Recurrent Attentive Kernel Learning for Shark Activity Recognition

Matthew Buchholz
California State University, Long Beach
Long Beach, CA, USA
matthew.buchholz@student.csulb.edu

Wenlu Zhang
California State University, Long Beach
Long Beach, CA, USA
wenlu.zhang@csulb.edu

Emily N. Meese
Texas A&M University
Galveston, TX, USA
emily.n.meese@gmail.com

Yu Yang
California State University, Long Beach
Long Beach, CA, USA
yu.yang@csulb.edu

Christopher G. Lowe
California State University, Long Beach
Long Beach, CA, USA
chris.lowe@csulb.edu

Hen-Geul Yeh
California State University, Long Beach
Long Beach, CA, USA
henry.yeh@csulb.edu

ABSTRACT

To understand the ecology and environmental impact of free-ranging sharks, it is essential to investigate patterns and drivers of their behavior. Recently, the powerful representations learned by Deep Learning models have been utilized within a Bayesian non-parametric framework in order to learn non-Euclidean similarity metrics for classification. In addition, attention mechanisms have proved state-of-the-art in modeling long-range dependencies between input features. In this work, we propose Recurrent Attentive Deep Kernel Learning for fine-scale shark activity recognition. We show that this flexible and general model can accurately model the relationship between a shark's acceleration dynamics and the behavior at multiple scales. We perform elaborate experiments to demonstrate the effectiveness of the proposed methods and give a discussion on interpretability of the models. [1].

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Gaussian processes**; • **Mathematics of computing** → **Time series analysis**.

KEYWORDS

Time Series Classification, Activity Recognition, Deep Kernel Learning, Attention Mechanisms, Recurrent Neural Network

ACM Reference Format:
Matthew Buchholz, Wenlu Zhang, Emily N. Meese, Yu Yang, Christopher G. Lowe, and Hen-Geul Yeh. 2021. Recurrent Attentive Kernel Learning for Shark Activity Recognition. In *MileTS '21: 7th KDD Workshop on Mining and Learning from Time Series, August 14th, 2021, Singapore.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/1122445.1122456

[1]https://github.com/csulb-ml/SharkBehaviorClassification

MileTS '21, August 14th, 2021, Singapore

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

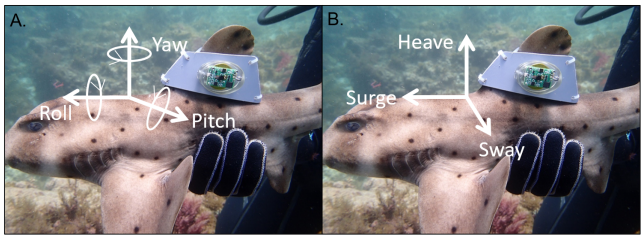


Figure 1: California horn shark tagged with a Cefas G6+ acceleration data logger demonstrating the differences between A) static acceleration (yaw, pitch, roll) and B) dynamic acceleration (heave, surge, and sway).

1 INTRODUCTION

Understanding and predicting patterns in the behavior of free-ranging animals is an essential and challenging task for animal conservation[20]. Due to the heterogeneous conditions of marine biological environments, marine predators modify their movement patterns in order to respond to environmental conditions and prey availability [22]. Studying the activity patterns of marine predators is essential towards gaining an understanding of their ecology within a given habitat [25] and successfully modelling their behavior and movement patterns would yield valuable insight into their ecological impact. California horn sharks (*Heterodontus francisci*), which play an important ecological role in regulating invertebrate communities of squid, crabs, urchins, and small fish [21] [23] [32], have been used successfully as a model species to represent non-obligate ram ventilating elasmobranchs, both in a laboratory and field setting [22]. While the fine-scale spatial and temporal movements and daytime sheltering behaviors of horn sharks have been recently explored, a greater understanding of their activity patterns and feeding behaviors would lead to an improved understanding of their overall ecology. This lack of understanding is primarily due to the difficulty of measuring such observations of the species in their natural environments over large temporal scales due to poor

visibility (for example, low lighting or cloudy water), deeper depths, and adverse environmental conditions [6].

Recent advances in biological sensor tagging have proven useful in monitoring fine-scale animal behavior [33]. In particular, Acceleration Data Loggers (ADLs) provide acceleration data that can be used to classify shark behavior [6]. Augmenting spatial movement data with ADL data can enable researchers to geo-reference these finer-scale activities and give an environmental context to movement patterns. This results in an explicit spatial-temporal representation of the horn sharks' energetic costs which can then be used to identify finer spatial and temporal scale behaviors.

ADLs have the ability to measure both static and dynamic acceleration for each spatial dimension. Static acceleration corresponds to 3D body position (x, y, z) and orientation (pitch, roll, and yaw). Dynamic acceleration represents 3D body movements (x, y, z; surge, heave, sway) as shown in Figure 1. From these, overall dynamic body acceleration (ODBA), which has been effectively used as a proxy for energy expenditure [9] [21] [22], can be calculated by taking the ℓ_1-norm of the dynamic acceleration vector (x, y, z):

$$ODBA = \sum_{i \in x,y,z} |d_i| = \sum_{i \in x,y,z} |r_i - s_i| \tag{1}$$

where d_i is the dynamic acceleration, r_i is the raw acceleration, and s_i is the static acceleration in the direction i.

In this paper, we investigate the use of deep kernel learning and attention mechanisms for modeling the univariate ODBA and multivariate static and dynamic acceleration time-series data. The models recognize the input as one of four categories of shark behaviors: Resting, Swimming, Feeding, and Non-Directed Motion (NDM). In our experiments, we obtain predictions for the behavior of horn sharks for sequences of 2 seconds sampled at 25 Hz. To summarize, our contributions are as follows:

(1) We propose a deep structured probabilistic sequence modelling approach with fine temporal resolution to the shark activity recognition task.
(2) We design a deep recurrent kernel learning model with self-attention for multivariate time-series classification that utilizes the powerful feature representation capabilities of deep learning for sequence learning with the non-parametric probabilistic representations of Gaussian processes.
(3) We conduct experiments on a real-world shark acceleration data set, which demonstrate the effectiveness of our models.
(4) We analyze the interpretablity of our architectural designs in the context of shark energy expenditure and ecology.

2 RELATED WORK

In the past decade, Deep Learning has revolutionized the fields of Pattern Recognition and Data Mining. This is in part due to the technological advancement of graphical processing units (GPUs), but also due to the development of heuristic techniques for training deep networks [11] [13] [17] [19]. For sequence modelling tasks, recurrent neural networks (RNNs) have been the de facto architecture. The task of shark activity recognition shares similarities with sequence modelling, with the key distinction being the explicit dependence on time. In both cases, the input to the models is sequence data which typically contain temporal information, such as semantics in the case of natural language or energy expenditure in the case of shark behavior. However, in language tasks the sequences have discrete indices while in time-series data, the sequences are indexed by continuous time values, sampled at uniform instances. Simply put, sequential data is a subset of time-series data, while the converse is not necessarily true.

While Deep Learning has permeated pattern recognition tasks in computer vision and NLP, there has been a gap in the presence of empirical studies done on the application of deep neural networks for time series classification [14]. Time series classification is a challenging task in data mining, but the field lacks large, generic data sets akin to ImageNet [28]. In addition, deep learning for time series classification lacks in-depth studies of data augmentation [7]. For this reason, we used re-sampling of varying sized windows to augment the data and handle class imbalance issue. The re-sampling method is discussed further in Section 4.1.

Traditional time series classification algorithms are either distance based methods using k-Nearest Neighbors (kNN) built on a metric between the time series, or feature based methods that extract features in the time or frequency domain which are then fed to classical machine learning classifier such as Support Vector Machines, Decision Trees, etc. Naturally, the choice of metric between time series datum is a fundamental problem in time series classification. This is a primary driving factor in our proposal to use deep kernel learning, which can learn non-Euclidean similarity metrics directly from the data [34]. The previous state-of-the-art methods consist of the Dynamic Time Warping (DTW) distance used with a kNN classifier [3] and later the Collective of Transformation-Based Ensembles (COTE) algorithm [4]. The COTE algorithm takes data transformed into multiple domains via ensembles of classifiers. The common theme is to transform the time-series data into some feature space where discriminatory features are easily separable. Deep kernel learning utilizes the representation learning capabilities of neural networks in conjunction with learning the structure of the similarity metric from data. This once again highlights the relevance of deep kernel learning in time series classification.

3 METHODS

3.1 Attention

In recent years, the concept of attention has revolutionized the tasks of machine translation and sequence modeling [5] [31]. Attention is a generalization of pooling which weights feature alignments using a kernel. Attention was proposed to combat the inability of RNNs to model long-range dependencies in an input sequence. We present three differing, yet complementary perspectives on self-attention mechanisms. From a representation learning point of view, self-attention mechanisms explicitly relate different temporal positions in an input sequence in order to form a more comprehensive feature representation. Alternatively, the attention function is a mapping from a vector-valued query and set of key-value pairs to a vector-valued output. Finally, from a Reproducing Kernel Hilbert Space (RKHS) point of view, the attention mechanism is the integral operator $T_K : L^2(X) \rightarrow L^2(X)$ induced by a covariance function

(kernel) $K : X \times X \rightarrow \mathbb{R}$:

$$[T_K f](x) = \int_X K(x, y) f(y) dy \tag{2}$$

We utilize the additive attention mechanism as described in [5]. In this attention mechanism, a context vector c is computed as a weighted sum of the outputs states of a bidirectional GRU:

$$c = [T_K \Phi](x_1, ..., x_T) = \sum_{j=1}^{T} K([h_T; y_T], [h_T; y_j]) y_j \tag{3}$$

with the kernel

$$K(q, k) = e^{V tanh(q^T W_q^T W_k k)} \tag{4}$$

In equation (3), Φ is the functional representation of the RNN, $x_1, ..., x_T$ is the input sequence, h_T is the final hidden state (where forward and backward hidden states are concatenated), y_j is the output of the RNN at time-step j, and $[\cdot\,;\cdot]$ is the concatenation operation. In the kernel function (4), W_q, W_k, and V are learned weight matrices corresponding to the query, key, and attention scores, respectively.

3.2 Deep Kernel Learning

We now continue with our discussion of kernels, but diverge from the realm of attention mechanisms. Kernel methods are a way of generalizing linear models to feature spaces with Euclidean geometry. These feature spaces can be very high, or even infinite-dimensional. The kernel then serves as a vehicle for computing dot products in feature space:

$$K(x, x') = \langle \phi(x), \phi(x') \rangle \tag{5}$$

In this way, the kernel can be seen, informally, as a similarity metric between data representations [29]. Kernel methods are identified by two components: the first being a task-specific loss function which determines the optimization problem used to learn from data and the second being the central object of this section, the choice of kernel. Jaakkola et al. [15] showed how changing the optimization procedure transforms the model from the support vector machine to the generalized linear model. The kernel enforces a structural prior on the space of functions that a learning algorithm can model. In the context of Gaussian processes (GPs), by the Moore–Aronszajn theorem [2], every symmetric, positive definite kernel is associated with a unique reproducing kernel Hilbert space (RKHS). A Gaussian process defines a distribution over the RKHS of functions defined by a kernel. One important structural prior enforced by GPs (via common kernels) is that of local constancy [10] [24]. In terms of the RKHS associated with these kernels, we are considering a space of locally smooth functions:

$$f(x) \approx f(x + \epsilon) \tag{6}$$

GPs place the structural prior that close-by data points are highly correlated [15] [29]. This prior is made explicit via a kernel (or covariance) matrix of a Gaussian distribution. This is the natural extension of the kernel trick, mentioned above, to the Bayesian framework. In the case of time-series modeling, this structural prior is a hindrance, considering that global structure, captured by long-range dependencies, can be of utmost important in complicated tasks like classification. One would like to "bake in" another structural prior, that of recurrent structure. This is the implicit bias present in recurrent neural networks.

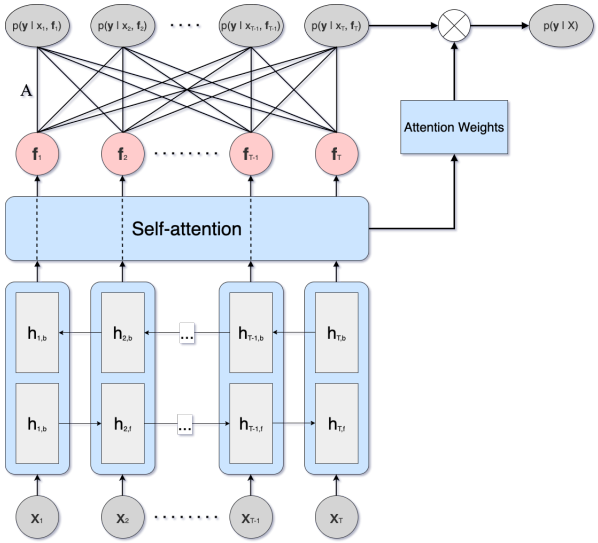


Figure 2: Probabilistic graphical model for Recurrent Attentive Deep Kernel Learning. Blue elements indicate deterministic paths, while red indicates stochastic paths where we place in bold the additive GPs, $\mathbf{f}_i$, to emphasize that they are vector-valued. Observed quantities are marked by shading. Dashed lines indicate that the input passes through directly as the output (the attention weights are only used to weight the class probabilities). $\otimes$ denotes batch matrix multiplication. A denotes the linear mixing matrix.

We now briefly sidestep our discussion of injecting the inductive bias from deep recurrent models to GPs in order to discuss a powerful end-to-end method for combining deep learning with Gaussian processes. In [34], the authors proposed an end-to-end method for learning the structure of kernels which incorporate the inductive biases of deep learning architectures under a scalable GP framework. A generic kernel with hyperparameters θ, can be adapted as:

$$k(g(x_i|w), g(x_j|w)|\theta, w) \tag{7}$$

where $g(x|w)$ is a deep neural network, parametrized by weights w. This deep kernel is then used as the covariance function for a GP, which can be viewed as a hidden layer with infinite width [18]. In addition, it is well known that Gaussian Processes can be extended to classification problems via the softmax likelihood [27]. In [35], this method is extended to classification with training via stochastic gradient descent (SGD). Here, J Gaussian processes are applied to subsets of the Q features learned by a deep neural network. These J Gaussian processes are then linearly mixed to produce multiple, correlated outputs (class probabilities in the multi-class classification setting):

$$p(y_i|f_i, A) = \frac{\exp(f_i^T A^T y_i)}{\sum_{c \in C} \exp(f_i^T A^T e_c)} \tag{8}$$

Here, each $f_i \in \mathbb{R}^J$ is a vector of Gaussian processes, $y_i \in \{0,1\}^C$ is the one-hot encoding of sample i's class, $A \in \mathbb{R}^{C \times J}$ is the learned mixing matrix, and e_c is the indicator vector for class c. The parameters of the deep network and hyperparameters of the additive GP layer are jointly trained by maximum likelihood (ML) of the marginal log-likelihood:

$$\log p(y) \gtrsim \mathbb{E}_{p(f|u)q(u)}[\log p(y|f)] - \beta KL[q(u)||p(u)] \tag{9}$$

where $q(u)$ is the variational distribution over the inducing function values and $p(u)$ is the prior over inducing function values. One can take the expectation over the data-generating distribution $p_{data}(x, y)$ to obtain the variational evidence lower bound (ELBO) loss function [12]. Here the Kullback-Leibler divergence (KL) is a regularization term pulling $q(u)$ closer to $p(u)$. We describe the proposed Recurrent Attentive Deep Kernel Learning algorithm in the following details. Figure 2 is showing a complete illustration of our proposed model.

- We implement Recurrent Neural Network in order to benefit from the inductive bias of recurrent structure in the learned kernel [1].
- We apply an additive GP to every output time-step of the RNN, in order to get a sequence of class-probabilities. These are independent GPs are trained jointly. From equation 8, we can view the mixing matrix A as a form of attention between GPs to model the correlations between different time-steps.
- We train the deep network and additive GP layers from scratch, as opposed to pre-training the deep network weights as in [35].
- We utilize the parametric predictive log-likelihood of [16] to enhance training of our deep GP layers.
- We use a self-attention mechanism to jointly enhance the learned representations used by the deep additive GP layers by learning long-range correlations in the data. In addition, the attention weights are used to weight the class-probabilities output by each additive GP per time-step.

4 EXPERIMENTS

4.1 Experimental Setup

The data sets have been collected from four California horn sharks in the Santa Catalina Island using two acceleration data loggers: the Cefas G6a+ and the Technosmart AxyDepth. The sharks were tagged 6 hrs prior to the trials in order to allow the shark to acclimate; subsequently the sharks were monitored to collect the ground-truth labels. Table 1 displays the number of California horn shark data per category amongst the seven laboratory trials. Class imbalance is present in all seven trials. To combat this imbalance, re-sampling of the time-series data has been performed. To be explicit, over represented classes are downsampled whereas under represented classes are upsampled. For re-sampling, each class is treated as a continuous time-series observation, and training/validation samples are taken by uniformly sampling 2 second (50 samples at 25Hz). For the training set we sample 2,000 sequences per class for a total of 8,000 training sequences, and for the validation set we sample 500 sequences per class for a total of 2,000 validation sequences. As we perform no re-sampling on the test set, we include the F1 score as a metric to account for class imbalance.

4.2 Experimental Results and Analysis

In our experiments, we have trained eight different deep networks in total, four models for each data set. In specific, we train a vanilla GRU, a GRU with additive attention (Attention GRU), a recurrent deep kernel learning model with a GRU backbone (RKL), and the recurrent attentive deep kernel learning architecture (RADKL). The results of all eight models can be seen in Tables 2 and 3(a-g)

The recurrent kernel learning and RADKL models are trained using SGD with Nesterov momentum [30] with a momentum decay factor of 0.9. The backbone GRU was trained with a learning rate of 0.1, and 1e-3 for the additive Gaussian process layer. The learning rate is decayed be 75% of its previous value every 50 epochs. We use L2 regularization with a weight of 1e-4 for the GRU weights. The additive GP layer is trained with no L2 regularization, but it should be noted that optimization via maximizing equation 9 contains implicit regularization via the KL-divergence term.

The machine used for training has the following configurations: NVIDIA Tesla V100-SXM2-16GB in Google Colab Pro. We implement all the deep learning models using the PyTorch and GPyTorch [8] frameworks. The RNNs takes 8 hours in the offline training procedure and the deep recurrent kernel learning models take approximately 24 hours in the offline training procedure. For all models, inference takes around 1 minute to process 4000 samples.

The classification performance of our models is measured by accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic curve (AUC ROC). Considering the amount of class imbalance, we use F1-score as the primary evaluation metric for the performance of our models. Likewise, the classification metrics (precision, recall, and F1) are all class-weighted. The classification results are shown in Table 2 and the confusion matrices of each model are shown in Tables 3(a-g). On the static and dynamic acceleration data, the proposed Recurrent Attentive Deep Kernel Learning (RADKL) model outperforms GRU and Attention GRU. In addition, our experiments show that the proposed attention mechanisms yield an increase in performance over the recurrent kernel learning (RKL) model without attention.

To give a holistic and general evaluation of the proposed model, we give a comparison of the two sub-tables in Tables 2. In specific scenarios, we see significant degradation of all models, including the baselines. To start, comparing the RKL and RADKL models in Tables 2a) and 2b), we see a drop in performance for the RKL model when only using one feature (ODBA). We attribute this to a lack of features from which the Gaussian processes (GPs) can learn rich statistical representations. By contrast, the RADKL model with additive GPs at every time-step, can accurately capture statistical information throughout the entire sequence.

It is important to note the increased intrepretability of the proposed model over standard neural network approaches. In the current framework, the neural network can be seen as learning a neural representation of the data, which is used to form the interpretable kernel hyperparameters of the Gaussian Processes. The explicit use

of Gaussian Processes effectively makes the network's hidden layer infinite-width [18]. In this way, we can benefit from the expressivity of the infinite basis set underlying a GP and the interpretability of GPs as defining an explicit prior over the class of functions our network learns.

By demonstrating the effectiveness of our proposed methods on the difficult and budding task of time-series classification using deep learning, we hope to see more rigorous, theoretical developments marrying deep learning and the theory of stochastic processes. For our task, we note that for modeling the acceleration dynamics of free-ranging sharks, it may be more informative to use dynamic acceleration vs. ODBA, which only yields absolute changes in dynamic acceleration and treats acceleration in each direction independently. Using static and dynamic acceleration yields a more robust model that can more accurately discriminate between similar activity classes (NDM and resting). We hope that this work paves the way for future research into the applicability of probabilistic graphical models and kernel methods in the time-series community.

5 CONCLUSION AND FUTURE WORK

In conclusion, we propose a novel deep network for learning kernels that can model long-range dependencies and recurrent structures by utilizing self-attention mechanisms. We apply our proposed models to a novel data set for shark behavior classification, to demonstrate the effective and complementary nature of deep learning, kernel methods, and probabilistic modeling. For shark activity recognition, future work might want to investigate deeper the effect of non-stationarity in acceleration data on model performance. Using modelling techniques from stochastic processes could help guide future models for this domain. Finally, we intend to extend the work presented here to use attention weights as a weighting over kernels in order to yield a time-series Gaussian process mixture model. In addition, the intrinsic connections between deep kernel learning, metric learning, and self-attention are ones that can shape the future for Deep Learning research.

REFERENCES

[1] Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. 2017. Learning scalable deep kernels with recurrent structure. *The Journal of Machine Learning Research* 18, 1 (2017), 2850–2886.

[2] Nachman Aronszajn. 1950. Theory of reproducing kernels. *Transactions of the American mathematical society* 68, 3 (1950), 337–404.

[3] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* 31, 3 (2017), 606–660. https://doi.org/10.1007/s10618-016-0483-9

[4] A. Bagnall, J. Lines, J. Hills, and A. Bostrom. 2015. Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. *IEEE Transactions on Knowledge and Data Engineering* 27, 9 (2015), 2522–2535. https://doi.org/10.1109/TKDE.2015.2416723

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. (2015). http://arxiv.org/abs/1409.0473

[6] L. R. Brewster, J. J. Dale, T. L. Guttridge, S. H. Gruber, A. C. Hansell, M. Elliott, I. G. Cowx, N. M. Whitney, and A. C. Gleiss. 2018. Development and application of a machine learning algorithm for classification of elasmobranch behaviour from accelerometry data. *Marine Biology* 165, 4 (2018), 62. https://doi.org/10.1007/s00227-018-3318-y

[7] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2018. Data augmentation using synthetic data for time series classification with deep residual networks. *CoRR* abs/1808.02455 (2018). http://arxiv.org/abs/1808.02455

[8] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. 2018. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *arXiv preprint arXiv:1809.11165* (2018).

[9] Adrian C. Gleiss, Rory P. Wilson, and Emily L. C. Shepard. 2011. Making overall dynamic body acceleration work: on the theory of acceleration as a proxy for energy expenditure. *Methods in Ecology and Evolution* 2, 1 (2020/08/23 2011), 23–33. https://doi.org/10.1111/j.2041-210X.2010.00057.x

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[12] James Hensman, Alexander Matthews, and Zoubin Ghahramani. 2015. Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics*. PMLR, 351–360.

[13] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15)*. JMLR.org, 448–456.

[14] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery* 33, 4 (2019), 917–963. https://doi.org/10.1007/s10618-019-00619-1

[15] Tommi S Jaakkola, David Haussler, et al. 1999. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems* (1999), 487–493.

[16] Martin Jankowiak, Geoff Pleiss, and Jacob Gardner. 2020. Parametric Gaussian process regressors. In *International Conference on Machine Learning*. PMLR, 4702–4712.

[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. (2015). http://arxiv.org/abs/1412.6980

[18] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. 2017. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165* (2017).

[19] Min Lin, Qiang Chen, and Shuicheng Yan. 2014. Network In Network. (2014). http://arxiv.org/abs/1312.4400

[20] Zac Yung-Chun Liu, Jerry H. Moxley, Paul Kanive, Adrian C. Gleiss, Thom Maughan, Larry Bird, Oliver J. D. Jewell, Taylor K. Chapple, Tyler Gagne, Connor F. White, and Salvador J. Jorgensen. 2019. Deep learning accurately predicts white shark locomotor activity from depth data. *Animal Biotelemetry* 7, 1 (2019), 14. https://doi.org/10.1186/s40317-019-0175-5

[21] Emily Nicole Meese. 2019. *Diel, Fine-Scale Spatial Movements and Activity Patterns of California Horn Sharks, Heterodontus francisci.* California State University, Long Beach.

[22] Emily N. Meese and Christopher G. Lowe. 2020. Active acoustic telemetry tracking and tri-axial accelerometers reveal fine-scale movement strategies of a non-obligate ram ventilator. *Movement Ecology* 8, 1 (2020), 8. https://doi.org/10.1186/s40462-020-0191-3

[23] Emily N. Meese and Christopher G. Lowe. 2020. Environmental effects on daytime sheltering behaviors of California horn sharks (Heterodontus francisci). *Environmental Biology of Fishes* 103, 6 (2020), 703–717. https://doi.org/10.1007/s10641-020-00977-6

[24] Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. 2006. Universal Kernels. *Journal of Machine Learning Research* 7, 12 (2006).

[25] YP Papastamatiou and CG Lowe. 2012. An analytical and hypothesis-driven approach to elasmobranch movement studies. *Journal of Fish Biology* 80, 5 (2012), 1342–1360.

[26] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, and J Vanderplas. 2011. "Scikit-learn: Machine learning in Python." Journal of machine learning research 12 (Oct), 2825–2830. *URL: http://jmlr.org/papers/v12/pedregosa11a. html* (2011).

[27] Joaquin Quinonero-Candela and Carl Edward Rasmussen. 2005. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research* 6 (2005), 1939–1959.

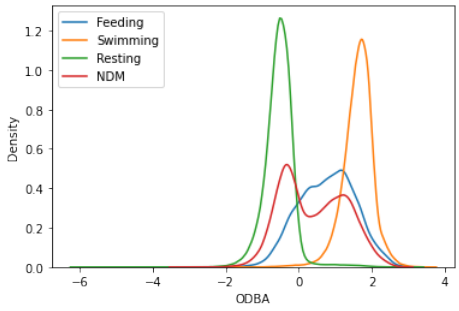
[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. 2014. ImageNet Large Scale Visual Recognition Challenge. *CoRR* abs/1409.0575 (2014). http://arxiv.org/abs/1409.0575

[29] Alex J Smola and Bernhard Schölkopf. 1998. *Learning with kernels*. Vol. 4. Citeseer.

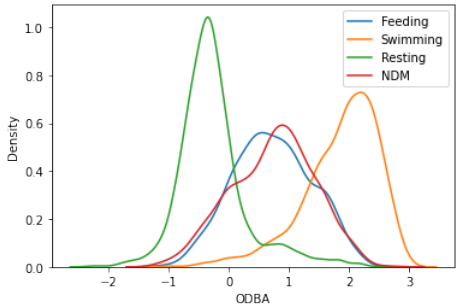
[30] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the Importance of Initialization and Momentum in Deep Learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28 (ICML'13)*. JMLR.org, III–1139–III–1147.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[32] Jeremy J. Vaudo and Michael R. Heithaus. 2009. Spatiotemporal variability in a sandflat elasmobranch fauna in Shark Bay, Australia. *Marine Biology* 156, 12



(a) Training class densities



(b) Testing class densities

Figure 3: Estimated class-wise densities of Overall Dynamic Body Acceleration (ODBA) data in the logarithmic-scale.

(2009), 2579–2590. https://doi.org/10.1007/s00227-009-1282-2

[33] A D M Wilson, M Wikelski, R P Wilson, and S J Cooke. 2015. Utility of biological sensor tags in animal conservation. *Conserv Biol* 29, 4 (Aug 2015), 1065–1075. https://doi.org/10.1111/cobi.12486

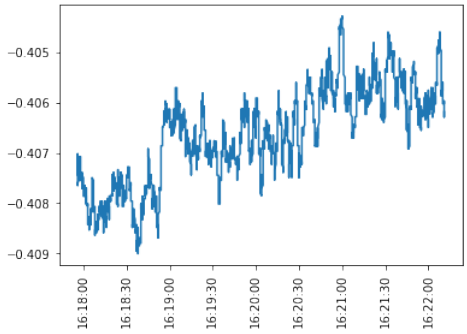
[34] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. 2016. Deep kernel learning. In *Artificial intelligence and statistics*. PMLR, 370–378.

[35] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. 2016. Stochastic variational deep kernel learning. *arXiv preprint arXiv:1611.00336* (2016).

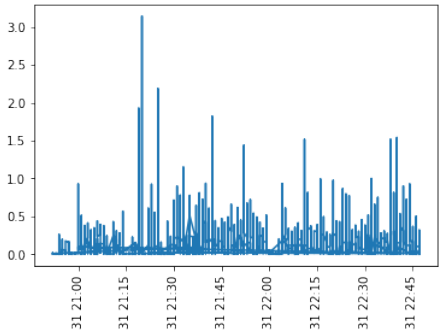
A DATASET

We also observe that each category has different distributions of ODBA data in Fig. 3a). Further, each experiment has differing class-wise densities. We attribute this to similarities in densities of classes in the test set, that are not present in the majority of the training samples in Fig. 3). For instance, we observe that NDM samples are commonly misclassified as feeding in the test set. The densities are estimated using scikit-learn's [26] Kernel Density module, utilizing a Gaussian kernel. To ensure that the class-wise densities are maintained between train, validation, and testing we shuffle and split all the data. Of the 1,741,004 data points, 25% is held out for testing, and then 20% of the remaining training data is held out for validation. In total, 1,044,602 training samples, 261,151 validation samples, and 425,251 testing samples are used.

Two separate data sets are compiled for our experiments: univariate time-series data containing ODBA and the multivariate time-series data with static & dynamic acceleration. The data are treated as sequences of vectors. The two data sets have slightly different pre-processing procedures. The ODBA data is found to be log-distributed in Fig. 3 and is thus analyzed in the logarithmic scale. So the signal (ODBA), which is used as a proxy for energy expenditure, may have a better interpretation in terms of relative, percent-based changes. Finally, both data sets are standardized to have zero mean and unit variance. The mean and standard deviation were calculated from only the training data. In addition, we find out that the ODBA data is stationary in Fig 4b, but the static/dynamic data was slightly non-stationary for certain windows/classes in Fig 4a. Exploring the effect of stationarity on these models may be of interest for future research, but for our experiments the overall effect of the non-stationarity was minimal. However, stationarity of the data for each class may play a role in the performance of our models on a per class basis, and may influence the optimal window sizes for each class.



(a) Static X component with upward trend



(b) Stationarity in Overall Dynamic Body Acceleration (ODBA) data

Figure 4: Examples of stationarity and non-stationarity in static acceleration and ODBA data at differing time-scales.

Table 1: California horn shark data distribution

Class \ Trial	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	Total
Feeding	5,700	350	200	1,375	875	2,900	2,100	13,500
Swimming	6,200	209,550	5,100	81,525	7,975	19,750	61,475	391,575
Resting	157,750	565,580	379,850	10,250	6,150	27,975	77,374	1,224,929
NDM	51,950	7,400	15,700	1,750	1,775	27,025	5,400	111,000
Total	221,600	782,880	400,850	94,900	16,775	77,650	146,349	1,741,004

Table 2: Classification Results - 2 second windows (Sequence length 50)

	Test Acc.	Precision	Recall	F1	AUC ROC
GRU	91.0%	0.9453	0.9103	0.9250	0.93415
Attention GRU	92.4%	**0.9480**	0.9237	0.9342	**0.93585**
RKL	89.5%	0.9129	0.8947	0.9011	0.92827
RADKL	**93.0%**	0.9425	**0.9298**	**0.9355**	0.91905

(a) Results on Overall Dynamic Body Acceleration (ODBA) data

	Test Acc.	Precision	Recall	F1	AUC ROC
GRU	92.1%	0.9504	0.9212	0.9320	0.96632
Attention GRU	93.8%	0.9519	0.9381	0.9436	0.96996
RKL	93.8%	0.9556	0.9378	0.9441	0.97420
RADKL	**94.9%**	**0.9573**	**0.9491**	**0.9523**	**0.97570**

(b) Results on static and dynamic acceleration data

Table 3: Confusion Matrices Results - 2 second windows (Sequence length 50)

(a) GRU - ODBA

T \ P	Feeding	Swimming	Resting	NDM
Feeding	31	3	0	7
Swimming	62	1442	1	30
Resting	6	3	4419	317
NDM	100	17	58	239

(b) GRU - Static and Dynamic

T \ P	Feeding	Swimming	Resting	NDM
Feeding	31	1	0	9
Swimming	22	1463	1	49
Resting	8	17	4396	324
NDM	45	12	43	314

(c) Attention GRU - ODBA

T \ P	Feeding	Swimming	Resting	NDM
Feeding	26	5	0	10
Swimming	70	1426	1	38
Resting	11	5	4519	210
NDM	74	15	75	250

(d) Attention GRU - Static and Dynamic

T \ P	Feeding	Swimming	Resting	NDM
Feeding	34	1	1	5
Swimming	21	1489	1	24
Resting	11	12	4513	209
NDM	44	18	70	282

(e) RKL - ODBA

T \ P	Feeding	Swimming	Resting	NDM
Feeding	20	6	0	15
Swimming	26	1465	1	43
Resting	9	270	4292	174
NDM	57	23	85	249

(f) RKL - Static and Dynamic

T \ P	Feeding	Swimming	Resting	NDM
Feeding	34	0	0	7
Swimming	8	1501	3	23
Resting	14	21	4448	262
NDM	26	14	41	333

(g) RADKL - ODBA

T \ P	Feeding	Swimming	Resting	NDM
Feeding	19	4	1	17
Swimming	36	1448	1	50
Resting	25	10	4565	145
NDM	49	30	105	230

(h) RADKL - Static and Dynamic

T \ P	Feeding	Swimming	Resting	NDM
Feeding	36	1	0	4
Swimming	12	1497	2	24
Resting	26	11	4574	134
NDM	35	14	80	285