# Convolutional Neural Network for BOC 3D Pin Power Prediction

Younduk Nam[a], Jin Young Lee[a], Hyung Jin Shim[b]

*a. KEPCO Nuclear Fuel Co., #242, Deadeok-daero, 989beon-gil, Yuseong-gu, Deajeon, 305-353, Korea*
*b. Seoul National University, Department of Nuclear Engineering, San 56-1, Shilim-Dong, Gwanak-Ku, Seoul, 151-742 South Korea*
**Corresponding author: ydnam@knfc.co.kr*

## 1. Introduction

A state-of-the-art nodal diffusion theory code is accurate and computationally fast enough for pressurized water reactor (PWR) core analysis. However, its computational cost is still a burden for the core loading pattern (LP) optimization which requires over than 10,000 different LP analyses. [1] For a super-fast reactor core analysis, artificial neural networks (ANN) models have been tried for the LP optimizations. [2] [3] Although these ANNs show promising validation-set results, it shows large error in the test-set results. Therefore, we made a better ANN architecture for more generalized BOC 3D pin power peaking factor (PPPF) prediction which utilizes the power of convolutional neural network (CNN).

## 2. Review of previous works

The history of nuclear power plant simulation has been solving transport and/or diffusion equation. Researchers must decide between computational accuracy and time. Even with the advances in computational resources, transport calculations are too computationally and costly for whole 3D core calculation. Therefore, the most widely used code system for 3D core calculation is a nodal diffusion theory code which is a combination of transport and diffusion calculations. However, the system is still too computationally expansive for LP optimization. Therefore, there is a need for a model that is fast with certain accuracy. We will introduce an CNN architecture that best predicts PPPF results of a nodal diffusion theory code.

## 3. Method

Modern PWRs are efficiently designed in terms of economy and safety. Therefore, every structure is simplified in the engineering sense. For example, the arrangements of fuel assemblies in a core are designed into a lattice shape. Because of these characteristics, it is conveniently spatially discreditable. Therefore, a CNN architecture is optimal for this problem. [4]

We have designed our architecture for OPR1000 Type plant. Although it is possible to make an architecture for all Korean Plant Types, (OPR1000 and APR1400) testing only on one plant type is a reasonable scope of this paper. Rest of the assumptions are in the 4.1 Data Generation section.

Before introduce our architecture, we have to define follow keywords:

- Optimization of Architecture: problem specific optimization such as spatial or time series data
- Optimization of Learning: solving problems within ANN and/or CNN such as vanishing gradient and/or overfitting

The finalized architecture is summarized in Figure 1.

- XS INPUT: 3D 4th quadrant node-wise XS Input
- CONV15[a]: identity filter for a node-wise relationship [5]
- BN1234[b]: batchnorm for vanishing gradients (VG) [4]
- CONV2-4[a]: convolutional filter for a spatial relationship[7]
- SE1[b]: Squeeze and Excitation for feature importance [8]
- SHORTCUT [b]: skip connection for VG [5]
- DECONV123 [a]: convolutional filter for pin power dist.
- PIN INPUT [a]: pin power from FFL file (Form Function)
- CONV6 [a]: convolutional for pin power
- MAX PIN OUTPUT [a]: 3D pin power output

a Optimization of Architecture
b Optimization of Learning

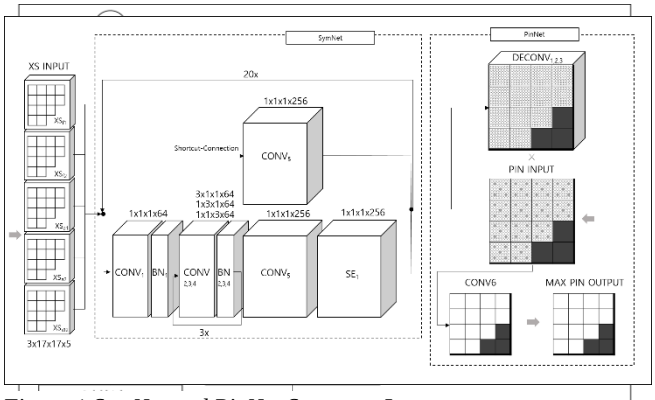


*Figure 1 SymNet and PinNet Summary Layout*

The architecture in Figure 1 looks very similar to residual neural network that is found in image recognition. [5] Since all of the Optimization of Learning layers in ResNet are proven in the residual neural network paper, we will not go over in this paper. Instead we will go over the differences in each layer and the reasons for each newly introduced layer.

For easier understanding of the ANN, layers are like functions and blocks are like modules or classes in

programming. Therefore, the entire ANN model acts like a program.

*3.1 XS INPUT*: Block

Like most ANN architectures, defining the input is the most important part of the Optimization of Architecture process. Therefore, we carefully choose our input as a building block of core simulation: initial node-wise macroscopic cross sections (XS) which are divided by the assembly discontinuity factors (ADF). The input shape of 2D node-wise full core XSs should be 34x34x5. It is 34x34 nodes because there are radially 15 assemblies with 2 reflectors on both sides. In order to save some memory and computational time, we took advantage of the core design: rotational symmetry. By simply taking 4$^{th}$ quadrant of full core and applying rotational padding, we can essentially make quadrant core into a full core. Therefore, our final quadrant input shape is 17x17x5.

For 3D calculation, we need to take account of axial nodes. Conventionally, the axial part of an assembly is divided into 26+2 nodes. Since they are not equally spaced, we need to recalibrate the axial nodes into equally spaces nodes. Upon testing several methods, three points (1/4, 2/4, 3/4) axial nodes worked the best. Finally, the 3D quadrant node-wise XS input shape is 3x17x17x5.

*3.2 SYMMETRY NEURAL NETWORK (SymNet)*: Block

Our architecture called Symmetry Network (SymNet) is a combination of three networks, ResNet, Inception-v3, and SE Network. [5][7][8] The summary of the SymNet is as shown in Figure 2.

There are two main differences between ResNet to SymNet. First, our input is rotationally symmetric. Therefore, we need ROTPADDING1 layers to trick the neural network to think that it is looking at the full core. Second, spatial dimensions must be preserved going from one layer to another. Often in the image recognition problem, pooling layers are introduced to make spatial information into the feature information. Because of this it is invariant to the rotation and translations. [4] However, we value the rotation and translation of all the assemblies. Therefore, we took out all the pooling layers from our network. Each relevant layer is explained in detail in the following sections.

*3.2.1 CONV234: Layer*

CONV234 layer is presented to find a spatial relationship between surrounding assemblies. Instead of 3x3 layer in ResNet, our convolutional layers have two 1x3 and 3x1 layers like Inception-v3, because it shows showed that it is better to implement two instead of one 3x3 layer because it produces a deeper network. [7]

The general knowledge is that it is efficient to build a deeper model with skipping connections. For example, two 3x3 layers are better than one 5x5 layer. Since at the end of our architecture, we need to see the full core (34x34), we need to stack at least 17 3x3 layers (2x17+1). Therefore, we stacked little more layers because it does not hurt the learning (20 stack). [5] The final output shape of CONV234 is 3x17x17x64.

*3.2.2 ROTPADDING1: Layer*

ROTPADDING1 layer is needed for quadrant inputs. Because our input only has 4$^{th}$ quadrant, the rotational symmetry padding must be applied to the beginning of each CONV234 layers. As shown in Figure 3, we apply rotational symmetry by copying white nodes 0, 1, 2 to corresponding gray nodes. The output shape of this padding is 3x19x19x64. After CONV234 layer the dimension goes back to 3x17x17x64. Therefore, we are not expending the dimensions.

*3.3 PinNet: Block*

PinNet block is necessary for pin peaking factor such as Fr (Radial Peaking Factor, 2D) and Fxy (Planar Peaking Factor, 3D like). In a two-step system, this process is called pin power reconstruction. From the calculated pin power distribution, the form function (FF) is simply multiplied to predict core wise pin power distribution. The form function is pin power distribution that was prepared during assembly-wise cross section generation. The following layers are essential layers for PinNet.

*3.3.1 DECONV123: Layer*

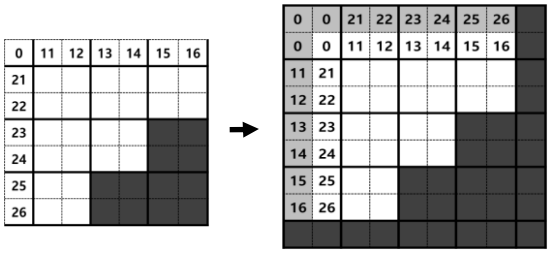


*Figure 3. Rotationally symmetric padding, left is node wise numbering of center and 2D axial nodes and right is how it is copied to the padding.*

DECONV123 layers first expand the dimensions from quadrant nodes (17x17) to quadrant pins (128x128). We perform deconvolution to keep the spatial relationships. The deconvolution also known as convolution-transpose is a widely used concept for dimensional expansion. We use 1x3x3x64 filters with 2 padding with a stride of 2. The output shape from this layer is 3x128x128x64.

*3.3.2 Pin Input: Layer*

Pin Input layer is in the shape of 1x128x128x1. From the assembly-wise burnup distribution, FF is looked up from FFL file from a two-step code system. This input is simply multiplied to the DECONV123 output values. Final output shape will be 3x128x128x64.

*3.3.3 CONV6: Layer*

CONV6 layer is there to find the final assembly-wise PPPF. It is possible to simply perform the max pooling, but for the stable learning, we just replace it with convolution layer with a 3x16x16x1 filter. Final output shape from this layer is 8x8x1 which is the assembly-wise PPPF. Finally, the output is compared to Max Pin Output.

*3.4 Max Pin Output: Block*

Our interest is the maximum value of 3D pin power distribution. Therefore, there were two possible candidates for our output shapes: 1x1x1 global PPPF or 8x8x1 assembly-wise PPPF. In order to predict 1x1x1 global maximum PPPF, we can make a fully connected network at the end. However, it is better to look at all 8x8x1 assemblies and train on them because it is essentially training on 64 more different cases of 1x1x1 global PPPFs. The model will be more generalized and be accurate. Therefore, our final output shape is 8x8x1 instead of 1x1x1.

## 4. Result

*4.1 Data Generation*

In order to verify the validity of the architecture, the target core is selected as the Korean Standard Nuclear Power Plant (OPR1000 Type) with 177 fuel assemblies. The feed assembly uses gadolinia as a burnable poison material. Four different types of burnable absorbers differing in the number and position were randomly positioned. The follows are assumed.

- Base on Low Leakage Loading Pattern
- Random Shuffle excluding the periphery locations
- No. of feed assembly is fixed (69 feeds) including center assembly
- Octant Symmetry

9158 loading patterns were produced using the assumed conditions and 3d core calculation code. Finally, for the test data we have prepared LPs that are generated differently from the trained and validation data. The 800 test LPs are generated so that it is closer to equilibrium LPs.

The computer resources for this test is as follows.
-CPU: Ryzen 1700 Processor.
-GPU: NVIDIA 1060.

*4.2 Architecture Validation*

In order to test the final architecture, we compare four architectures: with or without 3D XS input and FF input. As shown in Table 1, it is not possible to predict accurate PPPF with only 2D XS input. The averaged absolute error for PD is 0.3%. When we validated the Fr and Fxy on the same network, the absolute averaged error was around 1.8% and 3.1%, respectively. Since this is only BOC PPPF, the error can be greater if we expand this to full cycle depletion. Therefore, we need better architecture and inputs for Fr and Fxy.

The reference is 2D XS without FF. By adding just 3D XS, error was almost 1/2 from the reference. However, by doing so, the computational time and memory usage was almost doubled or even tripled from the reference. By adding just FF to the reference, the error was 1/3 from the reference. Finally, by adding both 3D XS and FF, the error was 1/3 from the reference. Therefore, we concluded that FF with 2D is the best choice for our applications. Since this was the validation phase, we need to verify the 2D XS with FF architecture with the test data in the following section.

*Table 1 mean square error for PD Fr Fxy prediction*

| | 2D XS[a] | 3D XS[a] | 2D XS[a] +FF[b] | 3D XS[a] +FF[b] |
|---|---|---|---|---|
| PD | 2 | 0.5 | - | - |
| Fr | 12 | 7 | 4 | 4 |
| Fxy | 61 | 12 | 7 | 6 |

a 2D or 3D XS input were used
b Form Function is used
*all values are divided by 1E-5

4. 3 Architecture Verification

The final architecture was verified with 800 test data. The most promising architectures are 2D XS + FF and 3D XS + FF. We validated the model (trained NN architecture) with 800 non-trained LPs but still within the distribution trained data. (Average Fxy 2.224, validation-set) Then we tested the model with 800 non-trained LPs that are not within the distribution of the trained data. (Average Fxy 1.599, test-set) All absolute percent errors in Table 2 are calculated with the following equations.

$$e = \text{abs}(\max(p_{pre}) - \max(p_{real}))/\max(p_{real})) * 100$$

where

$$p_{pre} = Predicted\ assembly\ wise\ PPPF$$
$$p_{real} = Real\ assembly\ wise\ PPPF$$

The reason why we consider only the maximum power from all assemblies is because only global PPPF is used in the core design. The rest of the assembly PPPF data are only for training as we mentioned in Max Pin Output Section.

As we can see from Table 2, when looking at the averaged absolute percent error, 3D shows better performance over 2D counterparts. However, when looking at the maximum absolute percent error, 2D shows better performance. This is more prominent on the test-set. We are currently not sure why this happening, but it is maybe from overfitting to our 3D data. The axial node spacing methods as described in Input Block Section must be checked in the later paper. However, both implementations are promising.

The computational time for 2D XS input and 3D XS input with FFs are respectively took around 0.025 and 0.040 seconds with the single GPU and respectively took around 0.055 and 0.161 seconds with the single CPU. Considering the computational time and the accuracy, 2D XS input with FF input is the best choice for our problem.

*Table 2 validation and test for Fxy prediction*

| | $err_{avg}$[a] | $err_{std}$[b] | $err_{max}$[c] |
|---|---|---|---|
| **OLL*** | | | |
| 2D validation-set | 1.60 | - | 11.26 |
| **SymNet, PinNet** | | | |
| 2D validation-set | 0.223 | 0.179 | 0.978 |
| 3D validation-set | 0.204 | 0.165 | 1.011 |
| 2D test-set | 1.104 | 0.991 | 4.240 |
| 3D test-set | 0.700 | 0.660 | 7.463 |

* OLL uses validation-set for testing (same distribution) [2]
a=average absolute percent error (%)
b=standard deviation of absolute percent error (%)
c=maximum absolute percent error (from 800 data)

## CONCLUSION

SymNet which is a variant of ResNet and PinNet were applied to 3D PPPF prediction. From the previous architecture, we have enhanced the capabilities of our architecture to better predict 3D PPPF. The results show that it can predict 3D BOC PPPF less than 1% error. Soon, we will expand this BOC 3D PPPF to full cycle depletion 3D PPPF. Moreover, in order to improve the performance of the final full cycle depletion model, data collection must be improved. Therefore, we will soon incorporate “reinforcement learning”. Finally, we have showed that 2D XSs and with FF architecture is computationally better in predicting 3D BOC PPPF with little to non-compromise in the accuracy.

## REFERENCES


[1] Tong Kyu Park, Han Gyu Joo, Chang Hyo Kim, Hyun Chul Lee, Multiobjective Loading Pattern Optimization by Simulated Annealing Employing Discontinuous Penalty Function and Screening Techinique , Nuclear Science and Engineering: 162, 1340147 (2009).
[2] 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
[3] Akio Yamamoto, Application of Neural Network for Loading Pattern Screening of In-Core Optimization Calculations, Nuclear Technology, 144:1, 63-75, DOI: 10.13182/NT03-A3429, (2003).
[4] Krizhevsky, Alex & Sutskever, Ilya & E. Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.
[5] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015-12-10). "Deep Residual Learning for Image Recognition". arXiv:1512.03385 [cs.CV].
[6] Sergey loffe, Christian Szegedy (2015-02-11). "Batch Normalization : Accerlerating Deep Network Training by Reducing Internal Covariate Shift". arXiv:1502.03167 [cs.LG].
[7] Christian Szegedy, Vincent Vanhoucke, Sergey loffe, Jonathon Shlens, Zbigniew Wojna (2015-12-02). "Rethinking the inception Architecture for Computer Vision". arXiv:1512.00567 [cs.CV].
[8] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu (2017-09-05). "Squeeze-and-Excitation Networks". arXiv:1709.01507 [cs.CV].