Deep Representation Learning for Orca Call Type Classification

Christian Bergler[1], Manuel Schmitt[1], Rachael Xi Cheng[2], Hendrik, Schröter[1], Andreas Maier[1], Volker Barth[3], Michael Weber[3], and Elmar Nöth[1]

[1] Friedrich-Alexander-University Erlangen-Nuremberg, Department of Computer Science – Pattern Recognition Lab, Martensstr. 3, 91058 Erlangen, Germany
`https://www5.cs.fau.de`
`{christian.bergler,elmar.noeth}@fau.de`
[2] Leibniz Institute for Zoo and Wildlife Research (IZW) in the Forschungsverbund Berlin e.V., Alfred-Kowalke-Str. 17, 10315 Berlin, Germany
[3] Anthro-Media, Nansenstr. 19, 12047 Berlin, Germany

Abstract. Marine mammals produce a wide variety of vocalizations. There is a growing need for robust automatic classification methods especially in noisy underwater environments in order to access large amounts of bioacoustic signals and to replace tedious and error prone human perceptual classification. In case of the northern resident killer whale *(Orcinus orca)*, echolocation clicks, whistles, and pulsed calls make up its vocal repertoire. Pulsed calls are the most intensively studied type of vocalization. In this study we propose a hybrid call type classification approach outperforming our previous work on supervised call type classification consisting of two components: (1) deep representation learning of killer whale sounds by investigating various autoencoder architectures and data corpora and (2) subsequent supervised training of a ResNet18 call type classifier on a much smaller dataset by using the pre-trained representations. The best semi-supervised trained classification model achieved a test accuracy of 96 % and a mean test accuracy of 94 % outperforming our previous work by 7 percent points.

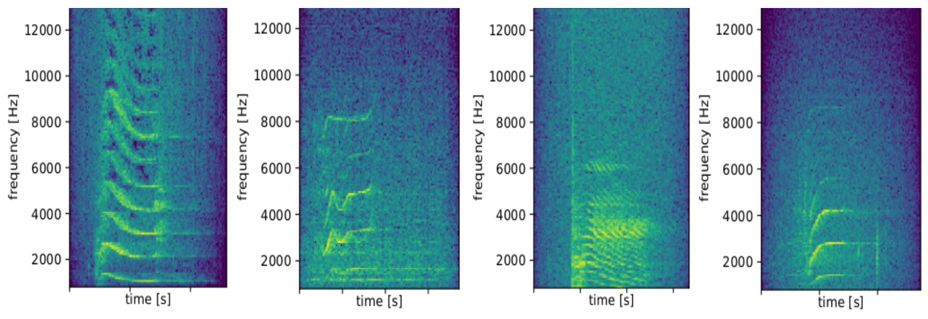
Keywords: Deep Learning, Classification, Representation Learning, Bioacoustics, Orca, Killer Whale, Call Type

1 Introduction

An increasing use of passive acoustic monitoring of various animal species result in massive quantity of bioacoustic data. For example, the Orcalab [20] has collected underwater recordings on killer whales for 23 years resulting in about 20,000 hours. There is a growing need for effective methods of automatic classification of bioacoustic signals. It offers significant advantages as in assessing large datasets, frees humans from time-consuming and labor intensive work, and offers rigorous and consistent results.

The authors would like to thank Helena Symonds and Paul Spong from Orcalab, and Steven Ness, formerly UVIC, for giving us permission to use the raw data and annotations from orcalab.org, and the Paul G. Allen Frontiers Group for their initial grant for the pilot research.

Killer whales *(Orcinus orca)*, the largest member of the dolphin family, are one of several species with relatively well-studied and complex vocal cultures [7]. Extensive research on killer whale acoustic behavior has been conducted on the resident fish-eating killer whales in the northeast Pacific. Resident killer whales live in stable matrilineal units [2]. Those matrilines, that often travel together to socialize on a regular basis, form subpods and pods [15, 8, 9, 2]. Apart from echolocation clicks and whistles, killer whales produce a number of social sounds with distinct frequency contours which are group specific. Those pulsed calls, the most common and excessively studied type of killer whale vocalization, are classified into discrete, variable, and aberrant calls. It typically shows sudden and patterned shifts in frequency, according to the pulse repetition rate, which is normally between 250 and 2000 Hz [10]. Acoustically related animals are assigned to a so-called clan, an acoustic grouping of pods that have one or more discrete calls in common [2]. Basically, all pods of a clan have a common repertoire of calls, with slight vocal distinctions in between [11]. Due to the resulting variety of call variants, group-specific dialects arise [11]. Those pod-specific dialects consist of up to 20 types of discrete calls each, and in total the northern residents vocal repertoire of discrete calls consists of more than 40 types [11, 9] (examples in Figure 1). Call structure variations can be observed in various shared call types [18]. Group-specific vocal signals are believed to play an important role in maintaining contact among members or coordinate group activities, especially when the group is dispersed and when visual signals can only be used in short-distance communication [10]. The current study builds on the previously achieved deep learning-based segmentation [22] and tries to improve our previous supervised call type classification result [22]. Unsupervised deep representation learning and subsequent classification could also be very helpful improving language or even speaker dependent classification/identification models. Moreover, unsupervised deep representation learning of (compressed) bottleneck features can support to identify and examine indigenous languages.



a) A4 N04 **b)** A5 N02 **c)** A12 N03 **d)** A36 N09

Fig. 1: Pod-specific (A4, A5, A12, A36) killer whale call type (N04, N02, N03, N09) spectrograms (sample rate = 44.1 khZ, FFT–size = 4096 samples (∼100 ms), hop–size = 441 samples (∼10 ms))

1.1 Related Work

Brown et al. [3] used dynamic time warping to compare the melodic contours of 57 captive killer whale vocalizations and k-means to cluster into 9 call types. Furthermore, Brown et al. [4] classified 75 killer whale calls into 7 call types using hidden Markov models and Gaussian mixture models resulting in more than 90 % agreement. Ness [19] classified between 12 various killer whale call types achieving an average accuracy of 76 % by using an SVM with a Radial Basis Function kernel. Brown et al. [5] did killer whale individual identification by distinguishing within four diverse animals via differentiating between one specific call type. Deecke et al. [6] introduced a method for dolphin and killer whale sound categorization via dynamic time warping and an adaptive resonance theory neural network. Mercado et al. [17] classified 242 humpback whale vocalizations via a combination of a source-filter model and an artifical neural network. Garland et al. [12] classified 1,019 Beluga whale sounds into 34 different call types using non-parametric classification tree analysis and random forest analysis achieving an accuracy of 83 %.

2 Methodology

Convolutional Neural Network (CNN)

Convolutional neural network (CNN) is a state-of-the-art end-to-end deep learning concept first used by LeCun et al. [16] for handwritten letter recognition. CNNs facilitate to efficiently handle 2-D input data (e.g. spectrograms). CNNs are designed after the traditional principle of pattern recognition, implementing covolutional layers for feature learning/extraction and subsequent fully-connected layers for classification [16]. Convolutional layers unite several very important architectural approaches: (1) local receptive fields, (2) shared weights, and (3) spatial/temporal sub-sampling (pooling) [16]. For a more detailed explanation of a CNN and its underlying concepts, see [16].

Residual Network (ResNet)

Training very deep neural networks in order to learn higher-level and more discriminative features results in various optimization problems (vanishing/exploding gradients, degradation problem) [14]. He et al. [14] introduced a residual learning framework in an architecture called residual network (ResNet), using residual mappings to not directly learn and optimize an unreferenced underlying mapping $H(x)$ with respect to the input x but rather a residual mapping $F(x) = H(x) - x$, in order to counteract the degradation problem. Moreover, He et al. [14] present different and typical ResNet architectures based on their number of concatenated layers which have proven successful in practice. For a more detailed explanation about deep residual learning, see [14].

Autoencoder

An autoencoder is a (deep) neural network architecture, trying to map a given input x to an output/reconstruction r via a hidden representation h [13]. This architecture consists of two basic components: (1) an encoder e acting as a function, mapping the input

x to the hidden layer representation h via $h = e(x)$, (2) a decoder d officiating as a function which maps the hidden layer latent code h to the output/reconstruction r via $r = d(h)$ [13]. In this work we used various *residual-based convolutional undercomplete autoencoders* constraining h to a smaller dimension than the input x [13] in order to learn an embedding h comprising the most prominent features via minimizing the loss $L(x, d(e(x)))$, penalizing the dissimilarity between x and $d(e(x))$ [13].

Deep Representation Learning

Representation learning is a way of learning useful data representations (features), directly on the given input data, rather than performing a labor-intensive feature extraction/selection based on handcrafted features (feature engineering), in order to utilize them for a subsequent classification task [1]. Usually the amount of unlabeled data is much higher than the one of labeled data. Consequently, a pure supervised training on limited labeled data mostly results in overfitting [13] and a lack of robustness/generalization towards unseen real-world data especially in case of extremely heterogeneous data corpora. Representation learning provides an opportunity of combining unsupervised and supervised learning by using the unsupervised learned task-related representations as initialization of the original supervised task in order to produce a more accurate and robust semi-supervised trained classification model [13]. In this study deep learning techniques were used to derive adequate feature representations.

3 Datasets and Data Distribution

3.1 Datasets

Orchive Annotation Catalog (OAC)

Ness [19] published the OAC dataset in cooperation with the Orcalab [20], comprising 15,480 labeled underwater events (stereo, sampling rate: 44.1 kHz) extracted from the Orchive [19]. The annotations include various killer whale sounds and several noise samples [22]. For later killer whale specific representation learning, we extracted all valid killer whale signals (killer whale calls, whistles, echolocations) from the OAC containing 7,903 killer whale samples with a total annotated time of 9.96 h.

Automatic Extracted Orchive Tape Data (AEOTD)

In order to provide further killer whale signals for representation learning, we filtered all killer whale sounds from the AEOTD dataset described in [22]. The entire dataset includes 1,667 killer whale sounds with an annotation time of 1.4 h.

DeepAL Fieldwork Data 2017/2018 (DLFD)

During our research expedition in northern British Columbia (2017/2018) we have collected additional multi-channel killer whale and noise data via a 15-meter research trimaran using underwater microphone arrays [22]. According to [22] we selected four different channels out of the multi-channel labeled killer whale sound events to furthermore increase our overall killer whale data for representation learning. The DLFD comprises 3,331 killer whale signals and an overall annotation time of 3.40 h.

Orca Segmented Data (OSD)

The OSD corpus is a result of a fully automatic segmentation using our trained ResNet18 classifier [22], distinguishing between killer whale and noise sound events. To enlarge the existing database even further, we automatically extracted killer whale signals detected by our segmenter. The resulting OSD corpus comprises 19,211 killer whale signals and an overall annotated time of 34.47 h. Thus, according to [22], there should be about 4 % false positives within the OSD corpus.

Call Type Catalogs

For training, validation and testing of the call type classifier we used the same data pool as described in [22], consisting of two different call type catalogs – Orcalab catalog (CCS) with 138 killer whale sounds containing 7 various call type classes, Ness catalog (CCN) with 286 killer whale signals including 6 different call types – plus an extension catalog (EXT) including 30 echolocations, 30 whistles, and 30 noise files manually selected from the Orchive data [22] in order to simulate a real-world scenario. In total this results in 12 classes, consisting of 9 various call types, echolocations, whistles, and noise samples summing up to 514 samples (see Table 1).

Table 1: Orca call type, echolocation, whistle, and noise label distribution of the CCS, CCN, and EXT data corpus

Orca Call Type/ Corpus	**N01**	**N02**	**N03**	**N04**	**N05**	**N07**	**N09**	**N12**	**N47**	**echo**	**whistles**	**noise**	**SUM**
CCS	33	10	—	21	14	18	26	16	—	—	—	—	**138**
CCN	36	—	56	60	—	31	70	—	33	—	—	—	**286**
EXT	—	—	—	—	—	—	—	—	—	30	30	30	**90**
SUM	**69**	**10**	**56**	**81**	**14**	**49**	**96**	**16**	**33**	**30**	**30**	**30**	**514**

3.2 Data Distribution

In Table 2 the data distribution of the entire representation learning and whole call type classification data corpus is described. The representation learning data listed in Table 2a consists of the aforementioned OAC, AEOTD, DLFD, and OSD corpus summing up to a total amount of 32,112 killer whale sounds. The call type classification dataset in Table 2b includes the previously illustrated CCS, CCN, and EXT dataset summing up to an overall amount of 514 signals [22]. For the entire representation learning data corpus every file was removed which belonged to the same tape as one of the signals from the call type classification dataset. Consequently, the training, validation and test signals of the representation learning corpus are completely independent of those from the call type classification dataset.

Table 2: Training, validation, and test distribution for representation learning and call type classification

Split/ Datasets		train		val		test	
		smp	%	smp	%	smp	%
OAC	**7,903**	5,832	73.8	1,171	14.8	900	11.4
AEOTD	**1,667**	1,172	70.3	260	15.6	235	14.1
DLFD	**3,331**	1,384	41.5	1,171	35.2	776	23.3
OSD	**19,211**	13,493	70.2	2,863	14.9	2,855	14.8
SUM	**32,112**	21,881	68.1	5,465	17.0	4,766	14.8

a) Representation learning data distribution

Split/ Datasets		train		val		test	
		smp	%	smp	%	smp	%
CCS	**138**	102	73.9	19	13.8	17	12.3
CCN	**286**	198	69.2	41	14.4	47	16.4
EXT	**90**	63	70.0	12	13.3	15	16.7
SUM	**514**	363	70.6	72	14.0	79	15.4

b) Call type data distribution

4 Experimental Setup

4.1 ResNet18 Autoencoder

In this work we used an undercomplete autoencoder based on the ResNet18 [14] architecture. Figure 2 visualizes the utilized network architecture. For the bottleneck layer, various layer types were investigated: (1) convolutional layer (1×1 convolution without stride to compress 512 channels to $4\times16\times8$ and back to $512\times16\times8$) and (2) fully-connected layer (max-pooling of $512\times16\times8$ to a 512-D latent layer and subsequent max-unpooling back to $512\times16\times8$). The ResNet18 encoder architecture was slightly modified in terms of removing the 3×3 (stride 2) max-pooling from the first residual layer in order to keep higher frequency resolutions within the early stages [22]. The decoder utilized transposed convolutions for upsampling and slightly differs from the encoder. In order to avoid artifacts in our last layer, potentially caused by transposed convolutions with stride 2, we already upsampled to 256×128 in the penultimate layer and processed a final transposed convolution (stride 1) to compensate such errors.

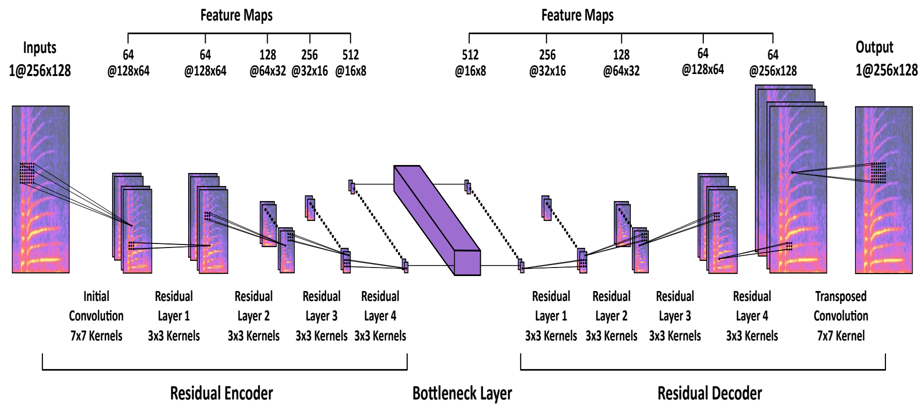

Feature Maps
Inputs 1@256x128
64 @128x64
64 @128x64
128 @64x32
256 @32x16
512 @16x8
Feature Maps
512 @16x8
256 @32x16
128 @64x32
64 @128x64
64 @256x128
Output 1@256x128
Initial Convolution 7x7 Kernels
Residual Layer 1 3x3 Kernels
Residual Layer 2 3x3 Kernels
Residual Layer 3 3x3 Kernels
Residual Layer 4 3x3 Kernels
Residual Layer 1 3x3 Kernels
Residual Layer 2 3x3 Kernels
Residual Layer 3 3x3 Kernels
Residual Layer 4 3x3 Kernels
Transposed Convolution 7x7 Kernel
Residual Encoder
Bottleneck Layer
Residual Decoder

Fig. 2: Architecture of the ResNet18 autoencoder with a parametric bottleneck layer

4.2 ResNet18 Call Type Classifier

Our call type classifier [22] is based on a ResNet18 architecture (feature extraction part) combined with a 512-D fully connected hidden layer and a subsequent 12-D output layer (classification part) in order to distinguish between 12 classes (see Table 1). Figure 3 visualizes the network architecture of our call type classifier [22].

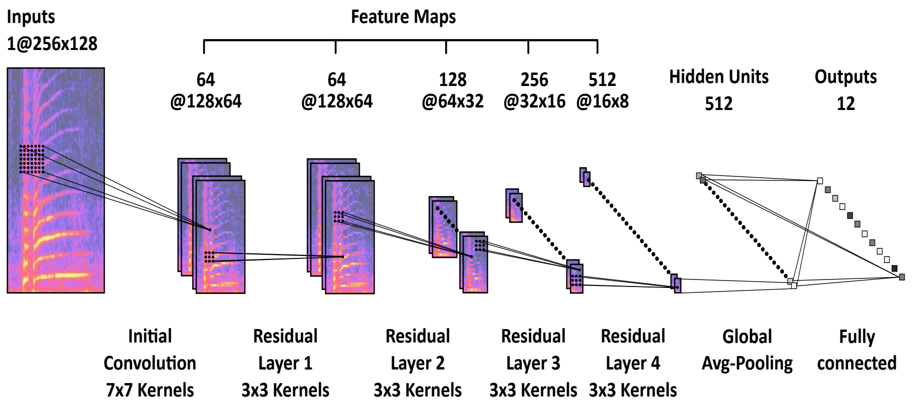

Inputs
1@256x128
Feature Maps
64 @128x64
64 @128x64
128 @64x32
256 @32x16
512 @16x8
Hidden Units 512
Outputs 12
Initial Convolution 7x7 Kernels
Residual Layer 1 3x3 Kernels
Residual Layer 2 3x3 Kernels
Residual Layer 3 3x3 Kernels
Residual Layer 4 3x3 Kernels
Global Avg-Pooling
Fully connected

Fig. 3: Architecture of the ResNet18 call type classifier [22]

4.3 Data preprocessing and augmentation

The ResNet18 based autoencoder (Figure 2) and call type classifier (Figure 3) used the same data preprocessing toolchain. As described in [22] every audio sample was converted to a 44.1 kHz mono wav signal, followed by a STFT (window size = 4,096, hop size = 441) transforming the audio to a power spectrogram [22]. The power spectrogram was converted to dB and further changed via various sequential ordered augmentation techniques, all using an uniform distributed random scaling [22]. Intensity (-6 – +3 dB), pitch (0.5 – 1.5), and time (0.5 – 2.0) augmentation were conducted first [22]. In a next step a linear frequency compression (fmin = 500 Hz, fmax = 10 kHz) was processed resulting in 256 frequency bins. Afterwards characteristic pitch and time augmented, frequency compressed noise files of the segmenter train set in [22] were added using a randomly chosen SNR between -3 and +12 dB [22]. Noise augmentation was only activated while training the classifier. A subsequent dB-normalization within -100 dB (minimum level) and +20 dB (reference level) was performed. To provide training clips of equal size we randomly chose a 1.28 s segment (if applicable zero-padding) of the final spectrogram resulting in a 256×128 large training sample [22].

4.4 Training, Validation, and Testing

All our trained models, implemented in PyTorch [21], used an Adam optimizer together with an initial learning rate of 10^{-5}, $\beta_1 = 0.5$, and $\beta_2 = 0.999$ [22]. The learning rate was decayed by $1/2$ after 4 epochs, and the entire training was stopped after 10 epochs without having any improvements on the validation set [22]. For deep representation learning, we utilized a batch size of 32 together with a weighted mean squared error (MSE) loss. For call type classification a batch size of 4 in combination with a 12-class cross entropy loss was used [22]. The lowest validation loss was selected as criterion for the best autoencoder model, whereas the highest validation accuracy was picked in case of finding the best call type classifier. The autoencoder was trained on the entire or portions of the data listed in Table 2a. The call type classifier was trained on the same data as in [22], listed in Table 2b. Furthermore, we computed a 10-fold cross-validation on the entire call type dataset in order to get a better impression about the overall model robustness.

4.5 Experiments

Our experimental setup is divided into three major parts: (1) investigations regarding the best ResNet18 autoencoder architecture trained on the entire or various data portions of Table 2a, (2) evaluating the impact of representation learning by training, validating, and testing our pretrained call type classifier on the given data corpus illustrated in Table 2b, and (3) analyzing the pretrained ResNet18 classifiers models by performing a 10-fold cross validation using the entire call type classification corpus in Table 2b. In (1) we examined two bottleneck architectures; linear versus convolutional architectures (see section 4.1) of our ResNet18 autoencoder (Figure 2) together with different data combinations: semi-automatic labeled data (entire representation corpora), fully hand-labeled data (only OAC corpus), and automatic labeled data (only OSD dataset), listed in Table 2a. In (2) we evaluated the call type classifier (Figure 3) by using the pre-trained autoencoder weights and calculating the accuracy based on 10 training/evaluation runs (Figure 5a). Moreover, in experiment (3), a 10-fold cross validation was conducted on the entire call type dataset listed in Table 2b in order to give an impression about the overall classifier robustness. Here we only evaluated the best pre-trained classifiers with respect to the trained data combinations (semi-automatic labeled, hand-labeled, automatic labeled; see Figure 5b).

5 Results

In this study six different ResNet18-based autoencoders were trained and analyzed utilizing linear or convolutional bottleneck layers for semi-automatic labeled, hand-labeled, and automatic labeled data corpora: (1) linear/convolutional autoencoder on the entire dataset listed in Table 2a (semi-automatic labeled data), (2) linear/convolutional autoencoder on the OAC dataset (hand-labeled data), and (3) linear/convolutional autoencoder on the OSD dataset (automatic machine-labeled data). Figure 4 shows the autoencoder signal reconstruction results based on the three different killer whale sound types taken from the call type test set listed in Table 2b. None of the reconstructed files was part of the training or validation. With respect to each trained dataset we only visualized the reconstructions of the autoencoders using the convolutional bottleneck layer, since they provided better results.

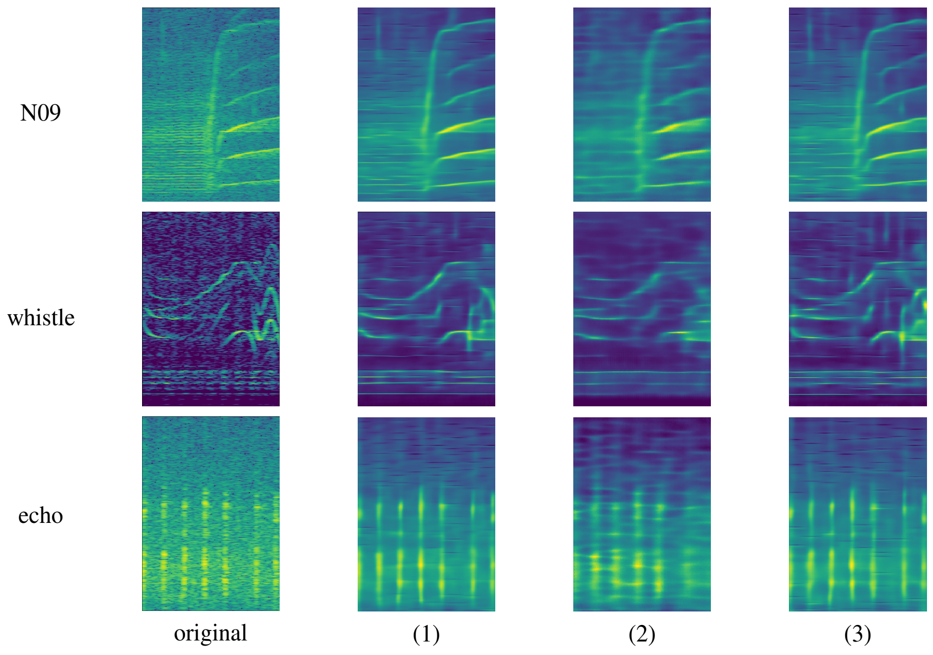


Fig. 4: Autoencoder (AE) reconstructions of killer whale sound types from the call type test set **(1)** AE (convolutional bottleneck) trained on the entire data listed in Table 2a (semi-automatic labeled data), **(2)** AE (convolutional bottleneck) trained on the OAC corpus (hand-labeled data) **(3)** AE (convolutional bottleneck) trained on the OSD dataset (automatic machine-labeled data)

All of our six pretrained autoencoder encoder parts were separately used for weight initialization of the call type classifier which was trained on the data listed in Table 2b. Therefore we removed bottleneck layer and decoder part in order to only use the encoder part combined with a global average pooling and subsequent fully-connected layer followed by a 12-dimensional output layer in order to classify between the different sound events (see Figure 3). Deeper layers learn more specific and high-level features. In the case of representation learning, the last residual layer learns to provide a good basis for a successful reconstruction rather than an accurate classification. Hence, we randomly initialized the last residual layer and did not use the pretrained weights in that case. Figure 5 visualizes the impact of deep representation learning with respect to the call type classification accuracy. All six pretrained autoencoders and their classification results are illustrated. The notation used for the various autoencoder variants consists of a number (1, 2, 3) illustrating the entire dataset (1), OAC dataset (2), and the OSD dataset (3) as well as a letter (c, l) describing if a convolutional (c) or linear (l) bottleneck layer was used (e.g. 3-c describes an autoencoder with a linear bottleneck layer trained on the OSD dataset). In addition a result without any pretraining (Figure 5a, row 4) was added. The statistics about the accuracy were based on 10 training/evaluation runs. Furthermore, we also put the mean test accuracy of our previous work [22] to the graph which corresponds to only 5 runs (Figure 5a, row 5). The average accuracy for every pretrained classifier was better than without pretraining. According to Figure 5a, the best test performance on average (94 %), smallest variance/stdv, and the best single model accuracy (96 %) was achieved by the pretrained classifier using the convolutional autoencoder (3-c) trained on the fully automatic machine-segmented OSD dataset. Figure 6 visualizes the confusion matrix of our best model (3-c, 96 % accuracy) compared to the matrix illustrated in our previous work [22]. In order to compute the overall classifier accuracy with respect to the entire call type dataset (Table 2b) we conducted a 10-fold cross validiation. For each dataset we selected the pretrained autoencoder version which led to a better subsequent classification accuracy. In all cases the autoencoder with the convolutional bottleneck layer outperformed the linear variant. Consequently, we used autoencoder 1-c, 2-c, and 3-c for weight initialization of the different classifiers to run the 10-fold cross validation. The classification results about the 10-fold cross validation are shown in Figure 5b. The best semi-supervised trained call type classifier (Figure 5b, 3-c) used the machine-segmented OSD dataset and achieved the highest mean test accuracy of 90 % whereas one test fold reached up to 98 %.

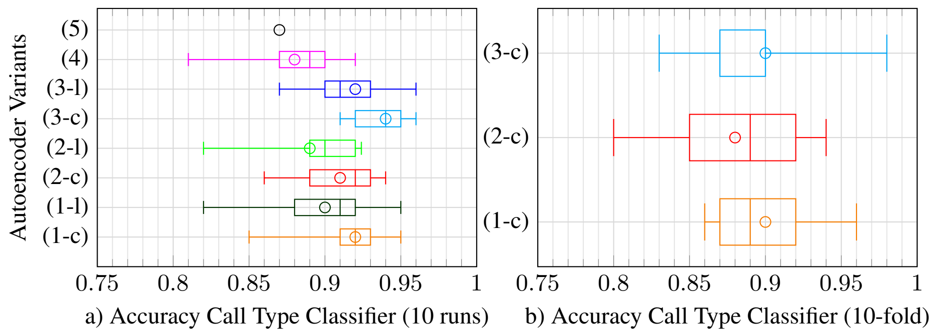


Fig. 5: a) Mean test accuracy of 10 train/evaluation runs: (1-c) convolutional, (1-l) linear AE on the entire data listed in Table 2a, (2-c) convolutional, (2-l) linear AE on the OAC corpus, (3-c) convolutional, (3-l) linear AE on the OSD dataset, (4) no pretrain (5) mean test accuracy of [22] **b)** Classifier accuracy in a 10-fold cross validation (Table 2b) for the top 3 AEs (1-c), (2-c), (3-c)

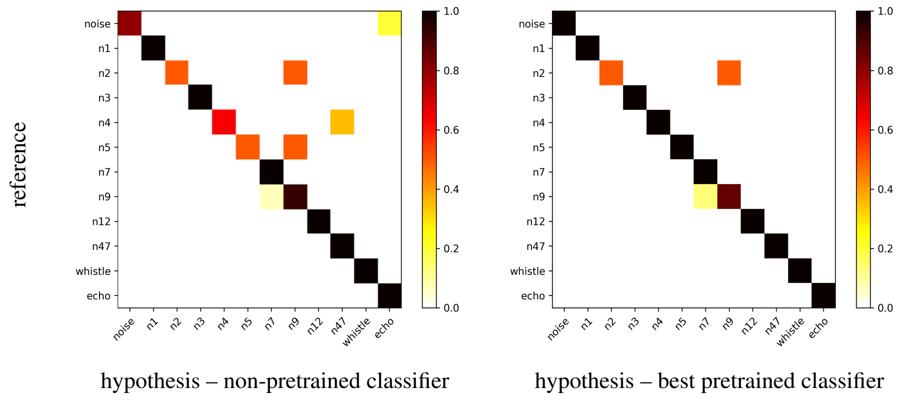


Fig. 6: Confusion matrix (12-classes) – non-pretrained call type classifier (ACC = 87 %) [22] vs. best pretrained call type classifier (ACC = 96 %))

6 Conclusion

In summary, deep representation learning has a significant positive influence on killer whale call type classification. Particularly important is the fact that regardless of the autoencoder architecture, as well as from the utilized data corpora, any form of representation learning has led to an improvement referring to the mean test classification accuracy. Moreover, pretraining on our fully automatic machine-labeled OSD corpus led to the best performance being a great indicator of having a robust and reliable segmentation process [22]. In future work we will segment the entire 20,000 h of underwater recordings for further killer whale representation learning. Moreover, we plan to investigate the huge variety of machine-segmented killer whale call types by using various feature learning techniques combined with subsequent clustering methods in order to establish a fully unsupervised pipeline for killer whale call type identification/classification. On the one hand, this allows us to explore finer, more significant and potential undiscovered killer whale call types. On the other hand, the entire call type classification can be performed completely independent from human perception. Furthermore, the unsupervised identified call types and possible sub-call types can be also used for subsequent supervised learning approaches.

References

1. Bengio, Y., Courville, A.C., Vincent, P.: Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence 35, 1798–1828 (2013)
2. Bigg, M.A., Olesiuk, P.F., Ellis, G.M., Ford, J.K.B., Balcomb, K.C.: Organization and genealogy of resident killer whales (Orcinus orca) in the coastal waters of British Columbia and Washington State. International Whaling Commission pp. 383–405 (January 1990)
3. Brown, J., Hodgins-Davis, A., Miller, P.: Classification of vocalizations of killer whales using dynamic time warping. JASA Express Letters 119(3), 617–628 (March 2006)
4. Brown, J.C., Smaragdis, P.: Hidden Markov and Gaussian mixture models for automatic call classification. The Journal of the Acoustical Society of America 125, 221–224 (2009)
5. Brown, J.C., Smaragdis, P., Nousek-McGregor, A.: Automatic identification of individual killer whales. The Journal of the Acoustical Society of America 128, 93–98 (June 2010)
6. Deecke, V.B., Janik, V.M.: Automated categorization of bioacoustic signals: avoiding perceptual pitfalls. The Journal of the Acoustical Society of America 119, 645–653 (2006)
7. Filatova, O.A., Samarra, F.I., Deecke, V.B., Ford, J.K., Miller, P.J., Yurk, H.: Cultural evolution of killer whale calls: background, mechanisms and consequences. Behaviour 152, 2001–2038 (2015)
8. Ford, J., Ellis, G., Balcomb, K.: Killer whales: the natural history and genealogy of Orcinus orca in British Columbia and Washington. UBC Press (2000)
9. Ford, J.K.B.: A catalogue of underwater calls produced by killer whales (Orcinus orca) in British Columbia. Canadian Data Report of Fisheries and Aquatic Science (633), 165 (1987)
10. Ford, J.K.B.: Acoustic behaviour of resident killer whales (Orcinus orca) off Vancouver Island, British Columbia. Canadian Journal of Zoology 67, 727–745 (January 1989)
11. Ford, J.K.B.: Vocal traditions among resident killer whales (Orcinus orca) in coastal waters of British Columbia. Canadian Journal of Zoology 69, 1454–1483 (June 1991)

12. Garland, E., Castellote, M., Berchok, C.: Beluga whale (Delphinapterus leucas) vocalizations and call classification from the eastern Beaufort sea population. The Journal of the Acoustical Society of America 137, 3054–3067 (June 2015)
13. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016)
15. Ivkovich, T., Filatova, O., Burdin, A., Sato, H., Hoyt, E.: The social organization of resident-type killer whales (Orcinus orca) in Avacha Gulf, Northwest Pacific, as revealed through association patterns and acoustic similarity. Mammalian Biology 75, 198210 (May 2010)
16. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE. vol. 86, pp. 2278–2324 (November 1998)
17. Mercado, E., Kuh, A.: Classification of humpback whale vocalizations using a self-organizing neural network. In: IEEE International Conference on Neural Networks - Conference Proceedings. pp. 1584–1589 (June 1998)
18. Miller, P., Bain, D.: Within-pod variation in the sound production of a pod of killer whales, Orcinus orca. Animal Behavior 60, 617–628 (2000)
19. Ness, S.: The Orchive: A system for semi-automatic annotation and analysis of a large collection of bioacoustic recordings. Ph.D. thesis (2013)
20. ORCALAB: A whale research station on Hanson Island. http://orcalab.org, last checked May, 2019
21. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NIPS 2017 Workshop (October 2017)
22. Schröter, H., Nöth, E., Maier, A., Cheng, R., Barth, V., Bergler, C.: Segmentation, classification, and visualization of orca calls using deep learning. In: International Conference on Acoustics, Speech, and Signal Processing, Proceedings (ICASSP) (May 2019)