FISH SPECIES RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

TAN YING YING

SUBMITTED TO THE

FACULTY OF ENGINERRING

UNIVERSITY OF MALAYA, IN PARTIAL

FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING

(MECHATRONICS)

2018

UNIVERSITY MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: TAN YING YING

Registration/Matric No: KQF170003

Name of Degree: MASTER OF ENGINEERING (MECHATRONICS)

Title of project Paper / Research Report: FISH SPECIES RECOGNITION USING CONVOLUTION NEURAL NETWORK

Field of Study: IMAGE PROCESSING，ARTIFICIAL INTELLIGENCE

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any except or extract from, or reference to or reproduction of any copyright work and its authorship have been acknowledged in this work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this work to the University Malaya("UM"), who henceforth shall be the owner of the copyright this work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature: Date:

Subscribed and solemnly declared before,

Witness's Signature: Date:

Name:

Designation:

ABSTRACT

Fish Recognition using machine learning is one of the significant breakthroughs that could be achieved by marine researchers and marine scientists. With the advancement of the machine learning in marine field, some of the problems that perplexed researchers can be solved especially in data collection. Application of machine learning to marine field is still immature, many aspects still need to be improved. Differentiating between two fish species with similar appearance is relatively challenging. On top of that, the angle of fish in the images and the background of the images can cause confusion to the recognition system. Therefore, it is quite challenging to build a fish recognition system. This study focuses on designing a fish recognition system by using Convolutional Neural Network (CNN). The proposed method employs Network-in-Network (NIN) model for fish recognition. NIN model using Multilayer Perceptron (Mlpconv) instead of linear filter and apply Global Average Pooling (GAP) for the last pooling layers. The result of NIN is then compared with a 3 layers CNN. To verify the utility of the proposed model, a set of data is prepared for prediction after training. The performance of the model assessed based on the F1-score of the test data. The accuracy of the developed system is 83%.

ABSTRAK

Pengecaman ikan dengan menggunakan penglihatan robotik merupakan salah satu kejayaan penting yang boleh dicapai oleh penyelidik dan saintis marin. Kemajuan penglihatan robotik dalam bidang marin telh menyelesaikan sebahagian masalah yang manganggu penyelidik terutamanya masalah pengumpulan data. Aplikasi penglihatan robotik masih belum matang, terdapat banyak aspek yang perlu diperbaiki. Pengenalian jenis ikan yang serupa adalah tugasan yang agak mencabar. Lebih-lebih lagi, sudut ikan dalam gambar serta latar belakang gambar boleh mengelirukan sistem. Oleh yang demikian, pembinaan sistem pengecaman ikan agak mencabar. Dalam kajian ini, tumpuan akan diberikan kepada mereka bentuk sistem pengecaman ikan berdasarkan Teknik Rangkaian Konvolusi Neural (CNN). Model yang digunakan untuk mereka bentuk sistem pengcaman ikan adalah model Rangkaian dalam Rangkaian (NIN). Model NIN menggunakan *Multilayer Perceptron* (Mlpconv) untuk menggantikan penapisan linear and menggunakan *Global Average Pooling* (GAP) unutk lapisan *Pooling* terakhir. Keputusan NIN akan dibandingkan dengan model CNN dengan 3 lapisan. Untuk mengesahkan utiliti model yang dicadang, satu set data telah disediakan untuk ramalan selepas habis latihan. Prestasi model akan dinilai berdasarkan *F1-score* data ujian. Ketepatan sistem pengecaman ikan yang dibina ialah 83%.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Universiti Malaya (UM) for providing me such a good study environment and facilities throughout my one-year degree study. I consider myself as a very lucky individual for able to grab the opportunity to study in such wonderful university. I feel grateful for having chance to meet many wonderful people and professionals who provide a pleasant experience throughout this master program.

In addition, I am using this great opportunity to express my deepest gratitude and special thanks to lecturers from Faculty of Engineering who have provided me with information and preparation during the process of conducting my study.

I am using this opportunity to express my deepest gratitude and special thanks to my final year project's supervisor, Ir. Dr. Chuah Joon Huang, who is Head of VIP Research Group. Inspite of being extraordinary busy with his duties, he took time out to guide and keep me on the correct path while dealing with my project.

Special thanks to my family members, coursemates, peers and lecturers for their moral support and technical support. I choose this moment to acknowledge their contribution.

I perceive as this opportunity as a big milestone in my skill development. I will strive to use skills and knowledge obtained in the best possible way.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

Adam	: Adaptive Moment Estimation
AI	: Artificial Intelligence
CNN	: Convolutional Neural Network
ECO	: Evolution-COnstruced
GAP	: Global Average Pooling
GLCM	: Gray level co-occurance matrix
HOG	: Histogram of Oriented Gradient
HSV	: Hue Saturation Value
Mlpconv	: Multilayer Perceptrons
PAF	: Part-Aware Feature
PCA	: Principal Component Analysis
R-CNN	: Fast Regions with Convolutional Neural Network
ReLU	: Rectified Linear Unit
RPN	: Region Proposal Network
SIOPL	: Scale-Invariant Part Learning
SVM	: Support Vector Machine

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

1.1 Introduction

This chapter describes the background and problem statements to give an idea of the contribution of this research work. The objectives and scope of the study are also described here.

1.2 Research Background

The purpose of this project is to develop a fish recognition system using Convolutional Neural Network (CNN). Fish recognition system plays significant roles in marine biology and aquatic science.

Marine researchers required to count, to observe and to differentiate underwater organisms in their research. Long-term data sets are needed to be acquired to delineate the ecology yet the dynamic condition of underwater environment. The inaccessibility of the marine environment leads to low sustainability in long term real time observation. Conventional method in retrieving data required long-term and labour-intensive effort(Hsiao, Chen, Lin, & Lin, 2014). Fish recognition system can assists researchers in the study of the aquatic habitat such as distribution of fish species, behaviour of different fish species as well as their interactions (Ding et al., 2017). Besides, a high efficient fish recognition system can reduces the time-consuming task by human observer(Xiu, Min, Qin, & Liansheng, 2015).

As pointed out by Chouiten (2013), fish recognition technology is helpful in wildlife monitoring for marine researchers and fish recognition for divers. This technology can be developed into an offline mobile application which enable user during diving.

1.2.1 Challenges in Fish Species Recognition

One of the main obstacles in fish species recognition is the noise and distortion of the images. The distortion included Gaussian noise, image blur and motion blur They are caused by different factors during image acquisition, for example quality of the camera used, the water quality, light intensity and the complexity of the underwater environment. However, there are various pre-processing techniques to minimize noise and distortion before images are fed into training model.

Another concern in this project is the optimum utilization of the system. Convolution neural network (CNN) is a multilayer neural network which consists of 5 types of layer: input layer, convolutions layer, pooling layer, fully-connected layer as well as output layer. Among the layers, convolutions layer and pooling layers can be used more than once in a system and both of these layers are arranged alternately(Ding et al., 2017). Increment of the neuron layer can increase the accuracy of the result, yet too much of neuron layers burden the processing system as well as reduce efficiency.

1.3 Problem Statement

Object recognition technique can be applied on variety applications, including ecology study, medical image processing, crime scene investigation and military application. Currently, machine vision and machine learning are getting popularized in assisting biologists or researchers in ecology studies such as aquatic ecosystem and forest ecosystem. One of the specific usage is fish identification.

In the process of identifying a specific object from images, images must undergo pre-processing stages before identification technique can be applied on the images. Different pre-processing algorithms applied to enhance the quality of images before further analysis. For instance, smoothing and size normalization. Thus, the research question is *how to implement a suitable process for the images to increase the accuracy of prediction?*

From the images captured from underwater, sometimes the fish in the image is hiding in seaweed or coral and sometimes the fish only occupies a small portion out of overall image. There are cases the colour of fish is identical to the background due to light intensity and protection colour of the fish skin. These circumstances increase the difficulty in training a CNN model. The second research question is *how to recognize the fish from images and identify their species?*

1.4 Objectives of the Research

From the research question stated in problem statement, the research objectives are:

1. To implement suitable image pre-processing technique on fish images.
2. To design and develop a convolutional neural network (CNN) system that can identify species of fish.
3. To evaluate the accuracy of the fish recognition system

1.5 Scope of Research

The scope of the research are as follows:

- The project is conducted by using Python.
- 10 fish species are tested.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter begins with an overview of artificial intelligence. Then, the past works in terms of object recognition as well as fish recognition. This chapter also discusses about the constraints in the research field and the significance of it. At the end of this chapter, a summary for the whole chapter was discussed.

2.2 Artificial Intelligence Overview

According to Appin Knowledge Solutions, Artificial Intelligence is categorized under one of the robotic field and it is defined as a branch of computer science and engineering that deals with intelligent behaviour, learning and adaption in machine. Based on Smith (2003), artificial intelligence is defined as below:

- An area of study in the field of computer science. AI is concerned with the development of computers able to engage in human-like thought processes such as learning, reasoning and self-correction.
- Artificial Intelligence is the concept that machines can be improved to assume some capabilities normally thought to be like human intelligence such as learning, adapting, self-correction, etc.

In simple, most definitions describe Artificial Intelligence as the system:

- which think like human
- which act like human
- which think rationally
- that act rationally

Artificial Intelligence diverges into different branches as shown in Table 2.1. Some types can standalone, some is independent on other types and sometimes they are design to work together to achieve better result.

Table 2.1: The classification of artificial intelligence.

Classification	**Explanation**
Pattern Recognition	A branch of machine learning that emphasizes the recognition of data patterns or data regularities in a given scenario or images.
Expert System	A knowledge-based system that employs knowledge about its application domain and uses an inferencing (reason) procedure to solve problems that would otherwise require human competence or expertise.
Search	Search algorithm is the universal problem-solving technique in artificial intelligence. There are three classes of problems that addressed by search algorithm which are single-agent path-finding problems, two-players games and constraint-satisfaction problems.
Neural Network	A massively parallel distributed processor made up of simple processing with that has a natural propensity for storing experiential knowledge and making it available for use(Haykin, 2009).
Evolutionary Algorithm	Evolutionary algorithm imitates the theory of evolution of biological creature. A component of evolutionary computation in AI. An evolutionary algorithm functions through the selection process in which the least fit members of the population set are eliminated, whereas the fit members are allowed to survive and continue until the better solutions are determined.
Fuzzy Logic	A logic operations method based on many-valued logic rather than binary logic (two-valued logic). Two-valued logic often considers 0 to be false and 1 to be true. However, fuzzy logic

	deals with truth values between 0 and 1, and these values are considered as intensity (degrees) of truth.
Predictive Analytics	Predictive Analytics uses many different techniques to analyse historical data and thus make predictions about future.
Deep Learning	Deep Learning is a set of deep structure machine learning algorithms which model high-level abstractions in data.
Inference	Inference has close relationship with deep learning. After training, inference provide correct answer by taking smaller batches of data(Copeland, 2016).
Natural Language Processing	A technique refers to the interaction between machine and human languages.
Machine Learning	An application of AI that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.
Optimization	An algorithm which used to seek for optimum solution, maximum and minimum in a mathematical function. The algorithm also been used to evaluate design trade-offs(Hardesty, 2015).

2.3 Image Pre-Processing

For the past four to five decades, the processing techniques develop rapidly especially when the machine learning is getting increasingly popular. Image processing is applicable to unmanned spacecraft, military usage, video surveillance, medical imaging, etc.

Image recognition is a multidisciplinary field which involves image processing, machine vision and artificial intelligence. Image recognition goes through phases such as image data sets acquisition, data sets pre-processing, feature extraction and classification. Image pre-processing is a significant step at which the images acquired are being normalized and remove variations to increase recognition rate for better feature extraction during next phase. (Alginahi, 2010)

There are various factors which affecting the quality of the image captured, including camera quality, video resolution, environment, lighting, noise, visual angle not perpendicular, etc. These factors induce the reduction in the recognition rate of the model or system during feature extraction. (Alginahi, 2010; Luo, Li, Wang, Li, & Sun, 2015)

In order to enhance the image's quality before object identification technique is applied, image pre-processing technique is necessary to be implemented on the images. For instance, noise removal, image enhancement, segmentation, normalization, etc.

Image enhancement improves the quality of images by removing noise, reducing blurring as well as increasing. Image enhancement can be performed on the images mainly via three enhancement techniques, contrast stretching, noise filtering and histogram modification(Chitradevi & Srimathi, 2014). Contrast stretching is that the contrast of the images obtained are enhanced by scaling gray level of each pixel, hence image gray levels occupy the entire dynamic range available. Image smoothing is applied when there are spurious noises and tiny gaps in curves or lines in the images. Smoothing technique can bridge the gaps and diminish the effect of noise prior to recognition. Noise filtering removes noise and unnecessary information from images. Histogram modification modifies image's characteristic.

Image segmentation is a crucial step in image processing at which it subdivides image into constituent area or objects. This process terminates once the attributes of interest isolated or extracted from the image. Segmentation is based on the discontinuities and similarities properties of intensity values. Discontinuity performs image partition based on sudden changes in intensity, for example points, lines and edges. As for similarity, it segments the image into similar region based on predefined criteria. Image segmentation applied to the images via edge detection techniques, thresholding techniques and clustering.

Various numerous techniques are introduced and applied to obtain better result in research. Table 2.2 shows the pre-processing technique which is applied before classification and recognition.

Table 2.2: The pre-processing technique used on different recognition method and their researchers.

Author	**Pre-processing Technique**	**Classification/ Recognition Method**
Ding et al. (2017)	Regularize, Uniform the size of data set	CNN
Jin and Liang (2017)	Improved Median Filter	CNN
Sun, Shi, Dong, and Wang (2016)	Fast Direct Super-Resolution (FDSR)	PCANet & NIN
Qin, Li, Liang, Peng, and Zhang (2016)	Foreground Extraction, Principal Component Analysis (PCA) filter	Cascade Deep Network
Huang (2016)	Grabcut algorithm, Gaussion filter	Hierarchy Classification
Kartika and Herumurti (2016)	K-means, HSV, quantization	Support Vector Machine, Naïve Bayes
Saitoh, Shibata, and Miyazono (2015)	Feature points-based approach, normalization (size and orientation)	Geometric features + Bag of visual words (BoVW) model

Researchers implement different pre-processing technique to their image datasets according to their models and the condition of the image datasets. From the literature review, the researchers who build CNN model for recognition normally resize images, regularize images and only apply a filter if the image dataset consists of noises.

2.4 Fish Recognition: Review

Over years, a lot of researchers have involved themselves in the fish recognition or fish classification topics. Starting from 1990's until now, different algorithms, classifiers and models being introduced to be implemented marine eco-study, for the purpose of fish classification, fish counting, ecosystem observation, etc. The increment of computer's computational power and the advancement of electronics and computer hardware provides researchers a better condition to implement complex and sophisticated models. Fish recognition systems can recognize fish in two different forms, i.e. picture and video. To achieve a high accuracy in fish recognition, researchers have develope and applied different methods to train the system. Many models such as convolutional neural network (CNN), Gray level co-occurance matrix (GLCM), Graph Embedding Discriminant Analysis are used for feature extraction and classification.

Convolutional neural network (CNN) is popular among researchers as there are a large number of researches use CNN to perform classification. Ding et al. (2017) compared CNN with different number of hidden layers which are 5 layers, 6 layers and 7 layers respectively. Jin and Liang (2017) suggested a model for fish image recognition with small sample size. Images are fed to train a CNN model. Pre-processed images are used to adjust the pre-trained neural network and finally test the capability of the system. Li, Shang, Hao, and Yang (2016) proposed a method which is the combination of Region Proposal Network (RPN) and Fast Regions with Convolutional Neural Network (R-CNN).

Sun et al. (2016) used 2 deep learning methods, PCANet and Network in Network (NIN) model to perform classification. A comparison was made between PCANet and

NIN on original underwater images (OR) and super-resolution underwater image (SR). Qin et al. (2016) proposed a framework named simple cascade deep network for fish recognition. Xiu et al. (2015) applied Fast Regions with Convolutional Neural Network (R-CNN) to a specific underwater environment.

Hasija, Buragohain, and Indu (2017) proposed Graph Embedding Discriminant Analysis which is an enhance version of Grassmannian Discriminant Analysis. The process of Grassmannian Discriminant Analysis involves the representation of different fish species as different image sets and perform image sets pairing. The image sets are model as subspace. Graph Embedding Discriminant Analysis improve the accuracy by using within-class and between-class separability graphs.

Huang (2016) proposed a method named hierarchy classification for fish recognition. The hierarchy classification proposed is improved by adding 2 rules to reduce the error in average accuracy.

Kartika and Herumurti (2016) proposed a method which combines k-means method and Hue Saturation Value (HSV) as an image segmentation method to classify Koi fish. K-means used to remove the background noise whilst HSV is used to obtain the colour features to determine the species of fish. Finally, apply Support Vector Machine (SVM) and Naïve Bayes for classification. Both method is tested with validation and without cross validation.

Zhang, Lee, Zhang, Tippetts, and Lillywhite (2016) proposed a method called Evolution-COnstructed (ECO) Features. ECO was used to construct efficient features without human experts for fish species classification. Feature construction is a process that discovers missing information regarding the connection between features and augments the space of features by inferring or creating extra features. ECO features employed standard genetic algorithm (GA) to detect series of image transforms with great differences. The construction of ECO feature involved different type of image transform such as Canny, Gaussian Blur, Median Blur, etc.

Chuang, Hwang, and Williams (2016) proposed a framework which consists of fully-unsupervised feature learning technique and error-resilient classifier.

Saitoh et al. (2015) proposed an approach with combines geometric features and Bag of visual words (BoVW) model.

Khotimah et al. (2015) proposed a method named Gray level co-occurance matrix (GLCM), a statistical method which uses spatial relationship of gray level image pixels to compute texture feature. The research is separated into 4 parts, i.e. training, segmentation, feature extraction and creating decision tree. The fish is separate from the background by using segmentation. The researchers classified the fish species using texture and shape of the Tuna fish. By using GLCM, the texture of the fish is represented with contrast, correlation, energy and homogeneity, inverse moment as well as entropy.

Chuang, Hwang, and Williams (2014) made comparison between 2 feature extraction methods, unsupervised method (scale-invariant part learning method (SIOPL)) and supervised method (part-aware feature (PAF)). For PAF, four experiments are conducted, experiment with specific features, experiment with specific features excluding tail texture, experiment with specific features excluding length ratio and experiment with specific features excluding eye texture. SIOPL focuses on three factor, fitness, separation and discrimination. The fish is divided to 4, 6, 8 and 10 parts and Histogram of oriented gradient (HOG) used to represent the body part appearance. SIOPL learn from training images and locate the useful part during testing. The feature extracted is used to train the classifier.

Table 2.3 shows the approach, number of datasets as well as the results for research works in fish recognition for the past 5 years.

Table 2.3: The approaches used for fish recognition.

Approach	Developer	Data Set	Accuracy	
CNN	Ding et al. (2017)	Image: 22437 Species: 4	7 layers	96.23%
			8 layers	96.51%
CNN	Jin and Liang (2017)	Image: 2120 Species: 10	Validation: 85.5% Testing: 85.08%	
RPN + R-CNN	Li et al. (2016)	Image: 16000 (training) 8000 (testing) Species: 12	82.7%	
PCANet & NIN	Sun et al. (2016)	Image: 22745 Species: 15	PCA (OR)	68.29%
			PCA (SR)	69.84%
			NIN (OR)	75.63%
			NIN (SR)	77.27%
Cascade Deep Network	Qin et al. (2016)	Image: 27370 Species: 23	DeepFish-SVM	98.23%
			DeepFish-SVM-aug	98.59%

			DeepFish-SVM-aug-scale	98.64%
			DeepFish-Softmax-aug	92.55%
			DeepFish-Softmax-aug-scale	98.49%
			DeepFish-CNN	98.57%
R-CNN	Xiu et al. (2015)	Image: 24272 Species: 12	81.4%	
Graph Embedding Discriminant Analysis	Hasija et al. (2017)	Image: 840 Species: 10	91.66%	
Hierarchy Classification	Huang (2016)	Image: 3179 Species: 10	Flat SVM	86.32%±5%
			Baseline Tree	88.08%±4%
			Automatically generated tree	90.01%±4%
SVM, Naïve Bayes	Kartika and Herumurti (2016)	Image: 281 Species:	SVM + Validation	97.15%
			SVM	94.50%

		9	Naïve Bayes + Validation	96.80%
			Naïve Bayes	97.92%
Evolution-COnstructed (ECO) Features	Zhang et al. (2016)	Image: 1049 Species: 8	98.9%	
Fully unsupervised feature learning technique + Error-resilient classifier	Chuang et al. (2016)	Image: 2195 Species: 7	Flat SVM	93.8%
			Hierarchy Full SVM	94.3%
			Hierarchy Partial SVM	98.4%
Geometric features + Bag of visual words (BoVW) model	Saitoh et al. (2015)	Image: 1620 Species: 129	-	
Gray level co-occurance matrix (GLCM)	Khotimah et al. (2015)	Image: 60 Species: 3	96.9%	
SIOPL, PAF	Chuang et al. (2014)	Image: 1325 Species: 7	PAF-all	93.65%
			PAF-noLenRatio	84.48%
			PAF-noTailTex	74.66%

			PAF-noEyeTex	81.76%
			SIOPL-4	98.94%
			SIOPL-6	99.32%
			SIOPL-8	99.92%
			SIOPL-10	98.43%
			SIOPL-8-NOpEC	94.15%

2.5 Convolutional Neural Network

Convolutional Neural Network (CNN) is one of the architecture which was enlightened by organisms' visual perception mechanism. It contains input layer, hidden layers and output layer. Hidden layers of conventional CNN is basically the combination of convolutional layers, pooling layers and fully-connected layers as shown in Figure 2.1 (Ding et al., 2017). The development of CNN emerged several types of layers.

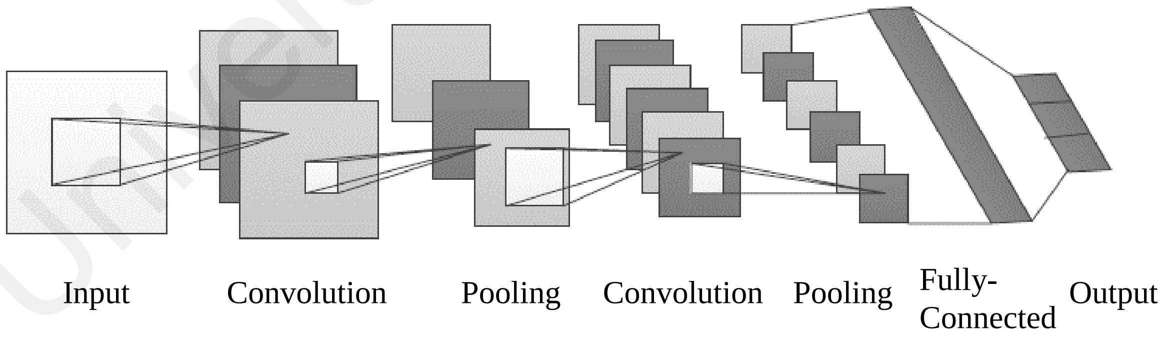


Figure 2.1: The structure of CNN. (Ding et al., 2017)

Convolutional layer is a weight-sharing network structure, it can be made up of several convolution kernels which are used to calculate distinct feature maps. There are five types of convolution layer which are tiled convolution, transposed convolution, dilated convolution, network in network and inception module. Pooling layer is used to reduce the spatial dimensions on a convolution neural network. Pooling can be classified into various types, such as L_p Pooling, Mixed Pooling, Stochastic Pooling, Spectral Pooling, Spatial Pyramid Pooling and Multi-Scale Orderless Pooling (Gu et al., 2018). Fully-connected layer combines all the distributed representations after ReLU layer and pooling layer, to form features with stronger capabilities. (Wu, 2017)

The advancement of technology, especially digital and electronics field produces computer with powerful computation ability. Powerful processor, larger memory capacity in computer memory (RAM) and larger memory capacity of graphic processor unit (GPU) enable human to build more complicated machine with human ability such as the ability of learning and classification. CNN shows improvement since the development of AlexNet in 2012. Table 2.4 shows the evolution of CNN into deeper architecture named deep learning.

Table 2.4: The evolution for CNN architecture

Architecture	**Explanation**
LeNet-5 (1998)	Convolutional Neural Network is developed by LeCun et al in 1998. LeNet-5 consists of 7 layers including 5 hidden layers.
AlexNet (2012)	AlexNet is developed by Alex Krizhevsky et al in ILSVRC 2012. The architecture is similar to LeNet-5, yet AlexNet is deeper with more stacked convolutional layers. The architecture consists of 8 layers including 5 convolution layers and 3 fully-connected layers. (Krizhevsky, Sutskever, & Hinton, 2012)
ZFNet (2013)	ZFNet is developed by the champions of ILSVRC 2013, Matthew D. Zeiler and Rob Fergus. The architecture consists of 9 layers including 7 hidden layers. (Fergus, 2013)
NIN (2013)	NIN is the acronym for Network in Network. NIN introduced 2 new concepts to CNN, Multi Linear Perceptrons (Mlpconv)

	and Global Average Pooling. NIN implemented a mini neural network, Mlpconv in convolution filter for better extraction and accuracy. Besides, fully-connected layer is replaced by activation maps. This model forms the foundation of Inception architecture. (Lin, Chen, & Yan, 2013)
GoogleNet/Inception (2014)	GoogleNet or also known as Inception is developed by the champion of ILSVRC 2014, Google. Inception consists of 22 layers deep CNN.
VGGNet (2014)	VGGNet is developed by Simonyan and Zisserman. VGGNet consists of 16 convolutional layers.
ResNet (2015)	ResNet is developed by Kaiming He et al. in ILSVRC 2015. ResNet can further classified to ResNet-34, ResNet-50, ResNet-100 and ResNet-150, at which the numbers indicate the number of layers for ResNet.

Based on Gu et al. (2018), there are improvements in CNN in six aspects after the success of AlexNet which are convolutional layer, pooling layer, activation function, loss function, regularization and optimization. Under each aspect, the layer further diversified into other layers as shown in Figure 2.2.

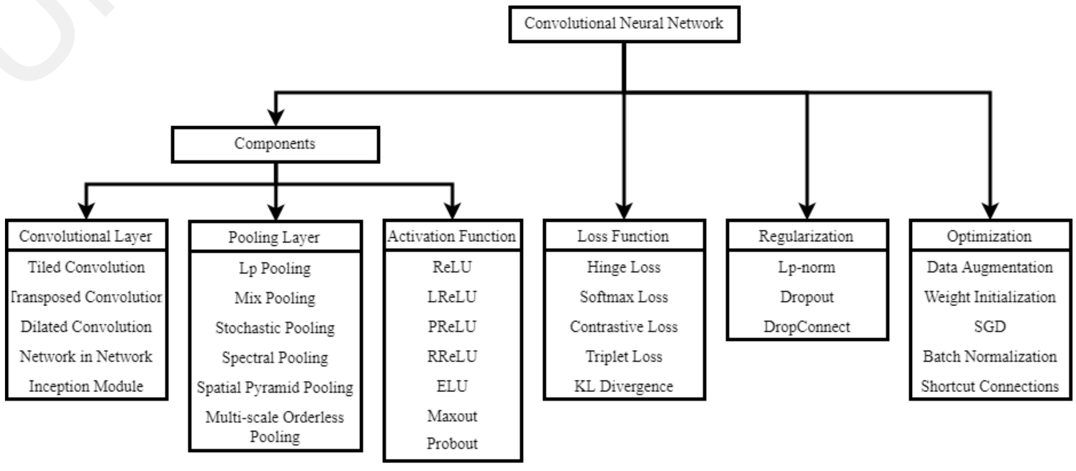


Figure 2.2 shows the improvement in CNN (Gu et al., 2018)

From the framework proposed by Jin and Liang (2017), the training data sets is acquired from ImageNet to train the convolutional neural network. The researchers trained 1000 categories with 1000 images per category. After Image de-nosing, the images are fed to train a CNN model. Pre-processed images are used to adjust the pre-trained neural network and finally examine the capability of the system.

Li et al. (2016) suggested a method which is the combination of Region Proposal Network (RPN) and Fast Regions with Convolutional Neural Network (R-CNN). RPN is modified and developed based on ZFNet. Xiu et al. (2015) applied R-CNN to a specific underwater environment.

Sun et al. (2016) used 2 deep learning methods to train the images, PCANet and deep learning to abstract features and apply linear SVM as classifier.

Framework proposed by Qin et al. (2016) is named as simple cascade deep network for fish recognition. The whole framework is made up of image pre-processing, 2 convolution layers with PCA filter, a non-linear layer with binary hashing, a feature pooling layer with block-wise histograms, spatial pyramid pooling layer and SVM as classifier.

Huang (2016) proposed a method named hierarchy classification for fish recognition. The hierarchy classification proposed is improved by adding 2 rules to reduce the error in average accuracy.

2.6 Summary

Based on the past reviews, different approaches were used in fish classification, most of them have achieved good results in fish classification. Different quality of fish images were used for classification. Most of the researchers preferred to use feature learning until lately, more CNN-based or deep learning fish recognition model are being used. For the researchers who tackle fish classification task with CNN or deep-learning preferred to use conventional convolutional layer with filter rather than pairing with nonlinear Multilayer

Perceptrons (Mlpconv). Based on Lin et al. (2013), the replacement of micro network can reduce the overfitting problems caused by fully-connected layer in conventional CNN and thus can increase the performance of the system. The target of this study is to implement NIN model for fish recognition. The performance of the model will be evaluate using F1-score.

CHAPTER 3: METHODLOGY

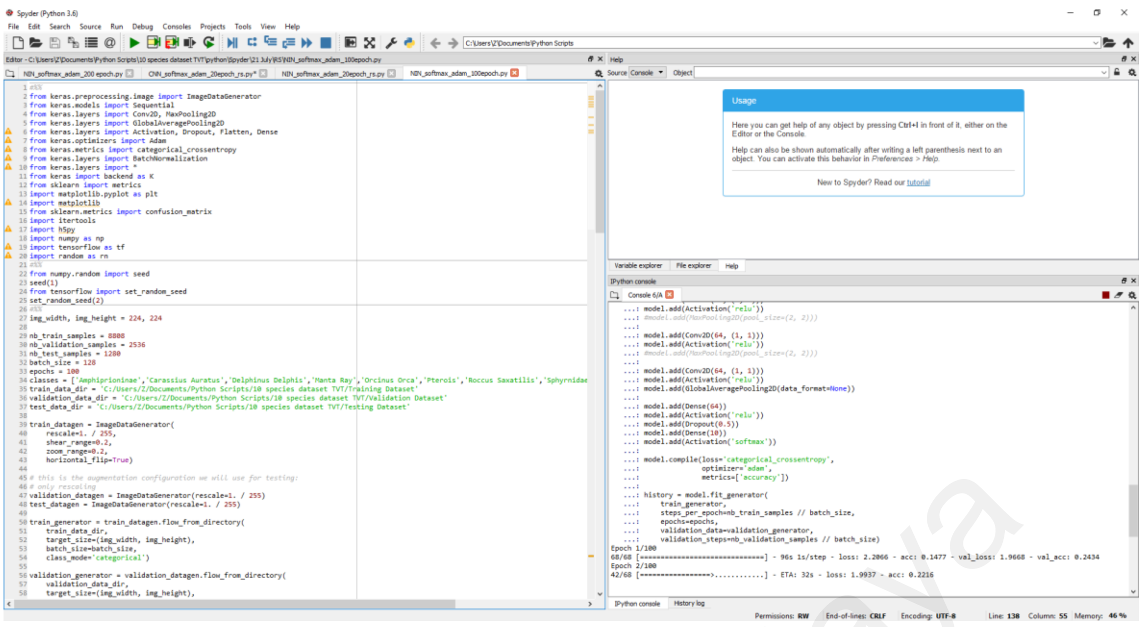
3.1 Introduction

The main purpose of this project is to frame a system which capable to classify fish species in images. From the past reviewed works, various method has been applied for classification and shows promising result in fish recognition. Researchers usually combine convolutional layers with other methods like PCA or deepen the convolutional layer, however not many researchers are implementing NIN model. NIN model modifies from ordinary convolution layer by replacing the filter with Mlpconv and using global average pooling layer instead of max pooling layer for the last pooling layer.

In this chapter, the recognition method and methodology to verify the proposed method are discussed in detail. The proposed method starts with general overview of the recognition process before further explaining the three phases involved in methodology including image acquisition, image pre-processing and image recognition.

3.2 Programming Language

Python Spyder is used as the programming environment to build the CNN model. Python is one of the well-known high-level programming language in coding for artificial intelligence. Python consists of extensive standard libraries and powerful datatypes, for example Keras, TensorFlow, numpy etc. These libraries enable user to implement or build AI models in more convenient way. Python has different integrated development environment (IDE) with different function such as Spyder and Pycharm. The interface of Python Spyder is shown in Figure 3.1.


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Z\Documents\Python Scripts
Editor - C:\Users\Z\Documents\Python Scripts\10 species dataset TVT\python\Spyder\21 July\RS\NIN_softmax_adam_100epoch.py
NIN_softmax_adam_200 epoch.py
CNN_softmax_adam_20epoch_rs.py*
NIN_softmax_adam_20epoch_rs.py
NIN_softmax_adam_100epoch.py
Help
Source Console
Object
Variable explorer File explorer Help
IPython console
Console 6/A
IPython console History log

```python
#%%
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import GlobalAveragePooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy
from keras.layers import BatchNormalization
from keras.layers import *
from keras import backend as K
from sklearn import metrics
import matplotlib.pyplot as plt
import matplotlib
from sklearn.metrics import confusion_matrix
import itertools
import h5py
import numpy as np
import tensorflow as tf
import random as rn
#%%
from numpy.random import seed
seed(1)
from tensorflow import set_random_seed
set_random_seed(2)
#%%
img_width, img_height = 224, 224

nb_train_samples = 8808
nb_validation_samples = 2536
nb_test_samples = 1280
batch_size = 128
epochs = 100
classes = ['Amphiprioninae','Carassius Auratus','Delphinus Delphis','Manta Ray','Orcinus Orca','Pterois','Roccus Saxatilis','Sphyrnidae
train_data_dir = 'C:/Users/Z/Documents/Python Scripts/10 species dataset TVT/Training Dataset'
validation_data_dir = 'C:/Users/Z/Documents/Python Scripts/10 species dataset TVT/Validation Dataset'
test_data_dir = 'C:/Users/Z/Documents/Python Scripts/10 species dataset TVT/Testing Dataset'

train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
validation_datagen = ImageDataGenerator(rescale=1. / 255)
test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
   ...: model.add(Activation('relu'))
   ...: #model.add(MaxPooling2D(pool_size=(2, 2)))
   ...:
   ...: model.add(Conv2D(64, (1, 1)))
   ...: model.add(Activation('relu'))
   ...: #model.add(MaxPooling2D(pool_size=(2, 2)))
   ...:
   ...: model.add(Conv2D(64, (1, 1)))
   ...: model.add(Activation('relu'))
   ...: model.add(GlobalAveragePooling2D(data_format=None))
   ...:
   ...: model.add(Dense(64))
   ...: model.add(Activation('relu'))
   ...: model.add(Dropout(0.5))
   ...: model.add(Dense(10))
   ...: model.add(Activation('softmax'))
   ...:
   ...: model.compile(loss='categorical_crossentropy',
   ...:               optimizer='adam',
   ...:               metrics=['accuracy'])
   ...:
   ...: history = model.fit_generator(
   ...:     train_generator,
   ...:     steps_per_epoch=nb_train_samples // batch_size,
   ...:     epochs=epochs,
   ...:     validation_data=validation_generator,
   ...:     validation_steps=nb_validation_samples // batch_size)
Epoch 1/100
68/68 [==============================] - 96s 1s/step - loss: 2.2066 - acc: 0.1477 - val_loss: 1.9668 - val_acc: 0.2434
Epoch 2/100
42/68 [=================>............] - ETA: 32s - loss: 1.9937 - acc: 0.2216
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 138 Column: 55 Memory: 46 %

Figure 3.1: The interface for Python Spyder

3.3 Variables

Before the building a CNN model, constant variables and measured variables need to be determined. The constant variables include the image number for samples, image's pixel, number of batches, number of epoch and random seed. The accuracy of the prediction for test samples is the measured variable for this project.

3.3.1 Constant Variables

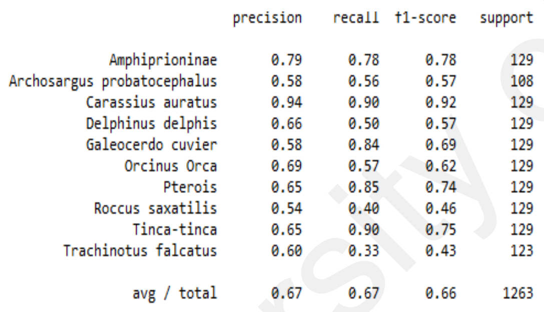
The image number is set to be 12579 images with 8804 images (70%) as training dataset, 2512 images (20%) for validation and 1263 images (10%) for testing. The images are standardizing to be 200×200 before feed to training.

Batch size is the number of training samples presented in a single iteration. Batch size giving impacts on the hyperparameters such as regularization factors. The batch size is set to be 128 samples. When the complete dataset finished forward and backward path once, it is called as epoch. Number of epoch can affect the loss and accuracy for both training and validation. Small number of epoch can lead to underfitting while too large number of epoch can lead to overfitting. The number of epoch is set to be 50 epochs, 200 epochs and 400 epochs for the NIN model used for fish recognition.

Random seed is set for result reproducibility. Due to the random nature of training algorithm, the result obtained differs each time retrain the model even with the same set of data. Random seed forcing the random initialization of the weights to be generated based on the seed set. There is no specific rule to set random seed, random seed is implemented to ensure the CNN model can be compared under the same manner as well as for result reproducibility.

3.3.2 Measured Variable

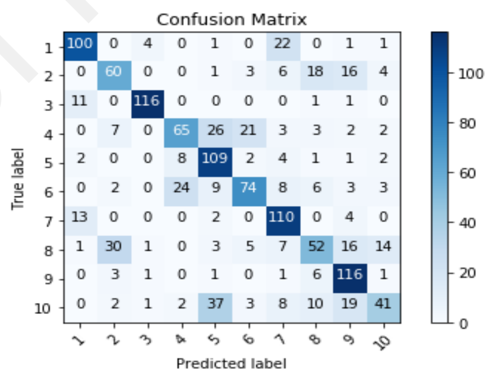
The measured variable in this project is mainly focused on the accuracy of prediction for the fish. The predictions are made using Keras prediction generator based on the training data. The accuracy of the prediction will be presented in percentage and confusion matrix as shown in Figure 3.2.



```
                           precision    recall  f1-score   support

             Amphiprioninae       0.79      0.78      0.78       129
Archosargus probatocephalus       0.58      0.56      0.57       108
          Carassius auratus       0.94      0.90      0.92       129
          Delphinus delphis       0.66      0.50      0.57       129
          Galeocerdo cuvier       0.58      0.84      0.69       129
               Orcinus Orca       0.69      0.57      0.62       129
                    Pterois       0.65      0.85      0.74       129
           Roccus saxatilis       0.54      0.40      0.46       129
                Tinca-tinca       0.65      0.90      0.75       129
        Trachinotus falcatus       0.60      0.33      0.43       123

                avg / total       0.67      0.67      0.66      1263
```



True label \ Predicted label	1	2	3	4	5	6	7	8	9	10
1	100	0	4	0	1	0	22	0	1	1
2	0	60	0	0	1	3	6	18	16	4
3	11	0	116	0	0	0	0	1	1	0
4	0	7	0	65	26	21	3	3	2	2
5	2	0	0	8	109	2	4	1	1	2
6	0	2	0	24	9	74	8	6	3	3
7	13	0	0	0	2	0	110	0	4	0
8	1	30	1	0	3	5	7	52	16	14
9	0	3	1	0	1	0	1	6	116	1
10	0	2	1	2	37	3	8	10	19	41

Figure 3.2: Percentage Table and Confusion Matrix.

In the percentage table, there are three types of percentage named precision, recall and f1-score. Precision, recall and f1-score are dependent on 4 parameters as shown in Table 3.1(Pedregosa, 2011).

Table 3.1: 4 parameters used for precision, recall and f1-score calculation (Pedregosa, 2011).

	Predicted Class		
Actual Class		**Class = Yes**	**Class = No**
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

where:	True Positive (TP)	= Both actual and predicted stated yes.
	True Negative (TN)	= Both actual and predicted stated no.

False Positive (FP) = Predicted yes but in actual no.
False Negative (FN) = Predicted no but in actual yes.

Precision is the ratio of True Positive to the total positive prediction as stated in Equation 3.1. Recall also known as sensitive, the ratio of the number of correctly predicted positive to the sum of True Positive and False Negative as stated in equation 3.2. F1-score is the average of Precision and Recall as stated in equation 3.3. F1-scores takes false prediction into account. The value of F1-score is used when class distribution is uneven.(Pedregosa, 2011)

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.2}$$

$$\text{F1} - \text{score} = \frac{2 \times (Recall \times Precision)}{Recall + Precision} \tag{3.3}$$

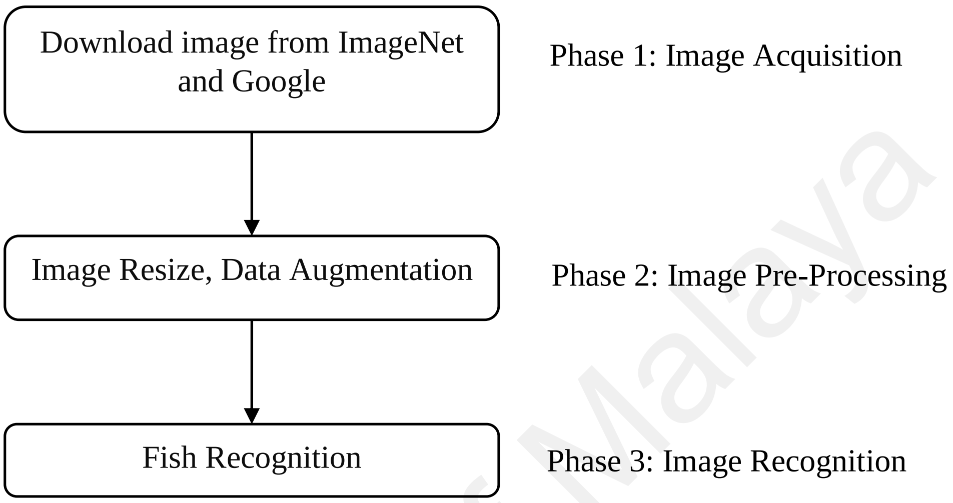
Confusion matrix is a table which shows the number of correct and incorrect prediction for the testing dataset. The number 1-10 on the predicted label and true label are the species of fish as listed in Table 3.2.

Table 3.2: The name of the fish for each number.

No	Species
1	Amphiprioninae
2	Archosargus probatocephalus
3	Carassius auratus
4	Delphinus delphis
5	Galeocerdo Cuvier
6	Orcinus Orca
7	Pterois
8	Roccus saxatilis
9	Tinca-tinca
10	Trachinotus falcatus

3.4 Recognition Method

Fish recognition involving 3 domain phases, which are image acquisition, image pre-pre-processing and image recognition. In order to train a convolutional neural network (CNN) model from scratch, large number of images required to feed the model. The images will obtain from ImageNet. Image pre-processing involves image segmentation and image normalization to increase the model's recognition rate. CNN will be implemented during image recognition phase. The flow of the process is illustrated in Figure 3.3.



Download image from ImageNet and Google
Phase 1: Image Acquisition
Image Resize, Data Augmentation
Phase 2: Image Pre-Processing
Fish Recognition
Phase 3: Image Recognition

Figure 3.3 shows the flow of the project.

3.5 Dataset

Image Data Sets for the project are downloaded from ImageNet. The dataset is downloaded from ImageNet and Google. It consists of 12579 RGB (Red, Green, Blue) JPEG images for 10 species of fishes. The images and classes of fish are listed in Table 3.3.

Table 3.3: Number of images for each fish species.

No	Images	Species	Number of Images
1		Amphiprioninae	1286

2		Archosargus probatocephalus	1077
3		Carassius auratus	1286
4		Delphinus delphis	1286
5		Galeocerdo Cuvier	1286
6		Orcinus Orca	1286
7		Pterois	1286

8		Roccus saxatilis	1286
9		Tinca-tinca	1286
10		Trachinotus falcatus	1214

The dataset is divided into 3 portion, 70 percent from each class is used as training dataset, 20 percent for validation and the remaining 10 percent used for testing. The number of images for each portion is shown in Table 3.4.

Table 3.4: The number of samples for training, validation and testing dataset.

No	Name	Training	Validation	Testing	Total
1	Amphiprioninae	900	257	129	1286
2	Archosargus probatocephalus	754	215	108	1077
3	Carassius auratus	900	257	129	1286
4	Delphinus delphis	900	257	129	1286
5	Galeocerdo Cuvier	900	257	129	1286
6	Orcinus Orca	900	257	129	1286
7	Pterois	900	257	129	1286
8	Roccus saxatilis	900	257	129	1286
9	Tinca-tinca	900	257	129	1286
10	Trachinotus falcatus	850	241	123	1214
Total		8804	2512	1263	12579

These three sets of data are stored under three main folders, training, validation and testing. Each of the folder consists of 10 sub-folders which filled with 10 different species of fish images respectively which as shown in Figure 3.4.

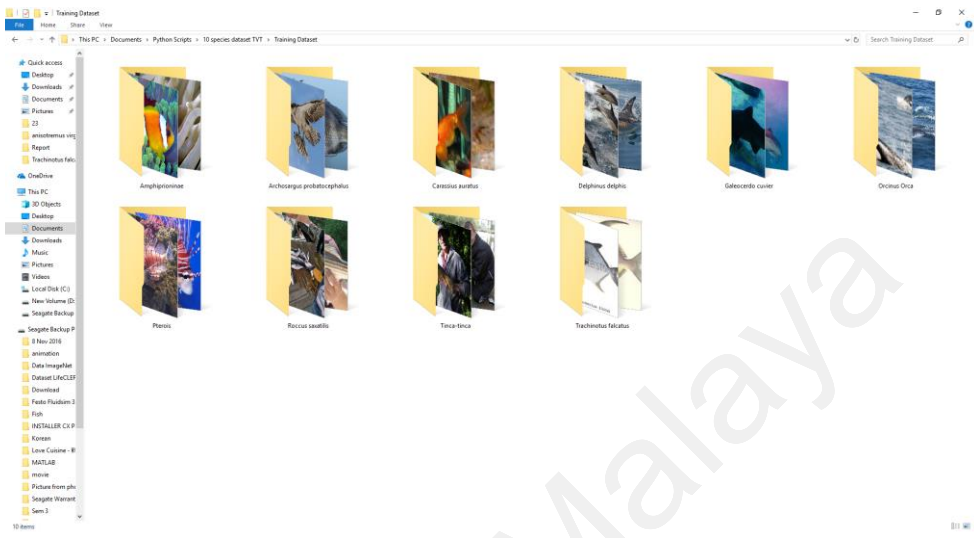


Figure 3.4 : Training Dataset Folders

3.6 Image Pre-processing

The image dataset has different sizes, hence images are resizing to 200×200 pixel before training. Besides, data augmentation also applies in this stage for better classification. Data augmentation creates new data by modifies existing data. Data augmentation includes several transformations, including images flipping, images rotation, images zooming, image cropping and images' colour varying. The purpose of implementing data augmentation is to reduce overfitting happens between training dataset and validation dataset. Images in training dataset folder will undergo rescaling, shear mapping, zooming and horizontal flip. The images for validation and testing will only undergo rescaling.

3.7 Convolutional Neural Network

CNN consists of 3 basic layers, convolutional layers, max-pooling layers and fully-connected layer. Convolutional layer is the combination of N×N neurons. The operation of convolutional layer similar to simple neural network as shown in Figure 3.5.

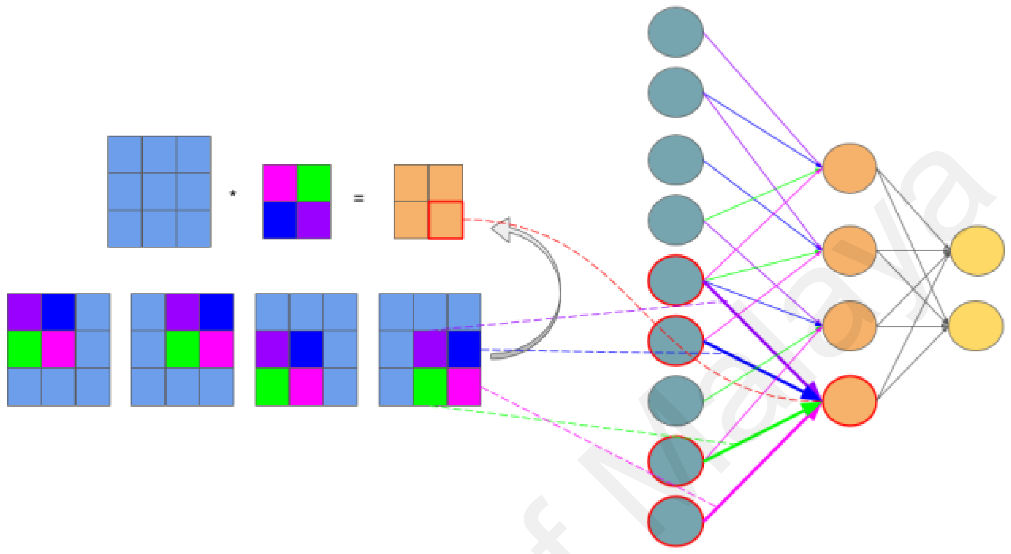


Figure 3.5: The relationship between convolution layer and simple neural network.

The equation for convolutional neural network forward propagation is stated in Equation 3.4 and Equation 3.5. Equation 3.4 shows the formula in convolving input vector at layer l with bias, while Equation 3.5 is the output vector at layer l. The $f(.)$ represents the activation function.

$$x_{i,j}^{l} = \sum_{m}\sum_{n} w_{m,n}^{l} o_{i+m,j+n}^{l-1} + b^{l} \tag{3.4}$$

$$o_{i,j}^{l} = f\left(x_{i,j}^{l}\right) \tag{3.5}$$

where: $l = l^{\text{th}}$ layer

$x =$ input with height × width which represented by i and j respectively.

$w =$ filter with dimension of $k_1 \times k_2$ which represented by m and n respectively.

$w_{m,n}^{l}$= weight matrix which connects neurons of layer l and the previous neuron layer.

b^{l} = bias for layer l

$o_{i,j}^{l}$ = output vector for layer l

Loss function is calculated using mean squared error as shown in Equation 3.6. Loss function use for the calculation of neurons' weight.

$$E = \frac{1}{2}\sum_{p}(t_p - y_p)^2 \tag{3.6}$$

where: y_p= actual output

t_p= target output

Backpropagation involved in weight updates, by applying chain rule as stated in equation 3.7.

$$\frac{\partial E}{\partial w_{m',n'}^{l}} = \sum_{i=0}^{H-k_1}\sum_{j=0}^{W-k_2} \delta_{i,j}^{l}\, o_{i+m',j+n'}^{l-1} \tag{3.7}$$

where: $\delta_{i,j}^{l} = \frac{\partial E}{\partial x_{i,j}^{l}}$, the measurement of how the change in pixel in input feature map affects loss function.

3.7.1 Network in Network Model

The CNN model used for this project is a Network-In-Network model. Conventional CNN model uses linear filter and nonlinear activation function such as sigmoid, tanh, etc. to scan and produce feature maps. As for NIN, it is similar to conventional CNN but it uses Mlpconv rather than linear filter as shown in Figure 3.6.

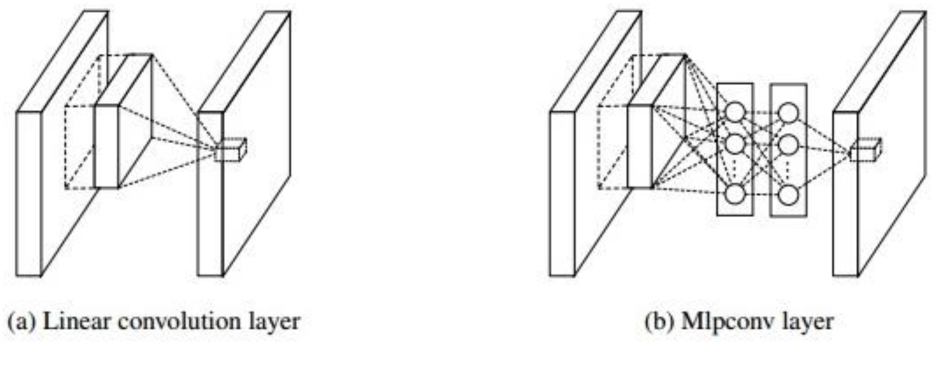


Figure 3.6: Conventional linear layer and Mlpconv layer

The calculation for Mlpconv is shown in Equation 3.8 and Equation 3.9.

$$f^1_{i,j,k_1} = max\left(w^1_{k_1}{}^T x_{i,j} + b_{k_1},0\right) \tag{3.8}$$

$$f^n_{i,j,k_n} = max\left(w^n_{k_n}{}^T f^{n-1}_{i,j} + b_{k_n},0\right) \tag{3.9}$$

where:	n	= number of layers in multilayer perceptron.
	f^1_{i,j,k_1}	= activation value of kth feature map at (i,j).
	w_k	= weight of the kth filter
	b_k	= bias of the kth filter

Pooling layer is a layer which reduce the number of connections between convolution layer to reduce the computation burden. A NIN model consists of 2 genre of pooling layer, which are max pooling layer and global average pooling layer (GAP). Max pooling layer separates the input images to non-overlapping rectangle depends on pooling size. For each of the partition part, the maximum number of each part is the output. For conventional CNN, the last layer of convolution layer is connected to fully-connected layer. However, this connection prone to overfitting. In NIN model, GAP is used to substitute fully-connected layer. GAP takes the average of the feature maps and fed to Softmax.

Rectified linear unit (ReLU) is the activation function which use in NIN model. ReLU is a half-rectified function. If the input is less than zero, output equals to zero. When the input is greater than zero, the value of output equals to the value of input. Its function is defined in Equation 3.10. The graph is shown in Figure 3.7.

$$f(x) = max(x,0) \tag{3.10}$$

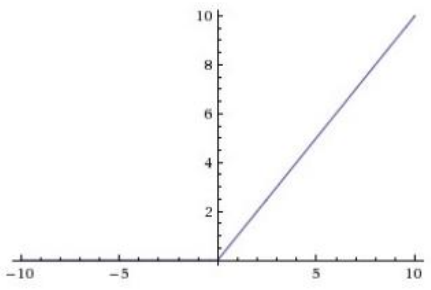


Figure 3.7: The graph for ReLU activation function.

Dropout is type of regularization method used to reduce overfitting. Overfitting occurs when the network is over-depending on any neurons, which memorize the features rather than learning. Overfitting is determined when training accuracy is greater than validation or testing accuracy as shown in Figure 3.8. Figure 3.9(b) illustrates the implementation of dropout which can reduce the network in interdependent learning among the neurons, only the weight for the included unit in Dropout will be updated. Dropout is needed when overfitting occurs. According to Baldi & Sadowski (2013), dropout=0.5 provides the best regularization, hence 0.5 is used as the value for dropout in this project if necessary.

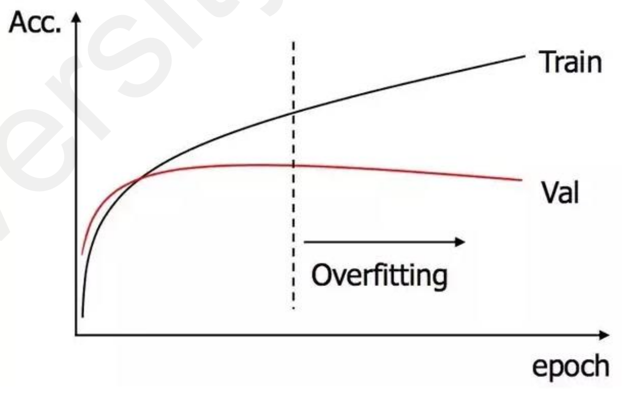


Figure 3.8: Overfitting graph

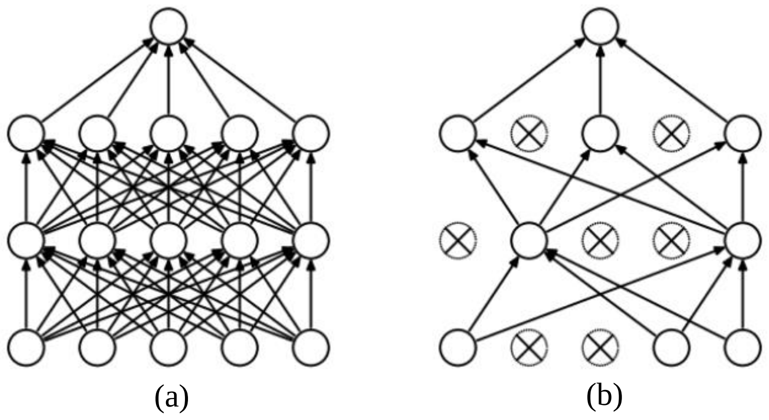


Figure 3.9: Standard neural network and neural network after dropout.

Softmax function computes the probability distribution of the feature over different features. The calculated probability is then used for identifying the classes of given input. Softmax is frequently used in multiclass classification. The equation for Softmax is stated in Equation 3.11.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad (3.11)$$

where: j = output units, j =1,2…,K.

z_j = $\beta_K . X_i$, β = bias, X = weight vector

z_k = $\beta_k . X_i$

Optimizer usually comes with back propagation. To obtain more accurate predictions, minimizing error of computed error is important. Error is the differences between the predicted outcome and actual outcome as show in Equation 3.12.

$$J(w) = p - \hat{p} \qquad (3.12)$$

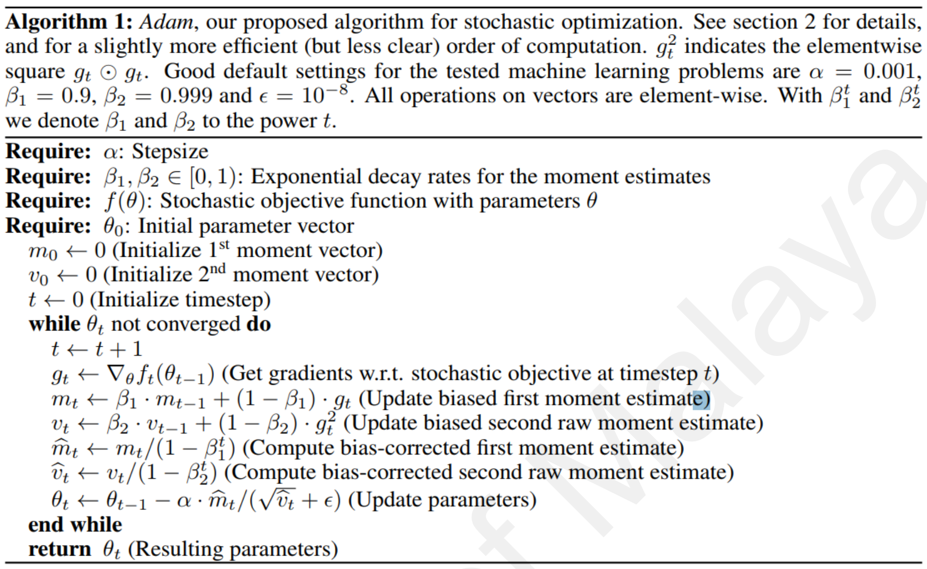
where: $J(w)$ = error

p = predicted response

$\hat{p}$ = actual response

Back propagation propagates current error back to previous layer and compute gradient to update the weight and bias in the neurons to reduce error. Optimization function is the function which used the gradients yielded by back propagation to modified weights. There are various optimizers can be used depends on the requirement such as

Adaptive Gradient Algorithm (AdaGrad), Adaptive Moment Estimation (Adam), Root Mean Square Propagation (RMSProp), etc. Adam is chosen as the optimizers for this project. Adam is well-known for its ability to achieve good result rapidly in deep learning. Figure 3.10 shows the Adam algorithm adapted from Kingma and Ba (2014).



Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

```
Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return $\theta_t$ (Resulting parameters)
```

Figure 3.10: The algorithm for ADAM optimizer. (Adapted from Adam: A method for Stochastic Optimization (Kingma & Ba, 2014))

Learning rate plays a very important role in improving the performance of the model. Learning rate is a hyper-parameter which control the speed in weight adjustment. The smaller the learning rate, the slower the gradient descent, less chance in missing local minima. The Equation 3.13 shows the formula for the relationship among weight, learning rate and gradient.

$$weight_{new} = weight_{previous} - learning\ rate * gradient \qquad (3.13)$$

Learning rate can be tuned based on the result from loss-epoch graph. Figure *3.11* shows the curve shape for different learning rate.

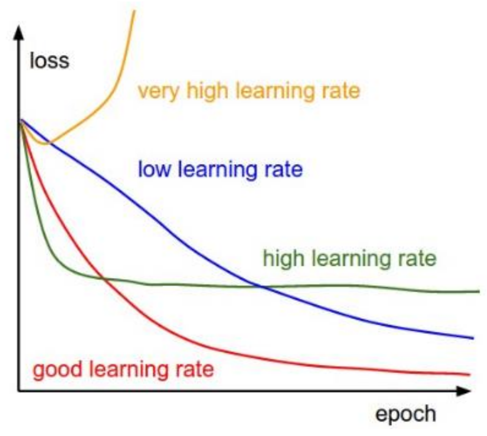


Figure 3.11: Graph for learning rate. Adapted from Andrej Karpathy (2018).

NIN consists of many layers of convolution layers and there are no specific or reference number for the filter used in NIN. The number of the filters for this project is set by referring the output neuron number from other researchers who did NIN model for their research. The NIN layers are listed in Table 3.5.

In total, there are three main layers, each main layer has the structure of C-A-C-A-C-A-P, first convolutional layer for each main layer is pair with two convolutional layers. The number of feature maps for 1st, 2nd and 3rd main layer is 32, 32 and 64, respectively. The 2nd and 3rd convolutional layer for each main layer has the same number as first convolutional layer, but with 1×1 size. The first and second pooling layer are using max-pooling while the last pooling layer is a GAP layer.

Table 3.5: The NIN layers

Layer	**Number of filter**	**Kernel size/Pooling Size/Dense Unit**	**Activation**
Convolution	32	4×4	-
Activation	-	-	relu
Convolution	32	1×1	-
Activation	-	-	relu
Convolution	32	1×1	-
Activation	-	-	relu
Max Pooling	-	2×2	-
Convolution	32	4×4	-
Activation	-	-	relu

Convolution	32	1×1	-
Activation	-	-	relu
Convolution	32	1×1	-
Activation	-	-	relu
Max Pooling	-	2×2	-
Convolution	64	4×4	-
Activation	-	-	relu
Convolution	64	1×1	-
Activation	-	-	relu
Convolution	64	1×1	-
Activation	-	-	relu
Global Average Pooling	-	-	-
Dense	-	10	-
Activation	-	-	Softmax

3.8 Summary

The method proposed is NIN model which will be used to train 8804 images. There will 2512 images for validation purposes and 1263 images will used to assess the performance of the classifier. The parameter for NIN model will be tuned throughout the training based on the model’s performance to obtain better result. All the results will be presented in Chapter 4.

CHAPTER 4: RESULT AND DISCUSSION

4.1 Introduction

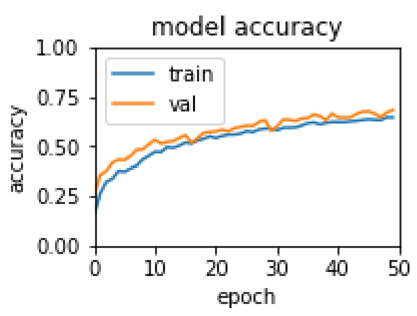
This chapter reports the results of the fish recognition. This study is expected to produce a high accuracy recognition system for fish.

There are total of 12579 fish images for 10 different classes of fish. 70 percent of the images used for training, 20 percent allocated as validation data and 10 percent for testing. The images are downloaded from ImageNet and Google.

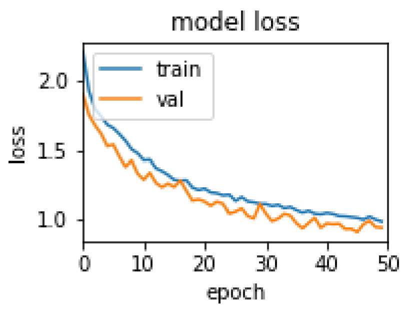
4.2 Results

To build a CNN from scratch, different parameters needs to be tuned including learning rate and number of epoch. The accuracy can vary depending on these parameter, to get a suitable learning rate, the training starts with the NIN model with the configuration stated in Table 3.5, learning rate of 0.001.

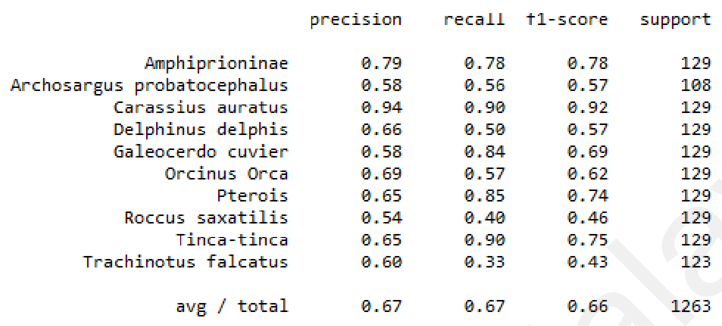
Figure 4.1 (a) and (b) shows the model accuracy and model loss for NIN model which tested for 50 epochs. From Figure 4.1 (c), the average F1-score for the prediction shows value of 0.66. Class *Carassius Auratus* obtained the highest F1-score with 0.92, the lowest is class *Trachinotus Falcatus* with F1-score of 0.43. According to the confusion matrix in Figure 4.1(d), there are quite a number of misclassifications for *Trachinotus Falcatus*, out of 123 images, the model misclassified 37 *Trachinotus Falcatus* images to be ofclass *Galeocerdo Cuvier.* This situation might be caused by insufficient training epoch.



(a)



(b)

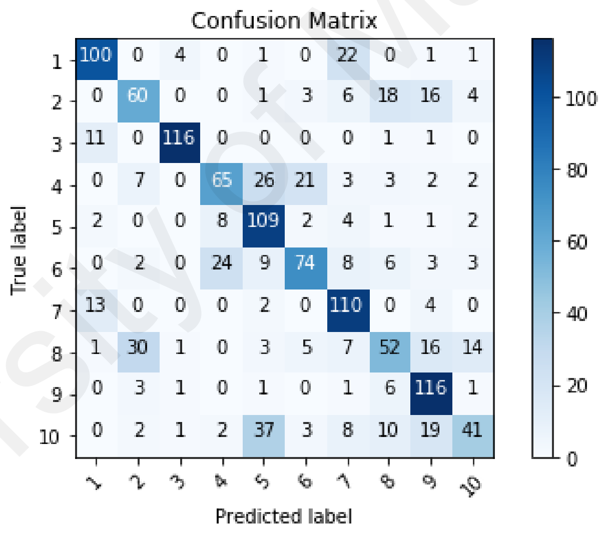


```
                              precision    recall  f1-score   support

               Amphiprioninae      0.79      0.78      0.78       129
Archosargus probatocephalus      0.58      0.56      0.57       108
            Carassius auratus      0.94      0.90      0.92       129
            Delphinus delphis      0.66      0.50      0.57       129
            Galeocerdo cuvier      0.58      0.84      0.69       129
                 Orcinus Orca      0.69      0.57      0.62       129
                      Pterois      0.65      0.85      0.74       129
            Roccus saxatilis      0.54      0.40      0.46       129
                  Tinca-tinca      0.65      0.90      0.75       129
         Trachinotus falcatus      0.60      0.33      0.43       123

                  avg / total      0.67      0.67      0.66      1263
```

(c)



Confusion Matrix

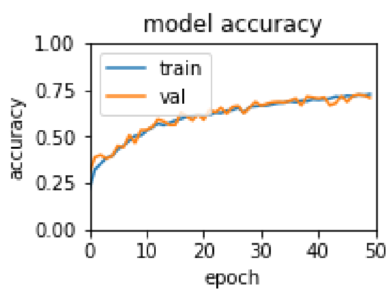
True label \ Predicted label	1	2	3	4	5	6	7	8	9	10
1	100	0	4	0	1	0	22	0	1	1
2	0	60	0	0	1	3	6	18	16	4
3	11	0	116	0	0	0	0	1	1	0
4	0	7	0	65	26	21	3	3	2	2
5	2	0	0	8	109	2	4	1	1	2
6	0	2	0	24	9	74	8	6	3	3
7	13	0	0	0	2	0	110	0	4	0
8	1	30	1	0	3	5	7	52	16	14
9	0	3	1	0	1	0	1	6	116	1
10	0	2	1	2	37	3	8	10	19	41

(d)

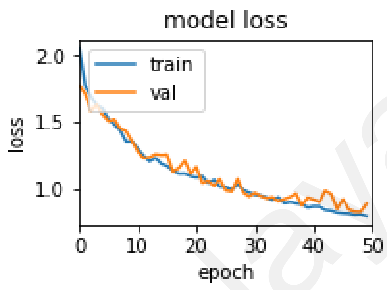
Figure 4.1: The result of NIN model for 50 epochs.

To improve the NIN model, the number of feature map for 3rd main layer is changed to 128 feature map and the dimension still remains the same. Figure 4.2 (a) and (b) shows the model accuracy and model loss for NIN model with learning rate of 0.001 which tested for 50 epochs.

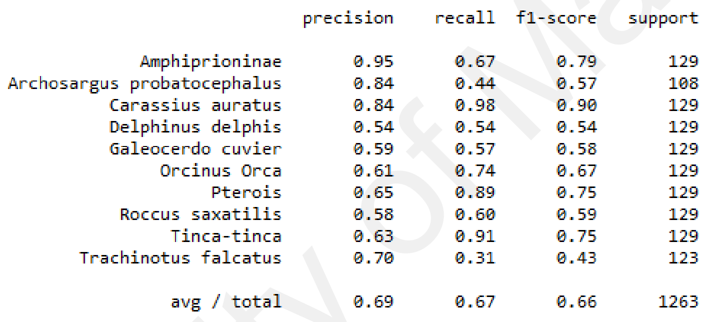
From Figure 4.2 (a) and (b), it is clearly shown that there is improvement after changing the number of feature map in NIN model. The graph shows training dataset and validation dataset has comparable performance. From Figure 4.2 (c), the average F1-score for the prediction shows value of 0.66. Class *Carassius Auratus* obtained the highest F1-score of 0.90. The lowest is class *Trachinotus Falcatus* with F1-score of 0.43. Observed from F1-score table and confusion matrix, the result is almost similar to the NIN before modification, the model is lacking of training.



(a)



(b)

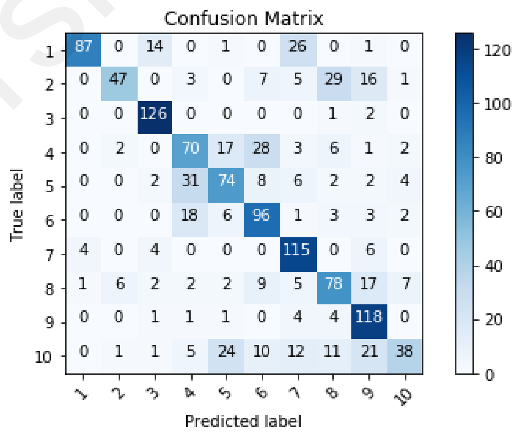


```
                             precision    recall  f1-score   support

             Amphiprioninae       0.95      0.67      0.79       129
Archosargus probatocephalus       0.84      0.44      0.57       108
          Carassius auratus       0.84      0.98      0.90       129
          Delphinus delphis       0.54      0.54      0.54       129
          Galeocerdo cuvier       0.59      0.57      0.58       129
                Orcinus Orca       0.61      0.74      0.67       129
                     Pterois       0.65      0.89      0.75       129
           Roccus saxatilis       0.58      0.60      0.59       129
                Tinca-tinca       0.63      0.91      0.75       129
        Trachinotus falcatus       0.70      0.31      0.43       123

                avg / total       0.69      0.67      0.66      1263
```

(c)

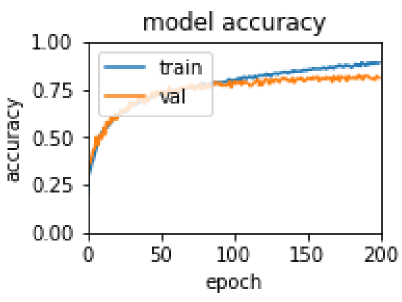


	1	2	3	4	5	6	7	8	9	10
1	87	0	14	0	1	0	26	0	1	0
2	0	47	0	3	0	7	5	29	16	1
3	0	0	126	0	0	0	0	1	2	0
4	0	2	0	70	17	28	3	6	1	2
5	0	0	2	31	74	8	6	2	2	4
6	0	0	0	18	6	96	1	3	3	2
7	4	0	4	0	0	0	115	0	6	0
8	1	6	2	2	2	9	5	78	17	7
9	0	0	1	1	1	0	4	4	118	0
10	0	1	1	5	24	10	12	11	21	38

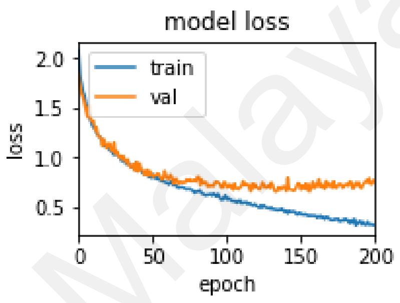
(d)

Figure 4.2: The result of NIN model (increased feature map) for 50 epochs.

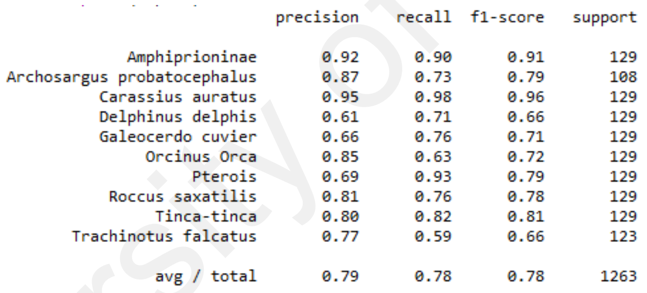
From Figure 4.3 (a) and (b), accuracy and loss for validation dataset converges from training dataset. On the other hand, the average F1-score shows improvement from 0.66 to 0.78. Class *Carassius Auratus* obtained the highest F1-score of 0.96. Class *Delphinus Delphis* and *Trachinotus Falcatus* have the lowest score of 0.66. From the confusion matrix, it is clearly shown that the model is prone to misclassify among *Delphinus Delphis*, *Galeocerdo Cuvier* and *Orcinus Orca*. This may due to these three species have closely identical features which confuse the model.



(a)



(b)

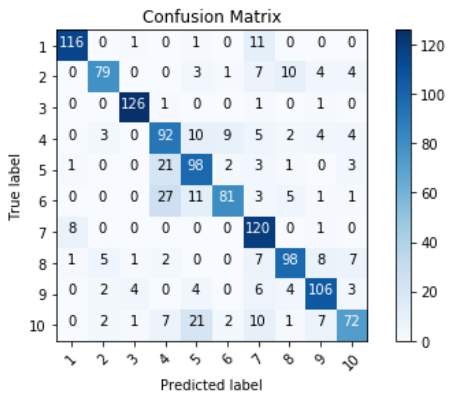


```
                            precision    recall  f1-score   support

            Amphiprioninae       0.92      0.90      0.91       129
Archosargus probatocephalus      0.87      0.73      0.79       108
          Carassius auratus       0.95      0.98      0.96       129
          Delphinus delphis       0.61      0.71      0.66       129
          Galeocerdo cuvier       0.66      0.76      0.71       129
               Orcinus Orca       0.85      0.63      0.72       129
                    Pterois       0.69      0.93      0.79       129
          Roccus saxatilis       0.81      0.76      0.78       129
                Tinca-tinca       0.80      0.82      0.81       129
       Trachinotus falcatus       0.77      0.59      0.66       123

                avg / total       0.79      0.78      0.78      1263
```

(c)

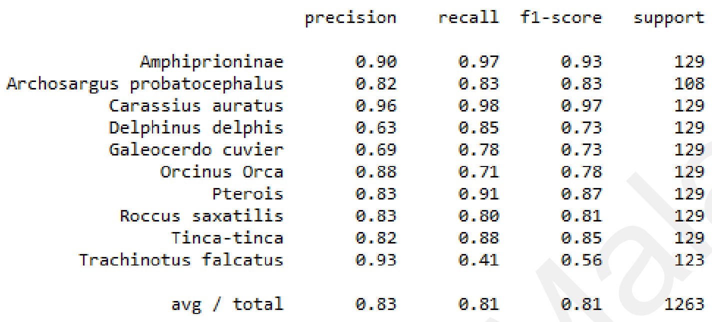


	1	2	3	4	5	6	7	8	9	10
1	116	0	1	0	1	0	11	0	0	0
2	0	79	0	0	3	1	7	10	4	4
3	0	0	126	1	0	0	1	0	1	0
4	0	3	0	92	10	9	5	2	4	4
5	1	0	0	21	98	2	3	1	0	3
6	0	0	0	27	11	81	3	5	1	1
7	8	0	0	0	0	0	120	0	1	0
8	1	5	1	2	0	0	7	98	8	7
9	0	2	4	0	4	0	6	4	106	3
10	0	2	1	7	21	2	10	1	7	72

(d)

Figure 4.3: The result of NIN model (increased feature map) for 200 epochs.

To improve the NIN model, the learning rate decreases from 0.001 to 0.0001. This model is trained for 200 epochs and 400 epochs, respectively. Figure 4.4 (a) and (b) shows result of prediction for NIN model with learning rate of 0.0001 which tested for 200 epochs. As compared to NIN model with learning rate of 0.001, the average F1-score shows higher accuracy of 0.81.

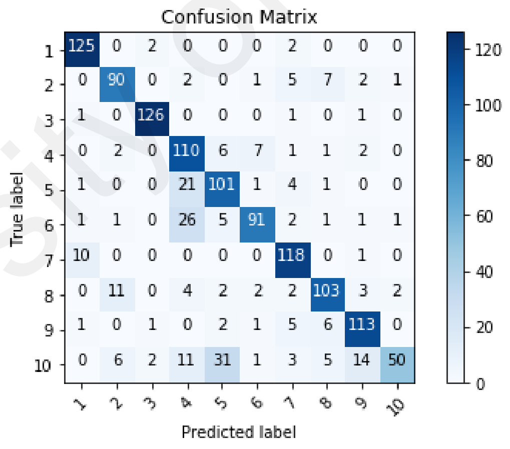


```
                             precision    recall  f1-score   support

              Amphiprioninae       0.90      0.97      0.93       129
Archosargus probatocephalus       0.82      0.83      0.83       108
            Carassius auratus       0.96      0.98      0.97       129
            Delphinus delphis       0.63      0.85      0.73       129
            Galeocerdo cuvier       0.69      0.78      0.73       129
                  Orcinus Orca       0.88      0.71      0.78       129
                       Pterois       0.83      0.91      0.87       129
            Roccus saxatilis       0.83      0.80      0.81       129
                   Tinca-tinca       0.82      0.88      0.85       129
          Trachinotus falcatus       0.93      0.41      0.56       123

                   avg / total       0.83      0.81      0.81      1263
```

(a)

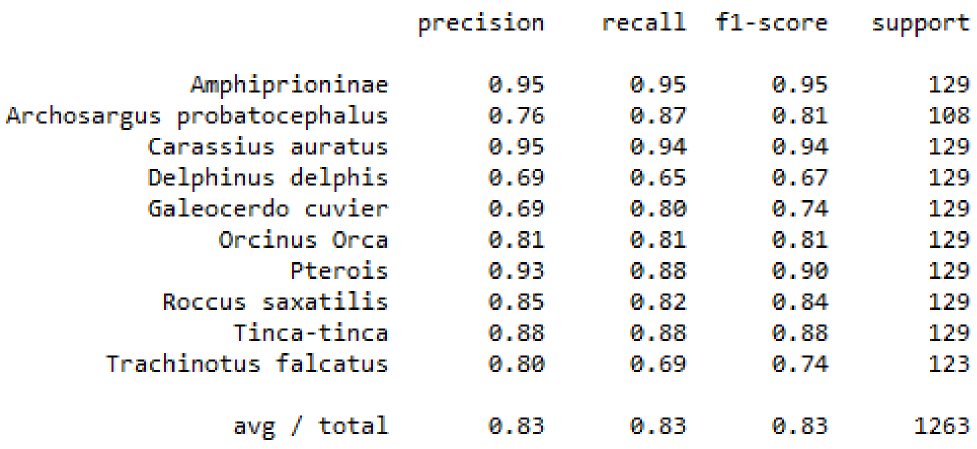


	1	2	3	4	5	6	7	8	9	10
1	125	0	2	0	0	0	2	0	0	0
2	0	90	0	2	0	1	5	7	2	1
3	1	0	126	0	0	0	1	0	1	0
4	0	2	0	110	6	7	1	1	2	0
5	1	0	0	21	101	1	4	1	0	0
6	1	1	0	26	5	91	2	1	1	1
7	10	0	0	0	0	0	118	0	1	0
8	0	11	0	4	2	2	2	103	3	2
9	1	0	1	0	2	1	5	6	113	0
10	0	6	2	11	31	1	3	5	14	50

(b)

Figure 4.4: The result of NIN model (increased feature map) with learning rate 0.0001 for 200 epochs.

Figure 4.5 (a) and (b) shows result of prediction for NIN model with learning rate of 0.0001 which tested for 400 epochs. The accuracy shows increment from 0.81 to 0.83.

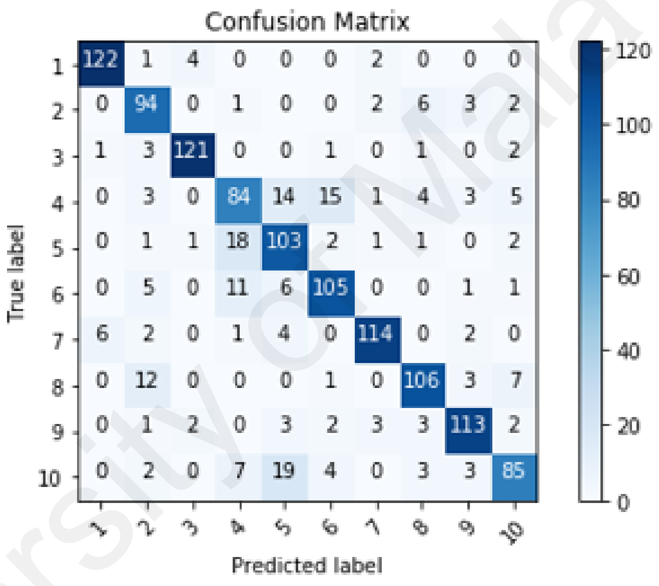


```
                            precision    recall  f1-score   support

             Amphiprioninae      0.95      0.95      0.95       129
Archosargus probatocephalus      0.76      0.87      0.81       108
           Carassius auratus      0.95      0.94      0.94       129
            Delphinus delphis      0.69      0.65      0.67       129
            Galeocerdo cuvier      0.69      0.80      0.74       129
                Orcinus Orca      0.81      0.81      0.81       129
                     Pterois      0.93      0.88      0.90       129
             Roccus saxatilis      0.85      0.82      0.84       129
                  Tinca-tinca      0.88      0.88      0.88       129
         Trachinotus falcatus      0.80      0.69      0.74       123

                 avg / total      0.83      0.83      0.83      1263
```

(a)



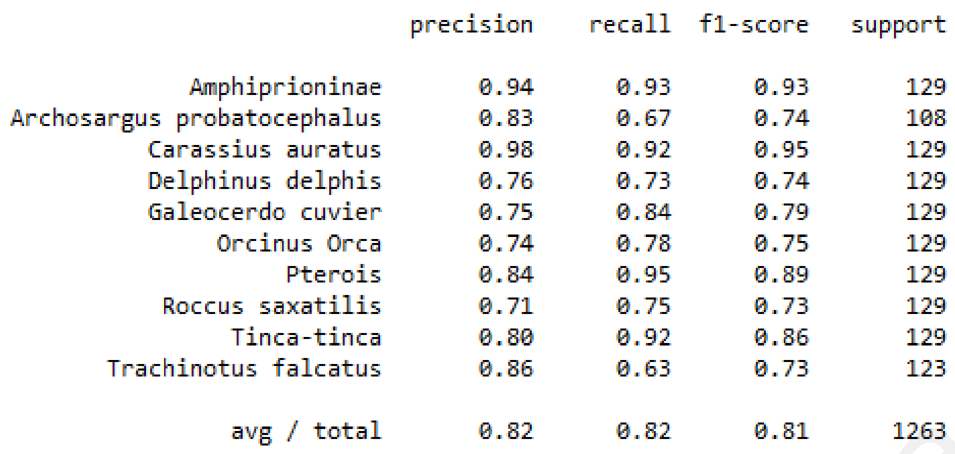
Confusion Matrix

True label \ Predicted label	1	2	3	4	5	6	7	8	9	10
1	122	1	4	0	0	0	2	0	0	0
2	0	94	0	1	0	0	2	6	3	2
3	1	3	121	0	0	1	0	1	0	2
4	0	3	0	84	14	15	1	4	3	5
5	0	1	1	18	103	2	1	1	0	2
6	0	5	0	11	6	105	0	0	1	1
7	6	2	0	1	4	0	114	0	2	0
8	0	12	0	0	0	1	0	106	3	7
9	0	1	2	0	3	2	3	3	113	2
10	0	2	0	7	19	4	0	3	3	85

(b)

Figure 4.5: The result of NIN model (increased feature map) with learning rate 0.0001 for 400 epochs.

The dataset is used to train using a CNN model with learning rate of 0.0001. The implemented CNN model has the architecture of 32C-P-32C-P-64C-P-FC. The dimension for filter is 3×3, the pool size for maxing pooling is 2×2. From Figure 4.6 (a), the average F1-score is 0.81, slightly less than NIN. For the prediction for CNN model, the lowest score is 0.73, while for NIN model, the lowest score is 0.67. Class *Delphinus Delphis* obtained low score for both model. It often misclassified as class *Galeocerdo Cuvier* and *Orcinus Orca* due to similar colour and appearance.

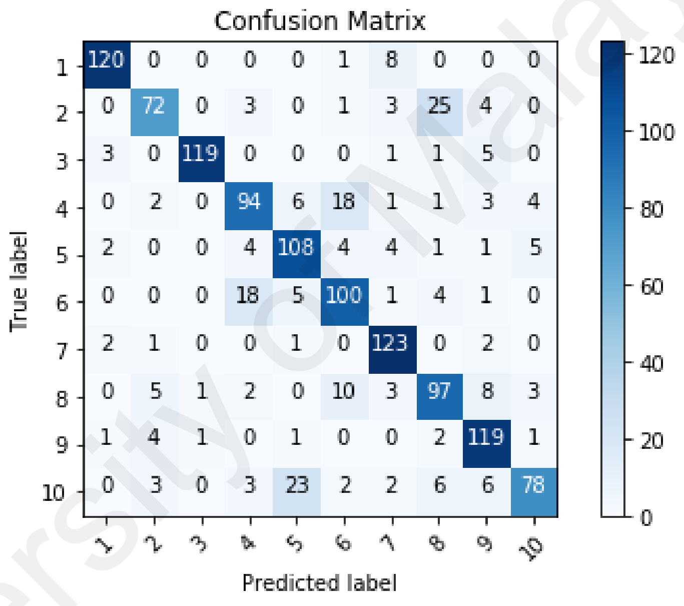


```
                            precision    recall  f1-score   support

           Amphiprioninae       0.94      0.93      0.93       129
Archosargus probatocephalus       0.83      0.67      0.74       108
          Carassius auratus       0.98      0.92      0.95       129
          Delphinus delphis       0.76      0.73      0.74       129
          Galeocerdo cuvier       0.75      0.84      0.79       129
               Orcinus Orca       0.74      0.78      0.75       129
                    Pterois       0.84      0.95      0.89       129
           Roccus saxatilis       0.71      0.75      0.73       129
                Tinca-tinca       0.80      0.92      0.86       129
        Trachinotus falcatus       0.86      0.63      0.73       123

                avg / total       0.82      0.82      0.81      1263
```

(a)

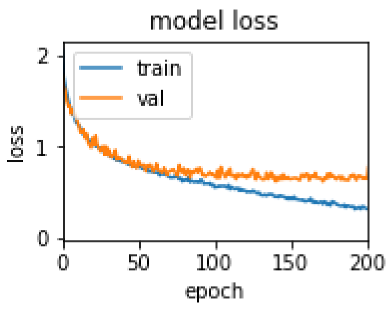


	1	2	3	4	5	6	7	8	9	10
1	120	0	0	0	0	1	8	0	0	0
2	0	72	0	3	0	1	3	25	4	0
3	3	0	119	0	0	0	1	1	5	0
4	0	2	0	94	6	18	1	1	3	4
5	2	0	0	4	108	4	4	1	1	5
6	0	0	0	18	5	100	1	4	1	0
7	2	1	0	0	1	0	123	0	2	0
8	0	5	1	2	0	10	3	97	8	3
9	1	4	1	0	1	0	0	2	119	1
10	0	3	0	3	23	2	2	6	6	78

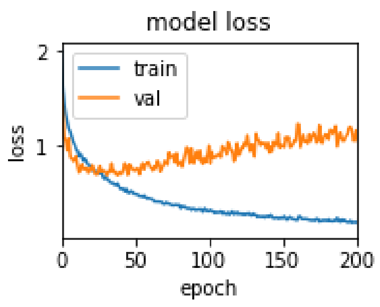
(b)

Figure 4.6: The result of CNN model for 400 epochs.

Figure 4.7 (a) and (b) shows the loss for NIN and CNN, respectively. Both model is having overfitting issue. As compare to NIN model, CNN model has severe overfitting issue due to the existence of fully-connected layer (Lin et al., 2013). These situations can be reduced by applying regularizers, dropout, batch normalization or early stopping.



(a)



(b)

Figure 4.7: The training losses and validation losses for NIN and CNN model.

Table 4.1 shows the time-consumed and average F1-score for the NIN model. The prediction's accuracy shows improvement when the model is trained with more epochs.

Table 4.1: The time cost and average F1-score for NIN model.

Model	Learning Rate	Number of Epoch	Time Costs	Average F1-score
NIN	0.001	50	65 min	0.66
NIN (increased feature map)	0.001	50	65 min	0.66
NIN (increased feature map)	0.001	200	4 hours 17 min	0.78
NIN (increased feature map)	0.0001	200	4 hours 23 min	0.81
NIN (increased feature map)	0.0001	400	8 hours 33 min	0.83
CNN	0.0001	400	8 hours 40 min	0.81

4.3 Summary

From the evidence collected from F1-score, the performance of NIN can be further improved by tuning parameter and increasing dataset especially for fish species which have identical features. Sometimes, the wrong prediction made by the model is due to the misleading of the position of the fish. For example, making prediction from dorsal fin in images. Insufficient information in testing data can lead to wrong prediction. This issue can be minimized and accuracy can be increased if a larger training dataset is being used.

CHAPTER 5: CONCLUSION

5.1 Conclusion

To train a CNN model from scratch, image dataset is very important. With large amount of dataset, a CNN can have a better performance. Different parameter as well play important roles in increasing the accuracy and enhance the model’s performance. For example, learning rate, batch size, image pixels, choice of optimizer, number and dimension of kernel, etc.

In this project, fish recognition model is designed using NIN model. 10 species of fish are used for classification. The NIN model with different feature map numbers is used to be trained with 0.001 learning rate. Then, the model is compared with NIN with learning rate of 0.0001. Softmax and Adam are used for classifier and optimizer, respectively. The NIN model with more feature maps which trained with learning rate of 0.0001 reaches the average accuracy of 83% after 400 epochs of training. After being compared with a conventional CNN, NIN less prone to overfitting.

5.2 Recommendation and Future Work

Even the model can used for fish recognition, yet there are still many aspects that need to be improved. Future work mainly focuses on optimizing and refining the model by feeding more images to train NIN model and tune the parameters. In addition, NIN can further developed incorporating with other AI approaches such as genetic algorithm, optimization etc.

REFERENCES

1. Alginahi, Y. (2010). Preprocessing Techniques in Character Recognition. In M. Mori (Ed.), Character Recognition. Retrieved from https://www.intechopen.com/books/character-recognition/preprocessing-techniques-in-character-recognition. doi:10.5772/9776
2. Andrej Karpathy, J. J. (2018). CS231n Convolutional Neural Networks for Visual Recognition.
3. Chitradevi, B., & Srimathi, P. (2014). An overview on image processing techniques. *International Journal of Innovative Research in Computer, 2*(11), 6466-6472.
4. Chouiten, M. (2013). *Underwater Real-Time Fish Recognition by Image Processing*.
5. Chuang, M. C., Hwang, J. N., & Williams, K. (2014, 24-24 Aug. 2014). *Supervised and Unsupervised Feature Extraction Methods for Underwater Fish Species Recognition.* Paper presented at the 2014 ICPR Workshop on Computer Vision for Analysis of Underwater Imagery.
6. Chuang, M. C., Hwang, J. N., & Williams, K. (2016). A Feature Learning and Object Recognition Framework for Underwater Fish Images. *IEEE Transactions on Image Processing, 25*(4), 1862-1872. doi:10.1109/TIP.2016.2535342
7. Copeland, M. (2016, 22 August 2016). What's the Difference Between Deep Learning Training and Inference? Retrieved from https://blogs.nvidia.com/blog/2016/08/22/difference-deep-learning-training-inference-ai/
8. Ding, G., Song, Y., Guo, J., Feng, C., Li, G., He, B., & Yan, T. (2017, 18-21 Sept. 2017). *Fish recognition using convolutional neural network.* Paper presented at the OCEANS 2017 - Anchorage.

9. Fergus, M. D. Z. R. (2013). Visualizing and Understanding Convolutional Networks. *CoRR, abs/1311.2901*.
10. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., . . . Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition, 77,* 354-377. doi:https://doi.org/10.1016/j.patcog.2017.10.013
11. Hardesty, L. (2015, 21 January 2015). Optimizing optimization algorithms. Retrieved from http://news.mit.edu/2015/optimizing-optimization-algorithms-0121
12. Hasija, S., Buragohain, M. J., & Indu, S. (2017, 17-19 Feb. 2017). *Fish Species Classification Using Graph Embedding Discriminant Analysis.* Paper presented at the 2017 International Conference on Machine Vision and Information Technology (CMVIT).
13. Haykin, S. (2009). *Neural Networks and Learning Machines*: Pearson.
14. Hsiao, Y.-H., Chen, C.-C., Lin, S.-I., & Lin, F.-P. (2014). Real-world underwater fish recognition and identification, using sparse representation. *Ecological Informatics, 23*, 13-21. doi:https://doi.org/10.1016/j.ecoinf.2013.10.002
15. Huang, P. (2016). *Hierarchical Classification for Live Fish Recognition*.
16. Jin, L., & Liang, H. (2017, 19-22 June 2017). *Deep learning for underwater image recognition in small sample size situations.* Paper presented at the OCEANS 2017 - Aberdeen.
17. Kartika, D. S. Y., & Herumurti, D. (2016, 12-12 Oct. 2016). *Koi fish classification based on HSV color space.* Paper presented at the 2016 International Conference on Information & Communication Technology and Systems (ICTS).
18. Khotimah, W. N., Arifin, A. Z., Yuniarti, A., Wijaya, A. Y., Navastara, D. A., & Kalbuadi, M. A. (2015, 5-7 Oct. 2015). *Tuna fish classification using decision tree algorithm and image processing method.* Paper presented at the 2015

International Conference on Computer, Control, Informatics and its Applications (IC3INA).

19. Kingma, D., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization*.
20. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet classification with deep convolutional neural networks*. Paper presented at the Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, Lake Tahoe, Nevada.
21. Li, X., Shang, M., Hao, J., & Yang, Z. (2016, 10-13 April 2016). *Accelerating fish detection and recognition by sharing CNNs with objectness learning.* Paper presented at the OCEANS 2016 - Shanghai.
22. Lin, M., Chen, Q., & Yan, S. (2013). Network In Network. *abs/1312.4400*.
23. Luo, S., Li, X., Wang, D., Li, J., & Sun, C. (2015, 12-14 Dec. 2015). *Automatic Fish Recognition and Counting in Video Footage of Fishery Operations.* Paper presented at the 2015 International Conference on Computational Intelligence and Communication Networks (CICN).
24. Pedregosa, F. V., G. , Gramfort, A. , Michel, V. , Thirion, B. , Grisel, O. , Blondel, M. , Prettenhofer, P. , Weiss, R. , Dubourg, V. , Vanderplas, J. , Passos, A. , Cournapeau, D. , Brucher, M. , Perrot, M. , Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research.* Retrieved from http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html
25. Qin, H., Li, X., Liang, J., Peng, Y., & Zhang, C. (2016). DeepFish: Accurate underwater live fish recognition with a deep architecture. *Neurocomputing, 187*, 49-58. doi:https://doi.org/10.1016/j.neucom.2015.10.122

26. Saitoh, T., Shibata, T., & Miyazono, T. (2015, 13-15 Nov. 2015). *Image-based fish recognition.* Paper presented at the 2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR).

27. Smith, S. S. (Ed.) (2003). Trident Press International.

28. Sun, X., Shi, J., Dong, J., & Wang, X. (2016, 15-17 Oct. 2016). *Fish recognition from low-resolution underwater images.* Paper presented at the 2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI).

29. Wu, J. (2017). *Introduction to Convolutional Neural Networks*: National Key Lab for Novel Software Technology,Nanjing University, China.

30. Xiu, L., Min, S., Qin, H., & Liansheng, C. (2015, 19-22 Oct. 2015). *Fast accurate fish detection and recognition of underwater images with Fast R-CNN.* Paper presented at the OCEANS 2015 - MTS/IEEE Washington.

31. Zhang, D., Lee, D.-J., Zhang, M., Tippetts, B. J., & Lillywhite, K. D. (2016). Object recognition algorithm for the automatic identification and removal of invasive fish. *Biosystems Engineering, 145*, 65-75. doi:https://doi.org/10.1016/j.biosystemseng.2016.02.013

APPENDIX A

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import GlobalAveragePooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.optimizers import Adam, RMSprop
from keras.metrics import categorical_crossentropy
from keras.layers import BatchNormalization
from keras.layers import *
from keras import backend as K
from sklearn import metrics
import matplotlib.pyplot as plt
import matplotlib
from sklearn.metrics import confusion_matrix
import itertools
import numpy as np

#set seed number
from numpy.random import seed
seed(1)
from tensorflow import set_random_seed
set_random_seed(2)

#define values
#define values
img_width, img_height = 200, 200
nb_train_samples = 8804
nb_validation_samples = 2512
nb_test_samples = 1263
batch_size = 128
epochs = 400

#locate images
train_data_dir = 'C:/Users/Z/Documents/Python Scripts/10 species dataset TVT/Training Dataset'
validation_data_dir = 'C:/Users/Z/Documents/Python Scripts/10 species dataset TVT/Validation Dataset'
test_data_dir = 'C:/Users/Z/Documents/Python Scripts/10 species dataset TVT/Testing Dataset'

#set format for Tensorflow and Theano
if K.image_data_format() == 'channels_first':
   input_shape = (3, img_width, img_height)
else:
   input_shape = (img_width, img_height, 3)


#Model
model = Sequential()
model.add(Conv2D(32, (4, 4), input_shape=input_shape))
model.add(Activation('relu'))

model.add(Conv2D(32, (1, 1)))
model.add(Activation('relu'))

model.add(Conv2D(32, (1, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (4, 4)))
model.add(Activation('relu'))

model.add(Conv2D(32, (1, 1)))
model.add(Activation('relu'))

model.add(Conv2D(32, (1, 1)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (4, 4)))
model.add(Activation('relu'))

model.add(Conv2D(128, (1, 1)))
model.add(Activation('relu'))

model.add(Conv2D(128, (1, 1)))
model.add(Activation('relu'))
model.add(GlobalAveragePooling2D(data_format=None))
model.add(Dense(10))
model.add(Activation('softmax'))

#Insert learning rate and compile model
learning_rate = 0.0001
decay_rate = learning_rate / epochs
adam = Adam(lr=learning_rate, decay=decay_rate)
model.compile(loss='categorical_crossentropy',
            optimizer='adam',
            metrics=['accuracy'])


#Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

#Rescaling for both validation and testing data
validation_datagen = ImageDataGenerator(rescale=1. / 255)
test_datagen = ImageDataGenerator(rescale=1. / 255)

#Read images from file
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical', shuffle=False)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical', shuffle=False)

#fits data generator to data
history = model.fit_generator(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size)

#Plot graph for accuracy and loss
plt.subplot(2,2,1)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
x1,x2,y1,y2 = plt.axis()
plt.axis((0,200,0,1))
plt.legend(['train', 'val'], loc='upper left')

plt.subplot(2,2,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
x1,x2,y1,y2 = plt.axis()
plt.axis((0,200,y1,y2))
plt.legend(['train', 'val'], loc='upper left')

#Make prediction
test_steps_per_epoch = np.math.ceil(test_generator.samples / test_generator.batch_size)
predictions = model.predict_generator(test_generator, steps=test_steps_per_epoch)
```

```
#Produce Precision, recall and f1-score table
predicted_classes = np.argmax(predictions, axis=1)
true_classes=test_generator.classes
class_labels = list(test_generator.class_indices.keys())
report = metrics.classification_report(true_classes, predicted_classes, target_names=class_labels)
print(report)

#Confusion matrix
cm = confusion_matrix(true_classes, predicted_classes)
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j]),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

cm_plot_labels = ['1','2','3','4','5','6','7','8','9','10']
plot_confusion_matrix(cm, cm_plot_labels,title='Confusion Matrix')

#save model
model.save('CNN_200_0.0001_3232128_m.h5')

#save weight
model.save_weights('C:/Users/Z/Documents/Python Scripts/10 species dataset TVT/Model/spyder/CNN_200_0.0001_3232128.h5')
```