Deep Leaning Models for Fish Detection in Noisy Environments

Marco Ceccato†

***Abstract*—Sonar based instruments are widely used in many commercial and scientific applications. One of these applications is the fish detection and classification task that usually is manually operated by fishermen and other experts by interpreting the so called fish arches. This field presents open challenges to automatize this process by automatically detecting and tracking schools of fish using deep learning frameworks. This work proposes multiple models to tackle different problems: the first one removes the noise from the raw measurements using a set of stacked denoising autoencoders, the second one estimates the number of targets by exploiting transfer learning and a convolutional classifier. The third one aims to track each target along his path. The performance of the models will be evaluated as a function of several parameters.**

***Index Terms*—Sonar FishFinders, Denoising, Signal Restoration, Deep Learning, Fish Classification**

I. Introduction

FROM the first ingenious experiment carried out by Leonardo Da Vinci in 1490, *sound navigation ranging (SONAR)* techniques have become fundamental in many scientific and commercial applications. SONAR has evolved in various forms, exploiting several types of signals and technologies to bring a wide range of solutions for military and civilian goals.

In sporting and fishing industry, as well in scientific monitoring of the fish ecology, the usage of SONAR is growing increasingly, since it may help to understand what is hiding under the surface of the sea. Commercial *fish-finders* are instruments used to locate fishes underwater by detecting reflected pulses of sound energy, showing to the fisherman the so called *fish arches*. Form these graphical representations of the reflected pulses, it is possible to infer the information about the fish such as shape, species, and its position. Expert fisherman can quickly deduce the above by looking to the arches, however, this ability requires experience and some theoretical notions in order to use these instruments properly. Advanced *fish-finders* interprets the raw arch measurements by producing as output the desired information about fishes relieving fishermen from the task of arches interpretation. Besides making easier the instrument's usage, the automatic classification of fishes can outperform the ability of the human interpreter, producing more accurate reports. As for many other fields, the potentials of machine learning and neural networks have proved to be extremely big. The first applications of neural networks in the fish classification has been investigated in [1] and [2] where four species of fish have been classified by the use of two transducers with variable orientation.

Albeit the recent advancements of classification models are able to produce satisfactory results for fishermen and others operator, some problem arises in case of monitoring targets that lie in the deep ocean due to the high noise levels that compromise the measurements. In those cases, the school of fish detection might be a hard task. Imagine the situation where the power reflected by targets has statistically the same magnitude of the transducer's sensitivity, then for operators get almost impossible to identify the useful signal that is surrounded by noise. Filters and image processing techniques can surely get rid of some noise, but when the SNR levels are too low, even advanced fish-finders might experience issues to process the reflected pulses and detect fish schools, requiring the most advanced techniques of image/signal restoration to be installed in the instrument.

The goal of this work is to propose and analyze how some of the most recently born deep learning tools can be used to build models able to detect and classify school of fish, under a very noisy environment and according to several parameters that characterize the fishes. Data used to learn and validate the models has been synthetically generated reproducing plausible fish arches, simulating the presence of fish schools that reflects pulses emitted from a transducer installed on the boat. The first proposed model aims to remove noise present on the image by learning a latent representation of the signals and then reconstructing them without noise. The second proposed model takes as input the previously learned latent representation and estimates the number of targets that compose the group. The paper is structured as follows. The second section will be about related works, whereas in the subsequent one, the dataset and pre-processing are described outlining the aimed operations we would like to perform and further assumptions made. In the fourth section, the proposed models and their training procedure are described in details whereas in the fifth part the performance achieved by models will be investigated with respect to several parameters and metrics. In the last section, some conclusion will be drawn.

II. Related Work

Fish classification by using underwater acoustic signals is divided into two main categories: the first one use uses received sounds to extrapolate information. They are powerful to classify and detect the presence of big fishes that emit sounds or vocalizations. Usually, frequency analysis is used to compute *spectrograms* and other data that can be fed into

†Marco Ceccato, email: ceccatom@dei.unipd.it

machine learning tools, like in [3]. The second category instead uses an active or passive transducer to collect all the reflected pulses and compute fish arches by means of range-time waterfall images. Actually, another approach can be used: visual recognitions systems, in fact, as for others animals in order to track and detect them [4], video and images can be used to track underwater fishes [5]. Most of the published works, related to underwater SONAR tackle the similar problem of object tracking and detection, regardless of fish theme. Deep learning nowadays outperforms human performance in so many tasks, and its applications spanning most of the scientific fields. Convolutional networks are effective on capturing spatial correlation, recurrent layers enhance temporal evolution whereas autoencoders make possible to learn latent representations: by combing all the above elements and building deep models, there are no restrictions on applications and new opportunities where these techniques can be exploited, and fish monitoring could be one of them.

III. Dataset and Processing

The dataset has been synthetically generated to replicate real fish arches, also known as *range-time waterfall images*. The image represents how the distance (ordinate axis) between the fish and the transducer changes in time (abscissa axis). Each waterfall contains 20 temporal samples and a spatial resolution of 1200 pixels. To simulate multiple fish species and different behavior, the fish arches are affected by the following scalar generation parameters:

- The *Velocity* that the fish schools have when passing under the transducer coverage. Its values are between 15 and 20
- *Width* of the fishes. The wider fish, the stronger pulse intensity is reflected and caught by the sensor. This results in more pixels triggered in the waterfall images.
- The *Number of targets* that compose the school of fish. The maximum number of fishes is 12. Fishes' paths are allowed to intertwine among them, however, the general behavior is that the group pass linearly under the transducer (or the boat passes over the fish).
- The *SNR* of the image. Once the waterfall-image is generated, in order to reproduce all the environmental noises that affect the signals reflected by fishes, additive noise is added to the image, making the *Signal to Noise Ratio (SNR)* values varying between 1 and 3. Note that due to the nature of the signals and the noise levels, the interpretation of the corrupted waterfalls make hard and sometimes even impossible for a human observer to distinguish the numerosity and the fish paths.

The dataset contains 60000 synthetic generated waterfalls. As already mentioned for each sample are associated with the generation parameters, the noisy waterfall and the noiseless one. The purpose of this work is to detect and track the fish paths and estimate the number of targets by using the only noisy measurement. Multiple approaches are feasible to achieve that goal, and this will be discussed in the next section. The dataset is divided into three fixed parts that will be used respectively to train, validate and test our models. The use of more advanced cross-validation schemes has not been considered mainly for two reasons: firstly because the size of data at our disposal is big enough to learn and compute the required metrics with small variance, and secondly is intrinsically related with computational cost needed for cross-validation schemes. The preprocessing of the raw measurements is the simple rescaling each input pixel between 0 and 1 with the following transformation:

$$f(x) = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

IV. Learning Frameworks

The models proposed by this work tackle different problems and therefore have separated structures. In this section, the two models and their training methods will be discussed more in details leaving aside their performance that will be treated in the next section.

A. DenoisingNet

DenoisingNet is the name of the first proposed model. It is a *Denoising Autoencoder* composed of 6 stacked *encoder-decoder blocks*. Its goal is to remove most of the noise contained in the waterfall image, producing as output an almost noise-free image that a knowledgeable operator can interpret without too many difficulties to collect all the information he needs. Building a deep network that, at each stacked block, reduce the bottleneck size, should impose to the model the learning of a latent representation where the noise is somehow orthogonal to the data and thus easier to be filtered out automatically. The 6 stacked autoencoders, named *EncoderBlocks*, have the same structure but different sizes and parameters. Each *EncoderBlock* has two parts: the encoder that has the following layers:

- 2D Convolutional layer
- *ReLU* activation function
- 2D Batch Normalization layer
- Dropout layer

Note that the order between *ReLU* and the BatchNorm layer has been investigated with the outcome that for this task the used order appeared to perform better. Albeit the normalization layers has been idealized to preserve a stable distribution for sequent activation functions [7], ReLU seems to be an exception, and it is commonly used with both orders. The decoder part is composed of the same layers to perform the opposite operation. Notice that the decoder side of the first *EncoderBlock* is a special case and after the convolutional layer has just the 2D BachNorm layer followed by the sigmoid activation. Table 1 summarize the parameters for each stacked block of the Network.

The training procedure consists of a *greedy block-wise* training. Each EncoderBlock is trained by freezing the external ones in order to prevent the *gradient-vanishing* problem the

Parameters	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6
Convolutional kernel size	(6, 3)	(4, 3)	5	4	4	(3, 4)
Encoder activation	*ReLU*	*ReLU*	*ReLU*	*ReLU*	*ReLU*	*ReLU*
Decoder activation	*Sigmoid*	*ReLU*	*ReLU*	*ReLU*	*ReLU*	*ReLU*
Convolution stride	(2, 1)	(2, 1)	(3, 1)	2	2	1
Convolution padding	(2, 1)	(1, 1)	(1, 0)	1	1	(1, 0)
Latent representation size	(600, 20)	(300, 20)	(100, 16)	(50, 8)	(25, 4)	(25,1)
Encoded channels	16	32	64	128	128	128

TABLE 1: *DenoisingNet* structural parameters, details about the 6 stacked *EncoderBlocks*

occurs in deep networks and helps to mitigate the difficult optimization problem of deep networks by better initializing the weights of all layers reducing the possibility to get stuck in poor solution [**?**]. The input data on the training phase are the noisy waterfalls whereas the output is forced to be equal to the noise-less data. Note that, contrary to what usually happens in autoencoders, in this case, supervised learning is performed. The used loss function is the *binary cross-entropy* to autonomously predict each pixel if it is active or not.

B. FishClassifier

FishClassifier is the second proposed model. The purpose of this network is, given the raw measurements, to estimate the number of fishes present in the scanned image.
This model exploit *transfer-learning* by using the encoder side of *DenoisingNet* and other additional layers to perform the classification task. Since the previous latent representation is supposed to embed all the necessary information to reconstruct the input, it may also be used to get an estimated for the number of targets. The former part of the module is made by a sequence of one-dimensional convolution layers and maxpolling layers whereas the second part of the network use several linear layers, ReLU activations, and Dropout layers. The details about structural parameters that compose the network, along with a general overview of the module, are shown in Figure **??**. The training of this classifier is divided into two phases:

1) The encoder network is used just a *feature extractor* and does not take part of the training procedure whereas the last part in the network learn how to count fishes
2) Both parameters of encoder and classifier are trained through the *fine-tuning* of the network.

The used loss function is the *CrossEntropyLoss* that follows a *LogSoftmax* activation. The number of output neurons is 13 (12 maximum targets and one additional for the case of zero fishes detected)

C. PathTracker

The third and last model aims to track and produce as output the path for each fish. In the interests of intellectual honesty, please note that this model at the moment it is not working properly and the purpose of this section is just to reveal the reasoning that is behind that model, requiring more investigation to fix and improve the implementation obtained so far. The main idea is to first use the pre-trained *DenosingNet* to restore a clean waterfall image and then apply sequentially some convolutional layers followed by recurrent layers. In this process, the time-range is not seen as an image whereas as a sequence of 20 columns with 1200 elements. This is done in the belief that layers like LSTM can capture the temporal evolution of the targets. The labels are obtained from the true path and reshaped to form a column of 1200 classes. Ideally, better suited loss function that should be used is the *Connectionist Temporal Classification loss* [8], however, an extension to the case of multi-label in the case of several fishes lies outside the scope of this work. More standard losses have been tested but the final decision on which is the best it not yet reached.

V. Results

In this section, the performance of the proposed models will be evaluated by considering several aspects. Let us start remarking what already mentioned at the beginning, the dataset has been divided into three different parts: one for the training part, the second one to evaluate the training procedure and tune hyperparameters by considering unbiased samples and the last one is used only carry out the results showed in this sections. The reasons for which cross-validations schemes are described in section III. Another important consideration regards the hyperparameters tunning, in fact, it must be claimed that the solution found might be sub-optimal due to the iterative search of best parameters using a "babysitting" approach. Further investigations and resource could be used to improve this complex by using random/grid search schemes. The last general consideration is that the comparison with other similar works is difficult for two reasons:

- The lack of one direct competitor that tackle the same kind of problem
- The synthetic nature of the dataset that is different from the other datasets and reals data.

After all these premises, let us start to evaluate *DenoisingNet*. The model has been trained using all samples of the training part, without excluding some of them with the belief that bad samples (with a lot of noise and small fishes) will make the network more robust. In Figure 3 and Figure 4 two test example can give a qualitative demonstration of the Denoising performance: from the noisy waterfall is hard to detect the paths whereas the denoised outputs, even some imperfections

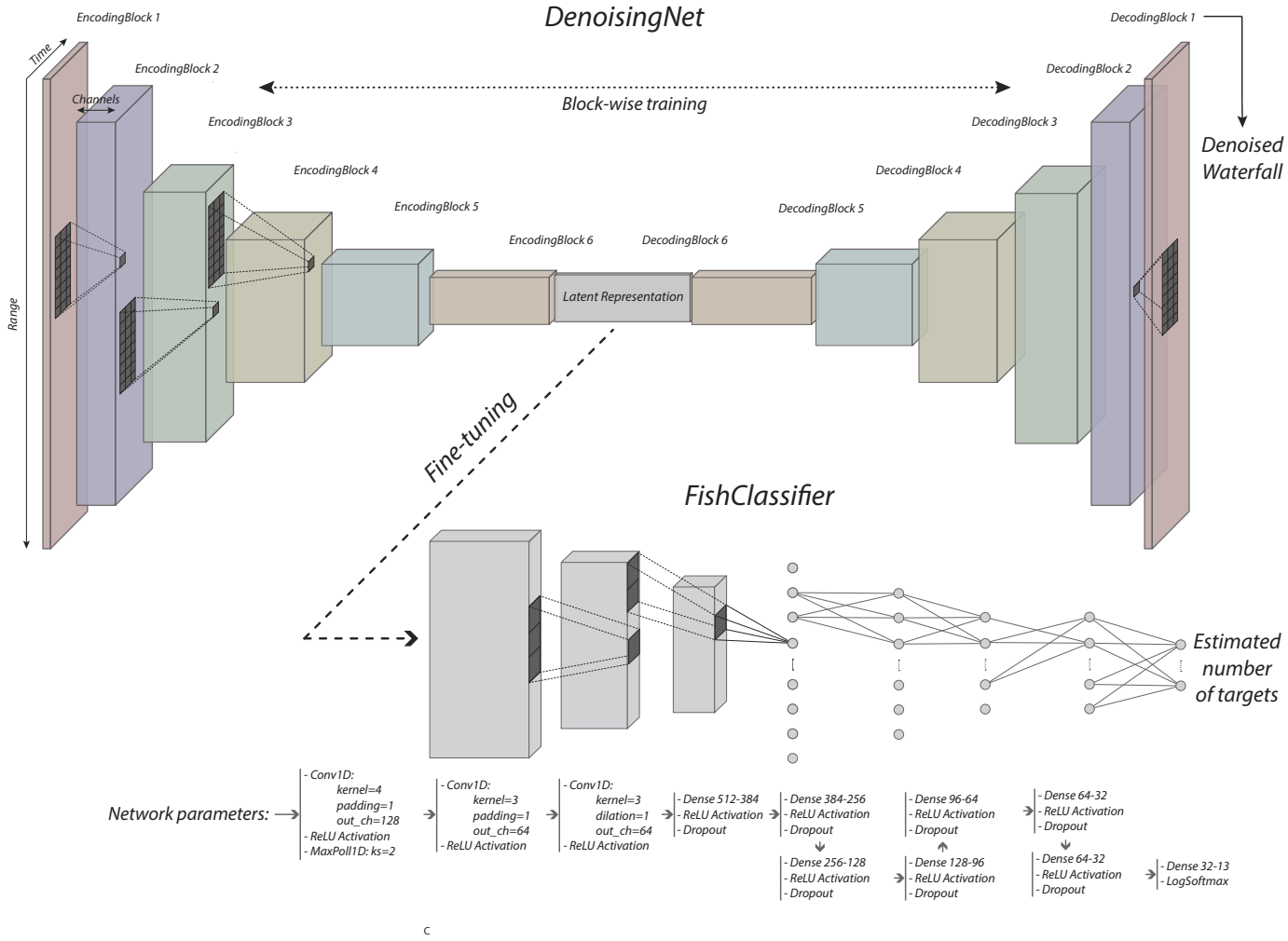

Fig. 1: Overview of the learning frameworks, and parameters of *FishClassifier* module

are still presents, it's easy to see them. If we have to consider the comparison between the denoised images and the ground truth an appropriate metrics have to be chosen. Since pixels' value can vary between 0 and 1, the problem can be viewed as a parallel binary classification of all the pixels. Once decided a proper threshold to classify pixels, standard metrics as accuracy, precisions and recall can be used. Due to the huge imbalanced classes and to the nature of the classification we are performing, the F-score seems to be more suited to for the evaluation. In Figure 5 the F-Score is shown as a function of the generation parameters used to generate the waterfalls. From that figure, some observation can be drawn:

- First of all, we note that the width of targets has a big impact on the performance (as expected)
- The velocity has a marginal influence on the F-score. By decreasing the velocity, the predictions of the models improve, but not so much.
- We can divide the behavior of the models in two SNR regions. Below 2 dB of SNR, the F-Score increase linearly as a function of the SNR, whereas in the other region the performance tends to saturate.

Let us consider now the second model. The task performed by *FishClassifier* is simply a classifier that exploits the transfer learning. A similar plot to evaluate the classification accuracy as a function the generation parameters is shown in Figure 6. The aforementioned observations can be referred also for this model. Another useful metric for multi-class classification is the confusion matrix. In Figure From the picture, we can conclude that besides the classification works quite well, the classification becomes more imprecise when the number of target increases.

VI. Concluding Remarks

Along with this paper, two different strategies to deal with noisy range-time waterfalls images obtained by SONAR transducers have been proposed and analyzed using synthetic simulated data. *DenoisingNet* can remove most of the noise and in parallel can learn a latent representation that can be exploited to transfer the learning for others tasks, like the one of estimating the number of targets that compose the schools

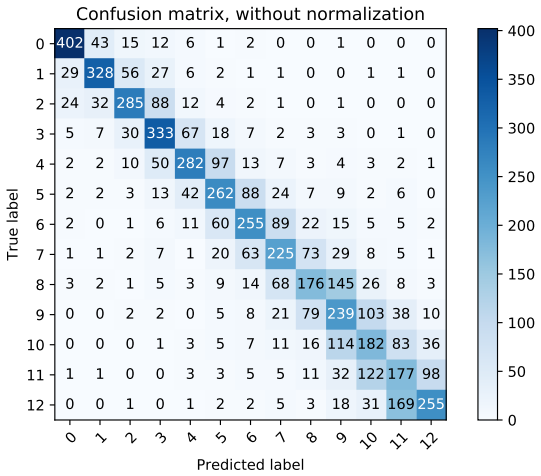

Fig. 2: Confusion matrix of the classification task performed by *FishClassifer*. Min SNR = 2, Min Width = 3

Parameter	*DenoisingNet*	FishClassifier
BatchSize	32	32
Dropout Factor	0.1	0.1
Learning rate	10^{-3}	10^{-4}
Epochs	5 per-block	$20 + 20$ (Fine-Tunning)
Optimizer	*Adam*	*Adam*

TABLE 2: Some of tested and tunned hyperparameters

of fish. Another model aims to Track each fish separately but require more investigations to be considered functioning.
More exhaustive hyperparameters search should be conducted to optimize the learning framework. Leaving aside the idea of directly use these models with real data, we can affirm that could be the base on which to build a valid classification and detection framework for commercial and scientific FishFinders.

REFERENCES

[1] P. H. Patrick, N. Ramani, W. G. Hanson, and H. Anderson, "The potential of a neural network based sonar system in classifying fish," in *[1991 Proceedings] IEEE Conference on Neural Networks for Ocean Engineering*, pp. 207–213, Aug 1991.

[2] N. Ramani and P. H. Patrick, "Fish detection and identification using neural networks-some laboratory results," *IEEE Journal of Oceanic Engineering*, vol. 17, pp. 364–368, Oct 1992.

[3] A. Kundu, G. C. Chen, and C. E. Persons, "Transient sonar signal classification using hidden markov models and neural nets," *IEEE Journal of Oceanic Engineering*, vol. 19, pp. 87–99, Jan 1994.

[4] C. J. Cohen, D. Haanpaa, and J. P. Zott, "Machine vision algorithms for robust animal species identification," in *2015 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pp. 1–7, Oct 2015.

[5] Zhang Yu, K. Mizuno, A. Asada, Y. Fujimoto, and T. Shimada, "New method of fish classification by using high-resolution acoustic video camera-aris and local invariant feature descriptor," in *OCEANS 2016 MTS/IEEE Monterey*, pp. 1–6, Sep. 2016.

[6] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized Denoising Auto-Encoders as Generative Models," *arXiv e-prints*, p. arXiv:1305.6663, May 2013.

[7] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv e-prints*, p. arXiv:1502.03167, Feb 2015.

[8] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, (New York, NY, USA), pp. 369–376, ACM, 2006.

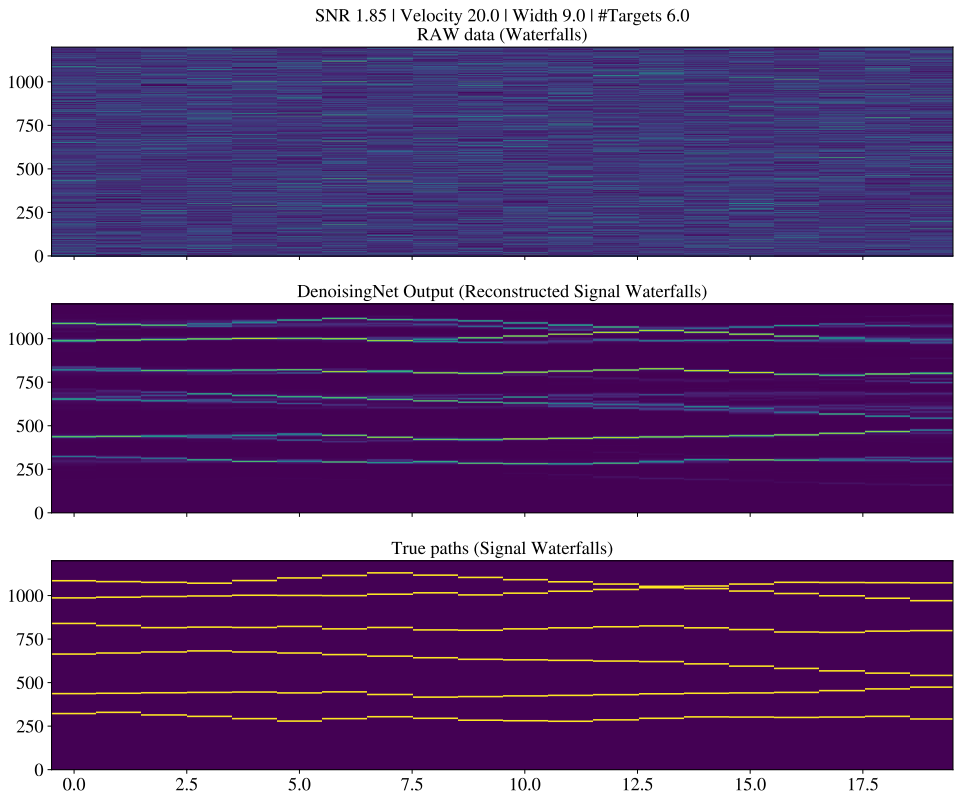
SNR 1.85 | Velocity 20.0 | Width 9.0 | #Targets 6.0
RAW data (Waterfalls)
DenoisingNet Output (Reconstructed Signal Waterfalls)
True paths (Signal Waterfalls)
0
250
500
750
1000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Fig. 3

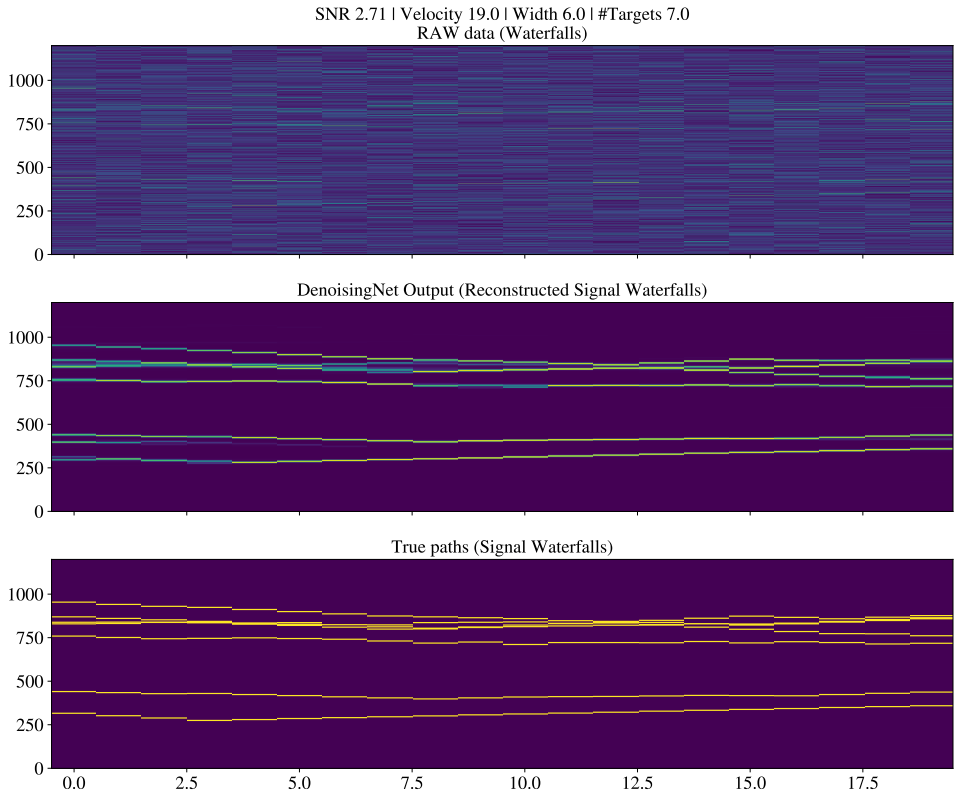
SNR 2.71 | Velocity 19.0 | Width 6.0 | #Targets 7.0
RAW data (Waterfalls)
DenoisingNet Output (Reconstructed Signal Waterfalls)
True paths (Signal Waterfalls)
0
250
500
750
1000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

Fig. 4

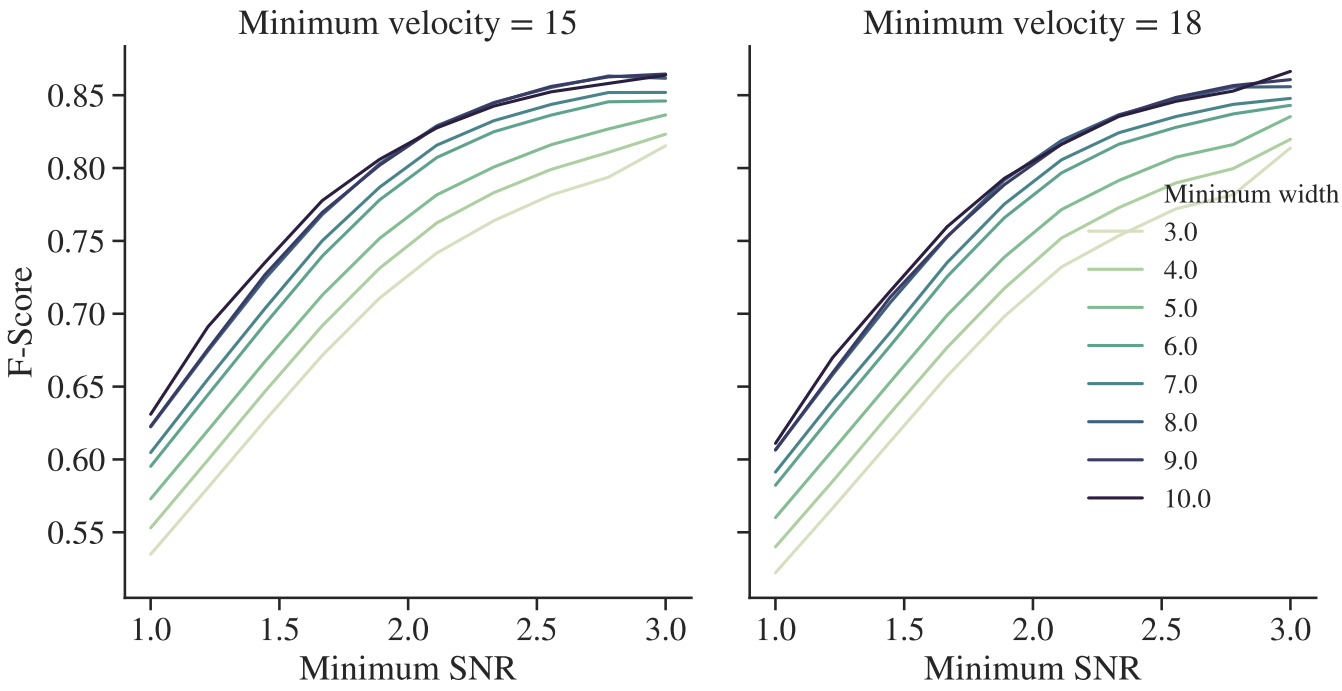

Fig. 5: F-Score for *DenoisingNet* as function the generation parameters. Threeshold = 0.5

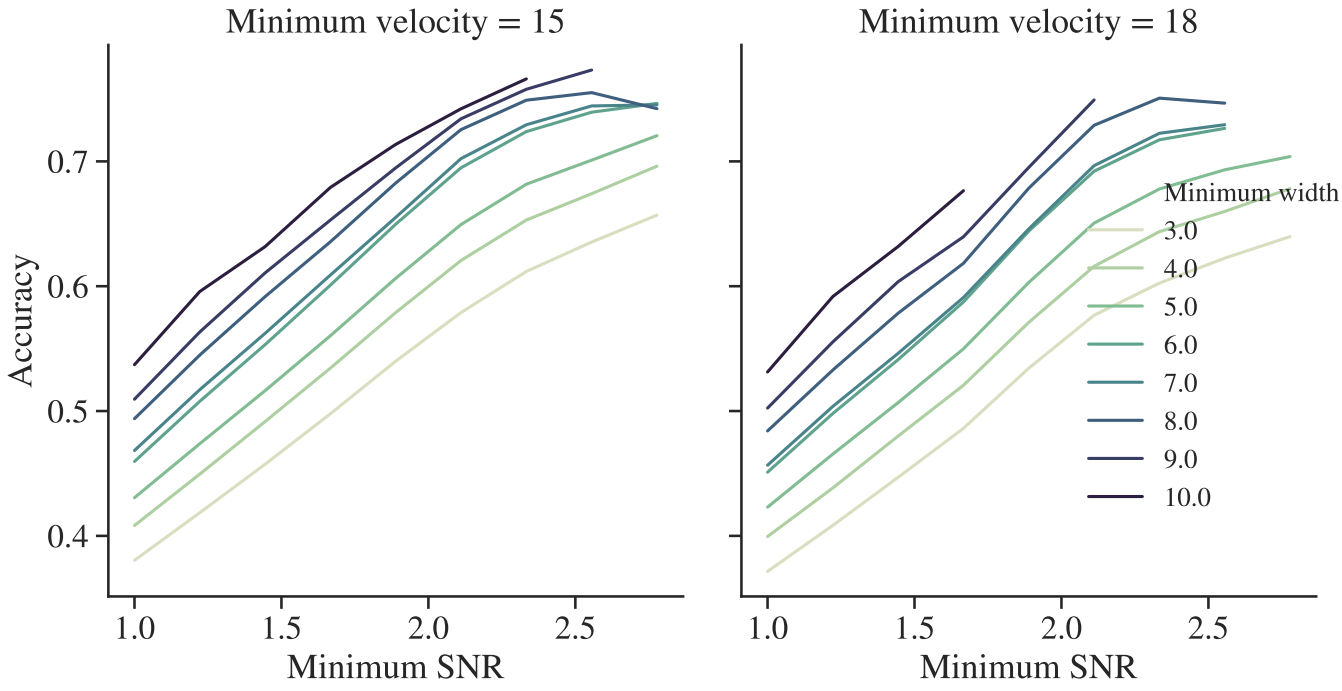

Fig. 6: Classification accuracy of *FishClassifier* varing the genereation parameters