Classification of DIDSON Sonar Images with Convolutional Neural Networks

Brandon Bicknell, brandon.bicknell@student.nmt.edu

Wing Ho Lin, wingho.lin@student.nmt.edu

Jerry Tafoya, jerry.tafoya@student.nmt.edu

Wesley Pick-Roth, wesley.pick-roth@student.nmt.edu

Abstract

In this paper, a dataset of DIDSON Sonar Images is examined and explored, with the intentions of implementing a Convolutional Neural Network model to classify with. A model using a VGG16 implementation and edited images was used to set a benchmark for later models. Using Laplacian of Gausian to preprocess images, a custom CNN was utilized to classify the data. The CNN was unstable, but promising, so it was utilized for the K Rolling Prediction Average method to try and draw out time series relations. Where the CNN was stable, this method provided useful convergences that didn't make the classifier overly rigid were it to be implemented as an observational classifier in reality. Ultimately, the research resulted in clear paths of improvement for the models developed during it.

1. Introduction

Underwater Classification is a field with numerous techniques, models, and equipment utilized for a variety of datasets. The goals of this paper were to explore wildlife classification. One of the most common forms of underwater imaging is sonar cameras, since normal cameras require ideal conditions to be utilized in an underwater setting.

However, Sonar images generally suffer from an abundance of noise and the lack of crisp visual detail as seen with normal cameras. As such, images would need to be preprocessed, and a classifier will have to be properly tuned to capture the sparse information of the object in question.

Once a dataset of Sonar Images was obtained, large amounts of work must be put in to determine how best to preprocess it. How the dataset is utilized is also important. Considerations must be made for building CNN's to classify it as individual images, or to take considerations of the dataset having time series relationships between the images.

2. Related Works

The desire to see more work in sonar classification led to the research expeditions which built the dataset used in this paper. McCann [1] created multiple datasets that are either annotated or raw images, in an attempt to ensure that there is enough data for research in this topic to continue.

The datasets were verified using video cameras and a PIT system. The fact that the Pit system was unavailable for the 2016 expedition meant that the sea lamprey observed with it in the 2013 one, could not be observed using video cameras as they were nocturnal. Trained experts manually reviewed the dataset. In a testing of the expert's accuracy, he was found to be 95% accurate with a subset of the 2013 dataset.

One of the datasets constructed was the SequenceFile Dataset, which is an alternative representation of sonar recordings aside from an image. This is a binary representation, that is converted from the three dimensional arrays (Width, Height, Frame) of other sonar clip files. The 3d array is segmented into sets of 200 frames with a key-value pair, which is stored as a flattened

array with a header. Each record overlaps 15 frames with the previous and next record, allowing algorithms that take advantage of the overlapping for distributed processes.

In terms of Image Processing, denoising Sonar Images is a well explored field. In Modalavalasa's paper [3], several techniques are discussed.

Spatial Filters such as average and maximum filters are used to smooth out a pixel using its neighbors. Mean Filtering is also a simple smoothing method considered, generally 3x3 to prevent severe smoothing. It also is intuitive and easy to implement. Median Filtering is a similar Average Filter to Mean, but it generally preserves useful details in the image better.

Gaussian Smoothing blurs an image using a Gaussian hump to assign weight to a pixel, which in most implementations goes to zero after three standard deviations from the kernel. This distribution matches with many real world behaviors, which is why it is used so commonly.

Laplacian Filters of a Gaussian Smoothed image highlights rapid changes in intensity, making it very useful for edge detection.

His work with sonar images preferred Gaussian and Medium filters, though the images used are more conventional sonar images from a radar, rather then a high intensity camera.

3. Dataset Analysis

The dataset being used is a series of DIDSON Sonar images collected at the Ocqueoc River in Michigan [1], with the goal of creating a large labeled dataset in which to classify seven fish. Two ventures were made to construct this dataset, one in 2013, and one in 2016. For this paper, the clips from the 2016 venture will be considered.

The 2016 section of the dataset contains 95 captures each consisting of 12411 frames of still images. Each of these images are 400x648 in size.

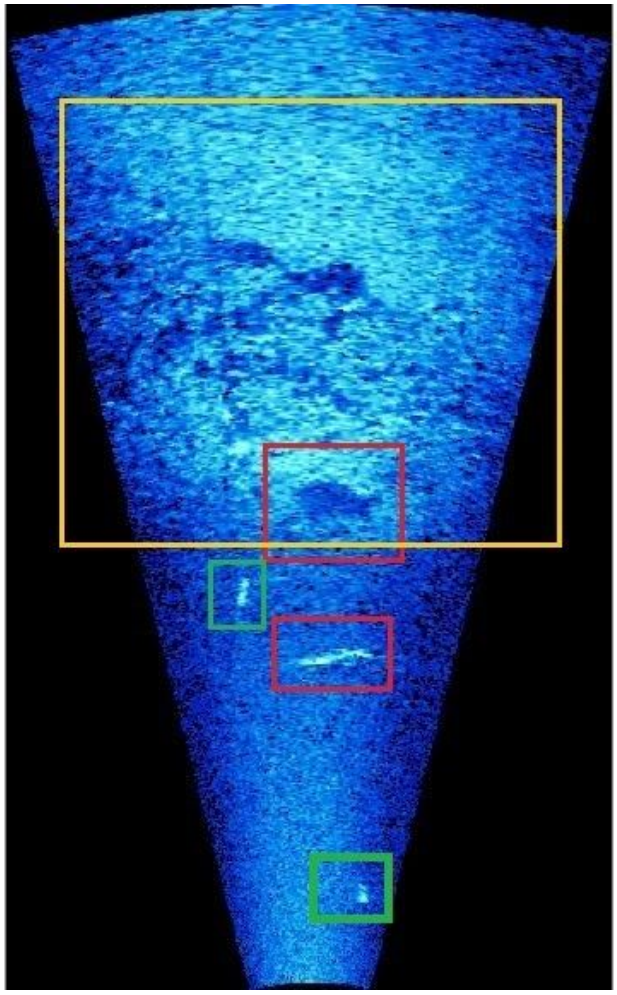

Figure 1, Annotated DIDSON Sonar Image in Dataset

The images contain 2 main areas, the upper area, marked in yellow, is excessively noisy, but is also the area where the shadow of the fish is contained. The bottom area is generally clear, though marked in green are examples of persistent noise that remain over the course of entire days of recording. Though in this example, the fish, marked in red, is in the cleaner lower area, the overall dataset contains fish swimming through the noisy area as well, making them hard to detect, even with the human eye.

3.1. Use of the Dataset

Using a DIDSON Reader [Resources 2], each clip can be opened and processed for our needs. The researchers through use of external devices, equipment, and on-site experts whose direct data is not available to us, identified and classified fish seen by the camera and recorded this information in an excel spreadsheet.[1] With the excel spreadsheet the researchers provided, the clips were all labeled or segmented into labeled subclips to be used.

Based on examinations of the visibility and availability of fish in the dataset; Carp, Suckers, and Bass were deemed the most appropriate fish to focus efforts for this research. They were the largest and most prevalent fish in this dataset.

3.1.1. Single Image Dataset

By referencing the excel spreadsheet and using the reader to explore each frame, the frames in which a fish appeared were determined and the clearest frames were utilized for the dataset. Each of these images could be labeled, allowing for full supervised learning of the dataset.

Through this technique, the dataset was created, consisting of 101 Bass, 114 Carp, and 100 Suckers. The primary goal of creating this dataset was to determine the usability of single image classification, as well as image preprocessing techniques to improve classification.

3.1.2. Image Clip Dataset

In addition to the above dataset, a second dataset was made with the goal of applying time series methods to. By taking the .ddf clips from the original dataset, a folder of ordered images from each clip were generated.

Seven clips were generated, most consisting of over 100 images, with some ranging up to 400. There were two for Bass, two for Carp, and three for Suckers. The process creates these images as grayscale, so the training of the model will differ from the Single Image Dataset.

4. Single Image Dataset Convolutional Neural Network

To properly handle the dataset we constructed, multiple Neural Networks were created to test with. To meet the needs of these models, the dataset was also edited to properly work with it.

4.1. VGG16

This model is a publicly available design that was chosen to weigh against the custom models later, to determine a benchmark to improve off of. The main goal of tuning this model was to adjust the Steps and Epochs to determine the best balance of settings. For the dataset, as Figure 2 shows, the best results were seen at around 8 Epochs and 14 Steps.

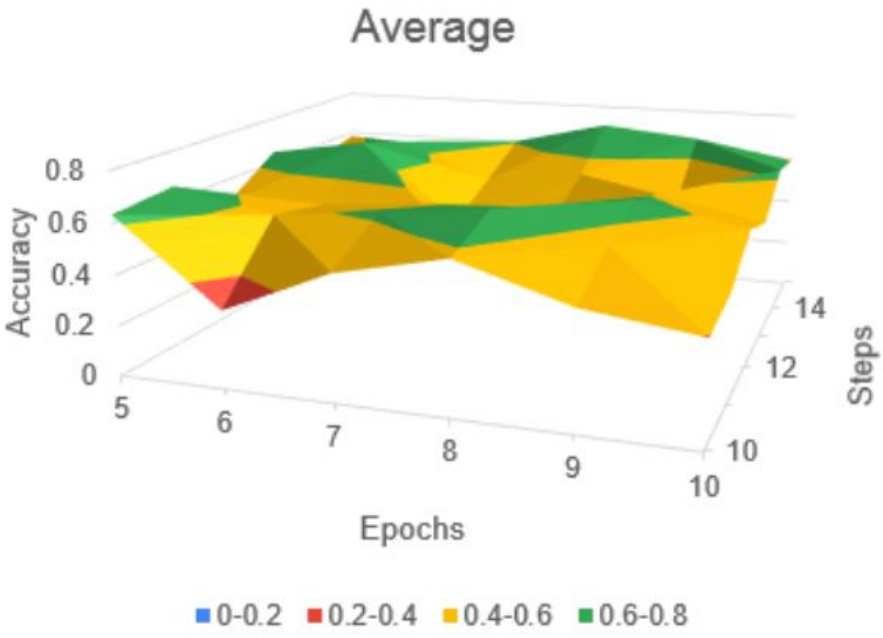

Figure 2, Testing Graph

4.1.1 Dataset Editing

Since VGG16 preferred squared images, the dataset was cut to give greater focus on the fish in the image. This was a manual process, and during which, the dataset was also further subcategorized. The categories were: clean, meaning that the fish was clearly visible in the image; cluttered, meaning that multiple fish were together in the image; and noisy, meaning the fish was within the yellow area marked in Figure 1.

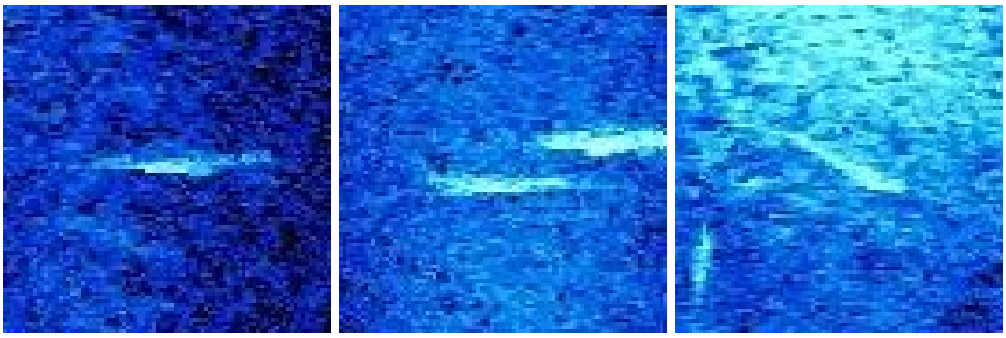

Figure 3, Clean (left), Cluttered (center), and Noisy (right) edited images

4.1.2 Results

Five testing trials were performed using the tuned model, to determine the consistency of the model. The score statistics were graphed to visualize the performance, while the specific true negative and true positive rates, accuracy and recall, were averaged into a table to draw more conclusions from.

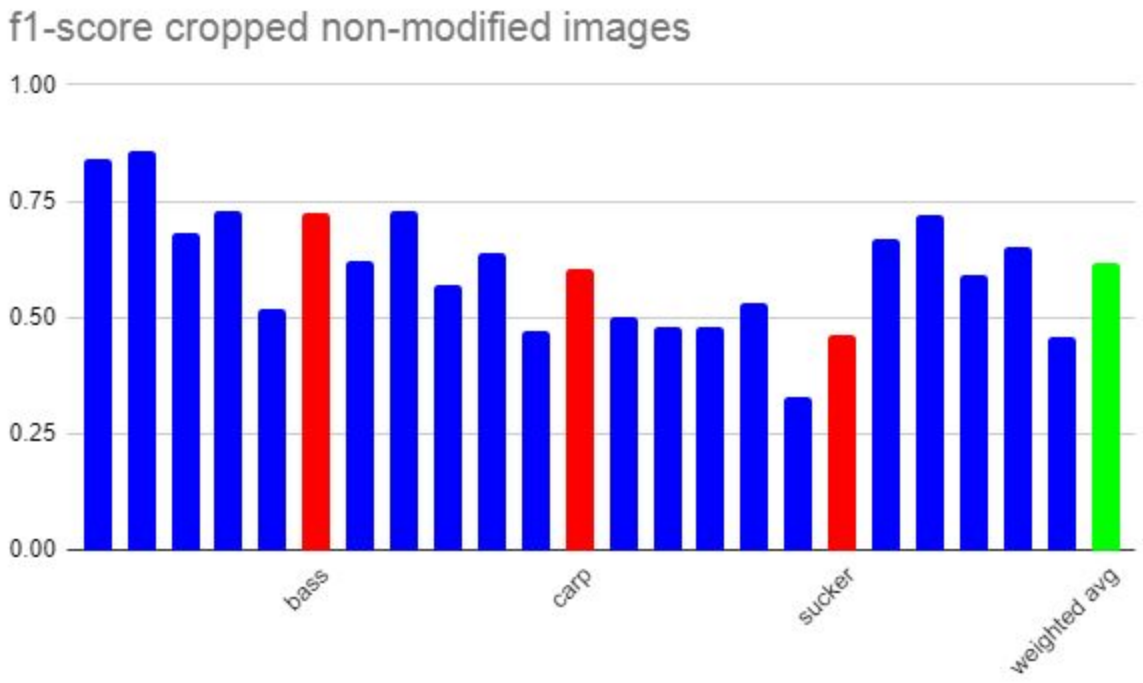

Figure 4, Results Chart

Fish	Precision	Recall
Bass	.74	.74
Carp	.58	.65
Sucker	.64	.4
Total	.65	.62

Figure 5, Results Table

Interpreting the results seen in Figure 2, the model was best able to classify Bass, while Carp and Suckers both had subpar results. The model was particularly bad at avoiding false positives for Carps, while with Suckers it gave many false negatives.

Ultimately, considering the model as a whole, it was moderately subpar for use in any serious application.

4.2. Custom CNN for Unedited Images

Since VGG16 required manual editing of the images, a different CNN was constructed to make use of automated image preprocessing. The CNN needed to work with grayscale images, since the automation generation of dataset clips outputted the images in that format. The images needed to be preprocessed, then pushed through the new network, so that the results can be verified and put to use for other methods.

4.2.1. Image Preprocessing

After considering numerous techniques, Laplacian of Gaussian was selected as the best method to increase the accuracy of the CNN. [Reference 3, Resource 14.15]

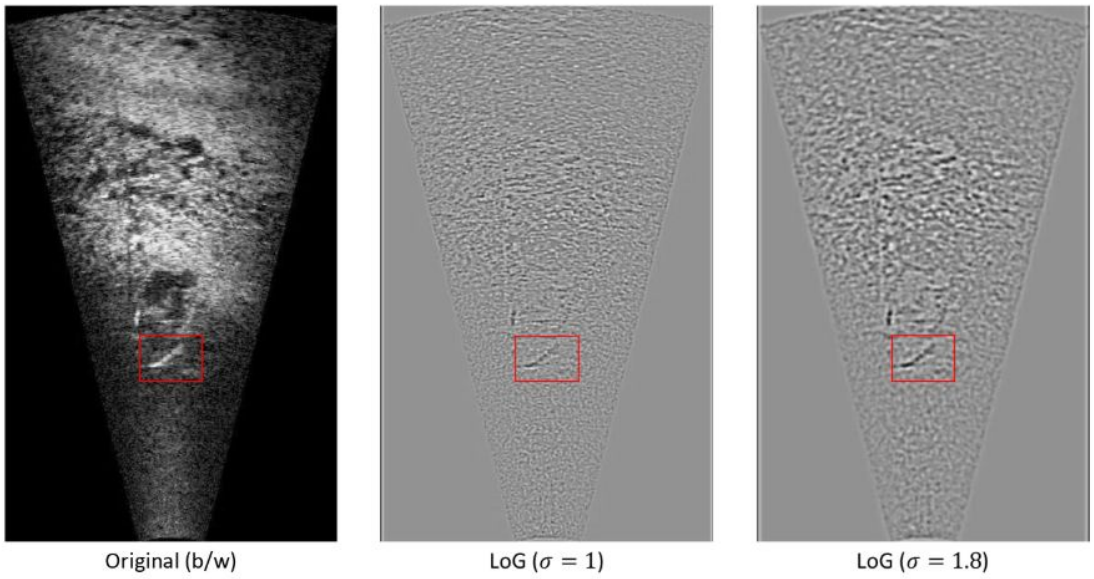

Figure 6, Sigma Contrast Chart

The method is to apply Gaussian Smoothing in order to apply a Laplacian Filter. This Filter is very useful for edge detection, which can draw out the fish from the rest of the image,

allowing for better classification. However, as Figure 6 displays, the Sigma value is very important to balance drawing out the fish, and drawing out the noise in the image.

4.2.2. Model Tuning

The CNN had 6 layers: Flatten, Dense, Dropout, Dense, Dropout, Dense. This was decided through following a general trend where models that used more layers had prediction values tended to be more evenly distributed among the categories. This is expected to have been the result of multiple, simple layers causing features from the images to become trivialized, especially since the fish in the image occupy a relatively small space relative to the rest of the image. The model in this paper, though initial testing showed slightly inconsistent results, avoided converging onto one prediction, so tuning was more likely to result in good results.

The Dense layers were given shape values of 100 and 50, respectively. The Dropout layers had the overfitting handling rate set to .3 and .5, respectively. Predictions were stored in files to be compared with predictions made with a differently tuned model or images with different preprocessing techniques.

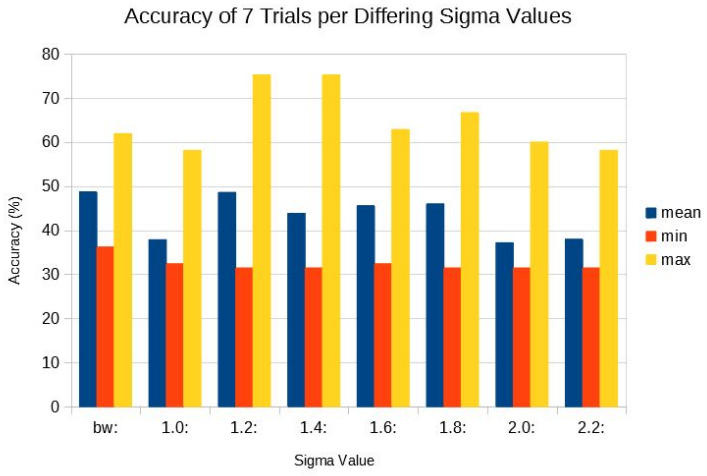

Figure 7, Sigma Trial Chart

An extensive test was done with the model to determine the best sigma value for the Laplacian Filter. As Figure 7 shows, images with Sigma transform values of 1.2, 1.4, and 1.8 showed the best maximum results. Combined with further tuning of the models, 1.2 was selected to have the greatest potential of giving a good classification.

To improve the results and speed up the retraining of the CNN, a bottleneck feature was implemented. [Resource 5] A bottleneck feature is a final, fully connected layer from another pre-trained model that can be used by the CNN to more easily identify certain features from a dataset. In this case, VGG16 was used.

4.2.3. Results

Multiple trials were performed with the intention of finding a consistent pattern. Eventually, the Network was able to gain semi consistent results, where two categories were properly classified with good accuracy, while one usually suffered.

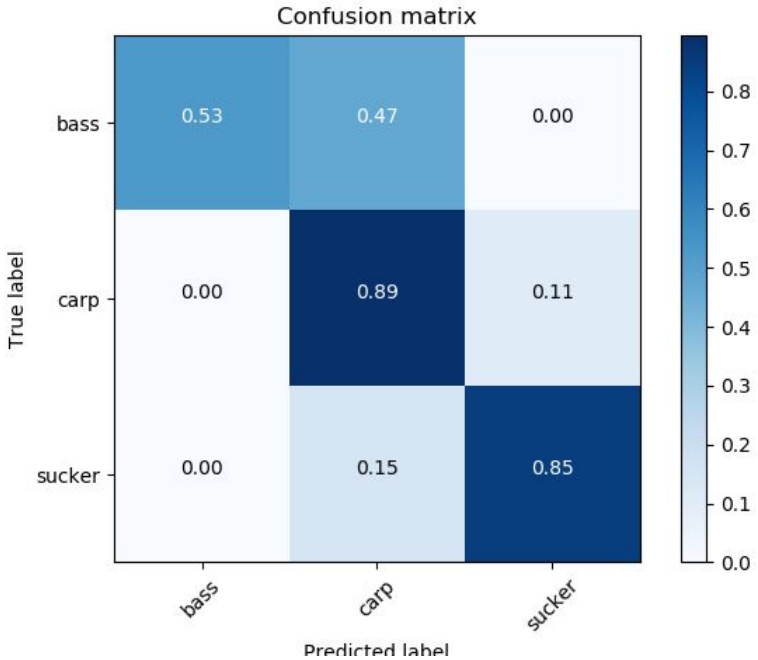

Figure 8, Confusion Matrix

Depending on the testing set, the accuracy of the test varied between 60 and 80%. However, there were some catastrophic sets where the model would only predict one category, resulting in a worst case accuracy of 33%. Through tuning, these cases were reduced, but not eliminated completely. Ultimately the model remained unstable and inconsistent, though the results were deemed useful enough for use in further methods.

5. Time Series Methods

Since the dataset exists as a series of video clips stored in .ddf format, making use of the inherent relationship between contiguous images would likely result in better classifications. Of the methods considered, the one that best works with the models already constructed was selected.

5.1. K Rolling Prediction Average

A single frame method that utilizes the conception that the model will show a trend of preference towards the correct classification that can be converged to through the use of averages. This is also balanced with how long an individual frame classification with the overall clip classification.

The algorithm takes a CNN trained for individual images, and predicts each frame in a clip of images. This can be done dynamically, as each frame is recorded, or for the sake of testing, the clip is already finished. Starting from the oldest classification, take the average of it and every previous one. This is the Rolling Average. Once the number of classifications being averaged exceeds the parameter value K, no longer consider the oldest classification in the average consideration. The stream of classifications from this algorithm should ideally converge

to one consistent classification, provided that the image clip is indeed one object throughout its duration.

K determines how long it takes for the model to classify against the current prediction before the Rolling Average changes.

5.2. Results

The model was run on seven clips, with multiple trials. However, due to the inconsistency on the base model, some of the clips did not give results that would be useful for this experiment. There were enough useful results though to draw conclusions on the nature of the method with the dataset.

K Value	K % of dataset	% of Bass	% of Carp
5	1.6	84.4	15.6
10	3.2	87.6	12.4
15	4.8	90.4	9.6
20	6.3	94.9	5.1
25	7.9	99	1
30	9.5	99.7	.3

Figure 9, Rolling Prediction Ratio Table

It was decided that the best way to determine classification convergence through K Rolling Average was to record every Rolling Average Classification at each frame of the clip, and get the ratio of classification for each category. Figure 9, performed on a clip of a bass, showed that increasing K would allow a quicker convergence to the consistent classification of bass. Once K reached 30, it had converged, and no further increase in K would change the ratio.

It is worth noting that if a clip is misclassified, it will converge to that misclassification, so the model must still be relatively accurate. However, if it has the behavior that more often than not it classifies an image correctly, then at a large enough K it would converge, even if the model is only moderately above random guessing.

As such, the fact that K converged at only 10% of the size of the dataset is rather promising, as it means the method isn't over sampling the data to draw its conclusions. This is also important because in a field implementation, a large K would mean the classifier is very unresponsive to sudden changes, such as a fish leaving the camera's view and a new fish entering.

6. Conclusion

In conclusion, the dataset proved to not be so inhibative that a model can't be produced for it. By focusing on single images, the models were able to achieve relatively acceptable results. However, the best performing model was still inconsistent, meaning that the current methods and models are not sufficient to be considered successful classifiers for it.

Adding in simple Time Series considerations as a post processing measure to the model were useful, allowing the possibility of a real time classifier. However, this method also fails to consider how the movement of a fish may allow better classification by the model. As such, there can still be further improvements made to the models in order to better utilize these untapped relations.

7. Future Work

As alluded to in the conclusion, the lack of full consideration of Time Series data inhibited the progress of the research. As such, a few improvements can be made with the current models, as well as new potentially better models.

As far as the work that currently exists, a complete and comprehensive analysis of the dataset, as well as significant testing on image preprocessing techniques would hopefully create images that would be ideal for use in a single frame CNN. The current CNN can also be edited more extensively, to determine the full combination of parameters and how they affect accuracy.

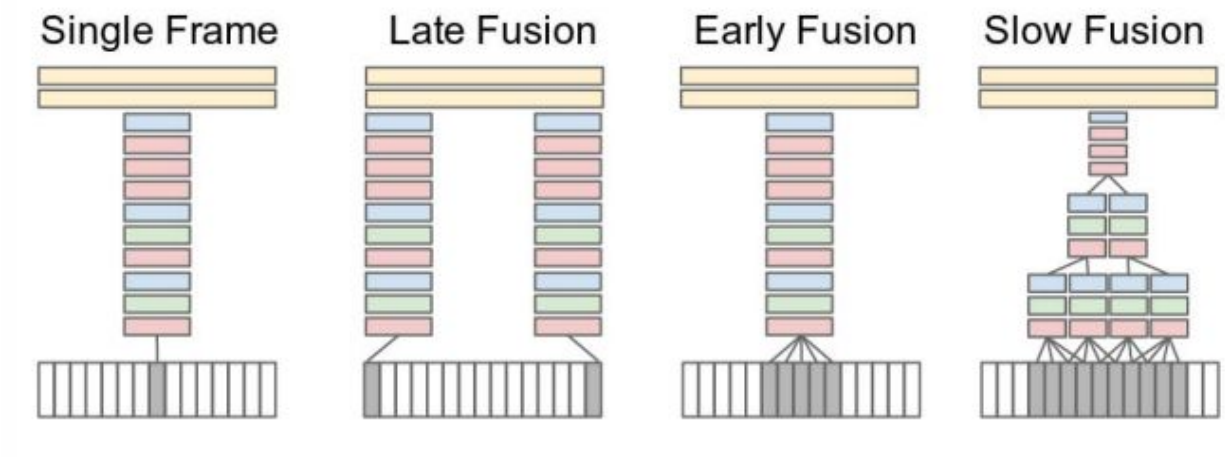

Figure 10, Time Information Fusion Graphic [2]

In terms of Time Series models, one that utilizes Time Information Fusion would be among the first to test with. As Figure 10 shows, all of the work in this paper only considered Single Frames. However, fusing the frames would perhaps expose some of the time series relationships that exist in the dataset.

Implementing Early Fusion would likely be the simplest, though Slow Fusion could possibly be the most useful, though the model implementing it would be very complicated and

finely tuned. Late Fusion is also a consideration, though clip size would need to be considered as another parameter, meaning testing would behave quite differently for it compared to the other methods.

8. References

1. McCann, E., Li, L., Pangle, K. et al. An underwater observation dataset for fish classification and fishery assessment. Sci Data 5, 180190 (2018). https://doi.org/10.1038/sdata.2018.190
2. Karpathy, Andrej. Large-scale Video Classification with Convolutional Neural Networks. 2014. Stanford University. https://cs.stanford.edu/people/karpathy/deepvideo/deepvideo_cvpr2014.pdf
3. Modalavalasa, Nagamani. A Quantitative Evaluation of Various Spatial Filters For Underwater Sonar Images Denoising Application. 2012. International Journal of Engineering. http://annals.fih.upt.ro/pdf-full/2012/ANNALS-2012-1-06.pdf

9. Resources

1. DIDSON Dataset: https://doi.org/10.6084/m9.figshare.c.4039202
2. Code to extract images from .ddf files: https://github.com/nilsolav/ARISreader
3. Basic CNN Tutorial: https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5
4. Explains usefulness of Dense and Dropout Layers: https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python

5. Explanation on Bottlenecking: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
6. CNN for CIFAR-10 Dataset: https://keras.io/examples/cifar10_cnn/
7. Watershed Algorithm: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html
8. Thresholding Algorithm: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html
9. Felzenszwalb and K-Means Algorithms: https://scikit-image.org/docs/dev/api/skimage.segmentation.html
10. VGG16 with Multiclass Variables: https://medium.com/learning-machine-learning/multi-class-fish-classification-from-images-with-transfer-learning-using-keras-335125637544
11. Sklearn Library for displaying results: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html
12. VGG16 Implementation: https://www.pyimagesearch.com/2019/06/24/change-input-shape-dimensions-for-fine-tuning-with-keras/

13. Introduction to Image Segmentation Techniques: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/
14. Explanation of Laplacian Filter: https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm
15. Explanation of Gaussian Smoothing: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm