

# Pre-trained Models for Sonar Images

Matias Valdenegro-Toro<sup>1</sup>Alan Preciado-Grijalva<sup>1,2</sup>Bilal Wehbe<sup>1</sup>

**Abstract**—Machine learning and neural networks are now ubiquitous in sonar perception, but it lags behind the computer vision field due to the lack of data and pre-trained models specifically for sonar images. In this paper we present the Marine Debris Turntable dataset and produce pre-trained neural networks trained on this dataset, meant to fill the gap of missing pre-trained models for sonar images. We train Resnet 20, MobileNets, DenseNet121, SqueezeNet, MiniXception, and an Autoencoder, over several input image sizes, from  $32 \times 32$  to  $96 \times 96$ , on the Marine Debris turntable dataset. We evaluate these models using transfer learning for low-shot classification in the Marine Debris Watertank and another dataset captured using a Gemini 720i sonar. Our results show that in both datasets the pre-trained models produce good features that allow good classification accuracy with low samples (10-30 samples per class). The Gemini dataset validates that the features transfer to other kinds of sonar sensors. We expect that the community benefits from the public release of our pre-trained models and the turntable dataset.

## I. INTRODUCTION

Machine learning is now ubiquitous in processing sonar images and performing perception of autonomous underwater vehicles. But it is limited by the size and quality of available training data, as capturing datasets with sonar images is more difficult than standard optical images.

In computer vision, pre-trained convolutional neural networks are used as a way to inject prior knowledge into a task [1]. These models are usually trained on large datasets such as ImageNet [2] or OpenImages [3], learning high quality features that are reusable for other tasks [4] such as object detection, semantic segmentation, etc, through transfer learning.

The application of transfer learning in sonar images is possible [5] [6], but it is limited by the availability of data and pre-trained models. Researchers do not generally share the models they train on their datasets, limiting advances in sonar perception.

There is clear need for pre-trained standard architectures with sonar images, which would reduce the sample complexity of classification tasks, serve as strong baselines for future work, and improve performance for other tasks like object detection or semantic segmentation, all with an architecture that is particularly tuned for sonar images, with one channel image data instead of the typical three channels (red, green, and blue). This motivates a tuning of particular architectures,

This work has been partially supported by the H2020-ICT-2020-2 ICT-47-2020 project DeepSense (Ref 101016958) funded by the European Union's Horizon 2020 research and innovation programme.

<sup>1</sup>German Research Center for Artificial Intelligence, 28359 Bremen, Germany matias.valdenegro@dfki.de

<sup>2</sup>Bonn-Rhein-Sieg University of Applied Sciences, 53757 Sankt Augustin, Germany alan.preciado@gmail.inf.h-brs.de

usually by reducing the number of filters and their overall width.

For this purpose, we produce pre-trained models of standard neural network architectures, namely ResNet20 [7], MobileNets [8], DenseNet121 [9], SqueezeNet [10], and MiniXception [11] on a turntable dataset with high variability captured using an ARIS Explorer 3000 Forward-Looking Sonar. We release<sup>1</sup> these models to the community, as a way to kick-start future research into computer vision with sonar images, in a similar way that models pre-trained on the ImageNet dataset have done for color images.

Our contributions are: we produce publicly available pre-trained convolutional neural networks for sonar images, at varying input image resolutions (from  $32 \times 32$  to  $96 \times 96$ ). Pre-training happens in a new dataset captured using an ARIS Explorer 3000 FLS, with objects in a rotating platform, allowing the capture of all object views. We call this dataset the Marine Debris Turntable dataset. We demonstrate the potential of our pre-trained models in two image classification tasks, one on the Marine Debris Watertank dataset<sup>2</sup>, and another in a binary classification dataset captured using a Gemini 720i Forward-Looking Sonar, which acts as an independent dataset.

We expect that the release of our turntable dataset and the pre-trained models in this dataset can help other researchers advance the field of computer vision on sonar images.

## II. STATE OF THE ART

Transfer learning has been a standard approach to improve performance in computer vision systems [4] for a long time, but these improvements have not been transferred to the acoustic imaging domain, mostly due to the lack of pre-trained models trained specifically for sonar images.

Valdenegro-Toro [5] showed that transfer learning in sonar images helps reduce sample complexity, with a slight performance loss if the transfer and feature learning datasets do not share objects in common.

Fuchs et al. [6] uses off the shelf models pre-trained on color images (ImageNet specifically), showing that these also improve performance in sonar image classification, but these models are very big, and require hacks in order to work with a sonar image (transforming from one to three channels), which wastes precious power and computation.

Lou et al. [12] uses GradCam in order to infuse localization capabilities in a ResNet18 trained for color images and to also generalize and improve performance on sonar images as evaluated on the watertank dataset. This can be considered an

<sup>1</sup>Available at <https://github.com/mvaldenegro/pretrained-models-sonar-images>

<sup>2</sup>This dataset is available at <https://github.com/mvaldenegro/marine-debris-fls-datasets/releases/>

advanced method of transfer learning as it combines features from optical and sonar data.

### III. PRE-TRAINED MODEL SELECTION

This section describes the architectures we selected to build pre-trained models:

**ResNet.** This model uses residual connections in order to improve gradient propagation through the network, and overall it allows much deeper networks. We use the ResNet20 variant, with 20 residual layers, which produce high quality features.

**MobileNet.** This architecture is designed to be lightweight and allow fast inference in embedded and mobile systems. It uses depthwise separable convolutions in order to reduce computation, which induces a trade-off between task performance (like accuracy) and computational performance. MobileNets are a good choice for resource constrained platforms like Autonomous Underwater Vehicles (AUVs), and the community would benefit from a version of this architecture trained for sonar images.

**DenseNet.** This architecture makes heavy use of feature reuse, with a set of dense convolutional blocks, each of them being a series of convolutional layers, where each layer receives as inputs all the feature maps from previous convolutional layers in the same block. This allows for very good task performance with a modest increase in the number of parameters, but it increases the amount of memory and computation required for a forward pass. This network is a good choice when very good task performance is required. We use the DenseNet121 variation which is the shallowest of the networks defined in [9].

**SqueezeNet.** The aim of this model is to reduce the number of parameters while maintaining competitive task performance, and this is achieved by carefully designing the so called Fire module which contains two sub-modules, one that squeezes information through a bottleneck (width lower than the current number of channels) and another that expands the amount of information. SqueezeNet has performance that is similar to AlexNet on ImageNet, but with overall 50 times less parameters, making it a good choice for task performance vs computation trade-off.

**MiniXception.** This network is a specific design of the Xception network to reduce the amount of computation, in particular for facial emotion recognition. Xception is another design that combines insights from MobileNets by using separable convolutions, into the design of the Inception network, which also performs quite well on the ImageNet competition.

**Autoencoders.** We train a Convolutional Autoencoder, which can be done without supervision, with the the following encoder architecture Conv2D(32,  $3 \times 3$ )-MaxPool( $2 \times 2$ )-Conv2D(16,  $3 \times 3$ )-MaxPool( $2 \times 2$ )-Conv2D(8,  $3 \times 3$ )-MaxPool( $2 \times 2$ )-Flatten()-FC( $c$ ), and decoder architecture: FC( $cn_w n_h$ )-Reshape()-Conv2D(32,  $3 \times 3$ )-UpSample( $2 \times 2$ )-Conv2D(16,  $3 \times 3$ )-UpSample( $2 \times 2$ )-Conv2D(8,  $3 \times 3$ )-UpSample( $2 \times 2$ )-Conv2D(1,  $3 \times 3$ ). This architecture is trained using a mean squared error loss, and the Adam optimizer, with the same setup as the other architectures. For feature

Architecture	Selected Width $w$	Use of $w$ in Architecture
Resnet20	32	$w$ is set as starting number of filters, duplicating itself after each residual stack ( $w = 2w$ ).
MobileNet	32	Base filters are set to $w$ , increasing first by a factor of four ( $w = 4w$ ), then duplicating themselves.
DenseNet121	16	$w$ is used as number of filters for each convolutional layer inside each dense block.
SqueezeNet	32	Base squeeze filters are set to $w$ , and expansion filters are set to $2w$ , duplicating itself after each fire module stack.
MiniXception	16	Stem widths are set to $[\frac{w}{2}, w]$ , and block widths to $[w, 2w, 4w, 8w]$ .

TABLE I: Selected width hyper-parameter for each pre-trained architecture, based on results from Figure 1.

extraction, only the encoder is used. The feature or code size  $c$  can be varied to produce different feature sizes. We vary  $c \in [4, 8, 16, 32, 64, 128]$ , with each value of  $c$  producing a unique autoencoder architecture.

#### A. Hyper-Parameter Tuning

As the original architectures we selected are designed for color images and their increased visual complexity, they need to be tuned for the specifics of sonar images, mainly a reduction in the number of parameters (to lessen the chances of overfitting), which can be done by adjusting the width of each layer. Depth of these networks should not be a problem.

Each architecture is slightly modified. We choose the shallowest variations of each architecture, and we reduce the width (by changing the number of filters or neurons) on each architecture, tuning them to produce maximum task performance, which usually happens with less than 128 filters (instead of up to 1024 filters in the original architecture). For this purpose we perform a grid search to decide the width of each architecture, on the Marine Debris Turntable training set at resolution  $96 \times 96$ , using K-Fold Cross Validation with  $k = 5$ . We evaluate widths varying in the range  $8 - 64$ , depending on the architecture. These results are shown in Figure 1.

We selected the width of each architecture to maximize mean test accuracy over folds in K-Fold cross-validation. The selected widths are displayed in Table I, as well as how the width  $w$  parameterize each architecture.

#### B. Input Image Variations

For each architecture, we train a set of models varying the input image size, including sizes  $s \times s$  for  $s \in [32, 48, 64, 80, 96]$ , as the original input size typically used in the ImageNet dataset ( $224 \times 224$ ) is too big for a sonar image, depending on frequency. This allows to learn features at multiple scales, and to evaluate the optimal image size. We will provide all models trained as the end user can select the best scale depending on their needs. Note that smaller scales also require less computation.

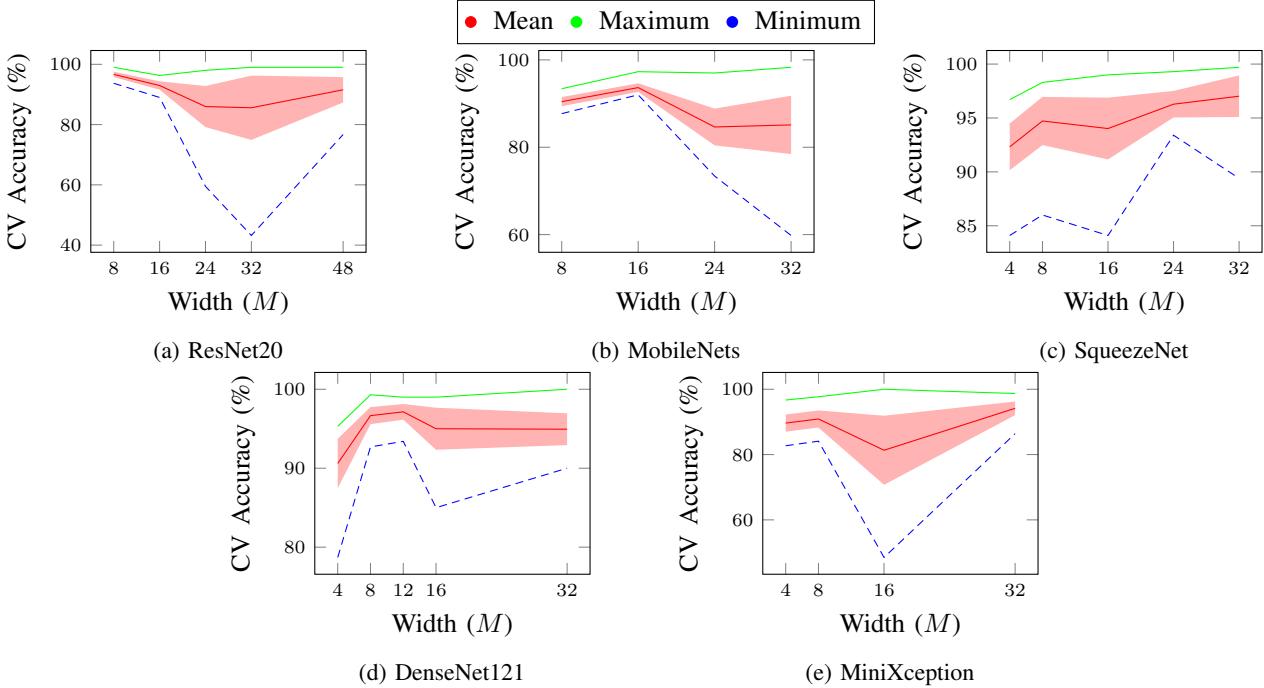


Fig. 1: Results from 5-Fold Cross validation on the Turntable training set, showing the effect of the width on accuracy for each selected architecture. Shaded areas represents one- $\sigma$  confidence intervals.

### C. Computational Performance

We believe that computational performance is important for the future users of our pre-trained models, as we expect that they will be used for real-time applications such as AUVs, which requires fast processing on low power embedded platforms. In this section we provide information on the computational requirements of each model.

We compute the number of FLOPs<sup>3</sup> required for a single forward pass for each model, as the input image size and widths are varied. These results are shown in Figure 2.

ResNet20 and DenseNet121 are the most computationally expensive models across all architectures, while MobilNets and MiniXception are the computationally lightest.

## IV. EXPERIMENTAL EVALUATION

We train the previously mentioned models (ResNet, SqueezeNet, DenseNet, MobileNet, and MiniXception) on the Marine Debris Turntable Dataset.

### A. Datasets

We use the following datasets for training and evaluation:

**Marine Debris Turntable.** This dataset contains 2471 images in total, over 18 different objects that are grouped into 12 classes. The dataset was captured using an ARIS Explorer 3000 forward-looking sonar, at the highest frequency setting. Each object is placed in an underwater turntable and rotated from the minimum to the maximum angle, rotated along the Z axis. This produces a comprehensive set of object views,

which is ideal to learn features from, as it contains many properties typical to sonar images that only happen on specific viewpoints, such as reflection artifacts, ambiguous poses, and sensor noise. A sample of these object classes is shown in Figure 4, while samples images of the turntable rotating on the metal box class is shown in Figure 6. The classes in this dataset are: bottle, can, carton, box, bidon, pipe, platform, propeller, sachet, tire, valve, and wrench.

The Turntable dataset has several configurations, and in this work we only use the platform image crop, and 12 classes. The dataset is split into training set with 1505 images, and a test set of 645 images. Note that the original samples of this dataset are stored in the same order as they are captured, which is induced by the turntable rotation. A train-test split is performed across this order for each object as if it was a time series, in order to prevent leakage across the turntable rotation, producing an independent test set.

This dataset will be used for training the architectures and produce pre-trained models for sonar images, as it is the most difficult dataset we have.

**Marine Debris Watertank.** This dataset contains 2627 FLS images captured with an ARIS Explorer 3000 forward-looking sonar at 3.0 MHz, split three way into training, validation, and test set [13]. In this work we only use the training set containing 1838 images, and the test set of 395 images. The dataset contains 11 classes including the background class. This dataset is used for evaluation of transfer learning using the pre-trained models that we propose in this paper. There is some intersection between the object classes in the Turntable and Watertank dataset (around 50% of objects), so it is not a

<sup>3</sup>Floating point operations like additions and multiplications.

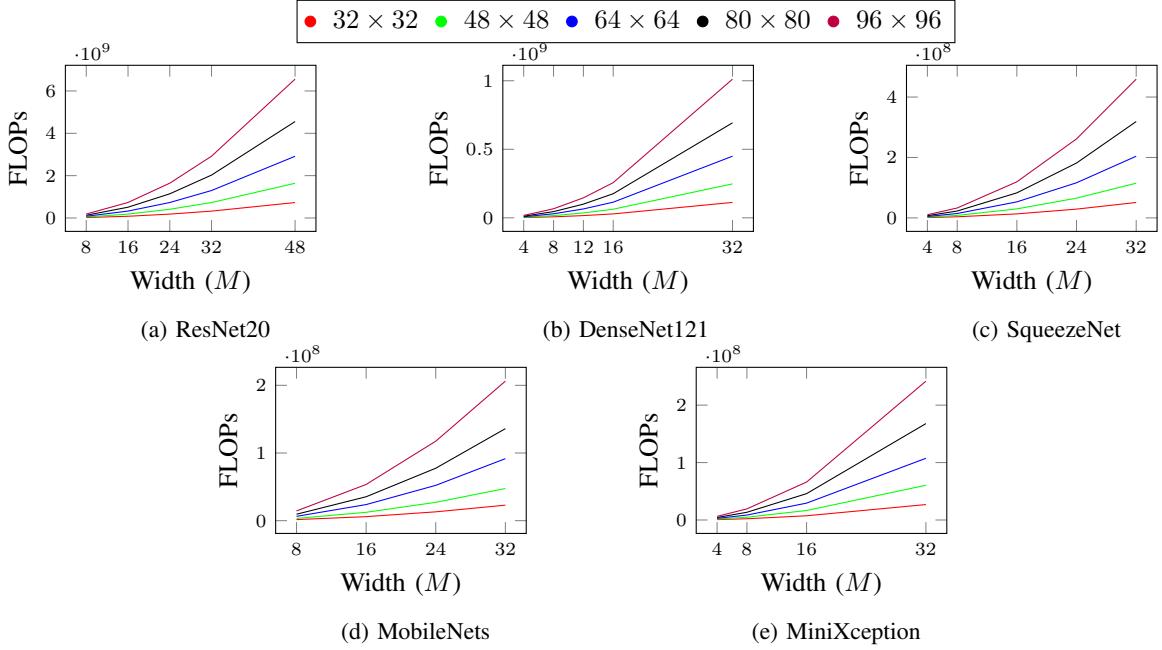


Fig. 2: Computational performance measured by FLOPs as width and image size is varied.

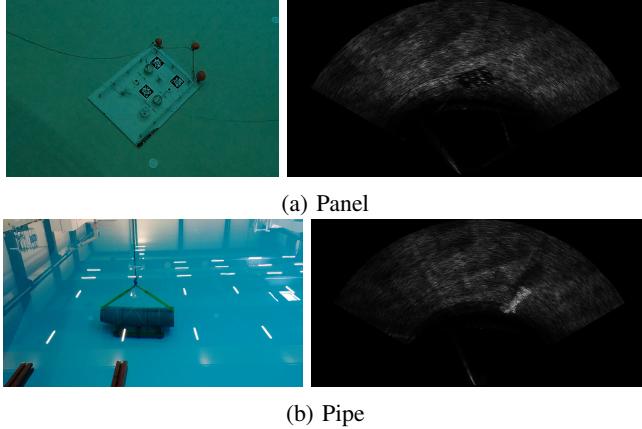


Fig. 3: Sample of optical and sonar images from the Gemini Panel-Pipe dataset.

completely independent dataset for evaluation.

**Gemini 720i Panel-Pipe.** This dataset contains 600 images captured using a Tritech Gemini 720i FLS. We captured two objects, a concrete pipe, and a hard plastic panel, leading to only two semantic classes. The training set is composed of 450 images, and the test set of 150 images. The dataset is mostly balanced, with 290 images for panel and 310 images for concrete pipe. One sample for each of these objects in optical and sonar modalities is shown in Figure 3. This dataset is used as a independent evaluation of transfer learning of our pre-trained models.

Both the Watertank and Turntable datasets are publicly available at <https://github.com/mvaldenegro/marine-debris-fls-datasets/releases/>

### B. Model Training

We train each architecture using the Adam optimizer with a learning rate  $\alpha = 0.001$ , for 200 epochs. We use a single set of data augmentations for all architectures, corresponding to random shift  $s_w, s_h$  along the width and height dimensions, sampled from a uniform distribution  $s_w \sim U(0, 0.1w)$  and  $s_h \sim U(0, 0.1h)$ , and random up-down and left-right flips with 50% probability each.

Normalization of this dataset proved to be a bit difficult, as the standard normalization by dividing pixel values by 255 did not work, and we resorted to subtracting the pixel mean computed on the training set, which allowed for all models to train correctly. We used the mean value  $\mu_p = 84.5$ . Note that this mean varies slightly depending on the input image size. Normalization can be applied using Eq 1.

$$x_{\text{norm}} = x - \mu_p \quad (1)$$

Assuming that the input image  $x$  contain pixels in the range  $[0, 255]$ . The training set and any new images are normalized in the same way.

Results of training all architectures on the Marine Debris Turntable dataset is shown in Table II. The best performing models overall are DenseNet and SqueezeNet. Highest accuracies are obtained with the largest input image sizes, while the smallest image size ( $32 \times 32$ ) is quite difficult to classify for ResNet and SqueezeNet, but the overall trend is that performance increases with the input image size.

### C. Transfer Learning on Watertank Dataset

For each architecture, we select the  $96 \times 96$  input image variation, and we perform transfer learning, targeting the Marine Debris Watertank dataset.

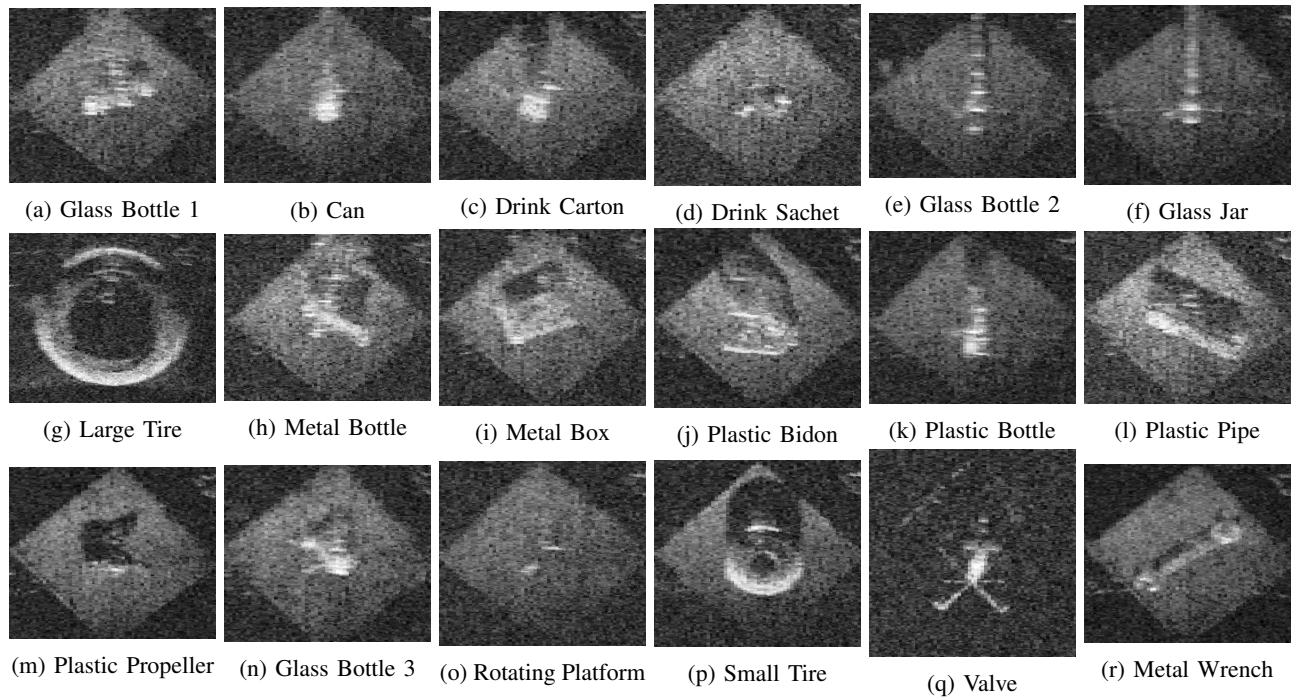


Fig. 4: Samples from the Marine Debris Turntable dataset, captured using a ARIS Explorer 3000 FLS, and containing 18 kinds of objects of varying materials and shapes, which are grouped into 12 semantic classes.

Model	Test Accuracy				
	32 <sup>+32</sup>	48 <sup>+48</sup>	64 <sup>+64</sup>	80 <sup>+80</sup>	96 <sup>+96</sup>
ResNet20	73.2%	96.8%	<b>99.1%</b>	90.9%	93.6%
MobileNet	95.7%	97.4%	96.9%	94.4%	96.7%
DenseNet121	<b>97.8%</b>	<b>99.4%</b>	98.9%	<b>99.1%</b>	<b>99.4%</b>
SqueezeNet	92.6%	98.8%	<b>99.4%</b>	90.7%	<b>99.7%</b>
MiniXception	96.9%	81.6%	61.9%	96.0%	93.8%
Linear SVM	93.3%	96.3%	97.5%	97.5%	97.4%
Image mean $\mu_p$	84.54	84.53	84.51	84.51	84.51

TABLE II: Classification results on the test set of the Marine Debris Turntable dataset, as the input image size is varied. The bottom row contains the detailed image mean used to train each of the models at the given input image size.

As transfer learning generally improves sample complexity (higher accuracy with less training samples), we sub-sample the training set to contain  $spc \in [1, 5, 10, 20, 30, 40, 50]$  samples for each class, which simulates a low-shot setting. We extract features in the sub-sampled training set, using a selection of layers close to the output in each architecture (usually the Flatten or last ReLU/Batch Normalization layer), and use them to train a support vector machine with  $C = 1.0$  and a linear kernel. We perform 10 runs of each experiment (value of  $spc$ ) and present the mean and standard deviation of accuracy.

The purpose of using a linear SVM is to evaluate the quality of the features, without going into difficulties with the

optimization process or tuning hyper-parameters of new fully connected layers or kernels for SVMs. It is a simple test for linear separability.

To quantify the sample complexity of each model, we plot test accuracy vs the number of samples in the training set, producing a curve we call the sample complexity curve, and the area under this curve defines a sample complexity metric. We call this metric area under the sample complexity curve (AUSCC), but note that this is not the same AUC as in the ROC curve. This allows to select the model that best performs across all samples per class points. The AUSCC is computed using the mean accuracy as the curve.

We use two methods as baselines. A linear SVM with  $C = 1.0$  on the raw pixels, and the features produced by the encoder trained in an unsupervised autoencoder, as the feature size  $c$  is varied from 4 to 128 features.

Overall results are presented in Table III. Individual performance with a selected set of layers to produce features are shown in Figure 5. We select the best layer for feature extraction based on the highest AUSCC. The best AUSCC is obtained by ResNet20 using the flatten\_5 layer (this is the last layer before the final fully connected layer), with other models closely following with slightly smaller AUSCC, around 1-2% less.

#### D. Transfer Learning on Gemini 720i Panel-Pipe Dataset

There is some intersection in the object classes between the watertank and turntable datasets. For a more independent evaluation of the feature quality produced by our pre-trained models, an independent dataset is required. For this purpose

PT Model	Layer	Test Accuracy						AUSCC
		1 Sample	5 Samples	10 Samples	20 Samples	30 Samples	50 Samples	
ResNet20	flatten_5	44.6 $\pm$ 7.5%	76.5 $\pm$ 4.2%	85.0 $\pm$ 2.8%	91.9 $\pm$ 1.5%	94.9 $\pm$ 1.0%	96.1 $\pm$ 1.1%	<b>89.28%</b>
	activation_93	39.6 $\pm$ 4.8%	74.4 $\pm$ 3.0%	83.5 $\pm$ 2.3%	89.7 $\pm$ 2.0%	93.0 $\pm$ 1.6%	95.6 $\pm$ 0.6%	87.47%
	activation_91	33.2 $\pm$ 4.4%	74.0 $\pm$ 2.6%	80.9 $\pm$ 2.1%	87.7 $\pm$ 1.3%	91.1 $\pm$ 0.8%	94.1 $\pm$ 1.0%	85.62%
MobileNet	conv_pw_11_relu	43.5 $\pm$ 5.0%	74.5 $\pm$ 2.2%	84.3 $\pm$ 3.5%	89.5 $\pm$ 1.0%	94.1 $\pm$ 0.9%	95.6 $\pm$ 1.1%	<b>88.06%</b>
	flatten_5	36.3 $\pm$ 1.7%	68.8 $\pm$ 2.8%	77.6 $\pm$ 2.2%	86.3 $\pm$ 2.6%	92.2 $\pm$ 1.8%	94.0 $\pm$ 1.3%	84.65%
	conv_pw_12_relu	40.3 $\pm$ 7.1%	70.6 $\pm$ 2.7%	80.4 $\pm$ 2.7%	89.2 $\pm$ 1.9%	92.9 $\pm$ 1.4%	94.7 $\pm$ 0.7%	86.46%
DenseNet121	conv5_block15_0_relu	41.3 $\pm$ 7.5%	72.9 $\pm$ 2.8%	85.8 $\pm$ 2.4%	92.0 $\pm$ 1.2%	93.9 $\pm$ 1.0%	95.8 $\pm$ 1.4%	<b>88.76%</b>
	conv5_block16_0_relu	40.5 $\pm$ 4.6%	74.0 $\pm$ 4.4%	83.6 $\pm$ 1.7%	91.2 $\pm$ 1.9%	94.7 $\pm$ 1.1%	96.9 $\pm$ 1.0%	88.58%
	avg_pool	33.8 $\pm$ 6.7%	62.7 $\pm$ 2.2%	75.2 $\pm$ 1.9%	83.5 $\pm$ 2.8%	89.4 $\pm$ 1.7%	92.5 $\pm$ 0.8%	82.12%
SqueezeNet	batch_norm_48	39.9 $\pm$ 5.6%	75.2 $\pm$ 2.9%	83.0 $\pm$ 1.8%	90.3 $\pm$ 0.7%	94.6 $\pm$ 0.6%	96.7 $\pm$ 0.9%	<b>88.48%</b>
	batch_norm_49	43.6 $\pm$ 3.8%	72.5 $\pm$ 2.6%	82.9 $\pm$ 3.0%	89.9 $\pm$ 1.5%	93.5 $\pm$ 1.0%	94.9 $\pm$ 1.0%	87.54%
	batch_norm_50	29.2 $\pm$ 5.1%	55.1 $\pm$ 4.5%	66.0 $\pm$ 2.6%	73.7 $\pm$ 1.2%	79.2 $\pm$ 2.3%	85.2 $\pm$ 1.5%	72.89%
MiniXception	add_19	45.5 $\pm$ 6.1%	75.9 $\pm$ 2.8%	84.4 $\pm$ 0.8%	91.4 $\pm$ 1.6%	93.6 $\pm$ 1.3%	96.3 $\pm$ 0.9%	<b>88.74%</b>
	add_18	44.3 $\pm$ 4.7%	72.5 $\pm$ 3.4%	81.4 $\pm$ 2.9%	87.5 $\pm$ 1.1%	91.8 $\pm$ 1.0%	95.1 $\pm$ 0.8%	86.23%
	conv2d_35	37.5 $\pm$ 5.8%	66.0 $\pm$ 2.8%	77.5 $\pm$ 2.0%	85.5 $\pm$ 2.1%	89.2 $\pm$ 1.9%	92.1 $\pm$ 0.9%	83.04%
Autoencoder	enc_code (128)	46.4 $\pm$ 3.2%	75.3 $\pm$ 3.2%	81.0 $\pm$ 2.7%	86.6 $\pm$ 1.1%	89.3 $\pm$ 1.9%	92.3 $\pm$ 1.0%	<b>85.11%</b>
	enc_code (64)	45.6 $\pm$ 5.2%	72.5 $\pm$ 2.6%	78.6 $\pm$ 1.9%	83.9 $\pm$ 1.7%	86.2 $\pm$ 1.2%	90.1 $\pm$ 1.3%	82.65%
	enc_code (32)	44.3 $\pm$ 8.8%	68.2 $\pm$ 2.3%	74.3 $\pm$ 3.1%	80.4 $\pm$ 2.0%	82.9 $\pm$ 1.6%	85.1 $\pm$ 0.6%	78.60%
Linear SVM	NA	40.2 $\pm$ 6.1%	71.5 $\pm$ 4.0%	81.3 $\pm$ 1.6%	88.8 $\pm$ 1.0%	89.6 $\pm$ 1.5%	92.9 $\pm$ 1.2%	85.3%

TABLE III: Transfer Learning results on the Marine Debris Watertank dataset, as the size of the training set is varied, and comparing multiple feature layers per model. Samples columns indicate how many samples per class are in the training set. AUSCC indicates the area under the sample complexity curve.

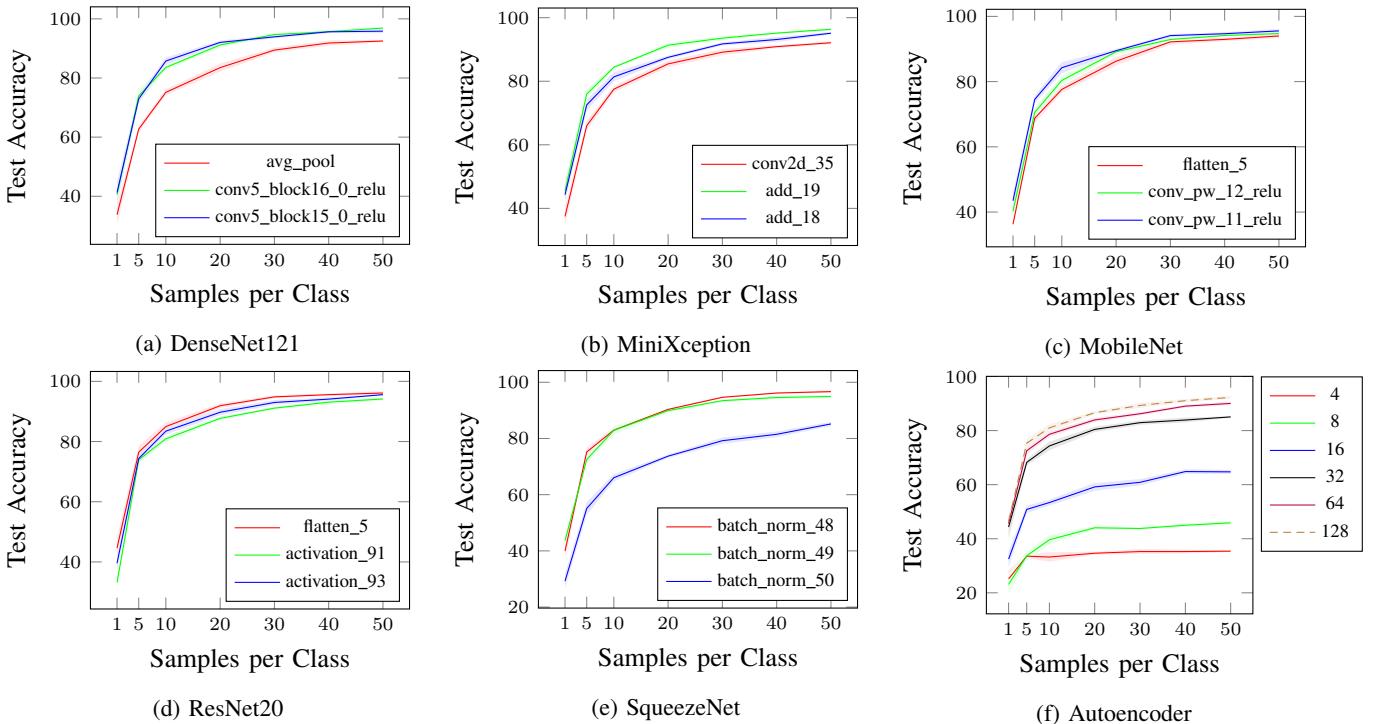


Fig. 5: Comparison of transfer learning on a low-shot setting on the Marine Debris Watertank dataset with multiple models and selected layers for feature extraction. Shaded areas represents one- $\sigma$  confidence intervals.

we use the Gemini 720i Panel-Pipe dataset, where the object classes are completely independent from the other two datasets, and also this dataset was captured using a different sensor (the Gemini 720i FLS), which can be used to evaluate cross-sensor transferability.

For this purpose we use the same evaluation protocol as with the watertank dataset, but only using the feature extraction layers that produce the best AUSCC for each model. Results are presented in Table IV and Figure 7.

Overall the best performing model in this experiment is

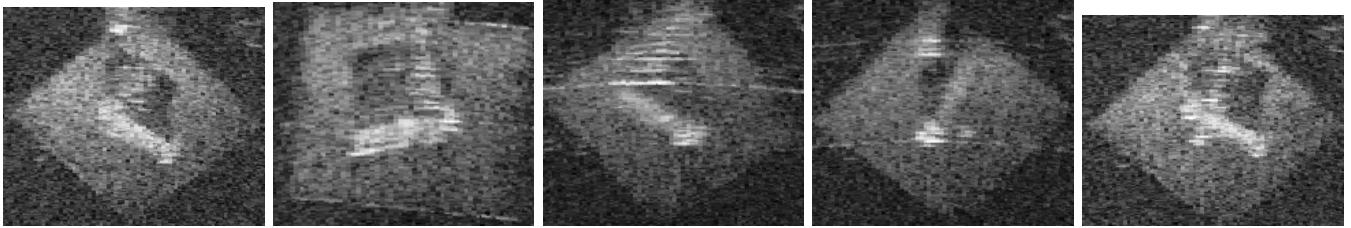


Fig. 6: Sample images from the Turntable dataset, showing different views of the metal bottle class, and how it visually changes as the turntable rotates.

MiniXception, with AUSCC of 96.31%, closely followed by SqueezeNet. There is a large variation across different number of samples in the training set, as ResNet20 performs better in the low-shot setting at one samples per class, but slightly under-performing for higher samples per class (like 10 and 20 samples).

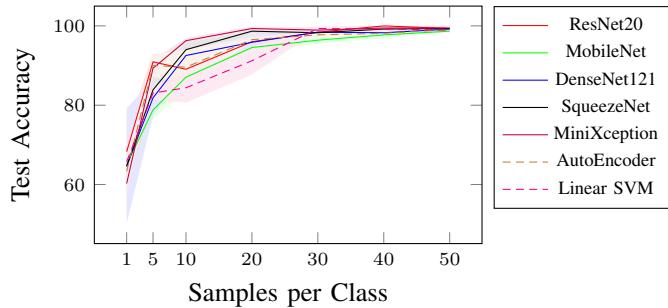


Fig. 7: Comparison of multiple models on Transfer Learning on the Gemini 720i Panel-Pipe Dataset. Shaded areas represents one- $\sigma$  confidence intervals.

### E. Discussion and Analysis

In both experiments we see an increase in the variation of accuracy with less samples, with one sample per class having the highest standard deviation, indicating that while good performance can be achieved, it depends on which exact samples are in the training set. More samples stabilize training.

The SVM baselines trained on raw pixel data are overall the weakest in performance, showing the importance of learning appropriate features specific to sonar images.

We believe that our results show that transfer learning using these pre-trained models can help reduce sample complexity, particularly in low-shot settings that are common in underwater perception.

Of the architectures we have selected, there is no particular architecture that outperforms the rest in all cases. This is known as the free lunch theorem in the ML community, where in average no ML model outperforms the rest. This motivates the selection and use of multiple models in this work, as some models will work better in some situations, while other architectures might perform adequately in other situations or modalities. For example, ResNet20 works quite well in the low-shot setting in the Gemini 720i Panel-Pipe dataset, but

when more samples are available (from around 10 samples per class), it is outperformed by SqueezeNet and MiniXception, while ResNet20 overall has the best performance in the Marine Debris Watertank dataset.

For the future user of these models, it is important to always evaluate multiple models and choose the one that fits their needs in terms of task performance, computation time, etc. Figures 8 and 9 can help make this decision.

### V. CONCLUSIONS AND FUTURE WORK

In this paper we have presented our concept of pre-trained convolutional neural networks for sonar images, as to fill the missing gap of pre-trained models specifically designed and tuned for sonar images. We trained our models on the turntable dataset, and evaluated our these models in the task of low-shot sonar image classification on the watertank (some object classes are common between turntable and watertank datasets) and panel-pipe datasets (no classes in common), showing that the learned features are useful for classification with low number of samples per class, outperforming the autoencoder and linear SVM baselines.

We expect that our pre-trained models will be used in autonomous underwater vehicles, as they are tuned for maximum classification and computational performance. This allows the designer to select the most appropriate architecture and input image size, and our paper provides information about the trade-offs between task and computational performance.

All pre-trained models in this paper are publicly available at <https://github.com/mvaldenegro/pretrained-models-sonar-images/> as Keras<sup>4</sup> HDF5 model files, which can be loaded in other frameworks and should serve as a basis for future research.

### ACKNOWLEDGEMENTS

The authors would like to thank Leonard McLean from the Ocean Systems Lab, Heriot-Watt University, for his help in capturing data used in this paper.

### REFERENCES

- [1] M. Huh, P. Agrawal, and A. A. Efros, “What makes imagenet good for transfer learning?,” *arXiv preprint arXiv:1608.08614*, 2016.
- [2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., “Imagenet large scale visual recognition challenge,” *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

<sup>4</sup>Trained using Keras 2.3.1 on TensorFlow 1.14.0

PT Model	Layer	Test Accuracy						AUSCC
		1 Sample	5 Samples	10 Samples	20 Samples	30 Samples	50 Samples	
ResNet20	flatten_5	<b>68.3 ± 11.4%</b>	90.9 ± 4.4%	89.1 ± 5.6%	96.0 ± 4.1%	98.4 ± 2.9%	99.3 ± 1.3%	95.00%
MobileNet	conv_pw_11_relu	65.1 ± 9.7%	78.8 ± 4.7%	87.1 ± 6.4%	94.5 ± 2.6%	96.4 ± 1.8%	98.7 ± 0.4%	92.20%
DenseNet121	conv5_block15_0_relu	64.7 ± 29.1%	81.9 ± 9.1%	92.5 ± 6.1%	95.9 ± 3.6%	98.4 ± 1.1%	99.2 ± 0.7%	94.14%
SqueezeNet	batch_norm_48	60.9 ± 7.6%	84.4 ± 9.7%	<b>95.9 ± 4.5%</b>	96.1 ± 4.6%	<b>99.5 ± 0.8%</b>	<b>100.0 ± 0.0%</b>	95.25%
MiniXception	add_19	60.3 ± 5.7%	89.3 ± 4.3%	<b>96.3 ± 1.4%</b>	<b>99.3 ± 1.0%</b>	<b>98.9 ± 1.3%</b>	<b>99.5 ± 1.1%</b>	<b>96.31%</b>
Autoencoder	enc_code	63.3 ± 7.3%	90.0 ± 5.6%	89.6 ± 8.8%	96.5 ± 1.4%	97.7 ± 1.2%	99.1 ± 1.0%	94.35%
Linear SVM	NA	65.7 ± 8.8%	83.1% ± 3.3	84.4 ± 7.5%	91.2 ± 6.9%	99.3 ± 1.0%	98.9 ± 1.3%	92.45%

TABLE IV: Transfer Learning results on the Gemini 720i Panel-Pipe dataset, using the best feature layers, as the size of the training set is varied. Samples columns indicate how many samples per class are in the training set. AUSCC indicates the area under the sample complexity curve.

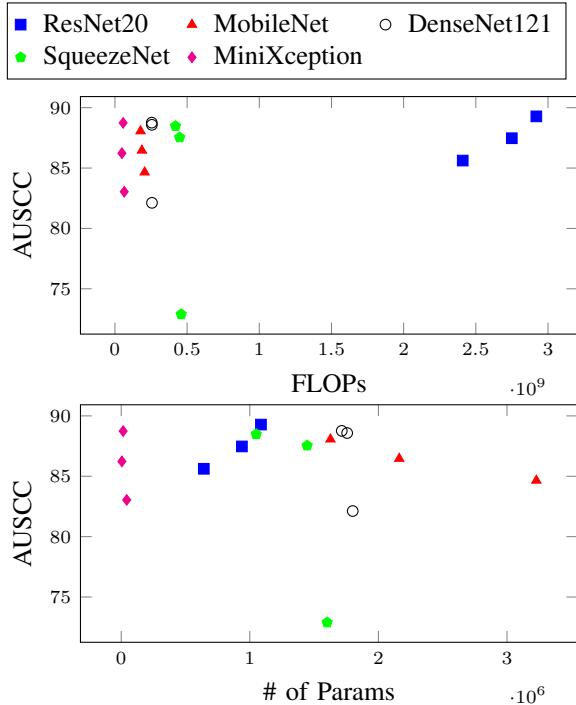


Fig. 8: Comparison of all architectures and selected layers on the Watertank dataset, in terms of AUSCC versus FLOPs and number of parameters in the model.

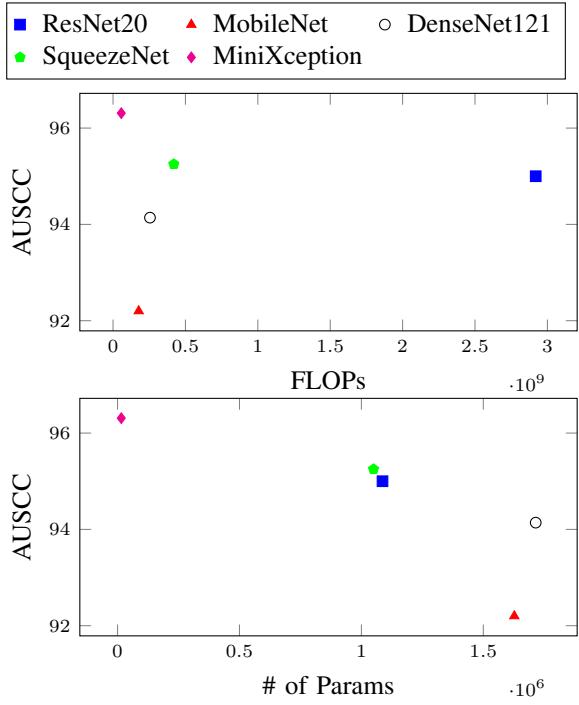


Fig. 9: Comparison of all architectures with their best layer on the Gemini 720i Panel-Pipe dataset, in terms of AUSCC versus FLOPs and number of parameters in the model.

- [3] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Mallochi, A. Kolesnikov, *et al.*, “The open images dataset v4,” *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.
- [4] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “Cnn features off-the-shelf: an astounding baseline for recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.
- [5] M. Valdenegro-Toro, “Best practices in convolutional networks for forward-looking sonar image recognition,” in *OCEANS 2017-Aberdeen*, pp. 1–9, IEEE, 2017.
- [6] L. R. Fuchs, A. Gällström, and J. Folkesson, “Object recognition in forward looking sonar images using transfer learning,” in *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pp. 1–6, IEEE, 2018.
- [7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilennets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *CVPR*, vol. 1, p. 3, 2017.
- [10] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.
- [11] O. Arriaga, M. Valdenegro-Toro, and P. Plöger, “Real-time convolutional neural networks for emotion and gender classification,” *ESANN*, 2019.
- [12] G. Lou, R. Zheng, M. Liu, and S. Zhang, “Automatic target recognition in forward-looking sonar images using transfer learning,” in *Global Oceans 2020: Singapore-US Gulf Coast*, pp. 1–6, IEEE, 2020.
- [13] M. Valdenegro-Toro, “Deep neural networks for marine debris detection in sonar images,” *arXiv preprint arXiv:1905.05241*, 2019.