calculated:

$$Pr_{person} = \frac{4829}{4829 + 415} = 0.920$$

$$Pr_{biker} = \frac{738}{738 + 582} = 0.559$$

$$Pr_{bus} = \frac{22}{22 + 8} = 0.733$$

$$Pr_{car} = \frac{14}{14} = 1$$

Person	**4829**	582	0	0
Biker	415	**738**	0	0
Bus	0	0	**22**	0
Car	0	0	8	**17**
None	410	173	1	6
	Person	**Biker**	**Bus**	**Car**

Table[2]: Confusion Matrix

The experiment was repeated for confidence score threshold 0.7 and the Intersection over Union threshold remained the same at 0.5.

Metric	Simple split (0.7 threshold)
Accuracy	0.746
Precision	0.776
Recall	0.950
F_1 score	0.854
mAP/wmAP	0.596/0.847

Table[3]: Simple split results confidence threshold 0.7

Person	**4661**	898	0	0
Biker	630	**661**	0	0
Bus	0	0	**25**	0
Car	0	0	6	**17**
None	207	67	1	6
	Person	**Biker**	**Bus**	**Car**

Table[4]: Confusion Matrix for confidence threshold 0.7

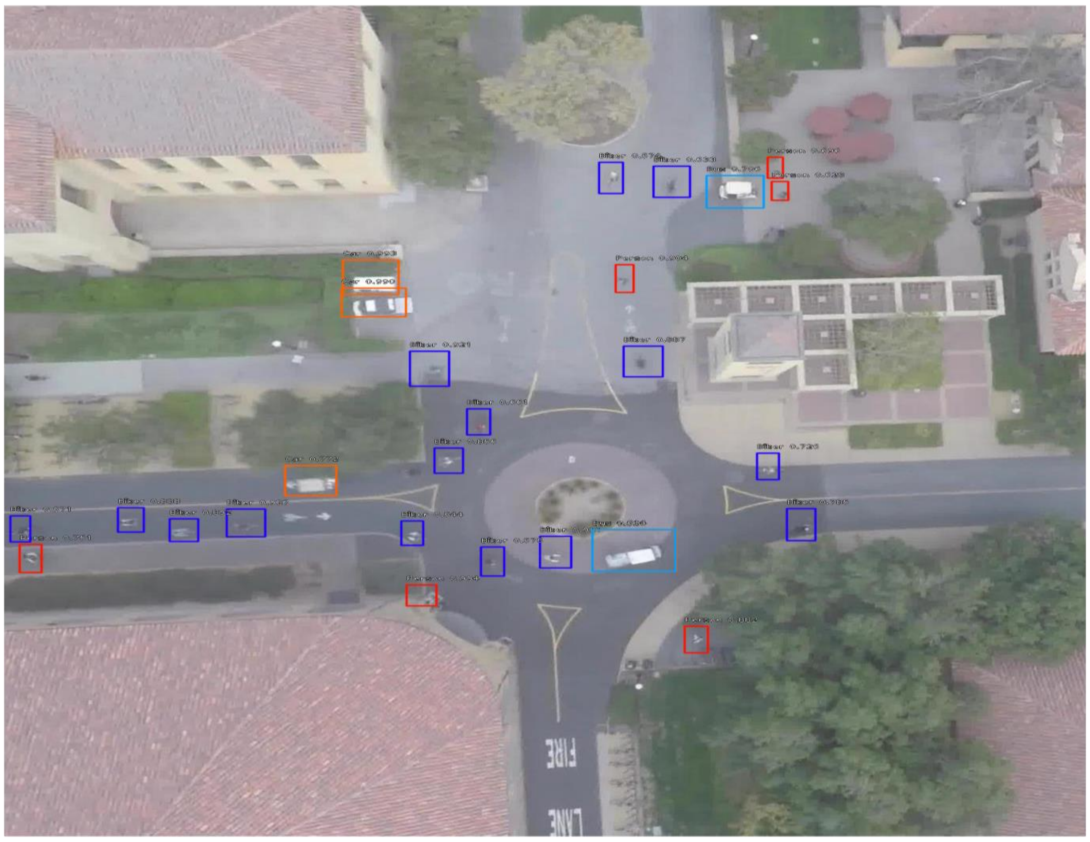


Figure 44: Example of Detection

6.2) K-fold Cross-Validation

Cross-validation is a technique for evaluating further a deep learning model and testing its performance. There are various CV algorithms, but for this evaluation the K-fold CV was chosen.

K-fold Cross-Validation

```
1  Split Data into Training/Testing sets.
2  Split the Training dataset into k equal (if possible) parts called folds.
3  For each fold:
4      | Train the model using the remaining training data.
5      | Evaluate the model using the current fold.
6      | Evaluate the model in the Testing set.
7  Extract a mean value from every Validation and Testing results.
```

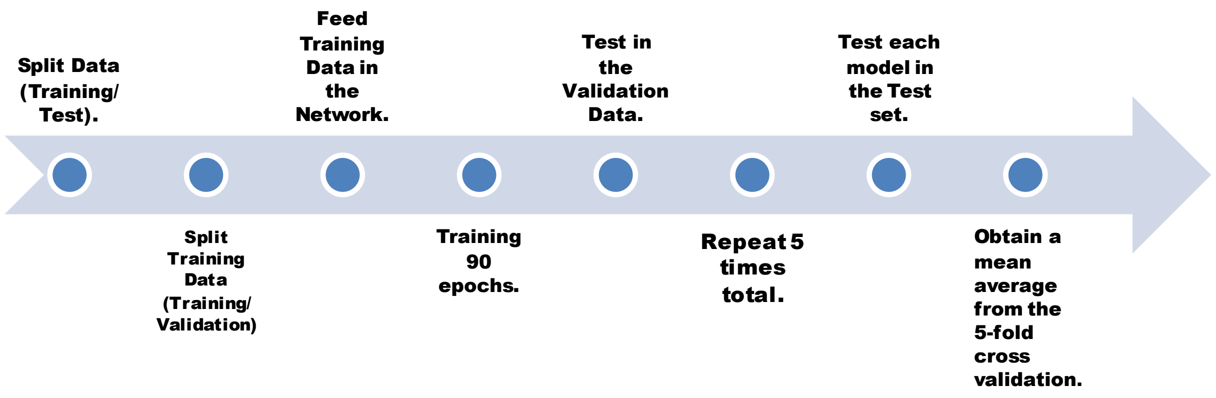

Split Data (Training/ Test).
Split Training Data (Training/ Validation)
Feed Training Data in the Network.
Training 90 epochs.
Test in the Validation Data.
Repeat 5 times total.
Test each model in the Test set.
Obtain a mean average from the 5-fold cross validation.

Figure 45: 5-fold CV pipeline

For convenience, the training set was split in 5 folds. For each validation and testing the Accuracy, Precision, Recall and F_1 score metrics were calculated. Finally, the mean values of those metrics were calculated as can be seen in Matrix No.5.

	1st fold		2nd fold		3rd fold		4th fold		5th fold		Mean	
Accuracy	0.795	0.789	0.792	0.748	0.584	0.657	0.689	0.742	0.821	0.796	0.736	0.746
Precision	0.865	0.857	0.866	0.853	0.818	0.849	0.841	0.859	0.853	0.852	0.848	0.854
Recall	0.908	0.899	0.902	0.900	0.670	0.655	0.792	0.788	0.955	0.943	0.845	0.837
F_1 score	0.886	0.888	0.883	0.886	0.737	0.711	0.816	0.817	0.897	0.901	0.843	0.840
	Val	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**

Table[5]: 5-fold Metrics

7. Two-phase FPN Retina model

In order to improve the RetinaNet model further, another layer of convolutions added. The improved RetinaNet utilizes an extra bottom-up pathway to deepen the network and further improve the quality of features extracted in each convolution block.

As seen in the image above the architecture has not changed drastically. The loss function remains the same as the original paper proposed, since the dataset is full of imbalances and Focal Loss is the best choice when it comes to class imbalances. The box regression loss also remains the same.

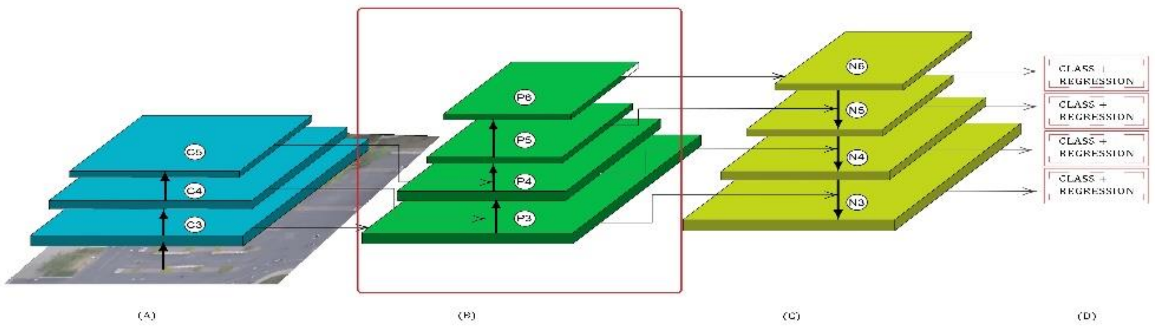


Figure 46: Modified RetinaNet

The extra bottom-up pathway is a combination of the Convolution blocks C3-C5, in order to produce even better high resolution / semantically low and low resolution / semantically high features. Since the output of C4 and P3 after the appropriate convolutions are the same, the kernel function is altered so that the merged outcome (P4) is a combination of features produced by different convolution kernels. As seen in Figure 46, an extra level has been added in the pyramid P6, which is simple another convolution block.

The rest of the architecture is the same as the original RetinaNet, with the difference that P6 passes through a 1x1 Convolution and produces N6. This new block, N6, contributes in the creation of N5 block after it gets up-sampled and merged with P5. Also since the N6 block has been created through a series of convolution, holds high quality features, therefore through a 3x3 convolution the output of N6 is fed as input into the classification and regression networks.

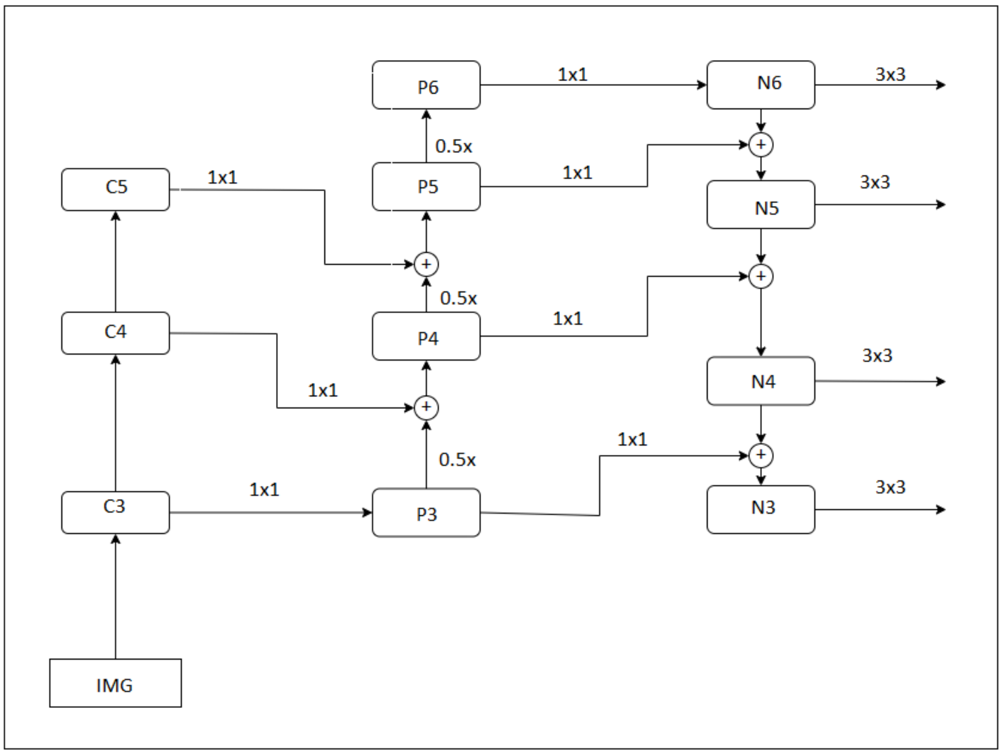


Figure 47: The new architecture

Just like the original Retina, the modified Retina was evaluated through Simple Split and k-fold Cross Validation. The training and testing process was the same as it was described in section 6.1 and 6.2. The following table displays the results for Simple Split evaluation of the model.

Metric	Simple split (0.6 threshold)
Accuracy	0.849
Precision	0.887
Recall	0.943
F_1 score	0.914
mAP/wmAP	0.632/0.897

Table[6]: Simple split results

From the results in matrix No.7 the precision for each class can be calculated:

$$Pr_{person} = \frac{4972}{4972 + 355} = 0.933$$

$$Pr_{biker} = \frac{842}{842 + 504} = 0.625$$

$$Pr_{bus} = \frac{22}{22 + 8} = 0.733$$

$$Pr_{car} = \frac{14}{14} = 1$$

Person	**4972**	504	0	0
Biker	355	**842**	0	0
Bus	0	0	**23**	0
Car	0	0	8	**17**
None	397	192	1	6
	Person	**Biker**	**Bus**	**Car**

Table[7]: Confusion Matrix

	1st fold		2nd fold		3rd fold		4th fold		5th fold		Mean	
Accuracy	0.811	0.808	0.776	0.756	0.603	0.688	0.749	0.777	0.860	0.819	0.759	0.770
Precision	0.887	0.846	0.844	0.877	0.834	0.854	0.866	0.898	0.901	0.875	0.866	0.870
Recall	0.912	0.901	0.909	0.920	0.705	0.703	0.819	0.800	0.963	0.966	0.861	0.858
F_1 score	0.899	0.873	0.875	0.897	0.764	0.771	0.841	0.846	0.930	0.918	0.861	0.861
	Val	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**	**Val**	**Test**

Table[8]: 5-fold Metrics

8.1) Comparison with RetinaNet

Due to the extra Convolution blocks in the modified RetinaNet, the training time is significant bigger, compared to the original Retina architecture. But since the goal of this diploma thesis is to compare the models only in terms of the metrics specified in previous section, the time complexity for the training and testing has not been taken in consideration.

When it comes to metrics such as Accuracy, Precision, Recall and F1 score, the modified model seems to perform better than the original:

- **6%** increase in terms of **Accuracy**
- **4%** increase in terms of **Precision**
- **1%** increase in terms of **Sensitivity**
- **2%** increase in terms of **F1-score**

All the above lead to the conclusion, that the extra layer of Convolution blocks, as well as the extra depth of the model improved the performance of the model in terms of "accuracy" metrics. It is worth noted that both models were trained with the same data for the same number of epochs. Furthermore, they both share the same loss functions for classification and regression loss.

8.2) Comparison with other models

The purpose of this work is to compare various Deep Learning models on the Stanford Drone Dataset and extract results about their performance. To make the comparison possible the following results have been taken by Mahdi Maktab Dar Oghaz, Manzoor Razaak, and Paolo Remagnino and their paper with title *"Enhanced Single Shot Small Object Detector for Aerial Imagery Using Super-Resolution, Feature Fusion and Deconvolution" [20].*

In their work they compared various single-stage and two-stage detectors, based on their mean Average Precision. In Matrix No.9 the comparison of YOLOv3, FasterRCNN, SSD, RetinaNet and modified Retina are displayed.

Model	mAP(%)
FasterRCNN	59.60
SSD	64.31
YOLOv3	57.42
RetinaNet	61.10
Modified Retina	63.27

Table[9]: Comparison of the based and proposed models with the literature

9. Conclusions

The two-phase FPN modified RetinaNet:

+ Outperformed the base model by 6% in terms of Accuracy
+ Outperformed most of the detectors in the literature
+ Suitable for small object detections such as people, cars, etc
+ Can be deployed in applications where accuracy is a critical factor

On the other side, the proposed model also has its drawbacks:

- Requires extremely high computational power systems for training
- Requires significant more time for training and prediction than the base model

9.1) Limitations

Deep Learning is a science that requires strong computational power in order to perform experiments in real data, with models that have millions of trainable parameters. Unfortunately, the Google Colab platform and their limited access to GPU made this study a rather challenging task. Moreover, the complexity of the models and the large size of the dataset demanded huge time amount to complete the training. As result, the margin for error was really small, because a failed training phase could mean hours of lost work.

9.2) Future Work

The whole comparative study was a good exercise to learn an object detector from start to finish, from the design and training process, to testing the model in real data and extract critical results. Furthermore, the modifications on the model helped understand how a simple change in the architecture can have significant impact in the accuracy of the model. A good way to continue this study will be to fine tune even more the training parameters of the model while also test various other loss functions such as the Kullback-Leibler divergence.

It is also worth mentioned that both of the models that were trained, had as input data from the Stanford dataset. That means that if we test the model with images or videos that have different angle or the camera is at different height, the models might not be able to detect correctly the subjects. One way to overcome those problems will be to use fusion. Aerial images from various angles and different heights so that the model can be used in to a real-life application.

Last but not least, it will be very interesting to deploy the modified Retina model into a real UAV or another surveillance system and try to make some real-life predictions.

References

[1] Object Detection, Wikipedia the free Encyclopedia.

[2] Keiron O'Shea, Ryan Nash, "*An Introduction to Convolutional Neural Networks*", November 2015.

[3] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar, *"Focal Loss for Dense Object Detection"*, February 2018.

[4] Karen Simonyan, Andrew Zisserman, *"Very Deep Convolutional Networks for Large-Scale Image Recognition"*, April 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, *"Deep Residual Learning for Image Recogntion"*, December 2015.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", May 2015.

[7] Tsung-Yi Lin, Piotr Dollar, Ross Girshick1, Kaiming He1, Bharath Hariharan1, Serge Belongie, *"Feature Pyramid Networks for Object Detection"*, April 2017.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, *"Rich feature hierarchies for accurate object detection and semantic segmentation"*, October 2014.

[9] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers , A.W.M. Smeulders, *"Selective Search for Object Recognition"*, February 2013.

[10] Ross Girshick, *"Fast R-CNN"*, February 2016

[11] Jan Hosang, Rodrigo Benenson, Bernt Schiele, "Learning non-maximum suppression", May 2017.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, *"Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks"*, Januray 2016.

[13] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, *"You Only Look Once: Unified, Real-Time Object Detection"*, May 2016.

[14] A. Robicquet, A. Sadeghian, A. Alahi, S. Savarese, *"Learning Social Etiquette: Human Trajectory Prediction In Crowded Scenes in European Conference on Computer Vision"*, 2016.

[15] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, Silvio Savarese, *"Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression"*, April 2019.

[16] Hossin, M., Sulaiman, M.N, *"A Review on Evaluation Metrics for Data Classification Evaluations"*, March 2015.

[17] Nvidia, Nvidia Tesla P100, *"The Most Advanced Datacenter Accelerator Ever Built Featuring Pascal GP100, the World's Fastest GPU"*, Whitepaper.

[18] Nvidia Tesla V100 GPU, *"The World's Most Advanced Data Center GPU"*, Architecture, Whitepaper.

[19] Daniel Berrar, *"Cross-Validation"*, January 2018.

[20] Mahdi Maktab Dar Oghaz, Manzoor Razaak, Paolo Remagnino,*"Enhanced Single Shot Small Object Detector for Aerial Imagery Using Super-Resolution, Feature Fusion and Deconvolution"*.