Region Based Object Detectors for Recognizing Birds in Aerial Images

A Thesis

Presented to

The Faculty of the Graduate School

At the University of Missouri

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

Implemented and Defended by:

Pedro Carrion Castagnola

Prof. Yi Shang, Advisor

May 2019

ACKNOWLEDGMENTS

I would like to thank Dr. Yi Shang for his guidance, support throughout my program, and for giving me the opportunity to work in this research. Also, I would like to thank the thesis committee members Dr. Guilherme DeSouza and Dr. Praveen Edara for his advice.

A special thanks to Peng Sun and Yang Liu from the Distributed and Intelligent Computing Lab for sharing their knowledge with me. Also, special thanks to Joel Sartwell and Andy Raedeke from the Missouri Department of Conservation for their support.

I would also like to thank Andrew Gilbert and Heath Hagy from the Illinois Natural History Survey, Forbes Biological Station, University of Illinois, Champaign, for their work in creating the bird dataset.

I would like to thank the representatives of the Missouri International Center. Especially to Lauren Pate and Jillian Kay who supported me during my program.

I would also like to thank my family and friends for their love, friendship, support, and for motivating me to do my best.

Finally, I would like to especially thank LASPAU and the Fulbright scholarship program for giving me the opportunity to do academic research and to live enriching experiences.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This project explores different types of deep neural networks (DNNs) for recognizing birds in aerial images based on real data provided by the Missouri Department of Conservation. The pipeline to identify birds from an image consist of two phases. First, region proposals are created by a DNN, where each region proposal is a sub-area of the image that possibly contains a bird. Second, a DNN is trained as a bird classifier using these region proposals. The bird detection performance is evaluated using the Precision, Recall and F1 scores on a separate test dataset. For the region proposal phase, a Region Proposal Network (RPN) has been implemented and tested, obtaining a Recall above 0.99, which means that the region proposal boxes cover almost all the birds. For the classification phase, a modification of Fast Region-based Convolutional Neural Network (Fast RCNN), a simple Convolutional Neural Network (CNN), and a Capsule Network, have been implemented and tested. For all of them, different hyper-parameters have been explored to increase the final F1 score. These models have been evaluated using two bird dataset variants: easy (with simple backgrounds) and hard (with complex background). Experimental results show that birds can be effectively recognized using the DNNs, especially in the easy dataset. Fast RCNN with a backbone architecture of ResNet50 and in conjunction with other techniques like Feature Pyramids Networks achieved the best results, with a maximum F1 score of 0.902. Simple CNN and Capsule Network achieved a score slightly above 0.8. The techniques used, datasets and results are analyzed to find the main causes of failures in some situations.

1. INTRODUCTION

The mission and vision of the Missouri Department of Conservation (MDC) are “to protect and manage the fish, forest, and wildlife of the state. Facilitate and provide an opportunity for all citizens to use, enjoy, and learn about these resources” [1]. This includes the protection and management of birds in different areas in Missouri.

The recognition and counting of birds in some areas can be used as an indicator to facilitate their protection. To calculate this information, aerial images can be taken over these areas. Then, the identification and counting of the birds can be done using image processing and artificial intelligence techniques instead of doing it manually which would be very difficult due to the large number of images and birds.

Many different image processing techniques can be used for image classification. Techniques with enormous progress over the last few years use Neural Network models which, in some cases, can even surpass humans on the recognition of objects [2]. Neural network techniques are a very active research area due to its excellent results.

For this research, the dataset “Little Birds on Aerial Images” (LBAI) will be used. This dataset contains aerial images of birds. LBAI dataset has been created using aerial images provided by the Andrew Gilbert and Heath Hagy from the Illinois Natural History Survey, Forbes Biological Station, University of Illinois, Champaign.

The goal is to localize all the birds on each input image. When using region-based object detectors there are two steps to achieve this goal. First, generate region proposals with a probability of finding any object on it. And second, use neural network models to classify these proposals.

The main motivation for this research is to try a relatively new type of Neural Network, Capsule Network, to recognize the birds. However, Capsule Network can only do classification and not localization of the birds. For this reason, a Region Proposal Network will be used to localize all the regions where there is a probability of finding a bird. This process is more effective than just use a sliding window or segmentation techniques. Also, with this pipeline, the same region proposals can be used on different Neural Network classifiers and their result can be compared.

This research has two main goals: evaluate the performance of different Neural Network classifiers and explore network parameters that can improve their results. The classification models that will be compared are ResNet50, Capsule Networks, and a simple Convolutional Neural Network.

2. RELATED WORK

In this section, the dataset and Neural Network models used for detection will be explained in detail.

2.1. Little Birds in Aerial Images (LBAI) dataset

LBAI dataset has been created using aerial images provided by the Andrew Gilbert and Heath Hagy from the Illinois Natural History Survey, Forbes Biological Station, University of Illinois, Champaign.

LBAI dataset creation is explained in the paper “Performance comparison of deep learning techniques for recognizing birds in aerial images” [3] developed by the Distributed and Intelligent Computing Lab of the University of Missouri. The dataset in this research corresponds to the “LBAI-B” dataset, which consists of 512 by 512 pixels images cropped from a larger image. All the images contain at least one bird and are divided into two categories: easy and hard. The hard dataset contains images with a more complex background than the images in the easy dataset, as for example, rivers, trees and different kinds of vegetations as backgrounds. Figure 1 shows examples of easy and hard images.

Easy and hard images are divided into 3 subsets each one: training, validation and test datasets. Only the training dataset will be used to train the model (weights of the network). The validation dataset will be used to select a model with a good generalization that does not overfit the training dataset. The validation dataset will also be used to explore how different parameters used during inference can improve the F1 score. Finally, after selecting the best model and parameters for the validation dataset, these will be used in the test dataset to calculate a final evaluation metric. Table shows the number of images for each of the subsets

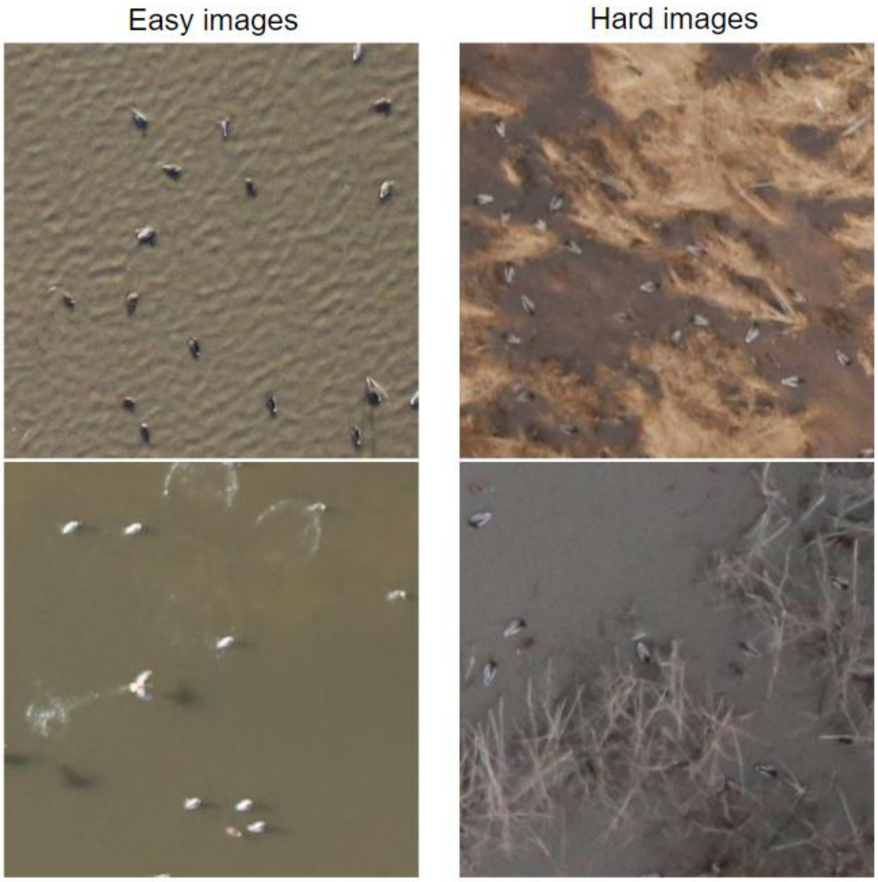


Figure 1. Samples of easy and hard images from LBAI dataset.

Dataset Difficulty	Subset	Number of images
Easy	Training	1730
Easy	Validation	484
Easy	Test	202
Hard	Training	1490
Hard	Validation	473
Hard	Test	93

Table 1. Number of images.

2.2. Neural Network Models for Image Classification

A Convolutional Neural Network, ResNet50, Feature Pyramid Network, and Capsule Network will be used in the pipelines for the region-based object detectors.

2.2.1. Convolutional Neural Network (CNN)

Convolutional Neural Network is a type of multi-layer neural network commonly applied for image analysis [4]. On a forward pass of an input image through the network, convolutional operations are performed on it, with filters of different sizes and weights. After the convolution operation, an activation function is applied like a sigmoid function or Rectified Linear Units (Relu). This is a convolutional layer and their outputs are known as feature maps. After this layer, is common to reduce the size of each feature map with a pooling operation. Many convolution and pooling operations usually are performed. The last part of the network consists of fully connected neurons and the final output correspond to the scores assigned to each possible class that needs to be predicted.

The network is trained using batches of training samples. A loss function for each batch is used in conjunction with the backpropagation algorithm to update the weights of the convolutional filters and neurons inputs. Finally, after many training iterations, these weights will be learned such that it can correctly classify the whole image.

2.2.2. ResNet50

Deep neural networks tend to be difficult to train. Adding more layers to a network is important to improve the results. However, when using too many layers the accuracy gets saturated and degrades [5]. ResNets (Residual Networks), solves this problem by taking the outputs of one layer and feed it into the output of another layer, deeper in the network. These shortcut connections do not add any extra parameter to the network and allows it to always gain accuracy as the network is deeper [5]. ResNet50 is a variant that uses 50 convolutional layers.

2.2.3. Feature Pyramids Network (FPN)

Feature Pyramids are a component to implement detection at different scales on neural network models. Its usage can improve the recognition of small objects [8]. Feature Pyramid Networks (FPN) can be used in conjunction with RPN or Fast RCNN backbones architectures (in this research, ResNet50 architectures).

FPN implements an architecture that consists of a bottom-up and a top-down pathway. The bottom-up pathway represents different feature maps at intermediate results from a ResNet architecture. Each feature map is usually half the size of the previous one, this means that it has a lower resolution but a semantically stronger value. The top-down pathway up samples the semantically strong feature maps using nearest neighbor up-sampling. Finally, the lateral connections are used to combine semantically strong feature maps with different resolution levels, which is the main purpose of using FPN [8]. The final predictions are made on each of these combined feature maps, as is shown in Figure 2.

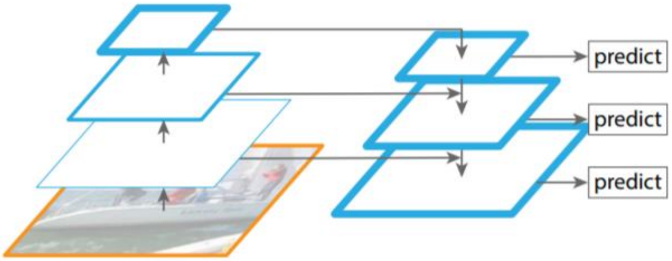


Figure 2. Feature Pyramid Network (FPN) pipeline. Source: from [8].

2.2.4. Capsule Network

Capsule network is a new type of Neural Network algorithm (2017), created by Geoffrey Hinton [6]. Capsule Network implements capsule layers. The main difference between capsule layers and convolutional layers are:

- Instead of using scalar outputs, it uses vector outputs
- Instead of any pooling technique, it uses dynamic routing by agreement algorithm.

Usually, the last two layers of a capsule network architecture are capsule layers and the rest are convolutional layers. Figure 3 shows an example of capsule network architecture for recognizing digits in the MNIST dataset [6]. For this architecture, “ReLu Conv1” and “PrimaryCaps” are the output of convolutional layers. “PrimaryCaps" is organized in groups, each group is an output of a convolutional operation over all the previous feature maps. Also, each group is composed of vectors. In Figure 3, there are 32 groups and each one contains 36 vectors of 8 elements each. The final layer “DigitCaps” contains 10 vectors, each one represents one output. All the vectors are also called capsules.

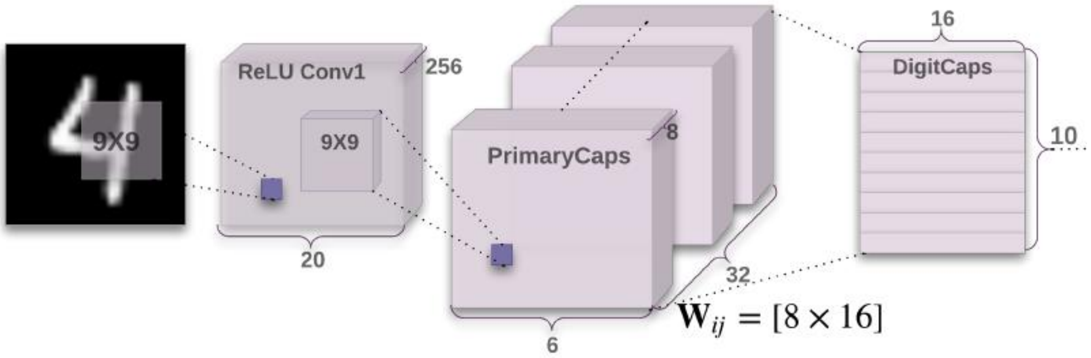


Figure 3. Capsule Network architecture for recognizing digits in the MNIST dataset. Source: from [6].

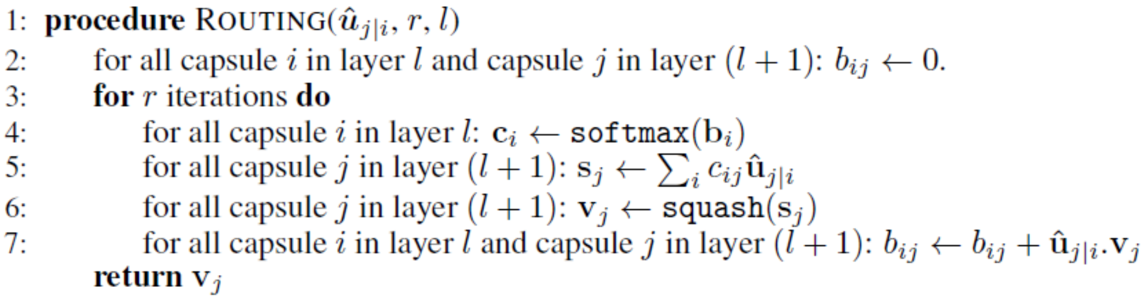
The vector outputs or capsules, has two main properties. First, its length represents the probability that an object exists. Second, its orientation represents instantiation parameters of an object, as for example position, size, rotation, etc. [6]

The output capsules are calculated using the following procedure. For each input capsule "i" and for each output capsule "j", a prediction "u" and a coupling coefficient "c" is calculated. The final output "s" is the weighted sum of the prediction and coupling coefficient.

$$\mathbf{s}_j = \sum_i c_{ij} \hat{\mathbf{u}}_{j|i} \, , \qquad \hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij} \mathbf{u}_i$$

The prediction is calculated by multiplying the input capsule vector by a weight matrix "W" (whose values will be updated during backpropagation). The coupling coefficient is calculated by measuring the agreement of the input and output vectors [6]. This agreement is calculated using the dynamic routing by agreement algorithm. All the coupling coefficient from one input vector to all outputs, sums to one so it represents how likely is to couple capsule "i" to output capsule "j". Also, this way of implementing the dynamic routing algorithm allows that an output object have a hierarchical relationship with the previous capsule, which means that there are hierarchical relationships between single and more complex objects. Thanks to this, it can correctly classify an object according to the spatial relationship of more single objects within it.

Figure 4 shows the dynamic routing by agreement algorithm [6]. The function “squash” normalizes the size of a vector between zero and one.



```
procedure ROUTING($\hat{u}_{j|i}$, $r$, $l$)
    for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$: $b_{ij} \leftarrow 0$.
    for $r$ iterations do
        for all capsule $i$ in layer $l$: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
        for all capsule $j$ in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
        for all capsule $j$ in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$
        for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
    return $\mathbf{v}_j$
```

Figure 4. Dynamic routing by agreement algorithm. Source: from [6].

Since each output vector contains estimation parameters of an object as pose (position, size, orientation), deformation, thickness, texture, etc. It is possible to reconstruct an image from the output vector (like an autoencoder network). This experiment was carried out in [6] with the MNIST dataset. Figure 5 shows how the reconstructions images change when one of the elements of the capsule vector slightly change and the name of the instantiation parameter it represents.

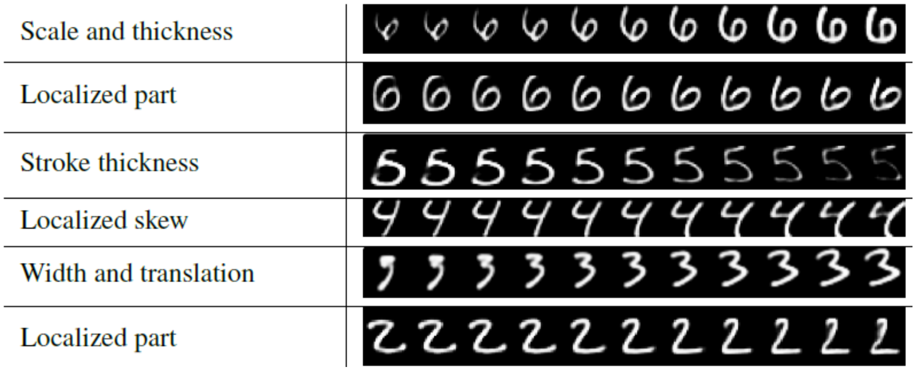


Figure 5. Reconstructions from an output capsule vector. Each row represents how they vary when one element of the vector slightly increases or decreases. Source: from [6].

3. DESIGN AND IMPLEMENTATION

This section shows the pipeline, configurations and other parameters that will be used in the final implementation of Region Proposal Network and the classification of proposal regions using different Neural Networks. It also explains how they will be implemented.

3.1. Dataset Labeling

In LBAI dataset each bird is not labeled by a bounding box. Instead, one coordinate point approximately at the center of each bird is known. From this point, a bounding box label is created assuming that all birds have the same size. However, not all the birds are the same size and some label points are not exactly in the center of the bird. Labels with a smaller size than the bird would make the bird more difficult to recognize and will impact in the final classification of the region proposals. In a previous research in [3] a 20 by 20 pixels label size is used, however, in this research 20 by 20 and a 30 by 30 label sizes will be used, and their results compared and discussed. Figure 6 shows some samples where 30 by 30 size labels captures a more complete shape of the birds.

Considering that LBAI dataset consists of easy and hard datasets, and that 20 by 20 and 30 by 30 pixels will be used for labels, there are in total four different datasets that will be used. Table 2 shows all these datasets. All the experiments will be carried out for each of these datasets.

Dataset number	Dataset Label Size (px)	Dataset Difficulty
1	20 x 20	Easy
2	20 x 20	Hard
3	30 x 30	Easy
4	30 x 30	Hard

Table 2. List of the four datasets used in this project.

Different detection methods have already been tested in [3] for LBAI. The results on this paper will not be compared with these results since it does not use the same evaluation script and in some cases use different label sizes.

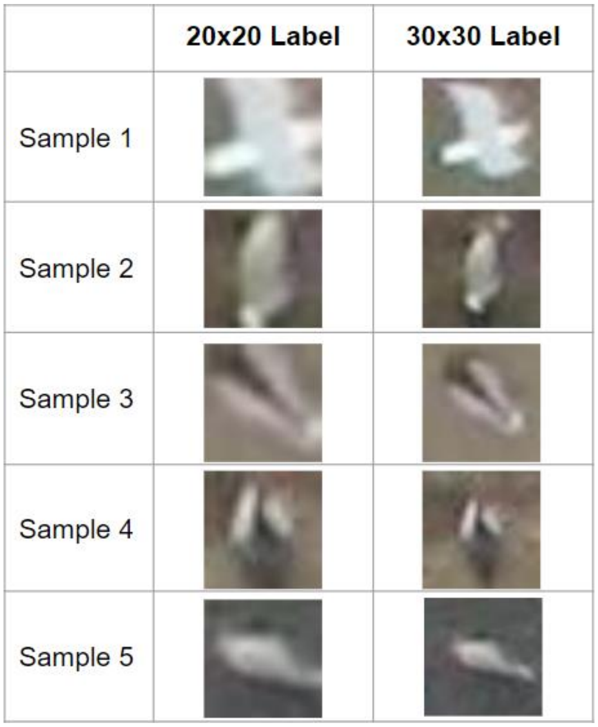


Figure 6. Ground truths difference using 20 by 20 and 30 by 30 pixels size when labeling.

3.2. Region Based Object Detector

A region-based object detector is composed of two stages. First, identify all possible regions where an object possibly exists. Different techniques could be used for generating region proposals like using a sliding window with a stride, using advanced image segmentation techniques or using Neural Networks. The region proposals are labeled as positive or negative samples according to its Intersection over Union (IoU) with the ground truth. The IoU measures how overlapped two areas are, its formula and a visual representation is shown in Figure 7. In the second phase, a classifier is used to predict the label of each region proposals. Figure 8 shows a region-based object detector pipeline.

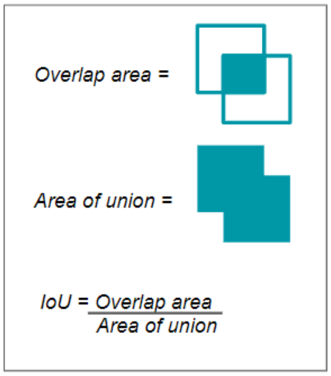


Figure 7. Intersection over Union (IoU)

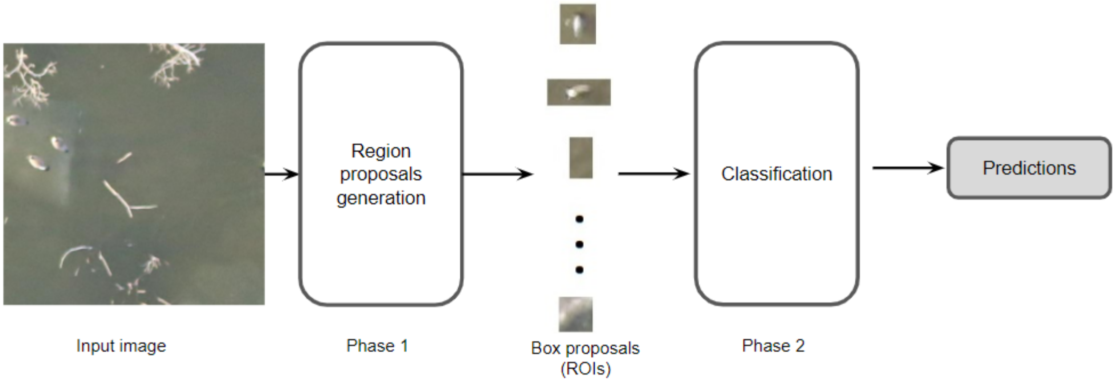


Figure 8. Region-based object detector pipeline.

For the region proposal phase, a Region Proposal Network (RPN) [7] will be applied. For the classification phase, three different kinds of Neural Networks will be applied: a ResNet50 using Feature Pyramid Network, a simple Convolutional Neural

Network and a Capsule Network. All these classification methods will be trained and tested using the same region proposals.

3.2. Region Based Object Detector Implementation

For the implementation of the codebase of a region-based object detector, two options were analyzed: Detectron [10] and Maskrcnn-benchmark [11]. Both programs have been created by the Facebook Research team to do research on object detection models. Both programs implement region-based object detections. However, the main difference is that Detectron uses a caffe2 framework [12] while Maskrcnn-benchmark uses Pytorch framework [13] which makes it faster. However, currently Maskrcnn-benchmark is not a stable program. For this reason, Detectron will be used for the implementation of Region Proposal Network, Feature Pyramid Network, and ResNet50.

Tool	Short description	Used to Implement
Detectron	Github project that can be used as codebase to test different state of the art neural networks on images. It has been developed using caffe2 framework	Region Proposal Network, Fast RCNN from precomputed proposals, Feature Pyramid Network, ResNet50
Pytorch	Python machine learning library. It provides tensor computation (work on GPU) and it can be used to implement deep neural networks	Simple Convolutional Neural Network, Capsule Network

Table 3. Programs and frameworks used for the implementation of Region-based object detectors.

For the creation of a simple CNN, Pytorch framework has been used [13]. Finally, for the creation of a Capsule Network model, a modification from the program

"Capsule-Network-Tutorial" [14], implemented using Pytorch, has been used. Table 3 shows a description and a list of the neural networks that each software implements.

Figure 9 shows the region-based object detector pipeline with the neural networks that are going to be implemented. The blue dotted line focusses the pipeline that is implemented in Detectron. Some simple changes have been made in the Detectron software to make the box proposals available to any other classification method, independently of the framework it uses. As is shown in Figure 9, the box proposals information is saved into a file after running the region proposal network inference. This file contains the box proposals coordinates, objectness scores, intersection over unions with ground truths and image file name it belongs. This file can be used with any classifier to train and test the classification network. Finally, the results containing the box proposals information plus a classification score for each one is saved into another file that can be used with the Detectron evaluation script to calculate the final classification metrics.

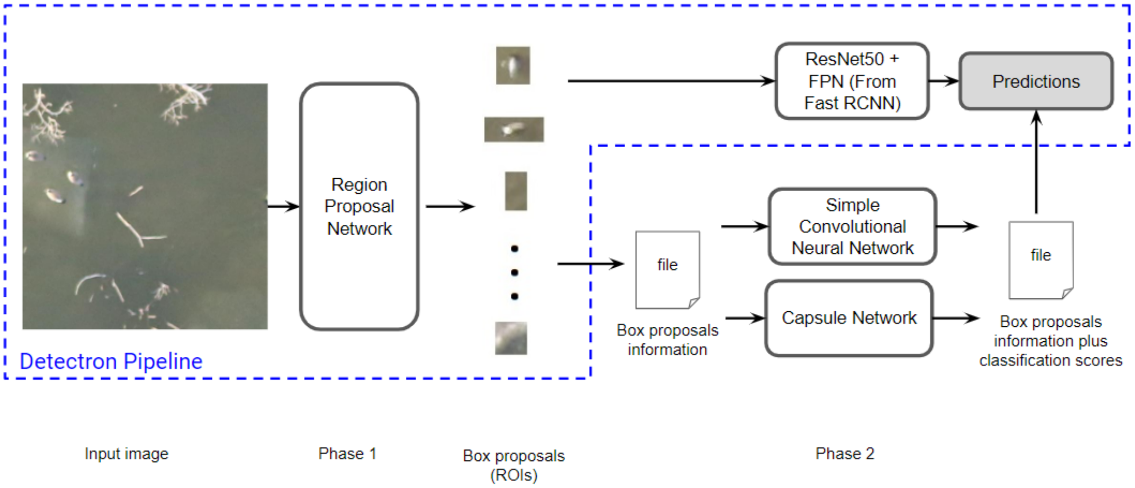

ResNet50 + FPN (From Fast RCNN)
Predictions
Region Proposal Network
Simple Convolutional Neural Network
file
file
Capsule Network
Box proposals information
Box proposals information plus classification scores
Detectron Pipeline
Input image
Phase 1
Box proposals (ROIs)
Phase 2

Figure 9. Region-based object detector pipeline and implementation.

3.3. Region Proposal Network

Region Proposal Networks (RPN) were first introduced in the implementation of Faster RCNN paper [7] as a technique to improve the training and inference time of Fast RCNN pipeline. The main idea of Faster RCNN pipeline is that a convolutional neural network can be used to generate region proposals and to make classifications at the same time [7].

An RPN takes an image as an input and outputs a set of proposal boxes, also called “Regions of Interest” or ROI. Each ROI have a score representing the probability of finding an object and its coordinates relative to the input image.

To generate region proposals, the pipeline in Figure 10 is applied. First, feature maps from the input image are created using a “backbone architecture”. For this project, a ResNet50 with Feature Pyramid Network (FPN) will be used. Then, one convolutional layer is applied to each possible position of a sliding window (blue square in Figure 10) in the feature maps. This sliding window has a corresponding relative position in the input image. Using the position of the center of this sliding window k regions are created. These k regions are called anchors (blue rectangles) and represent different portions of the input image by using different scales and aspect ratios for the sliding window [7]. The outputs of this pipeline have two fully connected layers “reg” and “cls”. “reg” layer predicts the coordinate regressions for each of the k anchors. “cls” layer predicts the foreground and background scores for each of the k anchors. The foreground score is also known as “objectness score” which is the probability of finding an object. Each anchor, including its coordinates regressors and objectness score, is a region proposal.

Since this pipeline uses Feature Pyramid Network (FPN), predictions at intermediate levels of the backbone architecture will be used as feature maps. This has not been included in Figure 10 for simplicity. The only difference is that a different anchor size will be used for each intermediate feature map. For this project, 5 anchor sizes have been employed, from 5 intermediate feature maps. For each anchor size, 1:1, 2:1 and 1:2 ratios between height and weight were used. This makes a total value of 15 anchors.

During training and inference, each anchor is labeled as a positive or negative input, depending on its intersection over union with the ground truth. The loss function is such that when it is minimized, a more accurate box regressor and objectness score, for each anchor, is obtained.

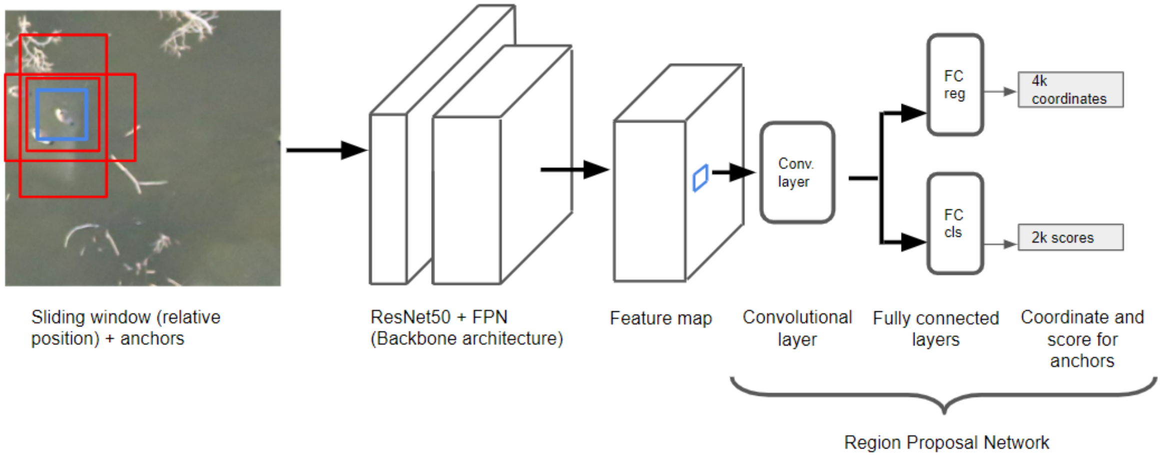


Figure 10. Region Proposal Network pipeline.

Many proposal boxes are generated (as many as possible sliding window position, times the number of anchors per input image). To reduce the number of proposals boxes some techniques are used. First, all the anchors that cross-boundary with the input image are removed. Second, a non-maximum suppression threshold is applied to the remaining boxes. This means that if two proposal boxes have an intersection over union higher than this threshold, the boxes with the lowest objectness score will be removed. Finally, only the top N boxes, based on their objectness score, are kept for future classification [7]. In the experiment section, the best value of N for the LBAI dataset is found. Figure 11 shows an example of the output region proposals (red squares) in one input image. It is not a problem that the regions of interest are highly overlapped, since most of them will be used only for training the classifiers and then removed with a non-maximum suppression operation after classification. The most important property of the box proposals is that they should cover the maximum amount of birds and that contain positive and negative samples to be used for the training of the classification networks.

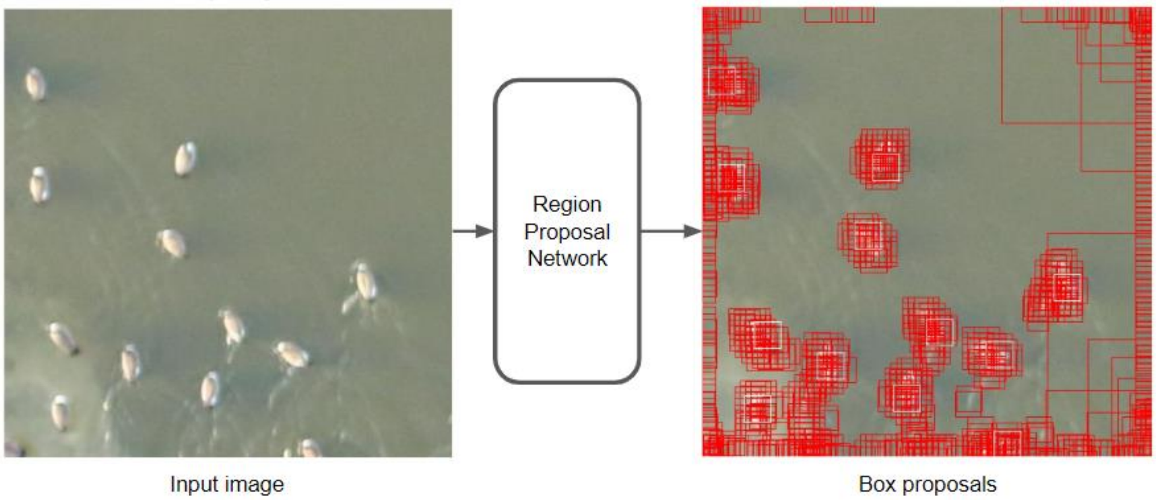


Figure 11. Region Proposals sample.

The pipeline in Figure 10 is already implemented in the Detectron software. However, some parameters are modified for LBAI dataset. Table 4 shows the most important parameters and their descriptions used for this research. The base learning rate, max iterations, and gamma were found by trying different values until a smooth learning curve is generated during training. Top 300, 800, 1000 and 2000 proposals will be evaluated in the experiment section. The rest of the RPN parameters have the values recommended in the Faster RCNN paper [7].

Parameter	Parameter description	Value
Base learning rate	Rate for adjusting the weights of the network.	0.001
Max iterations	Number of iterations for the network training.	90 000
Gamma	Learning rate equals $(Base\ learning\ rate) * (Gamma^{Step})$. For this model, step number is 0, 1 and 2 below iterations 60 000, 80 000 and 90 000 respectively.	0.1
RPN anchor start size	Since the network uses FPN, the anchor size is doubled at each FPN level. This network will use 5 levels.	10
RPN anchor aspect ratios	Aspect ratios to use for the anchors.	1:1, 1:2, 2:1
RPN non-maximum suppression threshold	If box proposals have an intersection over union higher than this value, the boxes with the lowest objectness score are removed.	0.7
Keep top box proposal predictions	Keep proposals with best objectness scores. 3 values will be evaluated at the implementation.	300, 800, 1000 and 2000

Table 4. Parameters for the Region Proposal Network implementation.

3.4. Mini-batches

The classification methods do not use all the box proposals (ROIs) for training the model. Instead, they use “mini-batches” which is a way to take random box proposals to improve the classification performance and to control the balance of the positive and negative samples for each training batch. Mini-batches were first introduced in the Fast RCNN paper [9]. The characteristics of mini-batches are the following:

- Take many random ROIs (box proposals) only for one or two images: this way some ROIs will be identical after the ROI pooling operation and removed.
- Balance input data: For some datasets, like COCO in Fast RCNN paper [9], having more background samples improved their metric score. In more simpler cases is possible to take the same number of samples for each class.
- Consider a ROI as foreground if its intersection over union with the ground truth is higher than a threshold.
- Consider a ROI as the background is its intersection over union with the ground truth is between two thresholds.
- Since random samples are taken, the training uses “iterations” instead of “epochs”.

Mini-batches are already implemented in the Detectron software. However, it could not be used for the simple Convolutional Neural Network and Capsule Network because they are not part of Detectron pipeline (see Figure 9). Because of this, an own implementation of mini-batches has been created to be used by these networks. The own mini-batch implementation is very similar to the one implemented in Detectron, the differences are:

- Instead of taking region proposals from one or two images, it takes random region proposals from any image.
- Instead of taking random input images, trains with all the input images. For this reason, it uses epochs instead of iterations.

Table 5 shows the parameters used in the configuration of mini-batches

Parameter	Parameter description	Value
Images per minibatch	Number of random images where the ROIS will be selected.	2
ROIs per minibatch	Number of ROIs to take from the "Images per minibatch".	128
Background - foreground minibatch ratio	Ratio of foreground and background samples for each mini batch.	50% - 50%
Foreground threshold	Intersection over union for box proposal to be considered a foreground sample.	0.5 or higher
Background threshold	Intersection over union for box proposal to be considered a background sample.	Between 0 and 0.5

Table 5. Parameters for the mini-batch implementation.

3.5. ResNet50 and Feature Pyramid Network Classification (Fast RCNN)

Fast RCNN is a region-based technique for object detection on images. It is composed of a Region Proposal section and a classification section. In the original paper different architectures for the classification section are tested [9]. In this research, only the classification part of Fast RCNN will be used.

Figure 12 shows the pipeline of this classifier. The region proposals (ROIs) will be the outputs of the region proposal network. These will be sampled using mini-batches. For each region proposal there is a 50% probability of flip images horizontally (image augmentation). The ROI pooling layers transform the proposals into the same size, without losing important information from the images. For this research, the technique ROI align will be used. It uses bilinear interpolation to find the value of each pixel of the reduced image [15]. The backbone architecture will use a ResNet 50 with Feature Pyramid Network. And the final fully connected layers predict the final class (bird or no bird) using a SoftMax operation. Table 6 shows the parameters, descriptions, and values used for the configuration of this network.

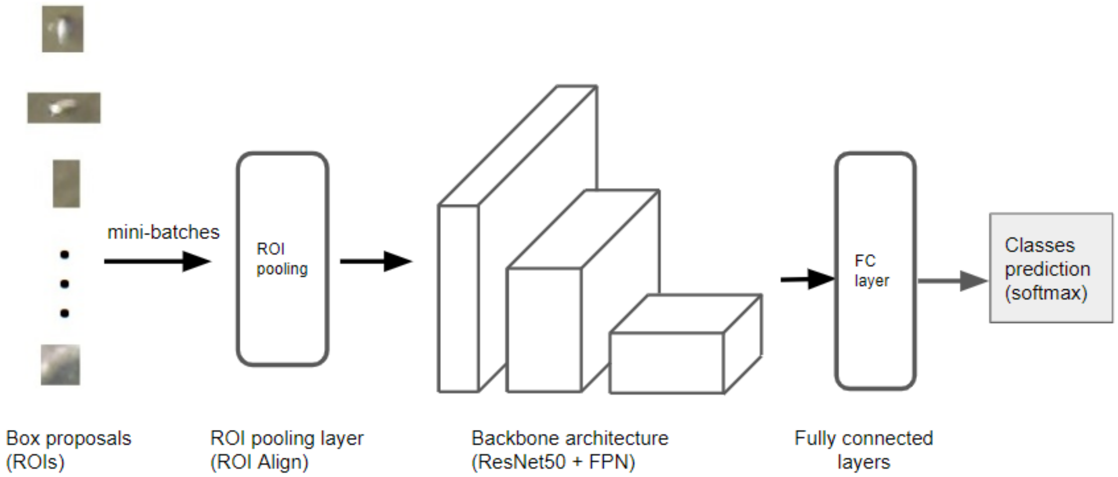

mini-batches
ROI pooling
FC layer
Classes prediction (softmax)
Box proposals (ROIs)
ROI pooling layer (ROI Align)
Backbone architecture (ResNet50 + FPN)
Fully connected layers

Figure 12. Fast RCNN with precomputed proposals pipeline.

Parameter	Parameter description	Value
Loss criterion, Optimizer	Loss function and optimization algorithm used during the training.	Own, SGD
ROI resizing	Technique to resize region of interest to a fixed size	ROI align
Augmentation (flip image horizontally)	Probability of flipping horizontally a region of interest from the mini-batch	50%
Number of training iterations	The number of iterations for the network training.	90 000
Base learning rate	The rate for adjusting the weights of the network.	0.0002
Gamma	Learning rate equals $(Base\ learning\ rate) * (Gamma^{Step})$. For this model, step number is 0, 1 and 2 below iterations 60 000, 80 000 and 90 000 respectively.	0.1

Table 6. Parameters for Fast RCNN implementation.

The original paper for Fast RCNN uses a different pipeline. The main difference is that the region proposal and the classification network shares feature maps. This makes the training much faster without losing accuracy. For this research, all the classification methods that are going to be compared must use the same box proposals. For this reason, the region proposal and the classification sections has been separated and the backbone architecture has been applied again in this pipeline to find the feature maps. The name Fast RCNN is being used in this research to name the ResNet50 with FPN classifier because it is the name used in the Detectron software. Since Region Proposal Networks and ROI align pooling techniques are used, Fast RCNN results should be very similar to the Mask RCNN and Faster RCNN methods.

3.6. Simple Convolutional Neural Network (SCNN)

Figure 13 shows the pipeline and the parameters of each layer of the simple Convolutional Neural Network architecture. Instead of using a ROI pooling technique to transform all the input region proposals into the same size, a simple resizing to a fixed size using aliasing filter is done. This has been implemented using the PIL python library. All proposal boxes will be cropped from the original images, resized to 30 by 30 pixels and used as inputs for the network. It is expected to not lose too much information given that the size of each bird is around 30 by 30 pixels or less. The simple convolutional neural network architecture consists of two sets of convolutional layers and max-pooling layers and, in the end, two fully connected layers that represents the probability of finding a bird or not. The values of the last layers are transformed using a SoftMax operation. Table 7 shows the parameters used for the network training.

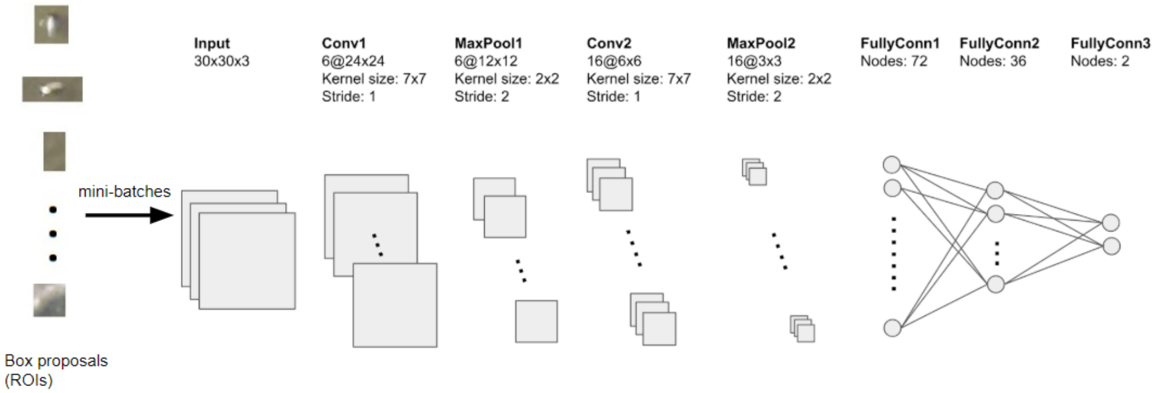


Figure 13. Simple Convolutional Neural Network architecture.

Parameter	Parameter description	Value
Loss criterion, Optimizer	Loss function and optimization algorithm used during the training.	Cross Entropy, Stochastic gradient descent
ROI resizing	Technique to resize region of interest to a fixed size	Aliasing filter
Augmentation (flip image horizontally)	Probability of flipping horizontally a region of interest from the mini-batch	50%
Learning rate	Rate for adjusting the weights of the network after each batch.	0.002
Momentum	Parameter used to reduce the learning rate during training.	0.9
Epochs	Total number of epochs to train the network.	50

Table 7. Main parameters for the implementation of the Simple Convolutional Neural Network.

3.7. Capsule Network (CapsNet)

The ROI sampling is identical to the used for the simple convolutional Neural Network. The architecture used for capsule Network classification is similar to the one used on "Dynamic routing between capsules" paper for the MNIST data [6]. The main difference is that a smaller number of feature maps are used in the first and second convolutional layers. This network uses two convolutional neural network layers, a primary capsule layer, and an output capsule layer. Figure 14 shows the parameters and architecture of the capsule network and Table 8 shows the parameters used for the capsule network training.

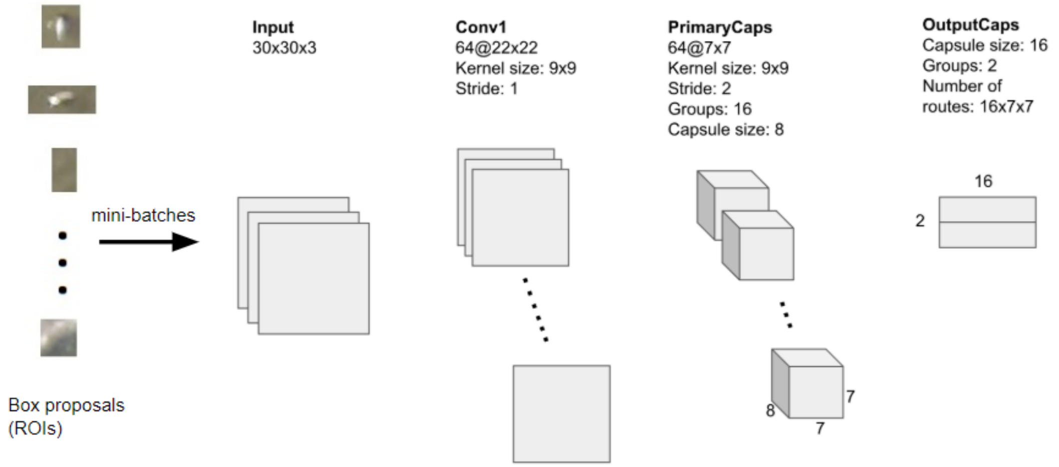


Figure 14. Capsule Network Architecture.

Parameter	Parameter description	Value
Loss criterion, Optimizer	Loss function and optimization algorithm used during the training.	Own, Adam
ROI resizing	Technique to resize region of interest to a fixed size	Aliasing filter
Augmentation (flip image horizontally)	Probability of flipping horizontally a region of interest from the mini-batch	50%
Learning rate	Rate for adjusting the weights of the network after each batch.	0.0001
Epochs	Total number of epochs to train the network.	50
Routing iterations	Routing iterations for calculating the agreement between the second to last capsule layer and the last capsule layer.	3

Table 8. Main parameters for the implementation of the Capsule Network.

As [6] suggests, the network can produce better results when including a reconstruction loss. This is the loss of the reconstructed image through a small reconstruction network. Figure 15 shows the reconstruction network that will be used for the reconstruction experiment.

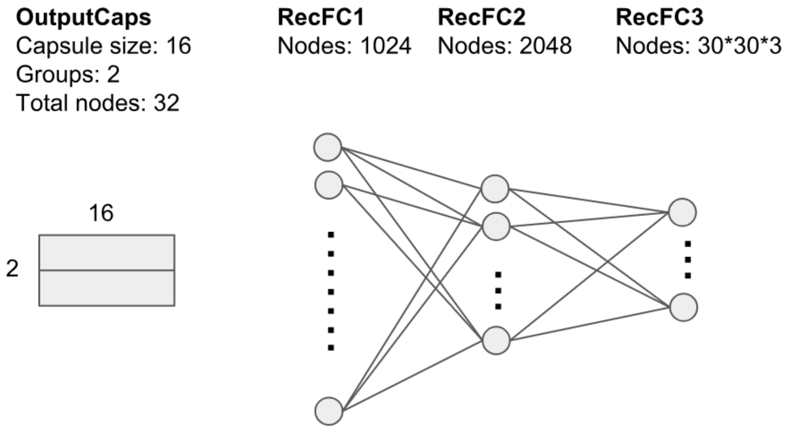


Figure 15. Output capsules reconstruction network.

4. EXPERIMENTS

This section shows the metrics used for the evaluation of the outputs of the networks, the experiments made to find a model that has a good generalization in the validation dataset, and the exploration of some parameters that improve the score metrics.

4.1. Region Proposal Network Evaluation

First, the metrics used to decide how good the outputs of a network are will be explained. Then, the selection of a good model that could generalize in the validation dataset will be explored. The last evaluation will find the best number of top box proposals to keep for each input image.

4.1.1. Metrics for RPN

It is important that the box proposals contain positive and negative samples that can be used for training and classification. It is also very important that the box proposal contains all the positive samples from the validation input images, otherwise the classifier will not be able to classify all the birds in this dataset.

As is suggested by the Faster RCNN paper [7], the metrics that will be used are the Recall and the Average Recall. The Recall measures the accuracy of the box proposals over all the ground truths. It will be used to measure the percentage of birds covered by the region proposals. The formula to calculate the Recall is:

$$Recall = \frac{tp}{tp + fn}$$

Where “tp” are true positive samples (region proposals that contains a bird). “fn” are the false negative samples (birds that were not covered by the region proposals). This is the value that should be minimized. A region proposal contains a bird if its Intersection over Union with the ground truth (see Figure 7) is above a threshold. By default, this threshold is 0.5.

The Average Recall will be used to know how good an RPN model is in comparison to others. To calculate the Average Recall the following procedure is carried out. First, calculate the Recall using different thresholds for the Intersection over Union of the Region Proposals with the ground truths. This defines whether each region proposal is a true positive instance or not. For this experiment, thresholds from 0.5 to 0.9 were used. Finally, the Average Recall is the average of all the Recall values. Figure 16 shows an example of the Recall versus the Intersection over Union Thresholds for one RPN model.

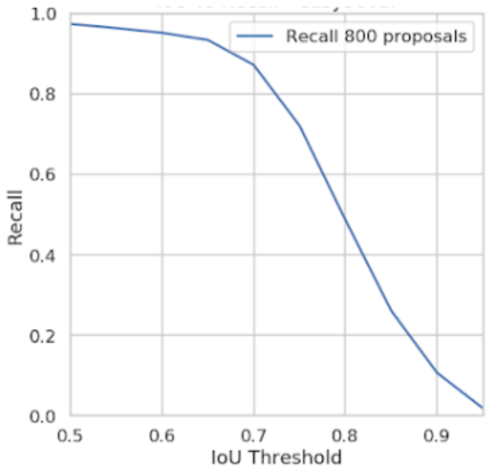


Figure 16. Example of a Recall versus Intersection over Union plot.

4.1.2. RPN Model Training

The Average Recall metric will be used to find a model that does not overfits the training dataset during the Region Proposal Network (RPN) training. It will also be used to evaluate how the models behave in each of the datasets. Four RPN will be trained, corresponding to the four LBAI datasets: the combinations of easy and hard datasets, and 20 by 20 and 30 by 30 labels. Figure 17 shows the average recall for the top 1000 proposals during training on the training and validation datasets. The selected model (weights of the network) will correspond to the iteration with the maximum Average Recall in the validation dataset. Table 9 shows information of the best Average Recall in the validation dataset.

The 30 by 30 label datasets generate much better Average Recalls than the 20 by 20 label datasets. Which means that the region proposal outputs will contain more positive samples. 30 by 30 labels datasets will have better classification results since they will have more positive samples to train and more birds that can be predicted in the test datasets.

The final selected model for the hard datasets tends to correspond to a much lower iteration than the easy datasets. The fact that the Average Recall of the validation dataset does not improves during training could just mean that the initial coordinates of the anchors were good enough to capture most of the birds in the validation datasets. The maximum average scores obtained does not have a significant variation between easy and hard datasets. However, the easy datasets tend to obtain a slightly better average recall. This could be due to the complex backgrounds in the hard datasets that makes the birds difficult to recognize.

Input Size	Dataset Difficulty	Dataset Type	Iteration of best Average Recall	Best Average Recall
20 x 20	Easy	Validation	80,000	0.539
20 x 20	Hard	Validation	40,000	0.543
30 x 30	Easy	Validation	80,000	0.651
30 x 30	Hard	Validation	20,000	0.630

Table 9. Selected models for the region proposal network.

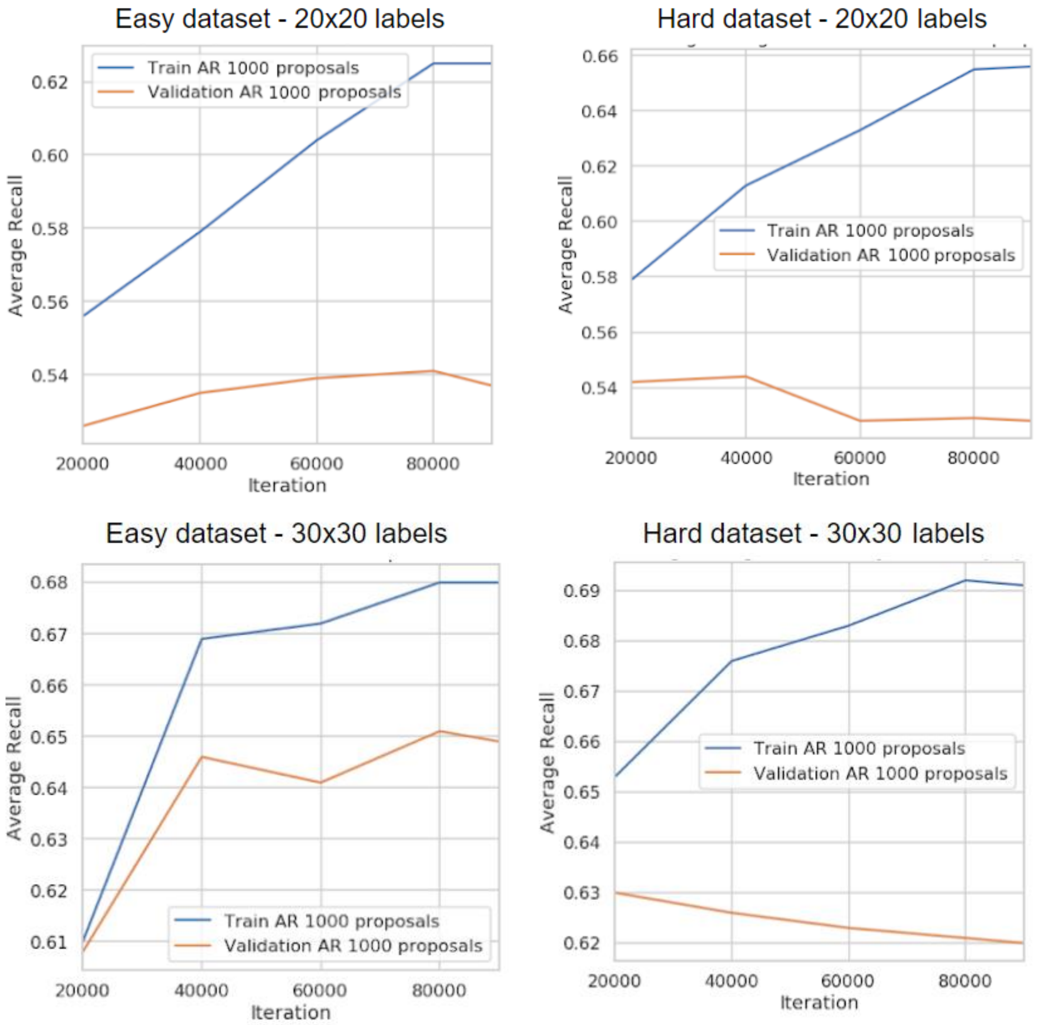


Figure 17. Average recall (AR) in training and validation datasets during training.

4.1.3. Top Region Proposals Selection

The goal of this experiment is to find the best value for the top proposals to keep for each input image in the validation dataset. As it was explained in the Region Proposal Network design section, after the RPN training each region proposal has an "objectness" score, which is a score associated with the probability of containing an object. Then, a non-maximum suppression can be used to remove some boxes. However, there are still too many boxes for each input image. By keeping a number of top proposals most of the boxes with a low objectness score will be removed. Boxes with low objectness scores do not contribute to the training of the classification network. Many of them contains aerial images of water from rivers which are almost solid colors. It is more important for the negative samples to contain information of other kind of backgrounds that are present in the image like light reflections on water, trees or rocks.

Figure 18 shows a graph of the recall versus the Intersection over Union thresholds for each of the four datasets. These graphs provide a visualization of the Recall over different Intersection over Union of the box proposal with the ground truths. A better model will have higher values at different thresholds. The graphs show that for all combinations of difficulty and label sizes, the top 800 proposals are enough to cover most of the birds in the images. The top 1000 proposals are almost identical than the top 800 proposals, in terms of Recall. The top 2000 proposals were also tested, but the results were identical to the top 1000 and were not included in the graphs. The top 800 proposals boxes will be selected as inputs for all the classifiers.

Table 10 shows the Recall and Average Recall of different number of proposals for the easy dataset using 30 by 30 labels. The column "Best Recall" can be interpreted as the percentage of birds capture for the region proposals. The Recall and the Average Recall shows that using more than 800 proposals does not improve the outputs of the region Proposal Network.

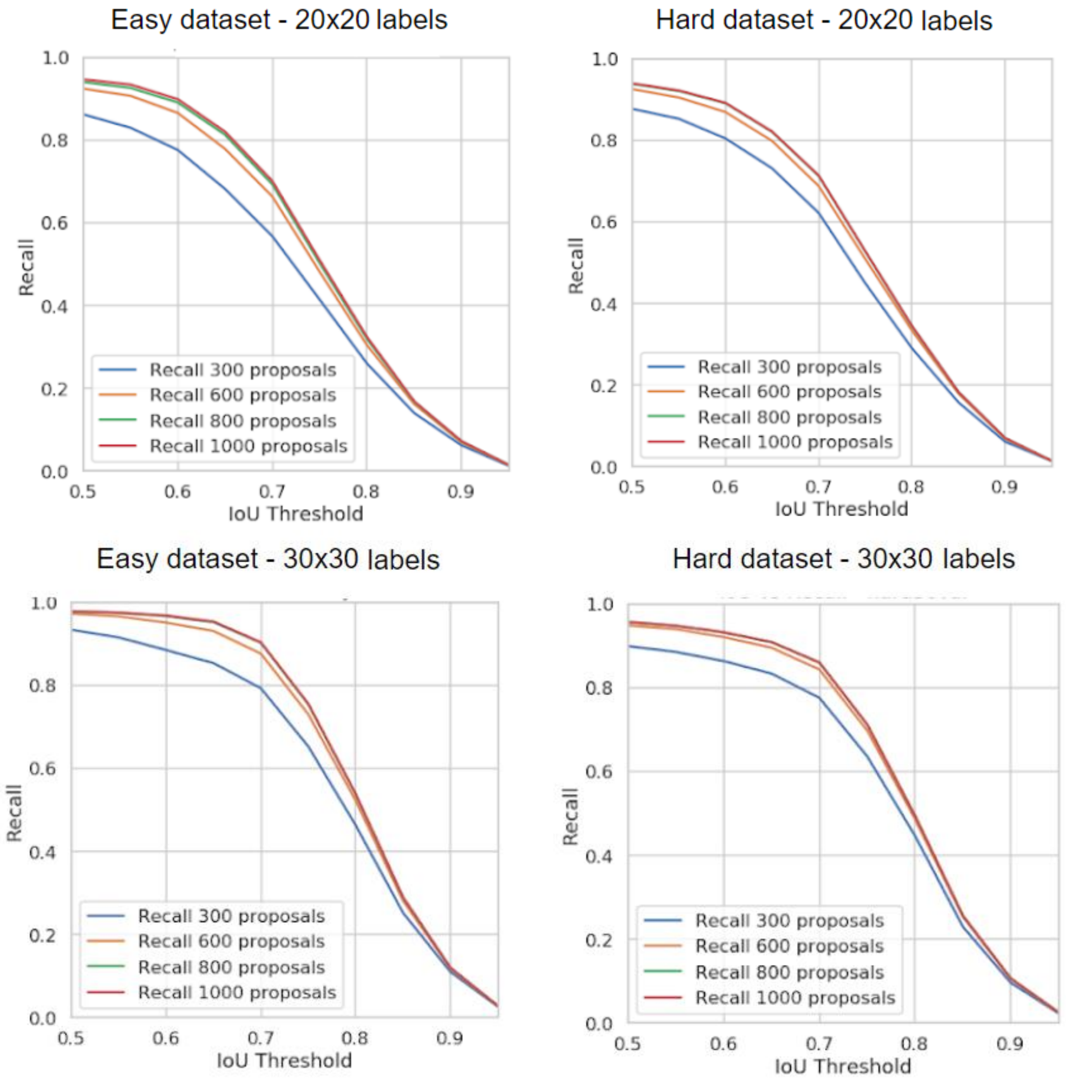


Figure 18. Recall versus Intersection over Union.

Input Size	Dataset Difficulty	Dataset Type	Number of top box proposals to keep	Best Recall (0.5 IoU threshold)	Average Recall
30 x 30	Easy	Validation	300	0.9339	0.589
30 x 30	Easy	Validation	600	0.9728	0.638
30 x 30	Easy	Validation	800	0.9780	0.651
30 x 30	Easy	Validation	1000	0.9784	0.651

Table 10. Metrics comparison for different number of top proposal boxes.

4.2. Object Detection Evaluation

For the evaluation of the training and final classification of the test proposals, the metrics Precision, Recall, and F1 will be used.

The precision measures how accurate our predictions are. The recall measure how accurate our predictions are over all the ground truth bounding boxes. Finally, the F1 score is the harmonic mean of precision and recall. The formulas of these metrics are shown below. “tp” are true positive, “tn” are true negative and “fp” are false positive instances.

$$Precision = \frac{tp}{tp + tn}$$

$$Recall = \frac{tp}{tp + fn}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

After the training of the classification model, a SoftMax operation will be performed to the two possible outputs, that represents “bird” or “no bird” categories. This means that each box proposal will have a “bird” score and a “no bird” score. For each region proposal the bird score, between 0 and 1, represents the probability of finding a bird in it. A box proposal will be considered as true positive if its intersection over union with the ground truth is, at least, 1 pixel.

4.2.1. Non-Maximum Suppression After Classification

After classifying all the region proposal, some filters can be applied to improve the score metrics. The first parameter that will be evaluated after classification is the non-maximum suppression threshold. All box proposals whose intersection over union with other box proposal is higher than this threshold will be removed and only the box with the highest score is kept. When two or more box proposals intersect the same ground truth bounding box, only the box with the highest bird score is considered as true positive. The other boxes are considered as a true negative instance only if they do not intersect any other ground truth. Figure 19 shows an example of this case. In this example, two box proposals (red squares), RP1 and RP2, are overlapped and each of them has a bird score. The classifier is expected to assign a higher score to RP1 since it contains more information of the bird. If the non-maximum suppression is not applied, RP2 will end up as a true negative instance and the Precision will decrease. However, if RP2 predicts a different bird in the image, the Precision would be right. In conclusion, the best value for this parameter depends on how overlapped the birds are in the dataset.

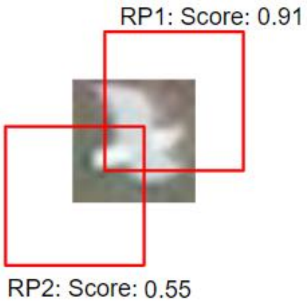


Figure 19. Overlapped region proposal over a ground truth bounding box.

Figure 20 shows the evaluation scores for a Fast RCNN classifier made at different non-maximum suppression thresholds. A low threshold means that most of the overlapped boxes will be removed. A high threshold means that the overlapped proposals boxes will be kept. Using a low threshold for the non-maximum suppression operation increases the overall Precision in the validation dataset. This means that in most of the cases the birds are not overlapped. The value 0.01 for non-maximum suppression after classification will be used on all the future evaluations.

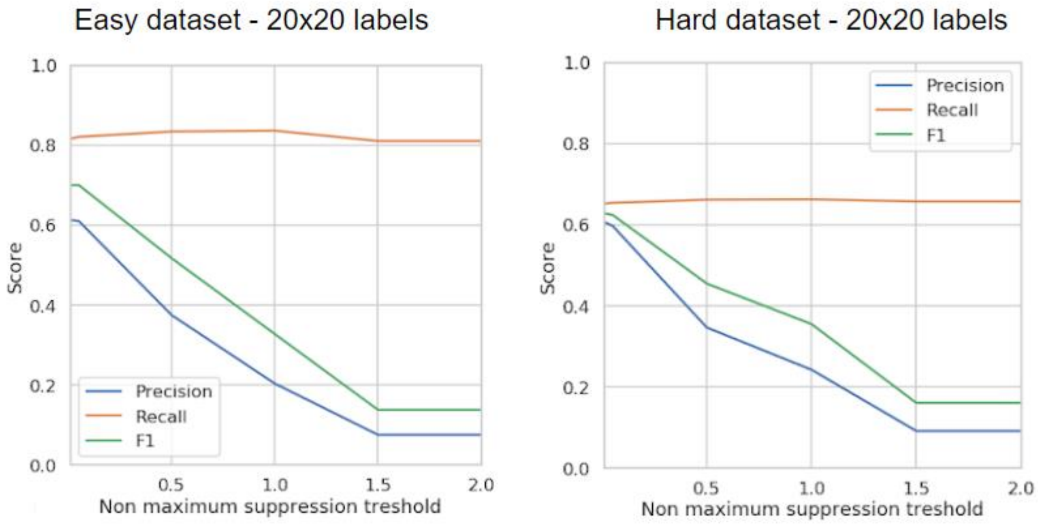


Figure 20. F1, Recall and Precision at different non-maximum suppression thresholds.

4.2.2. Minimum Score Threshold

After the classification, all the region proposals have a “bird” score. This score represents the probability that the box proposal contains a bird or not. Usually, a score threshold of 0.5 is used to separate bird and no bird. However, in some cases, a classifier can assign a score slightly above 0.5 to a region proposal with a low probability of being a bird.

All region proposals with a bird score lower than the minimum score threshold will be removed. The proposed method to increase the final score on the test dataset is to find a threshold that increases the F1 score on the validation dataset and then apply the same threshold for the test dataset. If the test dataset has the same distribution of the validation dataset then using its best minimum score threshold can improve the test F1 score. Since all the classifier produces very different bird scores, this parameter will be evaluated separately for each classification method and dataset.

The minimum score threshold balance the Precision and Recall scores. A higher threshold will produce a higher Precision score and a low threshold a high Recall. Figure 21 shows how the three metric scores varies over different thresholds for a fast RCNN classification.

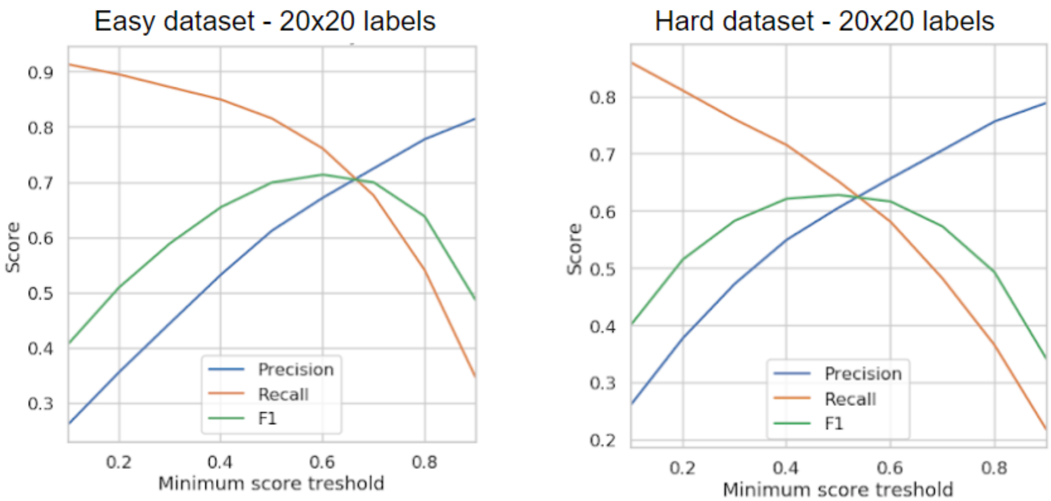


Figure 21. F1, Recall and Precision at different minimum score thresholds.

4.3. Classification Model Training

For each classification method, a model (weights of the network) will be selected. Also, there will be analyzed how the dataset label size and dataset difficulty impact the results during training.

4.3.1. ResNet50 with Feature Pyramid Network Classification (Fast RCNN)

Detectron allows saving models during the training process. Every 2000 iterations, all the weights of the models are saved. These models can be used to evaluate the training process. For simplicity, the F1 score will be calculated on the training and validation dataset for each of the models saved during the training process. Figure 22 shows the F1 score on the training and validation datasets during the training process. Table 11 shows the selected models for each dataset and the maximum F1 score reached in the validation dataset. There is not a significative difference between the 20 by 20 and 30 by 30 labels scores. This means that, if the validation and test datasets have similar distributions, there will not be a significative difference in the

Precision metric between in these two kinds of labels. However, the label size could contribute to a distribution difference between the test and validation datasets. As expected, the hard dataset obtained a lower F1 score during training due to its complex backgrounds.

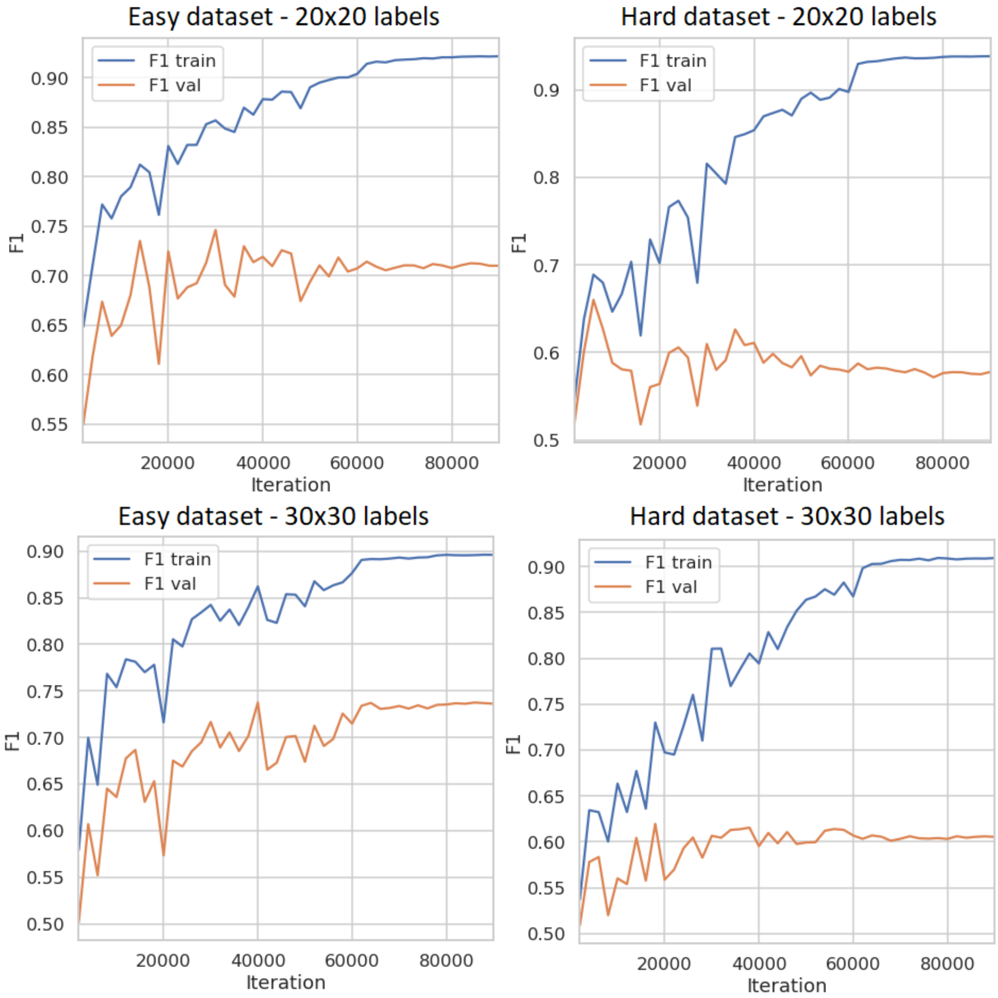


Figure 22. F1 score during training in the training and validation for the fast RCNN classifier.

Input Size	Dataset Difficulty	Iteration of Best Model	Best F1 Score
20 x 20	Easy	30,000	0.7463
20 x 20	Hard	6,000	0.6603
30 x 30	Easy	40,000	0.7375
30 x 30	Hard	18,000	0.6196

Table 11. Selected models for the Fast RCNN classification.

4.3.2. Simple Convolutional Neural Network Classification (SCNN)

For the Simple Convolutional Neural Network, the losses of the training and validation datasets were saved after each epoch. Figure 23 shows these values during training. The model that generates the minimum loss on the validation dataset will be selected.

Table 12 shows the information about the selected models. The results show that the 30 by 30 label datasets achieved lower losses than the 20 by 20 labels. This is the expected behavior since the RPN experiments demonstrated that the 30 by 30 label datasets capture more information of the birds.

Input Size	Dataset Difficulty	Epoch of Best Model	Best Loss
20 x 20	Easy	47	0.4830
20 x 20	Hard	47	0.4535
30 x 30	Easy	41	0.4323
30 x 30	Hard	42	0.3923

Table 12. Selected models for the simple convolutional neural network classification.

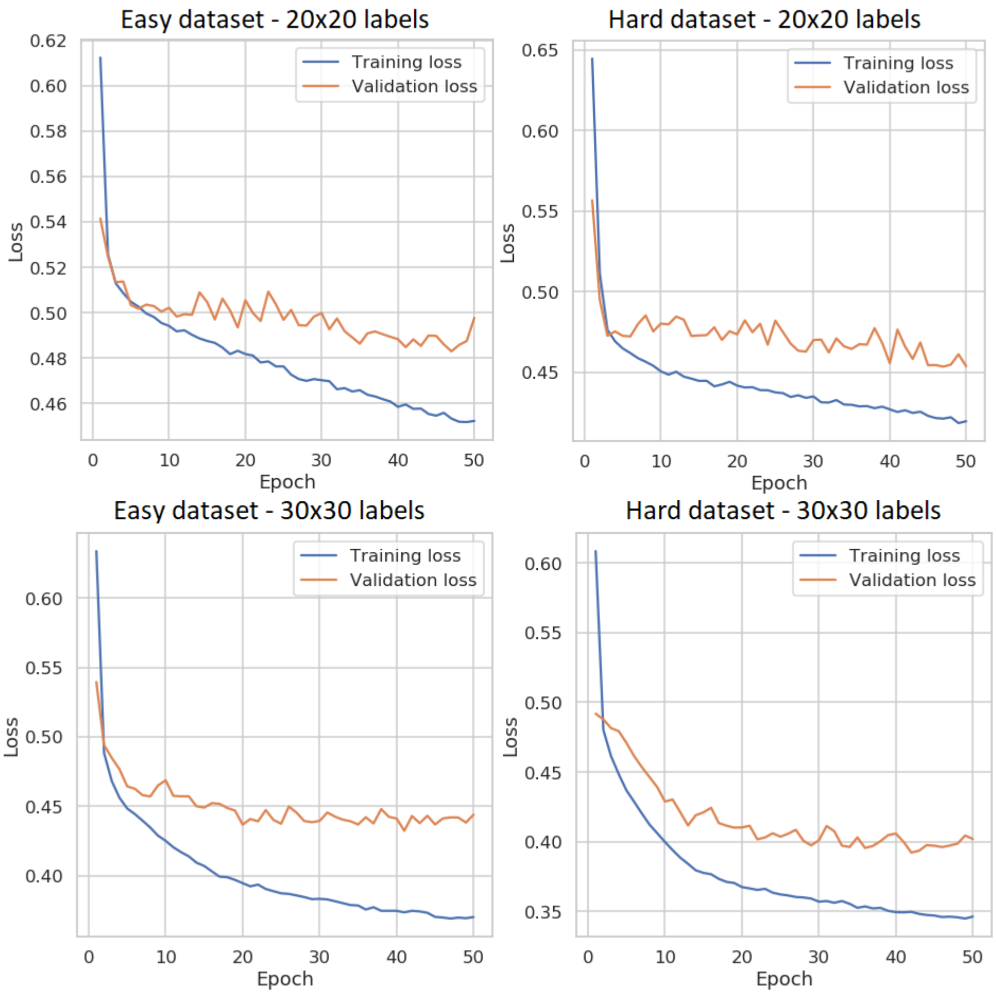


Figure 23. Loss during training using the simple convolutional neural network classification.

4.3.3. Capsule Network Classification (CapsNet)

The model training of the Capsule Network is identical to the simple Convolutional Neural Network. The loss is calculated for the training and validation datasets during the training of the network and the best model in the validation dataset is selected. Figure 24 shows these training statistics. Table 13 shows the selected models.

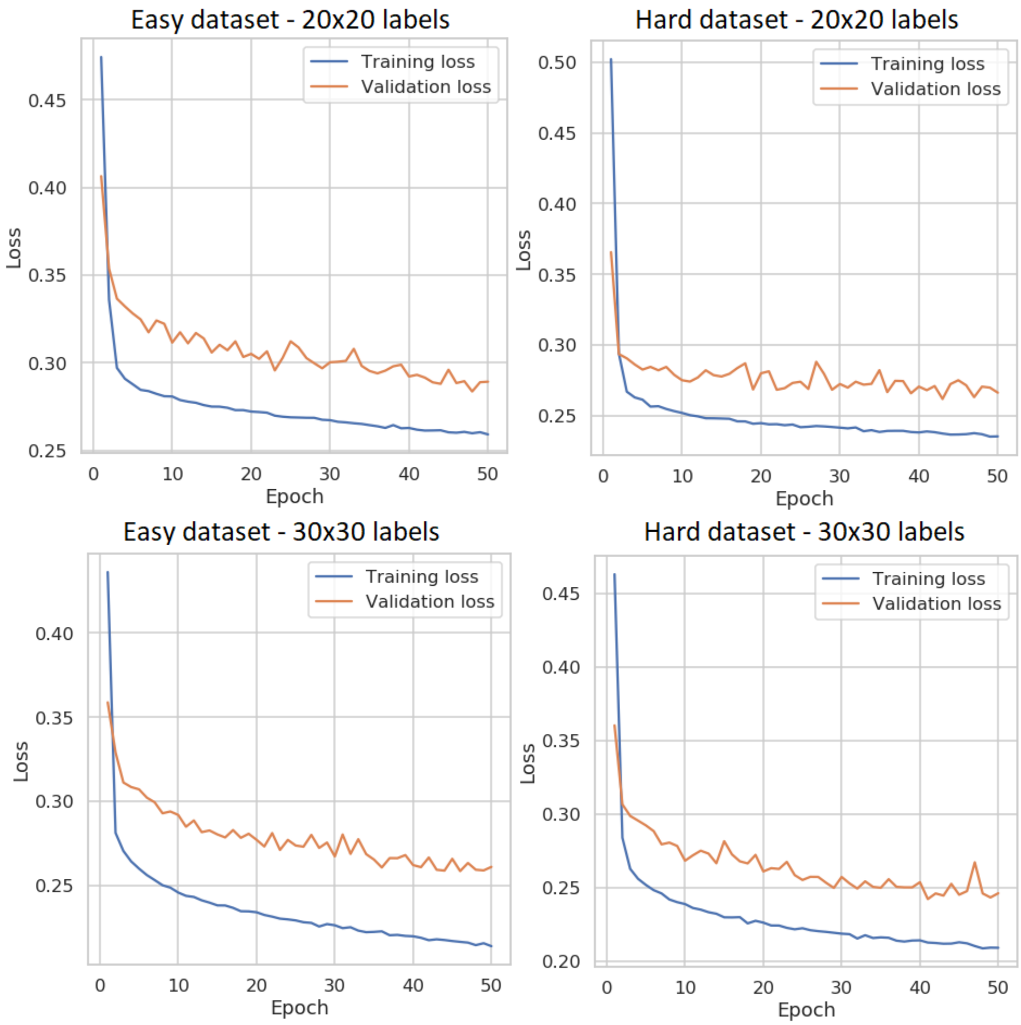


Figure 24. Loss during training using Capsules Network.

Input Size	Dataset Difficulty	Epoch of Best Model	Best Loss
20 x 20	Easy	48	0.2836
20 x 20	Hard	43	0.2618
30 x 30	Easy	46	0.2584
30 x 30	Hard	41	0.2421

Table 13. Selected models for the Capsule Network classification.

4.4. Capsule Network Reconstruction

As recommended in the Dynamic Routing Between Capsules paper [6], adding a reconstruction loss to the original loss function can improve the accuracy of the network. The reconstruction network in Figure 15 was implemented using the output capsule as inputs for the reconstruction network.

The loss of the whole network is the original loss of the capsule network plus a reconstruction loss, which is the mean squared error between the reconstruction and the original image. Using this new network, a new training was carried out using the easy dataset with 30 by 30 labels. Figure 25 (left) shows the training statistic of this new network. The selected best epoch corresponds to model 50. Figure 25 (right) shows the F1 scores of the training and validation datasets at different minimum score threshold.

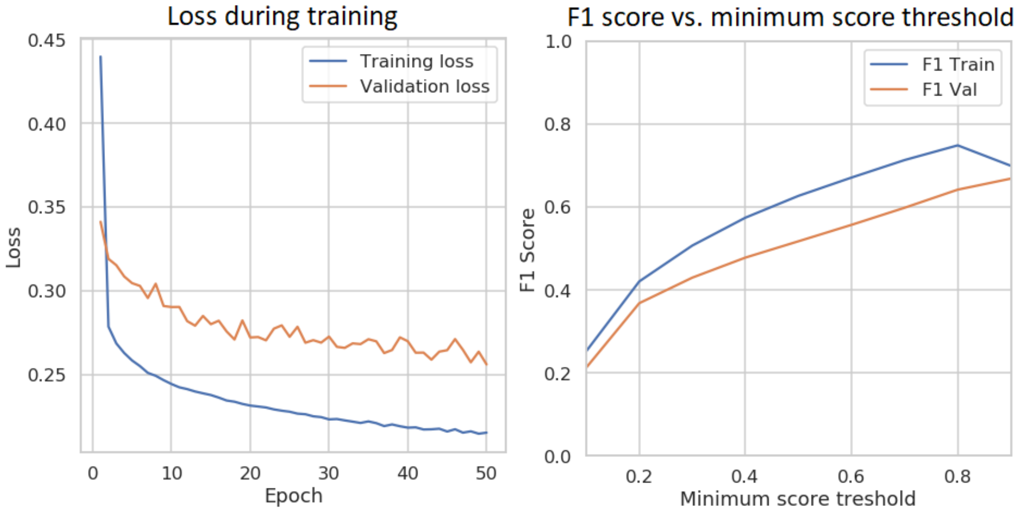


Figure 25. Training statistics (left) and F1 scores (right) of a model trained using an additional reconstruction network.

This new network has not produced a significant improvement in comparison to the capsule network without reconstruction. For this reason, its results were not included in the results section. However, the reconstruction images can give an insight of failures in some Capsule Network predictions. Figure 26 shows some reconstructions of region proposals from the test dataset whose label is bird. In this figure is noticeable that the capsule network failed to represent a shape of a bird. Some reasons for this failure are the high variance found in the shape of the birds, the blurriness of the images and some wrong labels. Figure 27 shows some samples of the bird images used for training for the 30 by 30 labels dataset that demonstrate these problems.

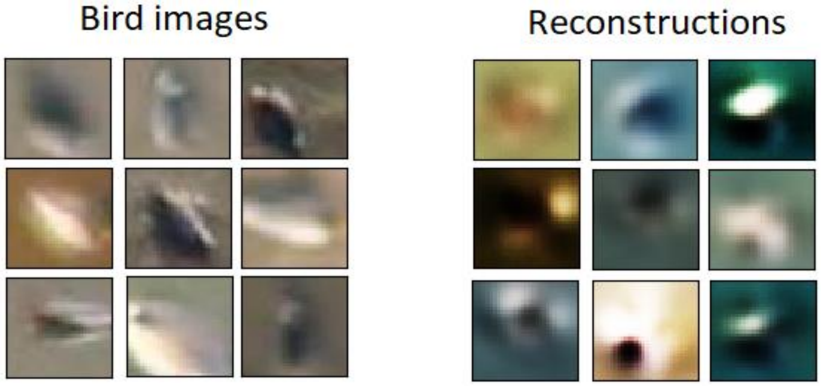


Figure 26. Bird reconstructions.

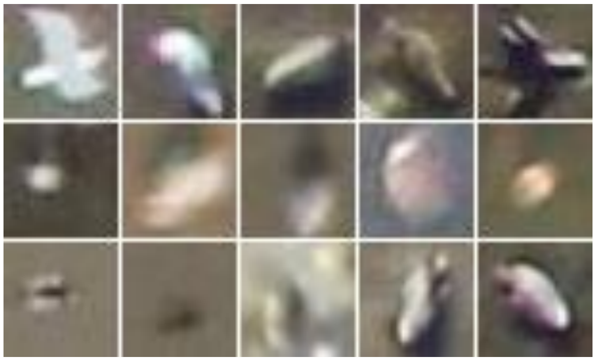


Figure 27. Bird images used for training for the 30 by 30 labels dataset.

4.5. Results in the Validation Dataset

In this section the three classification methods will be compared, and their results discussed using the validation dataset. For each of the four datasets and for each of the classification method, a Precision, Recall, and F1 score will be calculated. In each case, the minimum score threshold that maximize the F1 score in the validation dataset will be selected.

4.5.1. Easy Dataset

Figure 28 shows the F1 scores of the easy training and validation datasets using different minimum score thresholds. Table 14 shows the metrics scores for all the cases in the validation dataset.

The graphs that use a 30 by 30 labels shows a more similar distribution between the training and validation datasets than the ones that uses 20 by 20 labels datasets. Especially for the Fast RCNN classifier. This could mean that some information of the birds was missing when using 20 by 20 labels. Also, the maximum F1 score tends to be slightly higher when using 30 by 30 labels.

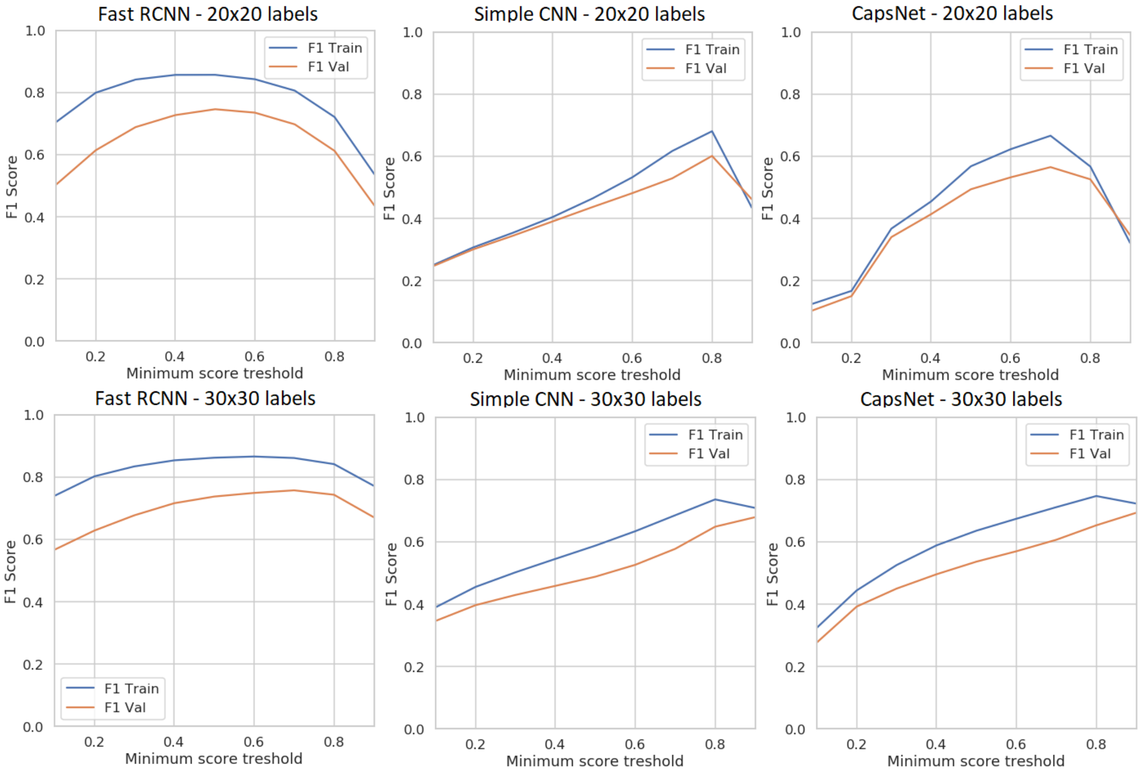


Figure 28. Classifiers F1 scores at different minimum score thresholds in the easy datasets.

The minimum score threshold is between 0.8 and 0.9 when using the simple convolutional neural network which means that this classification method is more uncertain than Fast RCNN classification. For the Capsule Network, all the F1 scores peaks when using a high value for the score threshold, this means that the Capsule Network tends to assign a high score to region proposals that are not birds. Thus, there is a high Recall when using Capsule Networks as a classifier, in comparison with the other classifiers.

Input Size	**Dataset difficulty**	**Classification method**	**Best threshold**	**Precision**	**Recall**	**F1**
20 x 20	Easy	Fast RCNN	0.5	0.7432	0.7494	**0.7463**
20 x 20	Easy	Simple CNN	0.8	0.4753	0.8181	0.6012
20 x 20	Easy	Capsule Network	0.7	0.4227	0.8526	0.5652
30 x 30	Easy	Fast RCNN	0.7	0.7799	0.7364	**0.7575**
30 x 30	Easy	Simple CNN	0.9	0.7207	0.6422	0.6792
30 x 30	Easy	Capsule Network	0.9	0.6885	0.6984	0.6934

Table 14. Classification metrics for the easy validation datasets.

4.5.2. Hard Dataset

In the hard dataset all the scores are lower than the easy scores due to the complex backgrounds. The main problem is that the classifiers tend the recognize background region proposals as birds (high Recall). The same conclusions as the easy dataset are reached when comparing the three classification methods: 30 by 30 labels obtained higher scores, the distributions of the datasets are similar and Fast RCNN is less uncertain about their predictions. Figure 29 and Table 15 shows the F1 scores visualizations and the information of the evaluations in the validation dataset.

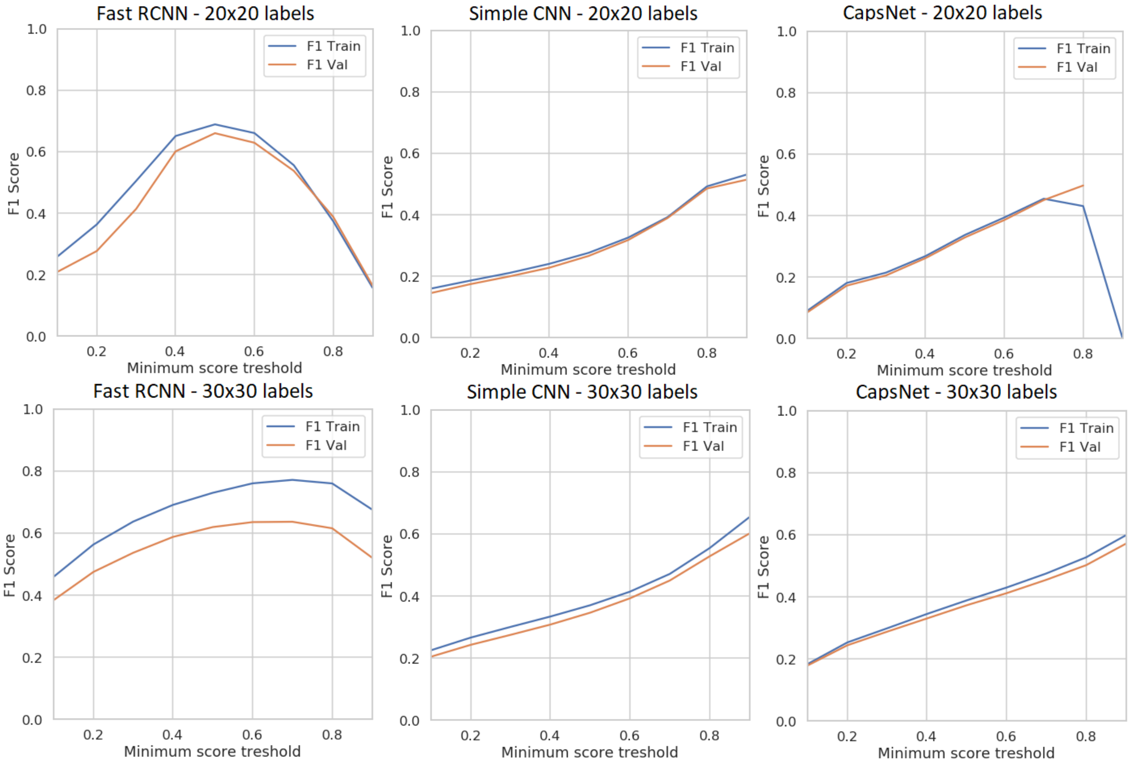


Figure 29. Classifiers F1 scores at different minimum score thresholds in the hard datasets.

Input Size	Dataset Difficulty	Classification Method	Best threshold	Precision	Recall	F1
20 x 20	Hard	Fast RCNN	0.5	0.6247	0.7001	**0.6603**
20 x 20	Hard	Simple CNN	0.9	0.5647	0.4718	0.5141
20 x 20	Hard	Capsule Network	0.8	0.6157	0.4180	0.4979
30 x 30	Hard	Fast RCNN	0.7	0.6461	0.6276	**0.6367**
30 x 30	Hard	Simple CNN	0.9	0.5594	0.6502	0.6014
30 x 30	Hard	Capsule Network	0.9	0.4590	0.7565	0.5714

Table 15. Classification metrics for the hard validation datasets.

5. RESULTS

This section shows the final inferences in the test datasets. All the results are analyzed and discussed. Also, some visualizations of the result are shown.

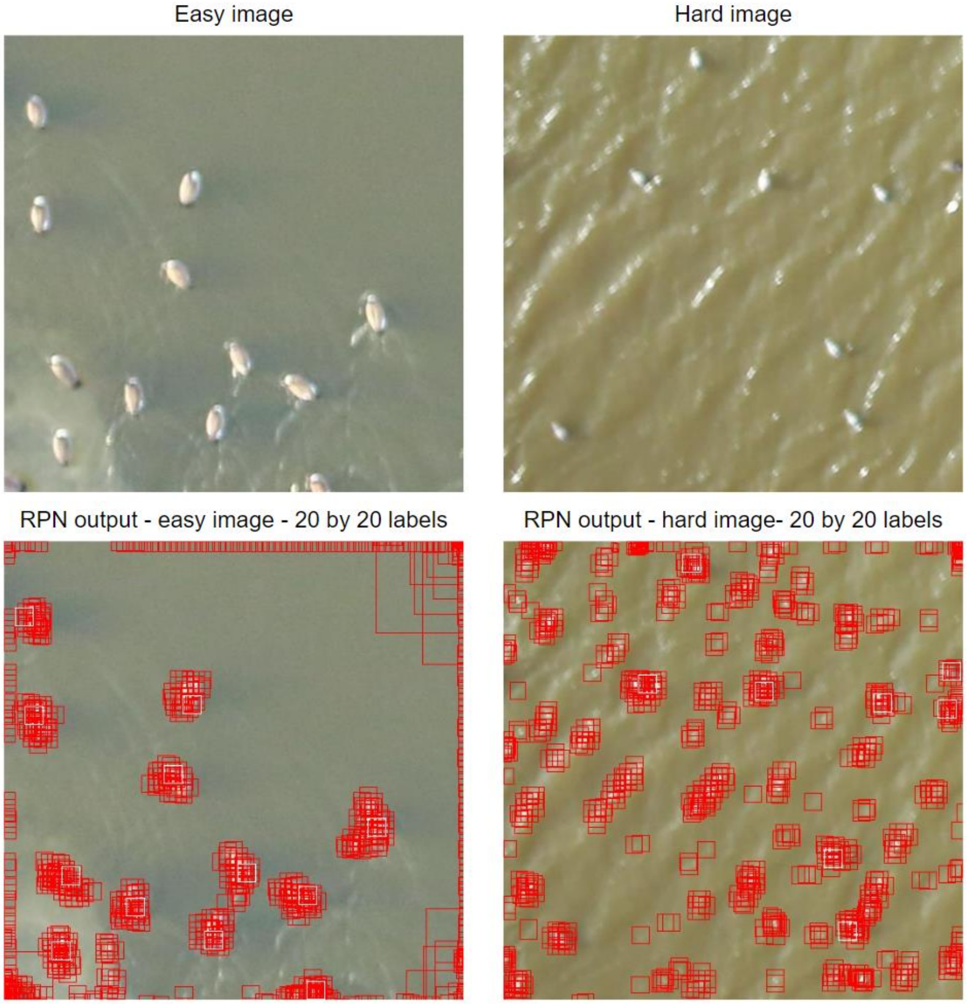
5.1. Region Proposal Network Results

After selecting the models with the best recall on the validation dataset and, as the experiments section suggested keep the top 800 region proposals per input image, the average recall for the test dataset was calculated and compared with the training and validation datasets. The average recall metric is used to compare how good the region proposal network is in generating new proposals on different datasets. The results are shown in Table 16. When using 30 by 30 labels the average recall is higher. This means that the region proposal covers more ground truths than the datasets that uses 20 by 20 labels. The recall at 0.5 threshold shows what percentage of the birds can be recognized using these proposals assuming that a region proposal is a true positive instance if its intersection over union with the ground truth is 0.5 or higher. Figure 30 shows some samples of the output proposals on the test dataset. The box proposals will be useful for training. However, most of the highly overlapped boxes will be removed with the non-maximum suppression operation after the classification.

Dataset Label Size	Dataset Difficulty	Dataset Type	Recall at 0.5 IoU threshold	Average recall
20 x 20	Easy	Training	0.994	0.619
20 x 20	Easy	Validation	0.939	0.533
20 x 20	Easy	Test	0.991	0.560

20 x 20	Hard	Training	0.997	0.612
20 x 20	Hard	Validation	0.938	0.542
20 x 20	Hard	Test	0.941	0.528
30 x 30	Easy	Training	0.994	0.678
30 x 30	Easy	Validation	0.978	0.650
30 x 30	Easy	Test	0.994	0.668
30 x 30	Hard	Training	0.987	0.653
30 x 30	Hard	Validation	0.974	0.629
30 x 30	Hard	Test	0.961	0.631

Table 16. Recall at 0.5 threshold and average recall of the region proposals.


Easy image
Hard image
RPN output - easy image - 20 by 20 labels
RPN output - hard image- 20 by 20 labels

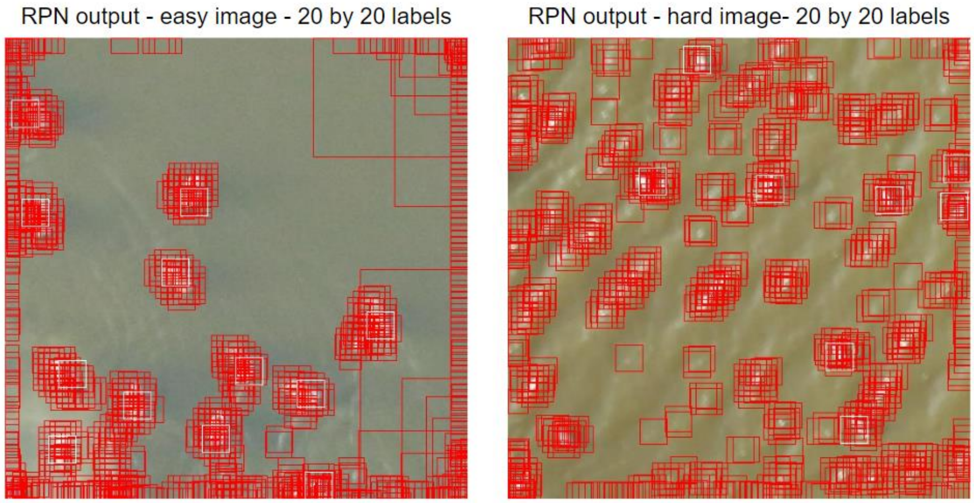

RPN output - easy image - 20 by 20 labels
RPN output - hard image- 20 by 20 labels

Figure 30. Samples of the output proposals on the test dataset. Red boxes are box proposals and white boxes are real ground truths.

5.2. Classification Results

All the results shown in this section used the same proposal boxes as inputs and were evaluated using the same evaluation script. In all cases the Fast RCNN classification achieved the best score in comparison to the other networks.

The test dataset ground truths are supposed to be undisclosed. However, in this section all the results in the test dataset, using different minimum score thresholds will be plotted to find how similar their distributions are in comparison with the training and validation datasets. This section shows all the F1 scores over different minimum score thresholds, from 0.1 to 0.9. The result reported in the tables correspond to the metrics in the test dataset using the best minimum score threshold in the validation dataset, which is not necessarily the maximum test score.

5.2.1. Easy Dataset

Figure 31 and Table 17 shows the F1 scores using different minimum score thresholds for all the easy datasets. When using 20 by 20 labels, the distribution of the test dataset is different than the distribution of the validation dataset. However, when using 30 by 30 labels the distributions seems similar. For this reason, the results when using 20 by 20 pixels region proposal tends to have low scores. Over all, Fast RCNN achieved the best score because, in comparison to the other classifiers.

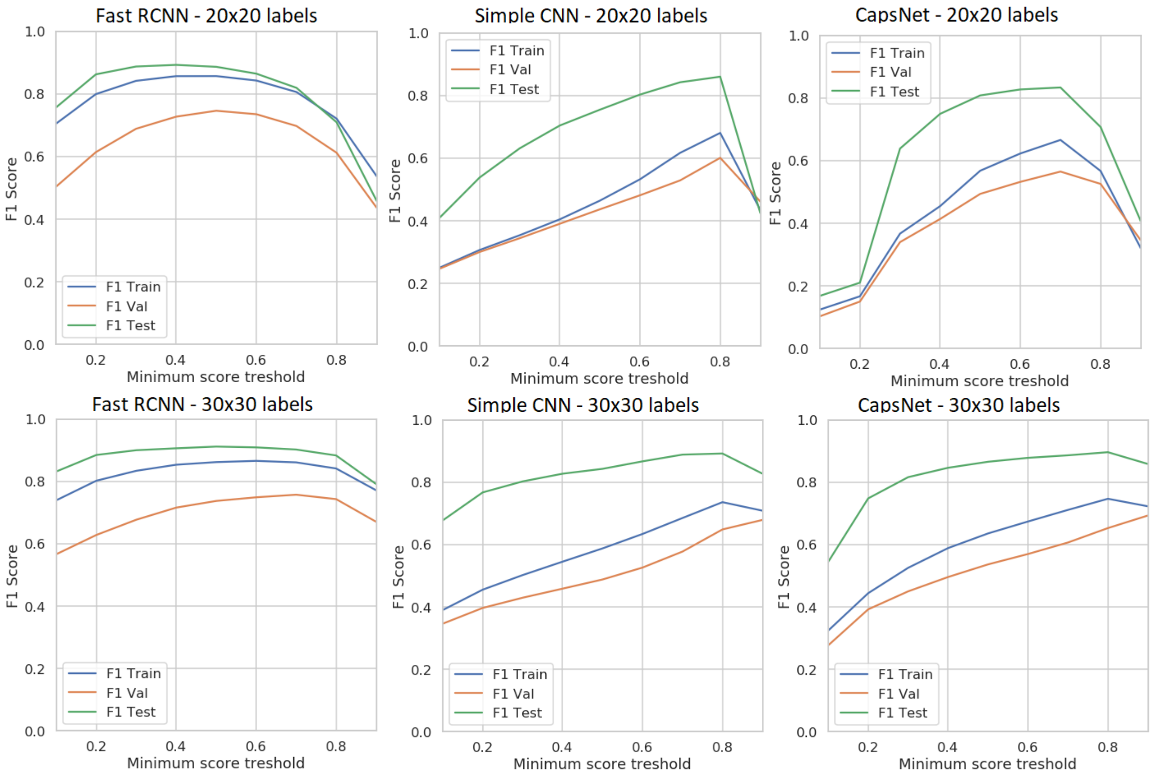


Figure 31. Classifiers F1 scores at different minimum score thresholds in the easy test datasets.

In the simple CNN and Capsule Networks all the scores peaks at a slightly higher threshold than Fast RCNN which means that there is more uncertainty and the classifier tends to recognize the backgrounds as a bird. Capsule Network results are slightly higher than the simple Convolutional Neural Network.

Input Size	Dataset Difficulty	Classification Method	Best threshold (validation)	Precision	Recall	F1
20 x 20	Easy	Fast RCNN	0.5	0.8831	0.8902	**0.8866**
20 x 20	Easy	Simple CNN	0.8	0.8492	0.8717	0.8603
20 x 20	Easy	Capsule Network	0.7	0.7970	0.8742	0.8338
30 x 30	Easy	Fast RCNN	0.7	0.9249	0.8809	**0.9024**
30 x 30	Easy	Simple CNN	0.9	0.9497	0.7335	0.8277
30 x 30	Easy	Capsule Network	0.9	0.9315	0.7964	0.8587

Table 17. Classification results for the easy test datasets.

5.2.2. Hard Dataset

Figure 32 and Table 18 shows the plot of the F1 scores and the final metrics obtained for the hard test datasets. The problems with the easy dataset seem to intensify with the hard datasets. The distribution of the datasets that uses 20 by 20 labels are very different that the distributions from the easy dataset. When using 30 by 30 labels the distributions seems similar and the scores are better.

Despite reaching good results at the peaks of the F1 scores, Capsule Networks tends to always have a high recall, which means that is more uncertain than the simple CNN at recognizing backgrounds.

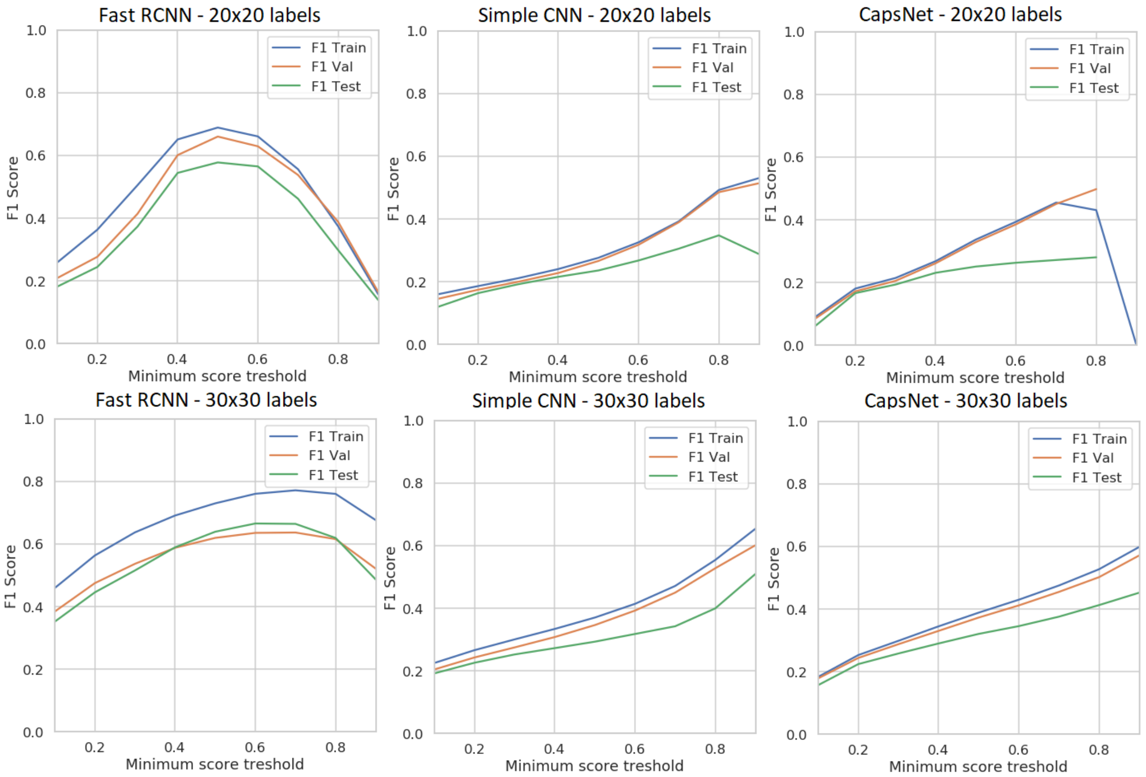


Figure 32. Classifiers F1 scores at different minimum score thresholds in the hard test datasets.

Input Size	Dataset Difficulty	Classification Method	Best threshold (validation)	Precision	Recall	F1
20 x 20	Hard	Fast RCNN	0.5	0.5317	0.6328	**0.5778**
20 x 20	Hard	Simple CNN	0.9	0.3063	0.2721	0.2882
20 x 20	Hard	Capsule Network	0.8	0.3156	0.2525	0.2805
30 x 30	Hard	Fast RCNN	0.7	0.6804	0.6492	**0.6644**
30 x 30	Hard	Simple CNN	0.9	0.4412	0.6033	0.5097
30 x 30	Hard	Capsule Network	0.9	0.3631	0.6000	0.4524

Table 18. Classification results for the hard test datasets.

5.3. Visualizations

Visualization for some easy and hard images are presented in this section. For all the visualizations, the green boxes represent the predictions made by the classifiers and the white boxes represent the ground truths.

5.3.1. Easy Dataset

Most of the easy dataset images are classified correctly. Figure 33 shows the visualizations of the classifications.

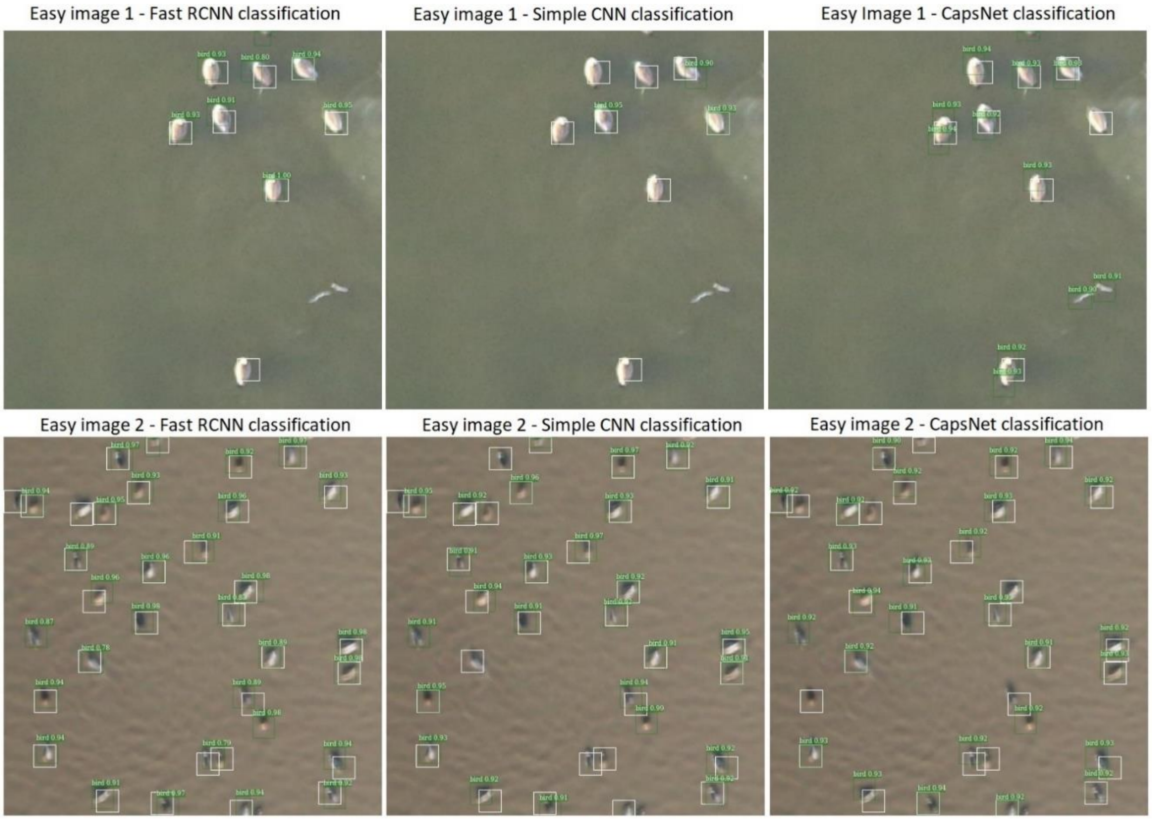
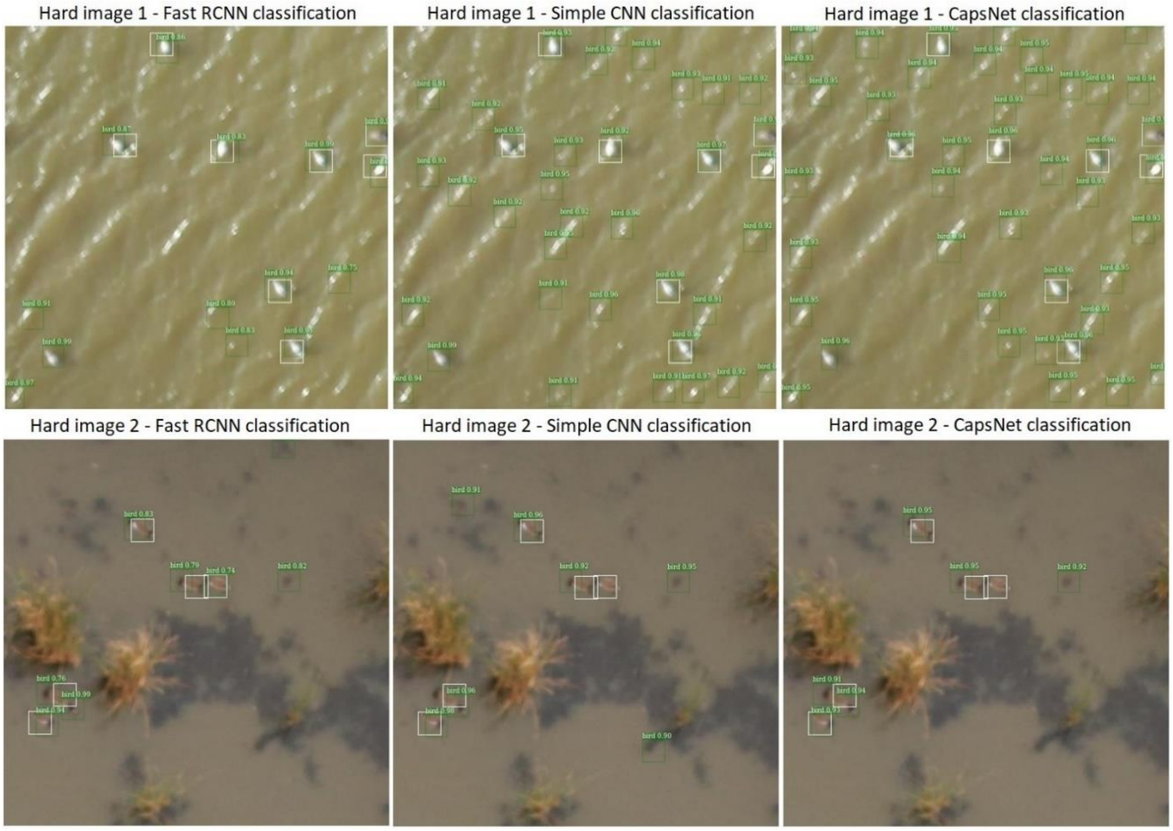

Easy image 1 - Fast RCNN classification
Easy image 1 - Simple CNN classification
Easy Image 1 - CapsNet classification
Easy image 2 - Fast RCNN classification
Easy image 2 - Simple CNN classification
Easy image 2 - CapsNet classification

Figure 33. Bird recognition visualizations in easy test images. Green boxes are the predictions and white boxes are the ground truths.

5.3.2. Hard Dataset

The hard dataset images tend to have high recalls and low precisions, which means that the networks tend to classify portions of the backgrounds as birds. This problem is present the simple convolutional neural network and, especially, in the capsule network outputs. Figure 34 shows visualizations of hard images.



Hard image 1 - Fast RCNN classification

Hard image 1 - Simple CNN classification

Hard image 1 - CapsNet classification

Hard image 2 - Fast RCNN classification

Hard image 2 - Simple CNN classification

Hard image 2 - CapsNet classification

Figure 34. Bird recognition visualizations in hard test images. Green boxes are the predictions and white boxes are the ground truths.

6. CONCLUSIONS AND FUTURE WORK

The conclusion section is an overview of all the ideas and explorations carried out in the design, implementation, experiment and result sections. The future work describes some options to improve the classification scores.

6.1. Conclusions

The focus of this research has been to explore region-based object detectors and to measure the performance of three different neural networks classifiers to recognize birds in aerial images. This has been a difficult task due to the low quality of the images, the different kind of shapes of the birds and the complex backgrounds that were misclassified as birds.

The localization and classification of the birds and the comparison of different classification neural networks has been achieved thanks to the implementation of a new pipeline that stores all the information of the region proposals in a file. This file can be used with an own implementation of a data loader and then with any classification method, independently of the framework or programing language it has been implemented.

Many tests have been carried out to find the best model for the region proposal network and the number of top proposal networks to keep. For the classification techniques the best model for all the classifiers, the optimum value for the non-maximum suppression within the outputs and a detailed exploration of how the minimum score threshold change over different values has been tested.

Despite the problems with the dataset, the classifiers achieved a relatively high score in the easy datasets. Fast RCNN classifier, Simple CNN and Capsule Networks achieved maximum scores of 0.9024, 0.8277 and 0.8587 respectively. The advanced techniques used in Fast RCNN like ResNets and Feature Pyramid Network obtained the best results in this dataset. The version of Fast RCNN tested in this research should be similar to Mask RCNN results, which uses the same techniques but reuse the region proposal network internally for the classification, making it faster to train. Mask RCNN is the result of years of development by the Facebook Research Team and is expected to obtain acceptable results in different datasets.

Datasets that used 30 by 30 pixels label produced better results on all classification methods, which could mean that there is more information of the birds when using these labels. On the easy dataset, the simple CNN and the Capsule Network achieved similar results, with a slightly better score for the Capsule Network. In the hard dataset, the simple CNN tends to have a better score than the Capsule Networks. This could be caused by the complex backgrounds that exist in the hard dataset. One cause for the low performance of the capsule network could be that it failed in representing a bird shape as a vector, given the high variance and blurriness of bird shapes. This has been demonstrated in the reconstruction experiment.

6.2. Future Work

A more accurate labeling is required for this dataset. It is not accurate to consider all the birds as the same size. Also, adding different labels for different kind of birds, or for birds in different positions (like flying) should improve the results and the neural networks classifiers can be compared independently of the problems with the datasets.

Capsule Network is a relatively new kind of network. Their first results with the MNIST dataset are promising. however, more research is still needed to make major improvements in the performance with other more difficult datasets. A recent new research proposes a “Matrix capsules with EM routing” [16] that, instead of using vectors to represent the instantiation parameters of an object, use matrices that allows a more complex representation of an object. A Matrix Capsules implementation could improve the results on LBAI dataset since the birds can adopt different kinds of shapes.

A new dataset that contains birds will be released by the Missouri Department of Conservation, with less blurriness and more accurate labels. The process and results described in this document can be used as a reference and baselines for creating new localization and classification models, and to know how to overcome some problems during their implementations.

REFERENCES

[1] Missouri Department of Conservation. About Us. https://mdc.mo.gov/about-us (Accessed 4/4/2019).

[2] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579.

[3] Liu, Y., Sun, P., Highsmith, M. R., Wergeles, N. M., Sartwell, J., Raedeke, A., ... & Shang, Y. (2018, June). Performance comparison of deep learning techniques for recognizing birds in aerial images. In 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC) (pp. 317-324). IEEE.

[4] LeCun et al. LeNet-5, convolutional neural networks. http://yann.lecun.com/exdb/lenet (Accessed 3/13/2019).

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[6] Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. In Advances in Neural Information Processing Systems (pp. 3856-3866).

[7] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[8] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2117-2125).

[9] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[10] Ross Girshick and Ilija Radosavovic and Georgia Gkioxari and Piotr Dollár and Kaiming He. Detectron. 2018. https://github.com/facebookresearch/detectron (Accessed 4/4/2019).

[11] Massa, Francisco and Girshick, Ross. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. 2018. https://github.com/facebookresearch/maskrcnn-benchmark (Accessed 4/4/2019).

[12] Facebook Open Source. Caffe2. A New Lightweight, Modular, and Scalable Deep Learning Framework. https://caffe2.ai/ (Accessed 4/4/2019).

[13] Pytorch. An open source deep learning platform that provides a seamless path from research prototyping to production deployment. https://pytorch.org/ (Accessed 4/4/2019).

[14] Higgsfield. Capsule-Network-Tutorial. https://github.com/higgsfield/Capsule-Network-Tutorial (Accessed 4/4/2019).

[15] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[16] Hinton, G. E., Sabour, S., & Frosst, N. (2018). Matrix capsules with EM routing.