HIGH SCHOOL EDITION
Journal of Student Research
Volume 11 Issue 3 (2022)

Fish Species Image Classification Using Convolutional Neural Networks

Anishka Mohanty[1], Guillermo Goldsztein[#] and Raphaël Pellegrin[#]

[1]Researcher Horizon Academics, India
[#]Advisor

ABSTRACT

This paper demonstrates the classification of various fish species using different machine learning methods. By incorporating machine learning algorithms, modeling, and training, the project classifies fish species using neural networks with the help of multiple features like length, width, and more. Ultimately, this project attempts a deeper analysis of the differences between determining fish species with PyTorch and TensorFlow. Convolutional Neural Networks (CNN) is a powerful algorithm that is used in image classification problems. Python has various libraries which can be used to build a model for the same purpose; the ultimate goal of this study is to see whether using different libraries will affect the accuracy. I would like to see whether new and more advanced methods can be used to classify large schools of fish rather than only in labs. I developed separate Python codes using PyTorch and TensorFlow individually. Using each code, I obtained results, and in the end, performed a comparative study between both to come to my conclusion. My main findings were that PyTorch gave a more accurate prediction comparing to TensorFlow. I believe this was the case because the PyTorch code incorporated neural networks with more layers, so it increased the training and validation accuracy. From here, it is evident that while neither method necessarily possesses setbacks, PyTorch has a significant edge in accuracy (99.75% to 87.22%). Therefore, when scientists apply classification with CNN, PyTorch may be more optimal for producing better results.

Introduction

Our underwater ecosystem sustains over 34,000 species of fish in fresh and salt waters as of 2021. As the fish population continues to ascend rapidly, eco-friendly fish classification through machine learning is an efficient method to help analyze various features of fish for the determination of their species. To date, there are barely any methods to analyze a large number of fish. Through pattern recognition and machine learning techniques, scientists can examine fish on a larger scale. In this paper, I used convolutional neural networks to identify fish species through binary classification across datasets.

Machine learning is an application of Artificial Intelligence (AI) to help construct detailed models based on given data. Using the patterns determined from the modeling allows the user to predict based on any additional data and therefore make a decision. The main aim of this project is to validate whether scientists can use AI, precisely machine learning for sustainable fish classification. Likewise, this project also aims to check for differences in accuracy and precision while using PyTorch and TensorFlow. This research will help scientists determine which method is more conducive for higher quality classification.

This problem that I worked on belongs to a set of approaches known as supervised learning. Supervised learning, in shorter terms, indicates any problem consisting of datasets with prescribed labels and features. Using these prescriptions, the data can be trained into algorithms to formulate predictions. A particular example consists of features $(x_1, x_2, \ldots, x_n)$ and label y. In this problem, each example, an image, consists of numerous features where one feature corresponds to a pixel in the image. A pixel can hold any integer from 0 to 255.

Normally, the label is an expected prediction of a model when new features from examples are inputted into the model.

In Section 2, I take a look at the data and in Section 3, I describe the different methods that have been used in the process to train the model. In Section 4, I obtain the results and discuss the quality of the model. Section 5 continues with a brief comparison between the results obtained with PyTorch versus TensorFlow.

Data

The project reuses a Kaggle Competition dataset with a file size of 2.76 KB which remains as the original source. This dataset consists of exactly 18,000 fish images including the ground truth files. Samples of the data can be seen below in Figure 1.

Figure 1. Sample of different species.

For training, the dataset was split into training and validation sets to build the model. The training dataset is the main dataset which is used to train the model and observe patterns. The validation set is used for validation of model created during training. Additionally, there is a testing dataset used to evaluate the trained model for accuracy. The training set always contains the most amount of data followed by a test set and then the validation set. Later, this data was split into 5040 training images, 2700 test data images and 1260 validation dataset images once the ground truth files were removed for TensorFlow. In the case of PyTorch, the data was

split into a training set (7290 images), test data (900 images), and validation set (810 images). The input shape for the TensorFlow problem was 200 x 200 versus the PyTorch one which was 128 x 128. Once the data was augmented, this is how it looked like:

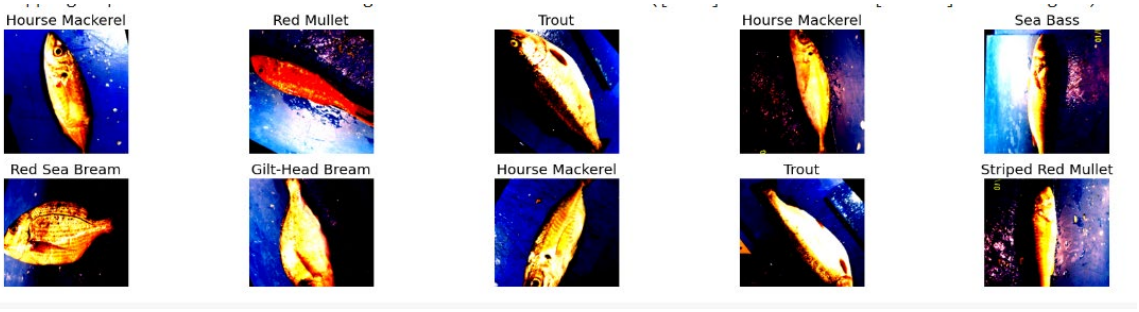

Figure 2. Data after augmentation

Method

Multi Classification Problem

A multi-classification problem is any problem that has more than two classes or categories. In this case, there are 9 categories. Hence, this problem can be considered as a multi-classification problem i.e., each example would belong to one of these categories. The categories were as follows: Black Sea Sprat, Red Mullet, Trout, Striped Red Mullet, Shrimp, Red Sea Bream, Gilt-Head Bream, Hourse Mackerel and Sea Bass. For each possible category, a new variable was created. Because the labels were strings, I converted each categorical value into a new categorical column and assigned a binary value to the column. It essentially formed a 1D array with nine components. For example, an image categorized "Shrimp" had an array 0,0,0,1,0,0,0,0,0]. This process is called One Hot Encoding. From the project, let's take a look at the dataset before and after One Hot Encoding.

	image	label
0	/content/Fish_Dataset/Fish_Dataset/Red Mullet/...	Red Mullet
1	/content/Fish_Dataset/Fish_Dataset/Trout/...	Trout
2	/content/Fish_Dataset/Fish_Dataset/Shrimp/...	Shrimp

Figure 3. Image directory with Class

Table 1. Image directory with label after One Hot Encoding

Image Directory	Black Sea Sprat	Red Mullet	Striped Red Mullet	Shrimp	Red Sea Bream	Gilt Head Bream	Hourse Mackerel	Sea Bass	Trout
/content/Fish_Dataset/Fish_Dataset/Red Mullet/...	0	1	0	0	0	0	0	0	0

/content/Fish_Dataset/Fish_Dataset/Shrimp/...	0	0	0	1	0	0	0	0	0
/content/Fish_Dataset/Fish_Dataset/Trout/...	0	0	0	0	0	0	0	0	1
/content/Fish_Dataset/Fish_Dataset/Red Sea Bream/…	0	0	0	0	1	0	0	0	0
/content/Fish_Dataset/Fish_Dataset/Black Sea Sprat/…	1	0	0	0	0	0	0	0	0
/content/Fish_Dataset/Fish_Dataset/Hourse Mackerel/…	0	0	0	0	0	0	1	0	0
/content/Fish_Dataset/Fish_Dataset/Sea Bass/…	0	0	0	0	0	0	0	1	0
/content/Fish_Dataset/Fish_Dataset/Gilt-Head Bream/…	0	0	0	0	0	1	0	0	0
/content/Fish_Dataset/Fish_Dataset/Striped Red Mullet/…	0	0	1	0	0	0	0	0	0

When the model is fed with a set of features as the input, it returns a prediction of the labels as its output. This output, generally denoted by $\hat{y}$, is also a 1D array with nine components. This can be expressed as $\hat{y} = [\hat{y}_1, \hat{y}_2, \hat{y}_3, \hat{y}_4, \hat{y}_5, \hat{y}_6, \hat{y}_7, \hat{y}_8, \hat{y}_9]$. The value of $\hat{y}$ is always a positive number and the sum $\sum_{k=1}^{9} \hat{y}_k = 1$. $\hat{y}_1$translates to the probability of the example being labeled "Black Sea Sprat", $\hat{y}_2$ interprets the probability of the example categorizing as "Red Mullet" and similarly for the other $\hat{y}$. Thus, if $\hat{y}_1$ is greater than the other 8 $\hat{y}$s, then the fish gets predicted as "Black Sea Sprat". Likewise, if $\hat{y}_8$ is larger than the other eight components, then the fish is predicted as "Sea Bass". This remains as a parallel case when any of the 9 components are greater than the rest.

Logistic Regression

Basics

Logistic regression is a machine learning technique that is used to develop models in binary classification problems. With respect to multi classification problems, it becomes Multinomial Logistic Regression instead of Binary Logistic Regression. In general, logistic regression is used to find the statistical probability of a category based on prior observations. The probability $P(c = 1|x)$ that the input x will be classified under class $c = 1$ is represented by the output $\sigma(x)$. The sigmoid function is the function

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad \cdots (1)$$

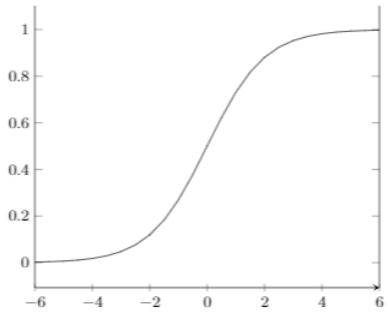

Figure 4. Sigmoid Activation Function - $\sigma(x)$

Some properties of the sigmoid function are:

1. Range: $\sigma(x)|\{0 < \sigma(x) < 1\}$
2. As the value of x decreases, it approaches towards 0 whereas, when the x value increases, it approaches towards but does not go beyond 1.
3. From Figure 2, it is seen that $\sigma(x)$ is an increasing function.
4. $\sigma(x)$= 0.5

Since this problem contains numerous parameters, it can be simplified generally into k features. Calling these features $x_1, x_2, \ldots, x_k$, the prediction $\hat{y}$ will be in the form

$$\hat{y} = \sigma(w_1x_1 + w_2x_2 + \ldots + w_kx_k + b) \quad \cdots (2)$$

where $w_1, w_2, \ldots, w_k$ and b are parameters that will help make the cross-entropy error on the training set minimized.

Categorical Cross Entropy Error

As seen in the basics, logistic regression essentially calculates the probability $P(c = 1|x) = \sigma(x)$. Categorical cross entropy error (a.k.a SoftMax error) also known as the loss function checks the difference between two probability distributions from the test data for multiclass classifications. Assume there are k features and 1 label. The features are denoted as $x_1, x_2, \ldots, x_n$ and the label as y. If there are t labels, they are denoted with $y_1, y_2, \ldots, y_t$. The predictions of label y for t labels are denoted by $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_t$. If there are s examples, the features of the i^{th} example are denoted by $y_{i1}, y_{i2}, \ldots, y_{ik}$ and the scaled labels as $y_{i1}, y_{i2}, \ldots, y_{it}$ for t labels. In the same manner, the predictions of the labels are written as $\hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{it}$.

The function of the cross-entropy error is

$$J_{CE} = \frac{-1}{s}\sum_{i=1}^{s}\sum_{j=1}^{k} y_{ij} \log\ (\hat{y}_i) + (1 - y_i)\log\,(1 - \hat{y}_i) \quad \cdots (3)$$

Here are some of the properties that are useful:

1. $J_{CE}(y_1, \hat{y}_1) \geq 0$
2. If $y_1 = \hat{y}_1$, then $J_{CE}(y_1, \hat{y}_1) = 0$
3. The loss will be zero when the prediction is one (same as the predicted label).
4. Since I used One Hot Encoding, the predictions for the other classes can be ignored and the loss for the selected class can be found. This is called categorical cross entropy.

Neural Networks and CNN

Neural networks are another mechanism used in machine learning to obtain more precise results and predictions. In essence, they are another type of function that is more complex than logistic regression. Neural networks parallel to neurons in our brain. In machine learning, the neurons pass on the input data through various functions to obtain the output. The images are transformed into numbers and are generally arranged in matrices or arrays. All those numbers are generally denoted by the letter x. X is not just one number but a series of images. The function of all these numbers is commonly denoted as $\hat{y}$ of X. Neurons can be grouped in the layers: Input layer, Hidden Layer and Output Layer. Figure 5 showcases a simple demonstration of neural networks.

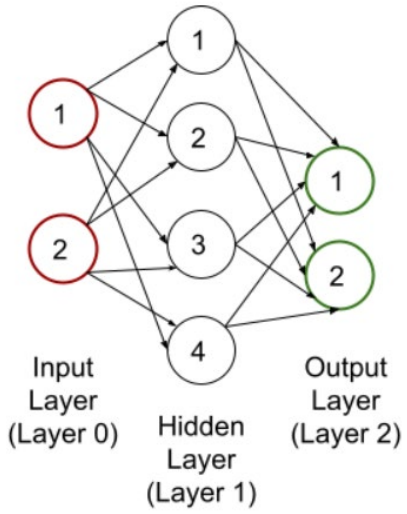

Figure 5. Simple Representation of Neural Networks. The input layer has 2 nodes, the hidden layer has 4 nodes, and the output layer has 2 nodes in this example.

Neural networks have several parameters:

1. They have k features and s labels. In this example, there are two features and two labels.
2. For the number of layers in the network, there are L layers where the input layer is not counted. Hence for Figure 5, there are two layers.
3. There is one input layer (layer 0), $L-1$ hidden layers and one output layer (layer 3).
4. Layer l has $n^{[1]}$ nodes where $n^{[0]} = k$ and $n^{[L]} = s$.
5. Every node in a single layer is connected to all the nodes in the subsequent layer.

In a multi classification problem, the activation function used here is the ReLU (rectified linear unit) function. The ReLU function is the function:

$$\text{relu}(x) = \begin{cases} x \text{ if } x > 0 \\ 0 \text{ if } x \leq 0 \end{cases} \qquad \cdots (4)$$

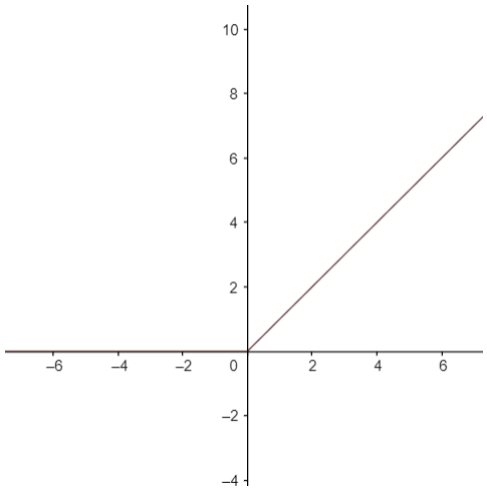

Figure 6. ReLU Activation Function - relu(x)

Convolutional Neural Networks also called CNN are a type of neural networks which are used for image classification. There are 3 types of layers in CNN: Convolutional Layer for converting the pixels into one value, Pooling Layer for reducing the dimensions and Fully Connected Layer where all inputs from one layer connect to every activation node in the next layer.

Results

I executed the project two ways, with PyTorch and TensorFlow. PyTorch is an advanced mechanism that optimizes and trains models for accurate results especially when there is a usage of neural networks. On the other hand, TensorFlow is a simple machine learning method also used in building models by taking inputs as multidimensional arrays. In both projects, I used an epoch of 10. An epoch is the number of times the computer will work through the training dataset. It is a hyperparameter, which means that it can be changed. I also looked at the validation and training accuracy and loss. The validation and training accuracy are accuracies based on the specific data set. Simultaneously, I determined the accuracy of the validation images and training images. Here are the results for PyTorch and TensorFlow:

	precision	recall	f1-score	support
Black Sea Sprat	0.91	0.85	0.88	321
Gilt-Head Bream	0.72	0.57	0.63	293
Hourse Mackerel	0.94	0.86	0.90	298
Red Mullet	0.98	0.99	0.98	305
Red Sea Bream	0.81	0.85	0.83	287
Sea Bass	0.69	0.86	0.77	290
Shrimp	0.99	1.00	1.00	314
Striped Red Mullet	0.85	0.97	0.90	301
Trout	0.94	0.85	0.89	291
accuracy			0.87	2700
macro avg	0.87	0.87	0.86	2700
weighted avg	0.87	0.87	0.87	2700

Figure 7. Results of Precision for Modeling

Table 2. Results For Modelling With TensorFlow

Epoch	Accuracy on Validation Images	Accuracy on Training Images

1/10	25.317	16.825
2/10	43.175	31.667
3/10	54.603	44.504
4/10	55.556	52.242
5/10	65.794	59.544
6/10	67.857	67.183
7/10	78.810	72.262
8/10	81.111	77.262
9/10	87.778	81.171
10/10	87.222	82.540

[1]All data are accurate with 5 significant figures. 3 decimal places have been used for the level of accuracy.

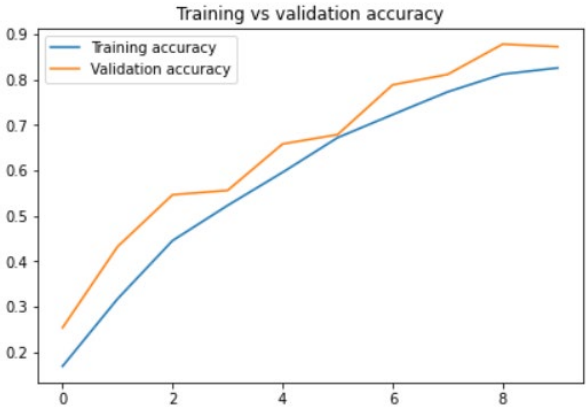

Figure 8. Results of Training vs Validation Accuracy

Figure 9. Results of Training vs Validation Loss

Table 3. Results For Modelling With PyTorch

Epoch	Accuracy on Validation Images	Accuracy on Training Images
1/10	98.025	84.733
2/10	98.642	97.901
3/10	99.136	98.861
4/10	99.506	99.053
5/10	99.877	99.204
6/10	99.630	99.314
7/10	99.877	99.424
8/10	99.753	99.588
9/10	99.630	99.506
10/10	100.000	99.739

[1]All data are accurate with 5 significant figures. 3 decimal places have been used for the level of accuracy.

Quality of a Model

To improve the quality of a model, one can add more layers in the neural network. As the number of layers increases, the complexity increases and error in the validation set decreases resulting in a more accurate prediction. More data can also be added to the training dataset for reduction of biasedness. Another method can be minimizing overfitting and underfitting.

In most data, the outputted labels are not 100% the predictions that will be expected. Underfitting is a term used when the model cannot capture a pattern of the inputted data, mainly due to the minimal amount of data being processed. One potential method to address this is by feeding in more data into the dataset. Again, the model can be made more complex by adding
more layers. Unlike this project, if there were to be some inconsistent data, missing information or a noise (meaningless/unnecessary data) created in the set, it would be recommended to remove them to prevent fluctuations in the accuracy.

On the other hand, overfitting occurs when $|y - \hat{y}| \ll \text{noise}$. When there is an overload of data fed into the model, categorization of the data becomes faulty due to an abundance of unnecessary data. The model starts to follow the pattern of meaningless data in turn giving a lower accuracy. As a solution, one might consider decreasing the overfitting by increasing the number of training examples, but this may not always be feasible. Instead, there are two possible methods that can be used:

Splitting of the training dataset further into training set and validation set

As seen in before, I split the dataset into training and validation. Unlike the training dataset which was used to train the model, the validation set remained unused for training and was thus better at revealing errors. In the validation set, a selection of various hyperparameters can be made to help minimize the error.

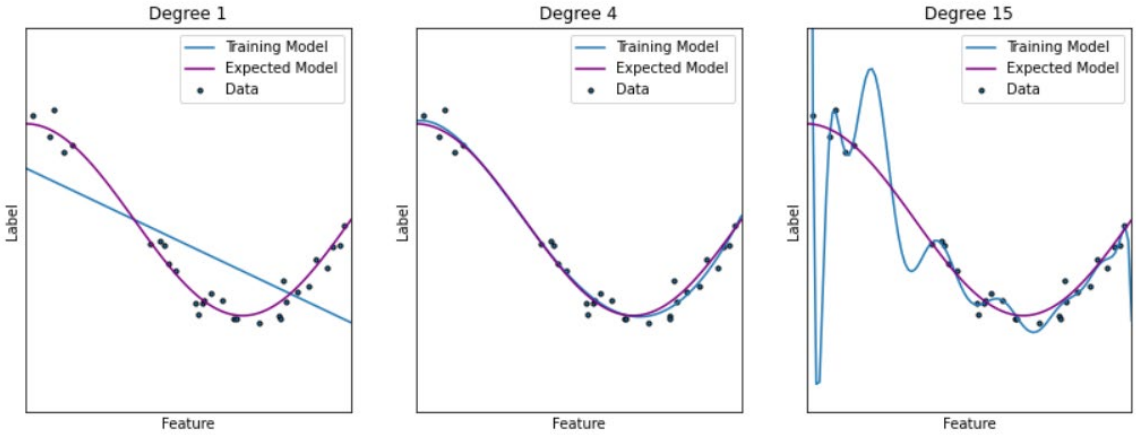

Figure 10. Underfitting and Overfitting with Different Degrees

From Figure 7, it is evident that Degree 4 gives the most accurate model where the error is the smallest.

Normally, the Mean Square Error (MSE) is a loss function which is used to minimize the error in a problem. The squared error is shown as $SE(\hat{y}, y) = (\hat{y} - y)^2$. The MSE is the average of all the squared errors and it can be denoted as

$$J_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y_n} - y_n)^2 \quad \cdots (5)$$

The smaller the error, the more accurate the predictions. I will not describe the algorithms used to find the error and hyperparameters as they are easily locatable through libraries like Keras.

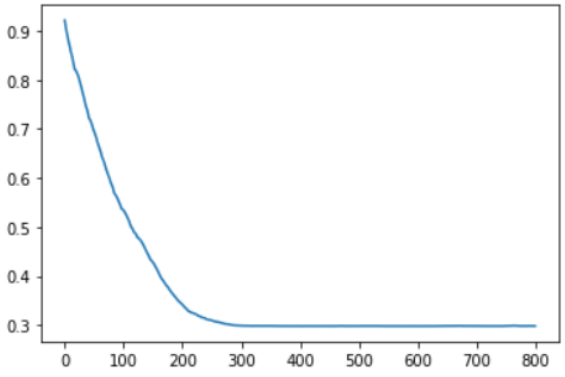

Figure 11. Mean Square Error

Regularization

Going back to Section 3.2.1, there are different parameters that are present in the function. In neural networks, they are called $w_{ij}^{[l]}$ and $b^{[l]}$ which minimize the error. Similar to Method 1, one may split the training dataset into a validation set and training set. Assume n to be the number of training examples (in this case, it was 5040 images or 7290 for PyTorch). The function for regularization is defined as:

$$J_{reg} = J + \frac{\lambda}{n}\Sigma(w_{ij}^{[l]})^2 \qquad \cdots (6)$$

Note that λ is a hyperparameter so its value can be altered to help find the smallest error in the validation set. While minimizing J selects parameters that attain accurate predictions on the training set, the addition of the second term in J_{reg} penalizes large values of the parameters $w_{ij}^{[l]}$ which smoothens the model. So, the minimization of the sum of both terms simultaneously tries to keep the predictions on the training set accurate and control the parameter's size.

Analysis of the Outputs

Accuracy of predictions is defined as the number of correct predictions divided by the number of total predictions made. Looking back at Table 2 and Table 3, I noticed that the accuracy obtained when using PyTorch was more precise and closer to 100%. With 99.75% accuracy on validation images, 99.84% accuracy on train images, and 99.67% accuracy on test images, I concluded that PyTorch appeared to be a more effective and precise mechanism for modeling large numbers of data. Since the accuracy was close to 1 in this case, the output may be correct. However, with TensorFlow, the precision of classification for a certain type of species like Sea Bass and Gilt-Head Bream could definitely be fixed with a better model. Having the same number of epochs, the validation loss in PyTorch was smaller in comparison to that of TensorFlow. TensorFlow also gave an accuracy of 87.2% so, possibly with an increase in epoch, the accuracy may increase. Having the accuracy of the validation set > accuracy of the training set indicated that there was no overfitting which was a good result.

Acknowledgments

I wish to express my sincere gratitude to my instructor, Professor Guillermo Goldzstein for his guidance all throughout the program. I would also like to thank my TA, Mr. Raphaël Pellegrin, and my family for their constant support.

References

GERON, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly.

Ulucan, O. (2021, April 28). *A Large Scale Fish Dataset*. Kaggle. Retrieved June 23, 2022, from https://www.kaggle.com/crowww/a-large-scale-fish-dataset

Manure, A., & Singh, P. (2020). *Learn tensorflow 2.0: Implement machine learning and deep learning models with python*. Apress.

Stevens, E., Antiga, L., Viehmann, T., & Chintala, S. (2020). *Deep learning with pytorch: Build, train, and tune neural networks using python tools*. Manning Publications.

Menard, S. W. (2010). *Logistic regression: From introductory to advanced concepts and applications*. SAGE.

Haykin, S. S. (2016). *Neural Networks and Learning Machines, 3d Edition*. Pearson.

Gulli, A., & Pal, S. (2017). *Deep learning with keras: Implementing deep learning models and neural networks with the power of python*. Packt Publishing.

G. Chen, P. Sun and Y. Shang, "Automatic Fish Classification System Using Deep Learning," 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), 2017, pp. 24-29, doi: https://doi.org/10.1109/ICTAI.2017.00016.

S. Liawatimena et al., "A Fish Classification on Images using Transfer Learning and Matlab," 2018 Indonesian Association for Pattern Recognition International Conference (INAPR), 2018, pp. 108-112, doi: https://doi.org/10.1109/INAPR.2018.8627007.

Xiu Li, Min Shang, H. Qin and Liansheng Chen, "Fast accurate fish detection and recognition of underwater images with Fast R-CNN," OCEANS 2015 - MTS/IEEE Washington, 2015, pp. 1-5, doi: https://doi.org/10.23919/OCEANS.2015.7404464.