Fish and Shellfish Classification using Convolutional Neural Networks

Mitchell J. Krystofiak

***Abstract*—In an effort to improve the efficiency of the aquaculture industry, UMCES and UMD have received funding from the USDA and NIFA to explore methods to incorporate advanced technologies in the practice while preserving ecosystems. One of many methods include using sensors to classify different types of aquatic species using underwater cameras and neural networks. Convolution Neural Networks (CNN's) are artificial neural networks primarily used in image recognition and processing. An image is entered as input and processed through layers that detect certain characteristics used to help determine what an image is. The CNN must be trained using a set of images with labels and then tested. Once the algorithm is complete, it will report on its accuracy. This project's main objective is to use CNN's to classify images of fish and shellfish using TensorFlow and Keras Python libraries. These libraries are fundamental to building and testing the CNN. This project also focuses on the applications of the Keras methods Tuner and Callbacks, which provide means of optimizing existing CNN's in order to build more accurate models.**

***Index Terms*—UMCES, UMD, USDA, NIFA, aquaculture, Convolution Neural Network, CNN, image recognition, accuracy, TensorFlow, Keras, optimization**

I. Introduction

Researchers at the University of Maryland Center for Environmental Science (UMCES) and The University of Maryland (UMD) have received funding from the United States Department of Agriculture (USDA) and the National Institute of Food and Agriculture (NIFA) to bring advanced technologies to the domestic shellfish aquaculture industry. The project aims to improve the bottom-culture of oysters and to bring about a major boost in the production by developing and incorporating these advanced technologies into shellfish farming [1]. These technologies include recent advances in the fields of robotics, agricultural automation, computer vision, sensing and imaging, artificial intelligence, and high performance computing. Another aim of this project is to develop an autonomous underwater vehicle equipped with sensors to help with the management of crops while having a low environmental impact [1].

Oyster and shellfish farming in the Chesapeake Bay region has continued to expand rapidly and, as a result, so has the seafood demand worldwide. These researchers seek to understand how these technologies make the production more sustainable and potentially even improve the environment [1]. With the aquaculture of shellfish being a strong driver of the coastal economy, the U.S. is lacking the basic technological advancements to meet the potential industry growth and expansion of nutritious protein source.

One practice that is currently used is known as dredging, where machinery drags a net across the bottom of an ocean, bay, or other body of water to scrape up buried shellfish. The process is highly damaging to the environment and destroys important habitats for oysters and other species [1].

Incorporating these advanced technologies would begin to enhance the productivity of the industry and profitability for both farmers and coastal economies while better protecting the fragile aquatic ecosystems [1].

II. Related Work

While there are many researchers that have contributed to the implementation of advanced technologies in aquaculture and other industries worldwide, one of the main motivations for this project comes from Timothy Riley, a student at the University of Maryland, Eastern Shore. His research focused in processing images of fish and being able to identify them using Convolutional Neural Networks (CNN) [2].

Dhruv Rathi, Sushant Jain, and Dr. S. Indu from the Delhi Technological University proposed a method to remove excessive noise in an image such as plants, dirt and non-fish bodies. The images were then processed using deep learning through the implementation of a CNN to classify several fish species and achieve a 96.29% accuracy [3].

III. Experimental Details

The first goal of this project was to build a CNN capable of classifying different species of fish. These images came from an online database known as Fish4Knowledge. There were a total of 5 fish species used for testing: Dascyllus reticulatus, Myripristis kuntee, Hemigymnus fasciatus, Neoniphone sammara, and Lutajanus fulvus. All of the images were assigned labels corresponding to the name of the fish. The images would be placed in directories labeled by species with a separate folder for training and testing. They would then undergo preprocessing, where they would be converted into NumPy arrays and normalized. NumPy is a powerful library capable of performing mathematical operations for scientific computing involving multidimensional arrays and matrices. NumPy arrays offer large amounts of flexibility and customization ideal for processing them through a CNN. Then, they would be loaded into the CNN. The CNN uses the training set to train the network, and then tests the testing set to confirm or deny it's predictions. Each of these training steps are known as epochs. After each epoch, it uses it's correct and incorrect predictions to update the weights of the network. The weights focus on the features of the images, such as the detection of edges. Finally, when the CNN completes the number of epochs specified, it reports on it's final training and testing accuracy.

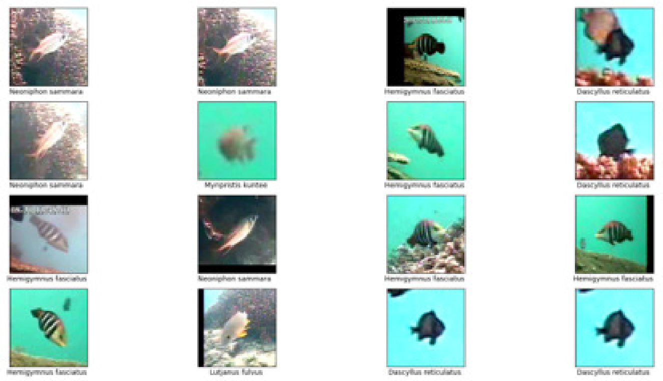


Fig. 1. Shuffled fish images of the 5 species tested.

The second goal of this project was to explore options available to optimize the CNN. When it comes to building these networks, it is difficult to know exactly how many layers to use for image processing, and it is also difficult to fine tune the parameters of these layers. The Keras library has methods called the Keras Tuner and Keras Callbacks. These methods are very useful for this particular goal. The Tuner is capable of running the CNN training sessions with a range of values instead of just one. For example, the hp.Int() method allows the user to set a minimum and maximum of values to test, and the hp.Choice() method allows the user to try different listed values. The Tuner will use a value in between, depending on the instantiation method chosen. This project will use the RandomSearch instantiation of the tuner, which randomly picks a value from either of the two methods. In theory, the Tuner will run through every option available, or until the number of trials is satisfied and report the highest resulting combination of values for the CNN [4].

In some instances, the act of training the CNN is difficult to visualize. Keras Callbacks is a useful method for being able to look at what is actually happening. There are many callbacks to implement, but the ones being used are Early Stopping, CSV logging and Model Checkpoint. Early stopping allows a user to set a value to monitor with a patience level. For example, the value being monitored could be validation loss and the patience could be 3. Patience is the number of epochs, or training sessions for the CNN, that can go by without seeing change. If there is no reported improvement, it will stop the CNN training process. CSV logging simply formats the results of the CNN training into a chart format. Model Checkpoint documents each stage of the process, depending on what the save frequency is set to. There are other methods under Keras Callbacks, but these are some of the main methods that will be used for CNN optimization [5].

The third goal of this project was to build another CNN capable of making a binary classification between a shellfish (oyster) or a fish. The aquaculture industry is heavily reliant on harvesting different shellfish, including many different types of crabs, clams and oysters. When building a sensor to detect different organisms underwater, being able to tell one organism

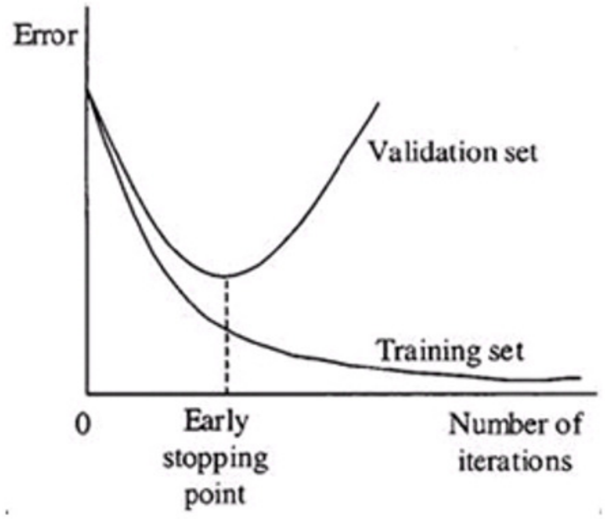


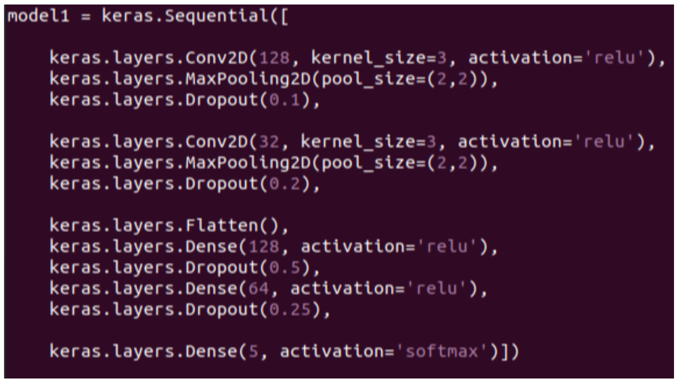
Fig. 2. A graph illustrating the purpose of Early Stopping.

from the other is a must. This part of the project aimed only to tell the difference between fish and oysters. The fish images used are a combination of the images described above and the oyster images come from a Google Drive collection.

Fig. 3. Example image of oysters to test.

IV. Results

Before the first model was trained and tested, the images from the 5 different fish species were loaded in and preprocessed. This included resizing the input images to 128x128 pixels, converting the images into a NumPy array, shuffling the images and normalizing them by dividing them by 255, because they are RGB images. These images were then put through the CNN created by using the Keras Tuner. This CNN consisted of two convolutional layers, two max pooling layers, four dropout layers and two dense layers. The Tuner optimized that the first convolutional layer should have 128 filters and the second should have 32. The CNN would run and then test the trained model, plotting the validation accuracy and training accuracy.



```
model1 = keras.Sequential([

    keras.layers.Conv2D(128, kernel_size=3, activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2,2)),
    keras.layers.Dropout(0.1),

    keras.layers.Conv2D(32, kernel_size=3, activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2,2)),
    keras.layers.Dropout(0.2),

    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dropout(0.25),

    keras.layers.Dense(5, activation='softmax')])
```

Fig. 4. CNN Model after using the Keras Tuner.

The second model did not make use of the Keras Tuner, and instead used a similar structure with two convolotional layers, two max pooling layers, a flatten layer and a fully connected dense layer. The main difference in this model was the images it was comparing and the way they were prepared. Since there were several problems in the first model, this model aimed to augment the images before processing them. The training images were normalized and then augmented with shearing, zooming and horizontal flipping. This would produce a larger data set with expanded angles. They were also resized to 120x120 pixels.

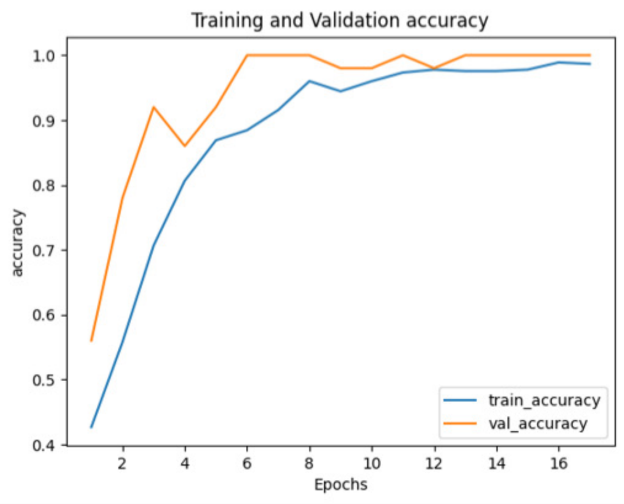


Fig. 5. Training and Validation Accuracy of first model.

The first model was then trained and tested to achieve varying results between %96 and %100. On some of the trials run, the program did end before all 20 epochs were completed as a result of the Early Stopping callbacks not seeing a significant improvement for 3 epochs. The second model also achieved unusually high results immediately after starting the training process. It was not able to correctly classify an image for testing after the model had trained.

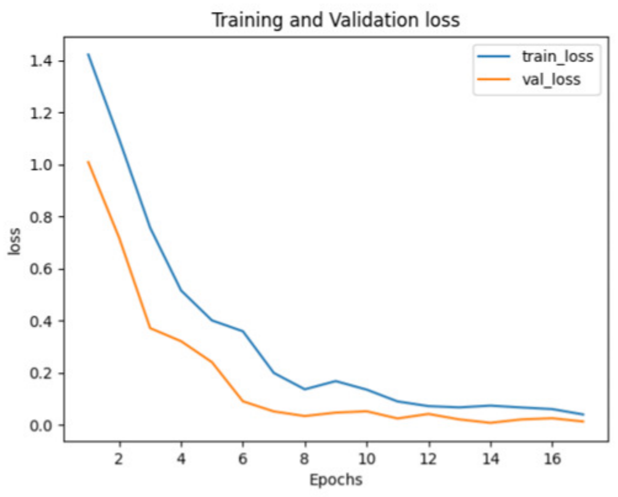


Fig. 6. Training and Validation Loss of first model.

V. Conclusions

In the training and testing phase of the CNN for the fish images, the validation set's accuracy and loss is higher than the training sets. In theory, this result normally wouldn't occur. The CNN should train better on data it has seen numerous times before and evaluate a new set of images less accurately. One of the possible causes of this result is the way the training data and testing data are separated. The way they are created could be adding bias to the training and testing phases. One way to test this is to implement the k-fold cross validation technique. Cross validation is a procedure used to evaluate models on limited data samples. The method involves shuffling the data set randomly, splitting the data set into k groups, separating out a test data set and training using the rest and retaining the evaluation score and discarding the model for each k iteration. This means that each sample is given the opportunity to be used in the testing hold out set once and used to train the model k-1 times [6]. Another solution to the validation accuracy issue is to use more image augmentation techniques to better prepare the model to test each CNN. For the first model, there approximately 200 images for training and 20 images for testing each species. For the second model, there are about 300-400 images for training and 30-40 for testing. This would constitute a small data set.

With the second model, there appears to be an error in the preprocessing stage of the CNN. Since the images are rather different in size, bring the oyster images down from 600x700 sized images (approx.) to a size below 200x200 appears to alter the key features of the image. Further research needs to be conducted, but some researchers have explored zero padding and interpolation with no improvement on accuracy [7].

References

[1] "Research team receives $10M to transform shellfish farming with smart technology," University of Maryland Center for Environmental Science, 29-Jun-2020. [Online]. Available: https://www.umces.edu/news/research-team-receives-10m-to-transform-shellfish-farming-with-smart-technology. [Accessed: 22-Jul-2021].

[2] T. Riley, "Classification of Fish Species using a Convolution Neural Network," May 2021.

[3] D. Rathi, S. Jain, and S. Indu, "Underwater Fish Species Classification using Convolutional Neural Network and Deep Learning," 2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR), 2017.

[4] C. Conol, "Hyperparameter Tuning with Keras Tuner," Medium, 02-May-2020. [Online]. Available: https://towardsdatascience.com/hyperparameter-tuning-with-keras-tuner-283474fbfbe. [Accessed: 22-Jul-2021].

[5] B. Chen, "A Practical Introduction to Keras Callbacks in TensorFlow 2," Medium, 09-Oct-2020. [Online]. Available: https://towardsdatascience.com/a-practical-introduction-to-keras-callbacks-in-tensorflow-2-705d0c584966. [Accessed: 22-Jul-2021].

[6] J. Brownlee, "A Gentle Introduction to k-fold Cross-Validation," Machine Learning Mastery, 02-Aug-2020. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/. [Accessed: 22-Jul-2021].

[7] Hashemi, M. Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation. J Big Data 6, 98 (2019). https://doi.org/10.1186/s40537-019-0263-7