# A DEEP LEARNING BASED SURROGATE MODEL FOR ESTIMATING THE FLUX AND POWER DISTRIBUTION SOLVED BY DIFFUSION EQUATION

**Qian Zhang[1*], Jinchao Zhang[1], Liang Liang[1], Zhuo Li[2] and Tengfei Zhang[3]**

[1]Harbin Engineering University
145 Nantong St, Harbin, Heilongjiang, 150001, China

[2]Sino-French Institute of Nuclear Engineering and Technology, Sun Yat-Sen University
No.2 Daxue Road, Zhuhai, 519082, China

[3]Shanghai Jiaotong University
800 Dongchuan Rd. Minhang District, Shanghai, 200240, China.

qianzhang@hrbeu.edu.cn,
13935397912@hrbeu.edu.cn, liangliang_ls@ hrbeu.edu.cn,
lizh333@mail.sysu.edu.cn, zhangtengfei@sjtu.edu.cn

## ABSTRACT

A deep learning based surrogate model is proposed for replacing the conventional diffusion equation solver and predicting the flux and power distribution of the reactor core. Using the training data generated by the conventional diffusion equation solver, a special designed convolutional neural network inspired by the FCN (Fully Convolutional Network) is trained under the deep learning platform TensorFlow. Numerical results show that the deep learning based surrogate model is effective for estimating the flux and power distribution calculated by the diffusion method, which means it can be used for replacing the conventional diffusion equation solver with high efficiency boost.

KEYWORDS: deep learning, surrogate model, convolutional neural network, diffusion

## 1. INTRODUCTION

The deep learning technology is widely used in the area of image identification and natural language processing. In recent years, the deep learning has been investigated in the area of numerical simulation. Several publications proposed studies on solving partial differential equation [1,2] and in the area of mechanics [3] and fluid dynamics [4], the deep learning approach was also discussed. The deep learning was also applied in the area of nuclear engineering with fault diagnosis and safety analysis [5].

The deep learning study on the topic of reactor physics has not been found before 2019. However, there are many studies on the surrogate model based on artificial neural network used for fuel management in 1990s [6, 7]. The basic idea is to predict the core parameters, such as $k_{eff}$, power peaking factor, with very high efficiency, replacing the regular diffusion solver in the industrial fuel management code. However, the conventional artificial neural network has the following limitations:
1. The accuracy of the prediction by the artificial neural network is limited due to small amount of training data;

2. The scale of the input vector is limited to octant-core assembly and the large-scale whole-core level prediction is difficult.
3. The surrogate model is trained based on limited types of fuel assemblies with specific enrichment and burn up. The surrogate model cannot be used for generalized purpose.

The conventional artificial neural network was not widely used in the fuel management code due to the above limitations. In addition, with the developing of CPU performance and the parallel technology, the diffusion equation solver is already very fast. Therefore the advantage brought by the conventional artificial neural network is not remarkable. However, in recent years, the deep learning approach featured by the deep convolutional neural network achieved great success in image identification and natural language processing due to improvement on accuracy. In addition, modern platform of deep learning training, such as the TensorFlow working on GPU, is available for large-scale training data.

With deep learning approach, the above limitations of the conventional artificial neural network can be broken through. In addition, although the two group diffusion equation solver is very fast for current computer, high-fidelity calculation with pin-by-pin configuration and with more energy group is still a challenge for modern computer in the scenario of fuel management optimization and real-time simulation of the core. To sum up, the deep learning approach has the potential to become an effective surrogate model for high-fidelity core calculation.

In 2019, the author of the paper proposed a deep learning model to predict the $k_{eff}$ of the core problem calculated by the diffusion method with higher accuracy than the conventional artificial neural network [8]. In addition, the model is based on the cross section input of each node, which means that the surrogate model is for generable purpose of core calculation. The original deep learning model is transformed from the VGG16 model [9]. This model is designed for the training task of outputting single parameter like $k_{eff}$ and power peaking factor. However, it cannot be used in predicting the distributed parameters in the core, such as the flux and power distribution. In this study, a new deep learning based surrogate model transformed from the FCN [10] is proposed for predicting distributed parameters of the core. Numerical results show that the deep learning based surrogate model is effective for estimating the flux and power distribution calculated by the diffusion code and the accuracy also superior to the conventional artificial neural network. The efficiency of the model is much higher than the conventional diffusion solving.

## 2. MODELS AND METHODOLOGY

### 2.1. General framework

The general framework of this study follows a typical machine learning procedure. First, for a specific reactor core geometry, the input-output data structure of the diffusion code is set. For a typical diffusion calculation code, the input is the few-group cross section and the output is the flux and power distribution. Supposing that the reactor is within a specific range of state, which means that the cross section of each node is within a specific range. Large amount of reactor core problem are generated by stochastically sampling the cross section of each node. The output of each reactor core problem is provided by an in-house 2-D multi-group diffusion code. The input-output data are collected as the training and validating dataset. The deep learning model with FCN is designed to fit the geometry of the core problem. The deep learning model, which will be introduced in detail in Section 2.3, follows a typical procedure of the neural network, including the loss function, the optimizer and the weights. Based on the TensorFlow platform, the deep learning model is trained with the input-output data. Using the training data provided by the 2-group diffusion calculation, the internal mechanism of the neutronic problem will be learned by the CNN. The general framework is given in Fig. 1.

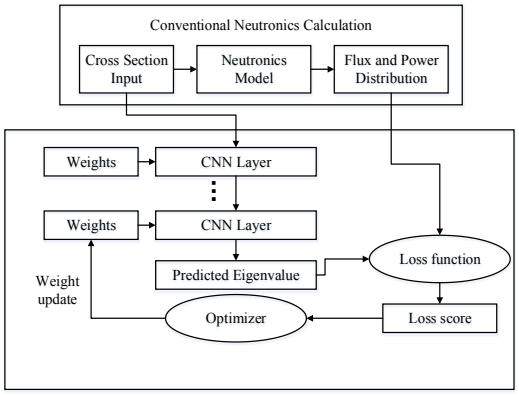



**Figure 1. The general framework of the deep learning model applied to neutronics**

### 2.2. The simplified reactor core problem

A quarter of the simplified 2-D reactor core model is shown in Fig.2. It contains fuel assembly and reflector. The size of each node is 10cm×10cm. The reflective boundary condition is used. The two-group homogenized cross section of the fuel and reflector is sampled stochastically from a given range. The cross section range is given in Table I.

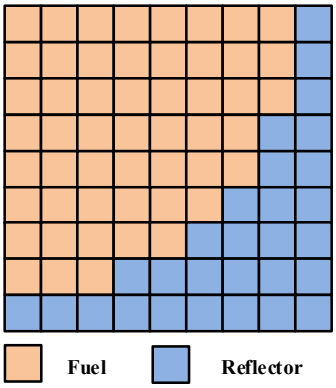



**Figure 2. Geometry of the simplified core.**

**Table I. The cross section configuration of the fuel and reflector.**

| Material Type | g | $D_g$ | $\Sigma_{ag}$ | $\nu\Sigma_{fg}$ | $\Sigma_{s12}$ |
|---|---|---|---|---|---|
| fuel | 1 | (1.120-1.630) | (0.008-0.012) | (0.004-0.005) | (0.011-0.026) |
| | 2 | (0.200-0.460) | (0.050-0.100) | (0.087-0.135) | |
| reflector | 1 | (2.000-2.000) | (0.000-0.000) | (0.000-0.000) | (0.040-0.040) |
| | 2 | (0.300-0.300) | (0.001-0.001) | (0.000-0.000) | |

$\chi_1 = 1.0 \quad \chi_2 = 0.0 \quad B_z^2 = 8.0 \times 10^{-5}$

In Table I, the cross section range of the fuel assembly is extracted from the life-time history of the operation PWR. It is noticed that for the reflector, the cross section is fixed. Massive reactor core input data is generated by stochastically sampling the cross section of each node within the range.

An in-house 2-D diffusion code based on the NEM (Nodal Expansion Method) is used for calculating the power and flux distribution. An in-house distinct wrapper code is used for generating the input card for the diffusion code and executing the task of solving parallelly.

### 2.3. Specification of the FCN

In the previous study on the prediction of $k_{eff}$ of 2-D reactor core. The well-known VGG-16 structure in the deep learning area is selected as the surrogate model. The essential idea is to directly apply the image identification function of the VGG-16 structure to the regression task of the core problem. First, the input data of cross section of the core is stored in the format of images. In the original publication of VGG-16, the training data and the validation data are the set of image files with 224×224 pixels. For each pixels, there are three color channels (RGB) and for each channel, 8 bit color values are from 0 to 255. For the core problem shown in Fig.2, the cross section input is treated similar to the image file. It can be seen as a special image file with 9×9 pixels and 7 channels per pixel. The original VGG-16 can be modified to fit in this type of images. The previous results show that the modified VGG-16 structure works very well in the prediction of $k_{eff}$.

However, the VGG-16 cannot be used for the prediction of distributed parameters. The FCN (Fully Convolutional Network) is another deep learning structure proposed for semantic segmentation. The illustration of FCN is shown in Fig. 3. The FCN uses a few up-sample technique and the deconvolution operation to achieve a pixel-to-pixel mapping. The details of the FCN can be found in Ref.10. Thus the deep learning model for the power and flux distribution of the core is inspired by the FCN to achieve the node-to-node mapping.

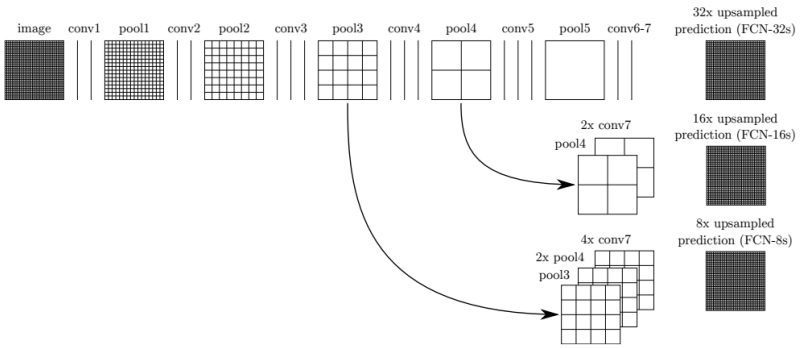



**Figure 3. Illustration of the FCN structure.**

In this study, a modified FCN structure is proposed to fit the 9×9 core size. The details of the FCN is shown in the Appendix A. A conventional neural network called MLP (Multi-Layer Perceptron), which was usually used in the prediction of core parameters in 1990s, is also given in Appendix A. The conventional MLP will be used as a comparison to prove that the performance of deep learning model is superior.

### 2.3. Specification of the training

For the training data, total 0.6 million samples of core problems are generated parallelly on workstation. Uniform distribution is used for sample generation and there is no correlation among each cross sections. For the in-house NEM diffusion code, no acceleration technique is used and 4mecshes are adopted for each code. The input cross section is feed to the deep learning model as it is while the power and flux distribution is normalized by setting the maximum power and flux to 1. 0.3 million samples are used as

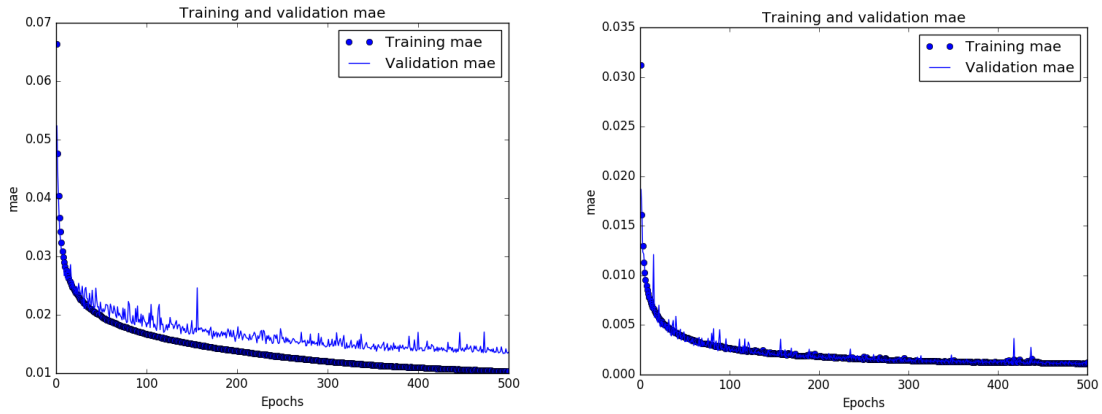
the training data for the deep learning model and the MLP while the other 0.3 million samples are used as the validation data. For both FCN and MLP, the ReLU (Rectifier Linear Unit) is selected for activation function, the MSE (Mean Squared Error) is selected as the loss function and the Adam optimizer is adopted. The learning rate is set to 0.0001.

The deep learning platform TensorFlow is used in this study. Upon TensorFlow, the Keras framework is used for establishing a convenient environment for designing and testing the neural network. The numerical experiment is performed on the Linux system with 3.30GHz Intel Core i9-7900X and Nvidia GeForce GTX 2080ti. In practice, the GPU version of TensorFlow is used for massive data training. 500 epochs of training is performed.

## 3. NUMERICAL RESULTS

### 2.1 Accuracy performance

First, the evolution of the MAE (Mean Absolute Error) of the power prediction of all the nodes in both training data and validation data with training epoch is analyzed. Fig. 4 gives the performance of MLP and FCN. For MLP, it is very obvious that although the MAE of the training data can reaches 0.01 at 500 epochs, the MAE of the validation data stays at 0.014 after 400 epochs. In the area of neural networks, the phenomenal is called overfitting. Overfitting indicates that the learning model is either too shallow or too deep for the learning task. However, the performance of the FCN is obviously better than the MLPS. MAE for both training data and the validation data are below 0.003. In addition, the overfitting does not happen for FCN.




**a) MLP** **b)FCN**

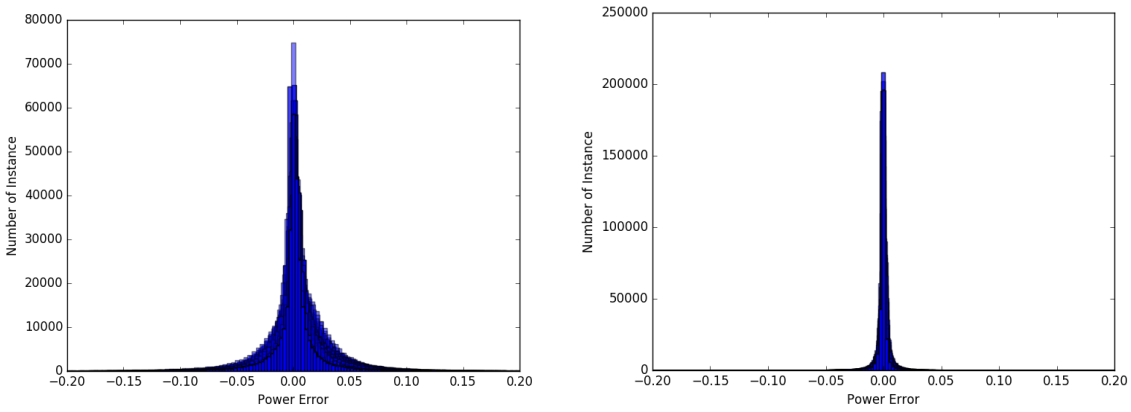
**Figure 4. The evolution of MAE with training of two models.**

Figure 5 shows the absolute error distribution of all the nodes of the core problems in validation data. For FCN, it can be seen that most of the prediction error are below 0.02 while the result of MLP is not as good as that of FCN.

The results of Figure 4 and Figure 5 are similar to the previous study of applying VGG-16 to $k_{eff}$ prediction. The deep learning approach also outperforms the conventional MLP in the previous study. The comparison lead to the confidence that the convolution operation and the maxpooling operation of multi-layers in the VGG-16 and the FCN improve the accuracy of the machine learning. The internal machinism is related to the performance of deep learning model on image identification, that the deep learning model can capture the in-depth feature of the neutronic problem. In addtion, in the structure of VGG-16 and

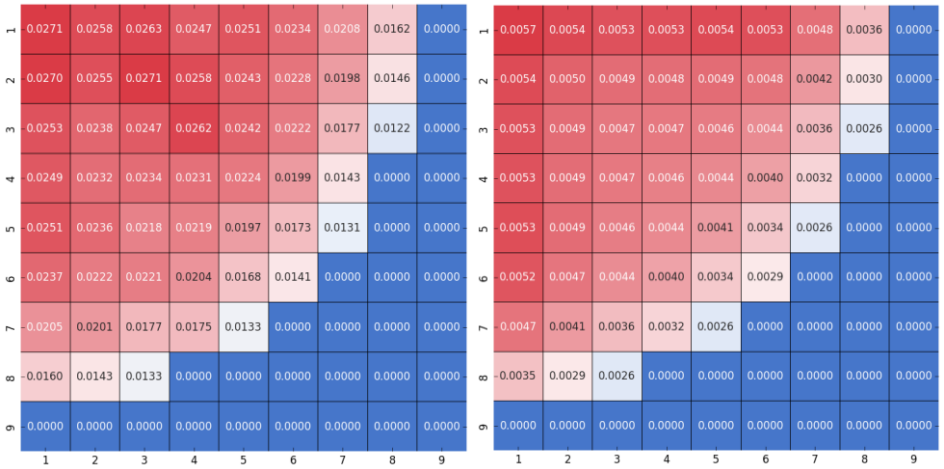
FCN, a 3×3 convolution operation is applied for each layer and this convolution operation matches the local interaction between adjacent fuel assemblies.

Figure 6 compares the mean absolute error for each node in the view of core map. It is shown that the performance of the FCN is better and the error of MLP is five times larger than that of FCN. The results also indicate that in the central region of the core, the predction error is larger in both MLP and FCN. Since the power value in the center has the larger probability to be larger that the periphery of the core, it is speculated that the performance of the deep learning or the conventional neural network have relationship to the absolute value of the output.




**a) MLP** **b) FCN**

**Figure 5. The absolute error distribution of the validation data for FCN and MPL**

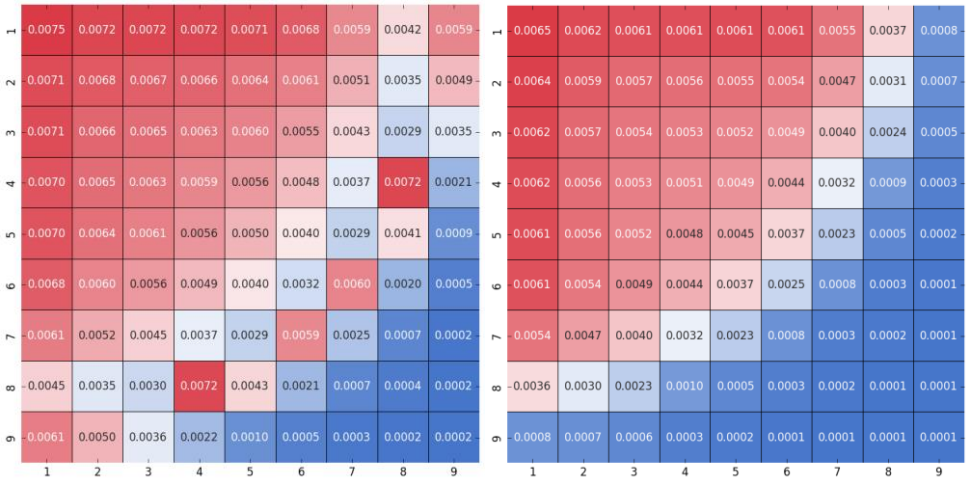



**a) MLP** **b) FCN**

**Figure 6. The MAE for each node in the core map**

Figure 7 shows the performance of the FCN on flux prediction. Using the in-house diffusion code to output the flux in fast group and thermal group, the training data and the validation data are regenerated. The FCN model is trained with the flux distribution. It is noticed that the specification of the flux prediction model training is the same as the procedure for power distribution. Results show that the FCN

model is also capable of predicting the flux distribution. The error of flux prediction is larger for the thermal group. The reason is similar to power distribution that the error is related to the absolute value of the flux and the value of the thermal flux have larger probability to be larger than the fast flux.




**a) Thermal Group** **b) Fast Group**

**Figure 7. The MAE of flux for each node in the core map**

### 2.1 Discussion on efficiency

In the process of evaluating the accuracy of the deep learning model, the efficiency of the model can also be tested. For the regular diffusion calculation of the core based on NEM, due to the four meshes in each node, the core problem calculations takes a few seconds. It takes a few hours to generate the 0.3 million samples of training data on the parallel workstation. However, for the deep learning model, evaluating the results of the 0.3 million samples in the validation data on GPU takes only 26s. In other words, 0.3 million calculation tasks of calculating the power and flux distribution can be done in 26s, leading to the calculation time of 0.000086s for each problem. The more important fact is that the efficiency of deep learning only depends on the scale of core problem and the internal layers of the network and it does not depend on the method used for data generation. Which means that one can even generate the power distribution for the core in Fig. 2 using transport equation solver or even Monte Carlo method, but the efficiency of the deep learning model keeps the same.
However, the cost of generating the FCN deep learning model is high. Not only large amount of training data is necessary, the training process done on the TensorFlow-GPU platform takes 12 hours on the 2080ti, which is one of the top GPUs. In addition, the deep learning model in this study cannot be generalized to other geometries or cross section ranges. This is the limitation of the deep learning model in this study.

## 4. CONCLUSIONS

In this study, the deep learning approach is tested on developing a surrogate model which potentially replaces the conventional neutronic calculations. Numerical results show that the FCN model proposed in this study is capable of predicting the power and flux distribution for a specific core under specific range of operation states. The efficiency of the deep learning is much higher than the conventional neutronic calculation. Although it has some limitations, it can be potential used in fuel management code and real-time core simulation for rapid neutronic calculations.

## ACKNOWLEDGMENTS

This work is supported by the fund provided by the following sources: Heilongjiang Province Science Foundation for Youths [QC2018003]; Fundamental Research Funds for the Central Universities [grant numbers GK2150260178]

## REFERENCES

1. J. Sirignano and S. Konstantinos, "DGM: A deep learning algorithm for solving partial differential equations." *Journal of Computational Physics* **375**, pp. 1339-1364 (2018).
2. J. Han, A. Jentzen and E. Weinan, "Solving high-dimensional partial differential equations using deep learning," *Proceedings of the National Academy of Sciences*, ***115***(34), pp.8505-8510 (2018).
3. D. Finol, L. Yan, M. Vijay and S. Ankit, "Deep convolutional neural networks for eigenvalue problems in mechanics," *International Journal for Numerical Methods in Engineering*, **118**(5), pp 258-275 (2019).
4. J. Ling, K. Andrew and T. Jeremy, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *Journal of Fluid Mechanics*, **807**, pp 155-166 (2016).
5. A. Saeed, et al. "Novel fault diagnosis scheme utilizing deep learning networks," *Progress in Nuclear Energy*, **118**,103066 (2020).
6. H. G. Kim, H. C. Soon, and H. L. Byung, "Pressurized water reactor core parameter prediction using an artificial neural network," *Nuclear Science and Engineering,***113**(1), pp 70-76 (1993).
7. M. G. Lysenko, H. I. Wong and G. I. Maldonado. "Neural network and perturbation theory hybrid models for eigenvalue prediction," *Nuclear science and engineering,* **132**(1), pp 78-89 (1999).
8. Q. Zhang, A deep learning model for solving the eigenvalue of the diffusion problem of 2-D reactor core, Proceedings of the Reactor Physics Asia 2019, Osaka, Japan (2019).
9. K. Simonyan and Z. Andrew, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:**1409**,* pp 1556 (2014).
10. J. Long, E. Shelhamer and T. Darrell. *Fully convolutional networks for semantic segmentation,* pp. 3431-3440, In Proceedings of the IEEE conference on computer vision and pattern recognition (2015).

## APPENDIX A

The Structure of FCN printed by Keras:

```
Layer (type)                 Output Shape              Param #     Connected to
input_1 (InputLayer)         (None, 9, 9, 1)           0
conv2d_1 (Conv2D)            (None, 9, 9, 64)          640         input_1[0][0]
conv2d_2 (Conv2D)            (None, 9, 9, 64)          36928       conv2d_1[0][0]
max_pooling2d_1 (MaxPooling2D) (None, 4, 4, 64)        0           conv2d_2[0][0]
conv2d_3 (Conv2D)            (None, 4, 4, 128)         73856
max_pooling2d_1[0][0]
conv2d_4 (Conv2D)            (None, 4, 4, 128)         147584      conv2d_3[0][0]
max_pooling2d_2 (MaxPooling2D) (None, 2, 2, 128)       0           conv2d_4[0][0]
conv2d_5 (Conv2D)            (None, 2, 2, 256)         295168
max_pooling2d_2[0][0]
conv2d_6 (Conv2D)            (None, 2, 2, 256)         590080      conv2d_5[0][0]
max_pooling2d_3 (MaxPooling2D) (None, 1, 1, 256)       0           conv2d_6[0][0]
up_sampling2d_1 (UpSampling2D) (None, 2, 2, 256)       0
max_pooling2d_3[0][0]
conv2d_7 (Conv2D)            (None, 2, 2, 128)         131200
up_sampling2d_1[0][0]
concatenate_1 (Concatenate)  (None, 2, 2, 384)         0           conv2d_6[0][0]
                                                                   conv2d_7[0][0]
conv2d_8 (Conv2D)            (None, 2, 2, 128)         442496      concatenate_1[0][0]
conv2d_9 (Conv2D)            (None, 2, 2, 128)         147584      conv2d_8[0][0]
up_sampling2d_2 (UpSampling2D) (None, 4, 4, 128)       0           conv2d_9[0][0]
conv2d_10 (Conv2D)           (None, 4, 4, 64)          32832
up_sampling2d_2[0][0]
concatenate_2 (Concatenate)  (None, 4, 4, 192)         0           conv2d_4[0][0]
                                                                   conv2d_10[0][0]
conv2d_11 (Conv2D)           (None, 4, 4, 64)          110656      concatenate_2[0][0]
conv2d_12 (Conv2D)           (None, 4, 4, 64)          36928       conv2d_11[0][0]
up_sampling2d_3 (UpSampling2D) (None, 8, 8, 64)        0           conv2d_12[0][0]
conv2d_13 (Conv2D)           (None, 8, 8, 32)          8224
up_sampling2d_3[0][0]
conv2d_transpose_1 (Conv2DTrans (None, 9, 9, 32)       4128        conv2d_13[0][0]
concatenate_3 (Concatenate)  (None, 9, 9, 96)          0           conv2d_2[0][0]
                                                                   conv2d_transpose_1[0][0]
conv2d_14 (Conv2D)           (None, 9, 9, 32)          27680       concatenate_3[0][0]
conv2d_15 (Conv2D)           (None, 9, 9, 32)          9248        conv2d_14[0][0]
conv2d_16 (Conv2D)           (None, 9, 9, 16)          4624        conv2d_15[0][0]
conv2d_17 (Conv2D)           (None, 9, 9, 2)           290         conv2d_16[0][0]
conv2d_18 (Conv2D)           (None, 9, 9, 1)           3           conv2d_17[0][0]
Total params: 2,100,149
Trainable params: 2,100,149
Non-trainable params: 0
```

The Structure of FCN printed by Keras:

```
Layer (type)                 Output Shape              Param #
dense_1 (Dense)              (None, 1024)              83968
dense_2 (Dense)              (None, 512)               524800
dense_3 (Dense)              (None, 256)               131328
dense_4 (Dense)              (None, 128)               32896
dense_5 (Dense)              (None, 81)                10449
Total params: 783,441
Trainable params: 783,441
```