Cumhuriyet Sci. J., 43(4) (2022) 726-738
DOI: https://doi.org/10.17776/csj.1163514

Cumhuriyet Science Journal

| csj.cumhuriyet.edu.tr | Founded: 2002 ISSN: 2587-2680 e-ISSN: 2587-246X Publisher: Sivas Cumhuriyet University

Machine Learning Applications to the One-speed Neutron Transport Problems

Recep Gökhan Türeci [1,a,*]

[1]*Kırıkkale Vocational School, Kırıkkale University, Kırıkkale, Türkiye.*
**Corresponding author*

Research Article

History
Received: 17/08/2022
Accepted: 26/10/2022

ABSTRACT

Machine learning is a branch of artificial intelligence and computer science. The purpose of machine learning is to predict new data by using the existing data. In this study, two different machine learning methods which are Polynomial Regression (PR) and Artificial Neural Network (ANN) are applied to the neutron transport problems which are albedo problem, the Milne problem, and the criticality problem. ANN applications contain two different activation functions, Leaky Relu and Elu. The training data set is calculated by using the H_N method. PR and ANN results are compared with the literature data. The study is only based on the existing data; therefore, the study could be thought only data mining on the one-speed neutron transport problems for isotropic scattering.

Keywords: Polynomial Regression, Artificial Neural Network, Leaky Relu activation function, Elu activation functions, Machine Learning.

[a] *tureci@gmail.com* *https://orcid.org/0000-0001-6309-6300*

Introduction

The neutron distribution is essential in nuclear reactor operations. Although the diffusion theory results are used in the nuclear reactor theory, the solution of the neutron transport equation is also important. However, the neutron transport equation has seven independent variables. Therefore, reasonable approximations can be used, and thus the one-speed, homogeneous medium and plane geometry neutron transport equation can be written. All neutrons have the same energy in the one-speed approximation. The secondary neutron number, c, is a constant in a homogeneous medium. There are two variables, which are spatial and angular variables, in the plane geometry approximation.

The one-speed, homogeneous medium and plane geometry neutron transport equation is given by

$$\mu\frac{\partial\Psi(x,\mu)}{\partial x}+\Psi(x,\mu)=\frac{c}{2}\int_{-1}^{1}\Psi(x,\mu')\,d\mu' \tag{1}$$

where x and μ correspond to the spatial and the cosine of direction respectively. $\Psi(x,\mu)$ is the neutron distribution at x point and μ direction, and c corresponds to the secondary neutron number. The solution of Eq. (1) has been studied with different numerical methods such as the S_N method [1], P_N and DP_N [2], and semi-numerical methods such as the Case method [3,4], the C_N method [5], the F_N method [6] and H_N method [7].

Machine Learning is the research topic that computer programs can learn from data [8]. After the learning process, the new data can be predicted by the learned programs. Especially as a result of the discovery of quantum mechanics and the breakthroughs in technology and computer systems, data scientists claim that we have a large pile of data that should be analysed. Machine learning studies are important today in terms of analysing this large data pile and creating smart systems that can decide on its own.

Machine learning applications are now an issue that is studied in neutron transport and reactor calculations like many fields with different applications. Chen *et.al.* [9] investigated kinetic models in linear transport theory by deep neural network. The mathematical structure of ANN is well defined in this reference. So we are not interested in the details of the mathematical background of ANN here. We are interested in only application of ANN to data. Whewell and McClarren [10] investigated the data reduction for neutron scattering and fission sources by DJINN (Deep Jointly Informed Neural Networks). The data requirements are reduced by 94 % of the original data according to this study. Xie *et al.* [11] developed ANN to solve neutron diffusion problems. They investigated two different approach which are boundary dependent method and boundary independent method. Zolfaghari *et al.* [12] searched the thermalization devises, which include collimators and moderators, by multilayer perceptron neural network. Chen *et al.* [13] investigated neutron and X-ray scatterings by machine learning.

In this study, we are not interested in solving any equation. We are interested in only data. Therefore, our study can be considered as data mining for one-speed neutron transport theory. The data required for machine learning is calculated by H_N method. The data is the training data for the one-speed neutron transport problems. These problems are half-space albedo, the Milne problem, and the criticality problem respectively. The albedo is defined as the ratio between the net outgoing and the net incoming neutron fluxes over a surface. The Milne problem deals with the finding of the extrapolation length which is the point where the flux is zero. The criticality problem is studied to find the criticality equation. This equation gives the relation between the secondary neutron number and the reactor thickness. These problems data, which are calculated with HN method, are examined with polynomial regression (PR) and the artificial neural network (ANN) in this study.

PR algorithm is run with the least squares method. The aim of PR is to find a regression relation which provides the training data set. To apply PR, numpy [14], scipy [15] and scikit-learn (sklearn) [16] modules in Python can be used to data. ANN algorithm is different from PR. ANN is an improved algorithm of the Logistic regression (LR) algorithm which gives only two different results, true or false. Moreover, ANN includes some hyperparameters which are dependent to the data set and the programmer. To determine these hyperparameters is based on experience over studying the data. To apply ANN, keras [17] and tensorflow [18] modules in Python can be used to data. ANN hyperparameters are given in the Table, but particularly two different activation functions, which are Leaky Relu (Leaky Rectified Linear Unit) [19] and Elu (Exponential Linear Unit) [20] activation functions, are studied in this study.

Material and Methods

The Polynomial Regression (PR) and the Results with PR

The PR is a fit application between (x, y), which are independent variable and the dependent variable, respectively. The polyfit method [14] in numpy module was used to the imported data for PR in this study. The polyfit method is based on the least squares method. The squares of the residuals, which are the differences between a calculated or an observed data and the fitted value, is performed minimizing in this method. Thus, the coefficients of the polynomial are determined. The polynomial is given as below:

$$P(x) = \sum_{n}^{N} a_n x^n = a_0 x^0 + a_1 x^1 + a_2 x^2 + \ldots + a_N x^N \qquad (2)$$

The independent variable is the secondary neutron number, c, in all studies. The dependent variables are albedo values, extrapolation distance, and critical thickness for each problem, respectively. Therefore, Equation (2) can be written as in Eq. (3).

$$P(c) = \sum_{n=0}^{7} a_n c^n = a_0 c^0 + a_1 c^1 + a_2 c^2 + \ldots + a_7 c^7 \qquad (3)$$

The Artificial Neural Network (ANN)

The basic of artificial neural networks is based on logistic regression [8]. A logistic regression, Figure 1, consists of an input layer, a summation layer, and an output layer. The result of the summation is applied to the activation function. If the activation value is bigger than a threshold value, then the output is determined. The output gives only two different results. Therefore, the logistic regression is also called as binary classification. This structure runs as a biological neuron cell. It consists of dendrites, nuclei of the cell, and axon. The dendrite is the input, and the axon is the output. The signals are collected by dendrites in the nucleus. This determines the status of the cell. If the collected signal value is bigger than a threshold value, then the next cell is activated. An artificial neural cell in Figure 2 works similarly to a biological cell.

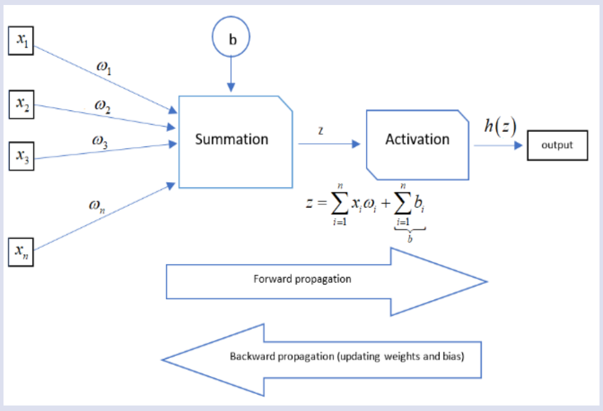

x_1
x_2
x_3
x_n
ω_1
ω_2
ω_3
ω_n
b
Summation
z
Activation
$h(z)$
output
$z = \sum_{i=1}^{n} x_i \omega_i + \underbrace{\sum_{i=1}^{n} b_i}_{b}$
Forward propagation
Backward propagation (updating weights and bias)

Figure 1. A logistic regression with computation graph.

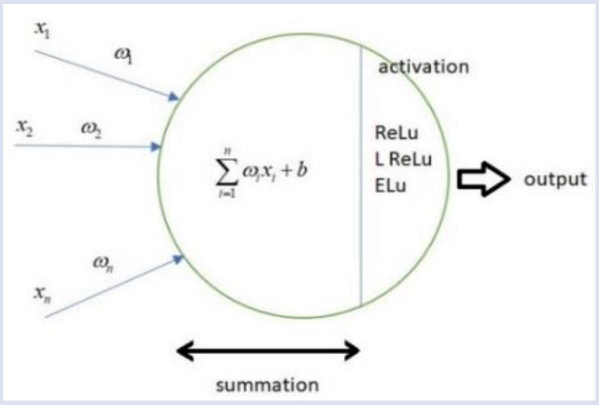


Figure 2. An artificial neuron cell in ANN.

Here x_i, ω_i and b correspond to input data, weights and bias in machine learning, respectively. The first transition from input to output doesn't generally give a good result, and it is called as the forward propagation. Therefore, the backward propagation is used. Thus, the weights and bias values are updated in the backward propagation process. The optimization function is used in this stage. It tries to minimize the loss value.

An ANN includes hidden layers unlike logistic regression. Thus, a typical ANN consists of an input layer, hidden layer, and the output layer as in Figure 3. An ANN study tries to find the weights, ω_i, and the bias, b, of the interested in problem.

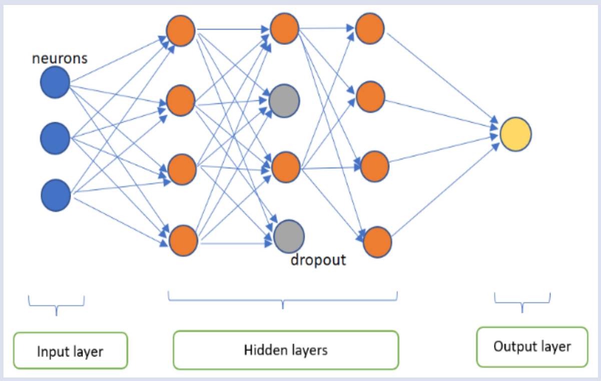


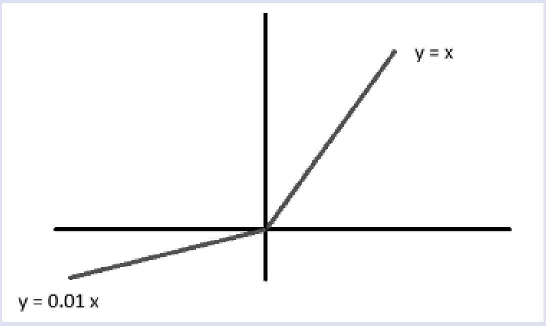
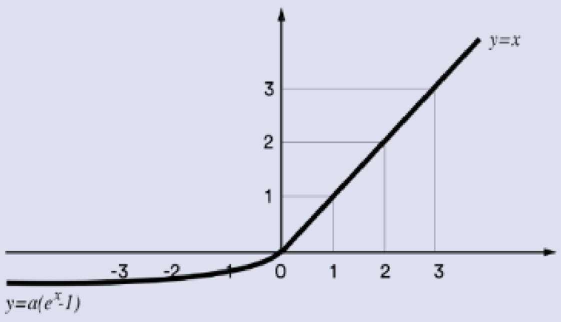
Figure 3. The structure of an Artificial Neural Network.

The connection of the layers is also important. Keras module gives us different model structures. If we have only one input layer and only one output layer, and if we don't use the output of another layer, then the sequential model [21] will be a good option to use.

In this study, ANN model is created by the sequential model, and the optimization function is selected as adam (adaptive moment estimation) [22] for the optimization function. There are different activation functions in ANN algorithms, and there are certain advantages and disadvantages of these functions. The properties of the activation functions used in this study are given in Table 1.

Leaky Relu activation function is focused to zero, and there are no killed neutrons. The function has 0.01x for the negative value. Elu (Exponential Linear Unit) activation function includes an exponential term and there are no killed neurons. However, it runs slowly in terms of the calculation time since it includes an exponential term.

Table 1. Activation functions and its properties used in this study

Mathematical definition of the activation function	The behaviour of the activation function
$h_{Leaky\,ReLu}(x) = \max(0.01x, x)$	
$h_{ELu} = \begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$	

Application of PR

Since we are interested in the data mining for one-speed neutron transport problems, we need a training data set. This data set is calculated by using the H_N method. The H_N calculations were performed with Mathematica 12.2 software, and *WorkingPrecision* value was selected as 32.

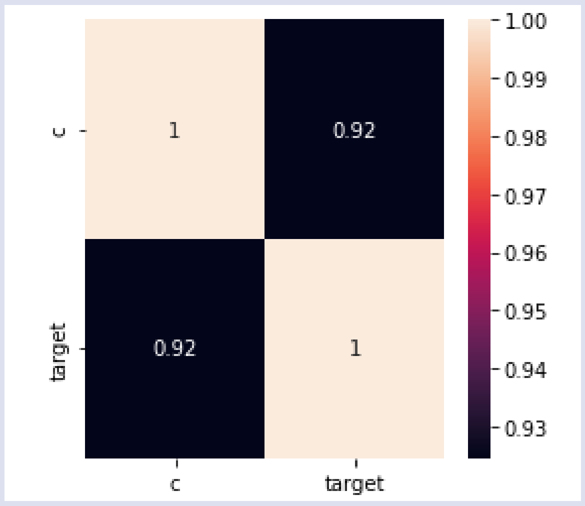
It is important that the data should contain thousands of data for the success of ANN. However, we calculated only 11 different data according to the secondary neutron number. We could calculate much more data at shorter intervals and calculate hundreds or thousands of data, but since we have not examined complex problems, we work with these 11 data, especially by forcing the neural network. The training data set is given in Table 2

Table 2. The training data set for varying c with HN method

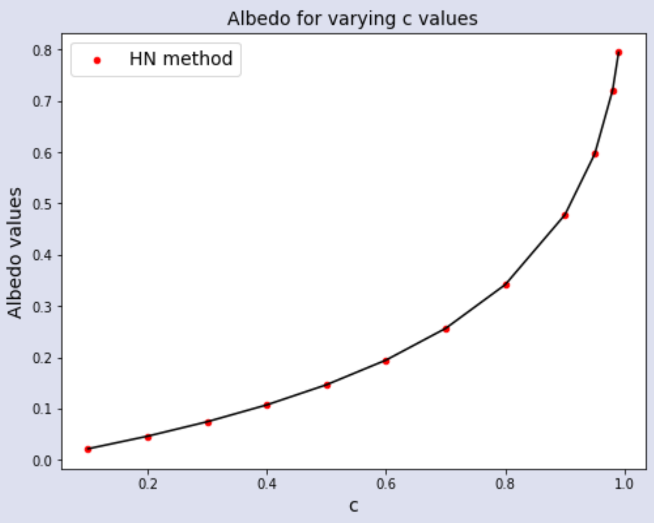
c	Albedo	c	Extrapolation distance	c	Critical thickness
0.10	0.0216976569566343	0.10	8.5382882681302200	1.10	4.2266192698707600
0.20	0.0462648688682488	0.20	3.9239076951733300	1.20	2.5787585222225100
0.30	0.0744514267270763	0.30	2.4947273586756700	1.30	1.8754510915842200
0.40	0.1073349048799300	0.40	1.8249003850628600	1.40	1.4732070964173800
0.50	0.1465444012833310	0.50	1.4408497695258000	1.50	1.2101130830389700
0.60	0.1947164506629280	0.60	1.1922598119270000	1.60	1.0239260606883200
0.70	0.2565567261583740	0.70	1.0180610666686400	1.70	0.8850734307673520
0.80	0.3418664995580910	0.80	0.8890546020955310	1.80	0.7775459392298720
0.90	0.4780245341005770	0.90	0.7895694514644200	1.90	0.6918667133043400
0.95	0.5966629255385620	0.95	0.7478782683064080	2.00	0.6220468763406790
0.98	0.7210172380719860	0.98	0.7249509445737220	2.50	0.4064698693764990

PR Application for Albedo Data

First, we should try to understand the correlation matrix. It gives the correlations between the variables which are called as features in the machine learning language. If the correlation between the two features is close to the unit value, then these features have a linear relation. If the correlation is close to the negative unit value, then the relation between the features is again linear, but the slope of the line is negative. If the correlation is close to zero, then it is understood that there is no relationship between the two features. The correlation matrix for the albedo is given in Figure 4-a.



(a)



(b)

Figure 4. (a) The correlation matrix for the secondary neutron number and the albedo values. (b) The behaviour of the calculated albedo values.

The target in Figure 4-a corresponds to the albedo. According to Figure 4-a, the correlation between the secondary neutron number and the albedo values is 0.92. This means that there is a strong relationship between c and albedo, and this relationship is a close linear behaviour. Figure 4-b represents the H_N results for the albedo and supports to the result of Figure 4-a.

Polyfit method in numpy module is used on the training data to apply polynomial regression. This analysis is performed for third-order, 4th order, 5th order, and 6th order polynomials. The coefficients for each polynomial regression and R^2 values are given in Table 3, and the graphics of the polynomials on the training data are given in Figure 5.

Table 3. The polynomial coefficients for the albedo problem

Polynomial Coefficients	Third order	4th order polynomial	5th order polynomial	6th order polynomial
a_0	-0.097160764740	0.102653259668	-0.122656652880	0.160801117064
a_1	1.268863094720	-1.301081444528	2.414879932232	-3.161361500426
a_2	-2.828679335978	6.614462062288	-12.913954563141	24.992183953605
a_3	2.426921750454	-10.538806383260	33.205888658331	-86.116136609767
a_4	-	5.925015300984	-37.689181901735	151.820348299569
a_5	-	-	15.929876969293	-131.434216376403
a_6	-	-	-	44.578524178006
R^2	0.9892459906109	0.9960750831911	0.9985235712740	0.9993852906092

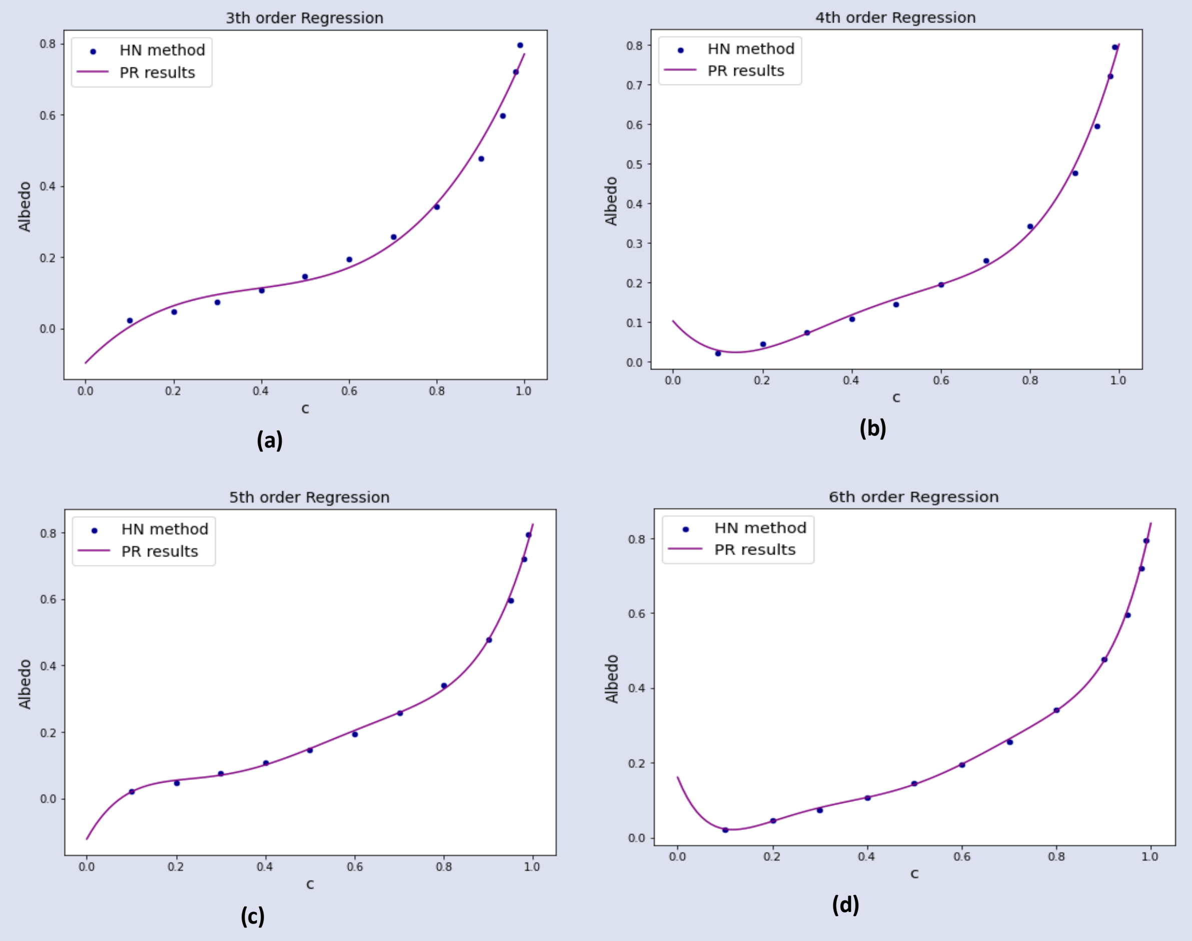


Figure 5. Polynomial regression for the albedo values with (a) third order, (b) 4th order, (c) 5th order and, (d) 6th order polynomials.

Since we know the polynomial coefficients, we can predict the new data. The predicted values will be given in ANN section together with the ANN results.

PR Application for The Milne Problem

The correlation matrix and the HN method results are given in Figure 6 where the target corresponds to the extrapolation distance. The correlation between the extrapolation distance values and c values has negative value. It corresponds to the extrapolation distance values decrease as c values increase. The polynomial coefficients are given in Table 4 with the R^2 values. The behaviours of the polynomials are given in Figure 7. The predicted PR results for the extrapolation distance are given together with the ANN results.

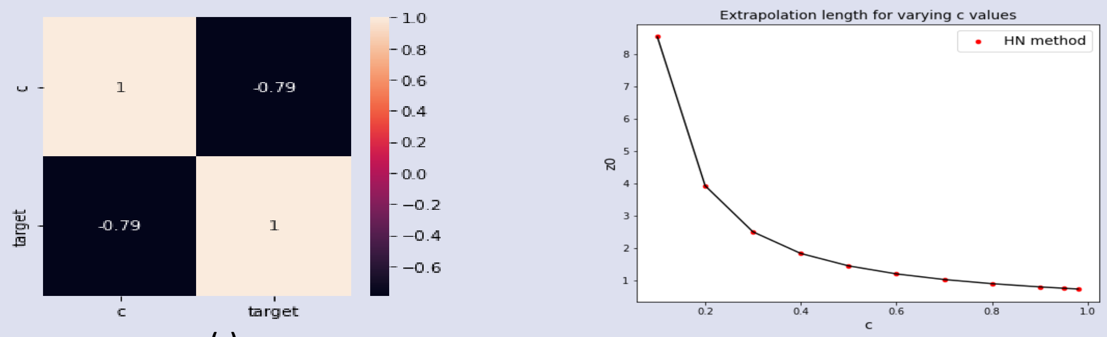


Figure 6. (a) The correlation matrix for the secondary neutron number and the extrapolation distance values. (b) The behaviour of the calculated the extrapolation distance values.

Table 4. The polynomial coefficients for the Milne problem

Polynomial Coefficients	Third order	4th order polynomial	5th order polynomial	6th order polynomial
a_0	12.731039590977	15.788245491725	18.485254724793	20.894687303262
a_1	-54.931030528914	-94.652754304187	-139.480703388677	-187.134618844626
a_2	83.735974910281	231.577867207033	469.633521969347	795.996424716015
a_3	-41.293833149226	-247.138836479952	-786.754877968709	-1823.178700875148
a_4	-	95.432074098969	640.269808323104	2302.409952872807
a_5	-	-	-201.601504460371	-1507.521802226705
a_6	-	-	-	399.319132359482
R^2	0.9714452701730	0.9940761925904	0.9990018141316	0.999872681941

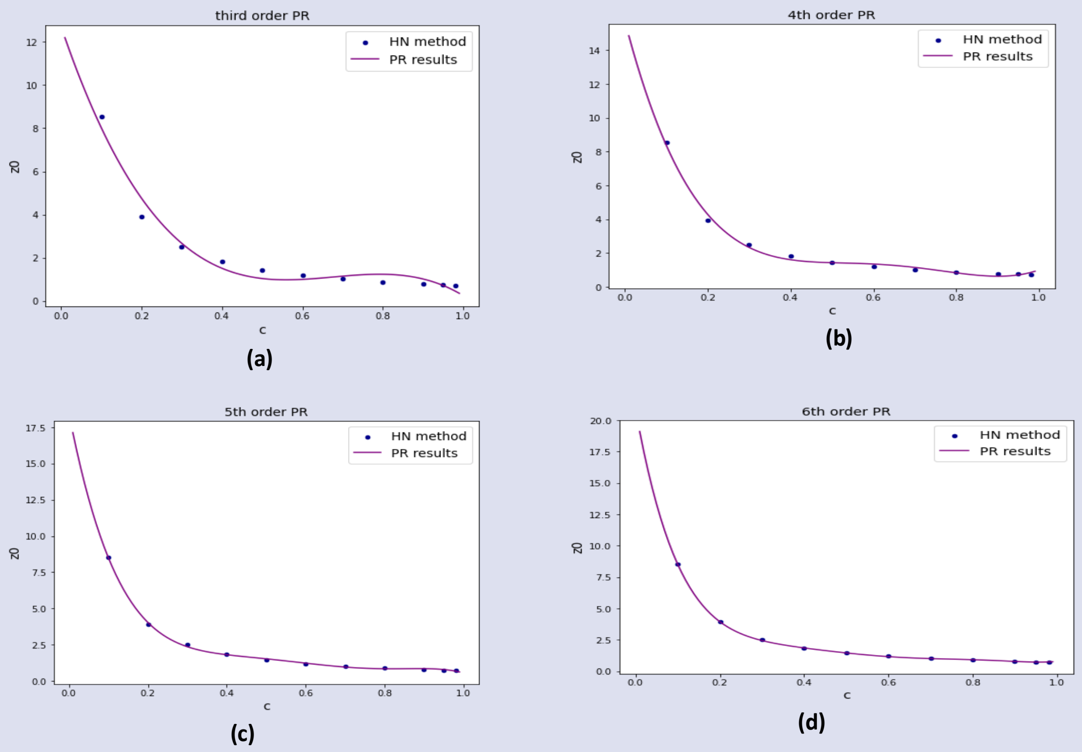
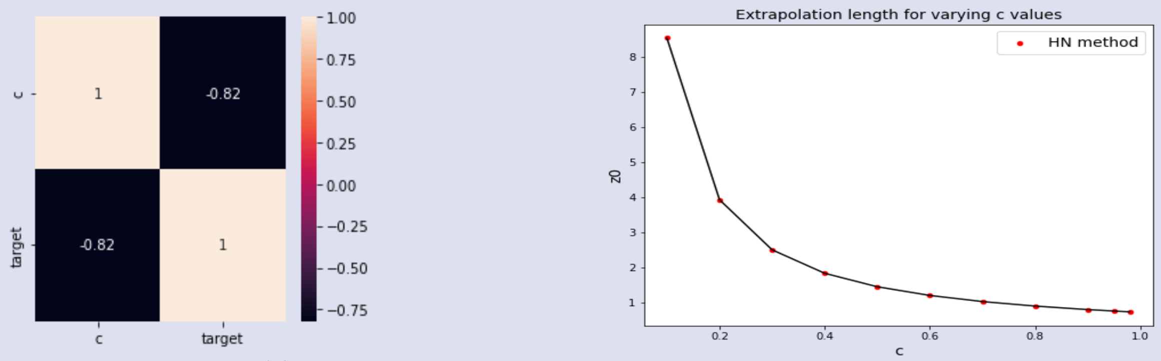


Figure 7. Polynomial regression for the extrapolation distance values with (a) third order, (b) 4th order, (c) 5th order, and (d) 6th order polynomials.

PR Application for The Criticality Problem

The correlation matrix and the H_N method results are given in Figure 8 where the target corresponds to the critical thickness values. The correlation between the criticality thickness values and c values has negative value. It corresponds to the critical thickness values decrease as c values increase. The polynomial coefficients are given in Table 5 with the R^2 values. The behaviours of the polynomials are given in Figure 9.

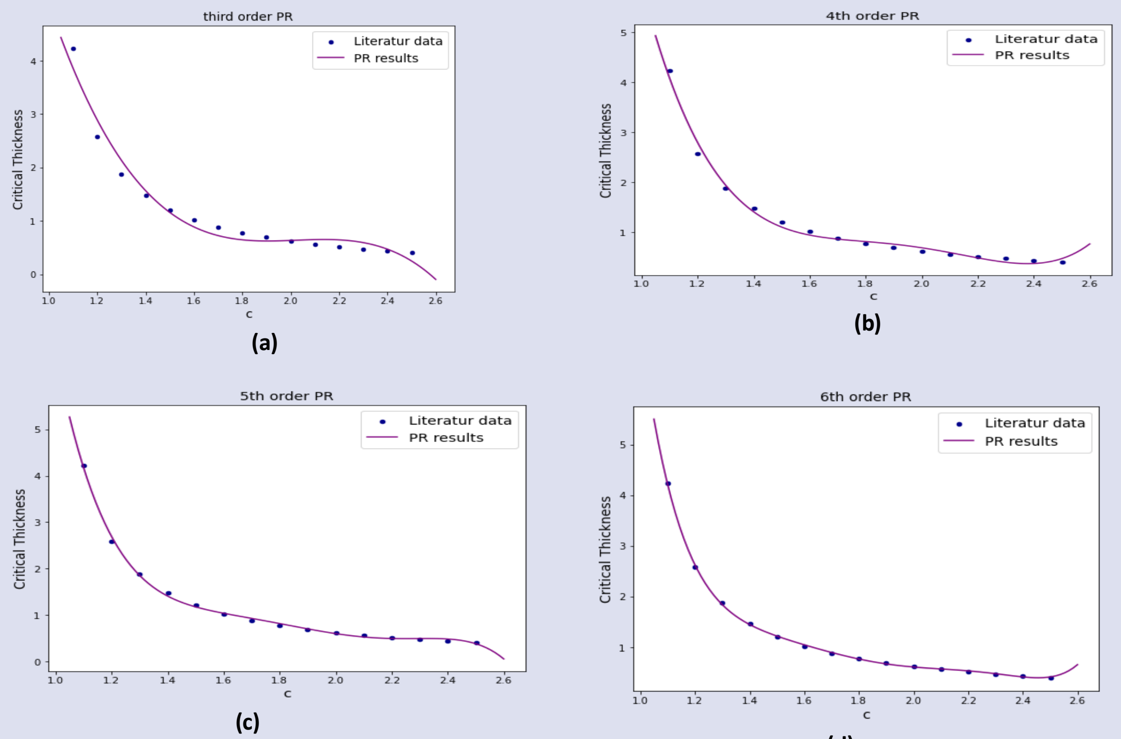


(a) (b)

Figure 8. (a) The correlation matrix for the secondary neutron number and the extrapolation distance values. (b) The behaviour of the calculated the extrapolation distance values.

Table 5. The polynomial coefficients for the critically problem

Polynomial Coefficients	Third order	4th order polynomial	5th order polynomial	6th order polynomial
a_0	36.033612931841	94.467135307170	244.733526359218	627.280616843006
a_1	-52.875734964750	-193.416454497710	-646.162511121329	-2030.175944634479
a_2	26.241293304929	149.504546920933	683.519407553603	2733.972521931903
a_3	-4.326510203990	-51.117712902239	-359.541070193650	-1952.377568565584
a_4	-	6.498778152535	93.803659999828	778.418589815078
a_5	-	-	-9.700542427477	-164.163088552406
a_6	-	-	-	14.302087604165
R^2	0.9707663267140	0.992025771469	0.9980065173723	0.9995529480494



(a) (b) (c) (d)

Figure 9. Polynomial regression for the critical thickness values with (a) third order, (b) 4th order, (c) 5th order, and (d) 6th order polynomials.

Application of ANN

ANN application is different from PR application. ANN includes hyperparameters which are depend on the problem and the user. It is important that there is no any linearity among these hyperparameters. The used hyperparameters for each problem are given in Table 6. Unfortunately, the values or situations of these hyperparameters depend on the experience and the type of problem. The ANN results for albedo are given in Figure 10 with the loss functions.

Table 6. The ANN hyperparameters used in this study

Hyperparameters	Albedo	The Milne problem	Criticality
Number of neurons	200	200	200
Epoch number	300	600	600
Batch size parameter	8	8	8
Test size	0.2	0.2	0.2
Validation split parameter	0.2	0.2	0.2
Number of hidden layers	5	5	10
Loss function	msle	msle	msle
Optimization function	adam	adam	adam
Activation function	LRelu / Elu(a=0.1)	LRelu / Elu(a=0.1)	LRelu / Elu(a=0.4)

msle: Mean square logarithmic error

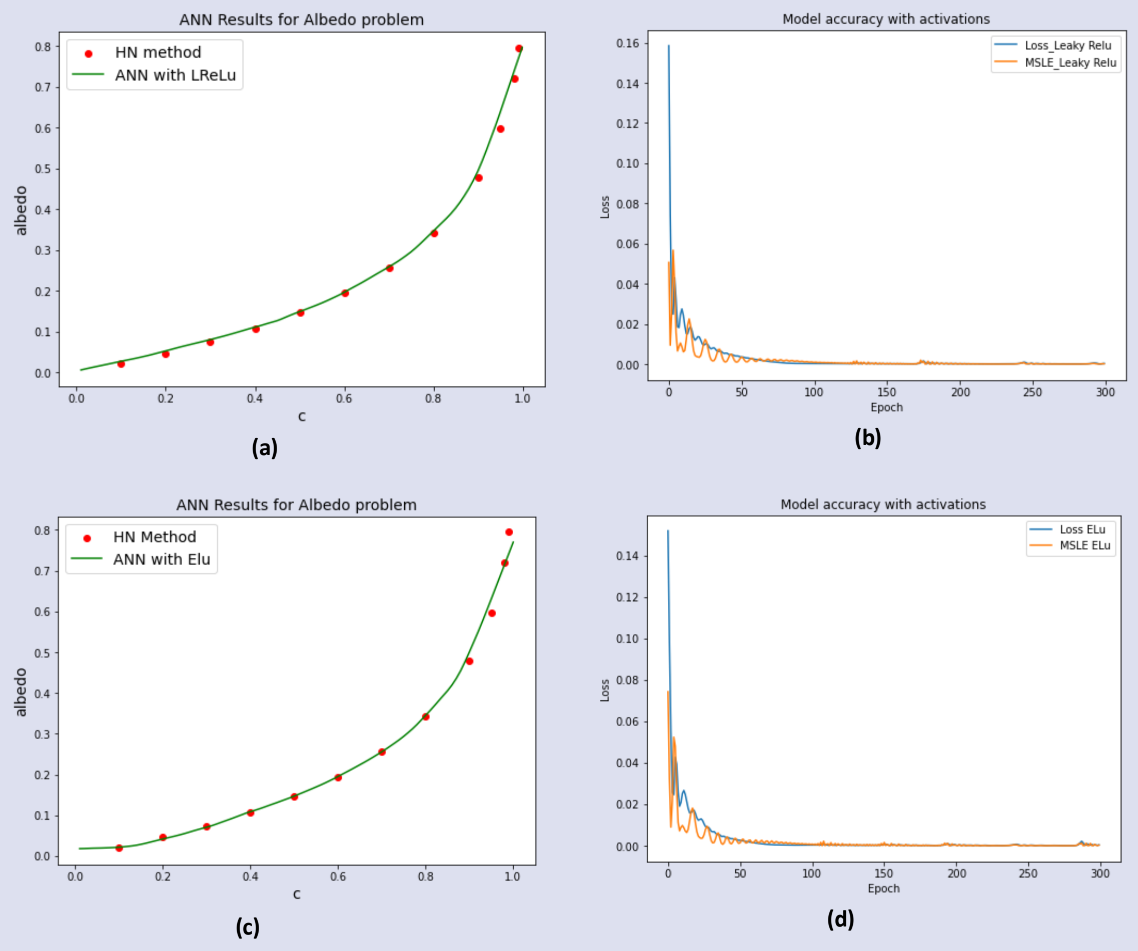
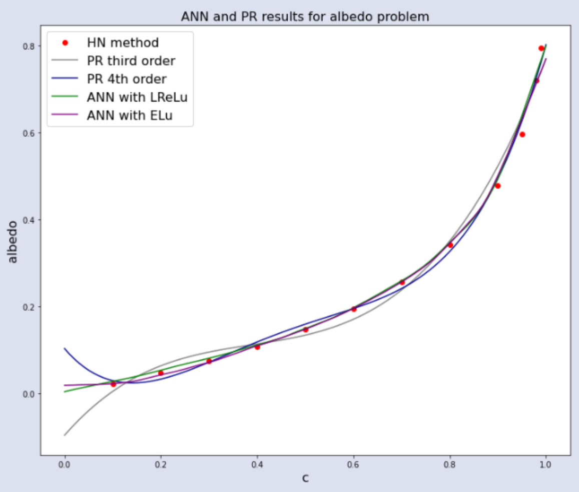
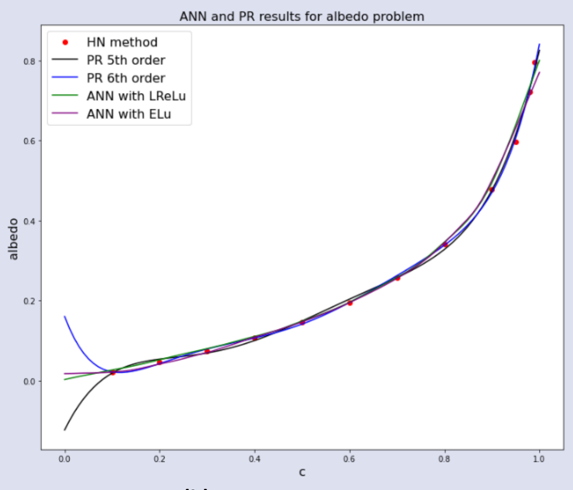


Figure 10. (a) ANN with Leaky Relu activation function, (b) Loss function for Leaky Rely, (c) ANN with Elu activation function, (d) Loss function for Elu.

Figures 11 and 12 show the ANN results with Leaky Relu and Elu activation functions and PR results with third order and 4th order and with 5th and 6th order PR results. These figures are given as separated not to cause confusion. According to the results PR gives good results for the training data set range. But PR results are not good for out of the training data set. While ANN results are reasonable. Table 7 represents the predicted values by PR and ANN predictions and the calculated data by H_N method. According to the tabulated results, ANN results are better than the PR results in the training data range.



(a)



(b)

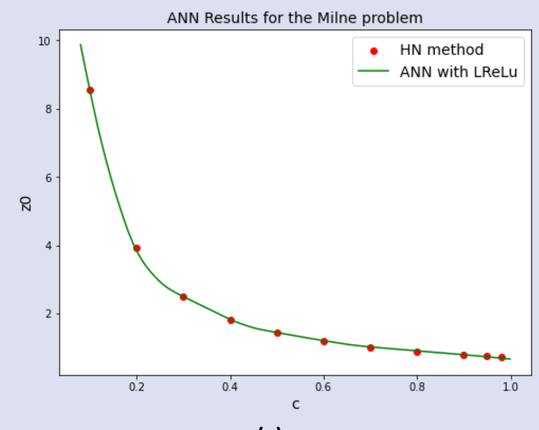
Figure 11. (a) ANN results and PR of third and 4th order polynomials,

Figure 11 (b). ANN results and PR with 5th and 6th order polynomials.

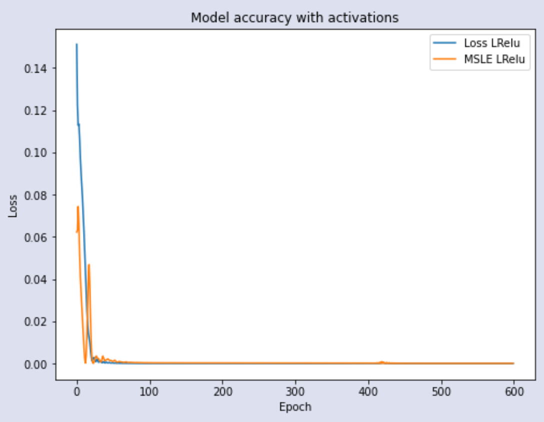
Table 7. The predicted albedo values with PR and ANN, HN results and the literature data

c	PR third order	PR 4th order	PR 5th order	PR 6th order	ANN LeakyRelu	ANN ELu	HN results	Ref [6]
0.05	-0.040485943	0.052855023	-0.030277386	0.055357485	0.015550368	0.019501526	0.010527886	-
0.10	0.003865673	0.028743431	0.019288064	0.022382941	0.027096316	0.021702837	0.021697657	0.0217
0.15	0.037714275	0.023747507	0.043210760	0.025665113	0.038747564	0.028760070	0.021697317	-
0.20	0.062880055	0.032185027	0.054203130	0.042993716	0.052679956	0.041934598	0.046264869	0.04626
0.25	0.081183203	0.049262519	0.061116334	0.062485542	0.066931598	0.055374168	0.059847996	-
0.30	0.094443911	0.071075264	0.069537638	0.079410954	0.080342233	0.070764676	0.074451427	0.07445
0.35	0.104482370	0.094607294	0.082387780	0.093521891	0.095019184	0.089193664	0.090220877	-
0.40	0.113118771	0.117731395	0.100518348	0.106881382	0.110837251	0.108875334	0.107334905	0.1073
0.45	0.122173307	0.139209104	0.123309139	0.122194567	0.126786008	0.127226219	0.126015681	-
0.50	0.133466167	0.158690711	0.149265541	0.141641226	0.148766205	0.147058889	0.146544401	-
0.55	0.148817544	0.176715258	0.176615896	0.166209819	0.170162231	0.169510245	0.169284290	-
0.60	0.170047629	0.194710540	0.203908873	0.195533031	0.196252808	0.195216626	0.194716451	0.1947
0.65	0.198976613	0.214993102	0.230610837	0.228224827	0.226612359	0.224009961	0.223497145	-
0.70	0.237424687	0.240768243	0.257703221	0.262719018	0.258634686	0.255775630	0.256556726	-
0.75	0.287212043	0.276130016	0.288279897	0.298609331	0.295483530	0.293667436	0.295279357	-
0.80	0.350158872	0.326061223	0.328144544	0.338490986	0.346499681	0.344976991	0.341866500	0.3419
0.85	0.428085366	0.396433420	0.386408019	0.390303793	0.403047055	0.405470312	0.400170417	-
0.90	0.522811714	0.494006916	0.476085728	0.470176740	0.494039237	0.500720441	0.478024534	0.4780
0.95	0.636158110	0.626430770	0.614694997	0.605774103	0.638666987	0.630749643	0.596668782	-

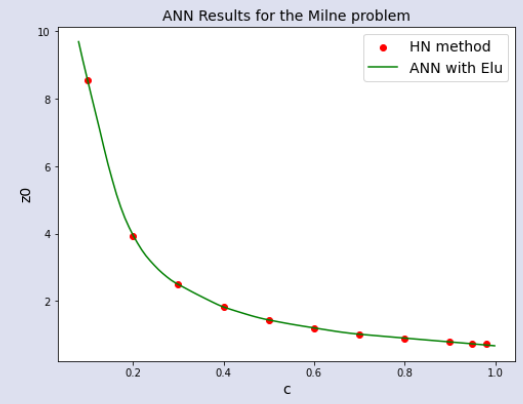
The Milne problem results with ANN are given in Figure 12. The predicted values are given in Table 8. ANN results are better than the PR results as in the albedo investigation. The comparisons are given in Figures 13 and 14. The critical thickness results with ANN are given in Figure 15 with the loss functions. The comparisons of the results with both ANN and PR are given in Figures 16 and 17. The predicted values and calculated values with H_N method are given in Table 9. Table 10 represents the CPU times for ANN calculations.



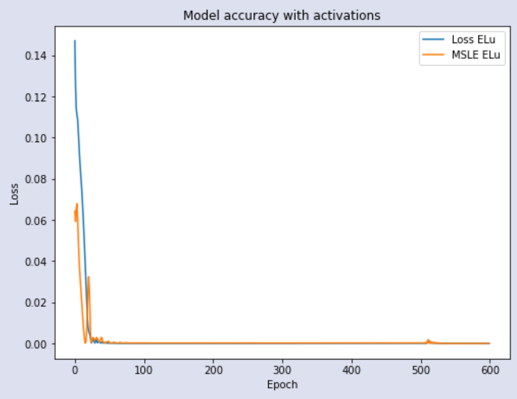
(a)



(b)

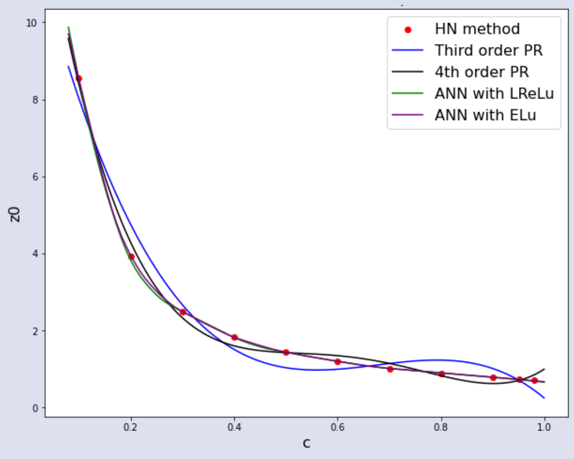


(c)

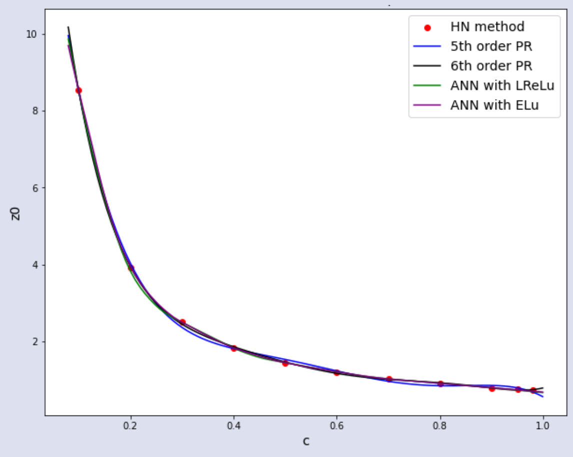


(d)

Figure 12. (a) ANN with Leaky Relu activation function, (b) Loss function for Leaky Rely, (c) ANN with Elu activation function, (d) Loss function for Elu.



(a)



(b)

Figure 13. ANN results and PR with third and 4th order polynomials.

Figure 14. ANN results and PR with 5th and 6th order polynomials

Table 8. The predicted albedo values with PR and ANN, HN results and the literature data

c	PR third order	PR 4th order	PR 5th order	PR 6th order	ANN LeakyRelu	ANN ELu	HN results	Ref [6]
0.05	10.188666273	11.604256540	12.590897686	13.313975286	11.840276718	11.550756454	18.17324720	-
0.10	8.034002454	8.401153104	8.508775693	8.533576061	8.537860870	8.543089867	8.538288268	8.53829
0.15	6.236077760	6.015053273	5.783433224	5.636852014	5.736626625	5.809600353	5.426971982	-
0.20	4.763921816	4.296389946	4.040335114	3.949196288	3.837925434	3.960925341	3.923907695	3.92391
0.25	3.586564248	3.109910836	2.978306502	2.992731577	2.928089142	3.025303841	3.054278391	-
0.30	2.673034679	2.334678464	2.361972772	2.445501237	2.494558573	2.496745348	2.494727359	2.49473
0.35	1.992362736	1.864070166	2.014199497	2.105152759	2.157500029	2.143542767	2.107561555	-
0.40	1.513578044	1.605778085	1.808532382	1.857113569	1.824761152	1.826388121	1.824900385	1.82490
0.45	1.205710227	1.481809179	1.661637210	1.647259169	1.581019044	1.614304304	1.609850598	-
0.50	1.037788910	1.428485213	1.525739783	1.459073629	1.440720677	1.441888928	1.440849770	1.44085
0.55	0.978843720	1.396442766	1.381065865	1.295302402	1.319161654	1.315809488	1.304544227	-
0.60	0.997904281	1.350633227	1.228281131	1.164097496	1.203084111	1.206238747	1.192259812	1.19226
0.65	1.064000218	1.270322798	1.080931103	1.069654976	1.096536160	1.100451231	1.098132352	-
0.70	1.146161157	1.149092489	0.957881098	1.007344810	1.017981768	1.019245267	1.018061067	1.01806
0.75	1.213416721	0.994838123	0.875756172	0.963333053	0.960831523	0.961277604	0.949094398	-
0.80	1.234796538	0.829770334	0.841381062	0.918696374	0.903681397	0.903938890	0.889054602	0.889055
0.85	1.179330231	0.690414567	0.844220129	0.858028921	0.846531689	0.846978545	0.836299972	-
0.90	1.016047426	0.627611078	0.848817303	0.782541528	0.789381742	0.790483654	0.789569451	0.789569
0.95	0.713977749	0.706514934	0.787236027	0.727653262	0.728278637	0.733815908	0.747878268	-

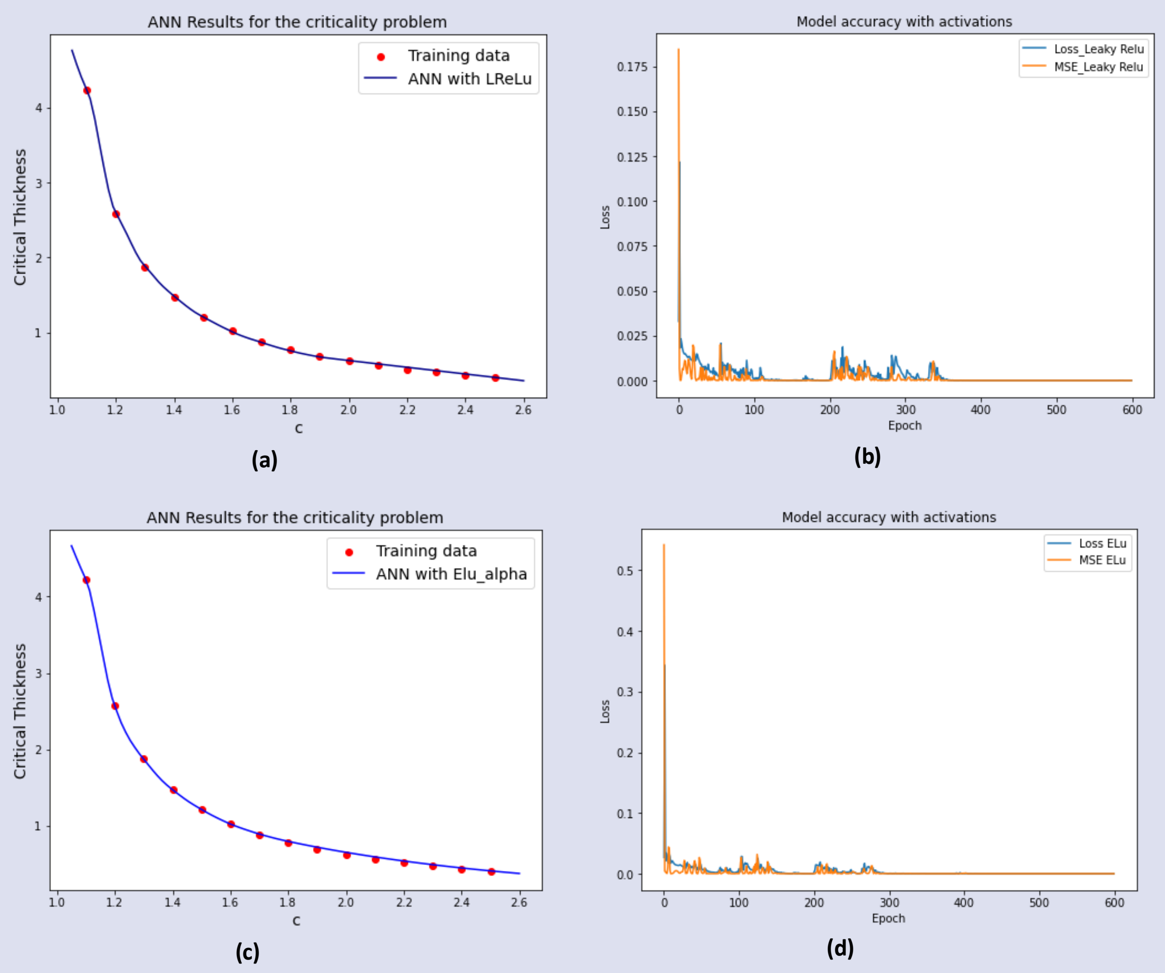
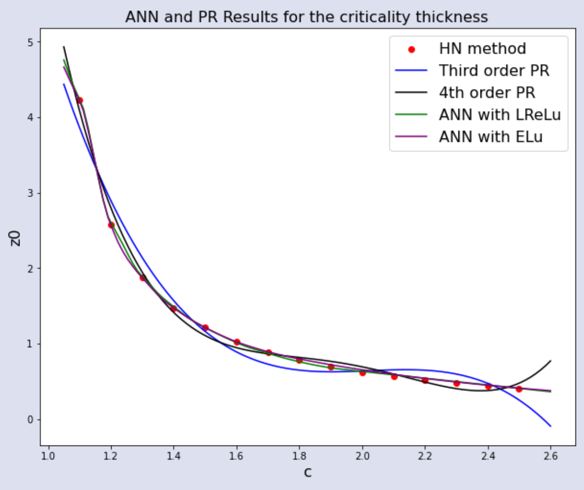
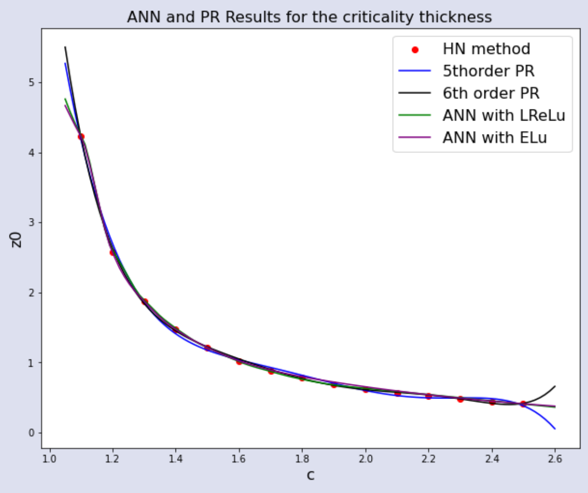


Figure 15. (a) ANN with Leaky Relu activation function, (b) Loss function for Leaky Rely, (c) ANN with Elu activation function, (d) Loss function for Elu.



(a)



(b)

Figure 16. ANN results and PR with third and 4th order polynomials.

Figure 17. ANN results and PR with 5th and 6th order polynomials

Table 9. The predicted critical thickness values with PR and ANN, HN results and the literature data

c	PR third order	PR 4th order	PR 5th order	PR 6th order	ANN LeakyRelu	ANN ELu	HN results	Ref [6]
1.05	4.436640713	4.932784128	5.267616683	5.500000720	4.759516716	4.665410995	6.600527483	-
1.10	3.863684288	4.086722354	4.179200859	4.212667744	4.239711285	4.222702503	4.226619270	4.22674
1.15	3.350546912	3.380727934	3.335961807	3.284674501	3.389765739	3.397710323	3.187897630	-
1.20	2.893983701	2.798395958	2.692706239	2.625772469	2.603260517	2.569414139	2.578758522	-
1.25	2.490749773	2.324296336	2.209944696	2.163469717	2.217116833	2.138610840	2.170721623	-
1.30	2.137600245	1.943973792	1.853527782	1.840655903	1.889006495	1.871923208	1.875451092	1.87766
1.35	1.831290235	1.643947868	1.594282390	1.613388171	1.664572239	1.643992543	1.650649761	-
1.40	1.568574859	1.411712922	1.407647933	1.448837946	1.487948179	1.470546722	1.473207096	1.47688
1.45	1.346209236	1.235738131	1.273312576	1.323398629	1.332890511	1.329103589	1.329304314	-
1.50	1.160948482	1.105467485	1.174849460	1.220954189	1.210703254	1.213001370	1.210113083	-
1.55	1.009547716	1.011319793	1.099352938	1.131308657	1.107202888	1.109943986	1.109702483	-
1.60	0.888762053	0.944688681	1.037074800	1.048776508	1.012116909	1.021637082	1.023926061	1.03039
1.65	0.795346613	0.897942591	0.981060505	0.970933957	0.933853269	0.952186227	0.949789829	-
1.70	0.726056511	0.864424782	0.926785412	0.897531141	0.871189833	0.891013980	0.885073431	0.89275
1.75	0.677646865	0.838453328	0.871791003	0.829565207	0.809955120	0.839110076	0.828092811	-
1.80	0.646872794	0.815321123	0.815321123	0.768514291	0.758651495	0.795110703	0.777545939	0.7863
1.85	0.630489413	0.791295876	0.757958201	0.715732404	0.714092731	0.755685329	0.732409325	-
1.90	0.625251840	0.763620111	0.701259481	0.672005211	0.677867234	0.718870461	0.691866713	0.70157
1.95	0.627915194	0.730511172	0.647393258	0.637266710	0.652532697	0.684144139	0.655258774	-
2.00	0.635234590	0.691161218	0.598775100	0.610476808	0.629557252	0.651447415	0.622046876	0.63257
2.05	0.643965147	0.645737224	0.557704080	0.589659799	0.607055962	0.620682538	0.591786523	-
2.10	0.650861981	0.595380984	0.525999009	0.572103739	0.584608793	0.591545403	0.564107544	-
2.15	0.652680211	0.542209106	0.504634661	0.554720714	0.562175453	0.563958168	0.538665175	-
2.20	0.646174953	0.489313017	0.493378006	0.534568018	0.539779484	0.537853360	0.515298304	-
2.25	0.628101325	0.440758958	0.490424436	0.509530217	0.517384887	0.513141751	0.493649386	-
2.30	0.595214444	0.401587991	0.492034001	0.479162122	0.494980454	0.489778817	0.473655588	-
2.35	0.544269428	0.377815991	0.492167630	0.445692651	0.472600222	0.467705846	0.455029751	-
2.40	0.472021393	0.376433650	0.482123370	0.415189600	0.450304508	0.446859717	0.437739484	-
2.45	0.375225457	0.405406479	0.450172606	0.398885300	0.427969217	0.427191317	0.421561906	-
2.50	0.250636738	0.473674805	0.381196300	0.414663185	0.405631959	0.408649623	0.406469869	-
2.55	0.095010353	0.591153769	0.256321214	0.488705251	0.383291245	0.391174197	0.392302667	-

Table 10. The CPU times in seconds

Activation function	Albedo	Milne's problem	Criticality
Leaky Relu	7.81	14.20	18.60
Elu	7.66	14.90	21.30

Conclusions

In this study, two different machine learning algorithms, polynomial regression and artificial neural network, were applied to isotropic scattering neutron transport theory problems. This study is about the data mining for the existing data for albedo, Milne problem, and criticality problem. These data are the training data. The training data for each problem was calculated by H_N method.

The success of machine learning applications depends on the size of the training data. We studied here with restricted data. We have only 11 different data for each problem. We could have created a larger data set, but we especially wanted to force the artificial neural network.

According to the results

Polynomial regression could give reasonable results for only training data range. PR results are not good for out of data ranges. Therefore, the predictions used PR should belong to the training data range.

ANN results are more successful both the training data range and out of the training data range. Although ANN calculations take more time than polynomial regression, the success of ANN is worth it.

ANN includes hyperparameters. Unfortunately, there is no any linearity among these hyperparameters. Used hyperparameters in this study are given in Table 6. When we create a larger neural network of 10 or 20 hidden layers, we see that ANN result becomes an underfitting situation, a linear behaviour. When we create a smaller neural network, we again see that ANN gives an underfitting result. This comparison is only for the number of hidden layers. Similar comparisons are valid for other hyperparameters such as neuron number, activation function, optimizers. For example, if we think that we can choose much more neuron number then, ANN will give underfitting results. The hyperparameters in this study are the valid for the training data set in this study. If we use richer data set, then the hyperparameters could be updated.

Although the values are completely different for the extrapolation distance and the criticality problem, the behaviour is similar. The extrapolation distance and the critical thickness values decrease as the secondary neutron number increase. However, ANN includes 5 hidden layers in the extrapolation distance calculations, 10 hidden layers in the critical thickness calculations.

Acknowledgment

This work was studied with the computer system which was collected with personal opportunities.

Conflicts of interest

There are no conflicts of interest in this work.

References

[1] Carlson B.G., Solution of the Transport Equation by SN Approximations. Los Alamos Scientific Laboratory, LA-1599, United States, (1955) 1-29.

[2] Lewis E.E., Miller W.F., Computational Methods of Neutron Transport. United States, (1984).

[3] Case K.M., Zweifel P.F., Linear Transport Theory. Addition-Wesley: MA, (1967) 1-270.

[4] Case K.M., Elementary solutions of the transport equation and their applications, *Annals of Phys.*, 9 (1) (1960) 1–23.

[5] Kavenoky A., The CN Method of Solving the Transport Equation: Application to Plane Geometry, *Nuclear Science and Eng.,* 65 (2) (1978) 209-225.

[6] Grandjean P., Siewert C.E., The FN method in neutron-transport theory. Part II: applications and numerical results, *Nucl. Sci. Eng.,* 69 (2) (1979) 161-168.

[7] Tezcan C., Kaşkaş A., Güleçyüz M.Ç., The HN method for solving linear transport equation: *theory and applications, JQSRT*., 78 (2) (2003) 243-254.

[8] Géron A., Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow, 2nd ed. O'Reilly Media, (2019).

[9] Chen Z., Andrejevic N., Drucker N.C., Nguyen T., Xian R.P., Smidt T., Wang Y., Ernstorfer R., Tennant D.A., Chan M., Li M., Machine learning on neutron and x-ray scattering and spectroscopies, *Chem. Phys. Rev*., 2 (2021) 031301.

[10] Whewell B., McClarren R.G., Data reduction in deterministic neutron transport calculations using machine learning, *Annals of Nuclear Energy*., 176 (1) (2022) 109276.

[11] Xie Y., Wang Y., Ma Y., Wu Z., Neural Network Based Deep Learning Method for Multi-Dimensional Neutron Diffusion Problems with Novel Treatment to Boundary, *J. Nucl. Eng*., 2 (2021) 533-552.

[12] Zolfaghari M., Masoudi S.F., Rahmani F., Fathi A., Thermal neutron beam optimization for PGNAA applications using Q-learning algorithm and neural network, *Sci. Rep*., 12 (2022) 8635.

[13] Zheng C., Liub L., Muc L., Solving the linear transport equation by a deep neural network approach, *Preprint submitted to Journal of Discrete and Continuous Dynamical System-S.,* 15 (4) (2021) 669-686.

[14] Numpy. Available at: https://numpy.org/doc/stable/user/index.html#user Retrieved August 2022.

[15] Scipy. Available at: https://scipy.org/ Retrieved August 2022.

[16] Sklearn. Available at: https://scikit-learn.org/stable/ Retrieved August 2022.

[17] Keras. Available at: https://keras.io/ Retrieved August 2022.

[18] Tensorflow. Available at: https://www.tensorflow.org/ Retrieved August 2022.

[19] Polyfit. Available at: https://numpy.org/doc/stable/reference/generated/numpy.polyfit.html Retrieved August 2022.

[20] Sutskever, I., Vinyals O., Le Q.V., Sequence to Sequence Learning with Neural Networks, arXiv:1409.3215v3., (2014).

[21] Kingma D.P., Ba J.L., Adam: A Method for Stochastic Optimization, arXiv:1412.6980v9., (2017).

[22] Xu B., Wang N., Chen T., Li M., Empirical Evaluation of Rectified Activations in Convolution Network, arXiv:1505.00853v2., (2015).

[23] Clevert D., Unterthiner T., Hochreiter S., Fast and accurate deep network learning by exponential linear units (Elus), arXiv:1511.07289v5., (2016).

[24] Atalay M.A. The critical slab problem for reflecting boundary conditions in one-speed neutron transport theory, *Annals of Nuclear Energy*., 23 (3) (1996) 183-193.