LightNILM: Lightweight neural network methods for non-intrusive load monitoring

Zhenyu Lu
Beijing Institute of Technology
Haidian, Beijing, China

Yurong Cheng
Beijing Institute of Technology
Haidian, Beijing, China

Mingjun Zhong
University of Aberdeen
Aberdeen, United Kingdom

Wenpeng Luan
Tianjin University
Tianjin, China

Yuan Ye
Beijing Institute of Technology
Haidian, Beijing, China

Guoren Wang
Beijing Institute of Technology
Haidian, Beijing, China

ABSTRACT
The aim of non-intrusive load monitoring (NILM) is to infer the energy consumed by the appliances in a house given only the total power consumption. Recently, literature have shown that deep neural networks are the state-of-the-art approaches for tackling NILM. For example, both sequence-to-sequence (seq2seq) and sequence-to-point (seq2point) learning models are the popular frameworks with typical network architectures such as convolutional neural networks (CNNs). However, these deep neural network approaches are computationally expensive and require huge storage for the purpose of prediction, and consequently would not be capable of deploying on mobile/edge devices. This paper addresses these issues for seq2point learning models by employing specifically designed network architectures which can be processed by using TensorFlow Lite to deploy on mobile phones. We show that our models only require 0.5% number of the parameters used in original seq2point models, whilst achieve comparable accuracy. Our models are then successfully tested on mobile phones with reasonable accuracy performance.

CCS CONCEPTS
• Computing methodologies → Machine learning; Neural networks.

KEYWORDS
Energy disaggregation, lightweight NILM, deep learning, edge NILM, sequence-to-point learning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM Reference Format:

1 INTRODUCTION
The purpose of NILM is to disaggregate the total energy consumption of a household into the individual appliances. It is very helpful for householders to save energy by providing them with how energy was used in their houses. For example, research has shown that NILM can help to reduce energy consumption in a household by 15% [5]. In addition, integrating NILM into smart meters could help energy providers to optimize grid operation models and customize specific tariffs for users based on their energy consumption habits [10].

Practically, there would be two approaches for the roll-out of a NILM system: Firstly, the NILM algorithms could be implemented by cloud computing which means the appliances are disaggregated on the cloud and then smart devices read the disaggregated signals from there; Secondly, the NILM algorithms are implemented by edge computing which means that NILM algorithms are directly embedded into edge devices including smart meters, smart phones, and embedded systems. The first approach would tolerate substantial computational costs required by NILM algorithms, but need significant resources for cloud computing and as well as high quality network connections. The scalability of the approach using cloud computing would be limited as the technique may be deployed in every house which thus require immense communications. Perhaps, data privacy must be seriously considered to ensure cloud computing not to put the privacy of personal data at risk when collecting, storing, transferring and sharing data. By contrast, the second approach may not have these limitations, and thus is fast, secure, and more reliable for real-world applications. For example, scalability would not be an issue as the edge computing device is standalone and thus can be independently distributed into every house; data privacy would not be an issue any more as the smart/edge device could be encrypted.

In this paper, we investigate three lightweight architectures on the seq2point learning frameworks [29] for deployment on mobile devices and so potentially other edge devices in general. In the following we call our approaches as LightNILM. These approaches can be compressed by using TensorFlow Lite and then deployed on mobile phone. Our approaches are most relevant to [19] which combines MobileNet [9] and Tensorflow Lite. [30] proposed a method similar to [19], which uses pooling layer and dropout to achieve available accuracy with fewer parameters. These approaches are deemed as preliminary results in our view as no comparisons with variants were provided in their paper. What is more, based on the
previous two methods, [3] proposed a NILM method for embedded systems using model compression methods and multi-task learning. The algorithm is not explicitly described in their papers. Indeed, our paper compares three approaches for LightNILM, which are sliced recurrent neural networks (SRNN), depthwise separable convolutional neural networks (DepthWCNN), and residual neural networks (ResNet). We test our LightNILM approaches by applying them to three different data sets which are UK-DALE, REFIT, and REDD. They were tested on mobile phone devices. This provides a thorough analysis over LightNILM showing that ResNet would be the most promising approach for mobile devices and potentially edge devices in general. In the rest of our paper, we briefly review the related work of NILM in Section 2. Then in Section 3, we introduce our three LightNILM approaches in detail. In Section 4, we introduce the data sets used in the experiments. In Section 5, we report the results and corresponding analysis of our LightNILM methods tested on three datasets. Finally, we conclude our paper in Section 6.

2 NON-INTRUSIVE LOAD MONITORING

This section briefly reviews relevant methods for NILM. Given a time series of the measured power readings of the mains electricity in a household represented by \( Y = \{y_1, y_2, \ldots, y_T\} \) where \( T \) denotes the time, the aim of NILM is to infer the energy consumption of all the individual appliances by only using \( Y \). NILM can be viewed as a single channel blind source separation problem as the total power reading is the sum of the individual appliances: \( Y_t = \sum_{i=1}^T x_{it} + \epsilon \), where \( x_{it} \) is the power reading of the \( i^{th} \) appliance and \( \epsilon \) is usually a Gaussian noise. The aim is then to infer \( x_{it} \).

Solutions to NILM include statistical methods, e.g., hidden Markov models (HMM) [2, 16, 33, 34], signal processing algorithms, e.g., graph signal processing (GSP) [7, 32], and deep neural networks [4, 6, 12, 20, 28, 29]. Among them, HMM and GSP algorithms would be scalable for edge computing; however, literature suggests that these approaches would not have better performance, compared to deep neural networks. The seq2seq and seq2point learning models [12, 29] are popular frameworks among those deep learning approaches. There have been various deep neural network architectures for implementing these frameworks in literature [11, 12, 20, 24, 26, 28, 29, 35]. However, these models could only be implemented with cloud computing due to the requirements of high computational costs and huge memory storage. There have been some work to extend seq2point learning models to edge computing. For example, the works of [1, 15, 25, 31] targeted to compress the seq2point model by pruning the model parameters; the work of [19] employs MobileNet [9] and Tensorflow Lite to compress the seq2point model. These approaches are promising for deployment on edge devices; for example, [19] has already tested the model on mobile phones. Some other lightweight NILM algorithms use compression methods and as well as traditional machine learning algorithms [18, 21–23].

Our aim in this paper is to adapt the seq2point learning model by using sliced recurrent neural network [27], depthwise convolutional neural networks [9], and residual neural network [8], which are then compressed by Tensorflow Lite so that they can be deployed on mobile devices.

3 LIGHTWEIGHT NILM MODELS

In this section, we describe the proposed lightweight NILM models which are adapted from the seq2point framework by using the following architecture strategies for the purpose of deploying on mobile devices.

3.1 Sliced recurrent neural networks

The first approach is to replace the CNN layers in seq2point except the first CNN layer by a SRNN block [27] with other layers being unchanged, leading to the SRNN architecture (Figure 1). The advantages are that SRNN has fewer parameters compared with CNN, and its parallelism accelerates training. Notably, the size of the dense layer is also largely reduced. Given an input sequence \( Y = \{y_1, y_2, \ldots, y_T\} \), we set a same recurrent unit \( S \) for each reading \( y_i \in Y \). Let \( k \) readings form a subsequence, e.g., \( k = 8 \) in Figure 1(b). So the output of the first layer has \([T/k]\) units denoted by \( \{y_1, y_2, \ldots, y_{T/k}\} \) which in turn is as the input of the second layer. This process repeats until the output has a single unit.

3.2 Depthwise separable convolutional neural networks

We can also apply depthwise convolutional neural networks (DepthWCNN) method [9] to break the large convolution kernels into the sum of several small convolution kernels, for the convolutional layers and dense layer. This method can reduce the number of parameters and the computational complexity of the convolutional layers and dense layer.

The input \( Y = \{y_1 \ldots y_T\} \) is treated as a \( T \times 1 \) sequence. Denote the feature map as \( F \), the input size of \( F \) as \( T \times M \), and the output size of \( F \) as \( T \times N \), where \( M \) (resp. \( N \)) is the number of input (resp. output) channels. Let the convolution kernel be \( K \) with size \( D_K \). If we apply standard convolution, the output of each layer \( G \) is computed as \( G = \sum_{i,m} K_{i,m} F_{k+i-1,m} \) where \( i \in [1, D_K], m \in [1, M], n \in [1, N] \).

The number of parameters of \( K \) is \( D_K \times M \times N \). So the computational complexity is \( D_K \times M \times N \).
The computational complexity is too large, so we apply the depthwise separable convolutions which is composed of a channel-wised convolution and a 1×1 standard convolution, which is called pointwise convolution. Let the convolution kernel be \( \hat{K} \), the output of each layer \( G \) assuming padding would be computed as \( \hat{G} = \sum_k \hat{K}_{m \times n}^t F_{k+i-1} \). Then the computational complexity of depthwise separable convolution is \( D_k \cdot M \cdot T + M \cdot N \cdot T \), which is the sum of the depthwise and pointwise convolution. We get a reduction in computational complexity of \( \frac{D_k \cdot M \cdot T + M \cdot N \cdot T}{D_k \cdot M + M \cdot N + T} = \frac{1}{D_k} + \frac{1}{M} \). What is more, the fully connected layer accounts for more than 99% of the parameters in the seq2point model which has more than 30 million parameters.

The fully connected layer could also be regarded as a larger convolution layer. The input and the parameters of the fully connected layer are \( a_1, \ldots, a_5, a_2, \ldots, a_3, a_1, \ldots, a_3, w_1, \ldots, w_1, w_2, \ldots, w_2, \ldots, w_3, \ldots, w_3, \ldots, w_3, w_2, \ldots, w_2, w_1, w_1, w_1, w_0, \ldots, w_0 \) respectively. So the fully connected layer can be expressed as matrices \( A = (a_{ij}) \) and \( W = (w_{ij}) \) (i = 1, \ldots, m; j = 1, \ldots, s).

If the kernel size of convolution equals to the sequence length, then the convolution progress could be expressed as: \( U = \sum_i \sum_j a_{ij} w_{ij} + b \), where \( U \) denotes one unit of fully connected layer and \( b \) denotes the bias of unit. Thus, similar to the above process we can also apply depthwise separable convolution to compress the fully connected layer. This model is shown in Figure 2.

3.3 Residual neural network

The final approach is to employ the residual neural network (ResNet) [8]. For each convolutional layer in the DepthWCNN based method, ResNet can help to reuse the results of features in the last layer, which can improve the computational accuracy.

The framework of ResNet is shown in Figure 3. The idea is simply to use Res-block to replace depthwise separable convolution layer. Res-block has two structures, where the first is shown in Figure 3(a) and the second is shown in Figure 3(b). The block1 is similar to dense-block except that res-block add the input feature map to the output of convolutional layer instead of concatenation. Thus the building block1 is defined as: \( G = H(F, W) + F \), where \( F \) and \( G \) are the input and output feature maps of the block, and \( W \) are the parameters of the convolutional layer. The function \( H(F, W) \) denotes the convolutional layer. The process requires much less computational complexity than convolution without extra parameters. But as is shown in Figure 3(a), it could not add the input feature maps to the output of convolutional layer directly because the input and output feature maps have different dimensions. So we use a pointwise convolution \( P \) to increase the dimension of input to the same as the output dimension of convolutional layer. Thus the building block2 is defined as: \( G = H(F, W) + P(F, W) \). The framework of ResNet is shown in Figure 3(c). When the input and output are of the same dimensions, block1 is used, otherwise block2 is used. A large separable convolution layer is used at the end of the model instead of a fully connected layer for the output.

4 DATA SETS

Three data sets UK-DALE [13], REFIT [17] and REDD [14] are used to evaluate the performance of the proposed models for NILM.

UK-DALE: The active power acquisition time of the UK-DALE data set is 1s-6s. The house 1 is used for training and validation, and house 2 used for testing. Five appliances are used which are kettle, microwave, fridge, dish washer, and washing machine.

REFIT: The active power sampling time of all appliances in the REFIT data set is 8s. The houses assigned for training, validation and test are shown in the Table 1.

REDD: The data acquisition time is 1s-3s. The houses 2&3 are used for training, house 3 used for validation and house 1 for testing.

Table 1: Splitting of REFIT data and the mean, standard deviation values for preprocessing on all datasets.

<table>
<thead>
<tr>
<th>Appliances</th>
<th>training</th>
<th>validation</th>
<th>test</th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mains</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>522</td>
<td>814</td>
</tr>
<tr>
<td>kettle</td>
<td>3, 4, 7, 8, 9, 12, 13, 19, 20</td>
<td>5</td>
<td>2</td>
<td>700</td>
<td>1000</td>
</tr>
<tr>
<td>microwave</td>
<td>10, 12, 19</td>
<td>17</td>
<td>4</td>
<td>500</td>
<td>800</td>
</tr>
<tr>
<td>fridge</td>
<td>2, 5, 9</td>
<td>12</td>
<td>15</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td>dish washer</td>
<td>5, 7, 9, 13, 16</td>
<td>18</td>
<td>20</td>
<td>700</td>
<td>1000</td>
</tr>
<tr>
<td>washing machine</td>
<td>2, 5, 7, 9, 15, 16, 17</td>
<td>18</td>
<td>8</td>
<td>400</td>
<td>700</td>
</tr>
</tbody>
</table>

5 EXPERIMENTAL RESULTS

Metrics and Settings: The proposed models are applied to the three data sets. For evaluating the performance, mean absolute error (MAE) and normalized signal aggregate error (SAE) are used. MAE is defined as \( \frac{1}{T} \sum_{t=1}^{T} |x_t - \hat{x}_t| \) where \( x_t \) and \( \hat{x}_t \) is the true power reading at time \( t \), \( \bar{x} \) is an mean power, and \( \sigma \) is a standard deviation of power reading. It could accelerate the convergence of model weight parameters. The means and standard deviations are shown in Table 1.

Accuracy: The results shown in Tables 2 and 3 indicate that in general our LightNILM models are comparable to the original seq2point in terms of MAE, but interestingly they outperform...
The ResNet block 2:

- **block 2**:
  - **Conv**
  - **block1**: Filters: 32, Strides: 1, Activ: Selu
  - **block2**: Filters: 48, Strides: 1, Activ: Selu

(c) The residual neural network

Figure 3: The ResNet architecture

Table 2: The results of the seq2point in [29] and the proposed LightNILM models in this paper applied to UK-DALE, REFIT, REDD datasets running on a desktop computer and a mobile phone. Best results are denoted in bold across methods.

<table>
<thead>
<tr>
<th></th>
<th>UK-DALE</th>
<th>REFIT</th>
<th>REDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metrics</td>
<td>MAE</td>
<td>SAE</td>
<td>Depth/Wise</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ResNet</td>
</tr>
<tr>
<td>Kettle</td>
<td>7.4</td>
<td>0.069</td>
<td>15.9</td>
</tr>
<tr>
<td>Microwave</td>
<td>6.8</td>
<td>0.130</td>
<td>22.7</td>
</tr>
</tbody>
</table>

... (Table 2 continues with similar entries for other models and metrics)

Table 3: The proposed LightNILM models in this paper applied to UK-DALE, REFIT, REDD datasets running on a mobile phone. Best results are denoted in bold across methods.

<table>
<thead>
<tr>
<th></th>
<th>UK-DALE</th>
<th>REFIT</th>
<th>REDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metrics</td>
<td>MAE</td>
<td>SAE</td>
<td>Depth/Wise</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ResNet</td>
</tr>
<tr>
<td>Kettle</td>
<td>33.7</td>
<td>0.051</td>
<td>31.0</td>
</tr>
<tr>
<td>Microwave</td>
<td>34.8</td>
<td>0.082</td>
<td>31.3</td>
</tr>
</tbody>
</table>

... (Table 3 continues with similar entries for other models and metrics)

Table 4: Numbers of parameters, and model sizes, and running time of the MobileNILM models.

<table>
<thead>
<tr>
<th></th>
<th>Seq2point</th>
<th>SRNN</th>
<th>DepthWCNN</th>
<th>ResNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Parameters (Million)</td>
<td>30.708</td>
<td>0.149</td>
<td>0.083</td>
<td>0.108</td>
</tr>
<tr>
<td>TensorFlow Model size (MB)</td>
<td>449</td>
<td>2.03</td>
<td>1.04</td>
<td>1.42</td>
</tr>
<tr>
<td>TFLite Model size (MB)</td>
<td>149</td>
<td>0.340</td>
<td>0.11</td>
<td>0.14</td>
</tr>
<tr>
<td>Run time (ms)</td>
<td>–</td>
<td>–</td>
<td>20.58</td>
<td>27.51</td>
</tr>
</tbody>
</table>

To summarize, ResNet would be the best model among LightNILM by comprehensively considering those factors of accuracy, computation speed and the model size.

6 CONCLUSIONS

Several lightweight NILM models based on neural networks are proposed. We compared these models with seq2point model on UK-DALE, REDD and REFIT data sets. We found that each of our models only needs no more than 0.5% numbers of parameters required by the original seq2point model, and remarkably our methods are comparable comparing to the original model.

It is noticed in preliminary studies that the learned parameters for seq2point model are redundant [1]. Our LightNILM methods using specially designed neural network architecture blocks can effectively reduce the number of parameters, whilst achieve comparable performance. Nevertheless, the accuracy of the disaggregation on mobile devices need to be further improved in future research.

ACKNOWLEDGEMENT

This work is supported in part by the National Natural Science Foundation of China (Key Program), NSFC-SGCC (U2066207).