Evaluation of Compressed Residential Energy Forecasting Models

Robert Fagaras, Cristina Nichiforov, Iulia Stamatescu and Grigore Stamatescu
Department of Automation and Industrial Informatics
University "Politehnica" of Bucharest
Bucharest, Romania
robert.fagaras@stud.acs.upb.ro, {cristina.nichiforov, iulia.stamatescu, grigore.stamatescu}@upb.ro

***Abstract*—Embedded energy gateways are increasingly being implemented for monitoring and control tasks in smart energy applications in buildings, cities, and localized electrical grids. These leverage state-of-the-art computational intelligence models for forecasting and anomaly detection. We present a study comparing various neural network model implementations for residential energy forecasting, both full-size models and compressed models using weight quantization and network pruning techniques. The evaluation is carried out on publicly available data consisting of 23 homes from the reference Pecan Street database. We analyse the optimal trade-off between accuracy, in terms of MSE quantitative metrics, and model complexity as reflected by network memory footprint for resource constrained embedded platforms. Using a pre-determined MSE threshold for residential energy forecasting, we achieve higher than 50% memory footprint reduction compared to the baseline scenario, using a combination of weight pruning and quantization, for both individual homes and global models that are trained across time series.**

I. Introduction

Consumer-side energy management assumes robust models of demand which are trained on long-term power measurement time series and contextual information. Several types of models can be trained such as classical system identification, machine learning and deep learning models for short-term load forecasting (STLF) tasks. The main characteristics of these categories [1] are briefly summarised as follows:

- classical system identification: offers good performance while requiring expert knowledge for selecting the best model structure, such as ARMA, ARIMA, ARIMAX; the methods are highly specific to the process/phenomena being modelled;
- machine learning: it allows the incorporating of expert domain knowledge in the feature engineering stage; provides good performance and adequate explainability e.g. decision trees, largest margin classifiers;
- deep learning: offers state-of-the art performance with very large number of training examples but can be opaque in terms of explainability and require resource intensive training; built-in stepwise feature extraction.

This work has been partially supported by the University Politehnica of Bucharest through the project Engineer in Europe, ME no. 140/GP/19.04.2021.

The predictive performance is highly dependant on both the quality and quantity of the training data with diverse datasets over longer periods of observation typically yielding the best results. State-of-the-art neural network structures exploit both long-term temporal and spatial dependencies to extract nonlinearities and perform fine grained discrimination. With regard to building energy consumption, daily, weekly and seasonal energy patterns are learned as well as variations from such patterns. Current challenges in electrical grid management imply ever decreasing time scales for measurement, inference and control in power systems. Edge inference for localized control of energy events [2] requires learning models that can output predictions under hard real-time constraints with low sampling rates. These typically run on resource constrained devices, which are closely coupled to and installed in physical proximity of the monitored equipment or process. The ability to achieve high quality predictions at the local level can be also used in anomaly detection and predictive maintenance for large scale equipment with economic and environmental impact.

Several methodologies have become available that allow the compression of full-scale neural network prediction and classification models to reduce their memory and computing footprint with bounded error metric degradation. The main categories identified and considered in this paper are: quantization - representing the model weight and bias terms on fewer bits in integer or binary formats and pruning - selecting the weights and neurons that contribute least to the output quality and discarding them. Suitable implementations as open source libraries and tools can be leveraged in a energy forecasting framework to select the lowest footprint algorithm that satisfies application performance metrics.

The main contribution of the work consists of a methodology to train, compress and evaluate energy forecasting models using state-of-the-art approaches and open source software instruments. This is accompanied by practical evaluation of the trade-off between model accuracy and model complexity in energy forecasting for dwellings. We compare a baseline fully connected feed forward neural network (FCNN) architecture trained on preprocessed features to various approaches for network compression. The results can be used for embedded deployment of the forecasting algorithms at the local level.

II. Related Work

We first review several timely contributions concerning computational intelligence models for energy forecasting. In [3] an artificial neural network (ANN) is presented for building energy consumption. The network uses the historical electric load alongside weather information, calendar data and holiday information as inputs and several experiments are carried out for a varying number of hidden layers (1-10) and different activation functions of the neurons (ReLU, LReLU, PReLU, ELU, SELU). The performance is evaluated based on the coefficient of variation of RMSE (CV-RMSE) and the mean absolute percentage error (MAPE). The best architecture exhibits a 3.4% MAPE value with six hidden layers.

A deep learning approach for energy forecasting is described in [4] where a convolutional neural network (CNN) is trained for short term load forecasting at the network level. Performance is measured using MAPE. The main result is that, using a standardized dataset, the CNN achieves below 1% prediction error, lower than the transmission system operator (TSO) forecasts for the specific case studies. An efficient method for energy demand forecasting is implemented by [5]. The authors present the usage of three machine learning techniques: extreme gradient boosting, categorical boosting, and random forest methods, with improved preprocessing through missing data imputation for ensemble predictions. The mean absolute error (MAE) and goodness of fit coefficient (R^2) are used for evaluation, while achieving an improvement ($R^2 = 0.92$) over off-the-shelf methods. A hybrid approach for energy forecasting is described in [6]. The core algorithm is based on support vector machines (SVM) with an enhanced scheme for feature selection using on random forests. Testing is carried out on publicly available power system data.

Recurrent neural network sequence models that exploit long-term dependencies in the input data for building energy forecasting are presented in [7] and [8]. These show an improvement over the conventional ANN structures albeit with limited control of the features and difficulty of guiding the forecasts by incorporating domain knowldege. Classification for dominant usage patterns based on computational intelligence feature extraction has been illustrated by [9].

III. Methodology

Modern learning architectures need considerable storage memory and computational power, which is a problem in applications that need to run on embedded systems [10]. Therefore, it is desired to identify the parameters of the model that have a low significance and eliminate them. Four categories of techniques for the compression of neural networks are identified in [11]: weight sharing and pruning, low rank factorization of parameter tensors, design of compact convolutional filters, knowledge distillation.

The common feature of methods based on weight sharing and pruning is the identification of redundant parameters and their elimination. A specific subcategory of these methods is quantization, which is a weight sharing strategy. This limits the number of bits used to represent each parameter, so that multiple groups of parameters are forced to share the same value.

Pruning is a common method used to insert zeros into weight tensors in a neural network. This involves applying a specific criterion to decide which weights will be set to zero. The problem is formalised in [12] as a combinatorial optimization problem to find the optimal pruned weight subset W' that preserves the accuracy of the pruned network with the original set of weights W:

$$\min_{W'} |C(D|W') - C(D|W)| \; s.t. \; ||W'||_0 \leq B \tag{1}$$

where $C(D|W)$ is the cost of the neural network trained on the D examples with weights W and the norm $||W'||_0$ bounds the number of non-zero parameters in B.

The method can also be applied to biases, but they are significantly less than the number of weights and their contribution to the output of a neuron is generally large, so it is not attractive to apply this method on them. Regarding pruning, there are two possibilities for its application: weight pruning and neuron pruning.

In the literature there are many heuristic criteria which are computationally efficient. One of these, is pruning by magnitude of kernel weights i.e. minimum weight. Minimum weight is a pruning criterion which helps eliminate the unnecessary values by ranking their importance based on their magnitude. The higher the value of a weight the more significant its contribution can be in activating the neuron, thus pruning those weights with values close to zero can have little to no impact on the accuracy of the model. In this case of pruning which uses the norm of a set of weights, the criterion is evaluated as follows:

$$\Theta_{MW} : \mathbb{R}^{C_{l-1} \times p \times p} \rightarrow \mathbb{R}, \tag{2}$$

with

$$\Theta_{MW}(w) = \frac{1}{|w|} \sum w_i^2, \tag{3}$$

where $|w|$ is dimensionality of the set of weights after vectorization and $p \times p$ is the size of the C_{l-1} kernels. The relevance of applying this type of pruning is that a convolutional kernel with low $L2-norm$ detects less important features than those with a high norm [12].

Neuron pruning consists of setting entire columns in the weight tensors to zero, therefore eliminating the contribution of a specific neuron. However, this technique can have a higher impact on the accuracy of the network. The reason behind using pruning for model compression is that sparse tensors are more compressible. By applying a simple file compression algorithm on a pruned model the disk size required for storage and transmission is reduced significantly.

An example of neuron pruning versus weight pruning is showcased in Figure 1 which shows how the latter better selectivity keeps the decrease in accuracy limited even for higher sparsity coefficient values.

Quantization is the process of reducing the number of bits used to represent a number. In the context of deep learning,

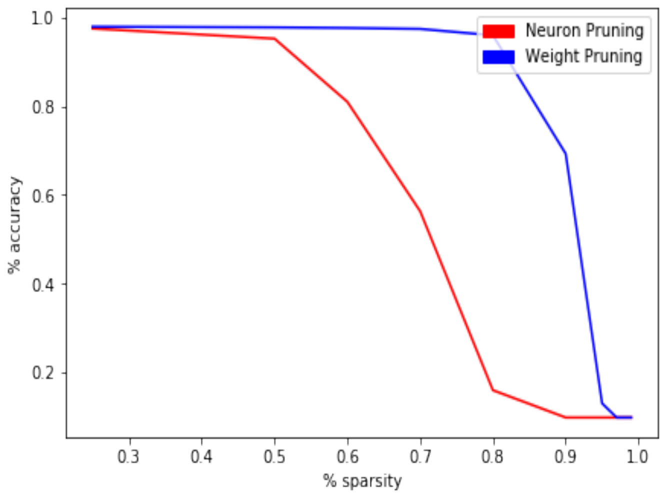

Fig. 1. Neuron pruning versus weight pruning

the predominant numeric format is 32-bit floating point (32FP) [13]. However, the need to reduce the computational power required for deep learning has led researchers to use a numerical format with lower accuracy. Therefore, it has been shown that an 8-bit representation (INT8) can be used to represent the weights and activations of neurons without significant loss of accuracy. Using an even smaller numeric format, 4/2/1 bits, is an active field of research that has also shown good progress [14]. As a theoretical generalisation uniform quantization can be formulated as an affine mapping between the real domain r and the quantized domain s as follows:

$$r = s \cdot (q - z) \tag{4}$$

where s is the scale factor and z is the zero point. For the domain of real values between $[r_{min}, r_{max}]$, the linear mapping to the minimum and maximum integer values, represented by $[0, 2^{b-1}]$ is achieved through:

$$r = \frac{r_{max} - r_{min}}{(2^b - 1) - 0} \cdot (q - z) \tag{5}$$

$$s = \frac{r_{max} - r_{min}}{(2^b - 1) - 0} \tag{6}$$

with b being the b-bit integer representation of the real number (neural network weight in our case).

Recent approaches have shown the efect of learning the quantization thresholds for fine grained optimization improvement [15]. In comparison to the pruning technique, which only affects the model while in a compressed form, therefore useful while transmitting the model, quantization affects the storage memory required for the model as well as the inference time since 8-bit operations are executed instead of 32-bits floating point operations.

We use the Mean Squared Error (MSE) for evaluation the network test set performance degradation with variable sparsity.

$$MSE = \sum_{1}^{n} \frac{(y_i - \hat{y}_i)^2}{n} \tag{7}$$

where y_i is the test set value, $\hat{y}_i$ is the model prediction for that value and n is the sample size. The MSE quantifies the sum of the squared bias and the variance in a statistical learning model.

IV. Results

For the experimental results we carry out the implementation steps illustrated in Figure 2. These are grouped into two main stages: the pre-processing includes both feature extractiom, selection and FCNN baseline model training, while the second stage covers the various model compression techniques analyzed in this work for our particular use case, namely residential energy forecasting. Based on the MSE analysis, the memory footprint and computational requirements, the resulting individual or aggregated models can be deployed on the local monitoring and control device, such as a Raspberry Pi class embedded device.

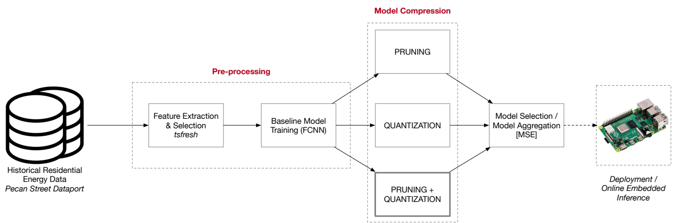
Model Compression
Pre-processing
PRUNING
Historical Residential Energy Data
Pecan Street Dataport
Feature Extraction & Selection
tsfresh
Baseline Model Training (FCNN)
QUANTIZATION
Model Selection / Model Aggregation [MSE]
PRUNING + QUANTIZATION
Deployment / Online Embedded Inference

Fig. 2. Data processing and compressed modelling pipeline for energy forecasting

A. Input data set and preprocessing for feature extraction

We illustrate our approach on publicly available residential energy data from the Pecan Street Dataport energy research platform. We select the California Residential dataset which contains rich energy data from 23 houses, sampled at 15-minute intervals for a full year, thus yielding around 35000 data samples for each home. The variable that we use for the modelling is the grid drawn power at each sampling interval. An example for one home is illustrated in Figure 3.

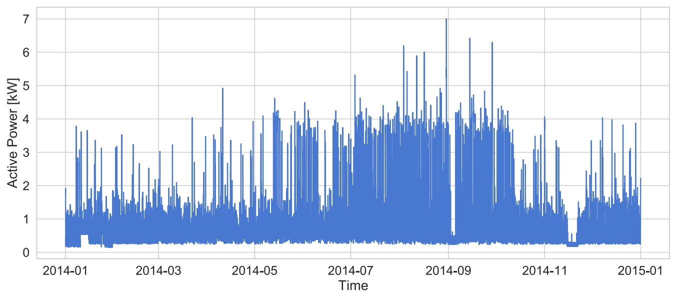

Fig. 3. Sample input data for one residential building

https://www.pecanstreet.org/dataport/

In the preprocessing stage we leverage the tsfresh [16] library for automated feature extraction and selection. These include statistical indicators, Fourier and wavelet coefficients, computed on rolling window intervals from the original time series. The feature extraction and selection step is performed for each of the 23 building energy traces. For our dataset the number of selected figures ranges from 255 to 298. Figure 4 illustrates the distribution of the values across the studied buildings.

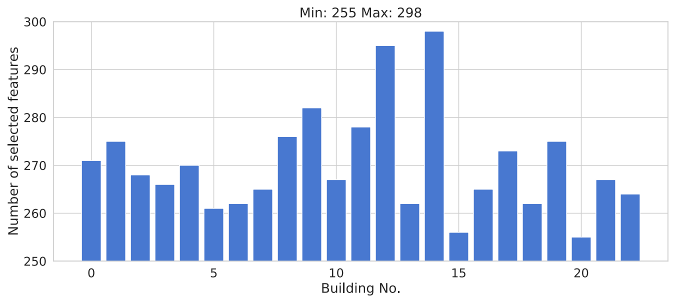

Fig. 4. Number of selected features across the residential buildings in the dataset

We select 164 common features across all the buildings which are fed (input layer size = 164) to a one hidden layer, 50 neurons per layer, fully connected feed-forward neural network (FCNN) with rectifier linear unit (ReLU) activation functions, serving as performance baseline for the compressed models. The features selection is carried out internally through a statistical procedure from a list of 800+ pre-computed features with different parameters. Selected features are ranked by their p value, with a small value indicating the rejection of the null hypothesis H_0 and thus statistical relevance. In our case, some representative feature categories ($p = 0$) used to train the residential energy forecasting models, are as follows: *sum_values*, sum calculation over the time series, *fft_coefficients*, Fourier coefficients of the one-dimensional discrete Fourier Transform as complex numbers with both real and imaginary parts, *change_quantiles*, average and absolute value of consecutive changes of the time series inside a defined corridor, parametrized by the quantiles ql and qh.

The goal is not to obtain the highest performing neural network model but rather to benchmark the compressed models, with the aforementioned compression techniques to a standardized baseline so that we can extrapolate expected improvements for large-scale experimental deployments in the future.

B. Evaluation of compressed models

The aforementioned model compression techniques are applied to a baseline model represented by a neural network which was trained with the data corresponding to a single house. The baseline model has been trained for 100 epochs obtaining a value of mean squared error of 0.167 on the test set. The implementation is performed in the Python programming language using the TensorFlow library [17] for the full neural network training. The compressed representations are achieved using TFLite, which has been shown to provide efficient computation and memory conversion for edge inference [18]. Since weight pruning has proven to have a much lower impact on the model accuracy while obtaining the same sparsity in the weight tensors as the neuron pruning, only the former one has been applied.

The minimum weight criterion, described in Section III, has been used to select the weights to be pruned. Each weight was ranked based on its magnitude represented by the absolute value of the weight. In order to achieve a target percentage of sparsity, all weights below a certain threshold were set to zero. The threshold is obtained as the highest rank times the target sparsity in percentage. Figure 5 presents the degradation of the considered metric with the increase in model sparsity.

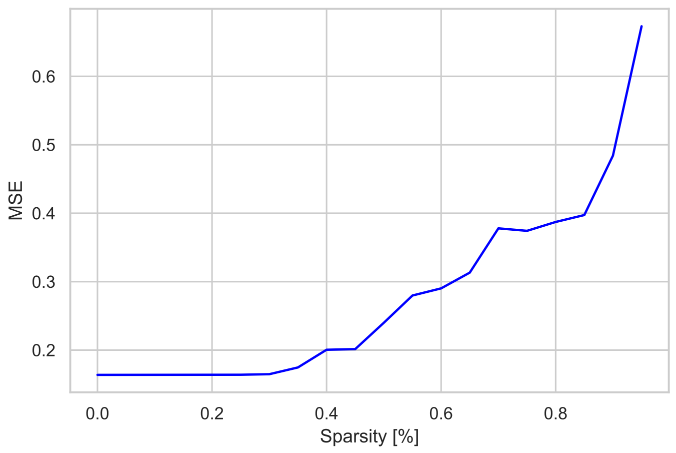

Fig. 5. MSE vs. Sparsity w/o retraining

The pruned model has the same disk size as the baseline model because the effect of pruning is seen when compressing the two models. Therefore a file compression algorithm is used and Figure 6 presents the reduction in disk size of the sparse compressed models compared to the compressed baseline model.

However, the obtained models require significant storage space and a GPU to run inference, which are not available on most of the embedded devices and since the desired outcome is to be able to run inference and store these models on embedded devices, converting the obtained models into TFLite models is necessary. Figure 7 presents the change in storage space in regard to the mean squared error for the converted TFLite models after applying the file compression algorithm.

Based on Figure 7, it can be observed that a model having less than half the memory size of the baseline one can be obtained with insignificant loss of accuracy. Taking into consideration that a mean squared error value of 0.25 would represent an acceptable error for the tackled problem, a model with 50% sparsity, requiring 2x less memory can be used.

The chosen quantization method is the post-training quantization. The weights of the baseline model have been first

https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html

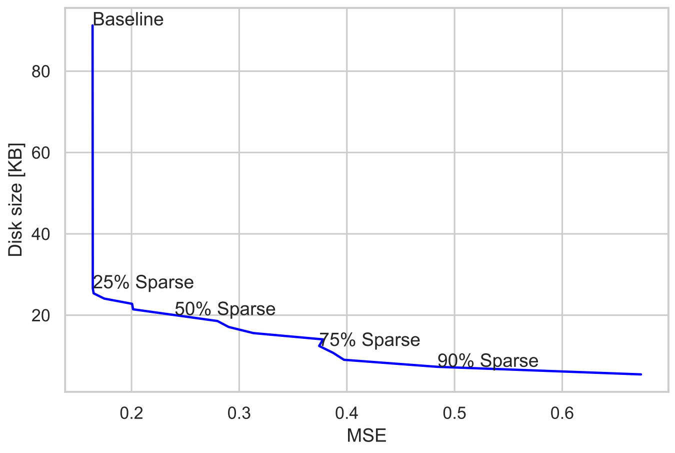

Fig. 6. MSE vs. disk size [KB]

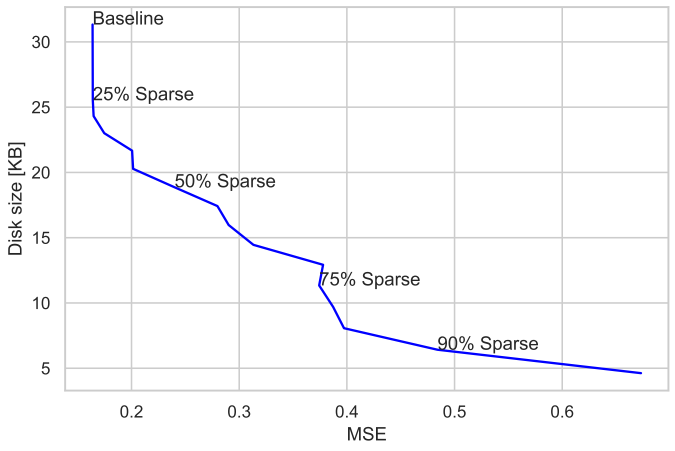

Fig. 7. TFLite pruned models MSE vs. disk size [KB]

quantized to a 16-bit floating point format,obtaining a model which requires 3 times less storage space while preserving the mean squared error of the baseline model. This type of quantization is useful for models that run inference on GPU since GPUs operate with 16-bit format, thus no additional conversion is required.

Moving further, weights have been quantized to 8-bit precision, obtaining a 80% smaller model, yet increasing the mean squared error. Applying the 8-bit quantization to the activations as well has resulted in a model similar in size with the one obtained from quantizing the weights only, but with a greater negative impact on the model's accuracy. The results of the methods mentioned above has been presented in Table I.

Since applying the 8-bit quantization only to weights has proven to significantly reduce the model required storage space while preserving most of the model accuracy, this method was chosen to be further applied to the pruned models, obtaining a model with a satisfying mean squared error while also requiring 6x less space than the original model.

As it can be seen in Figure 8 combining the two compression methods has successfully resulted in a 50% sparse model, requiring 85% less disk size for storage and obtaining a 0.248 mean squared error value on the test set, hence representing a valid candidate for a model to be deployed on an embedded device.

TABLE I
QUANTIZATION RESULTS

Model	Weight Format	Activation Format	MSE	Storage[KB]
Baseline	32	32	0.163	31.346
16-bit Quantized	16	16	0.17	10.36
8-bit Quantized	8	32	0.175	6.582
8-bit Quantized	8	8	0.79	6.303

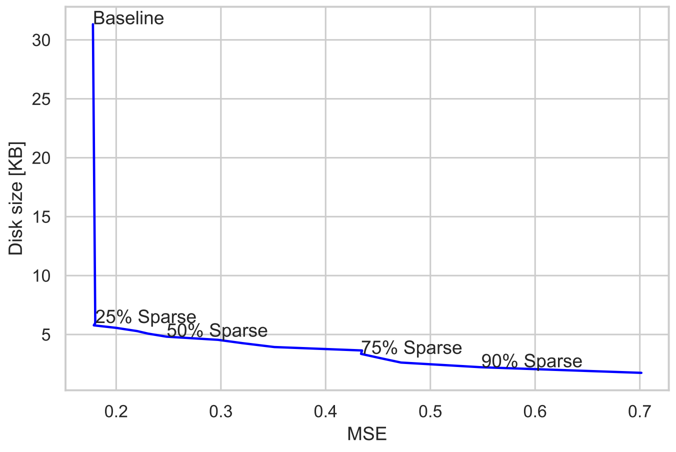

Fig. 8. TFLite pruned and 8-bit quantized model MSE vs. disk size [KB]

Results for the energy prediction using the 8-bit quantized and pruned model with 50% sparsity are show in Figure 9. The results show the prediction for 100 data points comparing the model's forecast against the real measurements.

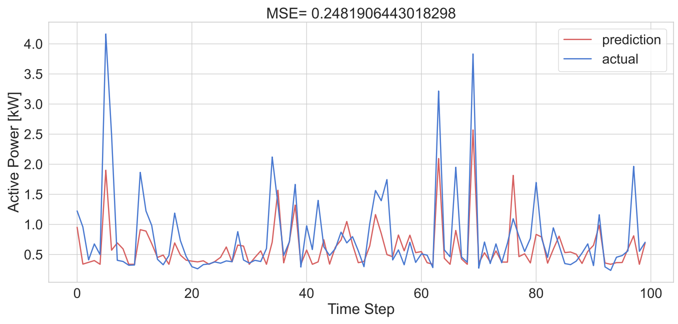

Fig. 9. Model predictions vs. actual values

A global model with the previously mentioned structure has been trained with the entire data set for all the 23 houses. After 100 epochs of training, the model's mean squared error has a value of 0.19. Taking into consideration the results obtained on the individual model, same pruning technique and 8-bit quantization have been applied to the global model. Figure

10 shows the degradation of the global model's accuracy with pruning and quantization.

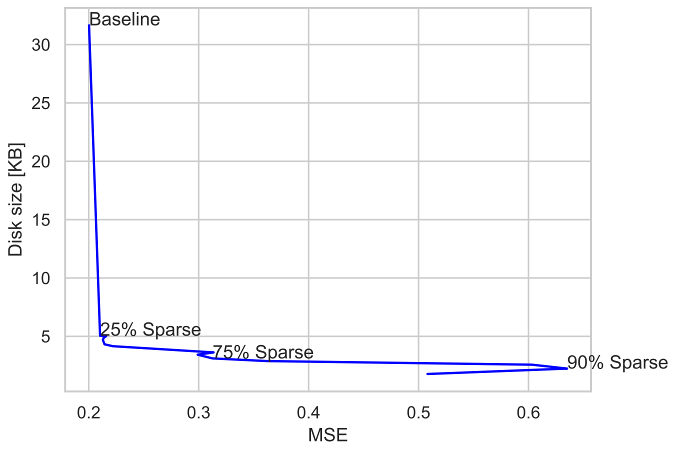

Fig. 10. Global TFLite model MSE vs. disk size[KB]

A 60% sparse global model has been selected and used to predict 100 data points from the energy consumption. The results are presented in Figure 11 and indicate a degradation of the model performance from a MSE of 0.19 to 0.26. However the robustness of the approach over residential clusters as compared to individual models at the home level should be considered. Table II presents the numeric values from Figure 7, Figure 8 and Figure 10. Based on these values we see that pruning the model beyond 50-60% sparsity has a great impact on model's accuracy and would require re-training of the model in order to compensate.

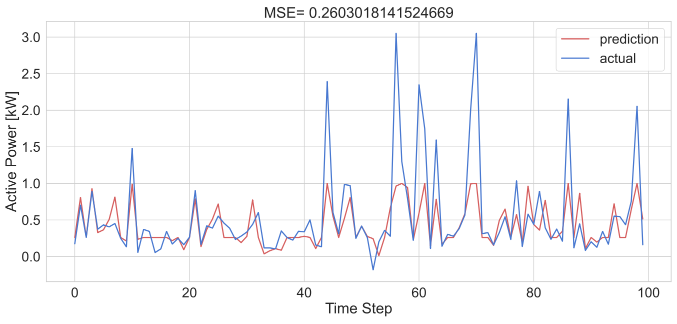

Fig. 11. Global model predictions vs. actual values

TABLE II
PRUNING RESULTS

Model MSE	Sparsity [%] 0	25	40	50	60	75	90
Pruned	0.163	0.164	0.2	0.239	0.29	0.374	0.484
8-bit Quant Pruned	0.177	0.18	0.22	0.248	0.32	0.433	0.548
Quant Pruned Global	0.2	0.21	0.212	0.214	0.26	0.312	0.635

V. Conclusion

The paper presented an evaluation of several neural network compression techniques for embedded energy management. We show how significant model size reductions can be achieved with limited degradation in the MSE performance metric for the prediction task. Future work is focused on deploying and evaluating the resulting model structures for online inference and local control on embedded edge devices.

References

[1] L. C. S. et al., "Multi-view neural network ensemble for short and mid-term load forecasting," *IEEE Transactions on Power Systems*, vol. 36, no. 4, pp. 2992–3003, 2021.

[2] G. Stamatescu, R. Entezari, K. Römer, and O. Saukh, "Deep and efficient impact models for edge characterization and control of energy events," in *IEEE 25th Intl Conf on Parallel and Distributed Systems*, 2019.

[3] J. Moon, S. Park, S. Rho, and E. Hwang, "A comparative analysis of artificial neural network architectures for building energy consumption forecasting," *International Journal of Distributed Sensor Networks*, vol. 15, no. 9, p. 1550147719877616, 2019.

[4] A. M. Tudose, D. O. Sidea, I. I. Picioroaga, V. A. Boicea, and C. Bulac, "A cnn based model for short-term load forecasting: A real case study on the romanian power system," in *2020 55th International Universities Power Engineering Conference (UPEC)*, 2020, pp. 1–6.

[5] P. Waqas Khan, Y.-C. Byun, S.-J. Lee, and N. Park, "Machine learning based hybrid system for imputation and efficient energy demand forecasting," *Energies*, vol. 13, no. 11, p. 2681, May 2020.

[6] G. Hafeez, K. S. Alimgeer, A. B. Qazi, I. Khan, M. Usman, F. A. Khan, and Z. Wadud, "A hybrid approach for energy consumption forecasting with a new feature engineering and optimization framework in smart grid," *IEEE Access*, vol. 8, pp. 96 210–96 226, 2020.

[7] C. Nichiforov, G. Stamatescu, I. Stamatescu, V. Calofir, I. Fagarasan, and S. S. Iliescu, "Deep learning techniques for load forecasting in large commercial buildings," in *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*, 2018, pp. 492–497.

[8] C. Nichiforov, G. Stamatescu, I. Stamatescu, and I. Făgărăşan, "Evaluation of sequence-learning models for large-commercial-building load forecasting," *Information*, vol. 10, no. 6, p. 189, Jun 2019.

[9] C. Nichiforov, G. Stamatescu, I. Stamatescu, and I. Făgărăşan, "Learning dominant usage from anomaly patterns in building energy traces," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020, pp. 548–553.

[10] R. Hadidi, J. Cao, Y. Xie, B. Asgari, T. Krishna, and H. Kim, "Characterizing the deployment of deep neural networks on commercial edge devices," in *2019 IEEE International Symposium on Workload Characterization (IISWC)*, 2019, pp. 35–48.

[11] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Processing Magazine*, vol. 35, no. 1, 2018.

[12] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient transfer learning," *arXiv e-prints*, 11 2016.

[13] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights," *arXiv e-prints*, p. arXiv:1702.03044, Feb. 2017.

[14] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] S. R. Jain, A. Gural, M. Wu, and C. H. Dick, "Trained Quantization Thresholds for Accurate and Efficient Fixed-Point Inference of Deep Neural Networks," *arXiv e-prints*, p. arXiv:1903.08066, Mar. 2019.

[16] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.

[17] M. A. et al., "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, Nov. 2016.

[18] A. Koubaa, A. Ammar, A. Kanhouch, and Y. Alhabashi, "Cloud versus edge deployment strategies of real-time face recognition inference," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2021.