Feature Bagging and Extreme Learning Machines: Machine Learning with Severe Memory Constraints

Kallin Khan[1], Edward Ratner[1,2], Robert Ludwig[1], Amaury Lendasse[2,3]

[1]Edammo Inc. Iowa City, USA

[2]Department of ILT, University of Houston Houston, USA

[3]Arcada University of Applied Sciences Helsinki, Finland

***Abstract*—With the onset of easy access to supercomputers with high amounts of memory available, machine learning algorithms have continued to increase the resources necessary to perform their data analysis. This paper aims to show development in the other direction, by showing that through the use of a combination of feature bagging and ensembles of Extreme Learning Machines (ELMs) it is possible to leverage machine learning, without loss of accuracy, on devices where Flash memory is very scarce, and Random-access memory (RAM) is even scarcer, such as on embedded systems. This novel strategy is called Feature Bagged Extreme Learning Machines (FB-ELMs).**

I. Introduction

Classification [1] is a machine learning process by which an algorithm attempts to correctly label samples of data based on their characteristics. The algorithm is trained in a supervised [2] manner, meaning that it's components are tuned based on examples whose classes are known. The process by which we train the base algorithm implemented in this paper, an ensemble of Extreme Learning Machines (ELM, [3]–[9]), is described in the next section.

The arrival of "smart" technology, such as phones, watches, and other new memory and computational limited devices, has created the need to develop algorithms that are not just accurate but also space and Central Processing Unit (CPU) sensitive. While it has become easy to rent a computationally-powerful and seemingly memory-infinite node in the cloud, it is now increasingly difficult to utilize home desktops and, even more difficult, embedded systems to perform machine learning analysis. On embedded systems there are often only hundreds of Kilobytes (Kb) to work with in Flash memory, and a fraction of that for Random-Access Memory (RAM). What if there were a way to modify an algorithm to reduce its memory footprint, without loss of accuracy? In this paper, we will propose a methodology to do just that by utilizing Feature Bagging [10], a method where only a subset of a sample's features are used to train each model or group of models, in combination with an ensemble of Extreme Learning Machines (ELMs). By utilizing this novel Feature Bagged-Extreme Learning Machine (FB-ELM) ensemble, this paper will show that performing machine learning analysis with severe memory constraints is not only possible but also highly accurate.

An advantage of ELMs is that it is relatively easy to export and import a trained ELM, the only values needed being the input and output weights of the network. With that in mind, our analysis will show the memory consumption only of predicting using the FB-ELM ensemble, since training an FB-ELM, exporting it's values, and then importing them onto an embedded system is straight-forward. With the memory constraints associated with training the algorithm taken off of the embedded system, the necessity for carefully selected data in training is eliminated.

Others [11]–[14] have proposed methodologies to address machine learning with memory constraints, but these methods sacrifice speed, need hyper-tuned training data, or must optimize an embedded system's instruction set. Our methodology does none of these things and yet performs with high accuracy despite the severe memory constraints. In the next Section, we will introduce the concept and methodology of Extreme Learning Machines. In Section III, we will discuss the Traditional ELM Ensembling Technique, and then in Section IV how we implement Feature Bagging to improve upon the traditional method. In Section V, the Experimental Data and Settings are presented. The Memory Usage Analysis and the Impact on Accuracy of this novel approach are summarized in Sections VI and VII. Finally, graphical results and conclusions are presented in Sections VIII, IX, and X.

II. Extreme Learning Machine

The Extreme Learning Machine [3]–[9], [14]–[19] is introduced as a generalized Single-Layer Feed-forward Network (SLFN) [7], [20]–[22]. This type of Network is capable of solving classification [23]–[25], regression [26], [27] and clustering problems [28], [29]. According to Huang et al. in [8], ELM has good generalized performance in most cases and the learning speed is thousands of times faster than conventional neural networks [6], [30].

ELM belongs to the family of Randomized Neural Networks (RNNs). Unlike traditional neural networks and learning algorithms, the ELM algorithm shows that hidden nodes can be randomly generated. Thus, the weights from the first layer can be independent from the training data. Because there is no dependence between the input and output weights, ELM has a non-iterative linear ordinary least square solution for the output weights, unlike the conventional Back-propagation training procedure [31]. On top of the distinct properties of ELM, Huang et al. in [20], [22] stated that ELM has the universal approximation capability, indicating that ELM can

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

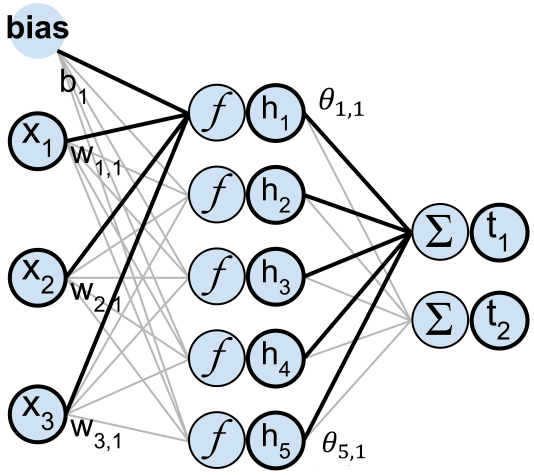

Fig. 1. ELM Structure

universally approximate any continuous target functions in any compact subset X of the Euclidean space $\mathbb{R}^n$ [32].

The rest of this Section gives a brief explanation of the original ELM. In order to keep a uniform meaning for notations throughout the paper, some of the original notations for ELM have been modified.

Figure 1 shows a typical structure of ELM, which contains three layers: the input layer, the hidden layer, and the output layer. Input layer weights ($\boldsymbol{w}$) and biases ($\boldsymbol{b}$) are randomly generated and don't involve in the further training anymore. $\boldsymbol{X} \in \mathbb{R}^{m\times d}, \boldsymbol{X} = (\boldsymbol{x_1}, \ldots, \boldsymbol{x_m})^T$ is the input data, with sample size m, and feature size d. Through the first layer, $\boldsymbol{\theta}$ is mapped to N-dimensional ELM random feature space. After the nonlinear transformation $\boldsymbol{f}$, the hidden layer output is:

$$h_i(\boldsymbol{x}) = f(\boldsymbol{x^T w_i} + b_i),\ i \in [1, N]. \tag{1}$$

f is also called the activation function. Many nonlinear function can be applied here, such as a sigmoid function. Other activation functions are listed in [20], [22]. The last layer is the ELM functional output:

$$f_{ELM}(\boldsymbol{x}) = \sum_{i=1}^{N} \theta_i h_i(\boldsymbol{x}) = \boldsymbol{h(x)^T \theta} = \hat{\boldsymbol{t}}, \tag{2}$$

Where, $\boldsymbol{h(x)} = (h_1(\boldsymbol{x}), \ldots, h_N(\boldsymbol{x}))^T$, $\boldsymbol{\theta}$ is the output weights $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N)^T$ and $\hat{\boldsymbol{t}}$ is the approximation of $\boldsymbol{t}$ — the true target value (i.e. labels, or regression values) of $\boldsymbol{x}$.

The last step for training an ELM is to determine the output layer coefficients: $\boldsymbol{\theta}$. If $\boldsymbol{T} = (\boldsymbol{t_1}, \ldots, \boldsymbol{t_m})^T$ is the corresponding target matrix of the input matrix $\boldsymbol{X}$, $\boldsymbol{\theta}$ should satisfy the following equation:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \left\| f_{ELM}(\boldsymbol{X}) - \boldsymbol{T} \right\|^2, \tag{3}$$

in which, ELM function: $f_{ELM}(\boldsymbol{X}) = \hat{\boldsymbol{T}}$ is an approximation of the true target matrix $\boldsymbol{T}$.

To simplify the problem, introduce $\mathbf{H} \in \mathbb{R}^{m\times N}$:

$$\mathbf{H} = \begin{pmatrix} h_1(\boldsymbol{x_1}) & \ldots & h_N(\boldsymbol{x_1}) \\ \ldots & \ddots & \ldots \\ h_1(\boldsymbol{x_m}) & \ldots & h_N(\boldsymbol{x_m}) \end{pmatrix}, \tag{4}$$

and the minimization problem in equation 3 can be rewritten as:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \left\| \mathbf{H}\boldsymbol{\theta} - \boldsymbol{T} \right\|^2. \tag{5}$$

III. Traditional ELM Ensembling Technique

As described above in Section II there is random initialization of the weights of the input layer of an ELM. This creates the issue of predictive stability for the algorithm as some initializations will inherently perform worse than others. Therefore in the practiced use of ELMs, a traditional ensemble is utilized. This traditional ensemble is composed of numerous ELMs which are each randomly initialized with different input layer weights. The training of a traditional ELM ensemble has two parts, which requires a split of the training set into two subsets: training and validation.

First with the training subset, after using the feature-wise means and standard deviations of the subset and normalizing the data, we compute the weights of the output layer for each ELM in the ensemble by the process described in Section II. Once finished computing these output layer weights, we complete the first part of the training step and move on to validation. In the second step of training, we utilize each individual model to make predictions on the second subset of our training data, the validation set. Since the input weights of an ELM are randomized, the models within the ensemble will have varying levels of performance. By training on a subset of the training data and predicting on the validation set, we evaluate the models that perform best on the data and give them more weight in the final decision. To execute this second step, we first normalize the validation set using the mean and standard deviation of the training set. Then each model in the ensemble predicts on each sample and is assigned a weight. The weight of each model within the traditional ELM ensemble is equal to its accuracy of classifying correctly the samples in the validation set.

After the models are trained and weighted, the traditional ELM ensemble can make predictions on the testing set. Each model classifies each sample, just as in the training set, however when it comes to choosing a final classification for the sample, the ensemble chooses the class that has the highest "weight sum" as so:

$$\boldsymbol{class} = \arg\max(s_1, s_2, \ldots, s_i). \tag{6}$$

Where, i equals the number of possible classes, and $s_1, s_2, \ldots, s_i$ represent the weight sums of classes $1, 2, \ldots, i$. The weight sum is calculated by summing the weights of all the models that predicted a sample as a certain class.

This weighting strategy mitigates the issue of predictive stability of an ELM caused by the model's random initialization. By utilizing numerous ELMs, in some cases several

hundred, then weighting each model based on its performance predicting the validation set with favorably initialized ELMs weighted more heavily, and finally using these weights to make an ensembled final classification as described above, it becomes possible to observe consistent and accurate results despite the random initialization of the ELM models. Not only that, but this weighting strategy has another advantage. Since every ELM is weighted based on its accuracy at classifying the validation set, we can assign a minimum threshold accuracy for giving a model any weight at all. This, in turn, would give ELMs that are poor at predicting a certain data set no say in the ensembled classification while at the same time giving relatively more say to the models whose weight is above the threshold.

IV. Feature Bagged-ELM Ensembling Technique

As stated, feature bagging [33] is a method in which separate models are trained on subsets of the original features. The training of each FB-ELM ensemble is nearly identical to that of the traditional ELM ensembling technique in Section III, which requires a split of our total training set into training and validation sets. However there is an additional pre-processing step before the training of the FB-ELMs. In this step the training data, composed of n features, is split column-wise into m "bags" each composed of n/m features (m must dived evenly into n). A process to select the number of bags to be used in an experiment, and which features belong to each bag, is dependent on the shape and composition of the data set. Each of these bags is composed of a unique set of features. Then for each of the n/m bags, we create a traditional ELM ensemble as described above in Section III. The ELMs in each traditional ensemble are trained on a single bag of features, instead of the entire feature set, rendering the models as FB-ELMs. Once the training step is complete, there is a weight assigned to each FB-ELM just as is done in the training of ELMs in the traditional ELM ensemble.

Once each FB-ELM is trained, we can move onto testing the ensemble. With each sample in the test set, every FB-ELM makes a classification prediction. Then for each of the ensembles, which again only used a single unique bag of features for training, the weight sums are computed. As utilized in Formula 6 and defined shortly after, the weight sum is calculated by summing the weights of all the models that predicted a sample as a certain class. The difference with a FB-ELM ensemble and a traditional ELM ensemble is that the former produces several sets of weight sums, one for each feature bag, and the latter only produces a single set of weight sums. Using Formula 6, an intermediate classification of the sample is produced from every set of weight sums, and the final classification is then the result of ensembling these intermediate classifications by a majority rule selection:

$$\boldsymbol{class} = mode(class_0, class_1, \ldots, class_i). \tag{7}$$

Where, i equals the number of features bags, and $class_0, class_1, \ldots, class_i$ represent the intermediate classifications, computed with Formula 6, for each bag.

V. Experimental data and settings

The data set in this experiment was provided by SwineTech, Inc, a company that specializes in improving animal welfare, productivity, and sustainability within the modern pork industry. SwineTech does this by producing SmartGuard® devices, which are systems of wearable devices and monitors that track sow behaviour relative to its piglets and environment in order to reduce piglet fatalities. The data collection and tracking of sows is performed on 32-bit Microcontrollers, which utilize ARM® Cortex® cores and are manufactured by Silicon Laboratories. The Microcontroller used in this experiment has 512Kb of internal Flash memory and 64Kb of RAM, and the data specifically provided to us for analysis is composed of accelerometer and gyroscopic readings of a sow both in motion and stationary.

The data set is composed of over 34,000 readings of the sow, each with 66 features, and across 28 distinct time series [34]–[36]. Based on these readings we perform a two part classification: first whether the sow is in motion or not, and second, if the sow is not in motion, whether it is sitting or standing. This classification prediction is done using the Feature Bagged-ELM Ensembling Technique described in the previous section.

Combining feature bagging with ELMs gives us the memory usage reduction needed to run machine learning on embedded systems. In order to leverage both feature bagging and all 66 features of our data set, we split our data set into 6 groups: the first containing features one through eleven, the second with features twelve through twenty-two, etc. Therefore there are a total of 6 ensembles of FB-ELMs, that each utilize completely different subsets of the feature space, and in total use the entirety of the 66 features of each sample.

Each of the six ensembles of FB-ELMs contains 900 models, for a total of 5400 models. As described in Section III, numerous models are utilized in order to overcome the predictive instability of a single ELM. After weighting each model based on its performance on the validation set, only the top ten models from each of the six ensembles are chosen to be used for predictions on the test set. Given the memory constraints of the embedded system, reducing the total number of models from 5400 to 60 through the weighting and selecting step is clearly advantageous to achieving the overall goal of space saving. By selecting the FB-ELMs with the best performance on the validation set, the overall accuracy of the ensemble is maintained.

In the test set, the classification decisions of the ten FB-ELMs in each group are ensembled to classify the test sample by the process described at the end of Section IV. The six decisions from each feature group are then ensembled for a final classification of the sample. A successful prediction by our algorithm is characterized as when the majority of the six FB-ELM ensembles classify a sample as in motion or stationary, and if stationary, whether the sow is sitting or standing, when that is the actual class.

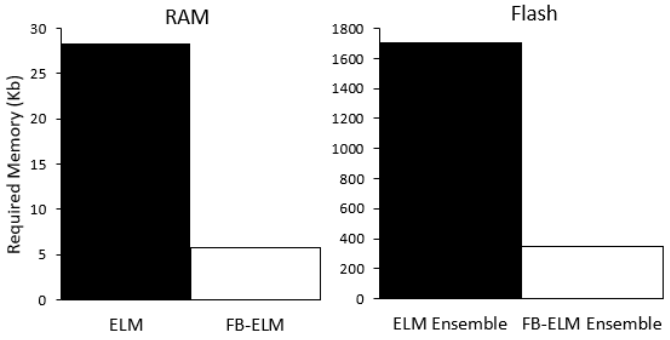

Fig. 2. Memory Usage

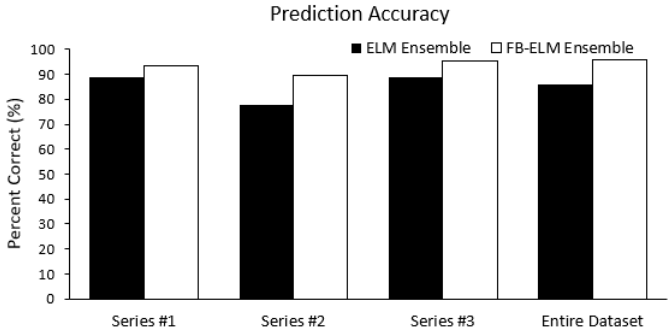

Fig. 3. Accuracy of Ensembles for Motion Prediction

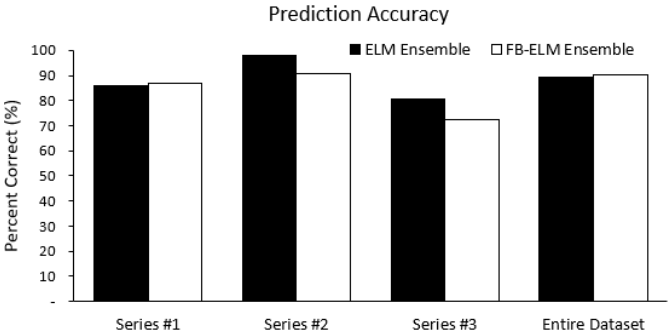

Fig. 4. Accuracy of Ensembles for Sit versus Stand Prediction

VI. Memory Usage Analysis

The two components of memory on an embedded system are Flash and RAM. All the information required to utilize an ensemble of ELMs can be stored in Flash, however to actually do computations with said information it must be loaded into RAM. For this reason, we illustrate the memory consumption of a single model for RAM, as well as the consumption of the entire ensemble for Flash.

The amount of memory required for a single trained ELM to classify a sample is relative to the number of features in the sample space (NF), the number of neurons in the input layer (NN), and the number of possible classes that can be predicted (NC).

Each step of classification with an ELM has its associated memory requirement, and the total memory requirement is the summation of these parts' sizes: the test sample (1 by NF), input weights (NF by NN), output weights (NN by NC), intermediate result of projection of sample with input weights (1 by NN), and final result of projection of intermediate result with output weights (1 by NC). A simplified formula is as follows:

$$(\boldsymbol{NN}+1)(\boldsymbol{NC}+\boldsymbol{NF})+\boldsymbol{NN}. \tag{8}$$

The resulting value of this formula should be multiplied by 32, the number of bits in a floating point number, in order to produce the final memory requirement for an ELM as it is represented in RAM.

In our experiment, the sample space is composed of 66 features, the number of neurons is 102, and the number of possible classes is 2. Using the above memory requirement formula, a single ELM requires 28.4Kb of data with these specifications. In our feature bagged example, with each ELM only utilizing 11 features each, the memory requirement is reduced nearly 5-fold to 5.8Kb.

While only a single ELM needs to be loaded into RAM at a time to make predictions, the entire ensemble must be stored in Flash memory of the device. The two ensembles for this experiment are composed of 60 models each. As expected the FB-ELM ensemble requires far less memory than the traditional ensemble: 348Kb and 1704Kb, respectively.

VII. Impact on Accuracy

In order to evaluate accuracy we implemented a 10-Fold Cross Validation (ie. training the models on 90% of the data and testing on the remaining 10%, repeated 10 times for each unique subset, and the final accuracy computed as the average accuracy of every set [37]) of the 28 times series dataset.

Despite each ELM being provided with less data in order to reduce their memory requirements, the ensemble of FB-ELMs performed better than the traditional approach. When we ran the 10-Fold Cross Validation to predict whether a sample was in motion or stationary, the ensemble of ELMs without feature bagging correctly classified each sample with an average accuracy across validation sets of 85%, while the feature bagging approach actually attained a better accuracy of 96%, as shown in Figure 3. We have included graphs of 3 of the 28 times series' actual classes along side both ensembles predictions in order to illustrate further the FB-ELMs ensemble superior predicting ability.

We then repeated the experiment, but now using the 10-Fold Cross Validation to predict whether a sample was sitting or standing. Even though the ensemble of ELMs without feature bagging sometimes performed better on a single series, it only correctly classified each sample with an average accuracy across validation sets of 89%, where the feature bagging approach again attained a better overall accuracy of 90%, as shown in Figure 4. We have included graphs of 3 of the 28 times series' actual classes along side both ensembles predictions.

VIII. Stationary versus In Motion Series' Examples

The examples below are composed of data from the first three time series in the data set, classified as in motion or as stationary. The top row of graphs are the actual classes of the subject, where a value of 0 means stationary and value of 1 means in motion. The second and third rows show the classifications of each sample by the standard ELM and the FB-ELM ensembles, respectively.

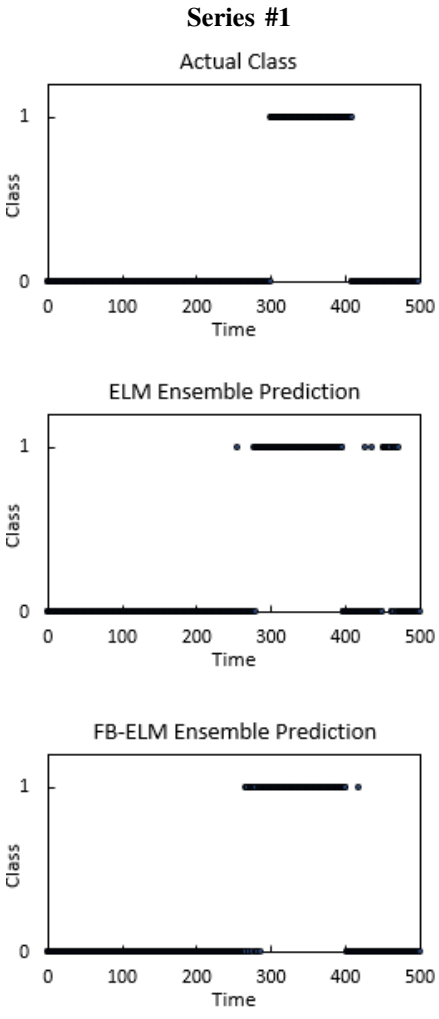

Fig. 5. Series #1 Motion Classes

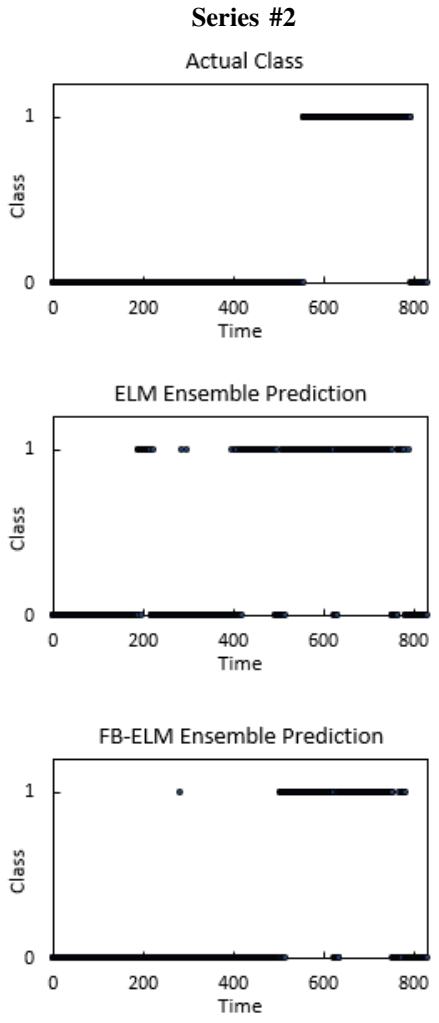

Fig. 6. Series #2 Motion Classes

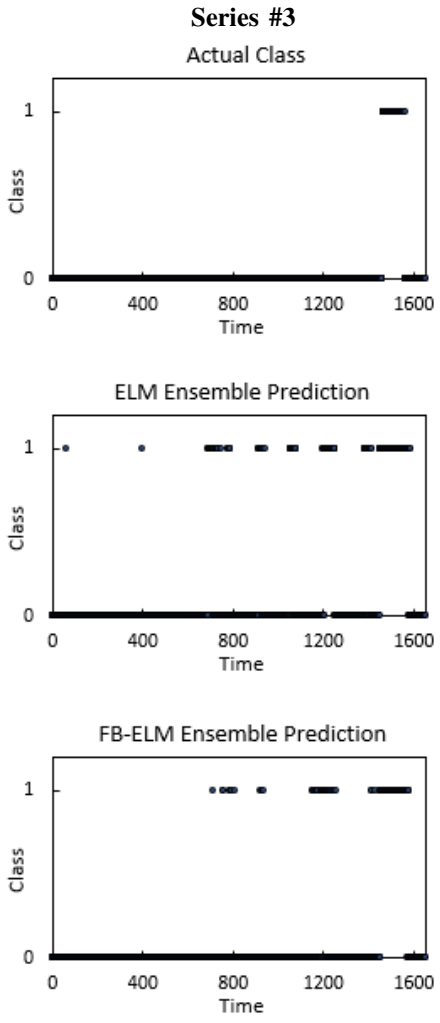

Fig. 7. Series #3 Motion Classes

IX. Standing versus Sitting Series' Examples

Similar to the previous page of examples, the examples below are composed of data from the first three time series in the data set but now classified as sitting or as standing. The top row of graphs are the actual classes of the subject, where a value of 0 means standing and value of 1 means sitting. The second and third rows show the classifications of each sample by the standard ELM and the FB-ELM ensembles, respectively.

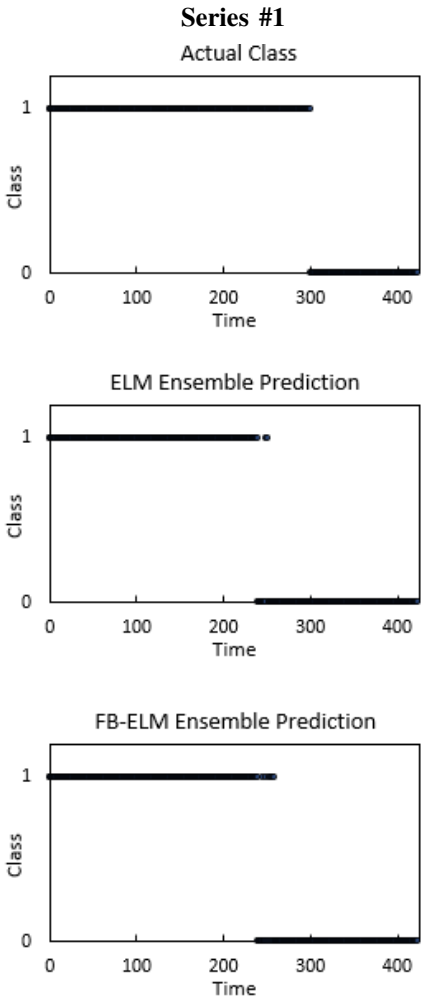

Fig. 8. Series #1 Position Classes

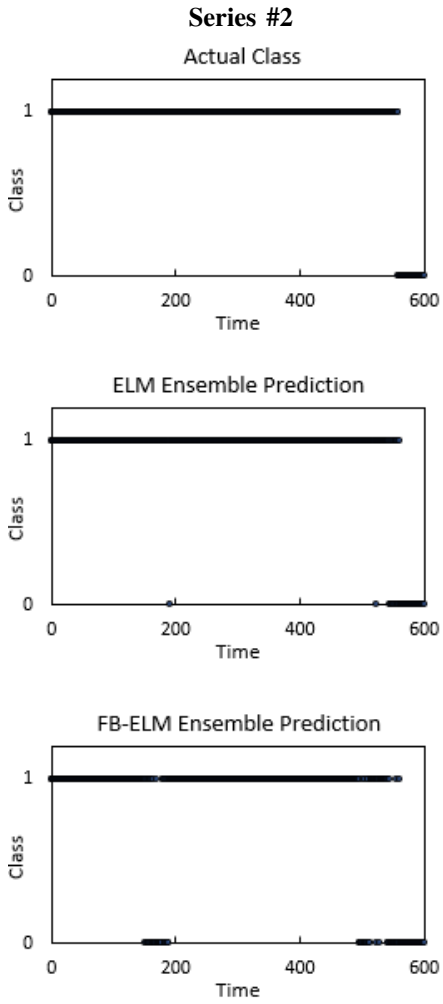

Fig. 9. Series #2 Position Classes

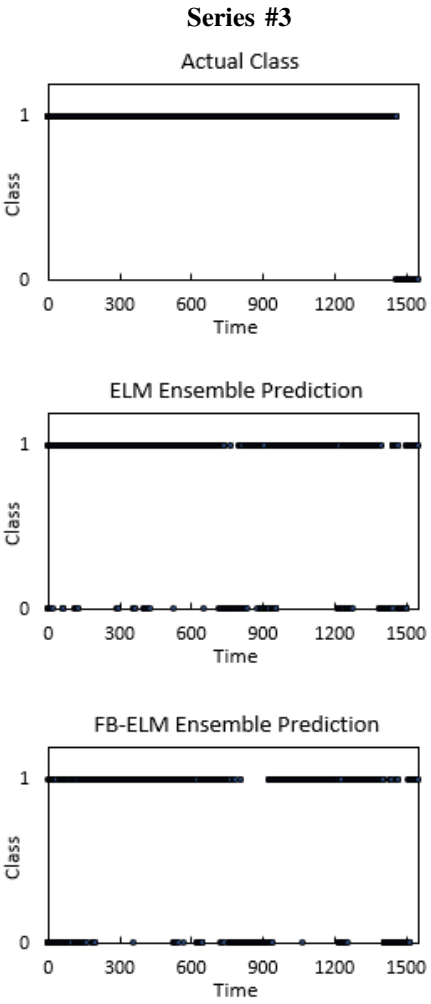

Fig. 10. Series #3 Position Classes

X. Conclusion and Future Work

The results of the experiment not only highlight the space saving advantage of combining feature bagging with an ensemble of ELMs, but also this method's capability to improve the accuracy of prediction. This strategy can reduce the overhead of a machine learning algorithm by nearly 5-fold, and perhaps more in other cases, making high-level data analysis available on devices with severe memory constraints such as embedded systems. Not only does this method account for severe memory constraints, it does so without a loss of accuracy. For both the sow transition and the sow sitting vs standing data sets in this experiment, the accuracy increased by 11% and by 1% with the feature bagged ELM ensemble over the accuracy of the traditional ELM ensemble where each model is fed all the data. In both data sets the accuracy improved with the FB-ELM implementation.

Regarding future work, a clear area for development is the automatic selection of features for each "bag." In this experiment, the number of feature subsets was six with each set containing eleven features. If the number of subsets was eleven with each containing six features, we may have yielded different results. Thus a method for subset size selection would be useful going forward. Another area for future work is to test this methodology on more data sets. It is unclear whether using feature bagging with an ensemble of ELMs will always improve the accuracy of prediction, as it did in this experiment. The focus of this paper was primarily on the space saving properties of this methodology, so we did not experiment with data from other domains. In order to evaluate the algorithm's impact on accuracy more research needs to be done.

REFERENCES

[1] A. Gritsenko, E. Eirola, D. Schupp, E. Ratner, and A. Lendasse, "Solve classification tasks with probabilities. statistically-modeled outputs," in *International Conference on Hybrid Artificial Intelligence Systems*. Springer, Cham, 2017, pp. 293–305.

[2] E. Liitiäinen, A. Lendasse, and F. Corona, "Non-parametric residual variance estimation in supervised learning," in *Computational and Ambient Intelligence*, F. Sandoval, A. Prieto, J. Cabestany, and M. Graña, Eds. Springer Berlin Heidelberg, 2007, pp. 63–71.

[3] D. Sovilj, E. Eirola, Y. Miche, K.-M. Björk, R. Nian, A. Akusok, and A. Lendasse, "Extreme learning machine for missing data using multiple imputations," *Neurocomputing*, vol. 174, pp. 220 – 231, 2016.

[4] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: Optimally pruned extreme learning machine," *IEEE Transactions on Neural Networks*, vol. 21, no. 1, pp. 158–162, Jan 2010.

[5] A. Akusok, K. M. Björk, Y. Miche, and A. Lendasse, "High-performance extreme learning machines: A complete toolbox for big data applications," *IEEE Access*, vol. 3, pp. 1011–1025, 2015.

[6] A. Gritsenko, Z. Sun, S. Baek, Y. Miche, R. Hu, and A. Lendasse, "Deformable surface registration with extreme learning machines," in *International Conference on Extreme Learning Machine*. Springer, 2017, pp. 304–316.

[7] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 2, 2004, pp. 985–990.

[8] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489 – 501, 2006.

[9] E. Cambria, G.-B. Huang, L. Kasun, H. Zhou, C.-M. Vong, J. Lin, J. Yin, Z. Cai, Q. Liu, K. Li *et al.*, "Extreme learning machines [trends controversies]," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 30–59, 2013.

[10] Tin Kam Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, Aug 1998.

[11] O. Good, "ai.googleblog.com/," Jul 2015. [Online]. Available: https://ai.googleblog.com/2015/07/how-google-translate-squeezes-deep.html

[12] K. Ng and R. P. Lippmann, "A comparative study of the practical characteristics of neural network and conventional pattern classifiers," in *Advances in Neural Information Processing Systems 3*, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds. Morgan-Kaufmann, 1991, pp. 970–976.

[13] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, and F. Kawsar, "An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices," in *Proceedings of the 2015 International Workshop on Internet of Things towards Applications*, ser. IoT-App '15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 7 – 12.

[14] A. H. de Souza, F. Corona, G. A. Barreto, Y. Miche, and A. Lendasse, "Minimal learning machine: A novel supervised distance-based approach for regression and classification," *Neurocomputing*, vol. 164, pp. 34 – 44, 2015.

[15] R. Hu, K. Ratner, E. Ratner, Y. Miche, K.-M. Björk, and A. Lendasse, "ELM-SOM+: A continuous mapping for visualization," *Neurocomputing*, vol. 365, pp. 147 – 156, 2019.

[16] F. Boemer, E. Ratner, and A. Lendasse, "Parameter-free image segmentation with SLIC," *Neurocomputing*, vol. 277, pp. 228–236, 2018.

[17] S. Roshan, Y. Miche, A. Akusok, and A. Lendasse, "Adaptive and online network intrusion detection system using clustering and extreme learning machines," *Journal of the Franklin Institute*, vol. 355, no. 4, pp. 1752 – 1779, 2018.

[18] Q. Yu, M. [van Heeswijk], Y. Miche, R. Nian, B. He, E. Séverin, and A. Lendasse, "Ensemble delta test-extreme learning machine (DT-ELM) for regression," *Neurocomputing*, vol. 129, pp. 153 – 158, 2014.

[19] R. Nian, B. He, B. Zheng, M. [van Heeswijk], Q. Yu, Y. Miche, and A. Lendasse, "Extreme learning machine towards dynamic model hypothesis in fish ethology research," *Neurocomputing*, vol. 128, pp. 273 – 284, 2014.

[20] G. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2012.

[21] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[22] G.-B. Huang, "What are extreme learning machines? filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle," *Cognitive Computation*, vol. 7, no. 3, pp. 263–278, Jun 2015.

[23] Guang-Bin Huang, Yan-Qiu Chen, and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 799–801, 2000.

[24] R. Moreno, F. Corona, A. Lendasse, M. Graña, and L. S. Galvão, "Extreme learning machines for soybean classification in remote sensing hyperspectral images," *Neurocomputing*, vol. 128, pp. 207–216, 2014.

[25] Y. Song, S. Zhang, B. He, Q. Sha, Y. Shen, T. Yan, R. Nian, and A. Lendasse, "Gaussian derivative models and ensemble extreme learning machine for texture image classification," *Neurocomputing*, vol. 277, pp. 53–64, 08 2017.

[26] Z. Li, K. Ratner, E. Ratner, K. Khan, K.-M. Bjork, and A. Lendasse, "A novel ELM ensemble for time series prediction," in *Proceedings of ELM 2018*. Springer, Cham, 06 2019, vol. 11, pp. 283–291.

[27] M. van Heeswijk, Y. Miche, E. Oja, and A. Lendasse, "GPU-accelerated and parallelized ELM ensembles for large-scale regression," *Neurocomputing*, vol. 74, pp. 2430 – 2437, 09 2011.

[28] A. Akusok, D. Veganzones, K.-M. Björk, E. Séverin, P. du Jardin, A. Lendasse, and Y. Miche, "ELM clustering – application to bankruptcy prediction," in *International work-conference on Time Series (ITISE)*, 2014, pp. 711–723.

[29] Y. Miche, A. Akusok, D. Veganzones, K.-M. Björk, E. Séverin, P. du Jardin, M. Termenon, and A. Lendasse, "SOM-ELM—self-organized clustering using ELM," *Neurocomputing*, vol. 165, pp. 238 – 254, 03 2015.

[30] R. Hu, V. Roshdibenam, H. J. Johnson, E. Eirola, A. Akusok, Y. Miche, K.-M. Björk, and A. Lendasse, "ELM-SOM: A continuous self-organizing map for visualization," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[31] S. Haykin, *Neural Networks: A comprehensive foundation*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2004.

[32] H. Yu, Y. Yuan, X. Yang, and Y. Dan, "A dynamic generation approach for ensemble of extreme learning machines," in *Advances in Neural Networks – ISNN 2014*, Z. Zeng, Y. Li, and I. King, Eds. Cham: Springer International Publishing, 2014, pp. 294–302.

[33] C. Sutton, M. Sindelar, and A. Mccallum, "Feature bagging: Preventing weight undertraining in structured discriminative learning," University of Massachusetts, Center for Intelligent Information Retrieval, Tech. Rep., 2005.

[34] A. Guillen, L. Herrera, G. Rubio, H. Pomares, A. Lendasse, and I. Rojas, "New method for instance or prototype selection using mutual information in time series prediction," *Neurocomputing*, vol. 73, no. 10, pp. 2030 – 2038, 2010.

[35] A. Lendasse, M. Cottrell, V. Wertz, and M. Verleysen, "Prediction of electric load using Kohonen maps - application to the Polish electricity consumption," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, vol. 5, May 2002, pp. 3684–3689 vol.5.

[36] A. Lendasse, M. Verleysen, E. de Bodt, M. Cottrell, and P. Grégoire, "Forecasting time-series by Kohonen classification," in *ESANN 1998, 6th European Symposium on Artificial Neural Networks*, 1998, pp. 221–226.

[37] A. Lendasse, V. Wertz, and M. Verleysen, "Model selection with cross-validations and bootstraps - application to time series prediction with RBFN models," in *Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003*, O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, Eds. Springer Berlin Heidelberg, 2003, pp. 573–580.