Using Step Variant Convolutional Neural Networks for Energy Disaggregation

Bingsheng Wang, Haili Dong, Chang-Tien Lu
Department of Computer Science
Virginia Tech
Falls Church, VA 22043
{claren89,hailid,ctlu}@vt.edu

Abstract

Energy disaggregation is the process of decomposing the total energy consumption of a household into its individual devices. In this paper, we propose Step Variant Convolutional Neural Networks (SVCNNs) for low-sampling-rate energy disaggregation through hard-coding the weight replication among feature maps. Unlike existing disaggregation techniques whose performance largely depends on prior knowledge or tuning parameters, our method is designed to reduce model complexity while preserving its discriminative capability. By restricting the variance of weights in feature maps, compared with Convolutional Neural Networks (CNNs), SVCNNs need much less tuning parameters. A backpropagation based algorithm is designed for learning model parameters. Experiments with real world dataset were conducted to demonstrate the effectiveness of our approach.

1 Introduction

The increase of fossil fuel consumption and greenhouse gas emission pose critical challenges to the global society and economy. A study conducted by BP shows that in 2011 the world consumed 4059.1 million tonnes of oil, 3222.9 billion cubic metres of natural gas, and 7780.2 million tonnes of coal for power generation, producing more than thirty thousand million tonnes of CO_2 [1]. WNA predicts that if the current energy usage patterns continue, there will be a 70% growth in electricity consumption from 2010 to 2035 (from 21,511 TWh to 36,600 TWh) [2]. Previous studies have shown that providing appliance-level information to end users significantly affect their behaviours and ultimately help them save a substantial amount of consumption [3, 4, 5]. This motivates us to employ machine learning techniques to decompose aggregated energy consumption into its constituent components and provide residents with detailed usage information.

In this paper, we focus on disaggregating low-sampling-rate (generally 1/900 Hz) smart meter readings. Kolter *et al.* proposed Discriminative Disaggregation Sparse Coding (DDSC) [6] by defining the regularized disaggregation error as the objective function, aiming at learning discriminative basis functions. However, it requires massive labelled data for tuning parameters. Sparse Coding with Featured Discriminative Dictionary [7] and Deep Sparse Coding [8] were developed to extend DDSC, but they either largely depend on prior knowledge or are too complex for achieving optimal solutions. The Hidden Markov Model (HMM) [9, 10], Hierarchical Probabilistic Model [11], and Factorial Hidden Markov Model [12, 13] (FHMM) based approaches were presented by incorporating prior knowledge or using sophisticated structure, such as correlation between appliances. However, the performance of these models is mainly determined by the quality of prior knowledge, and it's impractical to collect authoritative prior knowledge, especially when handling data in large areas with various consumption patterns.

Convolutional Neural Networks (CNNs) [14] have been successfully applied to numerous *recognition/classification* tasks, such as digit/object recognition and natural language processing [15], and the three key characteristics: local filters, pooling, and weight-sharing, enable CNNs to have strong discriminative capability. However, it lacks the mechanism to control weight-sharing among feature maps, which can reduce the computational load. To the best of our knowledge, there is no application of CNNs based method to the energy disaggregation task. We consider the new challenges for the application of CNNs based approach to *low-sampling-rate* energy disaggregation.

Due to high collection cost, invasion of privacy, and transmission reliability concerns, there is a lack of device level consumption data. Considering CNNs' independent structure of feature maps, several issues might appear, such as over-fitting or the lost of its generalization ability. To reduce the number of tuning parameters while maintaining the discriminative capability of CNNs, we propose a step variant constraint to control the weigh-sharing among feature maps, and show that it can significantly improve the disaggregation performance. The weight replication mechanism requires adjacent feature maps to share identical weights with only one weight to be sequentially learned. Not only does this structure substantially reduce the number of tuning parameters, but it also intensifies the correlation between feature maps.

2 Step Variant Convolutional Neural Networks

In this section, the terminologies for the energy disaggregation problem is first formulated, and then the SVCNNs are introduced for the disaggregation task.

Terminology: Assume there are M devices in a household, such as lamp, computer, and dish washer. For each device $j = 1, \ldots, M$, there is a corresponding energy consumption matrix $\mathbf{Y}^{(j)} \in R^{N \times Q}$, where N is the number of intervals in one day, and Q is the number of days. The $q^{\texttt{th}}$ column of $\mathbf{Y}^{(j)}$, denoted by $\mathbf{y}^{(j)}_{\cdot,q} \in R^{N \times 1}$, is the energy usage of the $q^{\texttt{th}}$ day for a particular device j, where $q = 1, \ldots, Q$. The $i^{\texttt{th}}$ element in vector $\mathbf{y}^{(j)}_{\cdot,q}$, denoted by $y^{(j)}_{i,q}$, is the energy consumption value of device j at interval i in day q, named as interval level consumption. We denote the aggregated energy consumption over all equipments as $\bar{\mathbf{Y}}$, $\bar{\mathbf{Y}} = \sum_{j=1}^{M} \mathbf{Y}^{(j)}$. During the training phase, we assume that we can access the individual device consumption matrix $\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \ldots, \mathbf{Y}^{(M)}$. During the testing phase, we can only access the aggregated consumption matrix $\bar{\mathbf{Y}}$, and the goal is to decompose $\bar{\mathbf{Y}}$ into its individual components $\hat{\mathbf{Y}}^{(1)}, \hat{\mathbf{Y}}^{(2)}, \ldots, \hat{\mathbf{Y}}^{(M)}$.

SVCNNs for the Disaggregation Task: We develop a method to enable the weight replication among feature maps for the disaggregation task. We illustrate this process using three inputs for constructing the step variant structure, but the procedure can be generalized for any number of inputs. As shown in Figure 1, there are three inputs in the local receptive, and four feature maps are considered for learning invariances. One typical feature map connects to all three inputs and is of the size 1×1. Under the setting of CNNs, there will be $4 \times (3 + 1) = 16$ tuning parameters in this layer. Assume the training data is limited, it's difficult to properly train so many parameters. Instead, we apply a step variant constraint among feature maps to reduce the number of learning parameters. In Figure 1, shared weights have the same color and they are constrained to be identical. Four feature maps are discriminated with different textures. The weight denoted by dash line is the variant weight while solid lines indicate identical weights, and only one variant weight needs to be learned for the other three feature maps. The upper bound for the number of feature maps is one plus the number of elements in the local receptive fields. Suppose there are K inputs in the local receptive field and $P (P \leq K + 1)$ feature maps are considered as filters. $P \times (K + 1)$ parameters need to be learned, whereas the number of parameters for Step Variant CNNs (SVCNNs) is only

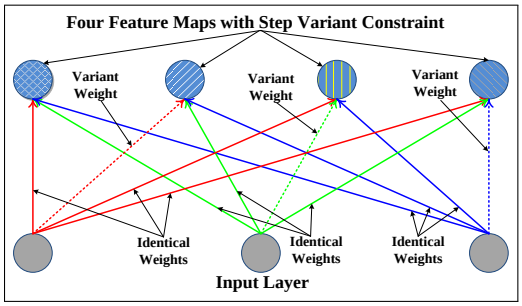

Figure 1: Illustration of the step variant feature maps.

$(K+1)+2(P-1) = K+2P-1$, which significantly reduces the number of parameters by $(K-1)\times(P-1)$ (K and P are usually much larger than 1 in real scenarios).

We intend to disaggregate the interval level aggregated consumption $\bar{\mathbf{Y}}_i$ into its individual components, where $i = 1, 2, \cdots, N$. Considering the influences of interval i's adjacent intervals, the input is set as $[\bar{y}_{i-2,q}, \bar{y}_{i-1,q}, \bar{y}_{i,q}, \bar{y}_{i+1,q}, \bar{y}_{i+2,q}]$. When $i \leq 2$ or $i \geq N-2$, we set the corresponding aggregated consumption outside the range of $[1, N]$ to be 0, indicating no contribution or influence. The aggregated consumption at interval i, i.e., $\bar{y}_{i,q}$, is disaggregated into several individual devices. Its two preceding and successive intervals' aggregated consumption are considered for extracting the local features. We set the size of local receptive field to be three, learn four feature maps with each local receptive field, and perform max-pooling on the learned feature maps.

The disaggregation process is clarified by showing how it works for an individual device. We set the input as the interval level aggregated consumption, and there is one output to estimate the consumption of a typical device. The structure for disaggregation with SVCNNs is a three layer network: The first hidden layer is a ***convolution*** layer, consisting of four feature maps. A unit in one feature map is connected to three consecutive aggregated interval based consumption. The ***sub-sampling*** layer (max-pooling layer) locates in the second hidden layer. Each unit in this layer computes the max value of its three inputs, multiplies it by a tuning weight, adds a tuning bias, and activates it with a sigmoid function. The ***output*** layer is the third layer, which is fully connected with the sub-sampling layer, and the result is activated with a sigmoid function.

A backpropagation based algorithm is designed to learn the parameters of SVCNNs. The Mean Squared Error (MSE) $E_{i,q}$ of the interval level consumption is employed as the loss function, where $i = 1, 2, \cdots, N$, and $q = 1, 2, \cdots, Q$. The feed-forwad computation and backpropagation to the output/sub-sampling layer are the same as those for CNNs. Here we concentrate on the backpropagation to the convolution layer, where the step variant constraint is applied. Formally, let $\mathbf{o} = [o_1, o_2, o_3]$ denotes the inputs in the local receptive field. $\mathbf{o^{(1)}}$ denotes the output of the convolution layer. $\boldsymbol{\delta}^{(1)}$ and $\boldsymbol{\delta}^{(2)}$ are respectively the backpropagation errors for the convolution and sub-sampling layer. Then the backpropagation error for the convolution layer is given by

$$\boldsymbol{\delta}^{(1)} = \mathbf{D}^{(1)} \circ \boldsymbol{\tau}^{(2)}, \tag{1}$$

$$\begin{aligned}
&\mathbf{D}^{(1)} = \left[\mathbf{D}_1^{(1)}, \mathbf{D}_2^{(1)}, \cdots, \mathbf{D}_M^{(1)}\right], \quad \mathbf{D}_j^{(1)} = \begin{bmatrix} o_{j11}^{(1)}(1-o_{j11}^{(1)}) & \cdots & o_{j1L}^{(1)}(1-o_{j1L}^{(1)}) \\ o_{j21}^{(1)}(1-o_{j21}^{(1)}) & \cdots & o_{j23}^{(1)}(1-o_{j23}^{(1)}) \\ \vdots & \ddots & \vdots \\ o_{jK1}^{(1)}(1-o_{jK1}^{(1)}) & \cdots & o_{jKL}^{(1)}(1-o_{jKL}^{(1)}) \end{bmatrix}, \\
&\boldsymbol{\tau}^{(2)} = \left[\boldsymbol{\tau}_1^{(2)}, \boldsymbol{\tau}_2^{(2)}, \cdots, \boldsymbol{\tau}_M^{(2)}\right], \quad \boldsymbol{\tau}_j^{(2)} = \mathrm{UP}\left(\mathbf{w}_j^{(2)} \circ (\boldsymbol{\delta}_j^{(2)})^T\right), \\
&\mathbf{w}_j^{(2)} = \left[w_{j1}^{(2)}, w_{j2}^{(2)}, \cdots, w_{jK}^{(2)}\right]^T,
\end{aligned} \tag{2}$$

where $j = 1, 2, \cdots, M$, K is the number of feature maps for device j, L is the number of units in each of the K feature maps, $\circ$ denotes element-wise multiplication, and $\mathrm{UP}(\cdot)$ in Equation (14) denotes the replication of the vectors for L times in the horizontal direction. Then the derivative is given by

$$\frac{\partial E_{i,q}}{\partial \mathbf{W}^{(1)}} = \boldsymbol{\delta}^{(1)}\mathrm{UP}(\mathbf{o}), \tag{3}$$

where $\mathrm{UP}(\mathbf{o})$ means to replicate $\mathbf{o} = [o_1, o_2, o_3]$ for M times in the vertical orientation.

3 Experimental Results

The REDD dataset [16] was used for empirical evaluation, consisting of 1 Hz power readings for 6 real homes. Since Home 5 in REDD contains only two devices, we didn't include it in our evaluation. To evaluate the disaggregation performance under the setting of low-sampling-rate environment, we aggregated the 1 Hz power readings to be 1/900 Hz.

We considered three baselines for comparison. The first one is Discriminative Disaggregation Sparse Coding (DDSC) model with Total Energy Priors (TEP) and Group Lasso (GL) extensions, i.e., DDSC+TEP+GL [6]. The second is Factorial Hidden Markov Model (FHMM) based approach

[12, 16]. Additionally, we considered CNNs as the third baseline. The device-level disaggregation performance is evaluated with precision, recall, and F-measure [13]. The precision is the fraction of disaggregated consumption that is correctly classified, recall is the fraction of true device level consumption that is successfully separated, and F-measure is $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$. We also evaluated whole-home level performance with Normalized Disaggregation Error (NDE).

70% homes (3 homes) were used for training and the other 30% (2 homes) for testing. The 3 training homes were randomly selected and the remaining 2 homes were used for testing. 10 runs were conducted and the average performance is reported in Table 1. There is a total of 38 categories of devices in the REDD dataset. Since not all the devices are shared by both the training and testing sets, we investigated ten devices for comparison. SVCNNs significantly outperformed other methods over 7 individual devices, and achieved close to the best performance for the other 3 devices. SVCNNs achieved 41.97% in average F-measure (the mean value of all devices F-measure) while CNNs, FHMM, and DDSC+TEP+GL respectively achieved 36.38%, 23.10%, and 20.34%. For devices 2, 7, 8, 10, CNNs can achieve close F-measure as SVCNNs. However, it only achieved less than 10% in F-measure for devices 1, 5, 9. This indicates that due to the independent structure among feature maps, it's difficult to properly train CNNs for disaggregation task. The whole-home level performance comparison is shown in Figure 2. As can be seen, SVCNNs outperformed CNNs, as well as FHMM and DDSC+TCP+GL with smaller normalized disaggregation error.

Table 1: Performance of test set (*Precision, Recall, F-measure*), and bold entries denote the best F-measure. (gfi: Ground Fault Interrupter)

Devices	SVCNNs	CNNs	FHMM	DDSC+TEP+GL
1 Bathroom_gfi	9.060%, 94.68%, **14.86%**	4.690%, 95.28%, 8.930%	2.330%, 98.41%, 4.560%	1.630%, 93.05%, 3.210%
2 Kitchen_outlets	29.77%, 70.51%, **41.86%**	26.28%, 73.28%, 38.69%	6.360%, 98.17%, 11.95%	7.750%, 23.69%, 11.67%
3 Lighting_1	38.07%, 55.78%, 45.25%	38.11%, 31.65%, 34.58%	37.78%, 62.49%, **47.09%**	27.42%, 73.76%, 39.98%
4 Lighting_2	67.17%, 79.28%, **72.72%**	51.97%, 86.48%, 64.92%	35.49%, 87.48%, 50.49%	29.23%, 67.01%, 40.70%
5 Lighting_3	11.01%, 85.07%, **19.50%**	2.680%, 99.99%, 5.210%	0.140%, 99.99%, 0.270%	0.050%, 58.92%, 0.100%
6 Mains_1	37.05%, 71.25%, **48.75%**	28.97%, 79.36%, 42.44%	90.96%, 22.61%, 36.21%	68.38%, 18.73%, 29.41%
7 Mains_2	75.02%, 58.13%, 65.51%	60.06%, 72.53%, **65.71%**	97.44%, 9.830%, 17.86%	94.77%, 12.62%, 22.27%
8 Refrigerator	50.99%, 54.12%, **52.51%**	51.66%, 50.33%, 50.98%	48.59%, 44.04%, 46.20%	39.36%, 51.57%, 44.65%
9 Smoke_alarms	10.05%, 73.12%, **17.67%**	3.720%, 99.24%, 7.170%	0.010%, 99.99%, 2.000%	90.05%, 0.010%, 2.000%
10 Stove	68.42%, 29.34%, 41.07%	59.86%, 36.28%, **45.17%**	7.840%, 85.04%, 14.36%	9.360%, 9.480%, 9.420%

4 Conclusion

In this paper, we present an effective approach for the low-sampling-rate energy disaggregation task. To overcome the limitations of existing disaggregation methods, we introduced Step Variant CNNs as an extension of CNNs by hard-coding the constraint of weight replication among feature maps. A backpropagation based algorithm is developed for the inference and learning of SVCNNs. The step variant method can help intensify the correlation between feature maps and enable the model to learn the discriminative features. This mechanism significantly reduces the number of tuning parameters, and alleviates the needs of massive labelled training data. The empirical evaluation on the real world dataset validated the effectiveness of the proposed method, and showed that SVCNNs outperformed the benchmark methods.

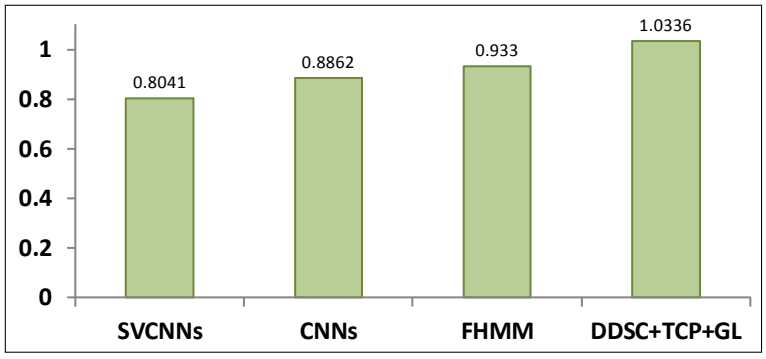

Figure 2: Test set *Normalized disaggregation error* on REDD, reporting the whole-home level performance.

References

[1] BP. Bp statistical review of world energy. 2012.

[2] WNA. World energy needs and nuclear power. 2012.

[3] C. Fischer. Feedback on household electricity consumption: a tool for saving energy? *Energy Efficiency*, 1(1):79–104, 2008.

[4] J. Froehlich, L. Findlater, and J. Landay. The design of ecofeedback technology. In *Proceedings of the SIGCHI*, pages 1999–2008, 2010.

[5] S. Houde, A. Todd, A. Sudarshan, J. Flora, and K. Armel. Real-time feedback and electricity consumption: A field experiment assessing the potential for savings and persistence. *Energy Journal*, 34(1):87–102, 2013.

[6] J. Kolter, S. Batra, and A. Ng. Energy disaggregation via discriminative sparse coding. In *Proceedings of the NIPS*, pages 1153–1161, 2010.

[7] B. Wang, F. Chen, H. Dong, A. P. Boedihardjo, and C.-T. Lu. Signal disaggregation via sparse coding with featured discriminative dictionary. In *Proceedings of the ICDM*, pages 1134–1139, 2012.

[8] H. Dong, B. Wang, and C.-T. Lu. Deep sparse coding based recursive disaggregation model for water conservation. In *Proceedings of the IJCAI*, pages 2804–2810, 2013.

[9] F. Chen, J. Dai, B. Wang, S. Sahu, M. Naphade, and C.-T. Lu. Activity analysis based on low sample rate smart meters. In *Proceedings of the ACM SIGKDD*, pages 240–248, 2011.

[10] O. Parson, S. Ghosh, M. Weal, and A. Rogers. Non-intrusive load monitoring using prior models of general appliance types. In *AAAI*, pages 356–362, 2012.

[11] B. Wang, H. Dong, A. P. Boedihardjo, F. Chen, and C.-T. Lu. A hierarchical probabilistic model for low sample rate home-use energy disaggregation. In *Proceedings of the SIAM Data Mining*, pages 704–712, 2013.

[12] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han. Unsupervised disaggregation of low frequency power measurements. In *Proceedings of the SIAM Data Mining*, pages 747–758, 2011.

[13] J. Kolter and T. Jaakkola. Approximate inference in additive factorial hmms with application to energy disaggregation. In *Proceedings of the AI Statistics*, pages 1472–1482, 2012.

[14] Y. Lecnn, L. Bottou, Y. Bengio, and P. Haffner. Gradient based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[15] R. Collobert and J.Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the ICML*, pages 160–167, 2008.

[16] J. Kolter and M. Johnson. Redd: A public data set for energy disaggregation research. In *Proceedings of the SustKDD*, pages 1–6, 2011.

[17] Raúl Rojas. *Neural networks: a systematic introduction*, chapter 7. Springer-Verlag Inc., New York, NY, USA, 1996. The Backpropagation Algorithm.

Appendix: Inference and Learning

We design algorithms to learn the parameters of SVCNNSs. The Mean Squared Error (MSE) of the interval based consumption is employed as the loss function of SVCNNs, for $\forall i = 1, 2, \cdots, N, l = 1, 2, \cdots, L$,

$$E_{i,l} = \frac{1}{M} \sum_{j=1}^{M} \left(y_{i,l}^{(j)} - \hat{y}_{i,l}^{(j)} \right)^2, \tag{4}$$

where $y_{i,l}^{(j)}$ is the ground truth and $\hat{y}_{i,l}^{(j)}$ is the estimated output of the SVCNNs.

In the follows, we present details of the algorithms for learning parameters. For illustration purpose, we use the B-diagram defined in [17] to show the backpropagation process. Since the learning process is identical for any interval in any day, here we illustrate this procedure using interval i at the l^{th} day.

Feed-forward computation: We compute the outputs of the networks from convolution layer to output layer (from layer 1 to layer 3), respectively denoted as $\mathbf{o}^{(1)}, \mathbf{o}^{(2)}$ and $\mathbf{o}^{(3)}$, where $o_j^{(3)} = \hat{y}_{i,l}^{(j)}$, and

$$\mathbf{o}^{(3)} = \left[\hat{y}_{i,l}^{(1)}, \hat{y}_{i,l}^{(2)}, \cdots, \hat{y}_{i,l}^{(M)} \right]^T.$$

Backpropagation to the output layer: We are looking for the first set of partial derivatives $\frac{\partial E_{i,l}}{\partial \mathbf{W}^{(3)}}$, where $\mathbf{W}^{(3)}$ denotes the weights for the connections between sub-sampling layer and output layer. The B-diagram based backpropagation path from the mean squared error of the network up to the output unit j is shown in Figure 3.

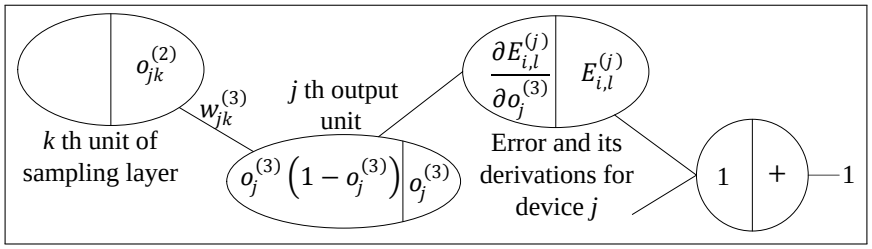

Figure 3: Backpropagation path up to the output unit j.

There are two parts in the B-diagram: the right side indicates the output of the primitive function while the left side denotes the derivation of the primitive function. From this path, the backpropagated error is given by,

$$\delta_j^{(3)} = \frac{\partial E_{i,l}^{(j)}}{\partial o_j^{(3)}} \times o_j^{(3)} \left(1 - o_j^{(3)}\right), \tag{5}$$

where $E_{i,l}^{(j)} = \frac{(y_{i,l}^{(j)} - \hat{y}_{i,l}^{(j)})^2}{M}$. By setting

$$\begin{aligned} \mathbf{d} &= \left[\frac{\partial E_{i,l}^{(1)}}{\partial o_1^{(3)}}, \frac{\partial E_{i,l}^{(2)}}{\partial o_2^{(3)}}, \cdots, \frac{\partial E_{i,l}^{(M)}}{\partial o_M^{(3)}} \right]^T, \\ \mathbf{D}^{(3)} &= \begin{bmatrix} o_1^{(3)}(1-o_1^{(3)}) & 0 & \cdots & 0 \\ 0 & o_2^{(3)}(1-o_2^{(3)}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & o_M^{(3)}(1-o_M^{(3)}) \end{bmatrix}, \end{aligned} \tag{6}$$

we can derive the backpropagated error for the entire output layer as,

$$\boldsymbol{\delta}^{(3)} = \mathbf{D}^{(3)} \mathbf{d}. \tag{7}$$

Then the derivative of $E_{i,l}$ over bias is directly given by Equation (7), while

$$\frac{\partial E_{i,l}}{\partial \mathbf{W}^{(3)}} = \boldsymbol{\delta}^{(3)} \mathbf{o}^{(2)}. \tag{8}$$

Back-propagation to the sub-sampling layer: Now we come to derive the partial derivatives $\frac{\partial E_{i,l}}{\partial \mathbf{W}^{(2)}}$, where $\mathbf{W}^{(2)}$ denotes the weights for the connections between convolution layer and sub-sampling layer. As shown in Figure 4, $o_{jk}^{(2)}$ is the output of unit k in the sub-sampling layer, where $k = 1, 2, \cdots, K$, and K is the number of feature maps.

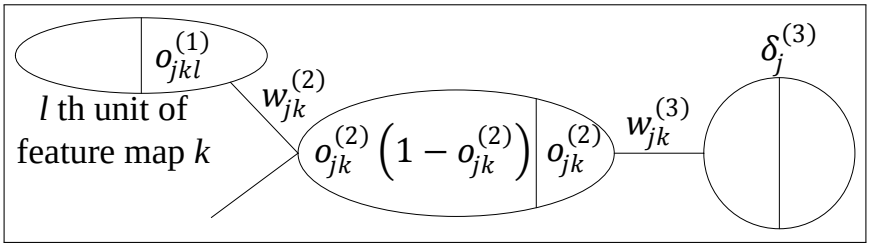

Figure 4: Backpropagation path up to the sub-sampling layer unit k for device j.

The backpropagated error for the path in the figure is,

$$\delta_{jk}^{(2)} = o_{jk}^{(2)} \left(1 - o_{jk}^{(2)}\right) w_{jk}^{(3)} \delta_j^{(3)}. \tag{9}$$

We define

$$\begin{aligned} \mathbf{D}^{(2)} &= \begin{bmatrix} o_{11}^{(2)}(1-o_{11}^{(2)}) & \cdots & o_{1K}^{(2)}(1-o_{1K}^{(2)}) \\ o_{21}^{(2)}(1-o_{21}^{(2)}) & \cdots & o_{2K}^{(2)}(1-o_{2K}^{(2)}) \\ \vdots & \ddots & \vdots \\ o_{M1}^{(2)}(1-o_{M1}^{(2)}) & \cdots & o_{MK}^{(2)}(1-o_{MK}^{(2)}) \end{bmatrix}, \\ \mathbf{W}^{(3)} &= \begin{bmatrix} w_{11}^{(3)} & w_{12}^{(3)} & w_{13}^{(3)} & w_{1K}^{(3)} \\ w_{21}^{(3)} & w_{22}^{(3)} & w_{23}^{(3)} & w_{2K}^{(3)} \\ \vdots & \vdots & \vdots & \vdots \\ w_{M1}^{(3)} & w_{M2}^{(3)} & w_{M3}^{(3)} & w_{MK}^{(3)} \end{bmatrix}. \end{aligned} \tag{10}$$

Thus, the backpropagated error for sub-sampling layer is given by,

$$\boldsymbol{\delta}^{(2)} = \mathbf{D}^{(2)} \circ \left(\mathbf{W}^{(3)} \circ \text{UP}\left(diag(\boldsymbol{\delta}^{(3)})\right)\right), \tag{11}$$

where $\circ$ denotes element-wise multiplication, $diag(\mathbf{X})$ represents fetch the elements on the main diagonal of the squared matrix $\mathbf{X}$ and form a column vector, and UP$(\cdot)$ indicates an up-sampling operation which simply replicates the input vector or element. In Equation (11), we replicate the $diag(\boldsymbol{\delta}^{(3)})$ four times in horizontal orientation to accommodate the multiplication operation. Obviously, the derivative of $E_{i,l}$ over the bias could be calculated by Equation (11), and

$$\frac{\partial E_{i,l}}{\partial \mathbf{W}^{(2)}} = \boldsymbol{\delta}^{(2)} \max\{\mathbf{o}^{(1)}\}, \tag{12}$$

where $\max\{\mathbf{o}^{(1)}\}$ denotes extracting the maximum value from each feature map.

Back-propagation to the convolution layer: The final step is to compute the partial derivatives $\frac{\partial E_{i,l}}{\partial \mathbf{W}^{(1)}}$, where $\mathbf{W}^{(1)}$ denotes the weights for the connections between input layer and convolution layer.

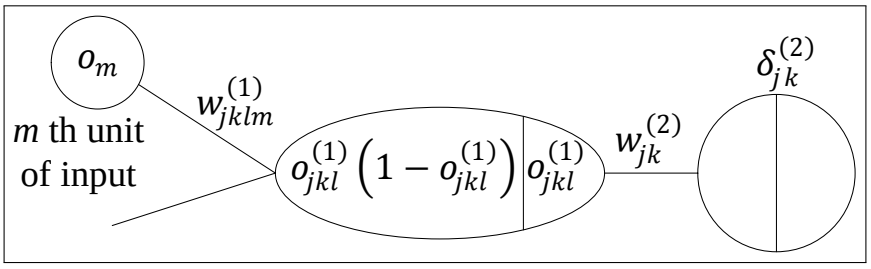

Figure 5: Backpropagation path up to the convolution layer unit l of feature map k for device j.

We show the backpropagation path up to the l^{th} unit in convolution layer. For convenience, we use o_m to denote the m^{th} aggregated input unit, where $m = 1, 2, 3$, corresponding to any consecutive three interval based aggregate consumption selected from $\bar{y}_{i-2,l}, \bar{y}_{i-1,l}, \cdots, \bar{y}_{i+2,l}$, and $\mathbf{o} = [o_1, o_2, o_3]$. $o_{jkl}^{(1)}$ represents the output of unit l in feature map k for device j, where $l = 1, 2, 3$. The backpropagated error can be computed by

$$\delta_{jkl}^{(1)} = o_{jkl}^{(1)}\left(1 - o_{jkl}^{(1)}\right) w_{jk}^{(2)} \delta_{jk}^{(2)}. \tag{13}$$

We use $\boldsymbol{\delta}_j^{(2)}$ to denote the j^{th} row of $\boldsymbol{\delta}^{(2)}$, and

$$\begin{aligned}
&\mathbf{D}^{(1)} = \left[\mathbf{D}_1^{(1)}, \mathbf{D}_2^{(1)}, \cdots, \mathbf{D}_M^{(1)}\right], \\
&\mathbf{D}_j^{(1)} = \begin{bmatrix} o_{j11}^{(1)}(1-o_{j11}^{(1)}) & \cdots & o_{j1L}^{(1)}(1-o_{j1L}^{(1)}) \\ o_{j21}^{(1)}(1-o_{j21}^{(1)}) & \cdots & o_{j23}^{(1)}(1-o_{j23}^{(1)}) \\ \vdots & \ddots & \vdots \\ o_{jK1}^{(1)}(1-o_{jK1}^{(1)}) & \cdots & o_{jKL}^{(1)}(1-o_{jKL}^{(1)}) \end{bmatrix},
\end{aligned} \tag{14}$$

$$\begin{aligned}
&\mathbf{D}^{(1)} = \left[\mathbf{D}_1^{(1)}, \mathbf{D}_2^{(1)}, \cdots, \mathbf{D}_M^{(1)}\right], \\
&\boldsymbol{\tau}^{(2)} = \left[\boldsymbol{\tau}_1^{(2)}, \boldsymbol{\tau}_2^{(2)}, \cdots, \boldsymbol{\tau}_M^{(2)}\right], \\
&\boldsymbol{\tau}_j^{(2)} = \text{UP}\left(\mathbf{w}_j^{(2)} \circ (\boldsymbol{\delta}_j^{(2)})^T\right), \\
&\mathbf{w}_j^{(2)} = \left[w_{j1}^{(2)}, w_{j2}^{(2)}, \cdots, w_{jK}^{(2)}\right]^T.
\end{aligned}$$

Consequently, the backpropagated error for the convolution layer is calculated by

$$\boldsymbol{\delta}^{(1)} = \mathbf{D}^{(1)} \circ \boldsymbol{\tau}^{(2)}, \tag{15}$$

where UP$(\cdot)$ in Equation (14) performs a replication of the vectors three times in the horizontal direction. Similarly, the derivative of $E_{i,l}$ over the bias can be computed by Equation (15), and

$$\frac{\partial E_{i,l}}{\partial \mathbf{W}^{(1)}} = \boldsymbol{\delta}^{(1)} \text{UP}(\mathbf{o}), \tag{16}$$

where UP$(\mathbf{o})$ means to replicate $\mathbf{o}$ for M times in the vertical orientation.