Convolutional Neural Network for Power Distribution Prediction in PWRs

Lee Jinyoung [a], Nam Younduk [a], Joo Han Gyu [b]
[a]*KEPCO NF, 242, 989 beon-gil, Daedeokdae-ro, Yuseong-gu, Daejeon, Republic of Korea*
[b]*Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 151-744, Korea*
Corresponding author: jinyounglee@knfc.co.kr

1. Introduction

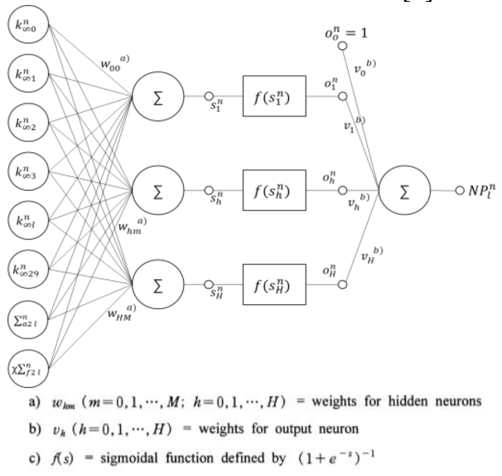
As computer specs have improved, 3-D core depletion computation speed has improved greatly, but it is still not enough to calculate a lot of loading patterns for optimization. In the past, there have been studies to improve this calculation speed using OLL(Optimization layer by layer), one of the artificial neural networks. As in the previous study, the purpose of this study is to construct an artificial neural network that predicts the 2-D power distribution.

In the previous study, if they looked at the fuel assembly one by one when analyzed loading pattern, we regarded loading pattern as a single image and looked at the relationship with the surrounding fuel assembly more importantly. Reflecting on this point, this research attempts to improve the speed and accuracy by converting the main neural network into CNN based on past research. In addition, burnup depletion was performed using an artificial neural network.

2. Review of previous works

2.1 Abstract

The optimization layer by layer (OLL) learning algorithm is applied (Fig. 1). To predict assembly-wise power and burnup distribution, the critical soluble boron concentration, and the pin power peaking factor (PPPF) with core burnup in the PWR using K-infinity and Macro XSs. The OLL trained neural networks can compute core depletion characteristics about 40 times faster than the modern nodal method code. [1]



a) w_{hm} $(m=0,1,\cdots,M;\ h=0,1,\cdots,H)$ = weights for hidden neurons
b) v_h $(h=0,1,\cdots,H)$ = weights for output neuron
c) $f(s)$ = sigmoidal function defined by $(1+e^{-s})^{-1}$

Fig. 1. Three-layer OLL network for prediction of normalized FA power

2.2 Improvement

In the previous research, it was applied to the optimization tool by using OLL networks, but the optimization method did not change, and it was not performed in this paper. The first improvement is that the main neural network is changed Convolutional Neural Network (CNN) from OLL. The input type is also changed. Instead of using a combination of K-infinity and specific macroscopic cross-sections, we use 5 types of the macro cross-section (fast/thermal nu-fission XS, fast/thermal absorption XS, fast to thermal scattering XS) that are used to calculate criticality. The notable improvement except for changing the artificial neural network is predicting the power distribution over the entire cycle as the only BOC macroscopic cross-section rather than predicting the power distribution by using the macro cross section for each burnup steps. And the other one is a reflector and moderator area is included for analyzing periphery area assembly power.

3. Method

3.1 Deep Learning Models

Deep learning is a class of machine learning algorithms that: [2]

- use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.

3.1.1 Supervised learning

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.[3] It infers a function from labeled training data consisting of a set of training examples.[4] In supervised learning, each example is a pair consisting of an input object (typically a vector) and the desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

3.2 Convolution Neural Network

A convolutional neural network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, RELU layer i.e. activation function, pooling layers, fully connected layers, and normalization layers. [5]

Description of the process as a convolution in neural networks is by convention. Mathematically it is a cross-correlation rather than a convolution (although cross-correlation is a related operation). This only has significance for the indices in the matrix, and thus which weights are placed at which index. Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. [6]

Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters. [7] For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters.

3.3 Residual Neural Network

A residual neural network is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections or short-cuts to jump over some layers. [8]

One motivation for skipping overlayers is to avoid the problem of vanishing gradients by reusing activations from a previous layer until the layer next to the current one learns its weights. During training, the weights adapt to mute the previous layer and amplify the layer next to the current. In the simplest case, only the weights for the connection to the next to the current layer is adapted, with no explicit weights for the upstream previous layer. This usually works properly when a single non-linear layer is stepped over, or when the intermediate layers are all linear. If not, then an explicit weight matrix should be learned for the skipped connection. Skipping initially compresses the network into fewer layers, which speeds learning. The network gradually restores the skipped layers as it learns the feature space. During later learning, when all layers are expanded, it stays closer to the manifold and thus learns faster. A neural network without residual parts explores more of the feature space. This makes it more vulnerable to perturbations that cause it to leave the manifold and necessitates extra training data to recover.

3.4 Architecture

The design of convolutional neural networks (CNN) with shortcut-connection which was utilized for the problem is shown in Fig. 2.

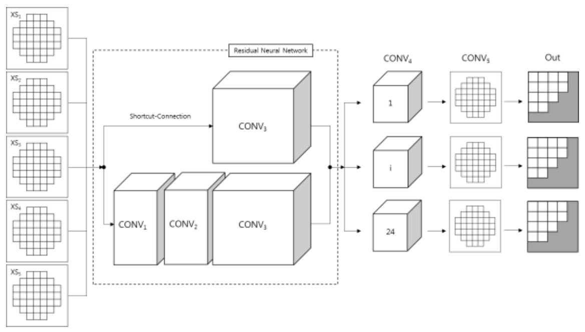


Fig. 2. Convolution neural network architecture with shortcut -connection for power distribution prediction

The 5 types of the macro cross-section are fast/thermal nu-fission XS, fast/thermal absorption XS, fast to thermal scattering XS. The macroscopic XSs used as input are taken from the lattice code calculation (KARMA) which is mainly used for commercial core analysis. It is node-wise (1/4 assembly node), not an assembly-wise. Five macro XS of each layer is represented as binary images of 34 × 34 (the complete input shape is [34×34×5]). A first convolution ($CONV_1$) is performed using a 1×1 filter, same padding, ReLU activation function and 64 channels (the output shape is [34×34×64]). The second convolution ($CONV_2$) has the same properties of ($CONV_1$) except for the size of filter—growing now to 3×3 filter (shape of [34×34×64]). The third convolution ($CONV_3$) has the same properties of ($CONV_1$) except for the number of channels—growing now to 256 (shape of [34×34×256]). There is shortcut-connection that only passed $CONV_3$ (shape of [34×34×256]). After adding two convolution layers, it is divided into 24 burnup step(i=1～24). The fourth convolution ($CONV_4$) is performed using a 2×2 filter, 2×2 strides, same padding, ReLU activation function and 128 channels (the output shape is [17×17×128]). The fifth convolution ($CONV_5$) is performed using a 1×1 filter, ReLU activation function and 1 channel (the output shape is [15×15×1]). After the fifth convolution, quadrant average values are generated using symmetry.

the resulting data (of shape [8×8×1]) is flattened to a single vector (of [64] elements).

The training outputs are taken from the 3-D core calculation (ASTRA). We utilized the assembly-wise power distributions at 24 core burnup steps (0, 50, 150, 500, 1000 to 20000 increased by 1000, MWD/MTU). The 2-D power distributions are quadrant symmetric values. It also includes the moderator area with zero-power. The reason for including it is that the area with zero-power also has spatial significance when estimating the distribution. A 3 × 3 filter is applied. This is because, when calculating the power distribution of the specific fuel assembly, it is most important what fuel assemblies come around. The larger the filter size, the closer it is to fully connected neural network, which results in unnecessary weight production and longer learning time. Normally, when analyzing images with CNN, the information is compressed and downsized. However, in the case of analyzing the loading pattern, the size was not reduced because the importance of the information was all the same for each location.

4.Result

4.1 Loading Pattern Random Generation

The core was decided by OPR1000 (177 fuel assemblies) for this supervised learning. The Feed Assembly uses gadolinia as a burnable absorption rod. Four different types of burnable absorbers differing in the number and position were randomly selected for each location. In addition, the following are assumed:

- Based on Low Leakage Loading Pattern
- Random shuffle without periphery location
- No. of Feed assembly is fixed (69 Feed)
- Octant Symmetry

9158 loading patterns were produced using the assumed conditions and 3-D core calculation code (ASTRA).

4.2 Supervised Learning

4.2.1 Optimized Layer by Layer (OLL) vs. Convolution Neural Network (CNN)

Using the 9158 loading patterns (8700 were used as actual learning data, and 458 were used for validation), we compared the OLL method and the CNN method. In order to compare only the differences according to the network, the number of parameters used in the network is made equal. In the case of OLL, k-inf and two macro cross-sections are used as in the previous study, and in CNN, five macro cross-sections are used as described above. The mean squared error is set to loss and learned to minimize it. The learning time is the same as the one hour, and at the completion of the learning, the loss can confirm that CNN is lower than OLL.

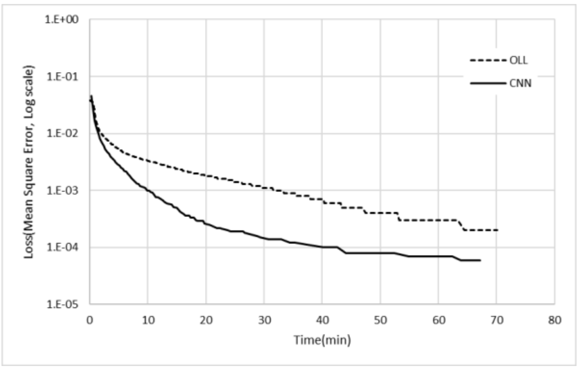


Fig. 3. Loss according to Learning time about OLL vs. CNN for BOC only

The predicted results for the 458 validation loading patterns are shown in the following table. There is no significant difference in the mean error, but the maximum error is almost twice the difference. If we calculate the fraction of the assemblies with over than specific absolute error, we can see that CNN accurately predicts the assembly-wise power distribution rather than OLL.

Table I: Power distribution prediction error of OLL and CNN

Network Type	e_{avg}[a]	e_{max}[b]	Frac. with e[c] > 3%	Frac. With e[c] > 5%
OLL	1.05	11.92	4.8	0.4
CNN	0.44	4.23	0.0	0.0

a= average absolute error (%)
b= maximum absolute error (%)
c= Fraction of the assemblies with absolute error (%, based on 13282 assemblies)

4.2.2 Convolution Neural Network for the whole cycle

9158 loading patterns and architecture (Fig. 2) are used. The mean squared error is also set to the loss. The loss according to the time is as follows. It takes about two minutes to learn once (epoch). It learned about 700 times to reach the desired loss.

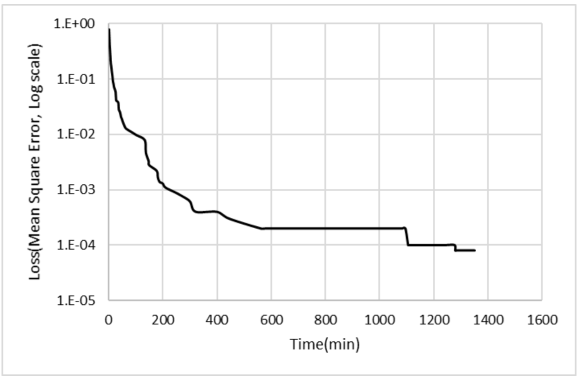


Fig. 4. Loss according to Learning time about CNN for the whole cycle

The graph (Fig. 4) shows enough learning, and the maximum errors by each location of 458 loading patterns for validation are as follows.

Table II: Power distribution prediction error of CNN for the whole cycle

	e_{avg}[a]	e_{max}[b]	Frac. with $e^c > 1\%$
CNN for whole cycle	0.13	1.69	0.005

a= average absolute error (%)
b= maximum absolute error (%)
c= Fraction of the assemblies with absolute error
(%, based on 596544 assemblies)

As a result of the validation, it was confirmed that the assembly with the absolute error exceeding 1% is 0.005% of the total assembly and that it is in good agreement with the 3-D core calculation.

4.3 Verification

Three loading patterns are selected for verification. The first is the optimized loading pattern (equilibrium cycle) that used to a recent cycle of OPR1000 and meets the assumed conditions, the second is that add feed assembly inside, and last is that increased neutron leakages by loading feed assembly at the periphery(outermost) location. The Computation time for each loading pattern is about 0.2 second in CPU(Intel i7-3770 3.40GHz, DDR3 16GB) and 0.05 second in GPU(NVIDIA GeForce GTX 1080 Ti 11GB). The results using the learned convolution neural network are as follows.

Table III: Power distribution prediction error of three verification models

	e_{avg}[a]	e_{max}[b]	Frac. with e^c > 3%	Frac. With e^c > 5%
CNN for the whole cycle				
Eq.	0.38	3.81	0.7	0.0
73Feed	1.30	2.80	0.0	0.0
73Feed + H.L.[d]	2.38	22.23	17.9	10.6
CNN for BOC only				
Eq.	0.48	1.96	-	-
73Feed	0.62	2.43	-	-
73Feed + H.L.[d]	3.27	17.35	-	-
OLL for BOC only				
Eq.	1.00	2.60	-	-
73Feed	0.83	2.62	-	-
73Feed + H.L.[d]	24.35	69.14	-	-

a= Average absolute error (%)
b= Maximum absolute error (%)
c= Fraction of the assemblies with absolute error
(%, based on 1248 assemblies)
d= High neutron leakage rather than assumed condition

For the first model, the mean and maximum errors were larger than the validation model but well predicted. The error of the model which added feed assembly inside also increased, but CNN predicts the power distribution using the relation between the assemblies, considering that the absolute error exceeding 3% does not occur. As with the third model that we have not seen the learning loading patterns, so we found that a large error occurs at periphery location. But, compared to OLL data, this also demonstrates that CNN predicts the power distribution using the relationships including a reflector and moderator area.

5. Conclusions

Convolutional Neural Networks were applied to predict the assembly-wise power distribution of the whole cycle, and accurate values were obtained in a very short time (less than 0.2 seconds by one loading pattern). It is expected that the assembly-wise maximum pin power distribution, critical boron concentration, and cycle length can be predicted using the same CNN module. This neural network module was developed to provide convenience to those who design the loading pattern. It is to acquire fast and precise data of required design information without 3-D core calculation. In order to improve this, it is necessary to perform reinforcement learning and to develop a module that creates a loading pattern that is equivalent to the actual design with different amounts of feed assembly and neutron leakage.

REFERENCES

[1] C. S. Jang, H. J. Shim, C. H. Kim, Optimization layer by layer networks for in-core fuel management optimization computations in PWRs, Annals of Nuclear Energy Volume 28, Issue 11, July 2001, Pages 1115-1132
[2] Deng, L., Yu, D., Deep Learning: Methods and Applications, Foundations and Trends in Signal Processing, 2014, pages 1-199.
[3] Stuart J. Russell, Peter Norvig, Artificial Intelligence: A Modern Approach, Third Edition, Prentice Hall, 2010.
[4] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar Foundations of Machine Learning, The MIT Press, 2012.
[5] "CS231n Convolutional Neural Networks for Visual Recognition", cs231n.github.io, Retrieved December 13, 2018.
[6] "Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation". Theano Development Team, Retrieved August 31, 2013.
[7] Hamed Habibi Aghdam, Elnaz Jahani Heravi, "Guide to convolutional neural networks: a practical application to traffic-sign detection and classification", Springer, May 2017.
[8] He Kaiming, Zhang Xiangyu, Ren Shaoqing, Sun Jian, "Deep Residual Learning for Image Recognition", December 2015.