

MARINE SEARCH AND RESCUE USING LIGHT WEIGHT
NEURAL NETWORKS

by

Salil Vishnu Kapur

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2019

© Copyright by Salil Vishnu Kapur, 2019

I dedicate this to my parents, teachers and friends, who have been a great support during my masters journey here at DAL. First of all I would like to thank Prof. Mae Seto and Prof. Stan Matwin who have been guiding me so nicely through this journey. I did surpass all the roadblocks that came down the lane, which couldn't have been possible If my professors didn't guide me in the right direction. And my father Mr. Sunil Kapur and mother Mrs. Neeru Kapur have been source of constant motivation, and their constant love and support has made me get up again and push harder than yesterday every time I did fall down.

Table of Contents

List of Tables	v
List of Figures	vi
Abstract	viii
List of Abbreviations Used	ix
Acknowledgements	x
Chapter 1 Introduction	1
1.1 Background and Motivation	1
1.2 Summary of Proposed Solution and Contributions	2
1.3 Thesis Outline	4
Chapter 2 Background and Related Work	6
2.1 Convolutional Neural Networks	6
2.2 Transfer Learning	8
2.3 Distillation of knowledge from neural networks	10
2.4 Confusion matrix	13
2.5 Efficacy measures	15
2.6 Efficiency measures	15
Chapter 3 Design and Methodology	18
3.1 Approach	18
3.1.1 Step 1 : Architecture	18

3.1.2	Step 2 : The distillation approach	19
3.1.3	Step 3 : Implementation description	20
3.2	Embedded system	20
3.3	Implementation architecture on the NVIDIA Jetson TX2	22
Chapter 4	Image Dataset Collection, Cropping and Labelling . .	23
4.1	Data Pipeline	23
4.2	Sources of Data Collection	23
4.3	Segregate Collected Videos into Two Categories	25
4.4	Cropping Objects and Labeling	26
Chapter 5	Experiments	28
Chapter 6	Results and Discussion	31
6.1	Efficacy Analysis	31
6.2	Efficiency Analysis	33
6.3	Contributions	36
Chapter 7	Conclusion	39
Bibliography	41
Appendix A	44
A.1	Different quality of data collection	44
A.2	Different distilled neural networks	47
A.3	Hardware Specifications	48

List of Tables

5.1	Teacher network architecture designed to use transfer learning .	29
5.2	Student network architecture designed.	30
A.1	Good quality video clips set from Youtube	45
A.2	Bad quality video clips set from Youtube	46
A.3	Good quality auxiliary relevant image set	46
A.4	Neural network model of size 379.7 MB	47
A.5	Neural network model of size 1.8 MB	47

List of Figures

1.1	Proposed architecture to be deployed on drone.	4
2.1	A basic single layer perceptron inspired from a neuron in brain	6
2.2	Transfer learning using pre-trained net weights.	9
2.3	Distillation architecture with explicit loss functions emphasized.	12
2.4	Confusion matrix definiton	14
3.1	The Nvidia Jetson Tx2 to integrate into the Hanseatic UAV. .	21
3.2	deployment on embedded system using c++, creating a new session and reading the saved session file to it, finally doing inference using the loaded model	22
4.1	data pipeline architecture	24
4.2	Distribution of imagery data sources on the internet.	25
4.3	Category ‘poor’ for training the teacher model: (a) and (b) are positive and negative examples, respectively.	26
4.4	Category ‘good’ for training the teacher model: (a) and (b) are negative and positive examples, respectively.	27
6.1	Comparing the three models on the basis of accuracy	32
6.2	Confusion matrix of the human-in-the-water classification with very few false positives and false negatives. This means that the efficacy is high.	33

6.3	Normalized confusion matrix with less relative percentage of FP and FN	34
6.4	Comparison of test accuracy with and without transfer learning	35
6.5	Different student models' computational requirements. Note, the 48.5 MB and 1.8 MB model have nearly same memory requirements.	36
6.6	Image processing rate for the three different size student network models (from L-R: 1.8 MB, 48 MB and 380 MB). The drop is gradual.	37
A.1	Hardware specifications of Nvidia Jetson TX2	48

Abstract

The research proposes an autonomous solution using deep learning techniques integrated on unmanned aerial vehicles (UAV) to effect a rapid and timely search in case of a man overboard (MOB) where a person has accidentally fallen off a ship. The UAV would be deployed from a ship. A light weight neural network model is to be integrated on a drone with limited processing resources. An image dataset was collected using open source videos based on people in the water (positive) and their negatives, object cropping was done by considering RGB spectrum. Three deep learning architectures have been developed considering the trade-off between the minimum memory utilization (efficiency) and maximum accuracy (efficacy). It is hypothesized that a small network can achieve results equivalent to a larger network when chosen on the efficacy/efficiency trade-off. On experimental evaluation, the smaller numbers of parameters yield the best *F1score* performance. Neither the very large 379.7 MB (94 million parameters) nor very small networks 1.8 MB (0.03 million parameters) gave the best performance; rather the mid-size network 48.5 MB (12 million parameters) achieved best results.

List of Abbreviations Used

MSAR	Marine Search And Rescue
SAR	Search and rescue
DL	Deep Learning
ML	Machine Learning
UAV	Unmanned Aerial Vehicle
MOB	Man overboard incident
DSP	Digital Signal Processors
SOTA	State of the Art
USV	Unamanned Surface Vehicle
MB	Mega Byte
SOTA	State of the art architectures
CNN	Convolutional Neural Networks
API	Application Programming Interface

Acknowledgements

I am extremely grateful and thankful to my supervisors, Prof. Stan Matwin and Prof. Mae Seto. They both have guided me through all the hurdles that came down the line in this Masters journey. Apart from learning the technical skills I have even mastered on qualities like discipline and planning which I think were my short comings and will definitely help me in future.

Also in this journey whenever I felt low it has been my parents who have held me up and made me believe in myself again. My father Mr. Sunil Kapur and mother Mrs. Neeru Kapur have been my inspiration for doing something great in the field of computer science, as they themselves have been in this field of computer science ever since it evolved on this planet.

I would like to share my gratitude to Prof. Evangelos Milios, Prof. Fernando Paulovich and Prof. Joseph Malloch for being my committee members. Also would like to thank my friends Kulwant, Sastry, Fateha, Yamini, Navkaran, Sid, Mohanish and Nikhil for always being there to cheer me up after coming back home from a very busy day.

Chapter 1

Introduction

A man overboard (MOB) incident refers to the situation where, due to some unanticipated event, a person falls from a ship or boat. These kinds of incidents have to be addressed on a priority basis since conducting a rescue in limited time is very critical. The reasons for the unfortunate incident can be varied; it can be due to a ship accident, excessive drinking etc. Upon notification of an MOB, search and rescue (SAR) operations must begin immediately but there are challenges to overcome for this.

To contribute to solutions for MOB incidents, this thesis proposes to use a vision system deployed on an unmanned aerial vehicle (UAV), or drone, to detect people in the water. The drone used is designed and operated by Hanseatic Aviation Solutions. The Nvidia Jetson TX2 is a candidate payload processing unit. This was selected because it is a typical advanced payload processor but it is still limited compared to state-of-the-art laptops or work stations. The vision system requirement is a lightweight neural network which is optimized in terms of both efficacy and efficiency.

There is another aspect of the drone solution proposed earlier, which is in regard to doing the processing in the cloud rather than doing it on the drone itself. This could solve the problem of limited hardware resources as the cloud has enormous computational capability. However, this solution depends on the internet bandwidth available. If there are bandwidth issues this could compromise a successful mission outcome.

1.1 Background and Motivation

The motivation to investigate neural network interpretability with respect to efficacy and efficiency was in the requirement to build an efficient vision-based detection system and to deploy it on an UAV with acceptable efficacy. This is applied to marine search and rescue (MSAR) operations in a man overboard (MOB) incident.

The approach was based on the concept of distillation [10]. It was first proposed by Geoffrey Hinton in the context of interpretability of deep neural networks. The concept is to use a model to train another model with different properties. For instance, using a "large" model to train a "lean", limited footprint model, deployable on an UAV.

In existing literature there are MSAR tools like rescue apparatus for sailboats[15], rescue systems [7], MOB safety systems [25] and alert and location systems [11] but they have the common issue of not being timely [12]. Sea-going vessels like cruise ships, oil tankers, etc. are not optimal for MSAR of a MOB incident. These ships can take 6-11+ minutes (depending on the ship length and speed) to execute the MOB turn, then there is the transit to the estimated point of the MOB and finally, the search begins. In currents, sea states and hypothermic temperatures a MOB has < 5-10 minutes before succumbing to hypothermia, exhaustion or exposure. Shore-based manned search and rescue vessels cannot be timely as it takes a while to mobilize from a greater distance. Manned MSAR vessels are not small, hence they cannot be deployed from all ships. With manned MSAR there is also the risk of the operators being overcome by harsh sea conditions.

1.2 Summary of Proposed Solution and Contributions

The proposed solution is based on using the distillation network approach. The motivation behind this thesis work is solely based on the reason that the distillation model was originally proposed for interpretability, but here it is used for efficiency instead.

The approach discussed here is using the full image classification, which is proposed to be used with the object cropping tool. An object cropped from the frames is to be given to the classification module for detecting whether the object is in the frame. The main goal for now is not localization, but in the future this proposed object cropping and classification modules together can be used for object localization.

Efficacy refers to the objective analysis of the network which is usually calculated in terms of *accuracy*, *precision*, *recall* and *F1-score*. Efficiency pertains to the computational power consumption (not studied here), memory utilization, execution speed

and throughput (images processed per unit time) of the neural network model. Understanding a model based on efficacy and efficiency measures is an interesting area, research in this area can yield new and useful insights. These insights can be used to build more efficient networks with roughly equal efficacy results.

In the distillation network approach [10] it was proposed that a teacher model be trained on a large collection of images and a (distilled) student model, which has the ‘knowledge’ of the teacher, without the memory requirements of a large model, be deployed on-board the UAV. This distilled model is an optimized solution to the efficacy versus efficiency trade-off. The student model module is implemented in C++ on-board the Nvidia Jetson TX2 which is a candidate microprocessor on-board the UAV. Figure 1.1 shows that initially images are fed into the distillation architecture. Firstly, the teacher network is trained using transfer learning and using this the student network is trained by using the distillation approach, detail of this is described in the chapters as follows. Then by experimenting with different parameters for this architecture the student models were trained and their session file was saved specific to each session. The final deployment was done on the Nvidia Jetson TX2 which is eventually to be deployed on the drone.

The final research goal is to contribute in the direction of deployable neural network for the embedded system which in our case is Nvidia Jetson TX2. In this area of research it is hard to get to an interesting point where the efficacy and efficiency trade-off is achieved because of the limited hardware constraints. The contribution is the development and implementation of a learned vision system on-board an unmanned aerial vehicle.

The architecture is explained in Figure 1.1 where the input data, which is set of collected images to be fed into the teacher network. That network is trained with transfer learning. This whole distillation module is designed on the basis of transfer learning being used to train the cumbersome teacher model and eventually, the distilled model learning from it. This module is explained in more detail in the next chapter. Eventually, the outputs from the teacher network are saved as soft targets which will be used to train the distilled model. In the next step, the student model learns by computing loss with the given labels and with the soft targets of the teacher model, which eventually is the minimization of two loss functions.

Then, session files are generated for distilled model designs to achieve the most optimal on the basis of efficacy and efficiency outcomes. A session file is a serialized collection of a model’s parameters. The most optimal session file is deployed on the payload processor on-board the UAV. The student model deployment module is coded in C++ for fast execution.

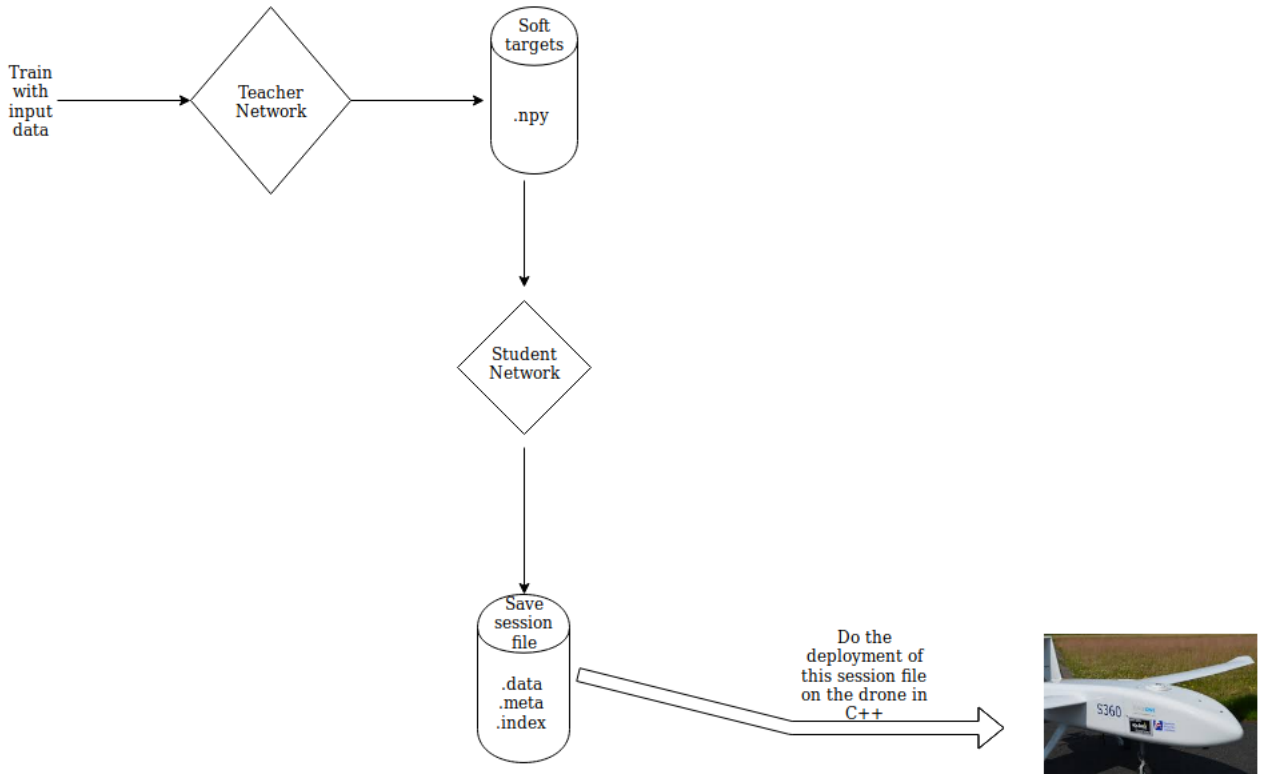


Figure 1.1: Proposed architecture to be deployed on drone.

1.3 Thesis Outline

The objective of this thesis is to gain insights into the concept of interpretability of neural networks with respect to the efficacy and efficiency of neural networks applied to visual searches for people in the water. The rest of this thesis is organized as follows: Chapter 2 describes the background and the related work; in Chapter 3, the design and methodology of the proposed solution is described; in Chapter 4, the data collection and data labelling for the training set is described; Chapter 5 details the experiments supporting the research; in Chapter 6, there is a discussion of the results

with some conclusions and finally, Chapter 7 discusses meaningful follow-on work.

Chapter 2

Background and Related Work

The literature review was performed in machine vision using deep learning techniques to develop a system which is optimal in terms of model efficacy and efficiency. A review of the existing literature demonstrates the novelty of the thesis contributions. To start, relevant background is presented. The following sub-sections describe the literature review that was done.

2.1 Convolutional Neural Networks

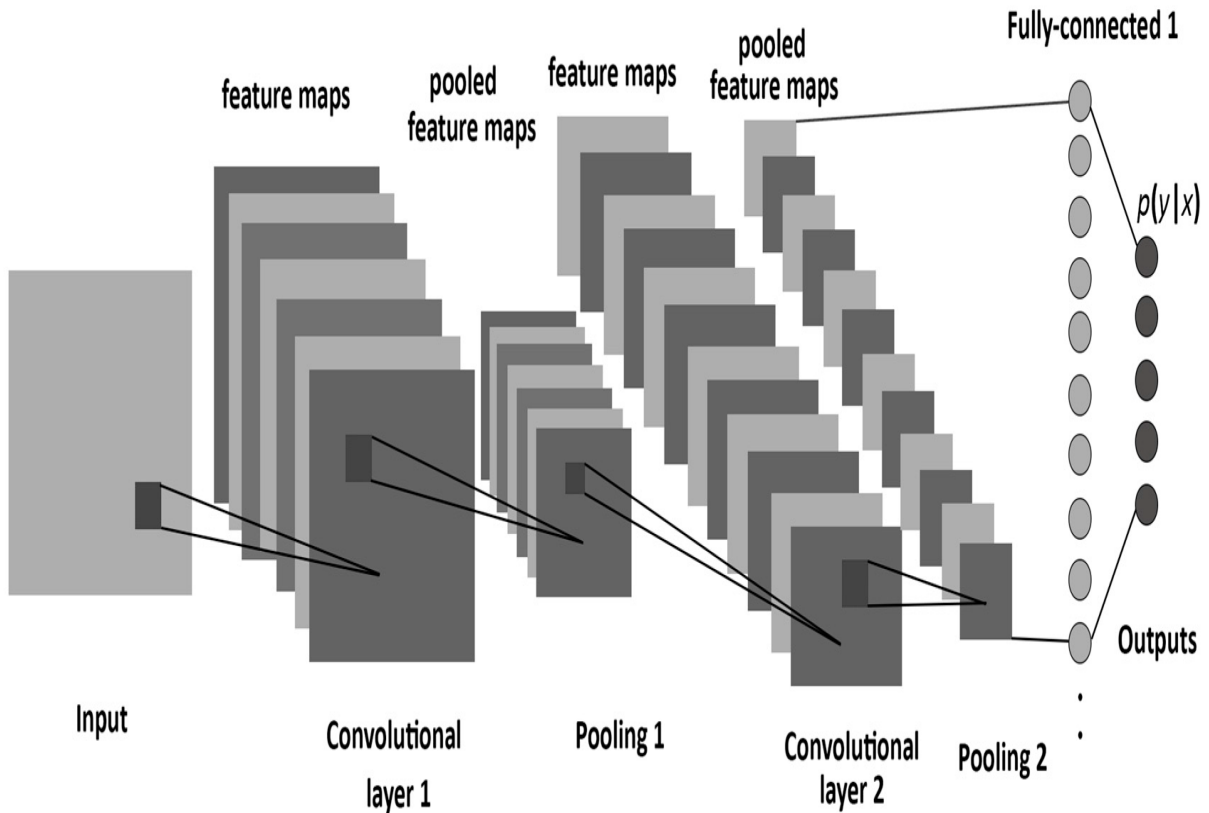


Figure 2.1: A basic single layer perceptron inspired from a neuron in brain

Convolutional Neural Networks (CNN) are state-of-the-art neural networks which

aim to find patterns in images by convolving the images, which is a multiplication of the image intensity with a filter matrix. CNNs are mainly used for multi or binary-class classification analysis. A detailed explanation of the convolutional neural networks is available here ¹.

Back propagation is used to train these neural network. The weights from the training are updated by adding derivatives calculated from the output towards the input direction. The back propagation rule is applied mathematically by taking the derivative using the multivariate chain rule.

State-of-the-art image classification with neural networks, based on convolutional architectures, have achieved notable results. A select few are reviewed here. However, they require non-trivial computational resources which is why these networks were not popular prior to the big data era [2]. Architectures like Inception V3 [23] and VGG 16 [20] have produced good results in terms of efficacy but they cannot run on light-weight embedded systems with ARM processors. Their main short-coming is the large number of parameters, and subsequently, the memory and processor speeds, that these networks require. If the objective is deploying on embedded systems, the recommendation is that these networks be pruned or distilled. With Inception V3, which has 94 million parameters and a 98 MB serialized session file for inference , it is not possible to deploy this on embedded processors like the Nvidia Jetson TX2 (which was designed for computer vision and other AI applications). Even if it could be deployed, the inference time would be large.

Behaviour-based computer vision solutions have been explored where a target is tracked based on its behavior (e.g. hold a constant heading) and not so much on its visual appearance. The literature survey of [18] shows the use of computer vision to detect humans in the water. The work is focused on object detection performed using background subtraction, optical flow and spatio-temporal filtering techniques and once detected, a moving object could be classified as a human being using shape, texture or motion-based features. With this approach, there is a high likelihood of not detecting the human (false negative) if there is no movement in the human or the case of a log that roughly resembles a human. However, in this thesis the video

¹<https://www.geekasservice.com/face-recognition-convolutional-neural-network-tensorflow/>

is captured at-altitude by a drone, so it is important to have an accurate (less false positives and negatives) human detection system.

In this research, CNNs are used in the vision system for the detection of a MOB. This algorithm is used with transfer learning and distillation which is described in detail in the next sections. With regards to the research, the interpretability concept is considered using both the efficacy and efficiency of the neural network models. Effort is put into experimenting with different network architectures to get an appropriately light weight network. Next, transfer learning and how it is used in the thesis is presented.

2.2 Transfer Learning

Transfer learning [26] is a technique in machine learning that stores knowledge gained while solving one problem and applying it to a different, but related, problem. With regards to transfer learning there are different paradigms to train the network. One way is to take the pre-trained weights as unchanging (frozen layer). The other method is to train the network and update the weights as the information back propagates in the network. For instance, the saved final weights of a neural network trained for image classification of birds in the morning could be used to initialize a neural network that is training for image classification of bird species at different times of the day (Figure 2.2, ²). In this way, the class A weights (birds in the morning) are used to train the class B weights (birds at different times of the day) where the first part of the network is frozen and the rest is updated with each iteration.

To detect humans, transfer learning with state-of-the-art image net classifiers were used. VGG 16 was used initially for this and then a relatively smaller network, like inception V3, was used and on top of it a few hidden layers were added to build the teacher network

Weights from training a network using transfer learning with the best efficacy results are saved. These weights are used to initialize another network for a similar class. As a result, the network tends to learn more quickly than with random weight initialization. Consequently, more incremental effort can be applied to determine the

²<https://medium.com/@subodh.malgonde/transfer-learning-using-tensorflow-52a4f6bcde3e>

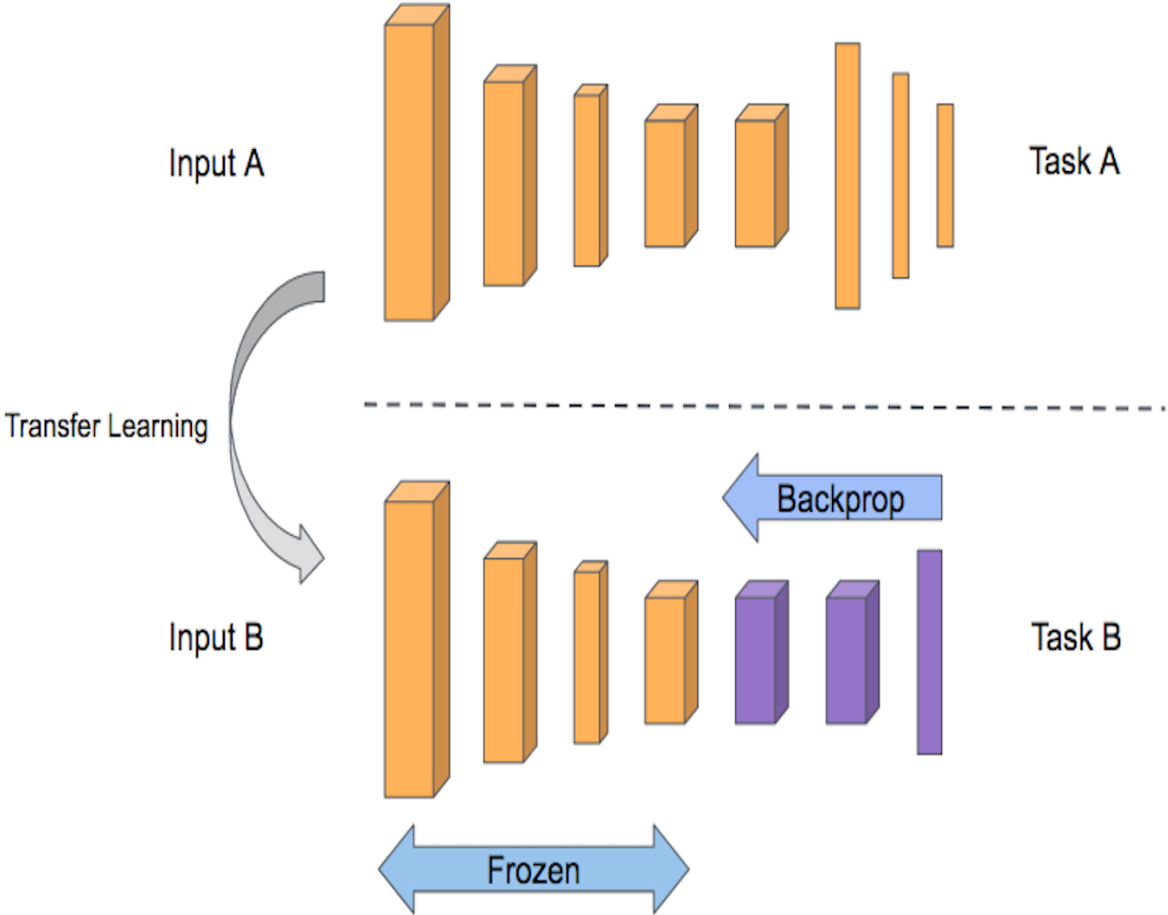


Figure 2.2: Transfer learning using pre-trained net weights.

weights for the new class. The next section discusses distillation of knowledge from neural networks and why it is relevant to this research.

Transfer learning has been significantly applied to different domains, for example, reinforcement [25] and unsupervised learning [2], for dimensionality reduction [18], etc. Their requirements are unrelated to the the motivation of this research which is limited on-board processing. There is no previously reported work on the optimization of networks for a vision-based system to be deployed on an embedded processor like the Nvidia Jetson TX2 – an objective of the thesis.

Transfer learning is used with the distillation approach where the unwieldy model is trained with a state-of-the-art neural network like VGG16 or Inception V3.

2.3 Distillation of knowledge from neural networks

The proposed solution uses the distillation network approach. The work is motivated by the sole reason that the distillation model was originally proposed for interpretability, but here it is used for efficiency instead. One solution is to use this distillation approach [10], where a smaller student network (distilled) is trained by the larger teacher network (unwieldy on an embedded processor). The construction of the teacher and student networks, and their objective functions, are described next.

Distillation is a regularizer technique which prevents the network from overfitting the solution by adding bias to the results. It is described in Figure 2.3, which shows two models trained using the same set of training images. One is the teacher model (cumbersome) and the other is the student model (light weight). This architecture can be executed in many different ways; here we will first of all train the teacher model with the training dataset, saving the predicted outputs. These will be used to train the student model. The convolutional part of the teacher model weights is initialized using the pretrained weights (transfer learning), and these are then fine tuned by training on the collected dataset (train). Next, using the same collected image dataset (train) the student model is trained by comparing the predictions with the true labels (loss 1) and also by comparing the predictions with saved predicted labels from the teacher model.

As shown in Figure 2.3, first, a trained teacher model, which gives soft labels as output, is fine-tuned with the help of a temperature hyper-parameter, T . Setting the appropriate value for the hyper-parameter is critical. The student network is trained by minimizing two loss functions. Soft targets are the probabilistic outputs of the neural network and these are computed by applying softmax to z_i , these computed targets are q_i (Equation 2.1). The value for hyperparameter T is 1 in this work. In Equation 2.1 the softmax function is applied after dividing the network output vector value by T , and then dividing the exponent of the value by summation of exponents over all the values of the vectors.

$$q_i = \frac{\exp(z_i/T)}{\sum \exp(z_j/T)}. \quad (2.1)$$

Soft targets from the teacher network output are used to train the student network by minimizing the two objective loss functions as in [10]

$$loss_1 = labels * \log(pred_{stud}) \quad (2.2)$$

$$loss_2 = pred_{teach} * \log(pred_{stud}) \quad (2.3)$$

$$loss = loss_1 + loss_2 \quad (2.4)$$

Therefore, the student network is trained by minimizing the loss function of Equation 2.4. The learning occurs in two steps. The first step is where the student learns without any other network support. This is achieved by minimizing a cross-entropy loss function of its predictions and the given labels as in Equation 2.2. In the second step, the distilled model (student) is trained using the unwieldy model (teacher) as mentioned in Section 2.3. Soft targets from the teacher model are computed by dividing the logits with the temperature hyperparameter and then taking the softmax of these (Equation 2.1). Next, the cross-entropy loss is calculated using the soft targets calculated in Equation 2.1 and the softmax output from the student network. The cross-entropy loss calculated for this in Equation 2.3 is minimized for optimization. Using the objective function in Equation 2.4 this combined loss is minimized to get effectively, a trained distilled network (student) with the fine-tuned parameters.

The teacher network is trained with the usual single objective loss function which is calculated by comparing the true labels with the output of the teacher (unwieldy) model after applying the softmax activation function. The combined objective loss function for training the student network, as stated in [10], is only possible when the correct labels are known for all or some of the transfer set, and then this method can be significantly improved by also computing the objective loss function (Equation 2.3) to train the distilled model to produce correct labels. Equation 2.2 is the cross-entropy with the soft targets and this cross-entropy is computed with the same high temperature in the softmax of the student (distilled) model as was used for generating the soft targets from the teacher (unwieldy) model. Equation 2.1 is the cross-entropy with the correct labels. This is computed using exactly the same logits in softmax of the distilled model but at a temperature of 1, both these objective functions are

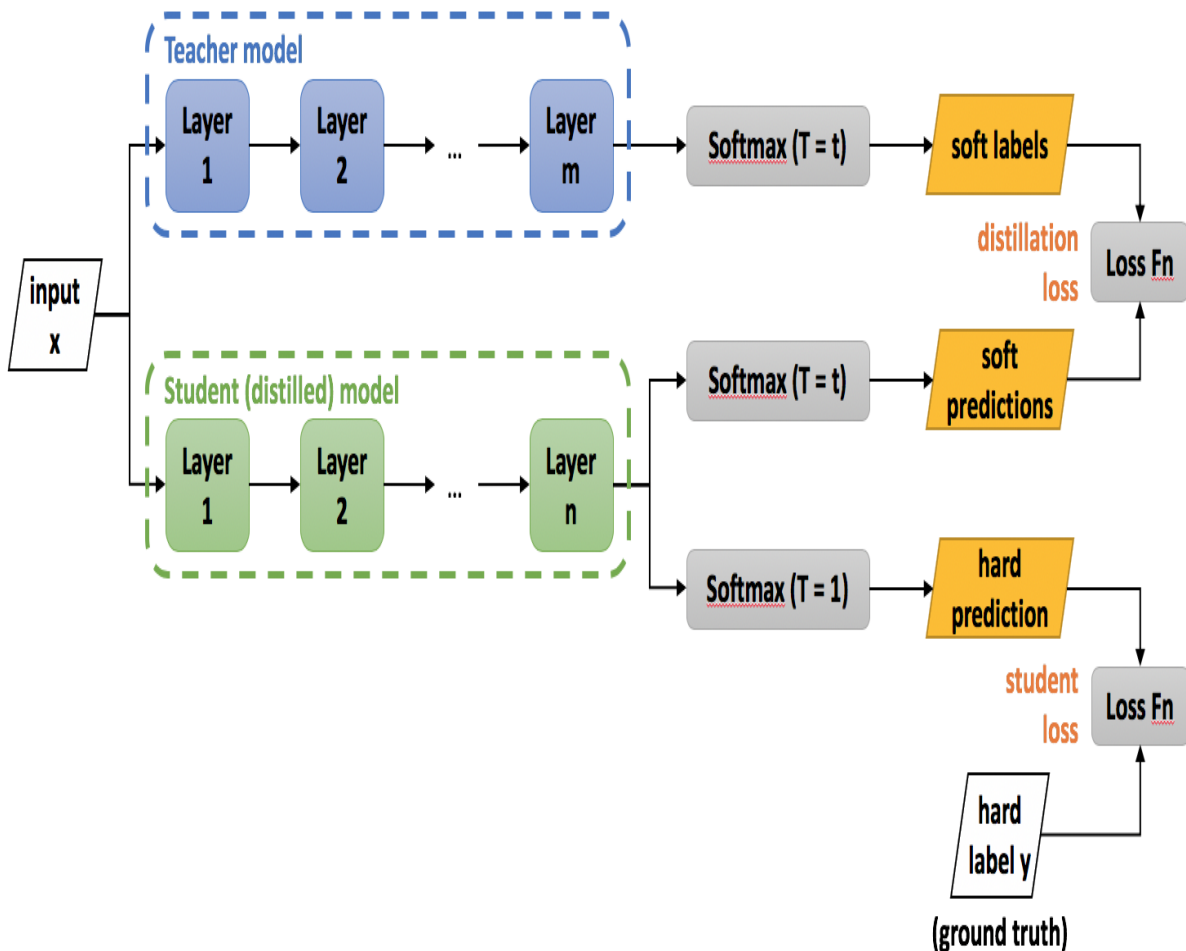


Figure 2.3: Distillation architecture with explicit loss functions emphasized.

based on the work of Hinton [10].

It was mentioned in [10] that the best results were generally obtained by using a considerably lower weight on the second objective function. Since the magnitudes of the gradients produced by the soft targets scale as $\frac{1}{T^2}$ where T is the hyper-parameter, it is important to multiply them by $\frac{1}{T^2}$ when using both hard and soft targets. This ensures that the relative contributions of the hard and soft targets remain roughly unchanged if the temperature used for distillation is changed while experimenting with meta-parameters.

Distillation and pruning techniques are widely used for creating networks that are computationally cheaper while keeping the efficacy unchanged. Work on network

distillation [10] ‘distills’ the knowledge from a large teacher network into a more compact student network. These authors study the concept of interpretability in the context of efficacy (accuracy, area under the curve and mean square error) only. Also, pruning techniques in the literature have the same concerns with efficacy. There is work in iterative growing and pruning for classification tree designs[9], feed forward networks [4], text [6] and near-optimal generalization [13] for the purposes of efficacy after reducing the network size. There is no consideration for efficiency either.

In the distillation [10] work there was significant work to train light-weight networks based on the efficacy results. Making predictions using a whole ensemble of models is cumbersome and may be too computationally expensive to deploy to a large number of users, especially if the individual models are large neural nets. Caruana and his collaborators [ref?] show it is possible to compress the knowledge in an ensemble into a single model which is much easier to deploy. This approach is further developed using different compression techniques. But this area of research does not align with distilling the knowledge from a single expensive neural network into a light-weight network based on the efficacy and efficiency trade-off.

Therefore, distillation is used in this research to build neural networks that are light-weight, out of the distilled networks that are trained. The networks are based on efficiency as well rather than just the concept of interpretability. The next sections details the efficacy measures used in this research. Subsequently, in the following three subsections, measures of analyzing the model for efficacy and efficiency of the neural networks is described. Subsections 2.1.4 and 2.1.5 pertain to efficacy and subsection 2.1.6 to efficiency.

2.4 Confusion matrix

The confusion matrix is a special case of a contingency table, with two dimensions: ”actual” (or ground truth) and ”predicted” (by the algorithm), across two classes. Each permutation of dimension and class is a variable in the contingency table. In machine learning (and other) classification approaches, a confusion or error matrix quantifies the performance of the classification. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa). The name stems from the fact that it clearly shows if the

classification algorithm is confusing two classes.

The confusion matrix capture the four performance measures for a predicted solution: true negatives (TN), false negatives (FN), false positives (FP) and true positives (TP). ‘True labels’ are those assigned as a dataset ground-truth for a specific classification. ‘Predicted labels’ are the outcomes from the algorithm being tested. In Figure 2.4 the matrix value true positive is where the predicted and true label cases match (predicted and true labels are positive). False positive is where the predicted and true label cases do not match (predicted label is negative and true label is positive). False negative is also a case where the predicted and true label cases do not match (predicted label is positive and true label is negative). True negative is the case where the predicted label matches the given one (predicted and true labels are negative).

True Label	Negative	TN	FP
	Positive	FN	TP
		Negative	Positive
		Predicted Label	

Figure 2.4: Confusion matrix definition

2.5 Efficacy measures

The four efficacy measures used here are the accuracy, precision, recall and F1-score. Accuracy refers to the correctly classified examples divided by the total number of examples available for classification as shown:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \quad (2.5)$$

Next, precision relates how precise the model is out of those predicted positives, and how many of them are actually positive:

$$precision = \frac{TP}{TP + FP}. \quad (2.6)$$

Recall calculates the proportion of true positives. It is the model performance metric used to select the best model when there is a high cost for false negatives.

$$recall = \frac{TP}{TP + FN}. \quad (2.7)$$

The *F1 – score* seeks a balance between *precision* and *recall*. But what is the difference between *F1 – score* and accuracy? Accuracy can be largely contributed by a large number of true negatives which in most circumstances does not focus on much, whereas false negative and false positive have associated costs (tangible & intangible) thus *F1 – score* is a better measure because of the balance it achieves between precision and recall and, for the dataset used here, there is an uneven class distribution (larger number of true negatives).

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall} \quad (2.8)$$

2.6 Efficiency measures

Efficiency is an important concerns of this research given the limited computation resources. Experimentation explores the memory required vs the model processing times and the number of images processed per unit time by the neural network architectures as additional performance metrics. These were the measures of efficiency.

Research and development using the Tensorflow API version 1.9.1 (macros coded in Python).

Remote server processing architectures are widely used for inference with deep learning methods. For example, real-time video data surveillance performed in the Cloud with on-board light-weight embedded processors like the Nvidia Jetson TX2. Remote server processing [22] of imagery, when the network connection is good, between the UAV and remote server. Once the processing is complete, the inference is transmitted back to the UAV.

As well, the use of remote server processing is, by necessity, less demanding given embedded processors. Usually, the processing is developed on desktop processors where there is ample computation resources to create the light-weight neural networks, Then, the light-weight neural networks are tested on the embedded processor.

Prototype solutions using neural networks for image classification have been reported with hefty Intel graphical processing units (GPU) for the same problem of detecting humans on the water [8]. They tried convolutional neural networks for the full image, sliding window and precise localization. But, the deployment onto an UAV has not been tested. This was a future work item for them. It is understood that performing inference on powerful Intel desktop processors is obviously more efficient than on a limited embedded ARM processor like the Nvidia Jetson TX2.

Consequently, the expanded interpretability which considers both efficacy and efficiency in this thesis is novel.

There is no real literature on methods that are currently used for detecting human on the water, which can be deployed on the UAVs with limited computational resource. Significant work is happening to develop solutions where the processing is performed remotely in the Cloud rather than on-board the UAV. So, the light-weight neural network approach is quite novel and is a unique solution. Beyond using the state-of-the-art architectures, there is additional focus to build a light-weight network which is both efficacious and efficient. Convolutional neural networks will be used for this vision-based problem.

The next section discusses a methodology to explore how neural network-based methods might be useful, or have a role, in maritime search and rescue.

Chapter 3

Design and Methodology

An objective of this research is insight into the trade-off between the efficacy and efficiency of neural networks. Once that is understood, the next objective is to design a network for the task of binary image classification. This requires a computationally efficient neural network architecture which can be deployed on an embedded system like an UAV. Neural networks are being used because the initial layers learn the micro features whereas the next layers learn the macro features such as the edges in an image or the different objects because of differences in feature vectors.

To achieve this, as justified in the previous chapter, the focus was on small convolutional neural networks for image classification. ‘Small’ makes it deployable on an embedded processor like the Nvidia Jetson TX2. The distillation work of [10] was explored towards this. They used a teacher and student network approach. Building on [10], the distillation concept of interpretability is extended to also include network efficiency, as defined earlier.

Next, the approach developed in this thesis for the research problem is described. First, is the use of transfer learning to train the teacher network which is explained in the transfer learning sub-section. Then, after the teacher network is trained, the student network is trained which is described in the distillation sub-section. Finally, a serialized session file of the student network is built which can be deployed, coded as a C++ module, on the Nvidia Jetson TX2 embedded processor.

3.1 Approach

3.1.1 Step 1 : Architecture

Different deep learning networks were considered for training a neural network which can be used for transferring the ‘knowledge’ to a light-weight neural network. The state-of-the-art networks, VGG16 and InceptionV3, were explored to initialize the

network weights using transfer learning. Consideration of the networks for transfer learning was based on analyzing networks of different sizes, with accepted state-of-the-art accuracy, on benchmarked datasets like Imagenet. The weights of VGG16 and InceptionV3 were downloaded from the internet, with memory sizes of 553.4 MB and 108.8 MB, respectively.

The teacher (unwieldy) model is trained using transfer learning. First, VGG16, which is significant in size compared to InceptionV3, was attempted. With both networks, comparable efficacy results were achieved so the significantly smaller network, InceptionV3, was preferred. Experiments were designed (and evaluated) based on their cross-entropy and objective loss functions. Then, the transfer learning trained teacher network was ‘distilled’ to create the light-weight student network.

Transfer learning, combined with the distillation architecture (with expanded interpretability), is a novel approach. The transfer learning module was coded in the interpretive scripting language, Python (version 3.6.5). Several mathematical libraries were used. The open source library tensorflow 1.9.1 was used. The initial development was performed on an Intel I7 8th Generation laptop with the Nvidia 1060 GEFORCE GTX GPU. The following section describes the distillation step developed with these resources.

3.1.2 Step 2 : The distillation approach

One of the objectives is to build a light-weight neural network, so the distillation approach [10] was pursued. Distillation was explored for its efficiency based on objective measures of the trade-off between efficacy and efficiency of the models. The unwieldy teacher model was trained with transfer learning which was subsequently used to train the light-weight distilled student model. The student model was built to achieve the best efficiency for a given efficacy. The transfer learning and distillation architecture modules were implemented in Python. Their implementation is described next.

3.1.3 Step 3 : Implementation description

The implementation was in the form of four developed modules: *manager*, *teacher*, *student*, *soft_targets*, *hard_targets*, *test*, *util*, *tensorboard_logs*, *test* and *clustering*. The *manager* module administers reading in the dataset and calling the models for training. The *teacher* module trains the teacher model using transfer learning via InceptionV3. The *student* module contains the complete model architecture with the ability to modify based on the experimentation. *soft_targets* and *hard_targets* save the probability outputs of the teacher model and final outputs after applying *argmax* from the student model, respectively. *test*, *util* and *tensorboard_logs* are self-explanatory. The *clustering* module contains the functionality to initially train a *k – means* model on the images dataset. The clusters are based on the different environmental conditions, which in this research consisted of morning, afternoon and night. But, this clustering module is not used, as in this module for every single cluster, that cluster specific images are supposed to be used and in some cases there might not be enough images relevant to a particular cluster. For instance, using *k – means* with morning, afternoon and night clusters there needs to be enough images for training the classifier corresponding to each cluster. Which is not a problem when training a single convolutional light weight neural network with the complete set of image dataset.

The complete transfer learning based distillation architecture is implemented in Python. When executed, it generates a session (*.sb*) file to deploy on the server. The generated session file is a serialized binary session file which is language independent and whose size depends on the number of parameters used. To implement the session file on the UAV a deployment module, written in C++, reads the session file and performs the inference. This C++ module is compiled and run on the Nvidia Jetson TX2 on-board the UAV.

3.2 Embedded system

An embedded system has logic that is administered by an on-board processor. Common examples include microwaves ovens and modern automobiles – hardware systems that execute software as part of their normal operation. Modern robots are examples

of embedded systems i.e. actuators and sensors that are integrated with a micro-processor for either reactive or deliberative control. Depending on how the hardware selection is done, if used for video games, rendering video, or machine learning, one will need something powerful. This is why the Nvidia Jetson TX1 and TX2 exist. Is it as fast as a desktop loaded up with an i7 and a GTX 1080? No, but that's not the point: a desktop built around an i7 6700K and a GTX 1080 will draw at least 300 Watts, whereas the Jetson TX2 only draws fifteen at full bore.

The Nvidia Jetson TX2 (Figure 1.1) is used here as it is the fastest, most power-efficient embedded artificial intelligence (AI) computing device. Successful work on satellite identification imaging [24], learning locomotion for quadruped robots [3] and the well known AI Challenge [17] motivated the decision to use this processor. It is a 7.5 Watt supercomputer which makes it possible to use AI on-board robots. It is built around an NVIDIA Pascal-family GPU and loaded with 8GB of memory and 59.7GB/s of memory bandwidth. It works with a variety of standard hardware interfaces that make it easy to integrate with many types of systems. The detailed specifications of this processor are shown in Appendix A.3. All the necessary drivers were installed onto the Jetson TX2 using jetpack 3.3.



Figure 3.1: The Nvidia Jetson Tx2 to integrate into the Hanseatic UAV.

Then, the TensorFlow [1] open source data flow programming tool was installed

from github, using the bazel build tool [19] on the Jetson TX2. TensorFlow is used for developing machine learning applications like convolutional neural networks. The next section describes the proposed architecture designed for human classification.

3.3 Implementation architecture on the NVIDIA Jetson TX2

The Jetson TX2 was installed with the necessary binaries for the deployment. Jetpack 4.3 was installed initially but this was downgraded to 3.3 because it was incompatible with the cuda, cuDNN version 7.1.5, required for TensorFlow 1.9.1.

After setting up the binaries on the Nvidia Jetson TX2, a deployment module in C++ is used for inference with the distilled model session file. As shown in the Figure 3.2, the distilled model session file is loaded and the computational graph is added to the present session. Then, the images are loaded to run the inference. Next, the output tensors from this model are used to calculate the predictions, which are compared with the given labels and finally, the accuracy of the inference is noted. C++ was used for the deployment, instead of Python, for performance. Python is an interpretive language and C++ is an object-oriented language that can be compiled. In the next chapter the image collection, cropping and labelling of the dataset for the training and inference of the models is discussed.

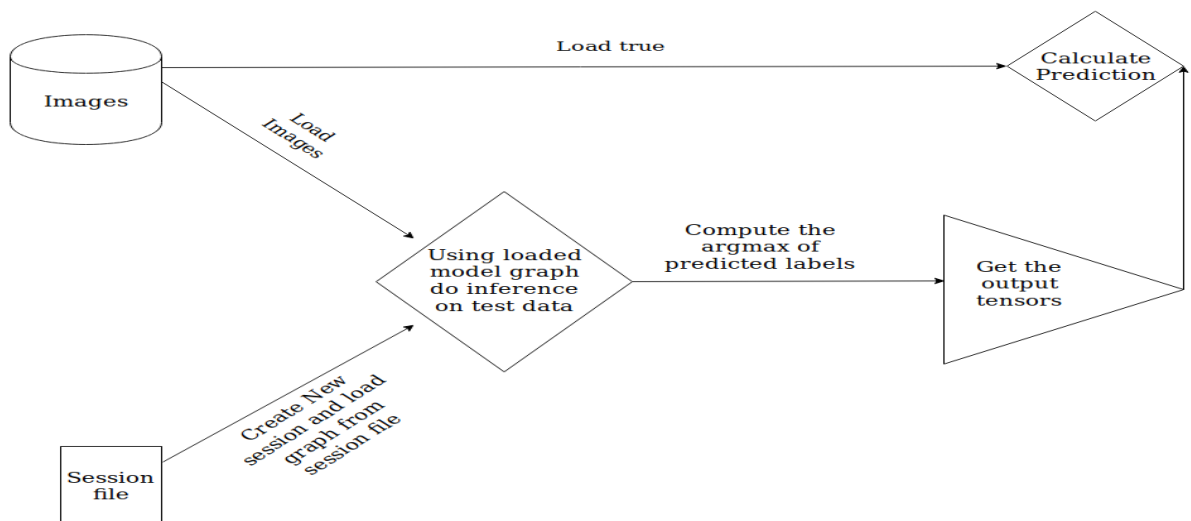


Figure 3.2: deployment on embedded system using c++, creating a new session and reading the saved session file to it, finally doing inference using the loaded model

Chapter 4

Image Dataset Collection, Cropping and Labelling

In this chapter, the dataset collection is described as well as its labelling. Significant effort was expended to find images from the internet and other sources related to humans in the water. Negative examples were also collected as there were no applicable dataset repositories to train the models. After exploring different data collection methods and ideas like vehicle dataset [21], the well known imagenet dataset [5], road traffic dataset [14] and lung image dataset [16], the conclusion was that there were no adequate datasets to draw from and that the internet was a good place to start. This is described next.

4.1 Data Pipeline

After searching the existing literature extensively, it was difficult to find an appropriate repository/dataset related to humans in/on the water. In the event of a MOB incident, the focus is on the rescue operation rather than capturing the MOB on video. Therefore, it became necessary to collect the data from the internet and other sources of related people activities in the water. Figure 4.1 shows the data pipeline architecture for the man overboard vision problem. Keywords relevant to the problem statement were devised to search on the internet. Then, the results of this online search were saved as video links in a text file.

4.2 Sources of Data Collection

Collection of on-line videos was done by designing a variety of keyword queries which could yield the most appropriate results from the internet. Different combinations of the following keywords were used: "rescue", "swimming pool", "water", "dog swimming", "ocean rescue", "sea rescue", "incredible rescue", "sailboat rescue", "human rescue", "puppy rescue", "ocean", "boat rescue", "helicopter", "plastic in ocean",

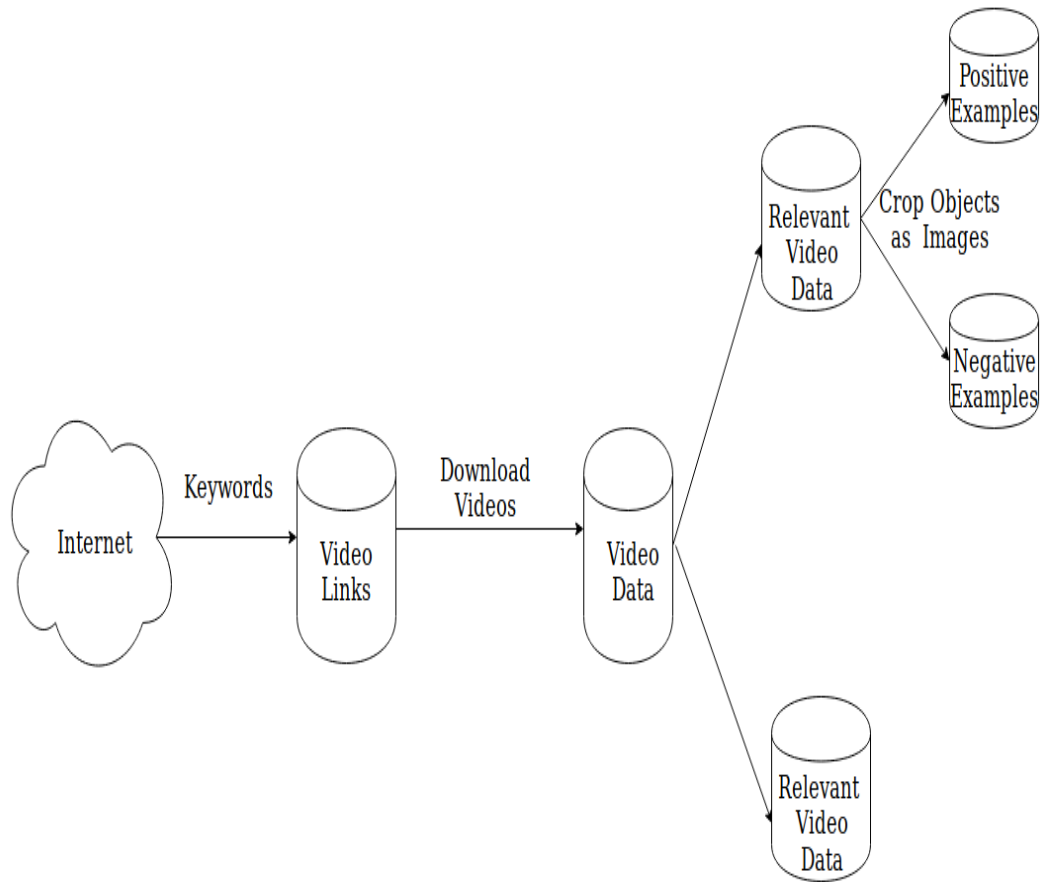


Figure 4.1: data pipeline architecture

"debris floating", "large debris", "passengers possessions floating", "floating pipe ocean clean up", "ocean trash solution", "USV", "AUV", "autonomous ocean exploration", "USV Ocean", "solar USV", "ocean gyre", "humpback whale rescue", "mermaids drone", "whale drone footage", "plastic ocean drone", "woman floating", "drone", "ocean drone footage" and "shark rescue survivor". The greatest contributors to the dataset was from online portals like Youtube, Google and other online news services that may have videos of humans in or on open waters.

In Figure 4.2 the distribution of data sources is shown. The objective was to collect videos, with as wide a variety, as possible. Then, the relevant parts from the videos were assembled into one long (45 minute) video run.

The videos with links was saved, with every search query that was executed. Then, the video links for relevant data (humans in/on the water) were downloaded



Figure 4.2: Distribution of imagery data sources on the internet.

as videos through the freely available link converter, [qdownloader](https://qdownloader.net/)¹. After converting these video links to videos, the ones with the best resolution, either 1080 or 720 pixels, were saved (refer to Appendix A.1 for the list of video data links). Then, the videos were manually segregated into categories.

4.3 Segregate Collected Videos into Two Categories

After collecting the video data they were segregated into relevant and irrelevant sets as far as training was concerned. Category ‘poor’ (Figure 4.3) is irrelevant for training and category ‘good’ (Figure 4.4) is relevant for training. Within each of these categories, there are further sub-divided into positive and negative examples. For example, as shown in Figure 4-3, 4-3a shows a human on land and 4-3b shows a bottle with garbage. These two images are completely irrelevant to the training. In Figure 4.4, the positive image example is Figure 4-4b of a person in the water and the negative image example is Figure 4-4a of a life jacket in the water. These two

¹ <https://qdownloader.net/youtube-video-downloader>

images are relevant for training purposes. The next task is to further analyze the good (relevant) videos.

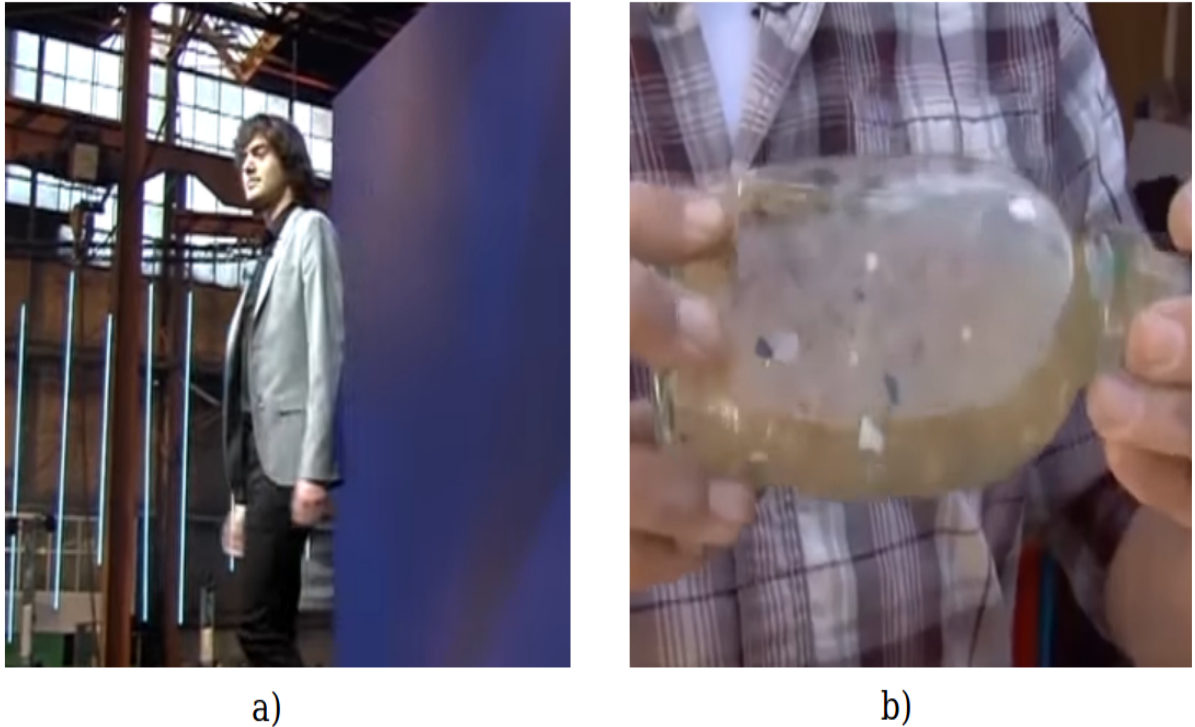


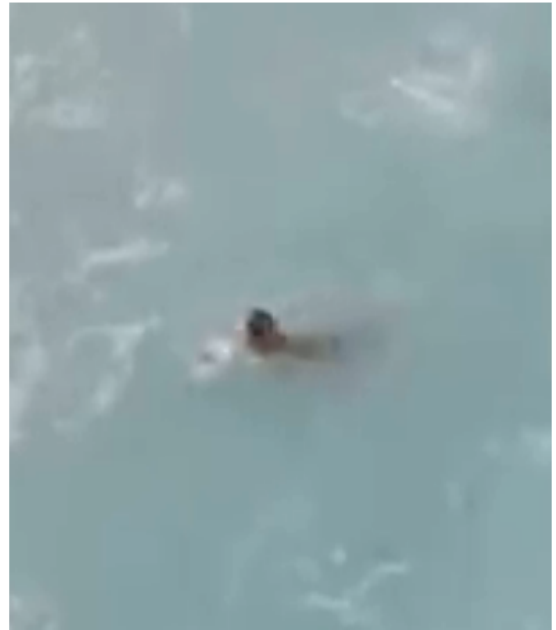
Figure 4.3: Category ‘poor’ for training the teacher model: (a) and (b) are positive and negative examples, respectively.

4.4 Cropping Objects and Labeling

Only good (relevant) videos were further processed. Out of all the good videos, further work was done to extract clips, out of long video sequences, to build an image dataset with the maximum number of positive and negative image examples and still maintain variety. The Vegas Pro tool (Windows OS) was used to assemble (concatenate) all these clips into a new video which had positive and negative examples in no particular order. This video was 44 minutes and 9 seconds in length. Then, each frame was extracted from this one video using the same Vegas Pro tool. A final subset of these frames were selected for still further processing. Objects in this final frame set were manually cropped and labelled (positive, negative) to yield 591 positive (human in/on the water) and 1060 negative (buoys, logs, life jackets, etc.) examples. Finally, this



a)



b)

Figure 4.4: Category ‘good’ for training the teacher model: (a) and (b) are negative and positive examples, respectively.

dataset of images was combined into one folder resulting in 1651 total images.

Giving the implemented architecture and the dataset, experiments were performed. Discussion on the experiments follows in the next chapter.

Chapter 5

Experiments

The objective of the experiments is to gain insight into the concept of distillation to assist in the efficacy-efficiency trade-off for neural networks.

The experiments were designed to address visual detection of humans in/on the water as would be expected in a MOB incident. The rationale to use the network distillation concept was discussed earlier. Effort was applied to build an image classifier based on convolutional neural networks. The unwieldy teacher model (Table 5.1) was trained with transfer learning. The pre-trained weights from the inception V3 network till the mixed middle layer were used for constructing the first part of teacher network. The second part of this network is constructed by applying pooling, then adding a global average pooling layer along with 3 fully-connected layers. As shown in Table 5.2, the student network on the contrary was built with a much smaller architecture.

The pre-trained weights from both the VGG16 and Inception V3 were considered for transfer learning (Chapter 3) to train the teacher model. Inception V3, rather than VGG 16, was chosen because the former had 23 million parameters as opposed to the latter, with 138 million parameters. All the images are approximately 299×299 pixels so the image dimension to read in was $299 \times 299 \times 3$. By using the Inception V3 pre-trained weights until the mixed middle layer, and with three fully-connected additional layers, the architecture was built as shown in Table 5.1. In this table the left column describes the layer type and the right column describes the number of neurons in that layer. After the last convolution layer, global average pooling was applied since it acts as a structural regularizer that explicitly enforces feature maps to be confidence maps of categories, and on top of this, three fully-connected layers were added. After every fully-connected layer, batch normalization was applied to keep the mean zero and to prevent the weights of previous layers from affecting the next layers.

teacher network	
type	input size
conv	$299 \times 299 \times 3$
conv	$149 \times 149 \times 32$
conv padded	$147 \times 147 \times 32$
pool	$147 \times 147 \times 64$
conv	$73 \times 73 \times 64$
conv	$71 \times 71 \times 80$
conv	$35 \times 35 \times 192$
$3 \times Inception$	$35 \times 35 \times 288$
$5 \times Inception$	$17 \times 17 \times 768$
$2 \times Inception$	$8 \times 8 \times 1280$
pool	$8 \times 8 \times 2048$
global average pooling	2048
fully connected	4096
batch normalization	4096
fully connected	1024
batch normalization	1024
fully connected	512
batch normalization	512
dropout	512
output	2

Table 5.1: Teacher network architecture designed to use transfer learning

Neural layers with 4096, 1024 and 512 neurons was used. This is based on the heuristics of the neural networks. Finally, the output layer was added with just two logits as output. The probabilities were interpreted using the softmax activation functions.

Images of the same size as those used in the teacher network were considered for the student network (Table 5.2). The 32-64-128 feature map sequence was used as this gives the best results in terms of accuracy and is also widely used by other similar state-of-the-art networks. Next, the number of neurons are downsampled to 64, and then a fully connected layer with 64 and 128 neurons is finally used.

Experiments to explore both efficacy and efficiency were performed on three different network architecture sizes. They are characterized by 96 million, 11 million and less than a million parameters. The size of their generated session files are 380 MB, 48 MB and 1.8 MB, respectively. Their specifics are summarized in Appendix A.2.

student network	
layer name	input size
conv	$299 \times 299 \times 3$
pool	$150 \times 150 \times 32$
conv	$150 \times 150 \times 64$
pool	$75 \times 75 \times 64$
conv	$75 \times 75 \times 128$
pool	$38 \times 38 \times 128$
fully connected	64
fully connected	128
output	2

Table 5.2: Student network architecture designed.

Their efficacy and efficiency measures were calculated to assess assess performance.

As mentioned earlier, the efficacy of a model is evaluated using measures like *accuracy*, *precision*, and *F1-score*. Efficacy was also more globally assessed with confusion matrices (absolute and normalized) which were plotted to show the strength of the classification algorithm and to understand which parts were penalized heavily. The efficiency of a student model is measured through its memory utilization, processing time, power consumption, memory utilization for computations, and throughput (images processed / second).

The results and discussion on the experimentation is presented next.

Chapter 6

Results and Discussion

6.1 Efficacy Analysis

Experimentation in regard to the performance of the three distilled networks (of different sizes) against the four efficacy measures towards insight on their inference abilities was done. The efficacy measure, accuracy, is plotted as a function of session file (model) size in Figure 6.1.

The four efficacy measures *accuracy*, *recall*, *precision* and *F1-score* with respect to the model size in ascending order, 1.8 MB, 48.5 MB and 379.7 MB, are as follows. Accuracy measures of the three models are 77.8%, 88.6% and 85.5%. *Precision* measures are 78.8%, 88.5% and 85.9%. *Recall* measures are 75.9%, 88.5% and 86.1%. *F1-score* measures are 76.1%, 88.4% and 85.5%. The test dataset is made nearly balanced so accuracy can be analyzed to compare the three models.

All the efficacy measures are calculated on the test dataset which is 166 images, of which 57% are positive examples (human in the image) and the other 43% are negative examples. After analyzing the accuracy of the models, 48.5 MB model was selected to assess the predictions for which the model correctly predicted that 67 images contained humans (true positives) and 80 images did not (true negatives). The model confused 9 images which actually had humans in the water (false negatives) and 10 images which did not have human in the water (false positives). As shown in Figure 6.2, these results show that the model has high efficacy.

The normalized confusion matrix is shown in Figure 6-3. It shows that out of the negative examples, 11% are false positives and 89% are true negatives. Out of the positive examples, 12% are false negatives and 88% are true positives.

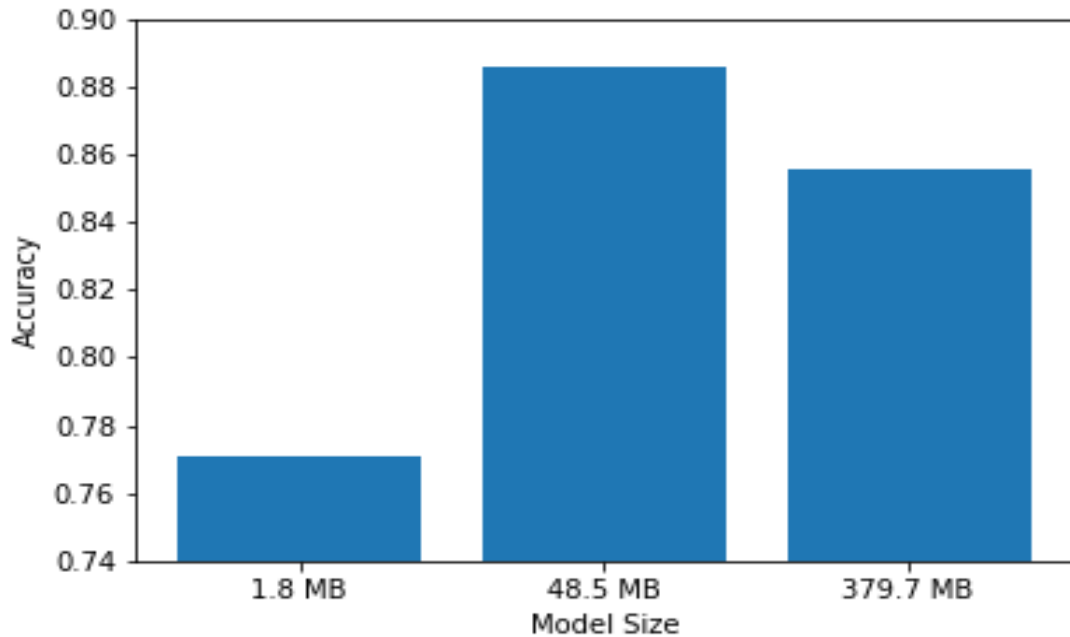


Figure 6.1: Comparing the three models on the basis of accuracy

The distilled neural networks used in the experimentation for all the architectures are summarized in Appendix A.2. These are the architectures used to analyze the efficacy of the neural networks presented in Figure 6.1. Table 6.2 pertains to the 48.5 MB distilled model which was chosen for implementation. Table A.4 in Appendix A.2 refers to the 379.7MB distilled model and Table A.5 in Appendix A.2 to the 1.8MB distilled model.

The model of size 48.5 MB was analyzed so as to understand if the model learnt anything from the teacher model (cumbersome) whose convolutional weights were initialized using transfer learning. So in one architecture the model was trained using transfer learning and in one without. This was specifically done to compare the two loss functions in terms of their contribution towards learning of a model, teacher loss (distillation) and student loss. For this an experiment was done once by training the student network with the guidance of the teacher network and once without guidance. The results were interesting. When the student model was tested for inference on the testing image dataset, student efficacy results were found to be better than for the one trained with teacher model guidance involving transfer learning. As can be seen

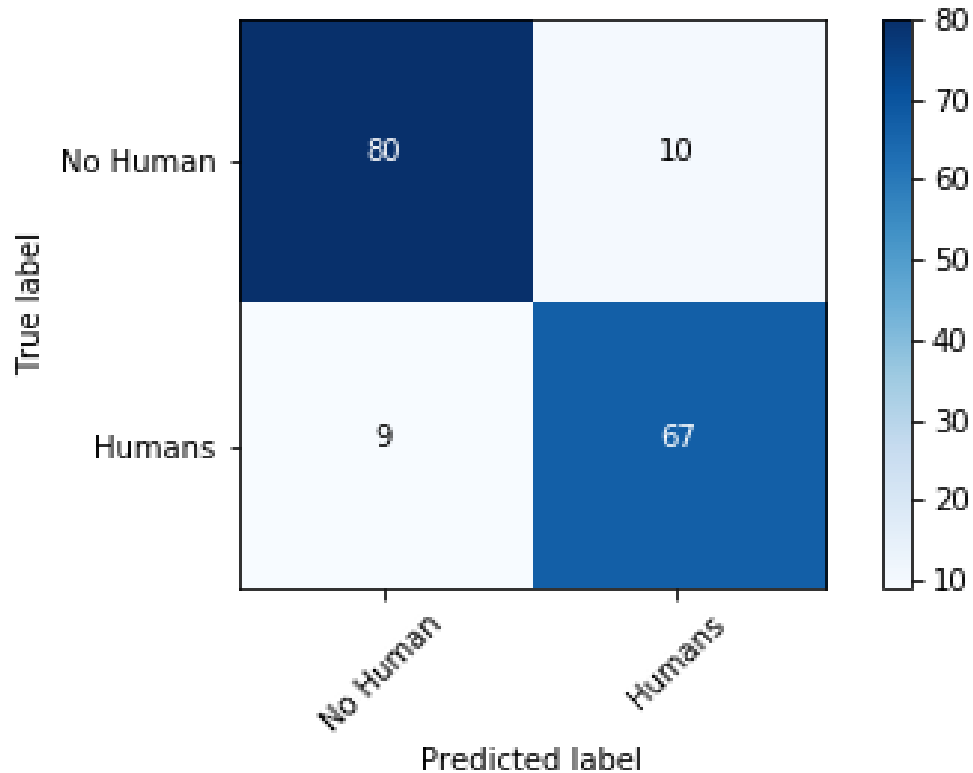


Figure 6.2: Confusion matrix of the human-in-the-water classification with very few false positives and false negatives. This means that the efficacy is high.

in Figure 6.4, the student model achieves around 90% accuracy when trained with teacher guidance and around 82.5% with out teacher guidance. This is a significant gain in terms of learning by having the teacher loss (distillation) function fine tuned with transfer learning along with the student loss.

6.2 Efficiency Analysis

The memory requirements, execution times and image processing rates are defined here as measures of efficiency.

To analyze the memory requirements on the embedded processor, which is an ARM, the different models were tested for inference and their memory use was plotted in Figure 6.4. The three different networks were of size 1.8 MB, 48.5 MB and 379.7 MB. The mean memory consumption of these three network models was 706.82 MB for the smallest, 822.02 MB for the medium and 2414.94 MB for the largest network

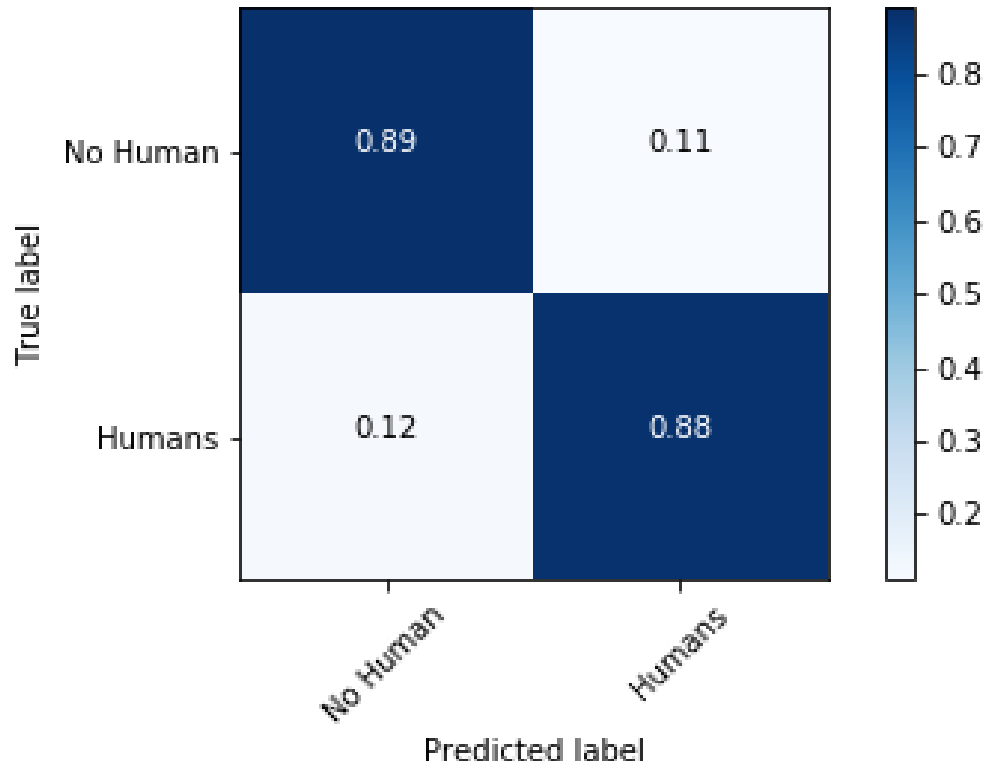


Figure 6.3: Normalized confusion matrix with less relative percentage of FP and FN

model. The three different models take different times for inference computational processing, and the times are around 6, 9 and 31 seconds, respectively.

Other experiments were performed with the three network architectures to assess how many images were being processed per second as another measure of efficiency. As shown in the Figure 6.5, the smallest (1.8 MB) model processes 2 images per second, the medium sized (48.5 MB) model processes 0.88 images per second and the the largest (380 MB) model processes 0.22 images per second.

Figure 6.4 shows that as the network size increases, the memory requirement and processing time both increase, as expected. Neural network size is based on the number of parameters involved in the construction. Specifically, as the model size increases from 48 MB to 380 MB, the required memory increases several times. This is a point of concern when working with resource constraints. When the network size is 1.8 MB, there is a marginal decrease in the memory requirement compared to the 48 MB case. This measure of efficiency shows that the difference in processing time



Figure 6.4: Comparison of test accuracy with and without transfer learning

between the 1.80 MB and 48 MB cases is marginal. Given the 48 MB case gives much higher efficacy (for the same efficiency as the 1.8 MB case) there is no reason not to select the 48 MB student network model.

Next, the image processing rate metric, images processed per second, is analyzed as another measure of network model efficiency.

It is equally important to analyze the results of the experiments in terms of number of images processed per second by the three neural network architectures. The experiment shown in Figure 6.5 has neural network architectures of sizes 1.8 MB, 48.5 MB and 379.7 MB. The images processed per second were 2.0, 0.88 and 0.24, respectively. The processing rate gradually decreased with the increase in network size. This trend can be extended to networks of different sizes, and immense scope lies in increasing the image process rate (efficiency) but at the cost of efficacy.

From the experiments, it was clear that with the network image classifier, the price for efficiency was at the cost of efficacy. It was interesting to note that neural network models of 1 MB and 48 MB (near 50x increase in memory) did not manifest in significant differences in their run-time memory requirements. These were compared to the full 388 MB neural network which used significantly more memory and required more processing time.

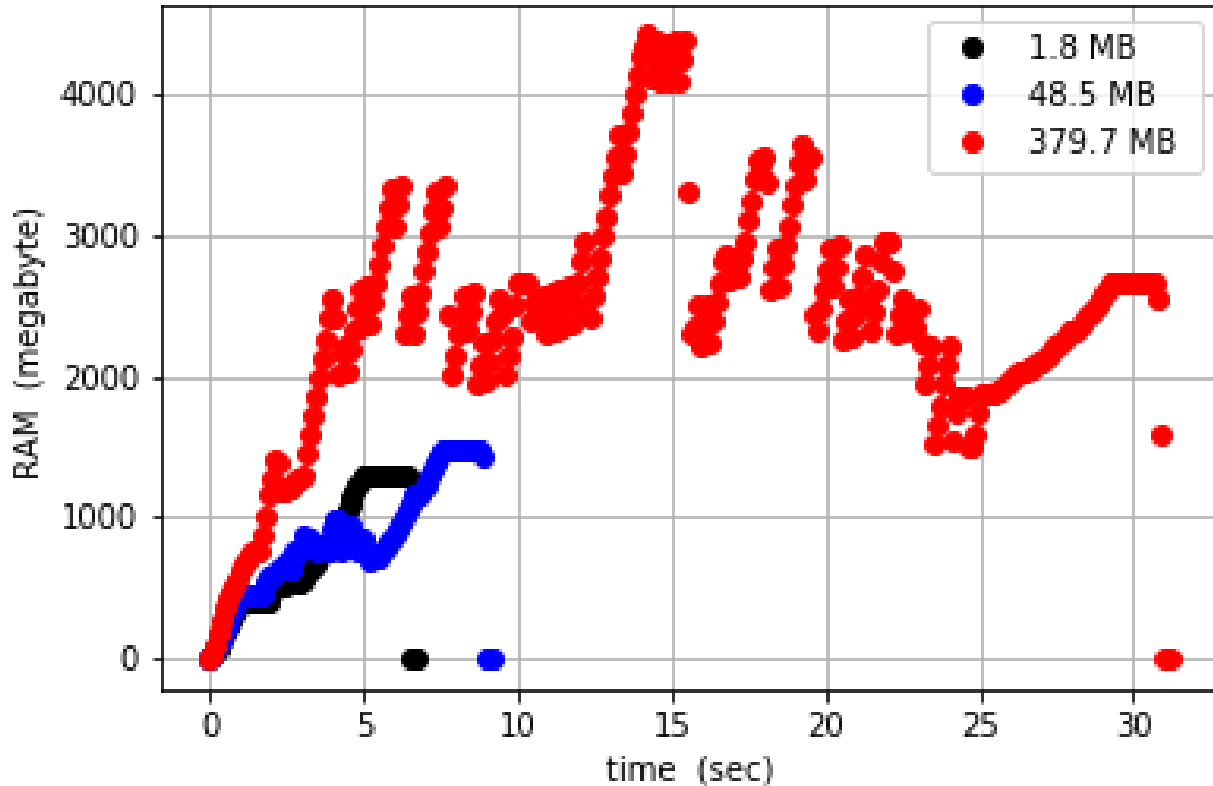


Figure 6.5: Different student models’ computational requirements. Note, the 48.5 MB and 1.8 MB model have nearly same memory requirements.

The research was in the direction of a new paradigm to use the concept of distillation and the teacher-student networks. There were difficulties to set up the embedded processor to build the TensorFlow API from the source code which required specific CUDA requirements. The research reached the stage where experiments could be executed to study the CNN efficacy and efficiency trade-off.

6.3 Contributions

The objective was the efficacy and efficiency in the interpretability of the neural networks considered. Towards this, available solutions were explored in the open literature on interpretability of neural networks. For the distillation technique selected, it

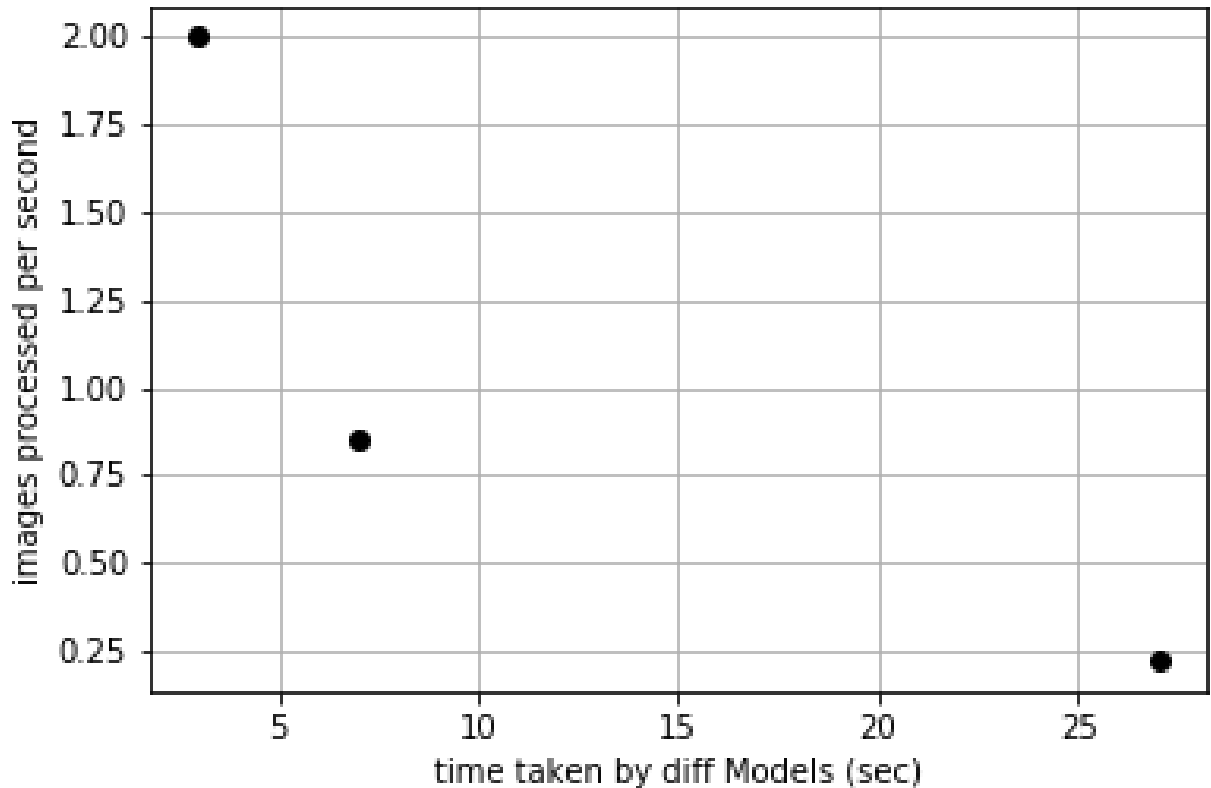


Figure 6.6: Image processing rate for the three different size student network models (from L-R: 1.8 MB, 48 MB and 380 MB). The drop is gradual.

was possible to explore the efficiency as well. As the implementation is ultimately on an unmanned aerial vehicle, the limitations of an embedded processor (Nvidia Jetson TX2) were considered. This stipulated a study that targeted light-weight neural network solutions with constraints on memory and processing power. This, motivated the efficacy and efficiency trade-off question.

With the distillation technique, a new paradigm to train the teacher network was explored using transfer learning with two state-of-the-art classifiers: VGG 16 and Inception V3. Inception V3 was ultimately chosen as it had much less parameters compared to the more unwieldy VGG 16 network. Distilling the knowledge from teacher to student has been the main point of exploration. This successfully yielded a much smaller student network with the insights of the teacher network and could

execute on the NVIDIA Jetson TX2 embedded processor.

Another important contribution was the 1651 images dataset produced for this problem since the literature survey could not identify available repositories. Objective keywords were compiled to facilitate the best internet searches. ‘Good’ videos were downloaded from Youtube, Google and other portals. Manual efforts were applied to crop and label these images. This dataset would be useful for follow-on research and other colleagues working in the same area.

Chapter 7

Conclusion

The experimental evidence confirms that the hypothesis has been proved as the neural network architecture of 48.7 MB achieved results similar to 388.8 MB; and for the efficiency measure, the results were even better. Also, the task of building a vision based system to be deployed on a drone was achieved. The solution is planned to be deployed with a object cropping tool to confirm whether or not there is a human in the cropped object image.

Extensive work has been undertaken by the deep learning community to solve domain-centric problems namely, detecting objects, generating images, etc. Therefore, this thesis focussed on neural network models that are both efficacious and efficient as it would be of wider interest. State-of-the-art neural networks were used in this thesis to train other networks using transfer learning. With the distillation approach, light-weight neural networks were successfully trained and deployed on an embedded processor.

Since the advent of the high performance computing platforms, there have been used extensively to run computationally expensive operations. The emerging use of machine learning in robotics has seen the focus shift from neural networks optimized with efficiency considerations in addition to the usual efficacy. This area was explored with an example embedded processor, the Nvidia Jetson TX2, and using its capabilities to drive the requirements for the neural network.

The MOB incident is a critical when it occurs, so the emphasis is not usually on capturing imagery. Consequently, there is very little in the way of image repositories or even related imagery to build a training set with. Nonetheless, an image set was culled from internet sources of people in/on the water.

The result of experimenting with different network architectures was a model which was very good in terms of both efficacy and efficiency. Though the research objective was to develop network models that were on the order of tens of megabytes

to megabytes, this process can be extended to reducing them to as small as kilobytes – at the cost of efficacy.

For future work, the next step is to design an automated decision tree-based search for the most optimized network given efficacy and efficiency requirements. Also, pruning techniques could be studied to analyze the efficacy-efficiency trade-off. Image cropping was labour intensive so future work could consider building an automated cropper for a video stream.

Bibliography

- [1] ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G., ISARD, M., ET AL. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (2016), pp. 265–283.
- [2] BROWN, B., CHUI, M., AND MANYIKA, J. Are you ready for the era of big data. *McKinsey Quarterly* 4, 1 (2011), 24–35.
- [3] BUONAIUTO, N., LOUIE, M., AARESTAD, J., MITAL, R., MATEIK, D., SIVILLI, R., BHOPALE, A., KIEF, C., AND ZUFELT, B. Satellite identification imaging for small satellites using nvidia.
- [4] CASTELLANO, G., FANELLI, A. M., AND PELILLO, M. An iterative pruning algorithm for feedforward neural networks. *IEEE transactions on Neural networks* 8, 3 (1997), 519–531.
- [5] DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., AND FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (2009), Ieee, pp. 248–255.
- [6] DODDINGTON, G. R., AND PAWATE, B. I. Efficient pruning algorithm for hidden markov model speech recognition, Dec. 11 1990. US Patent 4,977,598.
- [7] FRYER, D. M., HAYES, F. W., AND MARSHALL, R. S. Man overboard rescue system, July 8 1986. US Patent 4,599,073.
- [8] GALLEGRO, A.-J., PERTUSA, A., GIL, P., AND FISHER, R. B. Detection of bodies in maritime rescue operations using unmanned aerial vehicles with multispectral cameras. *Journal of Field Robotics*.
- [9] GELFAND, S. B., RAVISHANKAR, C., AND DELP, E. J. An iterative growing and pruning algorithm for classification tree design. In *Conference Proceedings., IEEE International Conference on Systems, Man and Cybernetics* (1989), IEEE, pp. 818–823.
- [10] HINTON, G., VINYALS, O., AND DEAN, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
- [11] HOLLAND, M. J., AND RAINEY, M. J. Man overboard alert and locating system, Oct. 31 1995. US Patent 5,463,598.
- [12] JENSEN, M. C., HINES, J. M., AND FOALE, C. M. Man overboard rescue. *Simulation* 57, 1 (1991), 39–47.

- [13] KEARNS, M. J., AND MANSOUR, Y. A fast, bottom-up decision tree pruning algorithm with near-optimal generalization. In *ICML (1998)*, vol. 98, Citeseer, pp. 269–277.
- [14] LEDUC, G., ET AL. Road traffic data: Collection methods and applications. *Working Papers on Energy, Transport and Climate Change 1*, 55 (2008).
- [15] McDONALD, C. D. Man-overboard rescue apparatus for sailboats, Aug. 10 1982. US Patent 4,343,056.
- [16] McNITT-GRAY, M. F., ARMATO III, S. G., MEYER, C. R., REEVES, A. P., McLENNAN, G., PAIS, R. C., FREYMAN, J., BROWN, M. S., ENGELMANN, R. M., BLAND, P. H., ET AL. The lung image database consortium (lidc) data collection process for nodule detection and annotation. *Academic radiology* 14, 12 (2007), 1464–1474.
- [17] NAPHADE, M., ANASTASIU, D. C., SHARMA, A., JAGRLAMUDI, V., JEON, H., LIU, K., CHANG, M.-C., LYU, S., AND GAO, Z. The nvidia ai city challenge. In *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI)* (2017), IEEE, pp. 1–6.
- [18] PAUL, M., HAQUE, S. M., AND CHAKRABORTY, S. Human detection in surveillance videos and its applications-a review. *EURASIP Journal on Advances in Signal Processing* 2013, 1 (2013), 176.
- [19] SADOWSKI, C., AFTANDILIAN, E., EAGLE, A., MILLER-CUSHON, L., AND JASPAN, C. Lessons from building static analysis tools at google.
- [20] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [21] SMITHERMAN, C. L. Vehicle based data collection and processing system and imaging sensor system and methods thereof, May 25 2010. US Patent 7,725,258.
- [22] ŠUL’AJ, P., HALUŠKA, R., OVSENÍK, L., MARCHEVSKÝ, S., AND KRAMAR, V. Examples of real-time uav data processing with cloud computing. In *Conference of Open Innovation Association, FRUCT* (2018), no. 23, FRUCT Oy, pp. 543–548.
- [23] SZEGEDY, C., VANHOUCHE, V., IOFFE, S., SHLENS, J., AND WOJNA, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2818–2826.

- [24] TAN, J., ZHANG, T., COUMANS, E., ISCEN, A., BAI, Y., HAFNER, D., BOHEZ, S., AND VANHOUCHE, V. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332* (2018).
- [25] VIGGIANO, A. G., AND LOSCHIAVO, M. J. Sailboat safety system for a person falling overboard, Mar. 3 2009. US Patent 7,497,181.
- [26] YOSINSKI, J., CLUNE, J., BENGIO, Y., AND LIPSON, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems* (2014), pp. 3320–3328.

Appendix A

A.1 Different quality of data collection

Here, all the datasets links have been provided that were collected during the research, this has been summarized using three tables. And these are the initial set of videos that were collected for extracting the frames out of these, which eventually were cropped in regard to extracting objects from these frames. So the collected videos and auxiliary images are basically of two types, one which refer to the good quality data and the other which refers to the irrelevant data or something that can be termed as garbage. The tables are shown with the two columns one with the index and the other with the URL.

The first table Table A.1 corresponds to the Youtube video dataset which is the useful videos dataset, having good amount of positive and negative examples. The richness of this source is both in terms of variety and quantity.

The second table Table A.2 corresponds to the Youtube video dataset which is the not relevant videos dataset, this is based on the quality of positive and negative examples. Like in these videos there aren't any good examples of human in water and if there are human instances in here, then they are not in regard to our MOB use case. An illustration of the quality in terms of positive and negative example in here is, a man giving speech on a podium and debris in a plastic bottle.

The third table Table A.3 corresponds to a few good quality google image search results which we consider quite relevant to our use case. Though we have taken just a few out of these image set into our use case.

Level 1 : Good quality video data	
index	URL
1	https://www.youtube.com/watch?v=JyfbFBdnKYM&t=29s
2	https://www.youtube.com/watch?v=fXgojI2VUCI&t=37s
3	https://www.youtube.com/watch?v=xGRWUQQJiIY&t=481s
4	https://www.youtube.com/watch?v=GWdiTpWaceo
5	https://www.youtube.com/watch?v=_Op03x7HEXU&t=21s
6	https://www.youtube.com/watch?v=yDrx5zi_mYU
7	https://www.youtube.com/watch?v=bSG3kPa-jOU
8	https://www.youtube.com/watch?v=M_IpPOUhjTY&t=78s
9	ME40testinginopensearegion(Done)Project_40_9
10	https://www.youtube.com/watch?v=1xcEAX_CDqA&t=125s
11	https://www.youtube.com/watch?v=_y1MOfUWyXI
12	https://www.youtube.com/watch?v=UIPSkUsQynI
13	https://www.youtube.com/watch?v=BHE3bFlutPI&t=6s
14	https://www.youtube.com/watch?v=YyK80erKEts&t=12s
15	https://www.youtube.com/watch?v=2kUNgD94CGO
16	https://www.youtube.com/watch?v=ivNnpbOR1BM
17	https://www.youtube.com/watch?v=Uxcmk1Gk00U&t=54s
18	https://www.youtube.com/watch?v=FuZGdVBCXI&t=54s
19	https://www.youtube.com/watch?v=3ge4KwckSY&t=54s
20	MeetSwarmDiver-MicroDivingUSVforOceanSwarming
21	https://www.youtube.com/watch?v=94KWh0W1h-A&t=1s
22	https://www.youtube.com/watch?v=GSMGkZBaWM
23	https://www.youtube.com/watch?v=StNZ3XUBDYw&t=104s
24	https://www.youtube.com/watch?v=vqZ-Cg_zxoY&t=23s
25	https://www.youtube.com/watch?v=Q5K1kckWImg&t=11s
26	https://www.youtube.com/watch?v=Qbusgy13LhE
27	https://www.youtube.com/watch?v=ebA811jL8cg&t=74s
28	https://www.youtube.com/watch?v=olYCEStQpHQ&t=32s
29	https://www.youtube.com/watch?v=BLG2v8yD0Fk
30	https://www.youtube.com/watch?v=LI2Tkprc1qM
31	https://www.youtube.com/watch?v=rh0548HxV2k
32	https://www.youtube.com/watch?v=txBTVAB0WJs
33	HedgehogRescue-seenthroughGoogleGlass
34	https://www.youtube.com/watch?v=CKPTHzEdpAU&t=28s
35	https://www.youtube.com/watch?v=0Wvi_Y7UJJI

Table A.1: Good quality video clips set from Youtube

Level 2 : Bad video quality data	
index	URL
1	https://www.youtube.com/watch?v=WvfXqEF1kZU&t=26s
2	https://www.youtube.com/watch?v=itaU7maHT8o
3	https://www.youtube.com/watch?v=TN1Y7u1x_zU
4	https://www.youtube.com/watch?v=PFwHcr_FMw8&t=14s
5	https://www.youtube.com/watch?v=w2H07DRv2_M
6	https://www.youtube.com/watch?v=1QR80yYHUzI
7	https://www.youtube.com/watch?v=5E06a3D_6M8
8	https://www.youtube.com/watch?v=IEII36wCXnA
9	https://www.youtube.com/watch?v=SPGR_rczwX0&t=5s
10.	https://www.youtube.com/watch?v=bVQ8i0UTbQg

Table A.2: Bad quality video clips set from Youtube

Level 1 auxiliary data source	
index	URL
1	https://www.google.ca/search?biw=1685&bih=892&tbm=isch&sa=1&ei=CyBjXNLcI4r4jwThsolo&q=debris+on+sea+pictures+taken+from+helicopter&oq=debris+on+sea+pictures+taken+from+helicopter&gs_l=img.3...2771.12061..12395...5.0..0.293.3294.0j26j1.....1...1.gws-wiz-img.eHNwMKqJxJ4#imgrc=x6alYbtpsrVqGM:
2	https://www.google.ca/search?biw=1685&bih=892&tbm=isch&sa=1&ei=GCBjXP_206K9jwTllqiwDA&q=boats+in+ocean&oq=boats+in+ocean&gs_l=img.3..0.193070.196878..197165...0.0..2.288.1887.0j13j1.....2...1..gws-wiz-img.....0..35i39j0i67j0i8i30j0i24.sX6fPzhcuko#imgrc=bYnzpra9sQqYmM:

Table A.3: Good quality auxiliary relevant image set

A.2 Different distilled neural networks

The other two types of network sizes are 1.8 MB and 380 MB. Other than the 48 MB network described in the experimentation section, the number of neurons are increased from 128 neurons to 512. Finally, these 512 neurons are attached to just 2 output layer neurons, this increases the network size to manifolds as the fully connected layer connecting to convolutional part is the most expensive operation. Next, 380 MB network is built by modifying it by doing a 1*1 convolution and then connecting the reduced feature map space to 128 neurons, which are finally connected to two output neurons.


Student Network	
Layer Name	input size
conv	$299 \times 299 \times 32$
pool	$150 \times 150 \times 32$
conv	$150 \times 150 \times 64$
pool	$75 \times 75 \times 64$
conv	$75 \times 75 \times 128$
pool	$38 \times 38 \times 128$
fully connected	512
output	2

Table A.4: Neural network model of size 379.7 MB

Student Network	
Layer Name	input size
conv	$299 \times 299 \times 32$
pool	$150 \times 150 \times 32$
conv	$150 \times 150 \times 64$
pool	$75 \times 75 \times 64$
conv	$75 \times 75 \times 128$
pool	$38 \times 38 \times 128$
1*1 conv	1
fully connected	64
output	2

Table A.5: Neural network model of size 1.8 MB

A.3 Hardware Specifications



JETSON TX2

TX2 4GB	TX2	TX2I
1.3 TOPs		
256-core NVIDIA Pascal™ GPU		
Dual-Core NVIDIA Denver 2 64-Bit CPU and Quad-Core ARM® Cortex®-A57 MPCore		
4GB 128-bit LPDDR4 Memory	8GB 128-bit LPDDR4 Memory	8GB 128-bit LPDDR4 (ECC support)
16GB eMMC 5.1	32GB eMMC 5.1	32GB eMMC 5.1
7.5W / 15W		10W / 20W
Gen 2 1x4 + 1x1 OR 2x1 + 1x2		
12x CSI2 D-PHY 1.1 lanes (2.5 Gbps/Lane)		
Two Multi-Mode DP 1.2 eDP 1.4 HDMI 2.0 Two 1x4 DSI (1.5Gbps/lane)		
—		
—		
No	Yes	No

Figure A.1: Hardware specifications of Nvidia Jetson TX2