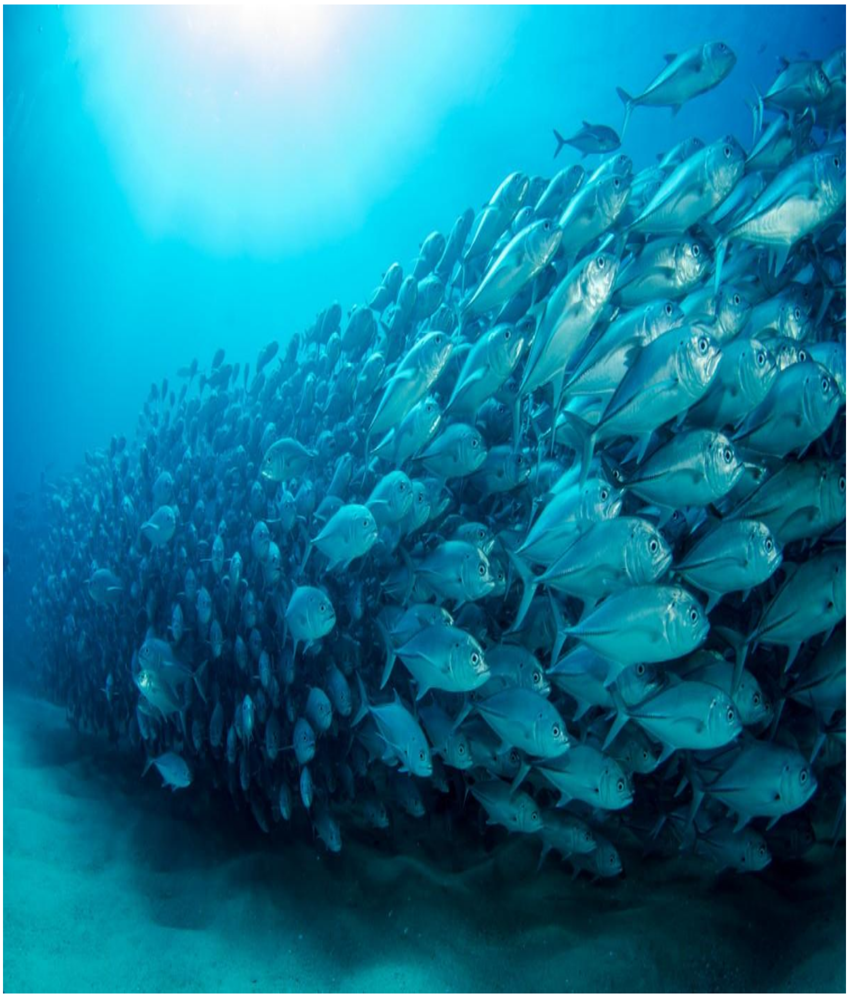

FISH CLASSIFICATION

Convolutional Neural Networks and Deep Learning

AUEB
MSc Business Analytics
Rapanou, Souliou, Katsaris, Polyzogopoulos

Big Data Content Analytics PT(2017 – 2018)

Contents

List of Tables

List of Figures

1. Description

For centuries, the expanse of the seas was seen as an inexhaustible source of fish and seafood. Today we know that more than ¾ of fish stocks are fully fished or even over fished. Unsustainable fishing endangers species and their habitants. [B1]

On top of the over - fishing comes the rapid releases of greenhouse gas emissions, that threaten fish life on a faster rate than man does. [B2]

The world is on the brink of a massive extinction event, according to the United Nations and entire species, such as the Pacific Bluefin tuna, the swordfish and others, are highly endangered and are at an all- time low [B1]

The latest global coral reef assessment estimates that 19 percent of the world's coral reefs are dead. Their major threats include warming sea-surface temperatures and expanding seawater acidification.

*The **International Union for the Conservation of Nature (IUCN)** declared in October that 38 percent of the 44,838 species it studied across the world are threatened with extinction.*

Zooxanthellae, the tiny organisms that give coral reefs their vibrant colors, are emigrating from their hosts in massive numbers as oceans heat up, killing themselves and the coral they leave behind - a process known as coral bleaching.

For all the above mentioned it is critical to monitor and detect the fish that are about to extinct or protect other species.

Monitoring natural populations is often a necessary step to establish the conservation status of species and to help improve management decisions. Nevertheless, many monitoring programs do not effectively address primary sources of variability in monitoring data, which ultimately may limit the utility of monitoring in identifying declines and improving management.

Nowadays, there have been already developed several applications for fish recognition and classification. For example, Fish Verify is an application that uses the latest in image recognition and artificial intelligence to identify fish species and all in one quick motion. Also provides all the regulations regarding the fishing in several areas. [B3]

"FishVerify is your Digital Regulation and Identification Guide.

Quickly learn if your catch is in season, how many you can keep the size limit, edibility, and much more."

It is actually feasible to build and develop models and applications to provide strong practical case in favour of monitoring programs that carefully address detectability and spatial variation for the conservation of endangered species.

The end users and the ultimate goal of the results come to argue on the investment of sufficient amount of time and resources. From simple anglers to whole organizations and governments may benefit from the provided service on decision-making, precaution measurements as well as protecting species about to extinct.

2. Mission

As mentioned in the previous section, the issue of the over – fishing takes important aspects when comes to make decisions and protect several species.

The scope of this analysis is to provide solutions on issues like the following:

Figure 1 : Issues and Over-fishing

On this report, we will try to cover the underwater fish monitoring through image classification processes and models. Although the presented approach is based on Fish Camera images from rivers and seas in can be used is several other monitoring cases.

Examples

Some examples of already existing applications of fish classification are:

1. ***FishCam***. The newly developed FishCam is used to monitor fish migration especially in fish passes. To avoid time and cost consuming field work for fish pass monitoring, this project aims to develop a semi-automatic monitoring system that enables a continuous observation of fish migration. The FishCam migration monitoring system consists of a detection tunnel and a high-resolution camera, which is mainly based on the technology of security cameras. The camera system is recording video clips of migrating fish and drifting particles with a size covering more than 1 – 3% of the number of pixels of the image. The underwater camera is installed in a pyramid stump shaped housing filled with freshwater to keep the influence of turbidity in the fish pass to a minimum. Migration of fish is recorded without contact and stress. The ongoing key challenge is the development of a robust image classification algorithm, which counts, measures and classifies the passing fish.

2. Applications of detection illegal fishing launched by European Union. [5] The ***FOCUS*** (Fisheries Open Source Community Software) community has been established with the purpose to create and maintain free software dedicated to the management, conservation and sustainable use of the fishery resources.

Our Goal

Our goal is to present a case study for **automatic fish species classification in underwater image monitoring**. The images are taken from coral reefs and rivers. The presented classification scheme in this study is based on **Convolutional Neural Networks (CNN)** that do not require the calculation of any hand-engineered image features. Instead, these networks use the raw image as input. On the context of Deep Learning, image classification is an added huge boost in the already rapidly developing field of computer vision. Many new applications of computer vision techniques have been introduced and are now becoming parts of our everyday lives.

Did you know..

Image Classification techniques have been developed since 1980s.

Business approach

Although, there are several studies on the current domain, of fish classification, our business approach of this analysis is a possible production application composed of a software and an underwater camera. The user may capture underwater footage, with final goal to take insight on the catch, learn and make decisions. This could be a low cost gadget along with the software to engage fishing or underwater experiences like scuba diving and snorkeling.

3. Data

The research and collection of the data is considered the most important part of the implementation of a Deep Learning model.

Although there are available a lot of fish datasets, providing a variety of photos as well as information regarding thousands of different species, they are only helpful for educational purposes. Actually very few datasets were suitable for the fish image Classification task.

Some of the available databases with fish data are the following:

- **FishBase;**
- **IGFA;**
- **The Fish database of Taiwan;**
- **UCDAVIS PISCES**.

All the above mentioned DBs provide basic information on species and families, underwater fish videos and photo galleries, varying due to the scope of each site.

After extended research we concluded on 4 datasets suitable for the task. Below are provided some details for each dataset.

The datasets

- **Fish Recognition Ground-Truth data** [B6]

This fish data is acquired from a live video dataset resulting in 27.370 verified fish images. The whole dataset is divided into 23 clusters and each cluster is presented by a representative species, which is based on the synapomorphies characteristic from the extent that the taxon is monophyletic. This data is organized into 23 groups, where the fish images and their masks are stored separately. Though, only the images were used to train the network.

- **Fish Dataset** [B7]

This fish dataset currently consisting of 3,960 images collected from 468 species. This data consists of real-world images of fish captured in conditions defined as "controlled", "out-of-the-water" and "in-situ". The "controlled", images consists of fish specimens, with their fins spread, taken against a constant background with controlled illumination. The "in-situ" images are underwater images of fish in their natural habitat and so there is no control over background or illumination. The "out-of-the-water" images consist of fish specimens, taken out of the water with a varying background and limited control over the illumination conditions. A tight red bounding box is annotated around the fish. Unfortunately, this dataset has only very few images for every species (about 20). We did try to train o model using this dataset, but the result was as, you can imagine, disappointing

- **Labeled Fishes in the Wild** [B8]

The labeled fishes in the wild image dataset is provided by NOAA Fisheries (National Marine Fisheries Service) to encourage development, testing, and performance assessment of automated image analysis algorithms for unconstrained underwater imagery. The dataset includes images of fish, invertebrates, and the seabed that were collected using camera systems deployed on a remotely operated vehicle (ROV) for fisheries surveys. Annotation data are included in accompanying data files (.dat, .vec, and .info) that describe the locations of the marked fish targets in the images. Unfortunately, this dataset did not include labels/classes of the fish. If we were interested in a fish/ no fish classification task this dataset would be perfect for training. We used though the file which contained "negative" non-fish images.

- **FishNet** [B9]

FishNet is the software part of the FishCam monitoring system. This system was developed at the Institut of Water Management, Hydrology and Hydraulic Engineering at the University of Natural Resources and Life Sciences, Vienna, Austria by Helmut Mader and Frederik Kratzert. The FishCam monitoring system enables contact-free observation of fish migration through underwater video observation in technical fish passes for e.g. functional monitoring. This was historically done by fish traps which mean a lot of stress for the fish, as well as time and cost consuming fieldwork. The open dataset consists of 7 different species of Austrian rivers and 7.900 images in total. The good thing about this dataset, although it is quite small is that among others it contains more common and more rarely seen fish in the dataset, as the situation is in rivers. The idea was to study, if it is possible to train a network also on species with sparser dataset and if the network can classify them evenly compared to species with lots of data available. In addition, the dataset also represents the different image qualities that occur in Austrian rivers and not only clear water examples so that we could test, how good such a system could run in operation year-round.

Final dataset

The final dataset, that was used to train the network model, was a conjunction of all the above mentioned datasets except the Fish Dataset. It appeals more interesting to expand the variety of cases captured by images like these.

The **Fish Recognition Ground-Truth** data and the **FISHNET** dataset are the core index of our final dataset and, as mentioned before, it was also used a file including “Negative” non-fish images from the **Labeled Fishes in the Wild** dataset.

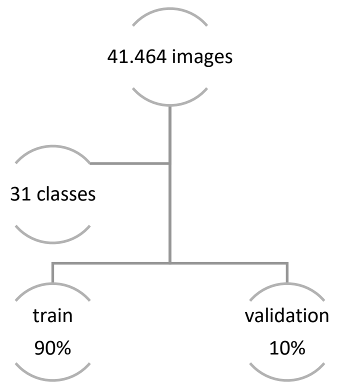

Figure 2 : Summary of the dataset used

The final dataset consists of more than **41.464** images divided into **31** different classes. Data was stored in subfolder; in which each name indicates a class / name of a fish.

Training and validation sets were split manually. **10%** of each folder was taken for the **validation** set, as the remaining **90%** left for the **training** set. In this way, the most frequent species would appear more frequently in the validation set.

The final dataset consists of **37.458** images for the training set, as for the validation are **4.006** images.

Note that a few predictions were also kept. Recall that the images were taken underwater. This actually meets our main goal of this project, which is to build an application that could help in fish classification and monitoring in specific areas of interest such as lakes, rivers or coral reefs.

On the contrary, the fact that images were taken from different distributions (sea and river) was a huge concern. It is possible that the network could learn to identify images based on the background of the picture rather than the fish itself. Fortunately, this was not the case.

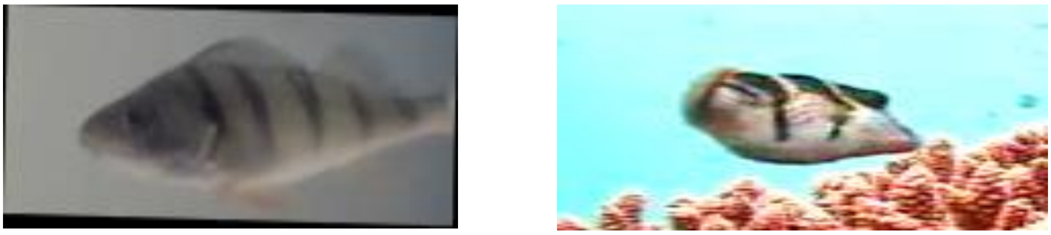

Figure 3 : Sample images

Image size

The images were of varying sizes. Due to the presence of fully connected layers in the neural network, the images being fed into the network will be of a fixed size. Due to this fact, before the image augmentation, images were preprocessed to the size which the network requires. That would be **64x64**. With the fixed sized images, we got the benefits of processing them in batches. All the labels of each subfolder, were imported and encoded through one hot vector.

Number of images

Another issue that was taken into consideration while processing the data, was the number of images available per class. For example, some species/ folders contained more than **2000 images** while others had less photos.

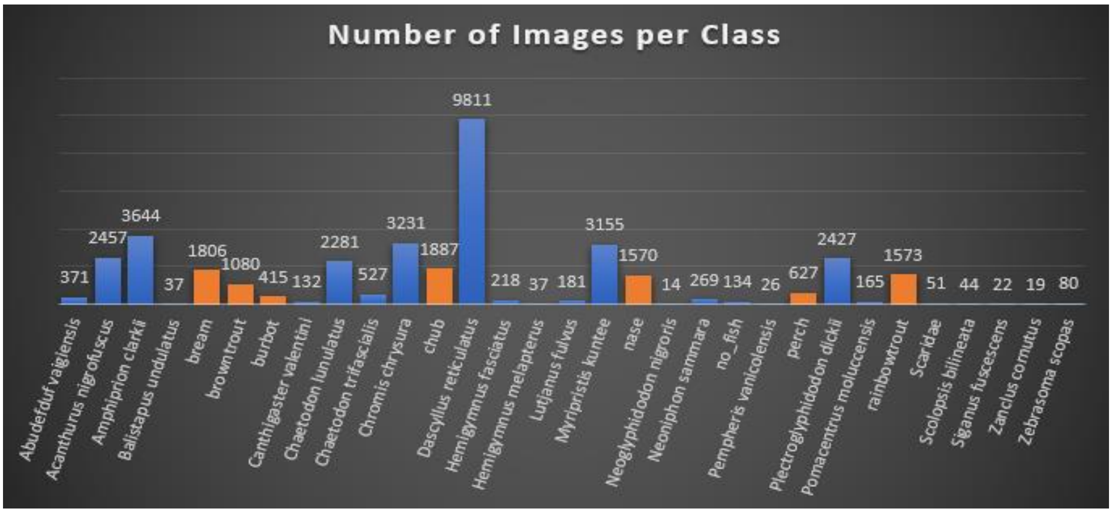

Figure 4 : Number of Classes

The decision was to leave the dataset untouched and proceed with some augmentation techniques to fix the issue of the number of images per class. The results were encouraging although not all augmentation techniques were suitable for the working dataset. After some experiments, random shifts, zoom and flip were chosen.

Data normalization, as it is an important step, ensures that each input parameter (pixel, in the specifc case) has a similar data distribution. This makes convergence faster while training the network. Data normalization is done by subtracting the mean from each pixel, and then dividing the result by the standard deviation. The distribution of such data would resemble a Gaussian curve cantered at zero. For image inputs the pixel numbers should be positive, so we might choose to scale the normalized data in the range [0,1]. The augmentation process was implemented through an ImageGenerator function with a batch size of 200.

The ultimate goal was the network to recognize fish presence in any part of an image, as a fish could be visible partially in the corner or edges of an image. For this reason, fish were shifted to various parts of an image, with final result of minor additional noise in some cases. Something equally remarkable was that the Neural Network should not learn to recognize heads or tails of the fish in a specific part of the image.

Other cases

A few more advanced techniques such as conditional GANs, Salt and Pepper noise or Lighting condition could also be used, but seemed to be much costlier than the technique already used.

4. Methodology

The architecture of the model build for image classification was **Convolutional Neural Network (CNN).**

A **convolutional neural network (CNN)** is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vision that includes image and video recognition, along with recommender systems and natural language processing (NLP). A neural network is a system of hardware and/or software patterned after the operation of neurons in the human brain. Traditional neural networks are not ideal for image processing and must be fed images in reduced-resolution pieces. CNN have their “neurons” arranged more like those of the frontal lobe, the area responsible for processing visual stimuli in humans and other animals. The layers of neurons are arranged in such a way as to cover the entire visual field avoiding the piecemeal image processing problem of traditional neural networks.

What was done

The process of finding the best architecture and hyper - parameters for the final CNN was really challenging. We started with a very simple model (two layer) and after making some experiments we ended up with a pre - trained model and more specifically the VGG16 model. Public available weights were used for the network and trained against the ILSVRC12 challenge data set..

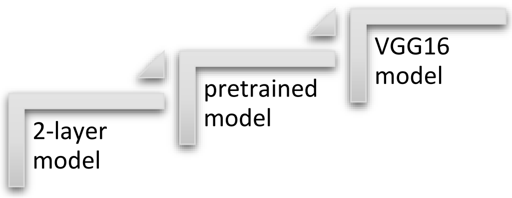

Figure 5 : Steps to decide the final model – Transfer Learning

This method is also known as **Transfer Learning**.

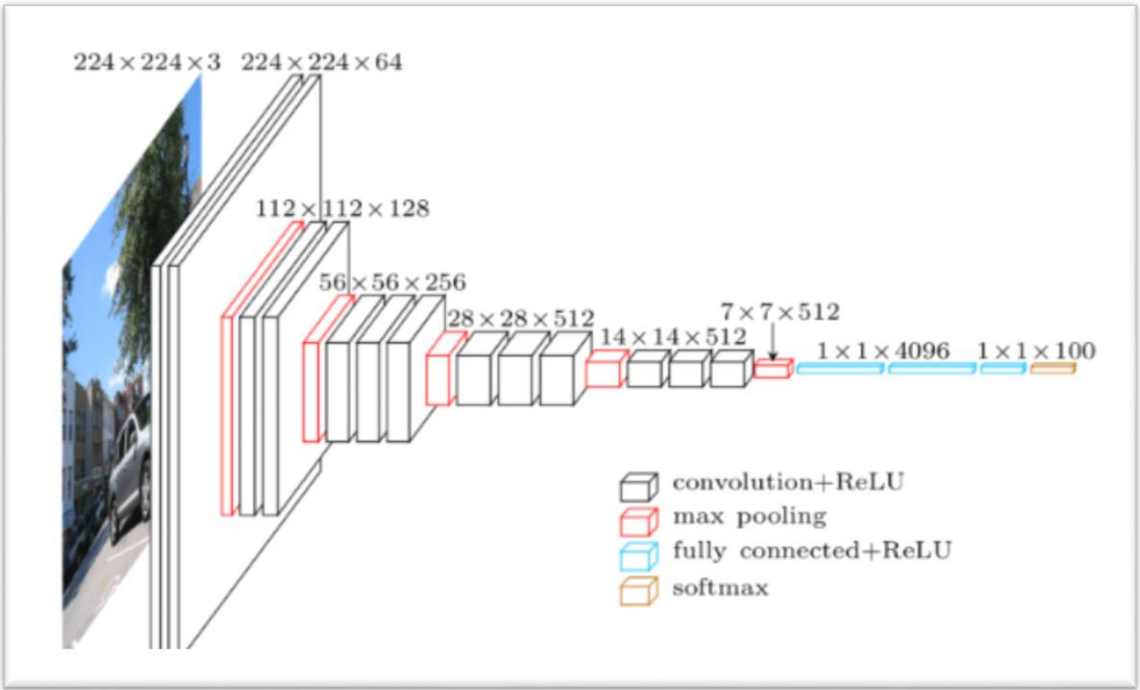

Figure 6 : A visualization of the VGG architecture

VGG is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. This network is characterized by its simplicity, using only 3×3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier [B10]. As a result, it has over 138 million parameters.

The "16" stands for the number of weight layers in the network. In 2014, 16-layer networks were considered very deep (although we now have the ResNet architecture which can be successfully trained at depths of 50-200 for ImageNet and over 1,000 for CIFAR-10). Compared to other famous models VGG16 is preferred for its simplicity. Unfortunately, there are two major drawbacks with VGGNet. First It is painfully slow to train and second the network architecture weights themselves are quite large (in terms of disk/bandwidth). In our case it was by far the slowest model in training process, but on the other hand, it was the best in terms of accuracy.

Transfer Learning

Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned (*Chapter 11: Transfer Learning, Handbook of Research on Machine Learning Applications, 2009.*). It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

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

The 6 different architecures of VGG Net. Configuration D produced the best results

Figure 7 : CNN Configuration

To meet the needs of the project, the model had to be fine – tuned.

There 31 classes, so the last layer performs 31 – way ILSVRC classification and thus contains 31 channels. In addition, as the model was extremely costly to train, the last two dense layers were removed and replaced with one of 128 units. Recall that the original VGG16 model had two dense layers both size of 4096. So, the final model consists of 14 layers (13 Conv layers and 1 Dense layer). The input size of the images was resized to 120x120 and the width of the network starts at a small value of 64 and increases by a factor of 2 after every sub-sampling/pooling layer. Every Convolutional layer has 3x3 filters with stride and padding equal to 1. The Max-Pooling layers are 2x2 with stride 2x2 and as mentioned before there is one fully connected layer with 128 units. The last layer is a softmax classification layer with 31 units and the activation function is the Relu. As a result, total parameters reduced by almost 120 million but at the same time it was still accurate enough. So, the final model has only 14,9 million parameters in total and 266.271 of them are trainable.

```
In [4]: model.summary()

Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 64, 64, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 64, 64, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 64, 64, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 32, 32, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 32, 32, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 32, 32, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 16, 16, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 16, 16, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 8, 8, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 8, 8, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               262272
_________________________________________________________________
dense_2 (Dense)              (None, 31)                3999
=================================================================
Total params: 14,980,959
Trainable params: 266,271
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 8 :Summary of the model

As far as concerns the hyper - parameters of the final model, Adam was used as optimizer with a learning rate of 0.001 (the other parameters of the optimizer kept at their default values) and a batch size of 200. Adam optimizer [B11]:

- Manage to convert in a reasonable time
- Learns the fastest
- Is more stable than other optimizers
- It does not suffer and major decreases in accuracy

The number of epochs, which is defined as the number of times the entire training set pass through the neural network was set to 3.

- Steps per epoch = 5000;

- Validation steps = 1500.

Unfortunately, because the training process was extremely time consuming (approximately 1 day per epoch) there was not enough time to train the model longer. Computationally expensive was to increase the input size of the images and add more dense layers. Perhaps this could have given even better results. Some few more variations were tried, but they didn't yield to better results.

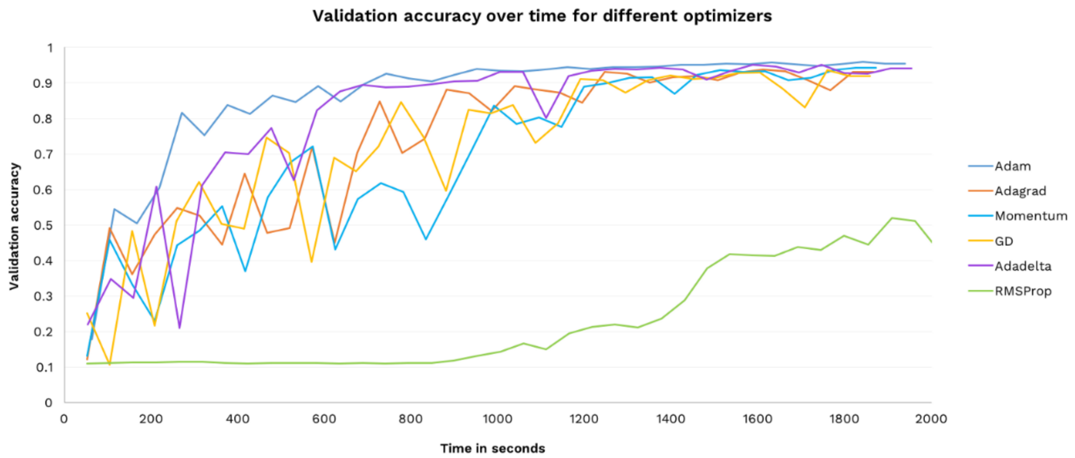

Figure 9 : Adam Optimizer performance

Summarizing

As mentioned above a lot of different models were tested. It is worth mentioning that the second-best model achieved 90% in validation accuracy. This model was made from scratch and was 6 layers deep (4 convolutional layers and 2 fully connected layers of size 512). The model had 3x3 filters with stride and padding set to 1. The first convolutional layers had 32 filter and they were doubled after each layer reaching 256 in the fourth layer. Input size, activation function (Relu) as well as the optimizer algorithm (Adam, lr=o.oo1) were the same as our VGG16 model. On the contrary this model used Dropout of 25% after every 2 convolutional layers and 50% after the first dense layer. As a result, there were more than 22 million weights. This model was trained for 10 epoch and the results were very promising as it reached 90% on validation accuracy and in the same time validation loss was equal to 0.41. Training lasted no more than 10 hours. It is observed that even though this model was not as deep as the VGG16, it performed really well

5. Results

Loss Function

After training, the convolutional neural network based on VGG16 resulted as shown Figure 10 : Network results. As it is observed in Figure 11 : Cross- entropy Loss Function the cross-entropy loss function for both training and validation reached a low plateau towards the end of training.

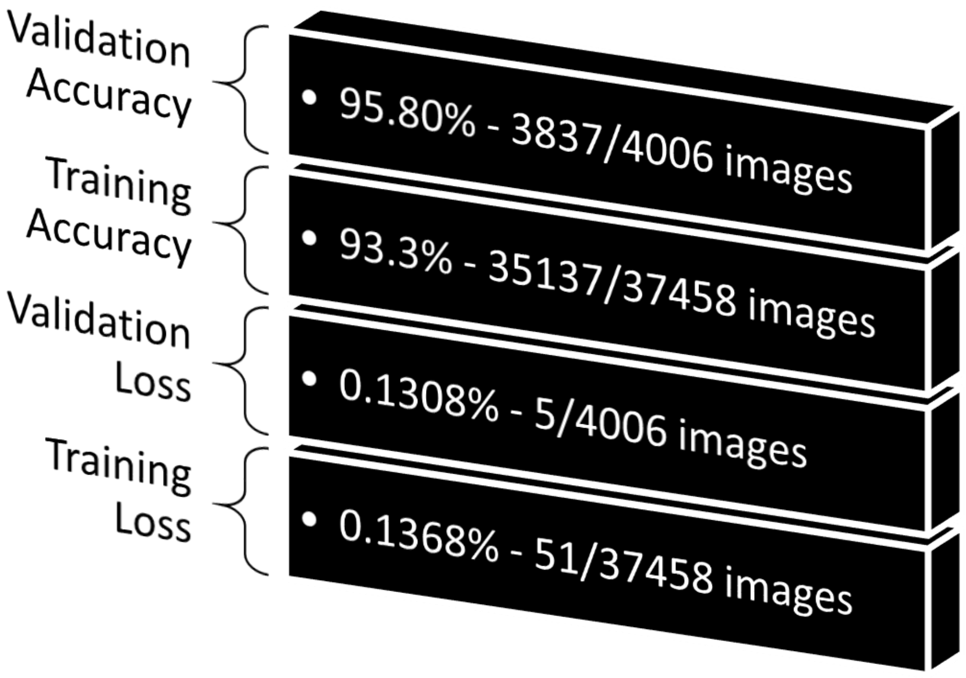

Figure 10 : Network results

Accuracy

In addition, accuracy gradually increased and reached a peak of 95% at the end of epoch 10. (Figure 12 : Model Accuracy). Recall that steps per epoch and validation set to 5000 and 1500 respectively which means that the epochs were actually about 200.

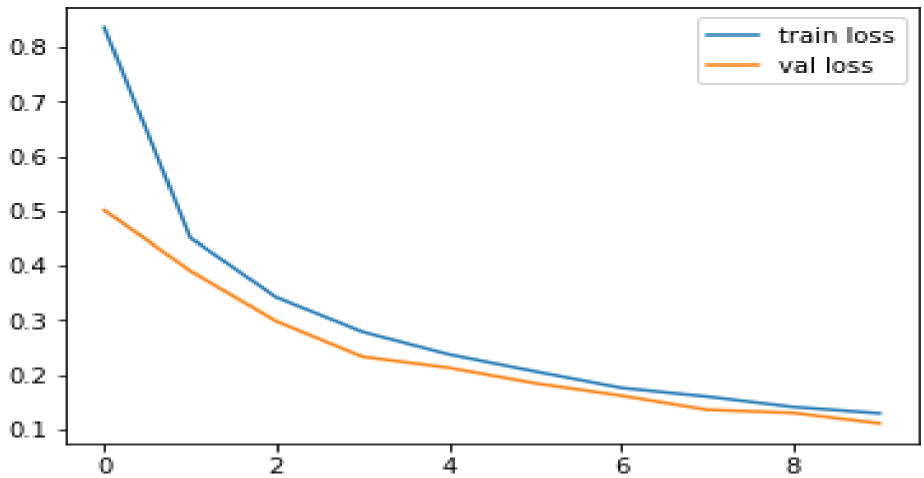

Figure 11 : Cross- entropy Loss Function

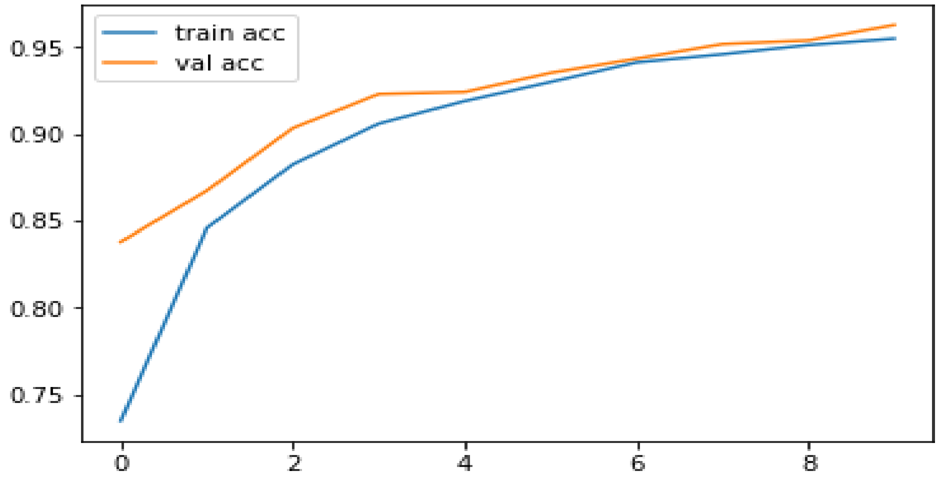

Figure 12 : Model Accuracy

In each one of the different 31 classes, the number of images varied, as well as the accuracy. On the Table 1 : Images & Accuracy per class, it is observed that there are 5 classes marked, where the accuracy of the VGG16 network was less than 80%. In most of the cases, in these 5 classes, there were very few images in the training set as well as in the testing set (as it is derived from the training set.

a/a	*Class*	*Number of Images*	*Accuracy*
0	Abudefduf vaigiensis	371	100,00
1	Acanthurus nigrofuscus	2457	55,00
2	Amphiprion clarkii	3644	99,60
3	Balistapus undulatus	37	85,70
4	bream	1806	85,10
5	browntrout	1080	89,10
6	burbot	415	53,50
7	Canthigaster valentini	132	100,00
8	Chaetodon lunulatus	2281	100,00
9	Chaetodon trifascialis	527	100,00
10	Chromis chrysura	3231	99,20
11	chub	1887	86,20
12	Dascyllus reticulatus	9811	99,90
13	Hemigymnus fasciatus	218	100,00
14	Hemigymnus melapterus	37	100,00
15	Lutjanus fulvus	181	100,00
16	Myripristis kuntee	3155	94,50
17	nase	1570	76,00
18	Neoglyphidodon nigroris	14	50,00

19	Neoniphon sammara	269	95,00
20	no_fish	134	96,00
21	Pempheris vanicolensis	26	100,00
22	perch	627	94,30
23	Plectroglyphidodon dickii	2427	99,60
24	Pomacentrus moluccensis	165	100,00
25	rainbowtrout	1573	85,60
26	Scaridae	51	100,00
27	Scolopsis bilineata	44	100,00
28	Siganus fuscescens	22	100,00
29	Zanclus cornutus	19	100,00
30	Zebrasoma scopas	80	58,30

Table 1 : Images & Accuracy per class

The remarkable point of the model is that while training, it mixed images from different classes. In other words, sea and river fish trained in the same way and recognized from the mask of the image itself and not from the background (i.e. water, rocks or corals).

Training & Testing

For training and testing the neural networks, the open source deep learning frameworks TensorFlow and Keras were used. All training and testing were performed on Windows 64-bit desktop personal computer. Computational resources were quite limited as the “best” central processing unit (CPU) uses an Intel Core i5 6300U, 8 GB random access memory, and no graphical processing unit (GPU). Training the final model on a computer like this took more than 3 days.

It is possible that the training and validation accuracy could have been even better if this model was trained for more than 10 epochs. In any case the success of transfer learning in this study is promising for the prospect of training Convolutional Neural Networks for other imaging tasks.

6. Members/Roles

Our team, the authors of this project, is composed of four members. Each one has a different background either technical or business.

At the very early stages of this project the responsibilities were divided into four parts. Each one had a different task. We shared information and insights through often team meetings (including skype calls during summer vacation).

The Team

Christos Katsaris has graduated from Computer Engineering and Informatics department of University of Patras and currently is working as Business and Technical Consultant at Entersoft. Christos focused on searching and finding the correct dataset. He was also responsible for the pre-processing of the data. Finally, he studied about augmentation techniques and suggested the most appropriate for our needs.

Argyro Rapanou has graduated from Computer Engineering and Informatics department of University of Patras and currently is working as a Software Test Engineer at Intrasoft International. She studied the theory behind the Neural Networks and suggested some architectures. She contributed on the coding part of the project as well as to the preparation of the deliverable (report and presentation).

Pavlos Polyzogopoulos has graduated from Department of Management Science and Technology of AUEB and currently is working Business Analyst at Henkel. He focused on the business plan of the project. Specifically, he searched useful information regarding the progress of CNNs in general and their business implementations. Finally, he focused on finding studies and applications that were close to the project's goal.

Georgia Souliou has graduated from Department of Informatics and Telecommunications of Athens University and currently is working as a programmer in NN Hellas. She focused on the coding part of the project implemented on Python. She studied basic tutorials and provided useful material to the rest of the team. Most of the implementation of the code is a result of her efforts.

Team work

Argyro and Pavlos focused on transfer learning models while Christos and Georgia on building CNNs from scratch. As it turned out the first models was better in terms of performance. Finding the best model including its hyperparameters was very challenging. As in the early stages of this project each one contributed on different parts like cleaning up the final code and adding notes, combining different parts of the final report and preparing the presentation.

Everyone's contribution was crucial for the implementation of this project. The team work was great and the result quit satisfying.

7. Bibliography

a/a	Link	Description
B1	https://www.fishforward.eu/	WWF
B2	http://www.worldwatch.org/node/5960	World Watch Institute
B3	https://www.prnewswire.com/news-releases/new-app-identifies-fish-informs-of-regulations-in-seconds-300382321.html	CISION
B4	https://fishverify.com/launch/	Fish Verify
B5	https://ec.europa.eu/fisheries/cfp/control/technologies_en	European Union
B6	http://groups.inf.ed.ac.uk/f4k/GROUNDTRUTH/RECOG/	Fish Recognition Ground-Truth data
B7	https://wiki.qut.edu.au/display/cyphy/Fish+Dataset	Fish Dataset
B8	http://academictorrents.com/details/41bc10c77d54b49fb0a96ff5d4a0814bc2ab7da7	Labeled Fishes in the Wild
B9	https://drive.google.com/file/d/0B3YsW-PFiJOLdkRxS05DdGRraTA/view	FishNet
B10	https://en.wikipedia.org/wiki/Softmax_function	Softmax function
B11	https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2	Adma Optimizer

Table 2 : Bibliography

8. Time Plan

The research for the whole project and the business case started back in July.

The first part of the project took almost 2 weeks. Early in September the testing models were set up and the experiments started.

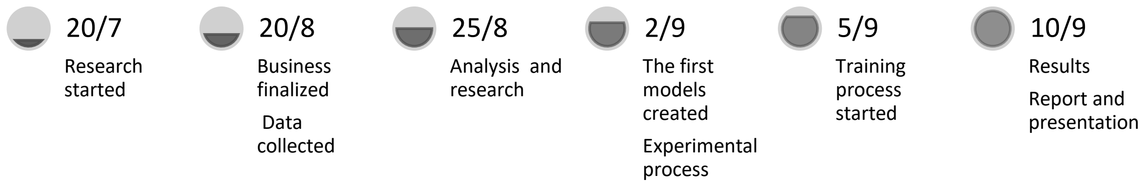

Figure 13 : Time Plan

The most time-consuming part of the project was the training as each model took 6 to 8 hours.

The whole training process lasted about two weeks. While proceeding with the research of the architecture of the models, the training process increased dramatically. It is remarkable that just an epoch could last 32 hours. To handle such issue, each member trained a different model (different architectures and variations of subgroups) and shared the results with the team.

During the final training stages, the final report was conducted along with the presentation. Though the results were collected and added very late, just before the deadline of the delivery.

9. Contact Person

The contact person of the Team is Pavlos Polyzogopoulos.

He represented the Team in all meetings with Mr. Papageorgiou and Mr. Pappas and kept email contact via email, for all the questions.

10. Notes and Comments

The main concern during the implementation of this project was **time** and **memory** resources

Training almost 20 different models and searching more was prohibitive it terms of time left till the delivery. Recall that the final model took about 4 days to train.

Although the results were quite satisfying in terms of accuracy, it was difficult to plan the execution from the beginning.

Note that none of the members were familiar with coding in Python, so there were a few challenging difficulties in the implementation. The most difficult part was the actual image processing, while tuning the models was time- consuming.

Online material and the Course lecture were helpful.

From the team.

This project gave us the opportunity to gain experience of learning about Convolutional Neural Networks. The era of deep learning and artificial intelligence is interesting and challenging.

Though, there is still room for improvement and learning. The course of Big Data Content Analytics was a huge motivation for going deeper into Neural Networks.

Teamwork and collaboration were the keys to project success. Every member of this team was committed to the common goal and the whole project is a team effort result.

11. The deliverable

The deliverable folder contains the code, written in python, the report and the presentation of the project. It is available on GitHub : https://github.com/gogosouliou .

*****End of document*****