



UiA University
of Agder

Fish Face Identification with Siamese Networks

Final deep learning project

by

Eivind Aamodt

in

IKT450

Deep Neural Networks project

Faculty of Technology and Science
University of Agder

Grimstad, November 2020

Contents

1	Introduction	1
2	State-of-the-art	2
3	Method	4
4	Results and discussion	6
4.1	Results from all the different runs	6
4.1.1	First run - face - left and right as same class	6
4.1.2	Second run - face - left and right different class	7
4.1.3	Third run - body - left and right as same class	8
4.1.4	Fourth run - body - left and right different classes	9
4.2	Comparing the differing runs	10
5	Conclusion	11
	References	11

1 Introduction

Havforskningsinstituttet is currently developing facial recognition to easily distinguish between individual fish, specifically corkwing wrasse, to track fishes throughout their entire lives. Different fish may look similar at first glance, but their patterns are different. They are currently physically tagging all of the fish, which is a very tedious process, and it can hurt the fish in some ways.

"To be photographed in the sea is not stressful to the fish, and you can photograph them on board a boat with minimal stress. The alternative is to continue as we do today by stunning the fish and tagging it to be able to recognise it later." - quote from Knutsen Sjørdalen[1].

Facial recognition for the fish will open up many possibilities regarding marine life research because it will aid the process of analyzing a staggering amount of data in a new way. Automating tedious and manual tasks is a very significant asset, as it is possible to upscale it easily. Analyzing millions of data points would be impossible to do manually, but it is only a matter of waiting for it to run through all the data if it gets automated.

UiA recently released an article on this project on their website linked here[1]. The article goes into more detail about why it is such a big deal and why it would help them out, but I will summarize the main points.

For humans, face perception is a very trivial task from birth. Newborns (1–3 days old) have been shown to be able to recognize faces even when they are rotated up to 45 degrees[2]. Being able to recognize faces is a neurological mechanism that humans use every day to distinguish between different people, pets, et cetera. So why is this task, which is seemingly so easy for humans, hard for computers? The answer to that is mainly lack of data. For Artificial Intelligence (AI) to train, there needs to be a lot of data. Having one single picture of a person is not enough data for it to recognize that person by normal means. Yes, it will recognize the person if it sees it at the exact same angle. But if the person looks to the side, or if another person looks similar to the first picture, the AI will struggle to classify it correctly[3].

This is where One-shot learning and Siamese neural networks comes in:

- One-shot learning aims to use only one or a few training samples to learn information about different categories. For example, normal classification uses a lot of data to output probabilities of each different output. This requires that the dataset you train on is similar to the dataset you test it on. With One-shot learning you only need one example from each class you want to predict. One classic use case is facial recognition on phones today[4].
- Siamese neural networks aims to find the difference between two inputs. This is done so you can use the difference between two inputs, together with different distance thresholds, to determine if it is the same person or someone completely different. Siamese networks makes it possible to train a model with a tiny number of images per class[5].

I will use these concepts to make a working Siamese neural network on the fish images they have

sent me. In total, the dataset contains around 4000 images of 500 different fish. There are, on average, four pictures of their head and four pictures of their body.

2 State-of-the-art

Siamese networks is a relatively new AI method that is being used. It is largely based on Convolutional Neural Networks (CNN). CNN is one of the most common ways to enable computer vision. It is used to enable computers to do image analysis, recognition, and classification through deep learning. The CNN deep learning algorithm takes an input image; then, it assigns importance to the different information from the image through weights it got from learning[6]. The information, such as color identification, shape identification, et cetera, is then used to differentiate classes from each other. Below is a figure of a traditional CNN: Figure 1.

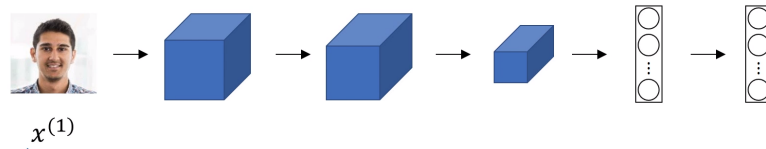


Figure 1: Traditional CNN[7]

In CNN's, the output would be probabilities of all the different classes. The highest probability would typically be the classification by sending the feature vector through a softmax or argmax function. It could, for example, have an output like:

- human: 75%
- dog: 18%
- cat: 7%

and then the output classification would be a human because it is the highest one[6]. Siamese networks use most of these techniques, but it does the ending classification part differently. Instead of passing the feature vector to the softmax or argmax function, the Siamese network sends two different images and generates two feature vectors[8]. The CNN weights must be the same for both pictures when making the feature vectors, so it is normal to train only one CNN and then send two images through the same one like so: Figure 2.

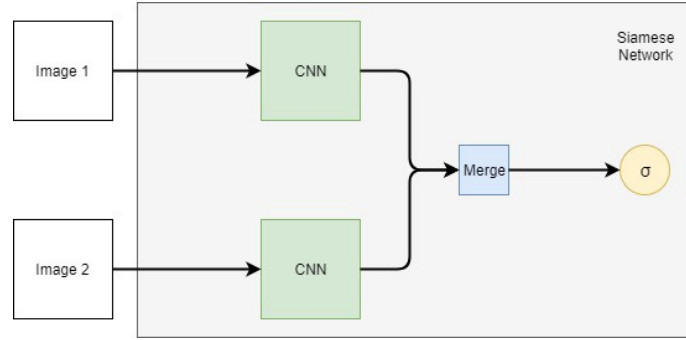


Figure 2: Siamese Network Architecture[9]

After sending two different images through the CNN, two different feature vectors are generated. These two vectors get compared by calculating the Euclidean distance between them[5]. The CNN weights should be trained to where similar input images give a low distance, while different input images give a high distance between them. The distance is then used to evaluate if it is the same object by using different distance thresholds. Training of Siamese networks is done with triplet loss or contrastive loss. This is done by having a reference image and then sending one positive image and one negative image, as shown in Figure 3. The goal of the entire learning process is then to put the positive image as close to the reference as possible and the negative image as far away as possible.

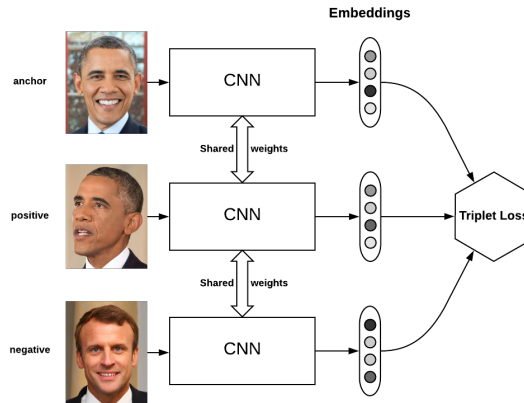


Figure 3: Triplet loss[10]

This architecture has been used a lot on face recognition, for example on an AT&T database of faces, but i could not find any other place where it has been used to identify fish. I did find someone who used CNN to classify between fish species, for example here[11] and here [12], but no one that has managed to identify fish especially from the same specie.

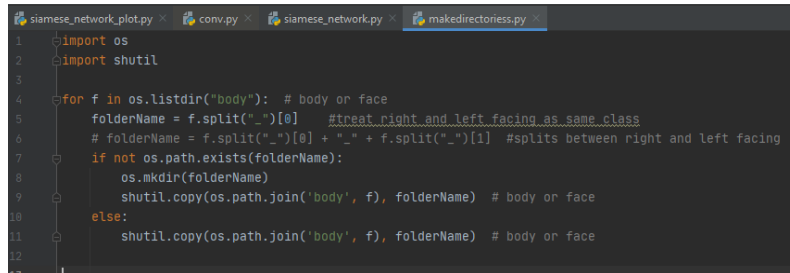
3 Method

I approached this task by doing a lot of research about Siamese networks in general, how they work, what they do, why they are used, et cetera. This gave me a good understanding of the task, as I had never heard of the architecture before this assignment. During my research, I found a really good example of someone that used a siamese network on an AT&T face dataset[13], which I read through some times to understand everything that he had done. He also uploaded his code and explained some of it here[14] which was extremely helpful for this assignment. As there is no point in reinventing the wheel, I decided to reuse a lot of his code to get a good basis that I could work from and modify to my liking. A link to his project on GitHub can be found here[15]

As his project was based on human faces and a completely different dataset, I had to make some modifications to get it to work correctly. As per the email I got with the fish dataset, it did not matter if I chose to focus on the head or body or treat left and right facing as the same class or two different classes. As I was unsure which one was the best, I decided to try all of the possibilities by making four different datasets that I would learn and test on.

- Face, where I split between left-facing and right-facing
- Face, where I kept left facing and right facing as the same class
- Body, where I split between left-facing and right-facing
- Body, where I kept left facing and right facing as the same class

As the original dataset only contained two folders, one for the full-body images and one for the face images, I decided to use python to make the four different datasets that I could learn and test on.

A screenshot of a code editor with four tabs: 'siamese_network_plot.py', 'conv.py', 'siamese_network.py', and 'makedirectories.py'. The 'makedirectories.py' tab is active, showing Python code that iterates through the contents of a 'body' directory. The code uses 'os.listdir' to get the directory contents, 'f.split' to process the filename, and 'os.path.exists' to check if a folder already exists. If not, it creates the folder with 'os.mkdir' and copies the file with 'shutil.copy'. Comments in the code indicate different ways to handle left and right facing images, such as 'treat right and left facing as same class' and 'splits between right and left facing'.

```
1 import os
2 import shutil
3
4 for f in os.listdir("body"): # body or face
5     folderName = f.split("_")[0] #treat right and left facing as same class
6     # folderName = f.split("_")[0] + "_" + f.split("_")[1] #splits between right and left facing
7     if not os.path.exists(folderName):
8         os.mkdir(folderName)
9     shutil.copy(os.path.join('body', f), folderName) # body or face
10 else:
11     shutil.copy(os.path.join('body', f), folderName) # body or face
12
```

Figure 4: Making the four different datasets

```

48
49     # face
50
51     #training_dir = "data/wrasse/training_head_sameway/"
52     #testing_dir = "data/wrasse/testing_head_sameway/"
53
54     training_dir = "data/wrasse/training_head_bothways/"
55     testing_dir = "data/wrasse/testing_head_bothways/"
56
57     # body
58
59     #training_dir = "data/wrasse/training_body_sameway/"
60     #testing_dir = "data/wrasse/testing_body_sameway/"
61
62     #training_dir = "data/wrasse/training_body_bothways/"
63     #testing_dir = "data/wrasse/testing_body_bothways/"
64

```

Figure 5: Using the four different datasets

After making all the different datasets and splitting them into training data and testing data, I changed some things in the data loader before getting it to work correctly. The code that I started with did not have a useful test, so I decided to completely remove his testing and make one myself. This included calculating the different accuracies like total accuracy, same fish accuracy, and different fish accuracy. I decided to split them up and calculate all of them because there are different use cases where too many false positives are worse than getting too many false negatives, and vice versa. I calculated these for many different distance thresholds from 0.3 to 2.9 to graph the differences.

The differences between the same fish accuracy and different fish accuracy are higher towards small and huge distance thresholds. When the threshold is tiny, it has to be very similar to be classified as the same fish. This makes it so even fish that are the same but look a bit different from the original picture get classified as different fish - we end up with false negatives.

The opposite is true when the threshold is high because it just has to look kind of similar to be classified as the same fish. This causes a lot of false positives, but at the same time, it does classify the same fish with up to 100% accuracy. Choosing between having a low and high threshold entirely depends on the use case. Some examples where the choice can vary:

- **High threshold:** If someone found out that there is a batch with a disease, it is essential to catch every single one, even if it contains some false positives. So the threshold gets increased.
- **Low threshold:** Suppose you have been giving some medicine to a batch, and you want to research if the medicine is effective. In that case, you may want to put a low threshold so you can be 100% certain that the fish you are using has gotten the medicine. In cases like this, it does not matter if you get some false negatives, as you got enough fish to research on. The dangerous part is getting false positives that skew the research negatively.

4 Results and discussion

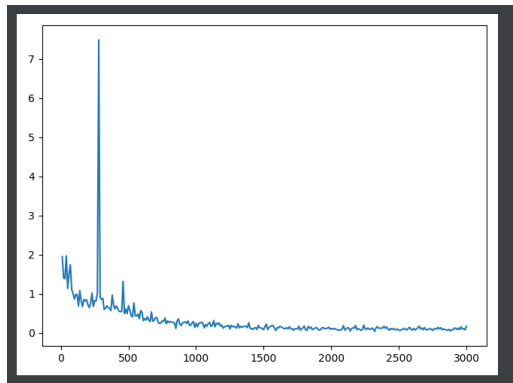
After running my implementation four times, one time for each of the different datasets I made, I got a lot of data and graphs as a result, so it will be beneficial to compare the differences.

4.1 Results from all the different runs

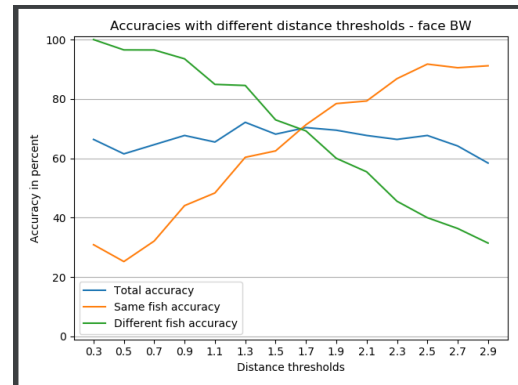
4.1.1 First run - face - left and right as same class

This dataset contained 507 different classes, so I decided to train on 450 of them and test on the remaining 57 classes. The loss got extremely low towards the end during training, as shown in Figure 6a. After training for 100 epochs, different distance thresholds were tried on the test set, and all of the accuracy numbers were turned into a graph. See Figure 6b.

With low distance thresholds the different fish accuracy is high and the same fish accuracy is low, and vice versa with high distance thresholds. The total accuracy seems to stay mostly the same, at around 60-70%. As for the examples in Figure 7, 0 means that they are the same, and 1 means they are different. The distance, prediction, and truth, are shown at the top of each image pair. Image 3 has a higher loss than image 4, but image 3 is classified as the same, and image 4 is classified as different. This is because they are examples of different distance thresholds. Image 4 is the same fish, but it got the wrong prediction because the threshold was 0.5.



(a) Loss over time - face bothways.



(b) Accuracy over time - face bothways

Figure 6: Graphs and stats from run

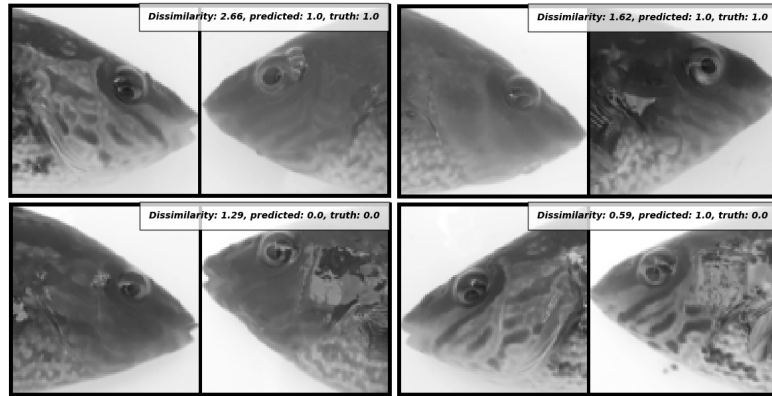
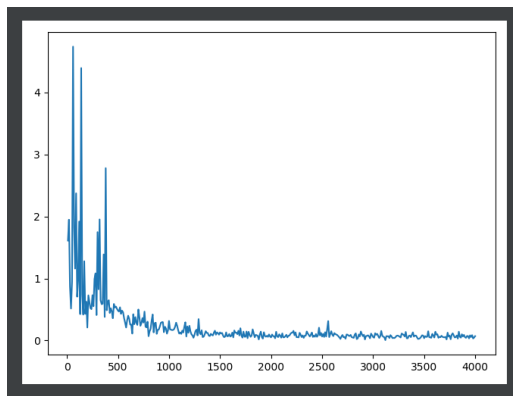


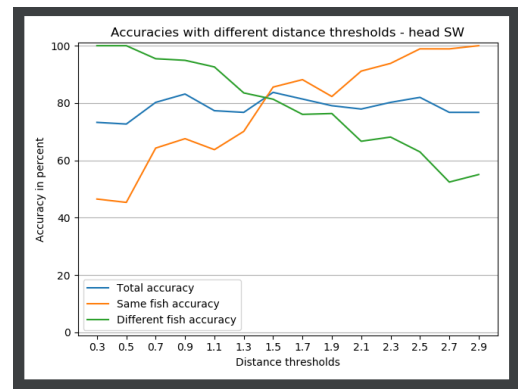
Figure 7: Examples - face bothways

4.1.2 Second run - face - left and right different class

When splitting left and right into different classes, it ended up with 982 total classes - up from 507 when they are considered to be the same class. The loss reached lower than it did previously. The accuracy when splitting the classes got higher, between 75 and 85%, but at the cost of classifying the same fish to be different if it gets one image from the right and one image from the left. Although, this should not matter as long as you have pictures from both sides of all of your fish. If you only got pictures of the right side, but the classification only got a picture of its left side, it will do poorly. Total accuracy is highest with 1.5 distance threshold, somewhat expected since it is in the middle. The biggest difference from this and the previous run is the same fish accuracy with low distance thresholds. for example, with 0.5 distance threshold, the same fish accuracy increased from 30% to 50%. I will not include examples as they are mostly the same as the previous ones in Figure 7.



(a) Loss over time - face sameway.



(b) Accuracy over time - face sameway

Figure 8: Graphs and stats from run

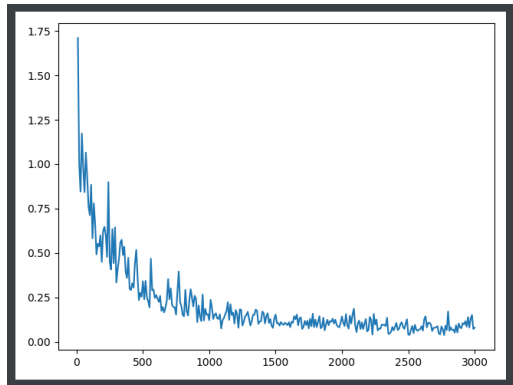
4.1.3 Third run - body - left and right as same class

This time around, the Siamese Network was ran on the body datasets, first with left and right as the same class. The accuracy was expected to be a lot lower on the body than on the faces because of one important thing: rotation. In many of the full-body images, the same fish were rotated different from one picture to another, even in the pictures where they looked the same way. This caused me to think that it would get thrown off by similar fish that had the same rotation, but it turns out that the Siamese network got smarter than expected. While angle and shape are important, the network learned to pay more attention to different pigmentation, patterns, and small differences in the skin.

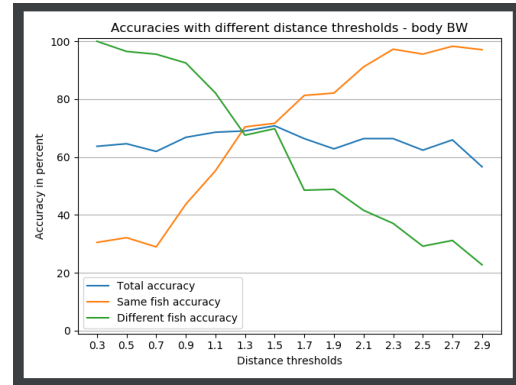
They have started with the fish species corkwing wrasse, since it is streaked with patterns. Later on, they will add more species of fish.

"All fish come in different patterns, and we call this the fish's fingerprint." [1]

The loss and accuracy of full-body images, where left and right-facing were treated as the same class, can be shown in Figure 12c and Figure 12d. The total accuracy ranged between 60 and 70%, with the highest hitting 71,7% with a distance threshold of 1,5. In the examples shown in Figure 10, the first pair shows that two images of the same fish, even looking the same way, can have widely different positioning and tail placement. This shows that the network has learned to look more at the skin's pigmentation and patterns than positioning and shape.



(a) Loss over time - body bothways.



(b) Accuracy over time - body bothways.

Figure 9: Graphs and stats from run

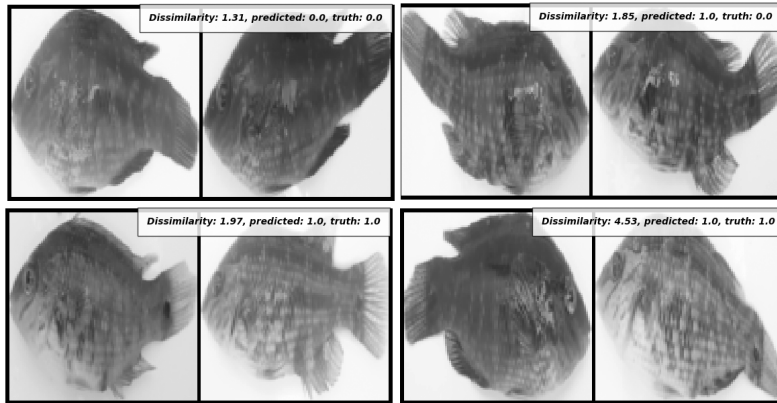
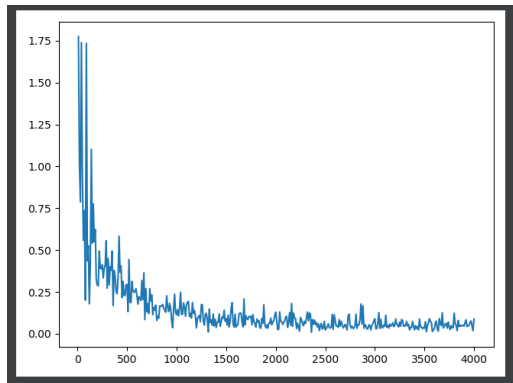


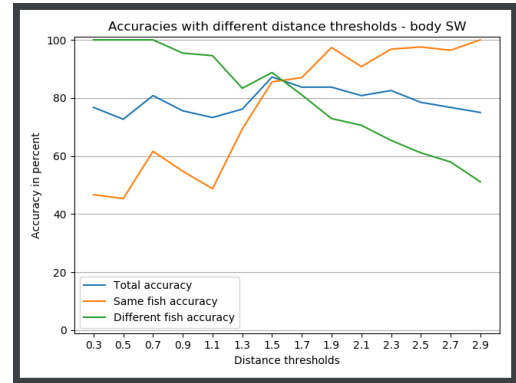
Figure 10: Examples - body bothways

4.1.4 Fourth run - body - left and right different classes

The last run was on the full body images again, but this time left and right looking fish were treated as different classes. This run actually got the highest accuracy of them all, ranging from 75 to 88%. These numbers are extremely high, which means that the network has gotten really good at learning exactly what differentiates different fish. The accuracy peaked at 87,2% accuracy at 1,5 distance threshold.



(a) Loss over time - body sameway.



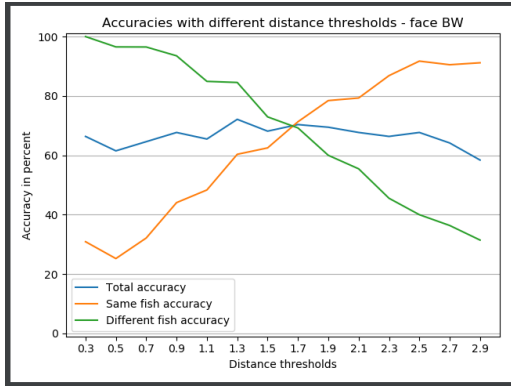
(b) Accuracy over time - body sameway.

Figure 11: Graphs and stats from run

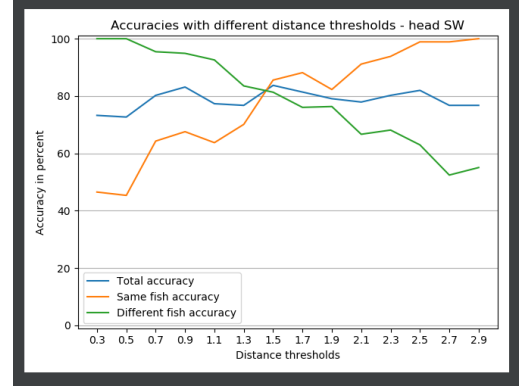
4.2 Comparing the differing runs

Section 4.2. The network's accuracy ranges from around 55% to 90%, based on the different run settings and distance thresholds. We can notice that the runs where right and left facing images were treated as different fish; the accuracy increased substantially. It is almost a 10 to 15% increase from when they are treated as the same class. When the fish face the same way, the distance between two images is lower than when they are facing opposite ways. This means that the network gets a lot higher same fish accuracy at lower distance thresholds because the distance between the same class's images is lower overall.

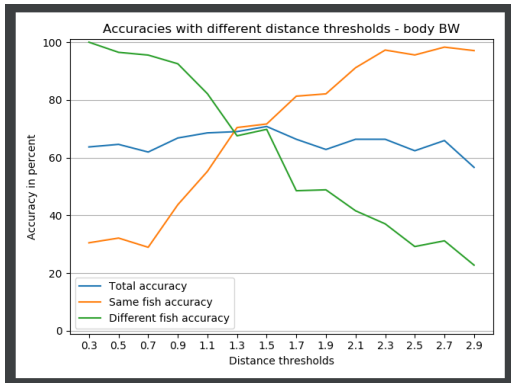
While the left and right facing change made a big difference, it seems like the face and full-body runs were very similar in total accuracy. This may end up with a slight edge towards full-body images, as they are easier to take than having to crop all of the images only to contain the face. The highest accuracy recorded from all of the runs were on the body images, with left and right being different classes and a distance threshold of 1,5. The accuracy peaked at 87,2%.



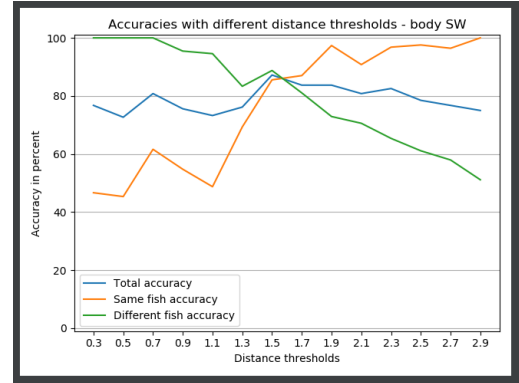
(a) Accuracy over time - face bothways.



(b) Accuracy over time - face sameway.



(c) Accuracy over time - body bothways.



(d) Accuracy over time - body sameway.

Figure 12: Accuracy graphs and stats from all of the runs

5 Conclusion

In conclusion, the assignment was to use facial recognition to identify if two images of fish are the same one of different fish. Being able to do so will help Havforskninginstituttet a lot, as it will make marine research simpler than it currently is. Currently they have to pick up the fish and manually tag it, so it can be recognizeable at a later time. This is very stressful for the fish, compared to being photographed directly in the sea. The scalability also increased tenfold, as it can be automated and huge databases can be made.

All in all, the siamese network worked really great at identifying if two images of fish are the same fish or different ones. The network managed to achieve an accuracy as high as 87.2%. There are choiced to be made when running the network based on what the needs are. If false positives are dangerous, while for example doing research on some specific fish, a lower distance threshold is favorable as it the fish has to be identical to be classified as such. When false negatives are dangerous, foe example if a batch has a disease and you need to remove every single one from that batch, a high distance threshold is favorable as it will give more leeway in the classification.

References

- [1] UiA Sissel Eikeland. *Developing facial recognition to track fish*. URL: <https://www.uia.no/en/news/developing-facial-recognition-to-track-fish>. (accessed: 26.11.2020).
- [2] Wikipedia. *Face perception*. URL: https://en.wikipedia.org/wiki/Face_perception#Development. (accessed: 26.11.2020).
- [3] Thalesgroup. *Facial recognition: top 7 trends (tech, vendors, markets, use cases and latest news)*. URL: <https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/biometrics/facial-recognition>. (accessed: 26.11.2020).
- [4] Ben Dickson. *What is one-shot learning?* URL: <https://bdtechtalks.com/2020/08/12/what-is-one-shot-learning/>. (accessed: 26.11.2020).
- [5] Ben Dickson. *A friendly introduction to Siamese Networks*. URL: <https://towardsdatascience.com/a-friendly-introduction-to-siamese-networks-85ab17522942>. (accessed: 26.11.2020).
- [6] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. URL: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. (accessed: 26.11.2020).
- [7] Deeplearning.ai. *C4W4L03 Siamese Network*. URL: <https://www.youtube.com/watch?v=6jfw8MuKwpI>. (accessed: 26.11.2020).
- [8] Enosh Shrestha. *Learning Similarity with Siamese Neural Networks*. URL: <https://medium.com/@enoshshr/learning-similarity-with-siamese-neural-networks-51c9ef534ae4>. (accessed: 26.11.2020).
- [9] Tirmidzi Faizal Aflahi. *Improving Siamese Network Performance*. URL: <https://medium.com/@kuzuryu71/improving-siamese-network-performance-f7c2371bdc1e>. (accessed: 26.11.2020).

- [10] Olivier Moindrot. *Triplet Loss and Online Triplet Mining in TensorFlow*. URL: <https://omoindrot.github.io/triplet-loss>. (accessed: 26.11.2020).
- [11] Vaneeda Allken et al. "Fish species identification using a convolutional neural network trained on synthetic data". In: *ICES Journal of Marine Science* 76.1 (Oct. 2018), pp. 342–349. ISSN: 1054-3139. DOI: 10.1093/icesjms/fsy147. eprint: <https://academic.oup.com/icesjms/article-pdf/76/1/342/31238015/fsy147.pdf>. URL: <https://doi.org/10.1093/icesjms/fsy147>.
- [12] et al. Muhammad Ather Iqbal Zhijie Wang. *Automatic Fish Species Classification Using Deep Convolutional Neural Networks*. URL: <https://link.springer.com/article/10.1007/s11277-019-06634-1>. (accessed: 26.11.2020).
- [13] Harshvardhan Gupta. *One Shot Learning with Siamese Networks in PyTorch*. URL: <https://hackernoon.com/one-shot-learning-with-siamese-networks-in-pytorch-8ddaab10340e>. (accessed: 26.11.2020).
- [14] Harshvardhan Gupta. *Facial Similarity with Siamese Networks in PyTorch*. URL: <https://medium.com/hackernoon/facial-similarity-with-siamese-networks-in-pytorch-9642aa9db2f7>. (accessed: 26.11.2020).
- [15] Harshvardhan Gupta. *Facial-Similarity-with-Siamese-Networks-in-Pytorch*. URL: <https://github.com/harveyslash/Facial-Similarity-with-Siamese-Networks-in-Pytorch>. (accessed: 26.11.2020).