Report on DNNs

Symbat Bekzhigit

June 2023

1 Custom model: 1 convolution layer

1.1 Model description

I created a custom CNN model with the following layers: Convolution, ReLU, Flatten, and Dense layers. The input and output channels for the convolutional layer was taken as 1 and 6 respectively, and the kernel size was 3. The input dimension for the Dense layer was computed as 4056 and the output dimension was taken as 10 to match the number of classes in the MNIST dataset.

I trained this model using 250 epochs on the MNIST dataset. The hyper-parameters used when training were: batch size = 4, learning rate = 0.001.

1.2 Original

1.2.1 Input and output distribution of non-linear functions

After loading the best model, I attached the forward hook to the ReLU layer (the only non-linear function in the model), and plotted the input and output distribution for this non-linear function. Below are the results:

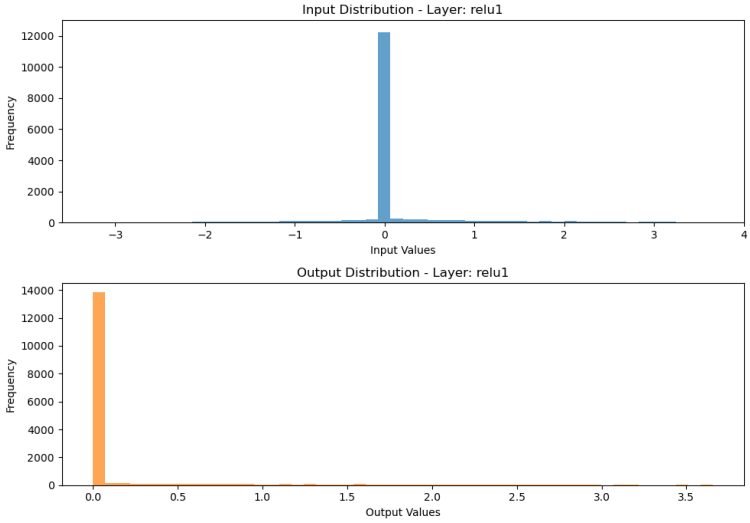


Figure 1: Input and output distribution for ReLU1

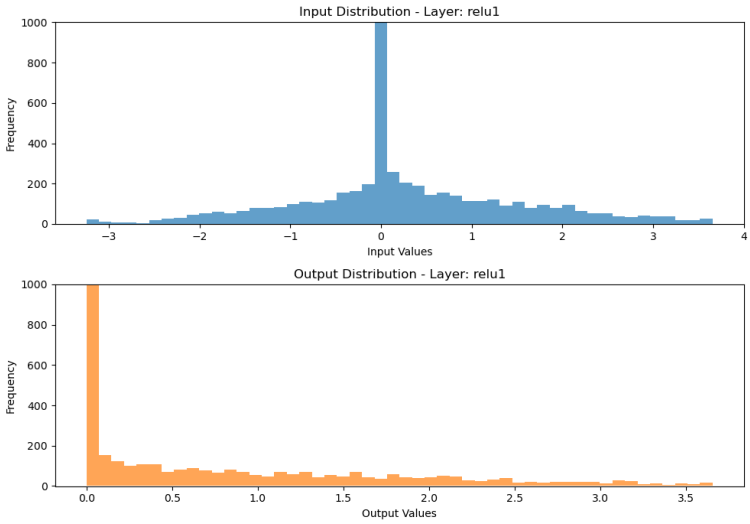


Figure 2: Input and output distribution for ReLU1 [zoomed]

1.2.2 Results

The best validation accuracy achieved when training the original model with 250 epochs was **97.290**.

1.3 Polynomial approximation without retraining

1.3.1 Approximation functions

This model has only 1 non-linear function which is a ReLU function. In order to approximate the ReLU function with a polynomial of a given degree, I used the **Remez algorithm** which is an iterative method used to find the best polynomial approximation of a given function over a specified interval. It aims to minimize the maximum error between the function and the polynomial approximation, also known as the maximum deviation.

The Remez algorithm iteratively improves the polynomial coefficients by focusing on reducing the maximum deviation. By adjusting the coefficients based on the largest error, it gradually refines the polynomial approximation to better match the original function within the specified interval.

In this case, the objective function measures the error between the ReLU function and the polynomial approximation, and the algorithm optimizes the polynomial coefficients to minimize this error. The resulting coefficients represent the best polynomial approximation of the ReLU function within the specified range.

1.3.2 Input and output distribution of non-linear layers after approximation

The input and output distributions of the non-linear (now polynomial) functions of the approximated model are as below:

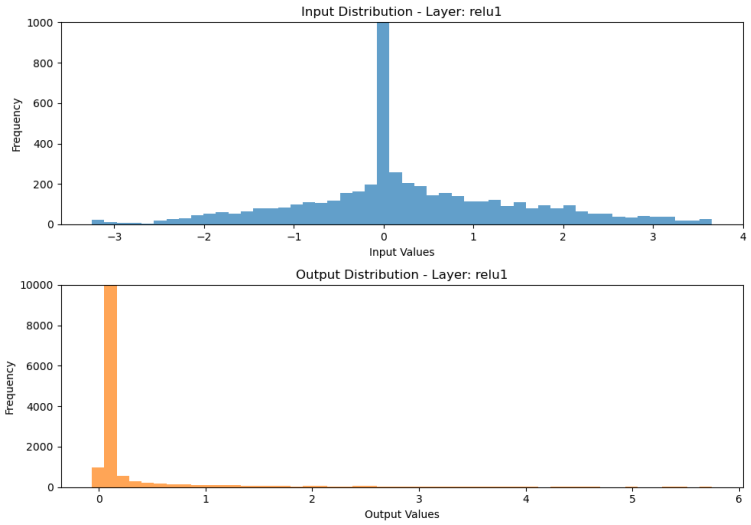


Figure 3: Input and output distribution for ReLU1 after polynomial approximation

1.3.3 Results

I tested the accuracy by trying different polynomial orders in the range of 2-8 and providing different input ranges. The closest-to-original accuracy I got was when approximating ReLU with a polynomial of **order 2** in the range **[-3,3]**. The coefficients corresponding to the order-2 polynomial equation $ax^2 + bx + c$ were **0.28124766, 0.5003125, 0.15625043** and the accuracy obtained was **96.840**.

2 Custom model: 2 convolution layers

2.1 Model description

This CNN model consists of 2 convolution and 2 non-linear functions, followed by flatten and dense layers. The input and output channels for the first convolutional layer were taken as 1 and 6, while for the second convolutional layer they were 6 and 12, respectively. The kernel size was 3, the input dimension for the dense layer was computed as 6912, and the output dimension was taken as 10 to match the number of classes in the MNIST dataset.

I trained this model using 250 epochs on the MNIST dataset. The hyperparameters used when training were: batch size = 4, learning rate = 0.001.

2.2 Original

2.2.1 Input and output distribution of non-linear functions

Input and output distribution for the first non-linear function:

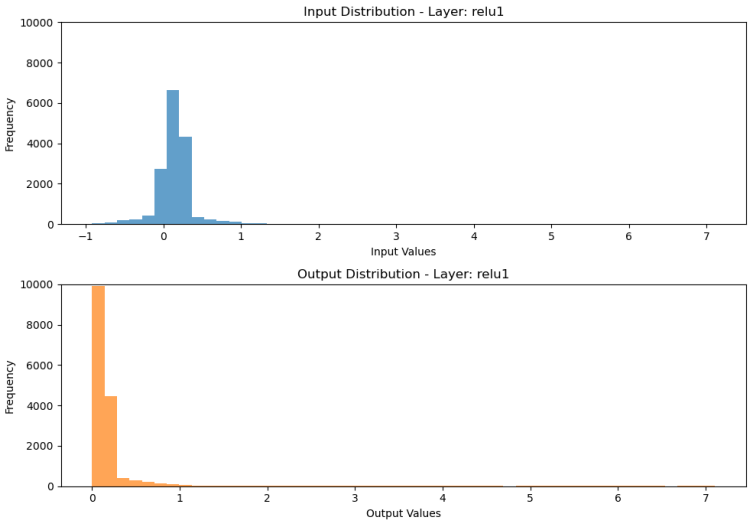


Figure 4: Input and output distribution for relu1

Input and output distribution for the second non-linear function:

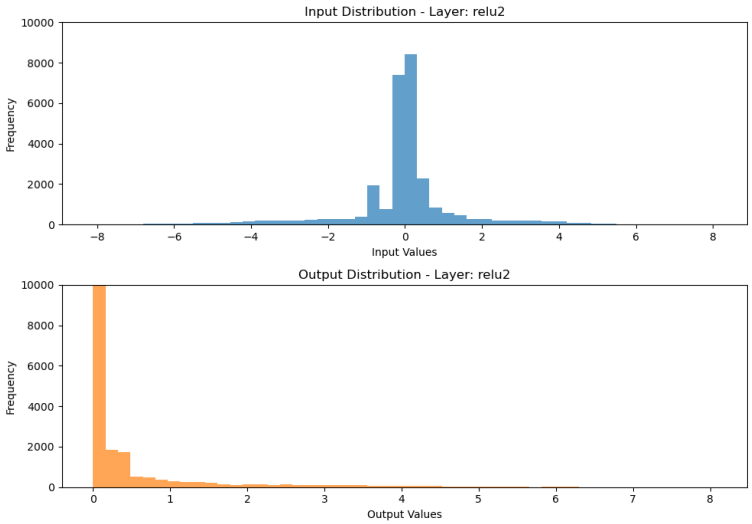


Figure 5: Input and output distribution for relu2

2.2.2 Results

The best validation accuracy achieved when training the original model with 250 epochs was **98.620**.

2.3 All-at-once polynomial approximation without retraining

2.3.1 Approximation functions

Based on the original input distribution, I approximated relu1 layer with a polynomial $0.08602914x^2 + 0.49189841x + 0.4987775$ which was obtained by running Remez algorithm for the range [-0.5,1] corresponding to the 94th percentile of the input data distribution.

Following the similar procedure, I approximated relu2 layer with a polynomial $0.18749648x^2 + 0.50046875x + 0.23437646$ corresponding to the input range [-2,2].

2.3.2 Input and output distribution of non-linear layers after approximation

Input and output distributions of the non-linear relu1 layer which is now approximated with a polynomial function are:

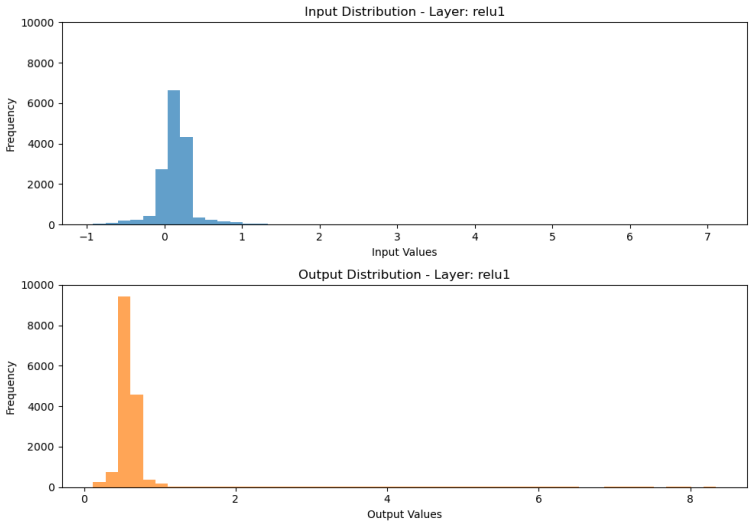


Figure 6: I/O distribution for relu1 after polynomial approximation

Input and output distributions of the non-linear relu2 layer which is now approximated with a polynomial function are:

2.3.3 Results

I compared accuracy values for different polynomial orders in the range of 2-8 and different input ranges. The polynomial approximations mentioned above resulted in the closest-to-original accuracy with the accuracy of **94.040**.

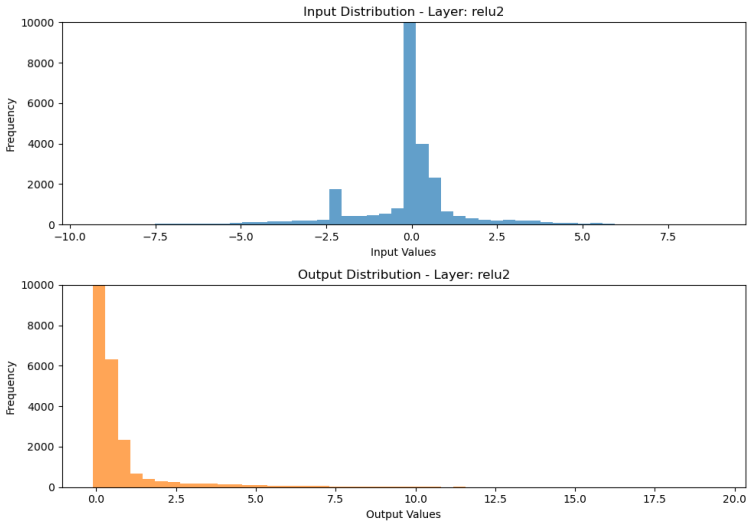


Figure 7: I/O distribution for relu2 after polynomial approximation

2.4 Step-wise polynomial approximation without retraining

1. Approximate the first non-linear function based on the input/output distributions.
2. Re-generate the input/output distributions.
3. Approximate the following non-linear function based on the newly generated input/output distributions.
4. Return to step 2 and repeat these steps until all non-linear functions have been approximated.

2.4.1 Approximation functions

Based on the original input distribution, the first approximation that I did was on the first non-linear function (relu1). It was approximated with a polynomial $0.08602914x^2 + 0.49189841x + 0.4987775$ which was obtained by running Remez algorithm for the range [-0.5,1] corresponding to the 94th percentile of the input data distribution.

Then, based on the new input distribution of the second non-linear function (relu2), the following polynomial approximation was applied: $0.23930681x^2 + 0.51551783x + 0.18492584$ which corresponds to the range [-2.71155783, 2.27614852] ($95th$ percentile of the data distribution).

P.S. I tested accuracy for 99th percentile as well, but it was resulting in a lower test accuracy than 95th percentile.

2.4.2 Input and output distribution of non-linear layers after approximation

Input and output distributions of the non-linear (now polynomial) functions of the approximated model for each iteration of the approximation are presented below:

Iteration 0: Input/output distributions before any approximation:
First non-linear function (relu1):

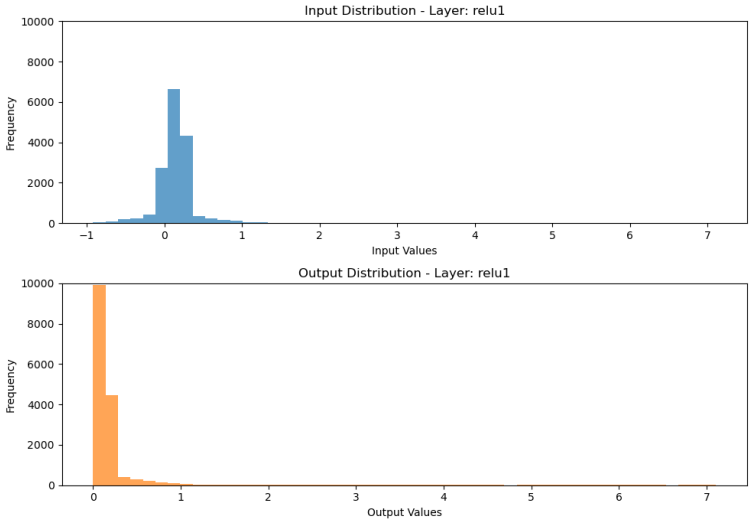


Figure 8: Input and output distribution for relu1

Second non-linear function (relu2):

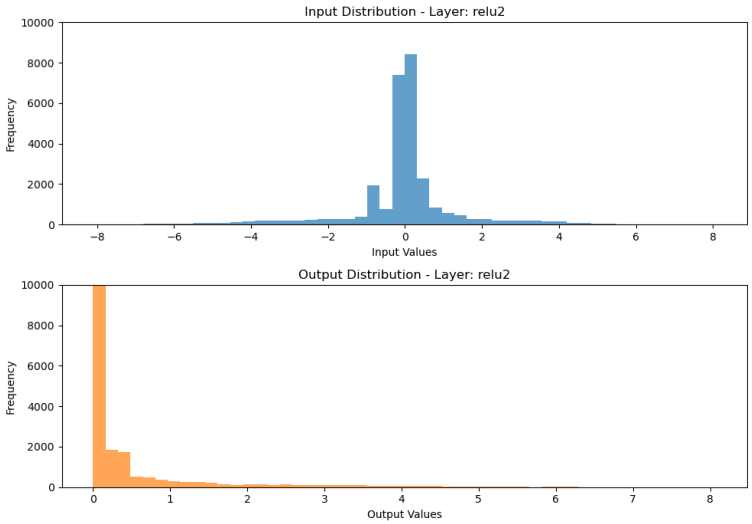


Figure 9: Input and output distribution for relu2

Iteration 1: Input/output distributions after the first approximation:
First non-linear function (relu1):

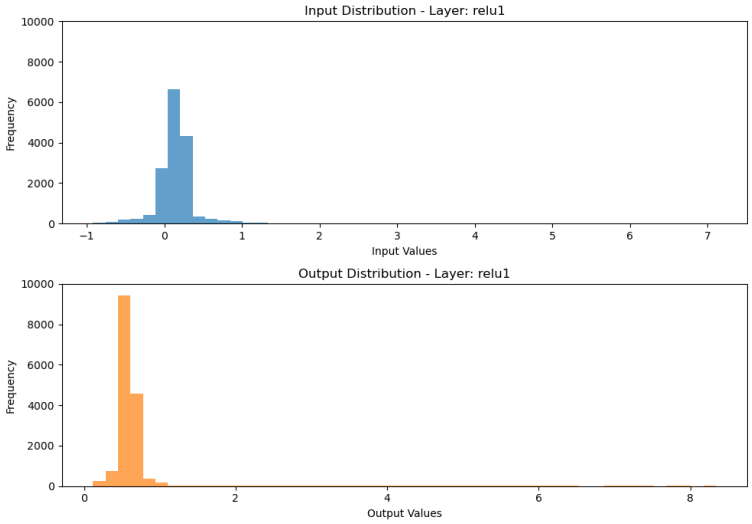


Figure 10: Input and output distribution for relu1

Second non-linear function (relu2):

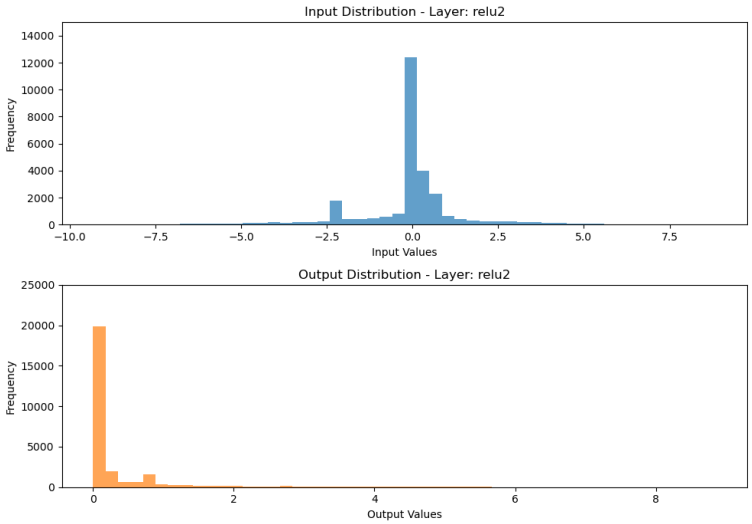


Figure 11: Input and output distribution for relu2

Iteration 2: Input/output distributions after the second approximation:
First non-linear function (relu1):

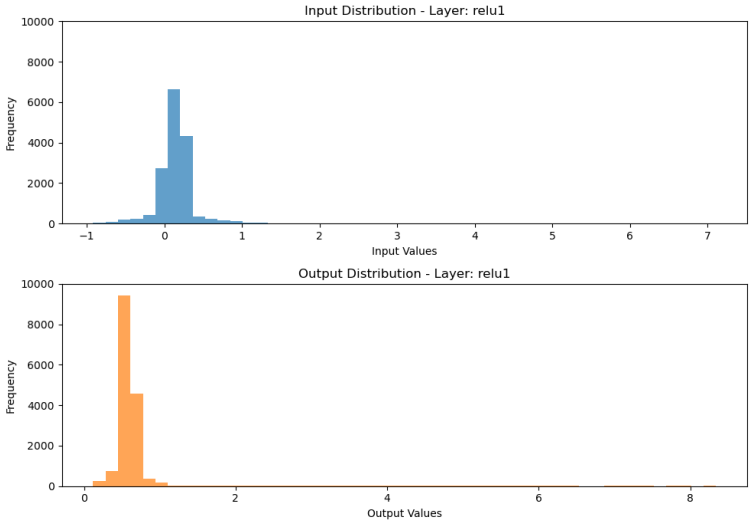


Figure 12: Input and output distribution for relu1

Second non-linear function (relu2):

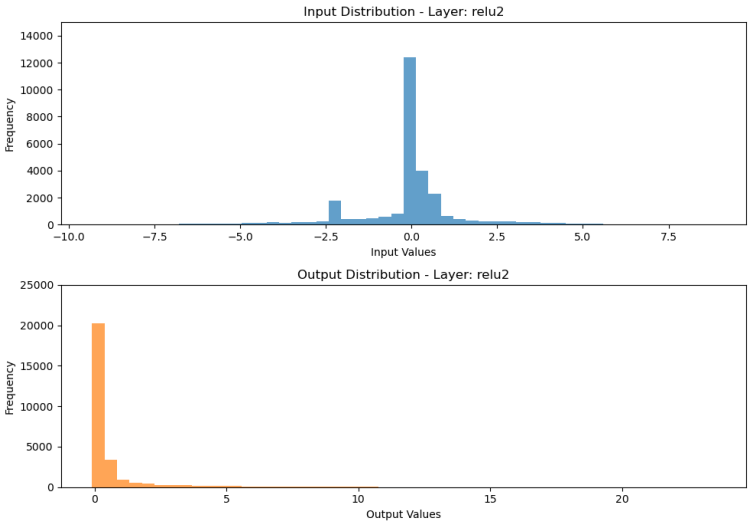


Figure 13: Input and output distribution for relu2

2.4.3 Results

The test accuracy after all layers have been approximated is **92.150**.

2.5 Step-wise polynomial approximation with retraining

1. Approximate the first non-linear function based on the input/output distributions.
2. Re-train the remaining model.
3. Re-generate the input/output distributions.
4. Approximate the following non-linear function based on the newly generated input/output distributions.
5. Return to step 2 and repeat these steps until all non-linear functions have been approximated.

How to re-train the model:
Consider that your model is $y = f_n(...f_2(f_1(x))...)$, where x is the input, y is the output, and f_i is a non-linear function. For simplicity, I omitted the linear parts.

First, approximate $f_1(\cdot)$ with a polynomial function $p_1(\cdot)$ based on the input/ouput distribution of the $f_1(\cdot)$. Then, create a model starting from the following layer, i.e., $y = f_n(...f_2(z)...)$, where $z = p_1(x)$, load the weights from the previous iteration (for the first iteration, load the weights of the original model), and re-train the model. Repeat the process until all layers have been approximated.

There is no retraining after the last layer has been approximated.

2.5.1 Approximation functions

Based on the original input distribution, the first approximation that I did was on the first non-linear function (relu1). It was approximated with a polynomial $0.08602914x^2 + 0.49189841x + 0.4987775$ which was obtained by running Remez algorithm for the range [-0.5,1] corresponding to the 94th percentile of the input data distribution.

Then, I created a new model and trained it using 50 epochs(best validation accuracy was 96.92) and based on the new input distribution obtained, I approximated the second non-linear function (relu2) with the following polynomial: $0.21345536x^2 + 0.06231828x + 0.00366859$ which corresponds to the range [-14.59009852, 1.0379966] ($95th$ percentile of the data distribution).

2.5.2 Input and output distribution of non-linear layers after approximation

Input and output distributions of the non-linear (now polynomial) functions of the approximated model for each iteration of the approximation are presented below:

Iteration 0: Input/output distributions before any approximation:
First non-linear function (relu1):

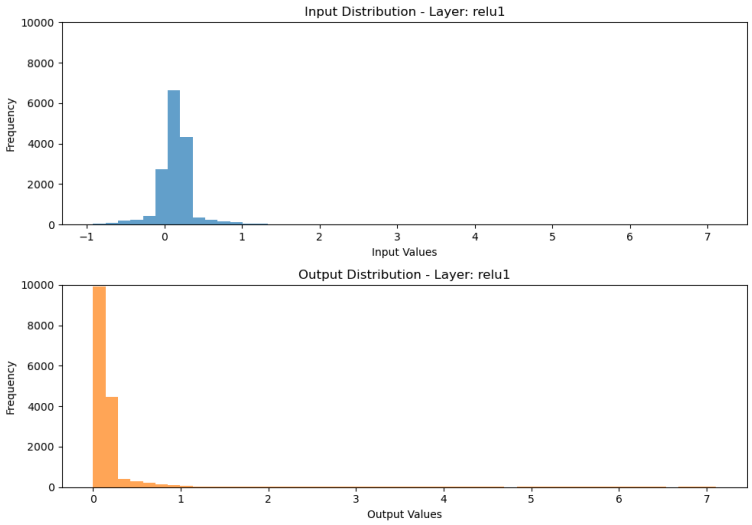


Figure 14: Input and output distribution for relu1

Second non-linear function (relu2):

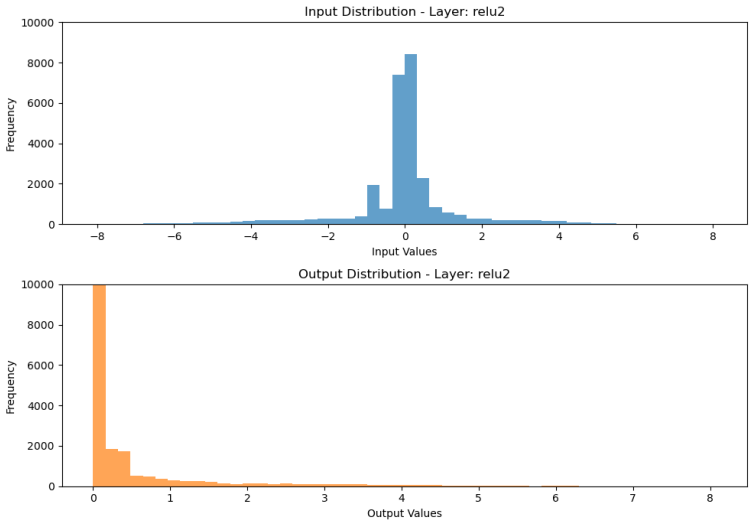


Figure 15: Input and output distribution for relu2

Iteration 1: Input/output distributions after the first approximation and after re-training:

The new model that we created has only one non-linear function. Input/output distributions of this non-linear function (relu2) are:

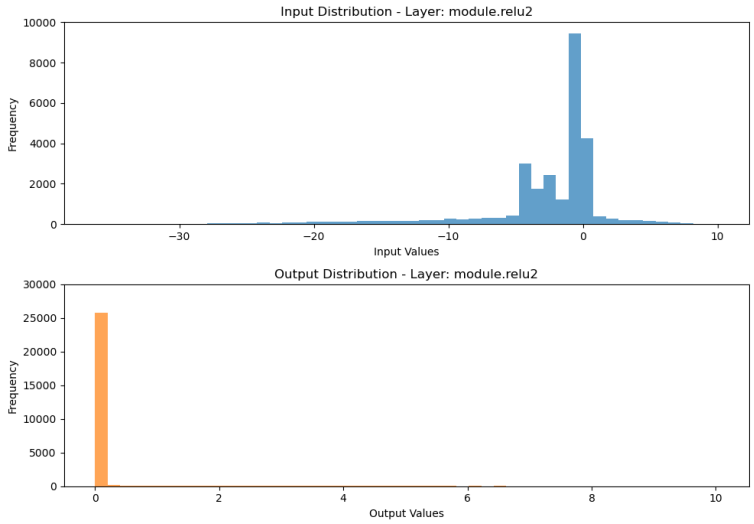


Figure 16: Input and output distribution for relu2

Iteration 2: Input/output distributions after the second approximation and after re-training:

After the second re-training, the new model created doesn't have any non-linear function anymore.

2.5.3 Results

Test accuracy after all layers have been approximated and re-trained was **95.200**.

3 LeNet-5 with CIFAR-10

3.1 Model description

This LeNet-5 model was created following the original LeNet-5 model's architecture. It consists of a two consecutive layers (layer 1 and layer 2) each comprising of a convolutional, Batchnorm, ReLU, and MaxPool, followed by alternating convolutional and dense layers. Input and output channels for the first convolutional layer were taken as 3 and 6, while for the second convolutional layer in the layer 2 they were computed as 6 and 16, respectively. The kernel size was 5, input and output dimensions for the first dense layer after layer 2 were computed as 400 and 120, for the second dense layer - 120 and 84, and for the last dense layer - 84 and 10.

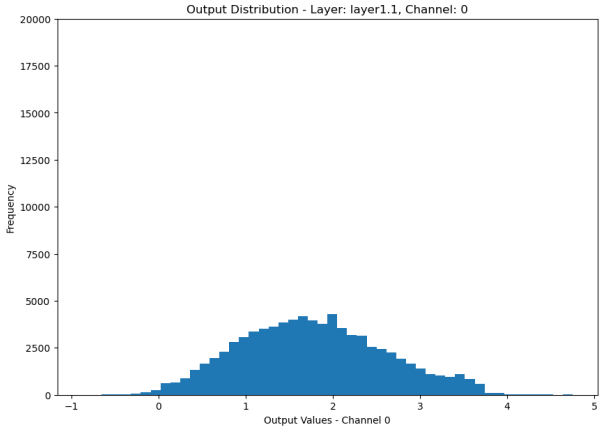
I trained this model using 300 epochs on the CIFAR-10 dataset. The hyperparameters used when training were: batch size was 128 for the training dataset and 100 - for the validation dataset , learning rate was 0.001, optimizer was Adam.

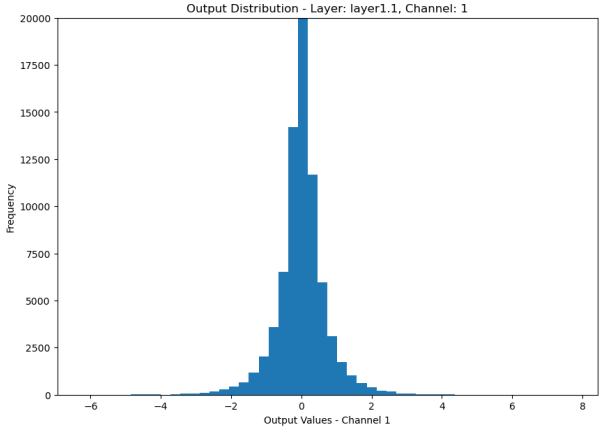
3.2 Original

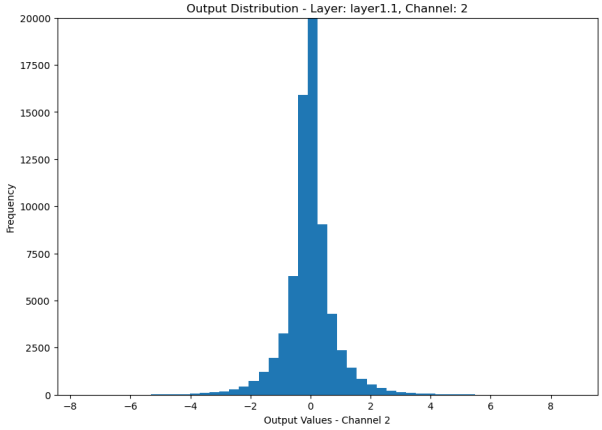
3.2.1 Input and output distribution of non-linear functions

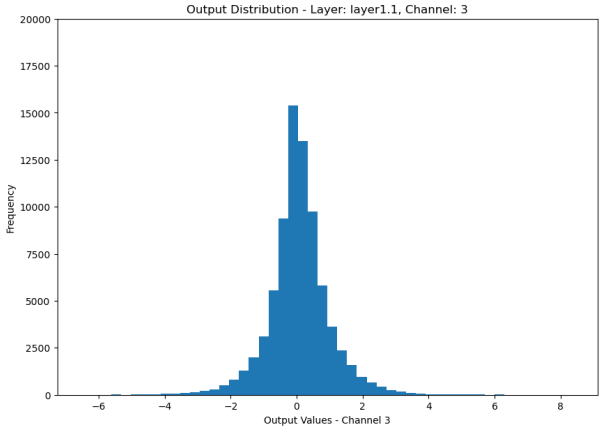
Output distribution for the BatchNorm layer that precedes the ReLU1 layer corresponds to the input distribution for the ReLU1 layer. Since I was looking at the distribution of the ReLU layers channel-by-channel, it was easier for me to plot the output distribution of the layer that precedes the ReLU layer as this output should correspond to the input of the ReLU layer. Based on the model architecture, the number of input channels for the first non-linear function should be 6, while for the second - 16.

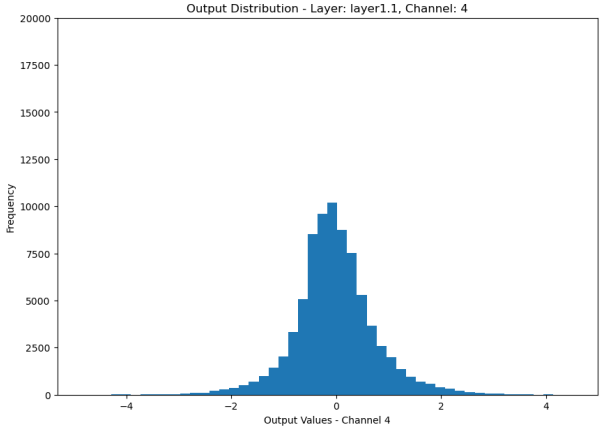
Output distributions for the layer that precedes the first non-linear function (ReLU1) channel-by-channel (this corresponds to the input distribution of the first non-linear layer):

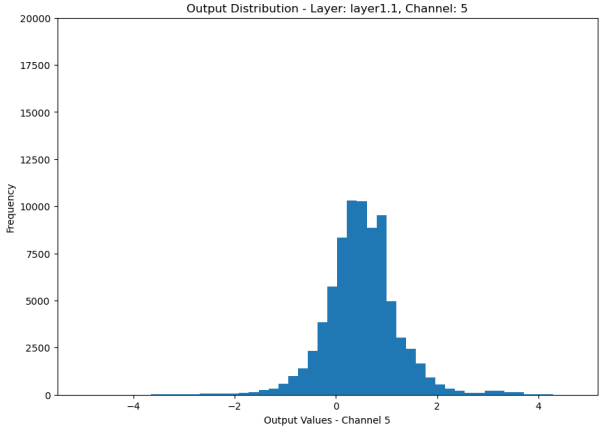

Output Distribution - Layer: layer1.1, Channel: 0
Frequency
Output Values - Channel 0


Output Distribution - Layer: layer1.1, Channel: 1
Frequency
Output Values - Channel 1

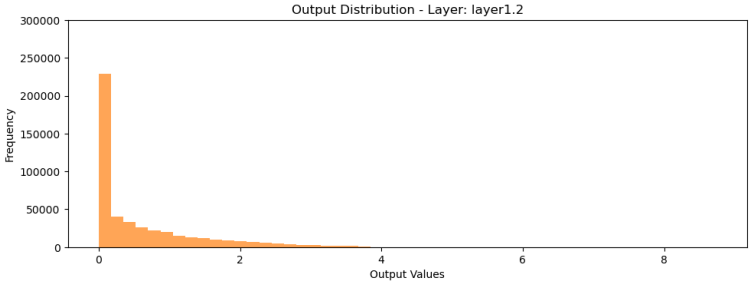

Output Distribution - Layer: layer1.1, Channel: 2
Frequency
Output Values - Channel 2


Output Distribution - Layer: layer1.1, Channel: 3
Frequency
Output Values - Channel 3

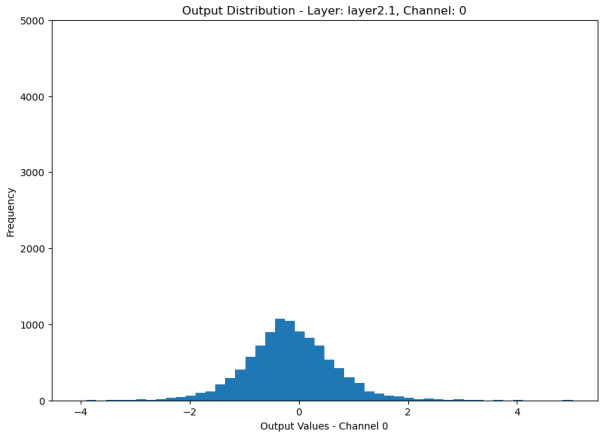

Output Distribution - Layer: layer1.1, Channel: 4
Frequency
Output Values - Channel 4

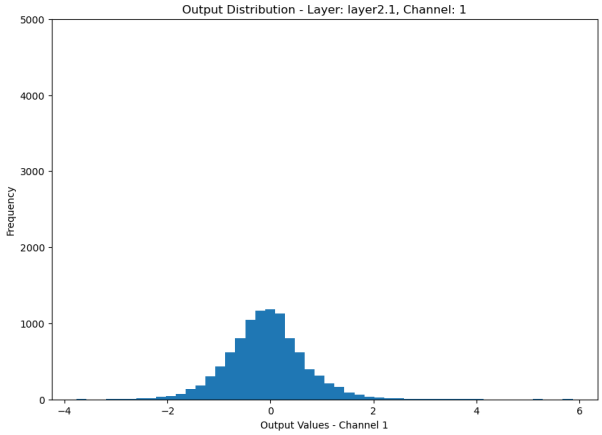

Output Distribution - Layer: layer1.1, Channel: 5
Frequency
Output Values - Channel 5

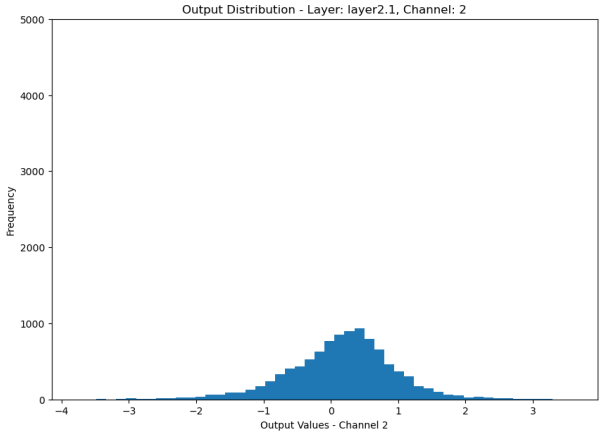
Output distribution for the first non-linear layer:

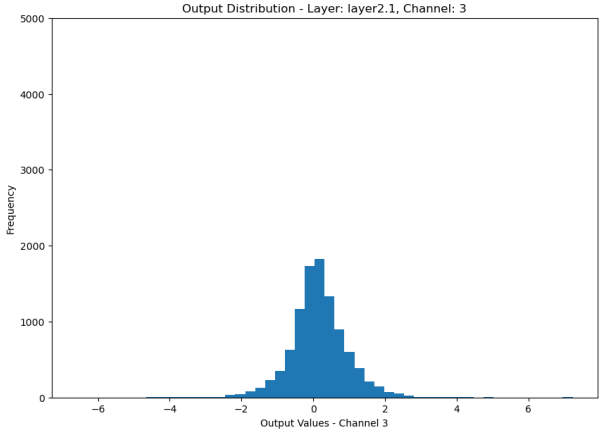


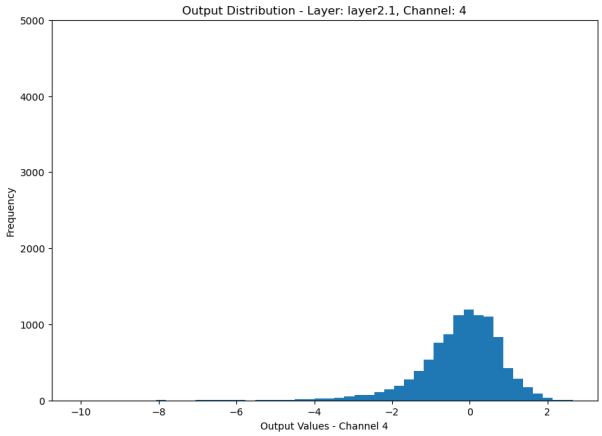
Output distributions for the layer that precedes the second non-linear function (ReLU2) channel-by-channel(this corresponds to the input distribution of the second non-linear layer):

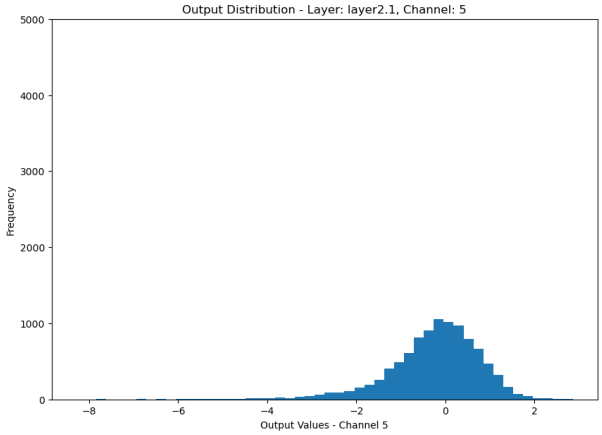


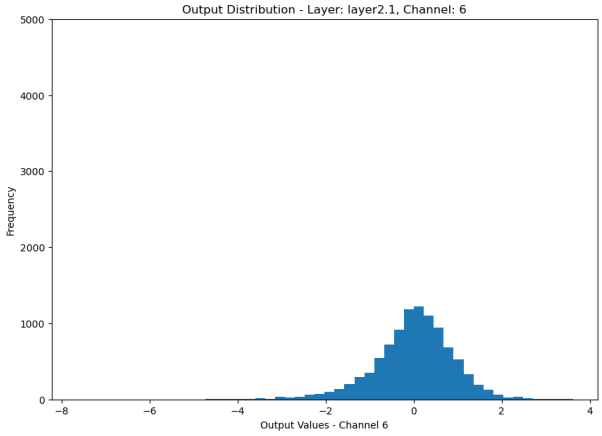


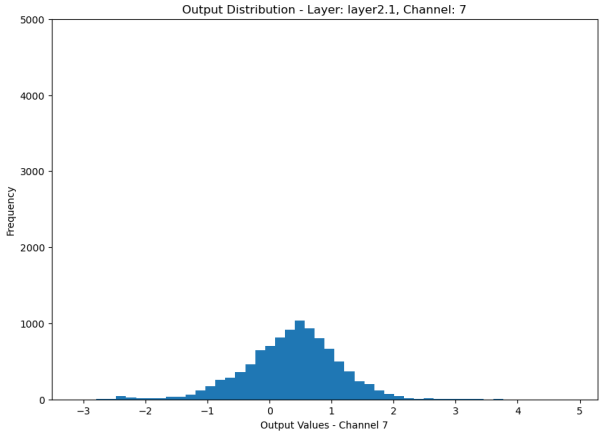

Output Distribution - Layer: layer2.1, Channel: 2
Frequency
Output Values - Channel 2
5000
4000
3000
2000
1000
0
−4
−3
−2
−1
0
1
2
3

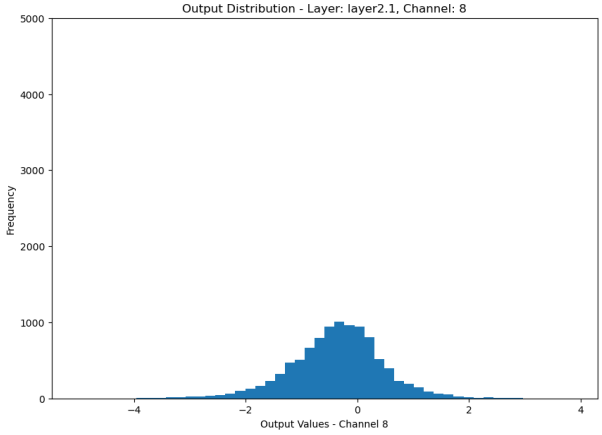

Output Distribution - Layer: layer2.1, Channel: 3
Frequency
Output Values - Channel 3
5000
4000
3000
2000
1000
0
−6
−4
−2
0
2
4
6

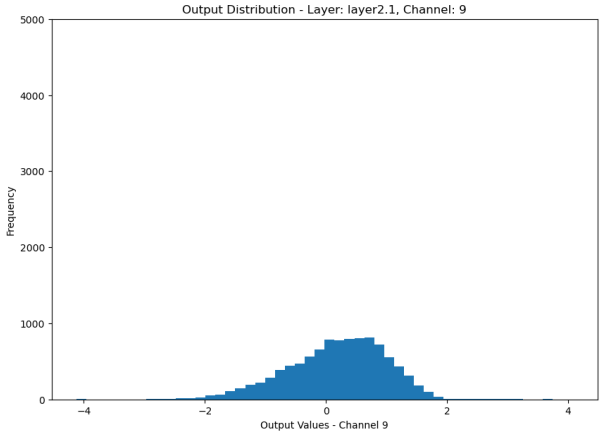

Output Distribution - Layer: layer2.1, Channel: 4
Frequency
Output Values - Channel 4
5000
4000
3000
2000
1000
0
−10
−8
−6
−4
−2
0
2

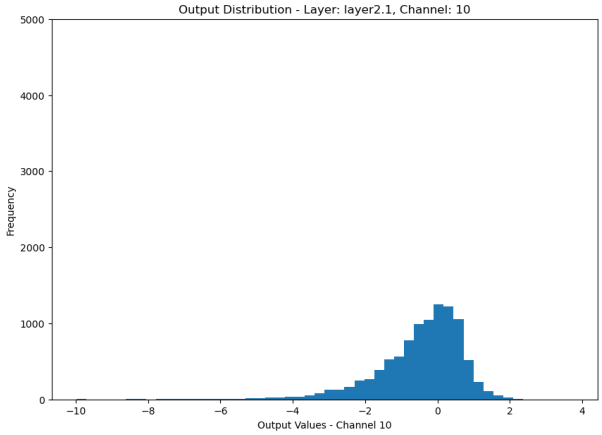

Output Distribution - Layer: layer2.1, Channel: 5
Frequency
Output Values - Channel 5
5000
4000
3000
2000
1000
0
−8
−6
−4
−2
0
2

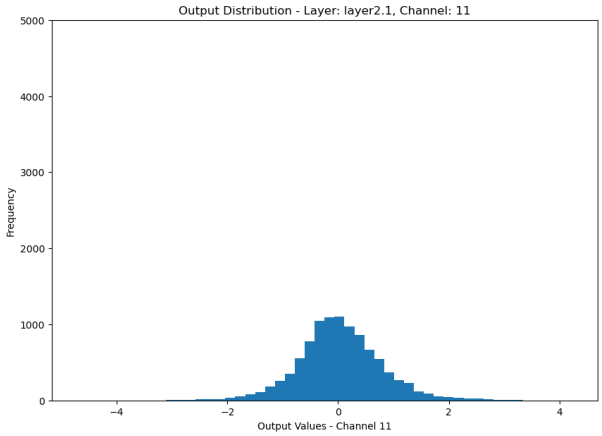

Output Distribution - Layer: layer2.1, Channel: 6
Frequency
Output Values - Channel 6

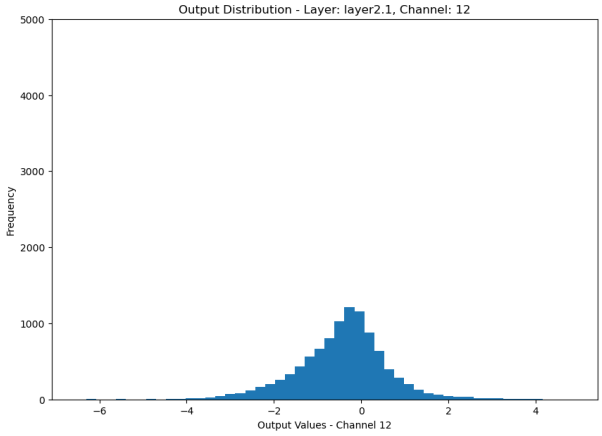

Output Distribution - Layer: layer2.1, Channel: 7
Frequency
Output Values - Channel 7

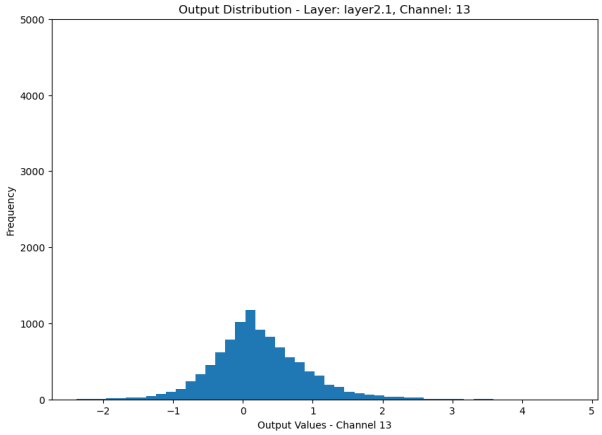

Output Distribution - Layer: layer2.1, Channel: 8
Frequency
Output Values - Channel 8

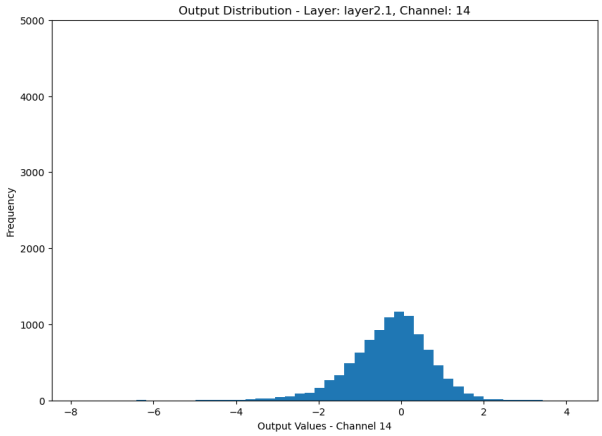

Output Distribution - Layer: layer2.1, Channel: 9
Frequency
Output Values - Channel 9

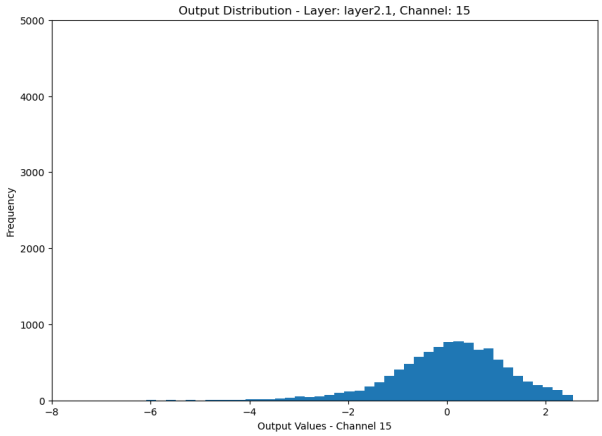

Output Distribution - Layer: layer2.1, Channel: 10
Frequency
Output Values - Channel 10


Output Distribution - Layer: layer2.1, Channel: 11
Frequency
Output Values - Channel 11

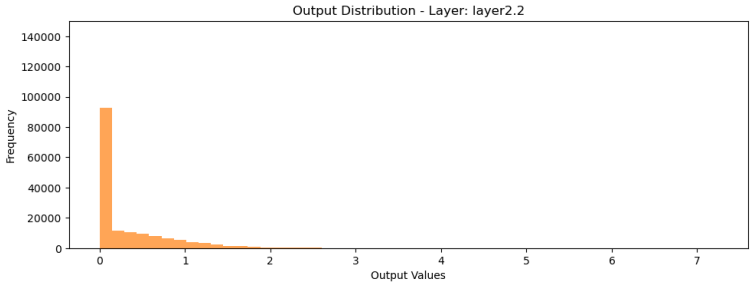

Output Distribution - Layer: layer2.1, Channel: 12
Frequency
Output Values - Channel 12


Output Distribution - Layer: layer2.1, Channel: 13
Frequency
Output Values - Channel 13

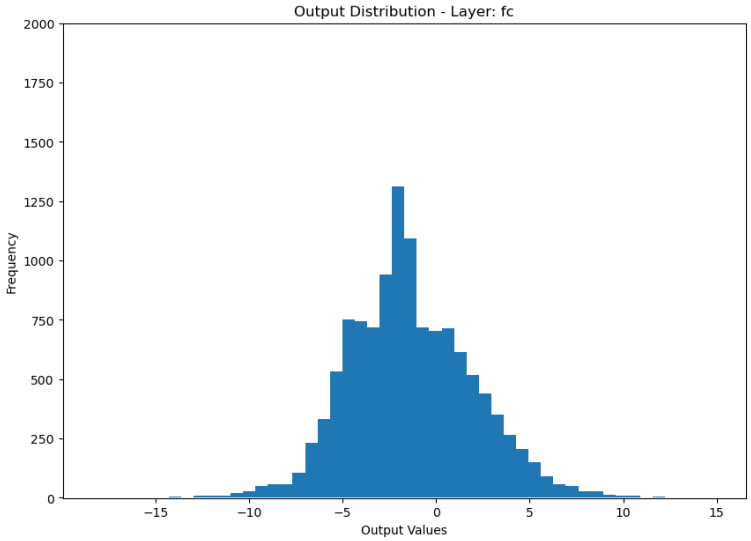

Output Distribution - Layer: layer2.1, Channel: 14
Frequency
Output Values - Channel 14



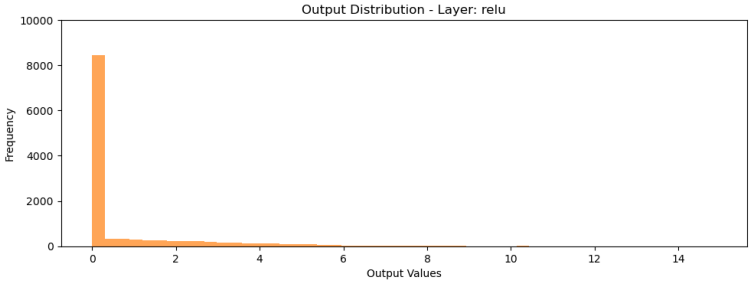
Output distribution for the second non-linear layer:



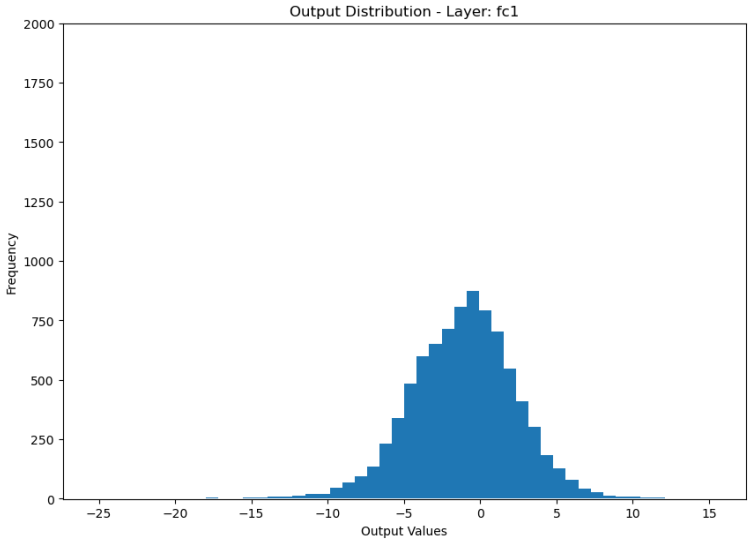
Output distribution for the fc layer that precedes the third non-linear function (this corresponds to the input distribution of the third non-linear layer):



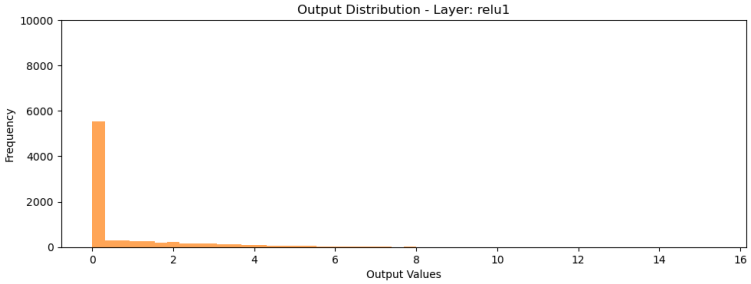
Output distribution for the third non-linear layer:



Output distribution for the fc1 layer that precedes the fourth non-linear function (this corresponds to the input distribution of the fourth non-linear layer):



Output distribution for the fourth non-linear layer:



3.2.2 Results

The best validation accuracy achieved when training the original model with 300 epochs was **74.410**.

3.3 All-at-once polynomial approximation without retraining

3.3.1 Approximation functions

Based on the output distribution of layer 1 channel 0, I approximated the ReLU function corresponding to channel 0 with a polynomial $(-8.9750817e-17)x^2 + (1.0000000e+00)x + (-1.6052492e-17)$ which was obtained by running Remez algorithm for the range [0.46376634, 3.27582252] which corresponds to the 95th percentile of the output data distribution.

Following the similar procedure, the approximations for the other channels of layer 1 were as follows:

Channel 1:
- Polynomial coefficients: [0.10073615 0.49954451 0.43624437]
- Polynomial order:2
- Range:[-1.07138963 1.07731597]

Channel 2:
- Polynomial coefficients: [0.11590477 0.50467029 0.37938594]
- Polynomial order:2
- Range:[-1.26440466 1.2029154]

Channel 3:
- Polynomial coefficients: [0.14231845 0.48606275 0.31052254]
- Polynomial order:2
- Range:[-1.37343976 1.60773209]

Channel 4:
- Polynomial coefficients: [0.11719888 0.49506172 0.37522415]
- Polynomial order:2
- Range:[-1.21394296 1.28097382]

Channel 5:
- Polynomial coefficients: [0.13338011 0.56409888 0.25045225]
- Polynomial order:2
- Range: [-0.58237541 1.64808909]

All the ranges I took for layer 1 were from the 95th percentile of the output data distribution and the polynomials were of degree 2.

Based on the output distribution of layer 2 channel 0, I approximated the ReLU function corresponding to channel 0 with a polynomial $0.11459485x^3 + 0.44900159x^2 + 0.38779171x + 0.07596087$ which was obtained by running Remez algorithm for the range [-1.44495196, 1.13801765] which corresponds to the 95th percentile of the output data distribution.

Following the similar procedure, the approximations for the other channels of layer 2 were as follows:

Channel 1:
- Polynomial coefficients:[0.11437956 0.48127193 0.38481549 0.02918413]
- Polynomial order:3
- Range: [-1.27707236 1.17838301]

Channel 2:
- Polynomial coefficients: [0.11859016 0.52625624 0.37171335 -0.03784024]
- Polynomial order:3
- Range:[-1.20771136 1.35504889]

Channel 3:
- Polynomial coefficients: [0.11131682 0.56092161 0.40091402 -0.09393557]
- Polynomial order:3
- Range: [-1.098682 1.48922923]

Channel 4:
- Polynomial coefficients: [0.12067533 0.42357506 0.37225169 0.09478388]
- Polynomial order:3
- Range:[-2.12187916 1.19289846]

Channel 5:
- Polynomial coefficients: [0.11795087 0.42472728 0.38084074 0.09857797]
- Polynomial order:3
- Range: [-2.14700266 1.16358748]

Channel 6:
- Polynomial coefficients: [0.1335584 0.45793959 0.3315968 0.04686932]
- Polynomial order:3
- Range: [-1.61581851 1.33573909]

Channel 7:
- Polynomial coefficients:[0.09241037 0.5766818 0.48611289 -0.1618587]
- Polynomial order:3
- Range: [-0.91316916 1.51683826]

Channel 8:

- Polynomial coefficients: [0.09385589 0.42643658 0.47868909 0.1543786]
- Polynomial order:3
- Range: [-1.763784 0.92272198]

Channel 9:
- Polynomial coefficients: [0.12336962 0.5144427 0.35654557 -0.01939235]
- Polynomial order:3
- Range: [-1.27954985 1.36190616]

Channel 10:
- Polynomial coefficients:[0.11914514 0.46435696 0.39307288 0.09083016]
- Polynomial order:3
- Range: [-2.74951476 0.94233208]

Channel 11:
- Polynomial coefficients: [0.11383551 0.52694279 0.38730147 -0.04211426]
- Polynomial order:3
- Range:[-1.15817505 1.30378051]

Channel 12:
- Polynomial coefficients: [0.11430667 0.43036365 0.39346369 0.10240064]
- Polynomial order:3
- Range: [-2.25549117 1.11285835]

Channel 13:
- Polynomial coefficients:[0.0791474 0.5763401 0.56756538 -0.22023227]
- Polynomial order:3
- Range: [-0.78160655 1.39680256]

Channel 14:
- Polynomial coefficients: [0.11666354 0.42324551 0.38506188 0.10161004]
- Polynomial order:3
- Range: [-1.92495916 1.15328768]

Channel 15:
- Polynomial coefficients: [0.17790828 0.47504963 0.24770337 0.01599586]
- Polynomial order:3
- Range: [-2.0236241 1.81560229]

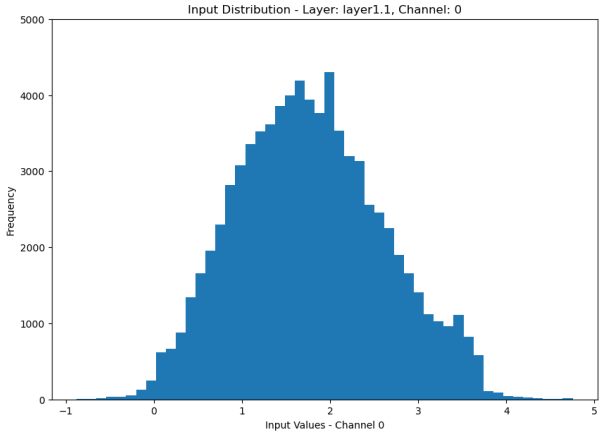
All the ranges I took for layer 2 were from the 95th percentile of the output data distribution and the polynomials were of degree 3.

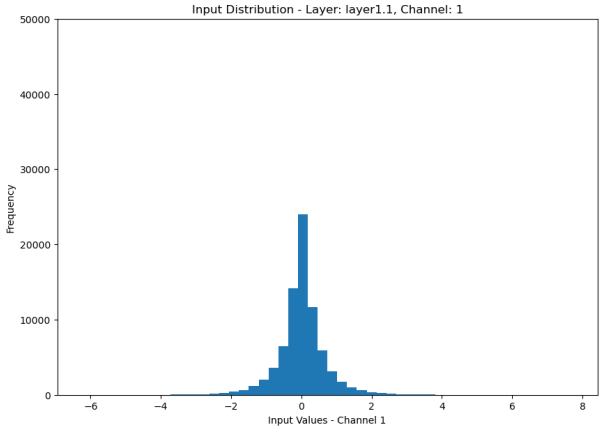
The polynomial approximation for the third non-linear function ReLU was $0.10022889x^3 + 0.45220605x^2 + 0.45763687x + 0.12770209$ which was obtained by running Remez algorithm for on the range corresponding to the 60th percentile of the input data distribution.

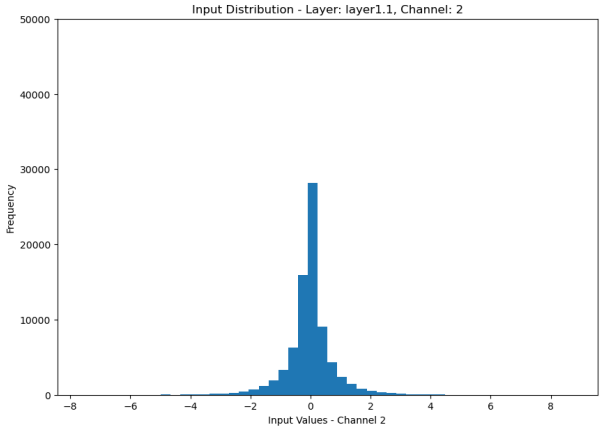
The final non-linear function was approximated with the polynomial $0.17983002x^3 + 0.44493856x^2 + 0.25278848x + 0.03992602$ which was obtained by running Remez algorithm for on the range corresponding to the 80th percentile of the input data distribution.

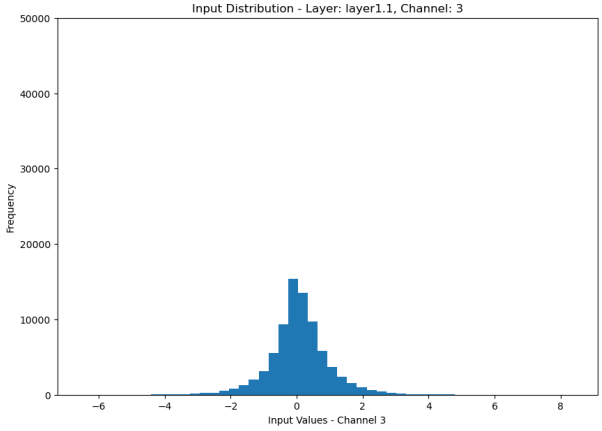
3.3.2 Input and output distribution of non-linear layers after approximation

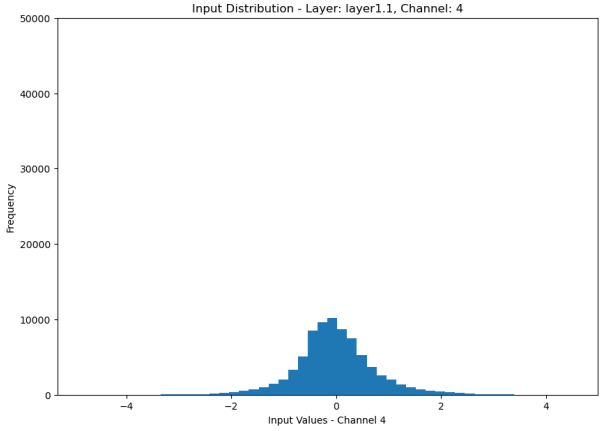
Input distributions of the first non-linear layer which is now approximated channel-by-channel with polynomial functions are as follows:

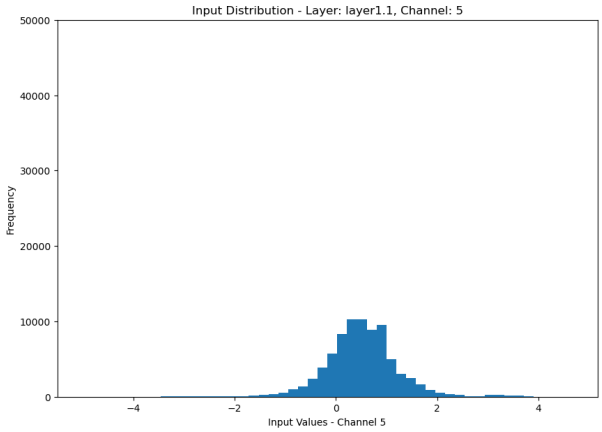




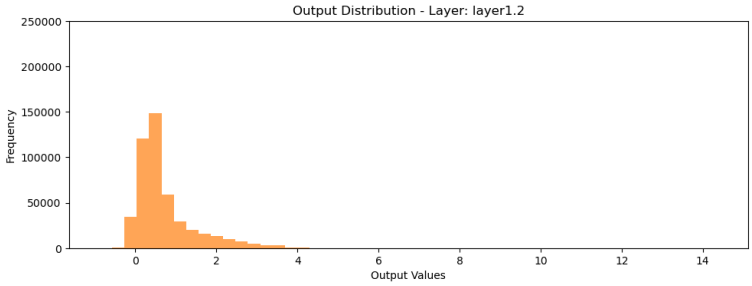

Input Distribution - Layer: layer1.1, Channel: 2
Frequency
Input Values - Channel 2


Input Distribution - Layer: layer1.1, Channel: 3
Frequency
Input Values - Channel 3

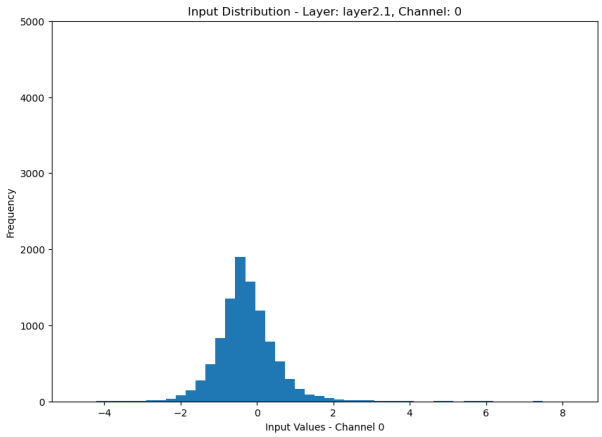

Input Distribution - Layer: layer1.1, Channel: 4
Frequency
Input Values - Channel 4

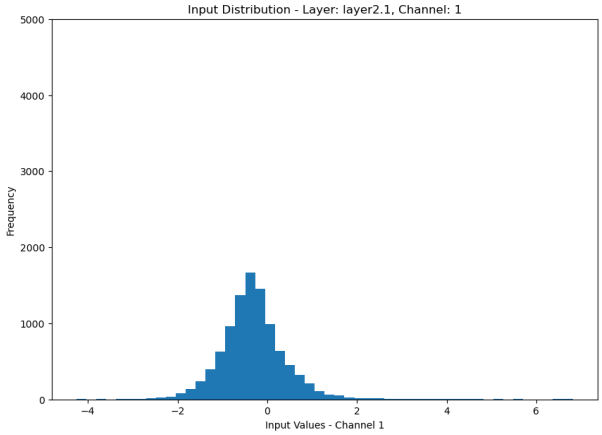


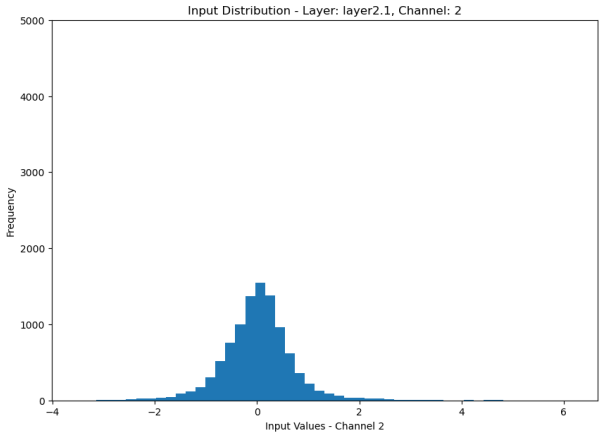
Output distribution of the first non-linear layer which is now approximated channel-by-channel with polynomial functions is as follows:

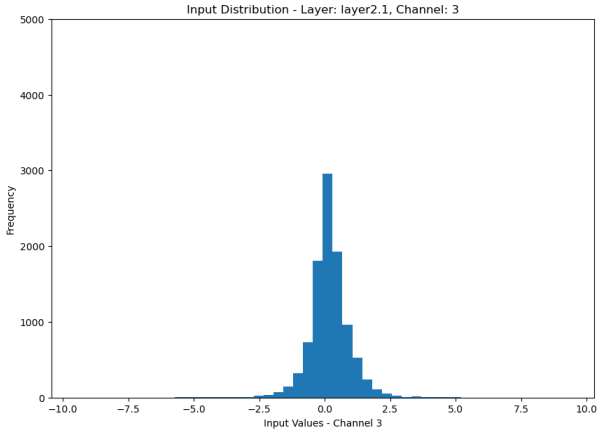


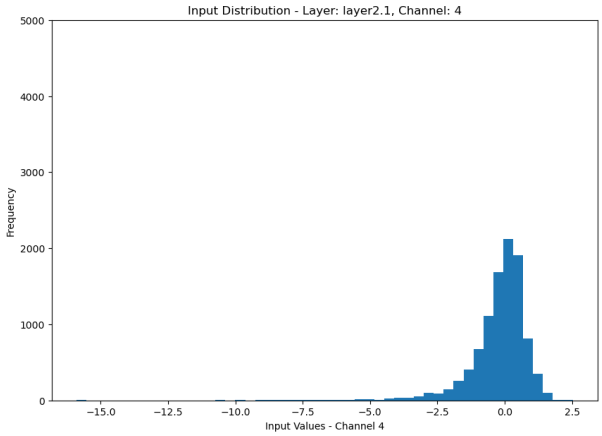
Input distributions of the second non-linear layer which is now approximated channel-by-channel with polynomial functions are as follows:

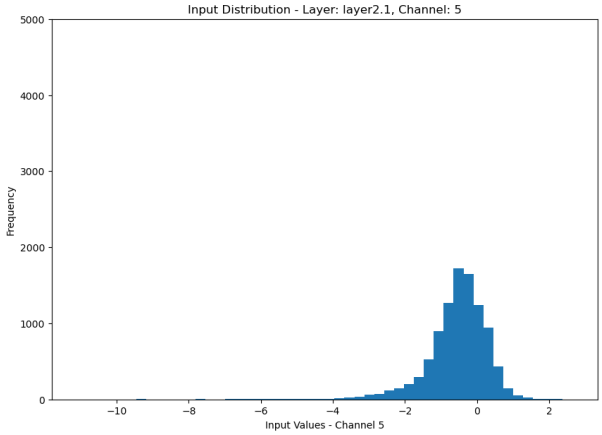


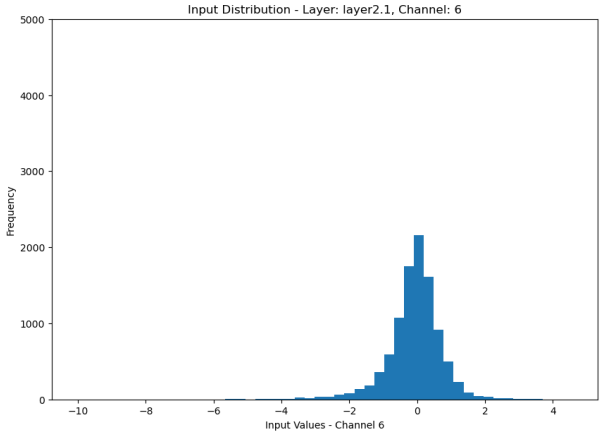


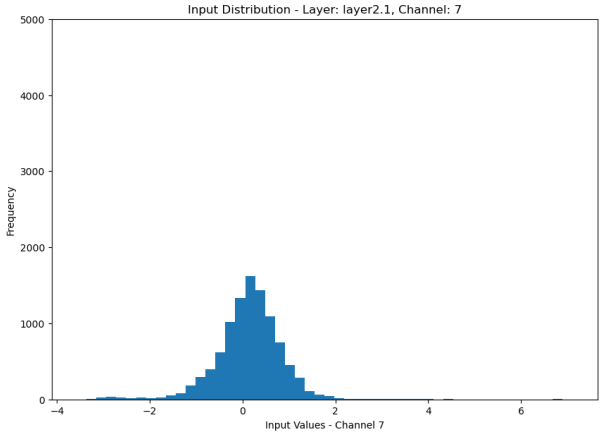

Input Distribution - Layer: layer2.1, Channel: 2
Frequency
Input Values - Channel 2
5000
4000
3000
2000
1000
0
−4
−2
0
2
4
6

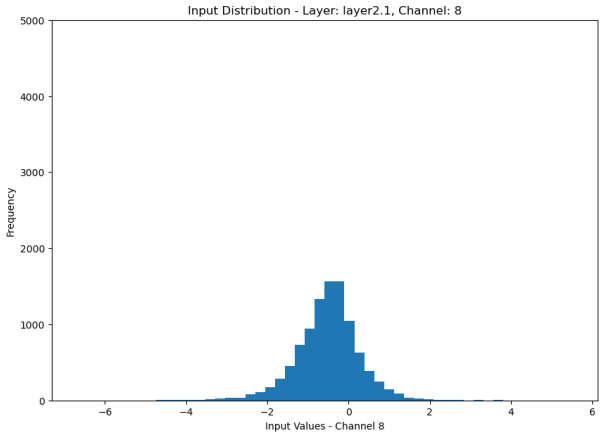

Input Distribution - Layer: layer2.1, Channel: 3
Frequency
Input Values - Channel 3
5000
4000
3000
2000
1000
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

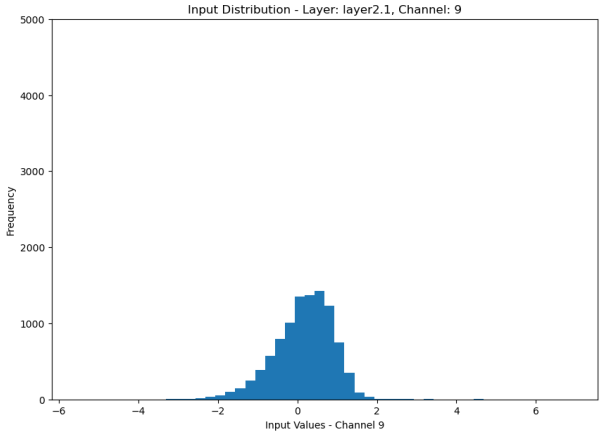

Input Distribution - Layer: layer2.1, Channel: 4
Frequency
Input Values - Channel 4
5000
4000
3000
2000
1000
0
−15.0
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5

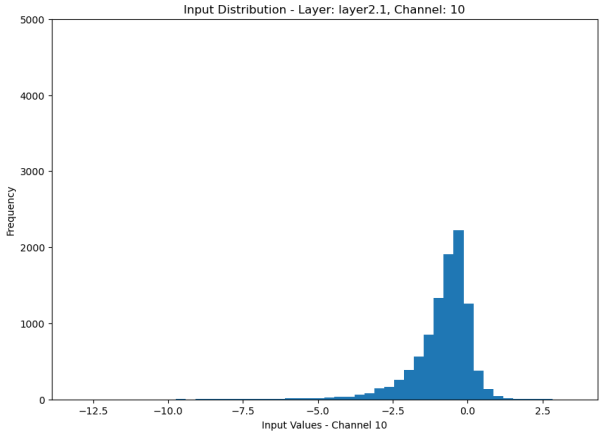

Input Distribution - Layer: layer2.1, Channel: 5
Frequency
Input Values - Channel 5
5000
4000
3000
2000
1000
0
−10
−8
−6
−4
−2
0
2

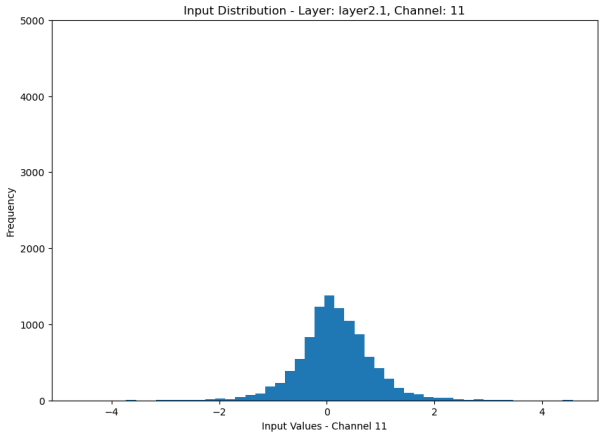

Input Distribution - Layer: layer2.1, Channel: 6
Frequency
Input Values - Channel 6

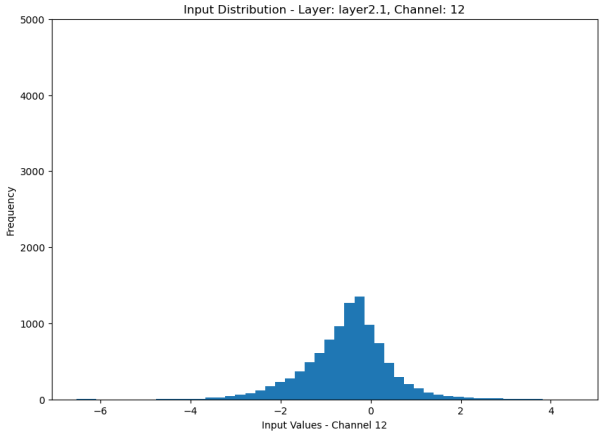

Input Distribution - Layer: layer2.1, Channel: 7
Frequency
Input Values - Channel 7

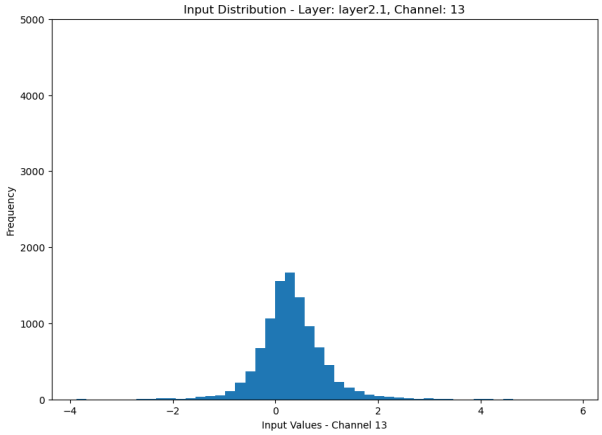

Input Distribution - Layer: layer2.1, Channel: 8
Frequency
Input Values - Channel 8

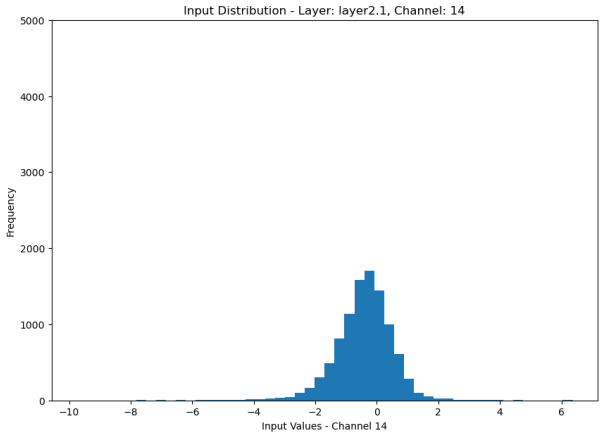

Input Distribution - Layer: layer2.1, Channel: 9
Frequency
Input Values - Channel 9

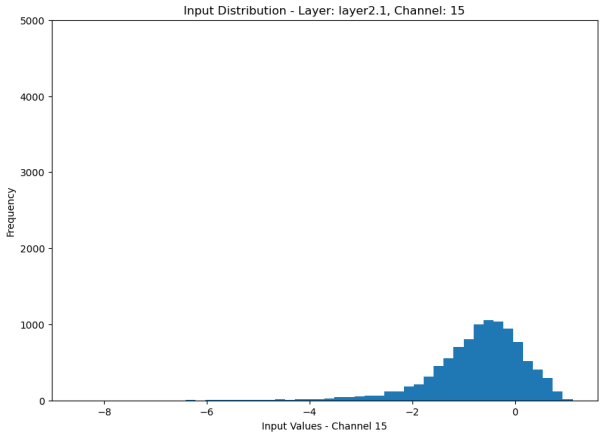

Input Distribution - Layer: layer2.1, Channel: 10
Frequency
Input Values - Channel 10


Input Distribution - Layer: layer2.1, Channel: 11
Frequency
Input Values - Channel 11

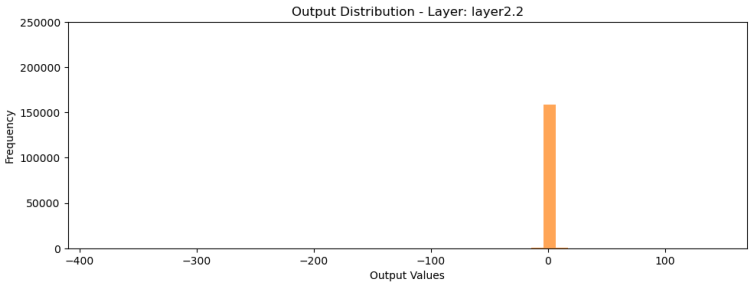

Input Distribution - Layer: layer2.1, Channel: 12
Frequency
Input Values - Channel 12


Input Distribution - Layer: layer2.1, Channel: 13
Frequency
Input Values - Channel 13


Input Distribution - Layer: layer2.1, Channel: 14
Frequency
Input Values - Channel 14



Output distribution of the second non-linear layer which is now approximated channel-by-channel with polynomial functions is as follows:



Input and output distribution of the third non-linear layer which is now approximated with a polynomial function is as follows:

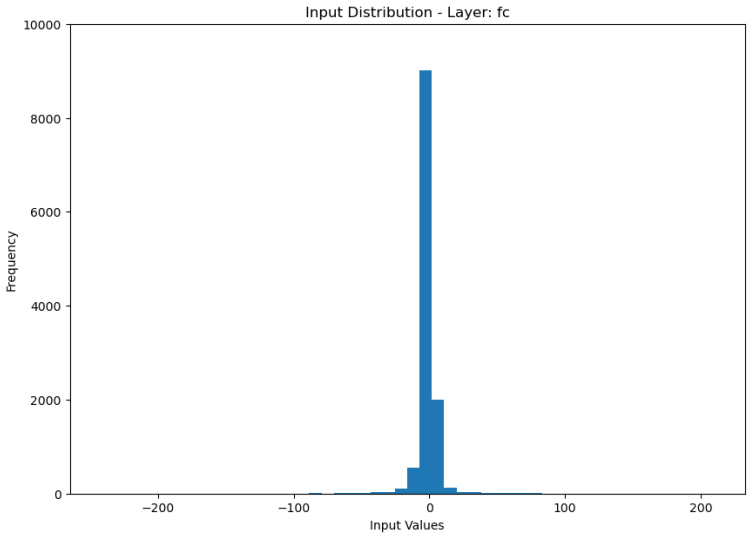


Figure 17: Input distribution for the 3rd non-linear layer after polynomial approximation

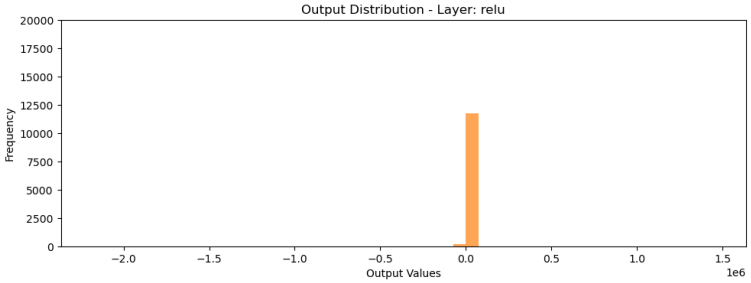


Figure 18: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and output distribution of the fourth non-linear layer which is now approximated with a polynomial function is as follows:

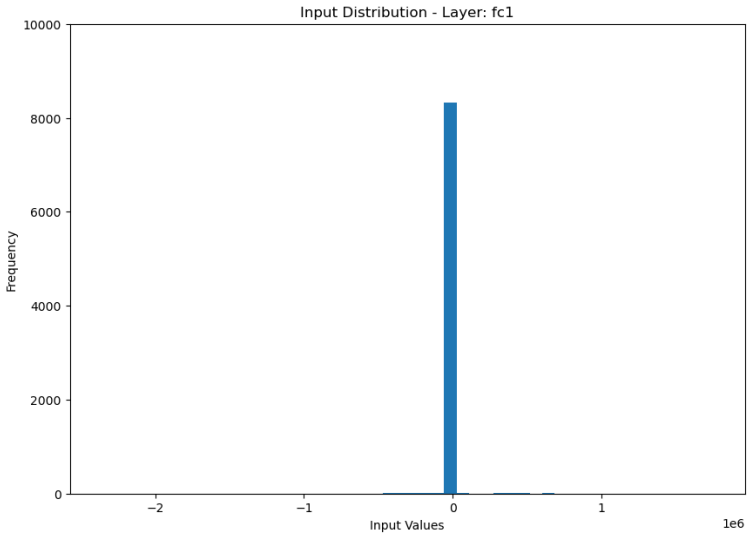


Figure 19: Input distribution for the 4th non-linear layer after polynomial approximation

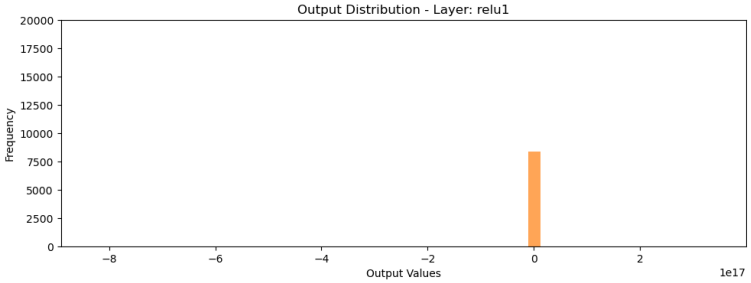


Figure 20: Output distribution for the 4th non-linear layer after polynomial approximation

3.3.3 Results

I compared the accuracy values for different polynomial orders in the range of 2-8 and different input ranges. The polynomial approximations mentioned above resulted in the closest-to-original accuracy I was able to obtain, with the accuracy of **31.680**.

3.4 Step-wise polynomial approximation without retraining

1. Approximate the first non-linear function based on the input/output distributions.
2. Re-generate the input/output distributions.
3. Approximate the following non-linear function based on the newly generated input/output distributions.
4. Return to step 2 and repeat these steps until all non-linear functions have been approximated.

3.4.1 Approximation functions

For the first non-linear layer, I used the polynomial approximations from the Section 3.3.1 because the input distribution for this layer didn't change.

For the second non-linear layer, I chose the below approximations based on the new input distribution using the Remez algorithm:

Channel 0:
- Polynomial coefficients:[0.07891308 0.47116248 0.48315975]
- Polynomial order: 2
- Range: [-0.9352953 0.3831088]
- Percentile: 85th

Channel 1:
- Polynomial coefficients:[0.12264847 0.52427468 0.36479421]
- Polynomial order: 2
- Range: [-1.41847011 0.89982887]
- Percentile: 95th

Channel 2:
- Polynomial coefficients:[0.18864926 0.4877378 0.23395884]
- Polynomial order: 2
- Range: [-1.84925799 2.11979212]
- Percentile: 99th

Channel 3:
- Polynomial coefficients:[5.972380923147114e-11, 0.9999999991482925, 2.5119429869491228e-09]
- Polynomial order: 2
- Range: [0.03526179,0.28692786]
- Percentile: 60th

Channel 4:
- Polynomial coefficients:[0.17503140610757423, 0.508317191245044, 0.24532546721440143]
- Polynomial order: 2
- Range: [-2.02341901,1.02935507]
- Percentile: 95th

Channel 5:
- Polynomial coefficients:[0.09243195152784954, 0.2939371737575117, 0.17523350217371844]
- Polynomial order:2
- Range: [-1.52455394,0.33426131]
- Percentile: 90th

Channel 6:
- Polynomial coefficients:[0.12399894103242884, 0.5237681472130017, 0.36056324450103827]
- Polynomial order: 2
- Range: [-1.4327821,1.02699707]
- Percentile: 95th

Channel 7:
- Polynomial coefficients:[1.1299211566395684e-10, 0.999999998504439, 4.136703590784663e-09]
- Polynomial order: 2
- Range: [0.03926453,0.3058726]
- Percentile: 60th

Channel 8:
- Polynomial coefficients:[0.14994534894951458, 0.4636568438405222, 0.24758069613808123]
- Polynomial order: 2
- Range: [-1.79119536,0.70978319]
- Percentile: 95th

Channel 9:
- Polynomial coefficients:[3.337269349756866e-11, 0.999999999674092, 6.747810643581431e-10]
- Polynomial order: 2
- Range: [0.05666251,0.42003211]
- Percentile: 60th

Channel 10:
- Polynomial coefficients:[0.05500764231778849, 0.08656475347055591, 0.02725745679946614]
- Polynomial order: 2
- Range: [-2.74640039,0.23929004]
- Percentile: 95th

Channel 11:
- Polynomial coefficients:[0.4329053269743921, 0.5017126917710021, 0.10152099591720011]

- Polynomial order: 2
- Range: [-4.65819931,4.57429171]
- Percentile: 100th

Channel 12:
- Polynomial coefficients:[0.1835433483605598, 0.4704100449678196, 0.20718346551460062]
- Polynomial order: 2
- Range: [-2.17764969,0.88842591]
- Percentile: 95th

Channel 13:
- Polynomial coefficients:[3.507631792785879e-16, 0.9999999999999961, 7.464104306148783e-15]
- Polynomial order: 2
- Range: [0.04793995,0.54073653]
- Percentile: 70th

Channel 14:
- Polynomial coefficients:[0.1264173908169634, 0.4646881013163209, 0.2947516515608367]
- Polynomial order: 2
- Range: [-1.50842328,0.60011556]
- Percentile: 90th

Channel 15:
- Polynomial coefficients:[0.07661564452893242, 0.1962985447275112, 0.09737094260018599]
- Polynomial order: 2
- Range: [-1.79011205,0.27648284]
- Percentile: 90th

The polynomial approximation for the third non-linear function ReLU was $0.12280088910055952x^3+0.38582881950575865x^2+0.2780240006478511x+0.05551885278121503$ which was obtained by running the Remez algorithm on the range [-3.12654644,0.5115771] corresponding to 75th percentile of the input data distribution.

The final non-linear function was approximated with the polynomial $0.18668402769149345x^3+0.3839898316449557x^2+0.18132630734542046x+0.023735530369107406$ which was obtained by running the Remez algorithm on the range [-4.76822624,0.77526128] corresponding to 60th percentile of the input data distribution.

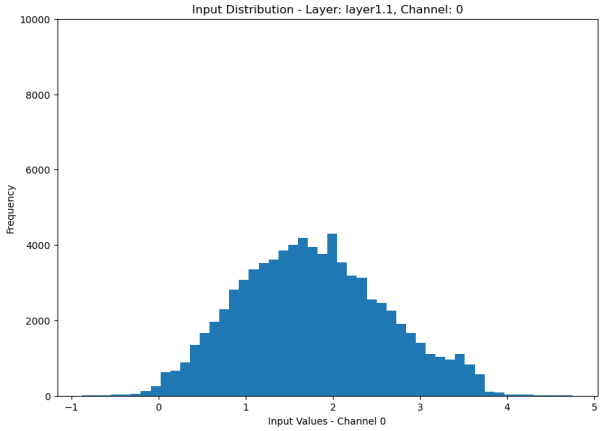
3.4.2 Input and output distribution of non-linear layers after approximation

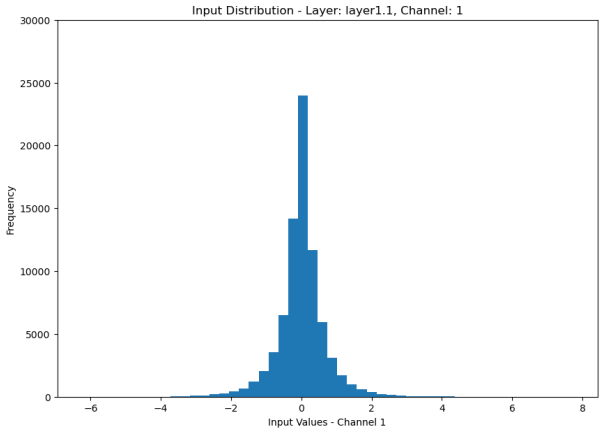
Input and output distributions of the non-linear (now polynomial) functions of the approximated model for each iteration of the approximation are presented below:

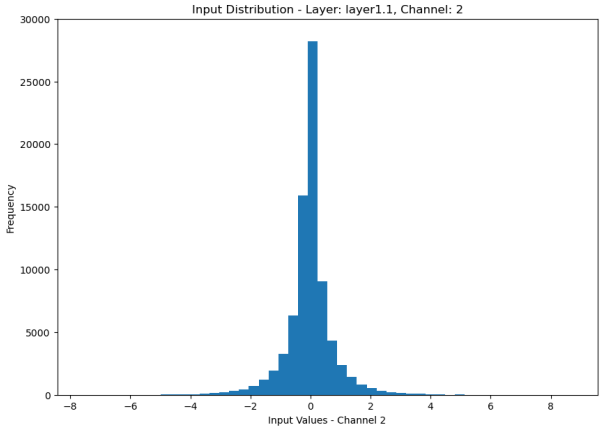
Iteration 0: Input/output distributions before any approximation were provided in the Section 3.2.1.

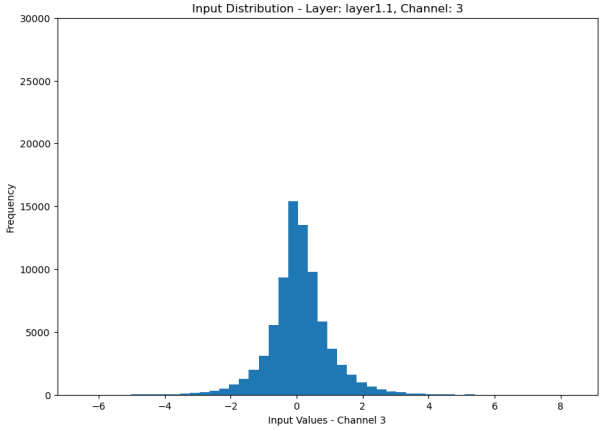
Iteration 1: Input/output distributions after the first approximation:

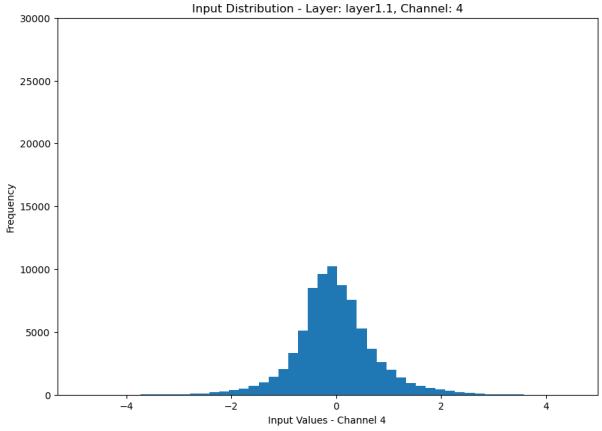
Input distributions of the first non-linear layer which is now approximated channel-by-channel with polynomial functions are as follows:

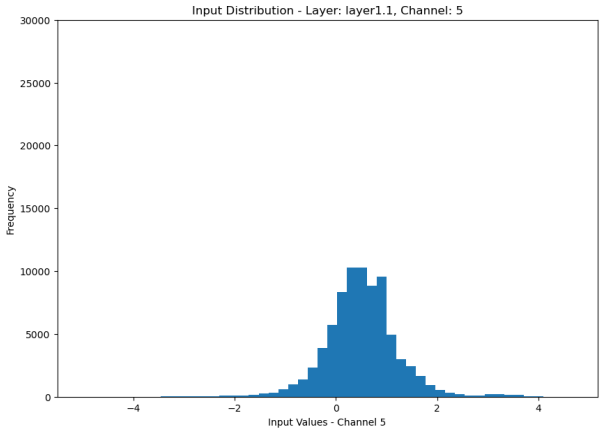




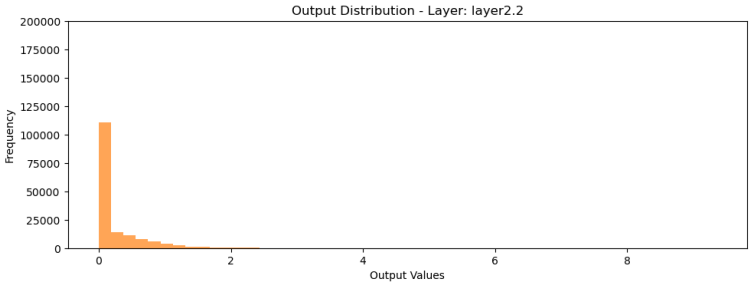

Input Distribution - Layer: layer1.1, Channel: 2
Frequency
Input Values - Channel 2


Input Distribution - Layer: layer1.1, Channel: 3
Frequency
Input Values - Channel 3

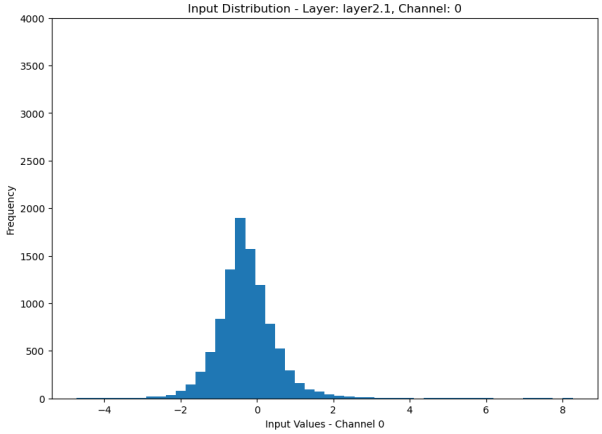

Input Distribution - Layer: layer1.1, Channel: 4
Frequency
Input Values - Channel 4

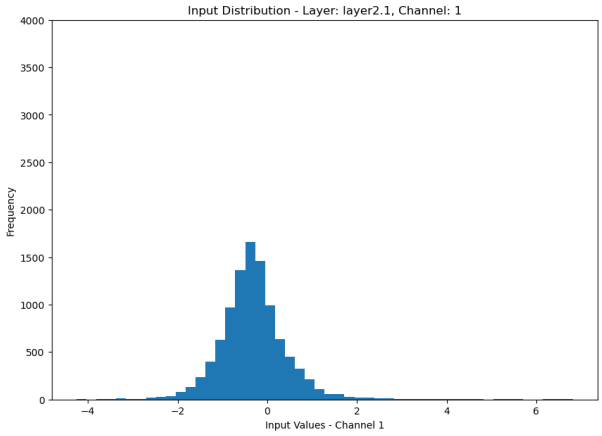


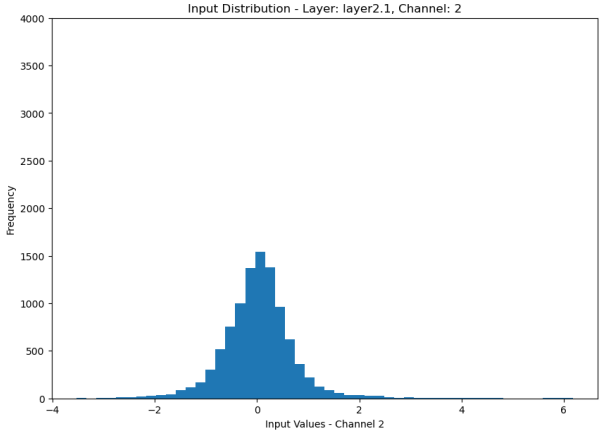
Output distribution of the first non-linear layer which is now approximated channel-by-channel with polynomial functions is as follows:

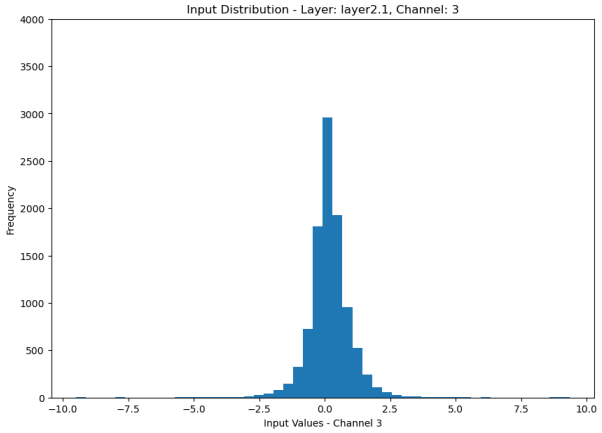


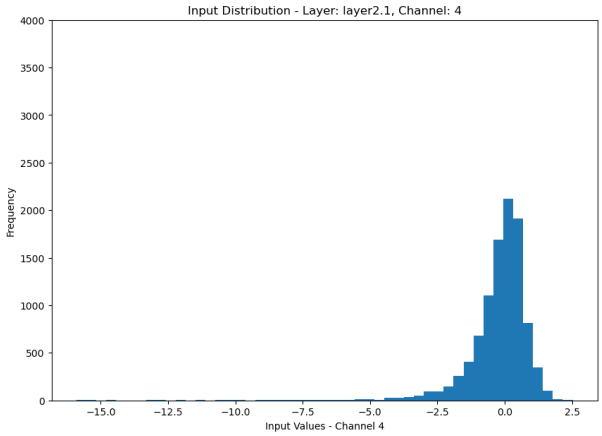
Input distributions of the second non-linear layer:

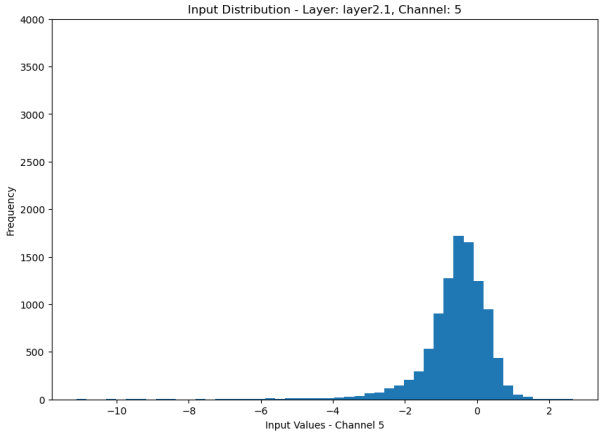


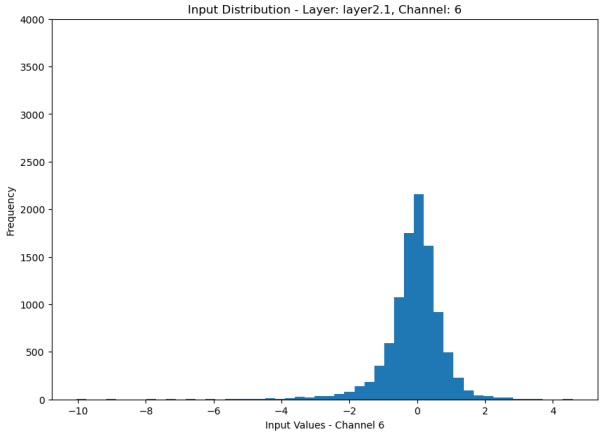


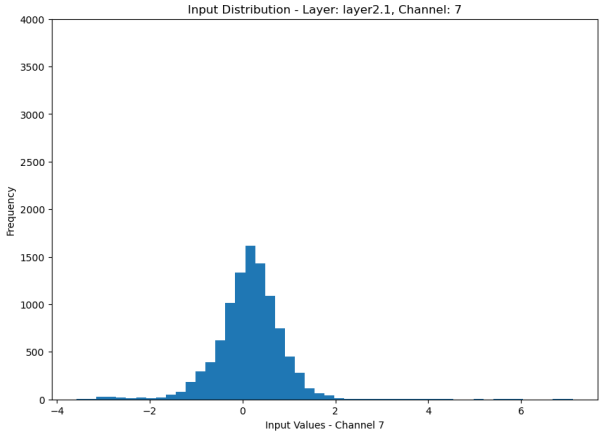

Input Distribution - Layer: layer2.1, Channel: 2
Frequency
Input Values - Channel 2
4000
3500
3000
2500
2000
1500
1000
500
0
−4
−2
0
2
4
6

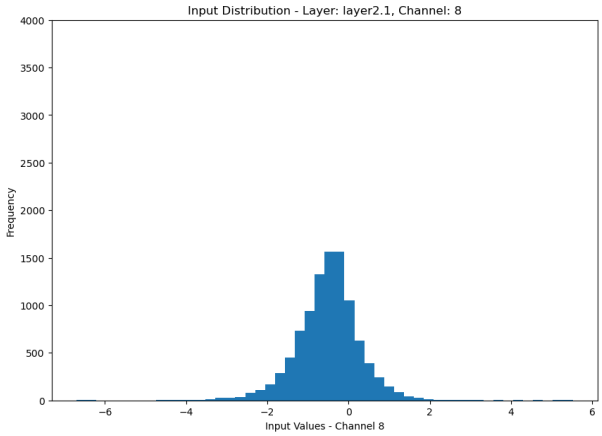

Input Distribution - Layer: layer2.1, Channel: 3
Frequency
Input Values - Channel 3
4000
3500
3000
2500
2000
1500
1000
500
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

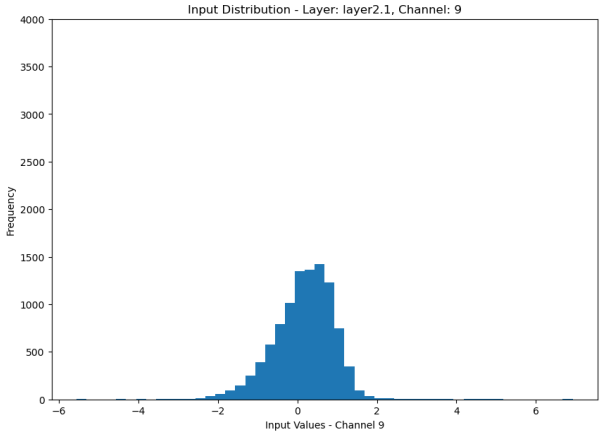

Input Distribution - Layer: layer2.1, Channel: 4
Frequency
Input Values - Channel 4
4000
3500
3000
2500
2000
1500
1000
500
0
−15.0
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5

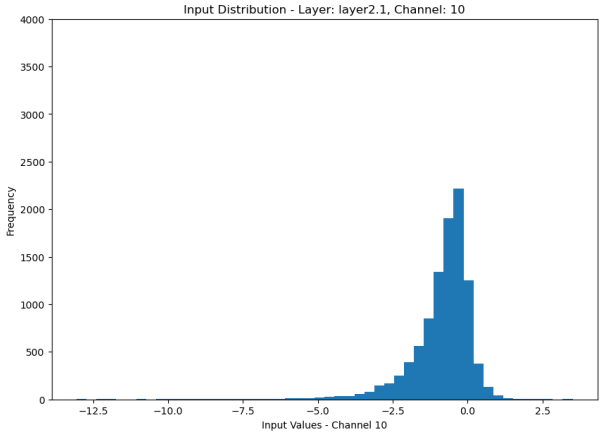

Input Distribution - Layer: layer2.1, Channel: 5
Frequency
Input Values - Channel 5
4000
3500
3000
2500
2000
1500
1000
500
0
−10
−8
−6
−4
−2
0
2

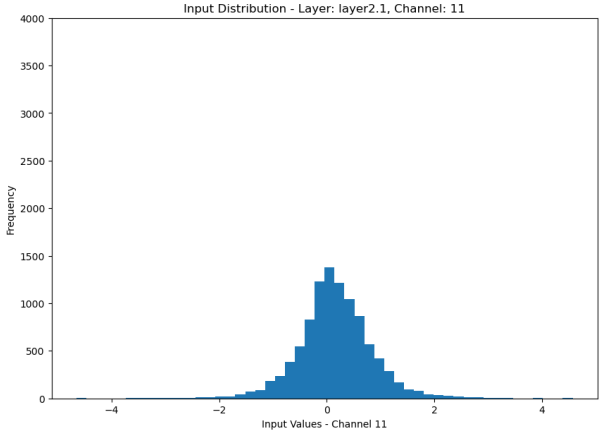

Input Distribution - Layer: layer2.1, Channel: 6
Frequency
Input Values - Channel 6

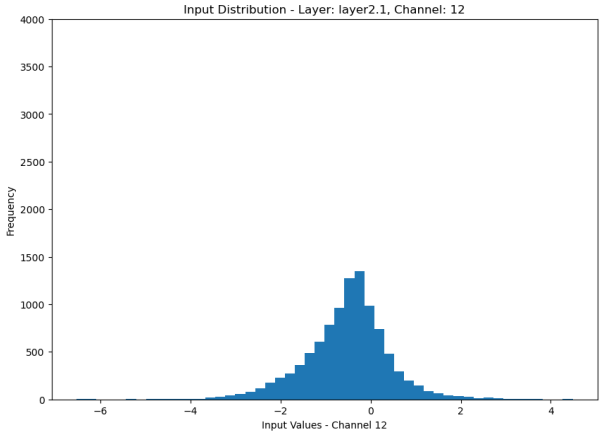

Input Distribution - Layer: layer2.1, Channel: 7
Frequency
Input Values - Channel 7

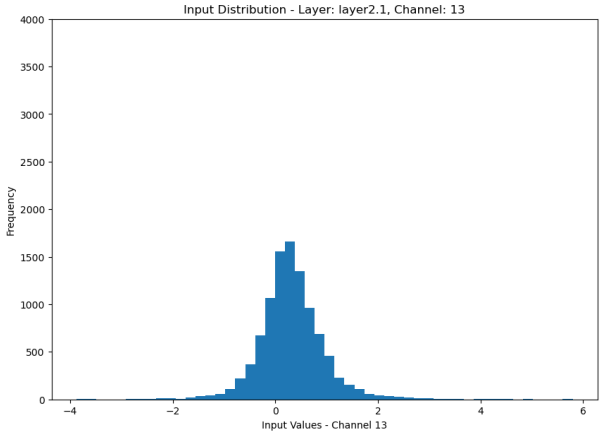

Input Distribution - Layer: layer2.1, Channel: 8
Frequency
Input Values - Channel 8

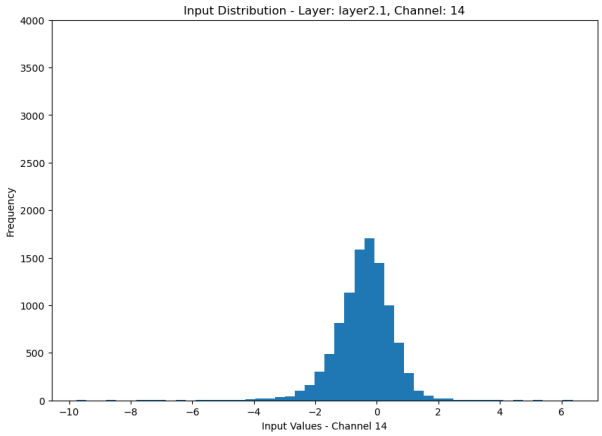

Input Distribution - Layer: layer2.1, Channel: 9
Frequency
Input Values - Channel 9

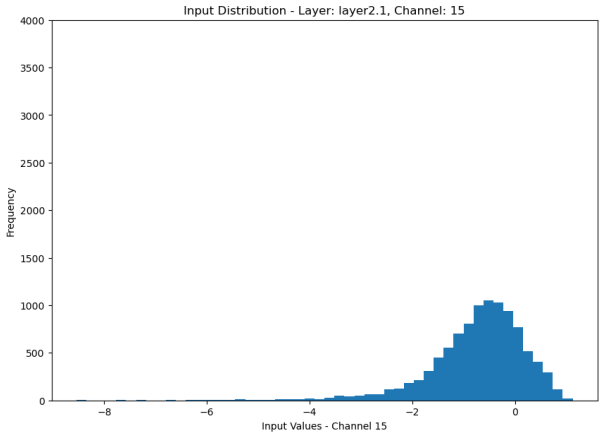

Input Distribution - Layer: layer2.1, Channel: 10
Frequency
Input Values - Channel 10


Input Distribution - Layer: layer2.1, Channel: 11
Frequency
Input Values - Channel 11

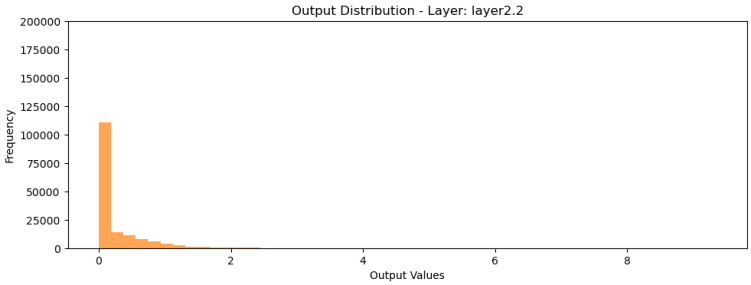

Input Distribution - Layer: layer2.1, Channel: 12
Frequency
Input Values - Channel 12


Input Distribution - Layer: layer2.1, Channel: 13
Frequency
Input Values - Channel 13


Input Distribution - Layer: layer2.1, Channel: 14
Frequency
Input Values - Channel 14



Output distribution of the second non-linear layer:



Input and output distribution of the third non-linear layer:

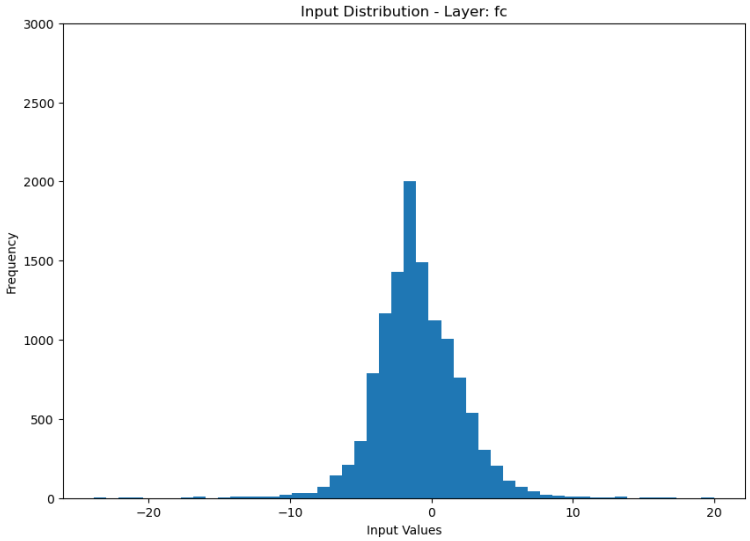


Figure 21: Input distribution for the 3rd non-linear layer after polynomial approximation

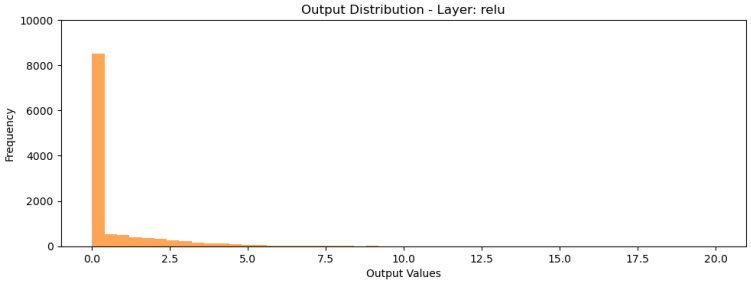


Figure 22: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and output distribution of the fourth non-linear layer:

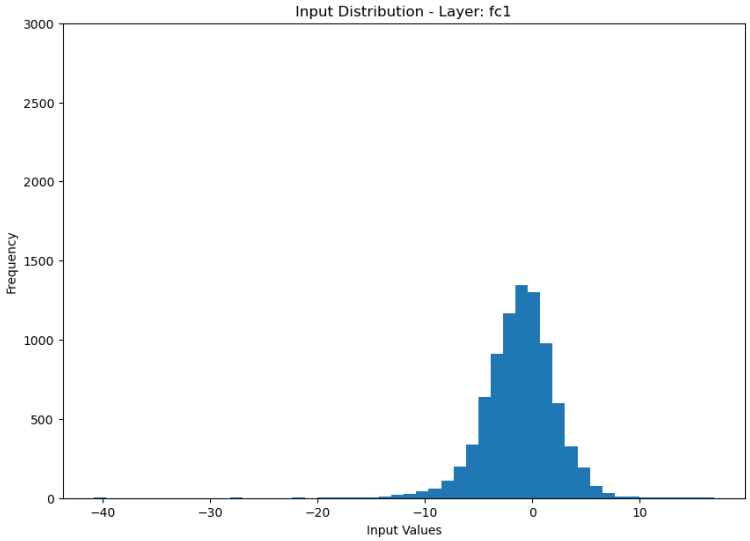


Figure 23: Input distribution for the 4th non-linear layer after polynomial approximation

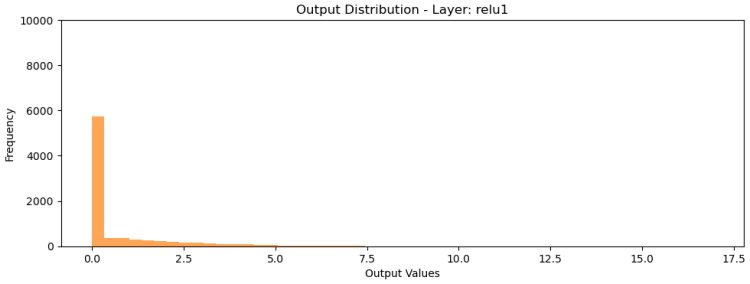


Figure 24: Output distribution for the 4th non-linear layer after polynomial approximation

Iteration 2: Input/output distributions after the second approximation (first and second non-linear layers approximated):

Input distribution of the second non-linear layer stays the same.
Output distribution of the second non-linear layer:

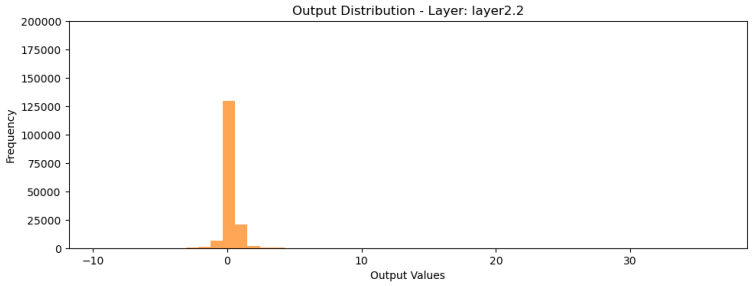


Figure 25: Input distribution for the 2nd non-linear layer after polynomial approximation

Input and output distribution of the third non-linear layer:

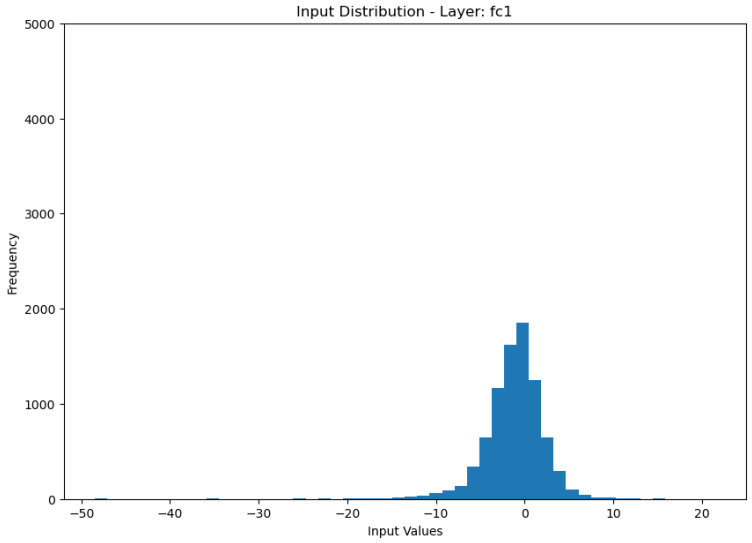


Figure 26: Input distribution for the 3rd non-linear layer after polynomial approximation

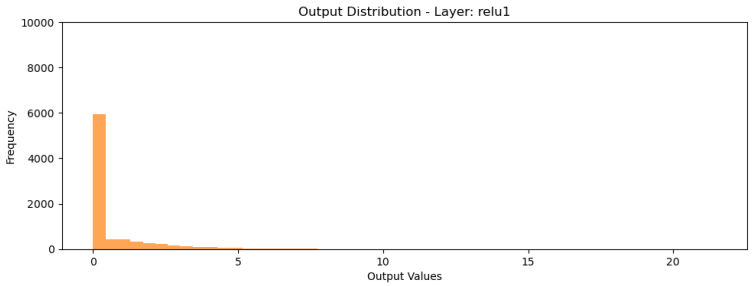


Figure 27: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and output distribution of the fourth non-linear layer:

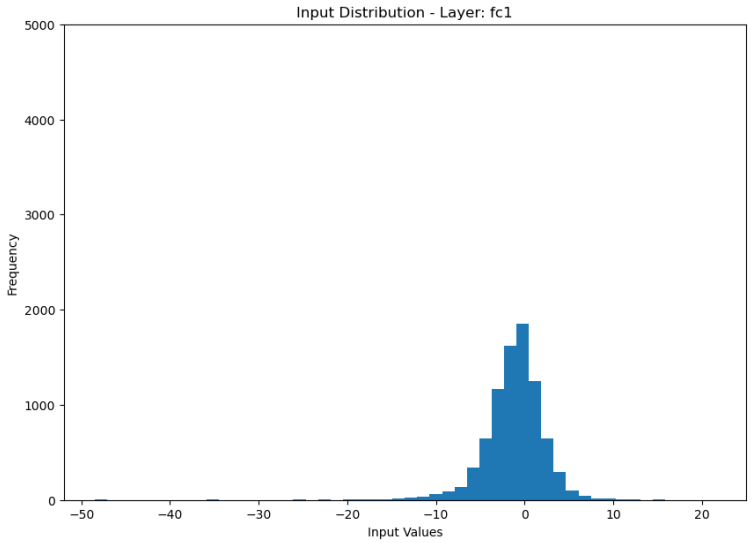


Figure 28: Input distribution for the 4th non-linear layer after polynomial approximation

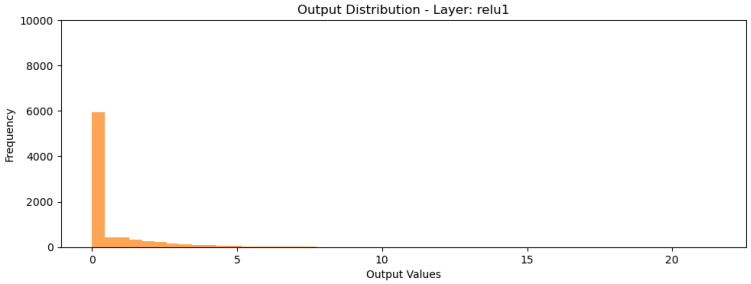


Figure 29: Output distribution for the 4th non-linear layer after polynomial approximation

Iteration 3: Input/output distributions after the third approximation (first, second, and third non-linear layers approximated):

Input distribution of the third non-linear layer stays the same.
Output distribution of the third non-linear layer:

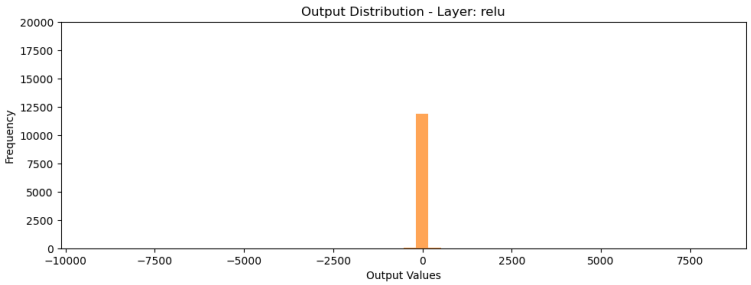


Figure 30: Ouput distribution for the 3rd non-linear layer after polynomial approximation

Input and output distribution of the fourth non-linear layer:

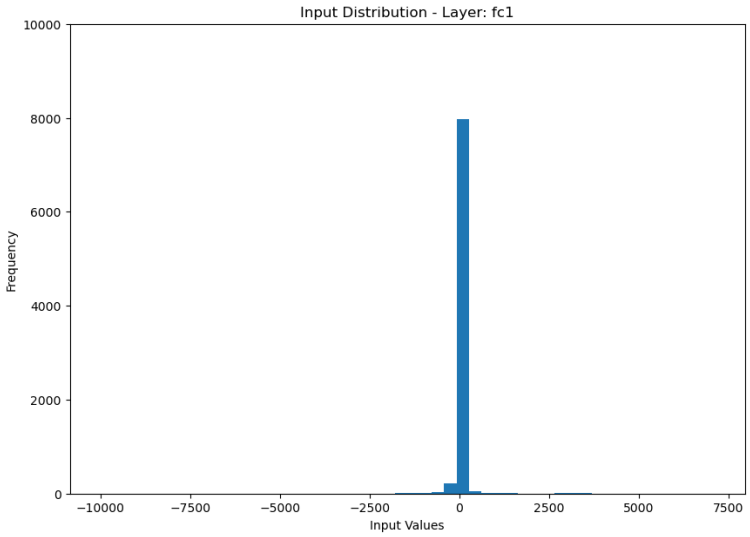


Figure 31: Input distribution for the 4th non-linear layer after polynomial approximation

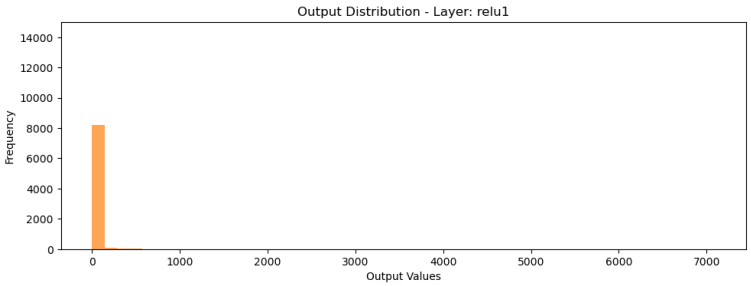


Figure 32: Output distribution for the 4th non-linear layer after polynomial approximation

Iteration 4: Input/output distributions after the fourth approximation (all non-linear layers approximated):

Input distribution of the fourth non-linear layer stays the same.
Output distribution of the fourth non-linear layer:

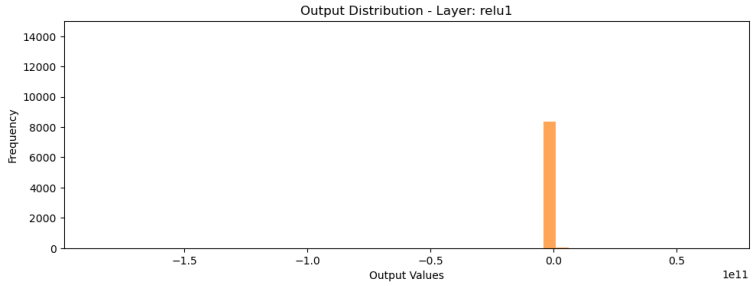


Figure 33: Ouput distribution for the 4th non-linear layer after polynomial approximation

3.4.3 Results

I compared the accuracy values for different polynomial orders in the range of 2-8 and different input ranges. The test accuracy after all layers have been approximated with the polynomials listed above is **28.430**.

3.5 Step-wise polynomial approximation with retraining

1. Approximate the first non-linear function based on the input/output distributions.
2. Re-train the remaining model.
3. Re-generate the input/output distributions.
4. Approximate the following non-linear function based on the newly generated input/output distributions.
5. Return to step 2 and repeat these steps until all non-linear functions have been approximated.

How to re-train the model:
Consider that your model is $y = f_n(...f_2(f_1(x))...)$, where x is the input, y is the output, and f_i is a non-linear function. For simplicity, I omitted the linear parts.

First, approximate $f_1(\cdot)$ with a polynomial function $p_1(\cdot)$ based on the input/ouput distribution of the $f_1(\cdot)$. Then, create a model starting from the following layer, i.e., $y = f_n(...f_2(z)...)$, where $z = p_1(x)$, load the weights from the previous iteration (for the first iteration, load the weights of the original model), and re-train the model. Repeat the process until all layers have been approximated.

There is no retraining after the last layer has been approximated.

3.5.1 Approximation functions

For the first non-linear layer, I used the polynomial approximations from the Section 3.3.1 because the input distribution for this layer didn't change.

Then, I created a new model and trained it using 250 epochs(best validation accuracy was 69.03) and based on the new input distribution obtained, I approximated the second non-linear function (layer2) channel-by-channel with the following polynomials:

Channel 0:
- Polynomial coefficients:[0.13407191250223544, 0.5177796335971374, 0.33082760941332967]
- Polynomial order: 2
- Range: [-1.5306311,1.24098322]
- Percentile: 85th

Channel 1:
- Polynomial coefficients:[0.1405591177685458, 0.5182650765333154, 0.3157275372819566]
- Polynomial order: 2
- Range: [-1.60663454,1.29334248]

- Percentile: 85th

Channel 2:
- Polynomial coefficients:[0.04616773561592291, 0.6287017472165743, 0.5427141164627037]
- Polynomial order: 2
- Range: [-0.17860017,0.63456402]
- Percentile: 65th

Channel 3:
- Polynomial coefficients:[0.0921718902945663, 0.5086055728448243, 0.4426362776015054]
- Polynomial order: 2
- Range: [-0.48759673,1.07472318]
- Percentile: 75th

Channel 4:
- Polynomial coefficients:[0.10423367646792704, 0.517678165826148, 0.42548474950442305]
- Polynomial order: 2
- Range: [-1.18956938,0.96586241]
- Percentile: 80th

Channel 5:
- Polynomial coefficients:[0.08684927536173426, 0.5069355663782869, 0.49246694440308025]
- Polynomial order:2
- Range: [-1.00420123,0.50507669]
- Percentile: 75th

Channel 6:
- Polynomial coefficients:[0.026923092501780876, 0.7039565664350007, 0.6097958714728304]
- Polynomial order: 2
- Range: [-0.09688054,0.44134008]
- Percentile: 60th

Channel 7:
- Polynomial coefficients:[0.18866041095392888, 0.5083827278887468, 0.21640759481066257]
- Polynomial order: 2
- Range: [-0.99968102,2.19996519]
- Percentile: 90th

Channel 8:
- Polynomial coefficients:[0.0938016879730664, 0.49828685070130363, 0.4444048069267591]
- Polynomial order: 2
- Range: [-1.0888302,0.51533505]
- Percentile: 75th

Channel 9:
- Polynomial coefficients:[0.021448738477327367, 0.9007087476306077, 0.09161259682462797]

- Polynomial order: 2
- Range: [-0.08896549,0.94084605]
- Percentile: 65th

Channel 10:
- Polynomial coefficients:[0.082204892753504, 0.5246301097930822, 0.5446835462187555]
- Polynomial order: 2
- Range: [-0.95077041,0.65290903]
- Percentile: 75th

Channel 11:
- Polynomial coefficients:[0.10562292870764717, 0.4987736382490293, 0.4160763160594291]
- Polynomial order: 2
- Range: [-1.11862929,1.13361974]
- Percentile: 80th

Channel 12:
- Polynomial coefficients:[0.14208266807930872, 0.40567660419751717, 0.20706575633200294]
- Polynomial order: 2
- Range: [-1.8363704,0.58228831]
- Percentile: 80th

Channel 13:
- Polynomial coefficients:[0.23217062431030397, 0.5019762732648405, 0.1794041373414507]
- Polynomial order: 2
- Range: [-1.27388889,2.69624479]
- Percentile: 95th

Channel 14:
- Polynomial coefficients:[0.07806391599288287, 0.5234139060383206, 0.5720664717779396]
- Polynomial order: 2
- Range: [-0.90219945,0.55541822]
- Percentile: 70th

Channel 15:
- Polynomial coefficients:[0.07439517310819502, 0.5472341758380768, 0.47937110387547177]
- Polynomial order: 2
- Range: [-0.33965491,0.89939006]
- Percentile: 65th

Then, I created a new model without the first and the second non-linear layers and trained it using 250 epochs (best validation accuracy was 67.52). Based on the new input distribution obtained, I approximated the third non-linear function (relu) with the polynomial $0.1502814176061165x^2+0.4752259774968486x+0.2980240676298627$ which was obtained by running the Remez algorithm on the range [-1.14617291,1.73951588] corresponding to 85th percentile of the in-

put data distribution.

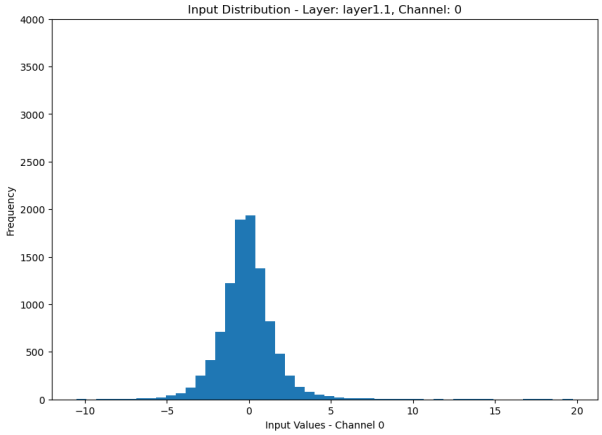
After that, I created another new model with only the third non-linear layer but there was no need to retrain this model because it now consisted of only ReLU and linear layers. Based on the new input distribution obtained, I approximated the fourth non-linear function (relu1) with the polynomial $0.027566475932797263x^2 + 0.9196264988494071x + 0.04697932280703512$ which was obtained by running the Remez algorithm on the range [-0.12216805,1.47718391] corresponding to 75th percentile of the input data distribution.

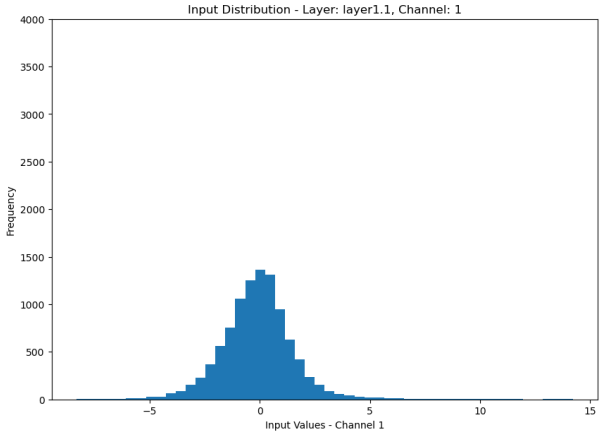
3.5.2 Input and output distribution of non-linear layers after approximation

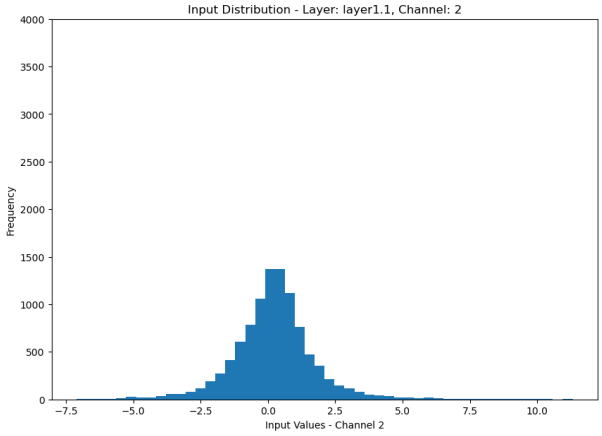
Iteration 0: Input/output distributions before any approximation were provided in the Section 3.2.1.

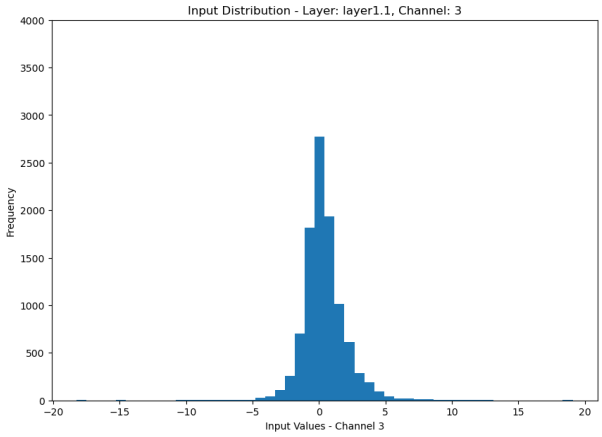
Iteration 1: Input/output distributions after the first approximation and after re-training.

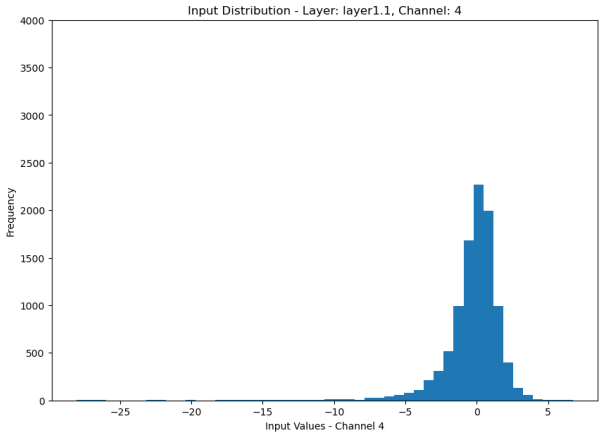
Input distributions of the second non-linear layer:

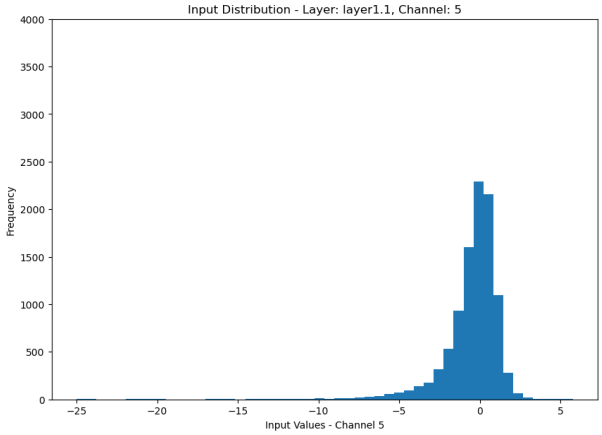


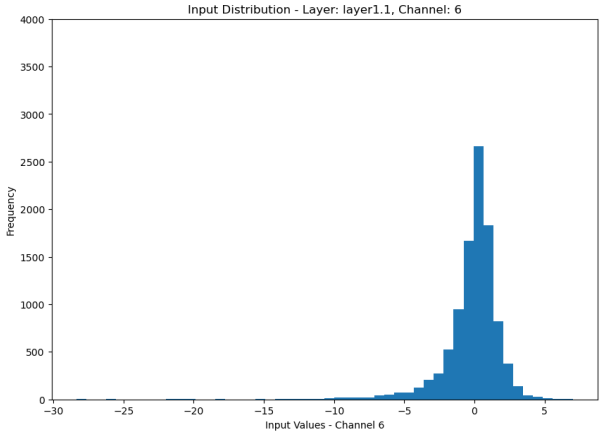

Input Distribution - Layer: layer1.1, Channel: 1
Frequency
Input Values - Channel 1

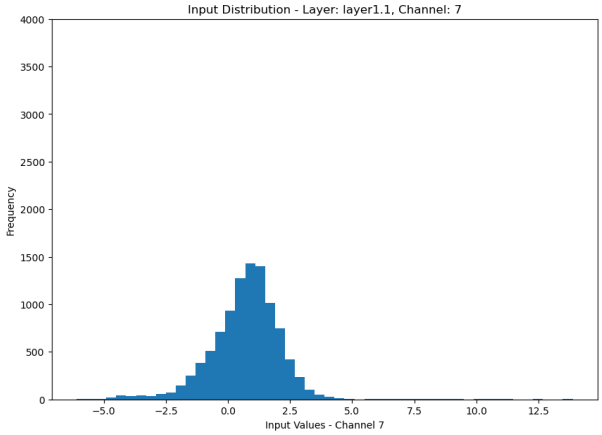

Input Distribution - Layer: layer1.1, Channel: 2
Frequency
Input Values - Channel 2

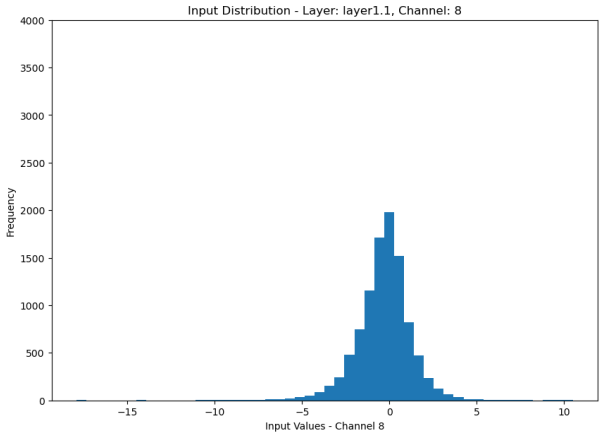

Input Distribution - Layer: layer1.1, Channel: 3
Frequency
Input Values - Channel 3

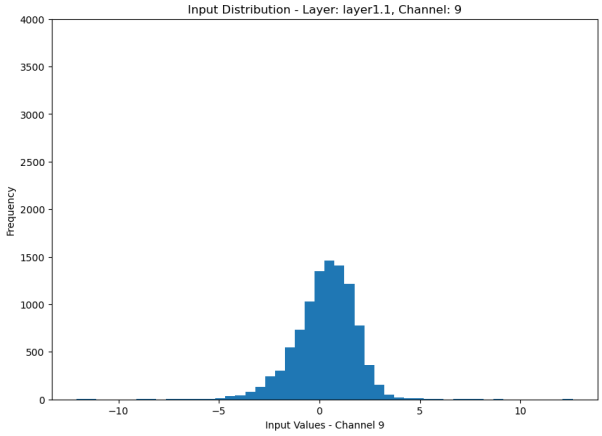

Input Distribution - Layer: layer1.1, Channel: 4
Frequency
Input Values - Channel 4
4000
3500
3000
2500
2000
1500
1000
500
0
−25
−20
−15
−10
−5
0
5

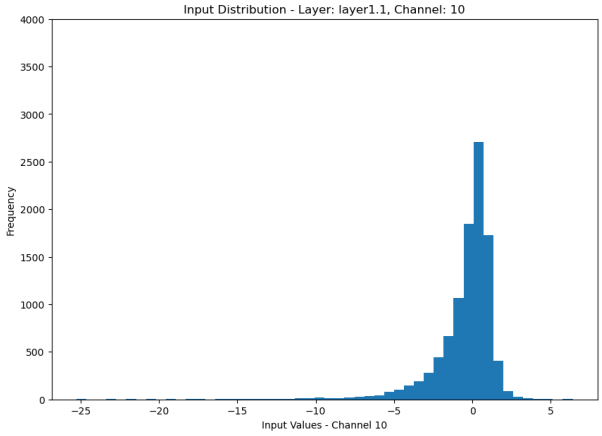

Input Distribution - Layer: layer1.1, Channel: 5
Frequency
Input Values - Channel 5
4000
3500
3000
2500
2000
1500
1000
500
0
−25
−20
−15
−10
−5
0
5

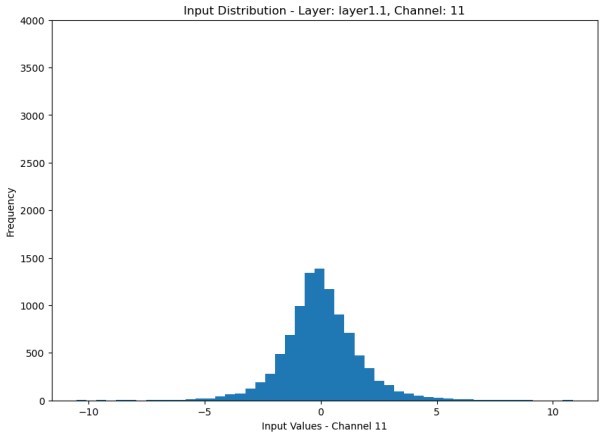

Input Distribution - Layer: layer1.1, Channel: 6
Frequency
Input Values - Channel 6

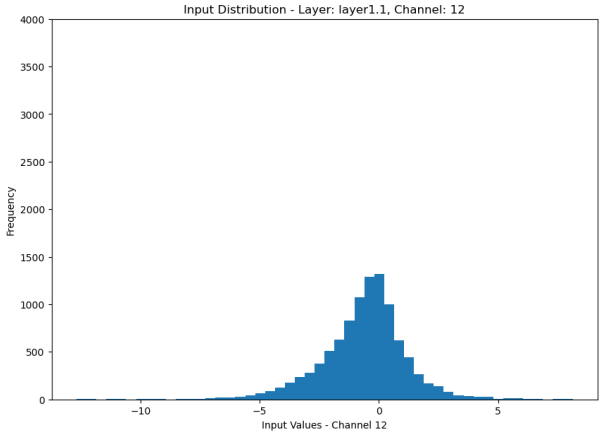

Input Distribution - Layer: layer1.1, Channel: 7
Frequency
Input Values - Channel 7

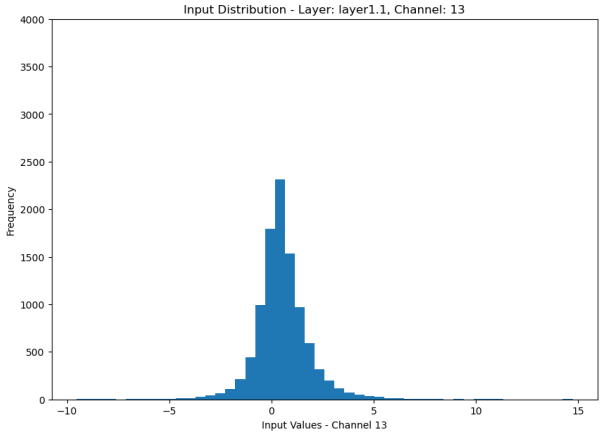

Input Distribution - Layer: layer1.1, Channel: 8
Frequency
Input Values - Channel 8

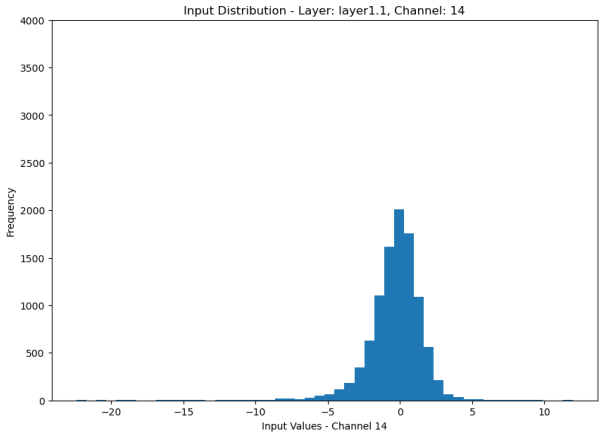

Input Distribution - Layer: layer1.1, Channel: 9
Frequency
Input Values - Channel 9

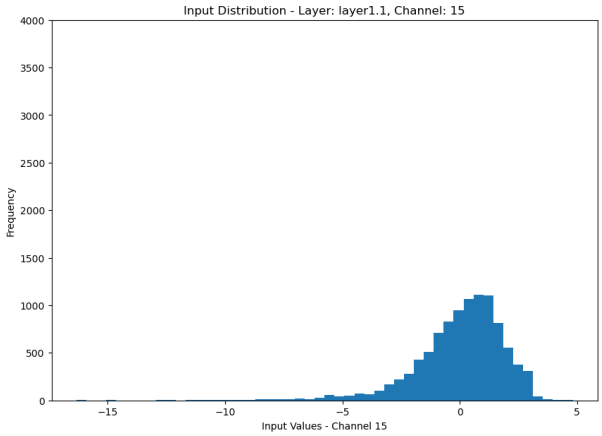

Input Distribution - Layer: layer1.1, Channel: 10
Frequency
Input Values - Channel 10


Input Distribution - Layer: layer1.1, Channel: 11
Frequency
Input Values - Channel 11


Input Distribution - Layer: layer1.1, Channel: 12
Frequency
Input Values - Channel 12


Input Distribution - Layer: layer1.1, Channel: 13
Frequency
Input Values - Channel 13


Input Distribution - Layer: layer1.1, Channel: 14
Frequency
Input Values - Channel 14



Input and output distribution of the third non-linear layer:

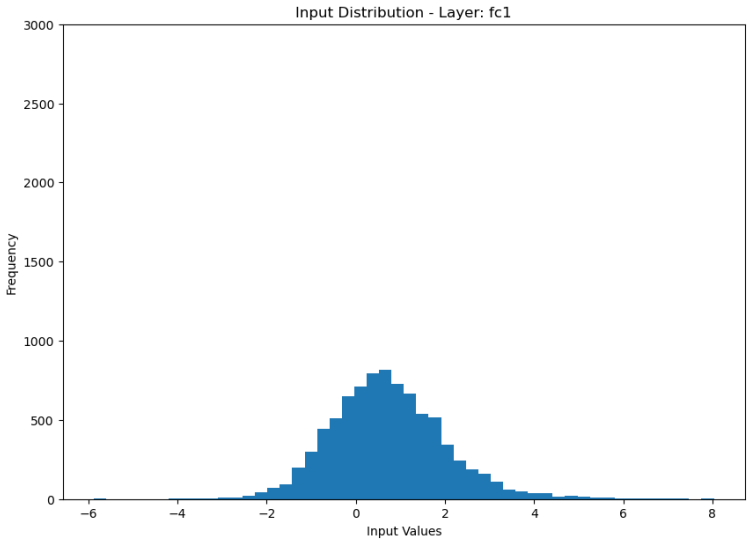


Figure 34: Input distribution for the 3rd non-linear layer after polynomial approximation

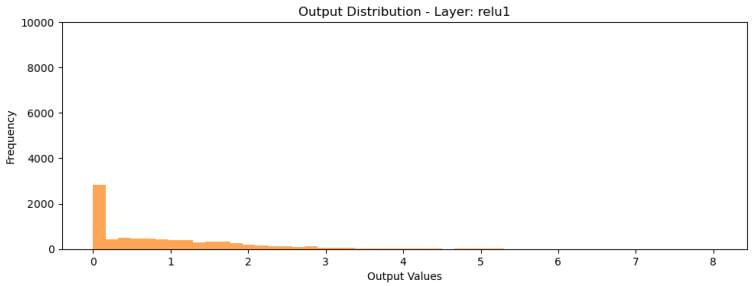


Figure 35: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and output distribution of the fourth non-linear layer:

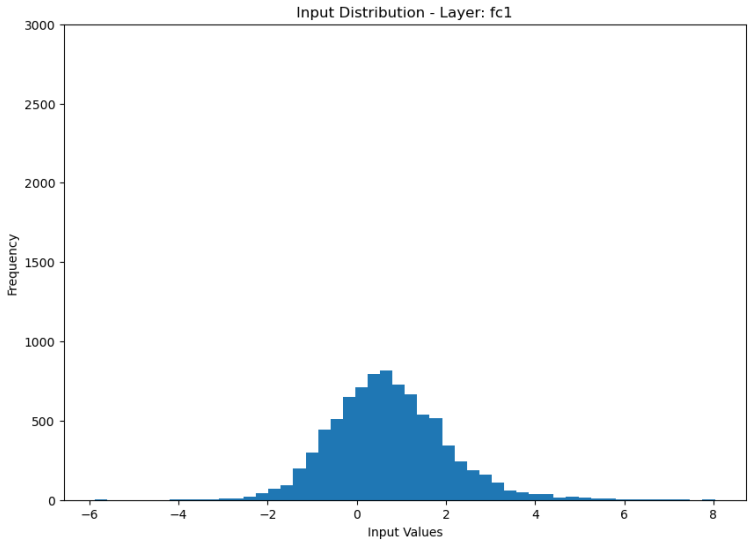


Figure 36: Input distribution for the 4th non-linear layer after polynomial approximation

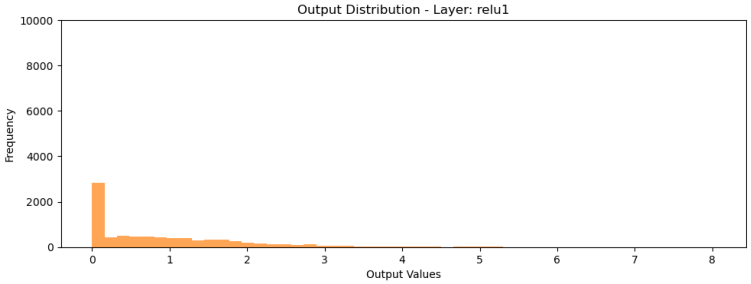


Figure 37: Output distribution for the 4th non-linear layer after polynomial approximation

Iteration 2: Input/output distributions after the second approximation and after re-training.

Input and distribution of the third non-linear layer:

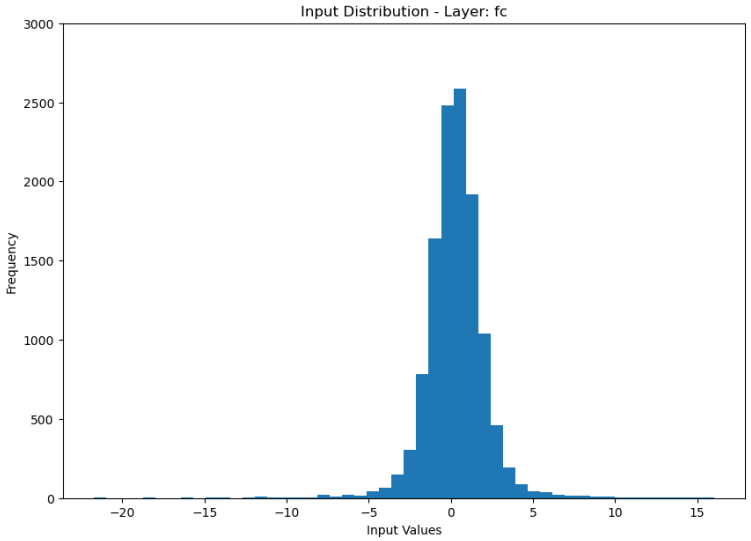


Figure 38: Input distribution for the 3rd non-linear layer after polynomial approximation

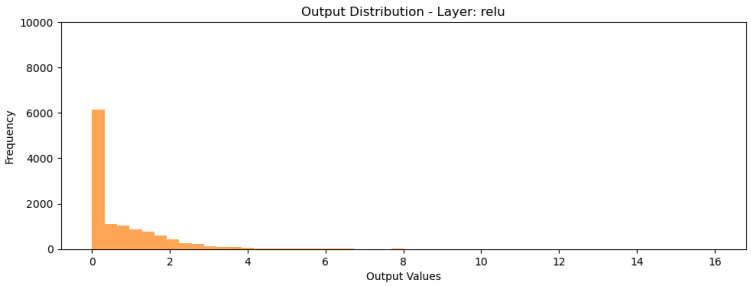


Figure 39: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and distribution of the fourth non-linear layer:

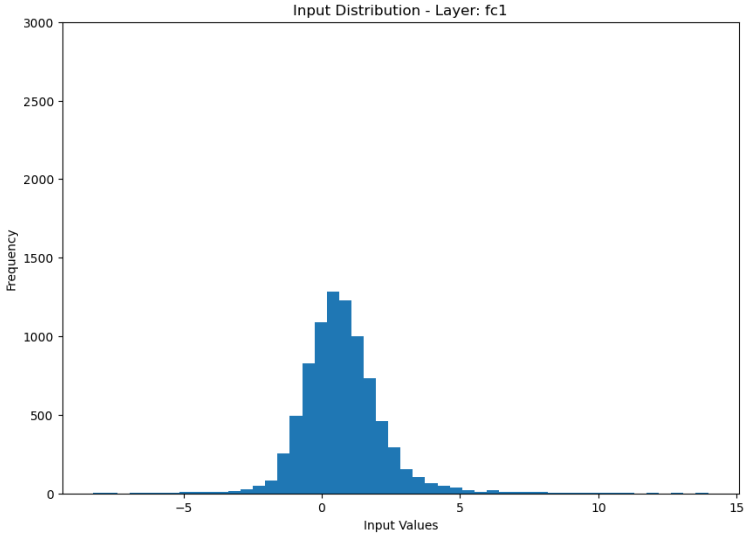


Figure 40: Input distribution for the 4th non-linear layer after polynomial approximation

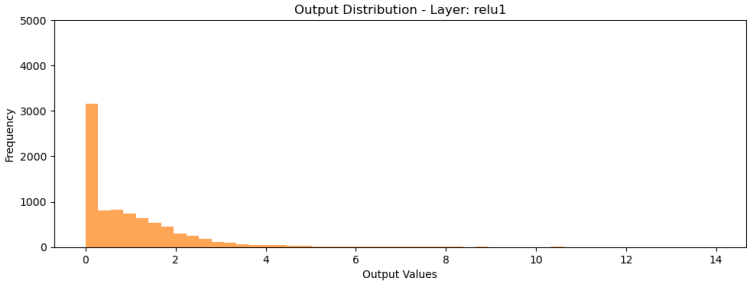


Figure 41: Output distribution for the 4th non-linear layer after polynomial approximation

Iteration 3: Input/output distributions after the third and fourth approximations (no need to re-train because there are only ReLU and linear layers in the new model):

Input and distribution of the third non-linear layer:

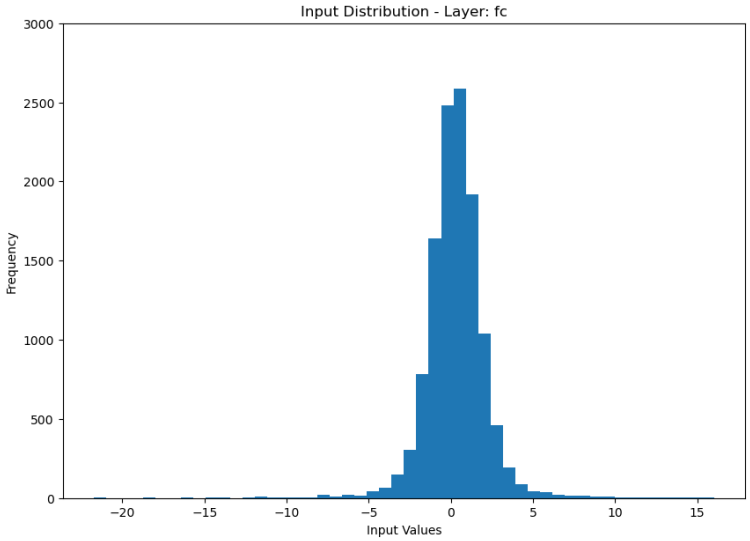


Figure 42: Input distribution for the 3rd non-linear layer after polynomial approximation

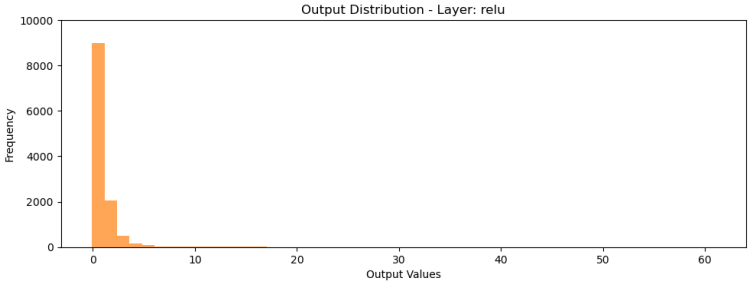


Figure 43: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and distribution of the fourth non-linear layer:

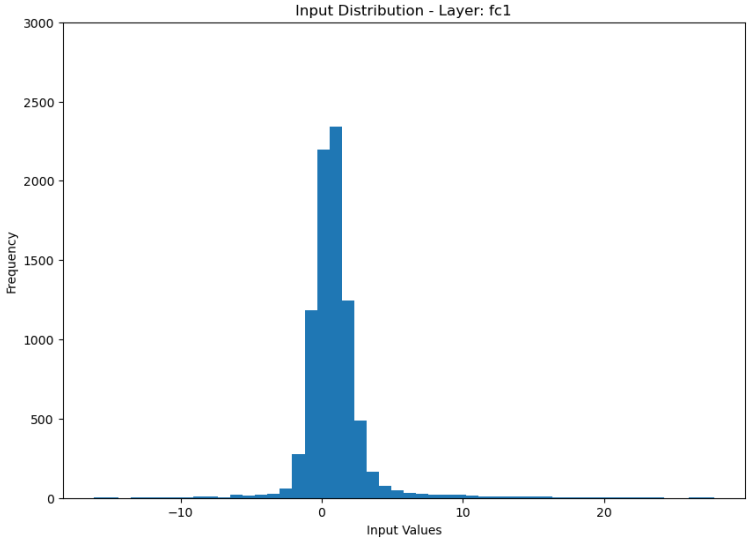


Figure 44: Input distribution for the 4th non-linear layer after polynomial approximation

3.5.3 Results

The test accuracy after all layers have been approximated is **59.150**.

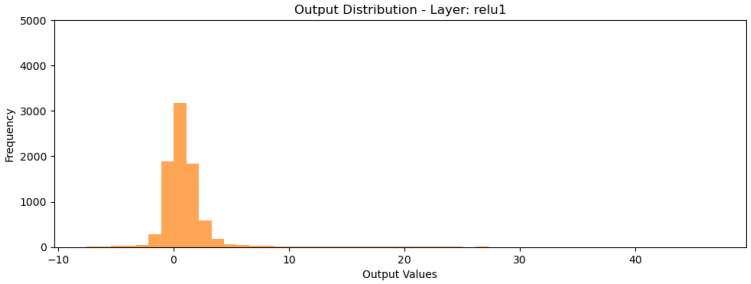


Figure 45: Output distribution for the 4th non-linear layer after polynomial approximation

4 LeNet-5 with Caltech-101

4.1 Model description

This LeNet-5 model was created following the original LeNet-5 model's architecture. It consists of a two consecutive layers (layer 1 and layer 2) each comprising of a convolutional, Batchnorm, ReLU, and MaxPool, followed by alternating convolutional and dense layers. Input and output channels for the first convolutional layer were taken as 3 and 6, while for the second convolutional layer in the layer 2 they were computed as 6 and 16, respectively. The kernel size was 5, input and output dimensions for the first dense layer after layer 2 were computed as 400 and 120, for the second dense layer - 120 and 84, and for the last dense layer - 84 and 10.

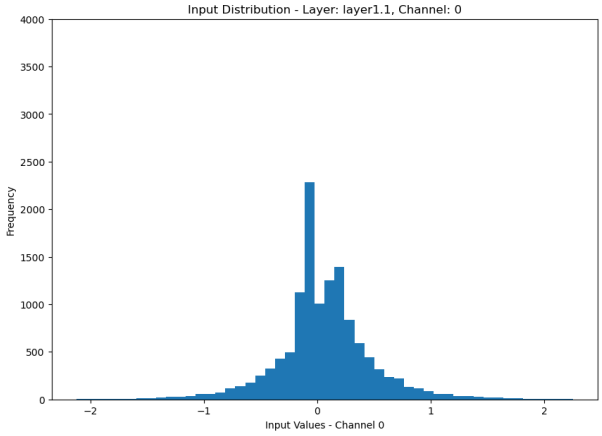
I trained this model using 300 epochs on the Caltech101 dataset. The hyperparameters used when training were: batch size was 16, learning rate was 0.001, optimizer was SDG.

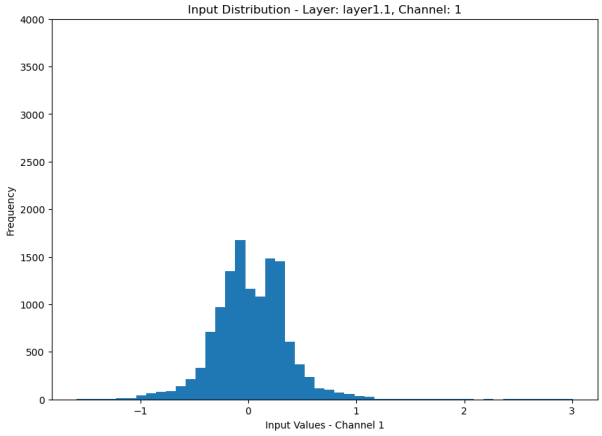
4.2 Original

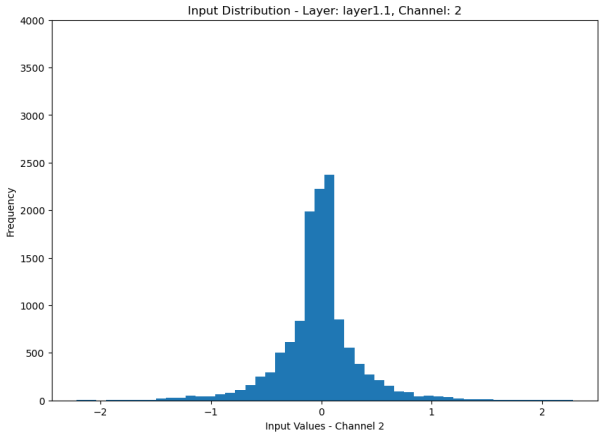
4.2.1 Input and output distribution of non-linear functions

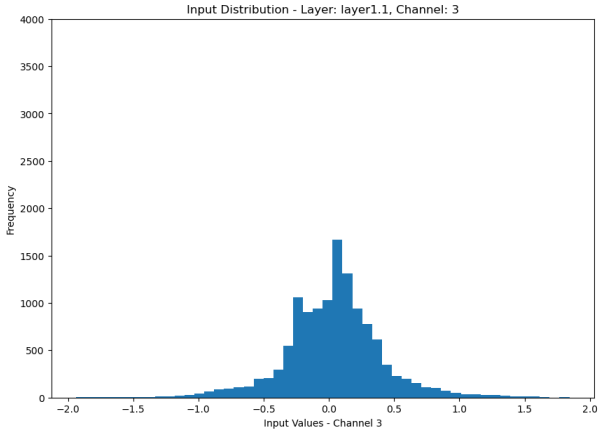
Output distribution for the BatchNorm layer that precedes the ReLU1 layer corresponds to the input distribution for the ReLU1 layer. Since I was looking at the distribution of the ReLU layers channel-by-channel, it was easier for me to plot the output distribution of the layer that precedes the ReLU layer as this output should correspond to the input of the ReLU layer. Based on the model architecture, the number of input channels for the first non-linear function should be 6, while for the second - 16.

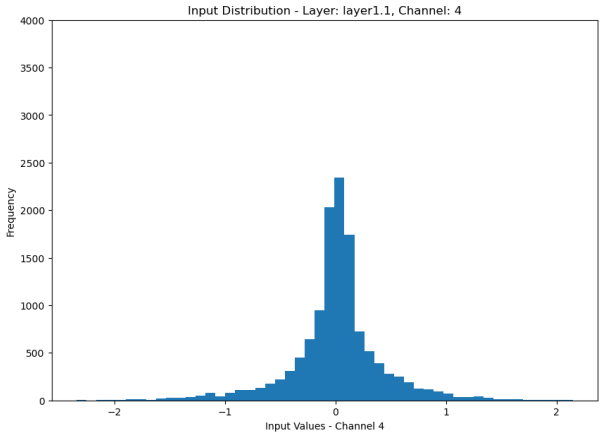
Input distributions of the first non-linear function layer channel-by-channel:

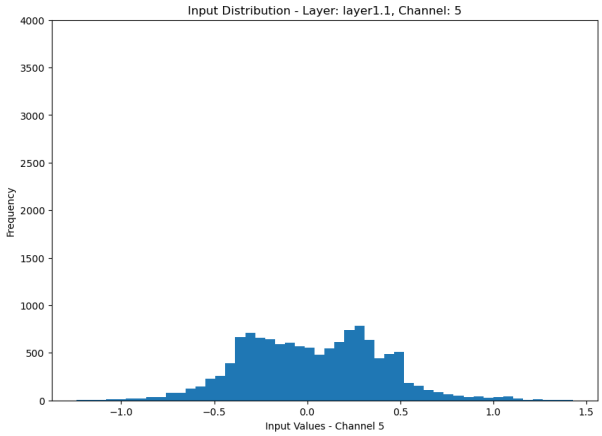

Input Distribution - Layer: layer1.1, Channel: 0
Frequency
Input Values - Channel 0


Input Distribution - Layer: layer1.1, Channel: 1
Frequency
Input Values - Channel 1

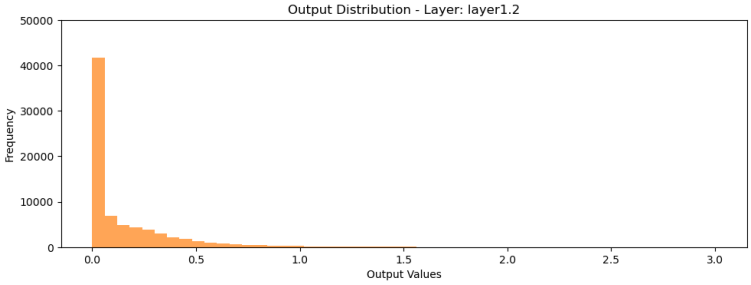

Input Distribution - Layer: layer1.1, Channel: 2
Frequency
Input Values - Channel 2


Input Distribution - Layer: layer1.1, Channel: 3
Frequency
Input Values - Channel 3

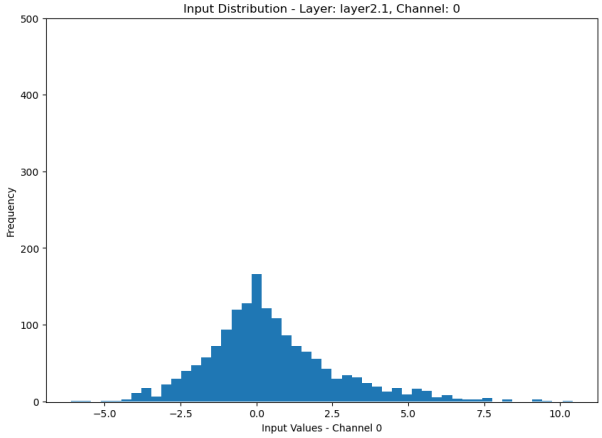

Input Distribution - Layer: layer1.1, Channel: 4
Frequency
Input Values - Channel 4

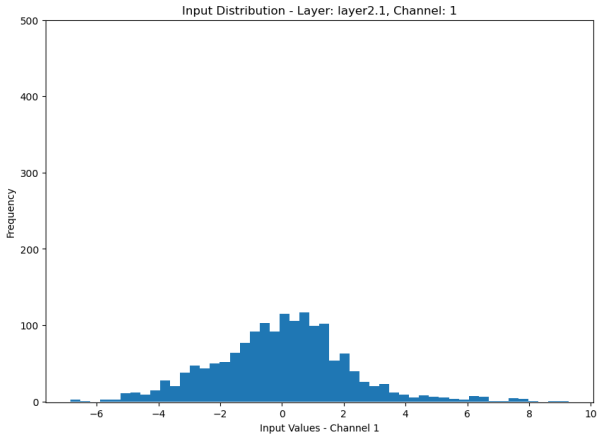

Input Distribution - Layer: layer1.1, Channel: 5
Frequency
Input Values - Channel 5

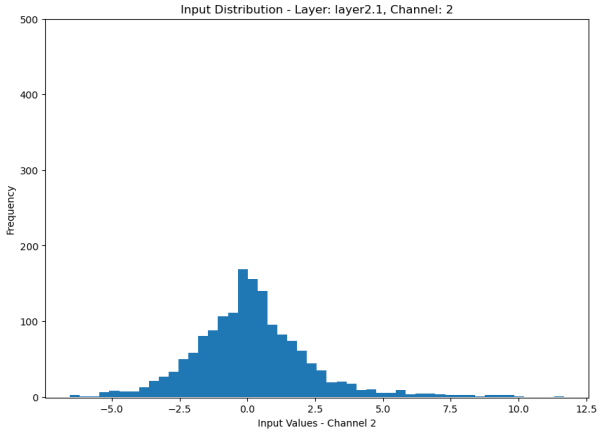
Output distribution for the first non-linear layer:

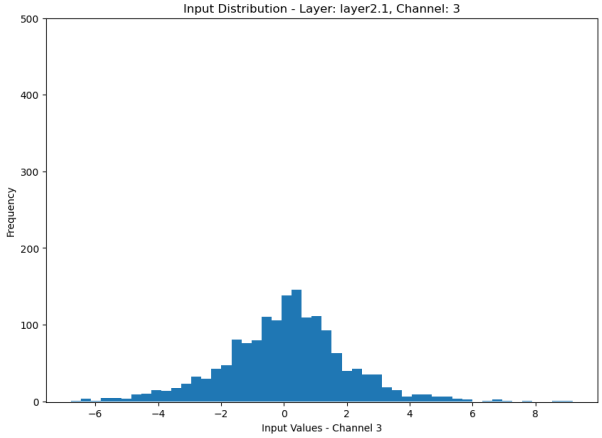


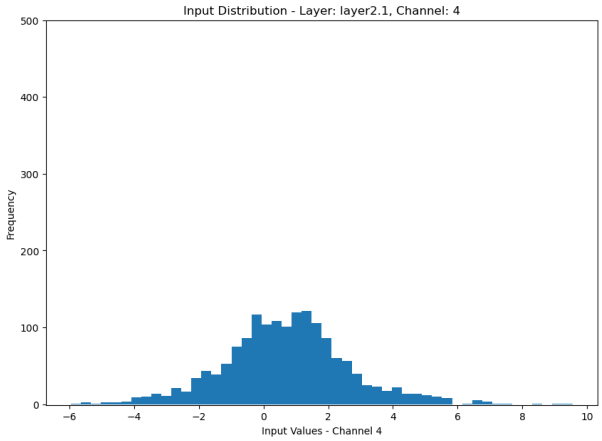
Input distributions of the second non-linear function layer channel-by-channel:

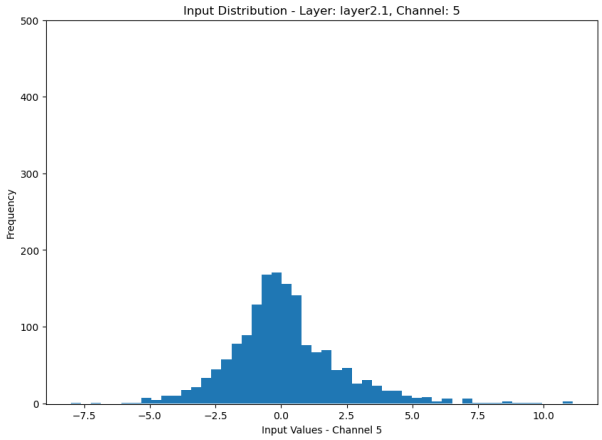


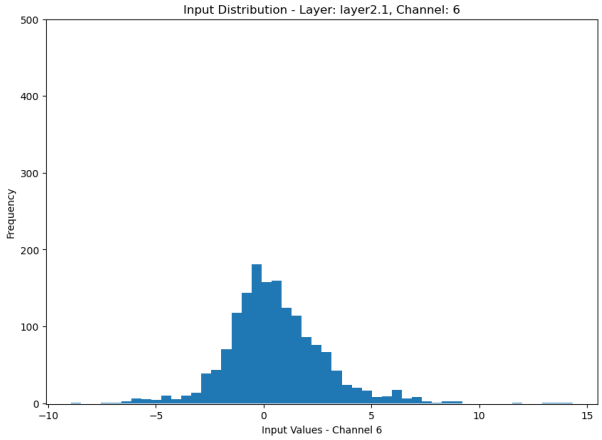


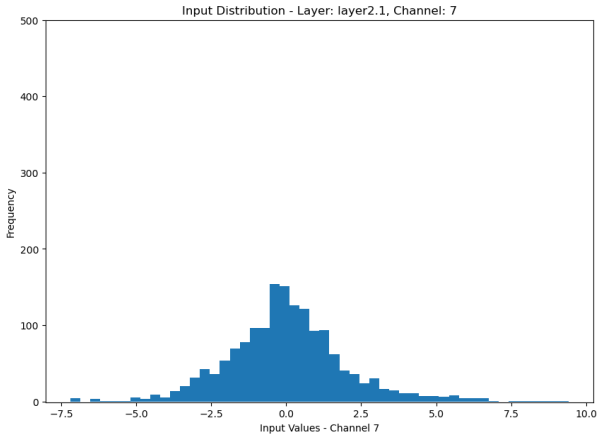

Input Distribution - Layer: layer2.1, Channel: 2
Frequency
Input Values - Channel 2
0
100
200
300
400
500
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5

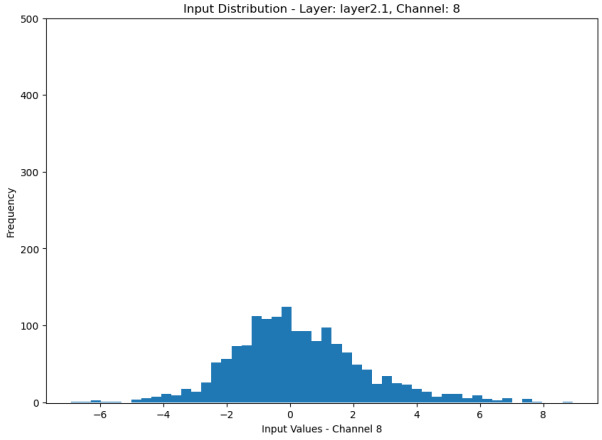

Input Distribution - Layer: layer2.1, Channel: 3
Frequency
Input Values - Channel 3
0
100
200
300
400
500
−6
−4
−2
0
2
4
6
8

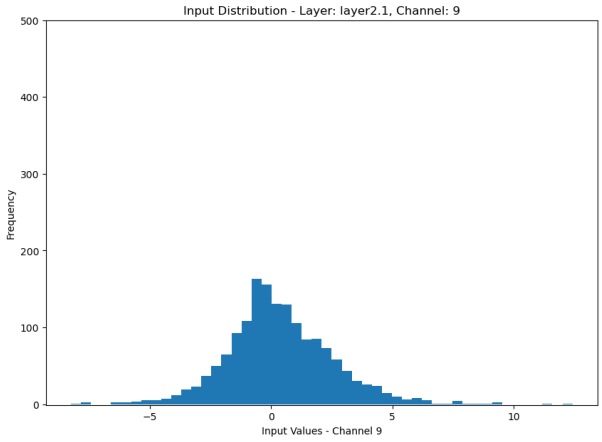

Input Distribution - Layer: layer2.1, Channel: 4
Frequency
Input Values - Channel 4
0
100
200
300
400
500
−6
−4
−2
0
2
4
6
8
10

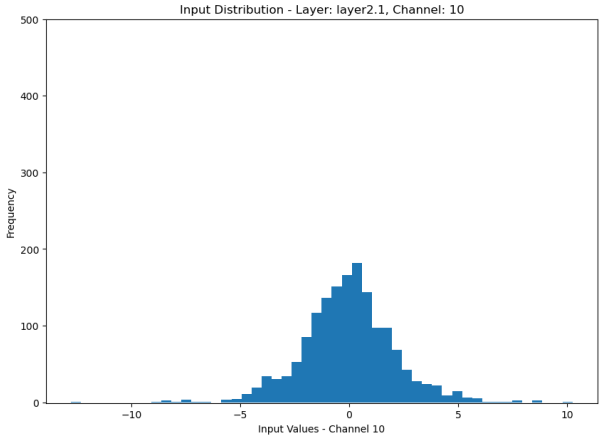

Input Distribution - Layer: layer2.1, Channel: 5
Frequency
Input Values - Channel 5
0
100
200
300
400
500
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

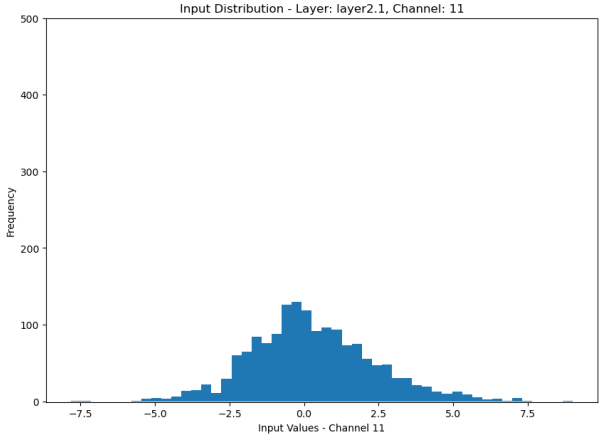

Input Distribution - Layer: layer2.1, Channel: 6
Frequency
Input Values - Channel 6

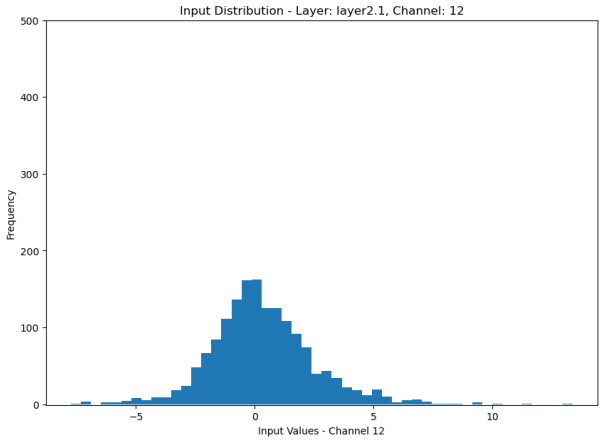

Input Distribution - Layer: layer2.1, Channel: 7
Frequency
Input Values - Channel 7

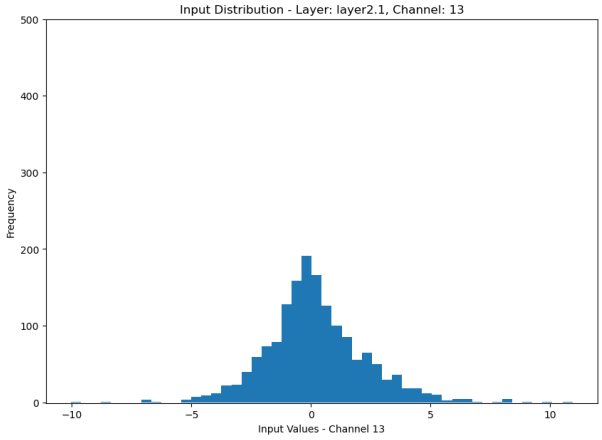

Input Distribution - Layer: layer2.1, Channel: 8
Frequency
Input Values - Channel 8

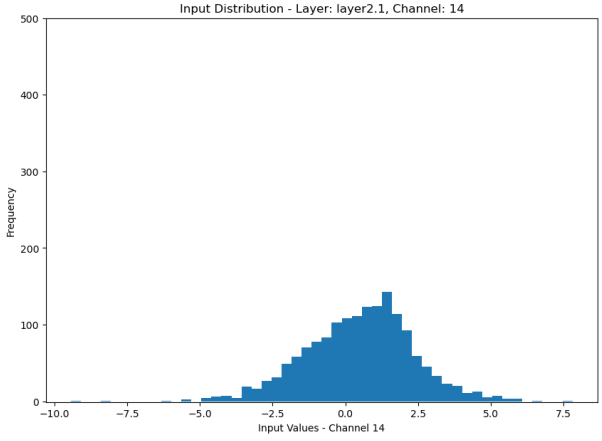

Input Distribution - Layer: layer2.1, Channel: 9
Frequency
Input Values - Channel 9

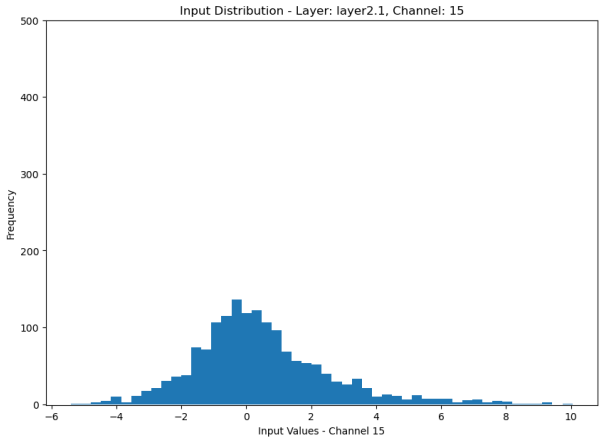

Input Distribution - Layer: layer2.1, Channel: 10
Frequency
Input Values - Channel 10


Input Distribution - Layer: layer2.1, Channel: 11
Frequency
Input Values - Channel 11

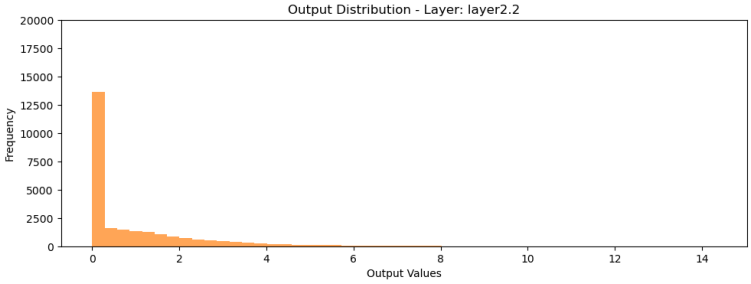

Input Distribution - Layer: layer2.1, Channel: 12
Frequency
Input Values - Channel 12


Input Distribution - Layer: layer2.1, Channel: 13
Frequency
Input Values - Channel 13


Input Distribution - Layer: layer2.1, Channel: 14
Frequency
Input Values - Channel 14



Output distribution of the second non-linear layer:



Input and output distribution of the third non-linear layer:

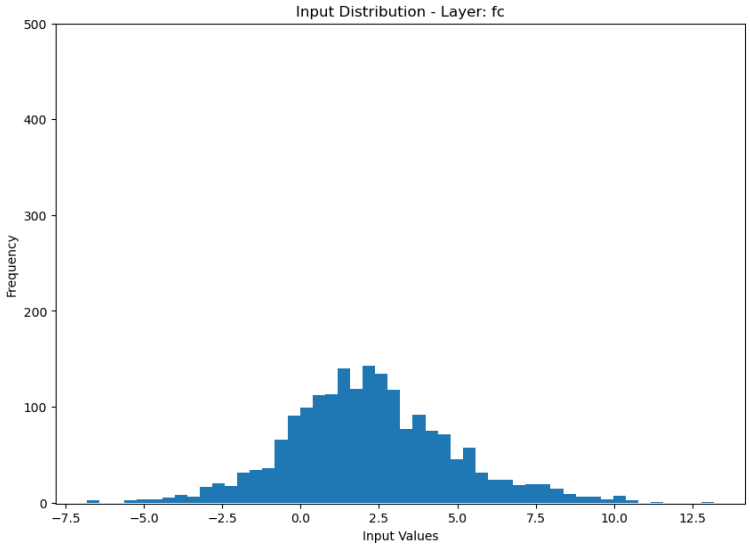


Figure 46: Input distribution for the 3rd non-linear layer

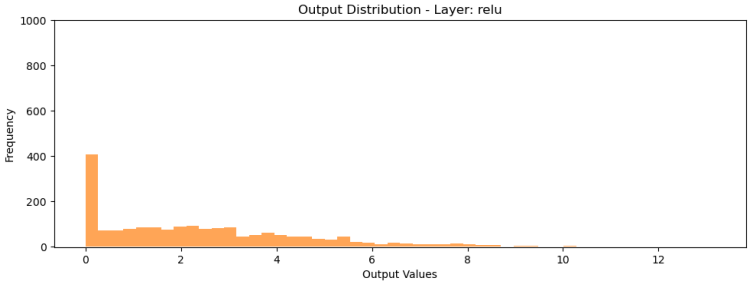


Figure 47: Output distribution for the 3rd non-linear layer

Input and output distribution of the fourth non-linear layer:

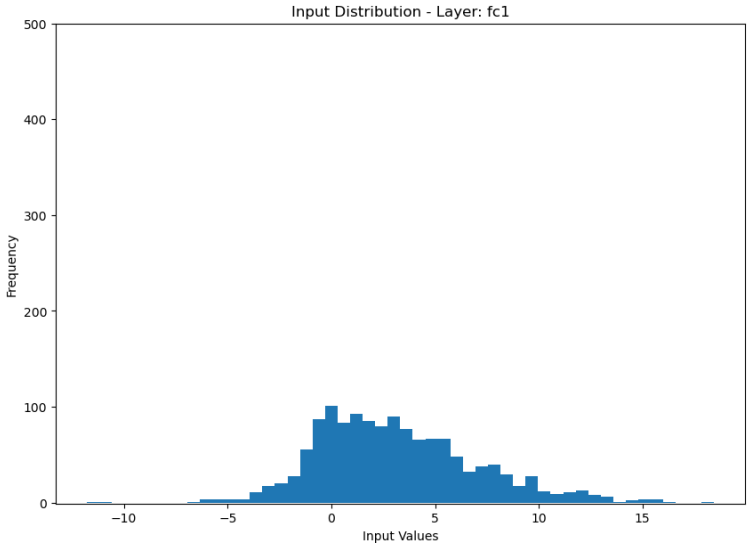


Figure 48: Input distribution for the 4th non-linear layer

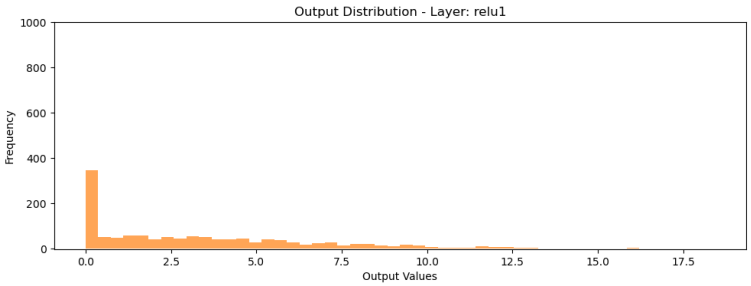


Figure 49: Output distribution for the 4th non-linear layer

4.2.2 Results

The best validation accuracy achieved when training the original model with 250 epochs was **59.447** (at 80 epochs).

4.3 All-at-once polynomial approximation without retraining

4.3.1 Approximation functions

Based on the output distribution of layer 1 channel 0, I approximated the ReLU function corresponding to channel 0 with a polynomial $0.11455314523094098x^2 + 0.4858826423375821x + 0.3858443466572174$ which was obtained by running Remez algorithm for the range [-1.10345144,1.29463036] which corresponds to the 99th percentile of the output data distribution.

Following the similar procedure, the approximations for the other channels of layer 1 were as follows:

Channel 1:
- Polynomial coefficients: [0.08666648777331669, 0.4921647998138579, 0.507952726334997]
- Polynomial order: 2
- Range: [-0.87996733,0.95852822]
- Percentile: 99th

Channel 2:
- Polynomial coefficients: [0.1051015511211718, 0.5075665405768744, 0.4188071600474574]
- Polynomial order: 2
- Range:[-1.16067903,1.07007534]
- Percentile: 99th

Channel 3:
- Polynomial coefficients: [0.1017769880439152, 0.4881737719212299, 0.4335226727902329]
- Polynomial order: 2
- Range: [-1.00128092,1.14214429]
- Percentile: 99th

Channel 4:
- Polynomial coefficients: - [0.11803114842861889, 0.5070950442319011, 0.37285521106461555]
- Polynomial order: 2
- Range:- [-1.30104382 1.20497499]
- Percentile: 99th

Channel 5:
- Polynomial coefficients: - [0.08598867258210668, 0.4812310532137813, 0.5163911501808258]
- Polynomial order:2
- Range: - [-0.78513326 0.98401485]
- Percentile: 99th

Based on the output distribution of layer 2 channel 0, I approximated the ReLU function corresponding to channel 0 with a polynomial $0.3945435891881062x^2 +$

$0.48015638025804536x + 0.11223959364353583$ which was obtained by running Remez algorithm for the range [-2.61338855,4.55753431] which corresponds to the 95th percentile of the output data distribution.

Following the similar procedure, the approximations for the other channels of layer 2 were as follows:

Channel 1:
- Polynomial coefficients:[0.33102236584874023, 0.5063734990866854, 0.1329103187891859]
- Polynomial order: 2
- Range: [-3.63883512,3.39719833]
- Percentile: 99th

Channel 2:
- Polynomial coefficients:[0.17991485183072076, 0.488895573224311, 0.24512356192370136]
- Polynomial order: 2
- Range: [-1.78133004,2.01377515]
- Percentile: 85th

Channel 3:
- Polynomial coefficients: [0.04290808143157443, 0.4840481572015568, 1.021872195390718]
- Polynomial order: 2
- Range: [-0.27001758,0.49590787]
- Percentile: 60th

Channel 4:
- Polynomial coefficients: [0.07742147954479897, 0.543529035219302, 0.5871287790061368, -0.2063671122650113]
- Polynomial order: 3
- Range:[-0.42123914,1.80587161]
- Percentile: 75th

Channel 5:
- Polynomial coefficients: [0.5567333870233839, 0.4755039138551996, 0.08040631776412928]
- Polynomial order: 2
- Range: [-4.46303606,6.4415098]
- Percentile: 99th

Channel 6:
- Polynomial coefficients: [0.08528745192894688, 0.5700284194879777, 0.3824215209446421]
- Polynomial order: 2
- Range: [-0.36697319,1.0624651]
- Percentile: 65th

Channel 7:
- Polynomial coefficients:[0.16650769774681792, 0.5041166244404814, 0.2640510038292501]

- Polynomial order: 2
- Range: [-1.81210245,1.73445142]
- Percentile: 85th

Channel 8:
- Polynomial coefficients: [0.34379070015154645, 0.47526526585674145, 0.13026605010934172]
- Polynomial order: 2
- Range: [-2.69971071,3.97858039]
- Percentile: 95th

Channel 9:
- Polynomial coefficients: [0.36024911212879956, 0.47556694808680394, 0.12424705987813629]
- Polynomial order: 2
- Range: [-2.89870236,4.16772151]
- Percentile: 95th

Channel 10:
- Polynomial coefficients: [0.1454246703156844, 0.5044986015372066, 0.30236039672393766]
- Polynomial order: 2
- Range: [-1.58534055,1.51085594]
- Percentile: 80th

Channel 11:
- Polynomial coefficients: [0.10266916210691876, 0.47555762355147, 0.4359363068143395]
- Polynomial order: 2
- Range:[-0.7595398,1.1884046]
- Percentile: 70th

Channel 12:
- Polynomial coefficients: [0.10232144818026598, 0.47787500489676593, 0.43516394595477476]
- Polynomial order: 2
- Range: [-0.70494965,1.18231945]
- Percentile: 70th

Channel 13:
- Polynomial coefficients:[0.24577625175783702, 0.4792451913717754, 0.18111080542034397]
- Polynomial order: 2
- Range: [-2.1786613,2.82592666]
- Percentile: 90th

Channel 14:
- Polynomial coefficients: [0.1920987436722642, 0.47536595801101855, 0.2330827063998248]
- Polynomial order: 2
- Range: [-1.44044909,2.2230213]
- Percentile: 985th

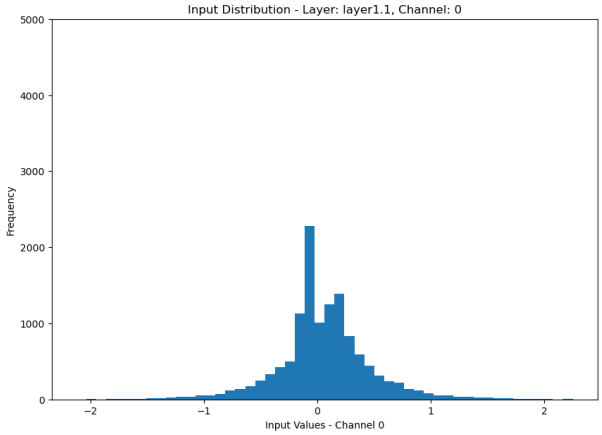
Channel 15:
- Polynomial coefficients: [0.2766797262327231, 0.4814106432242048, 0.1595535124071522]
- Polynomial order: 2
- Range: [-1.80030087,3.19562833]
- Percentile: 90th

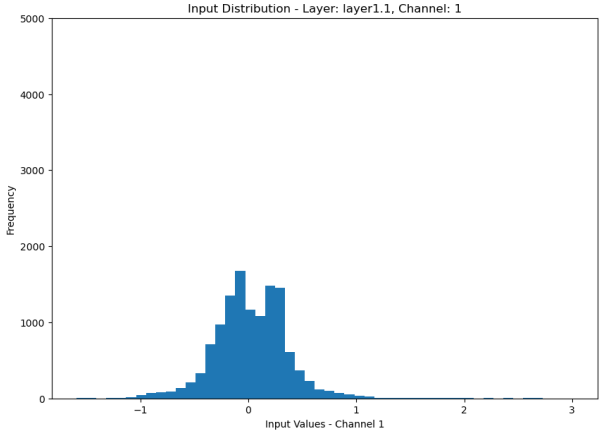
The polynomial approximation for the third non-linear function ReLU was $(-1.4038446135106956e-16)x^2 + 1.000000000000000x - 4.257818235399124e-17$ which was obtained by running Remez algorithm for on the range [0.16862178,4.23830595] corresponding to the 80th percentile of the input data distribution.

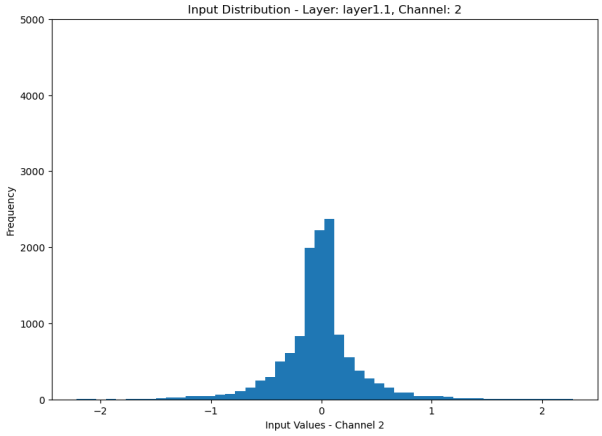
The final non-linear function was approximated with the polynomial $0.0025184466990086363x^2 + 0.9983479828047133x + 0.00022477380019400583$ which was obtained by running Remez algorithm for the range [-0.06002572,6.15109673] corresponding to the 80th percentile of the input data distribution.

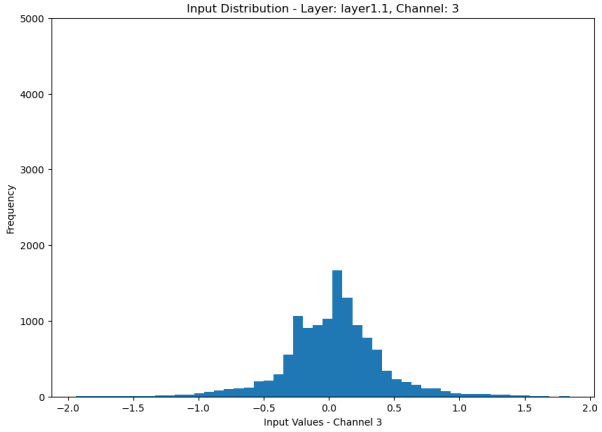
4.3.2 Input and output distribution of non-linear layers after approximation

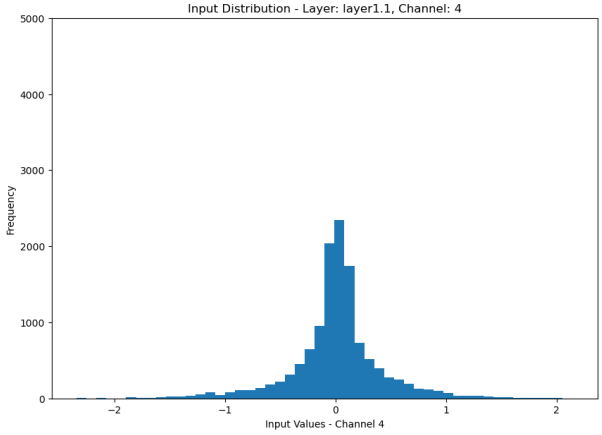
Input distributions of the first non-linear layer which is now approximated channel-by-channel with polynomial functions are as follows:

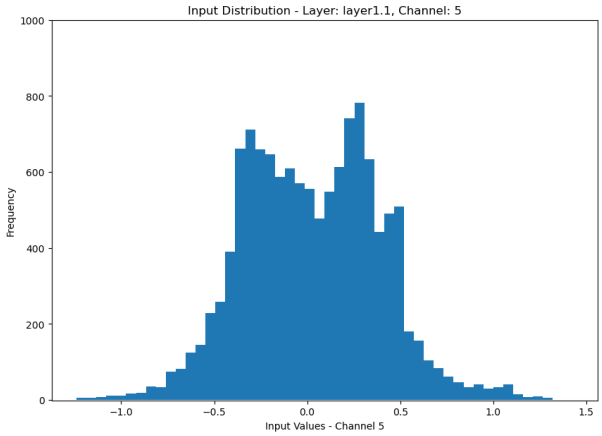




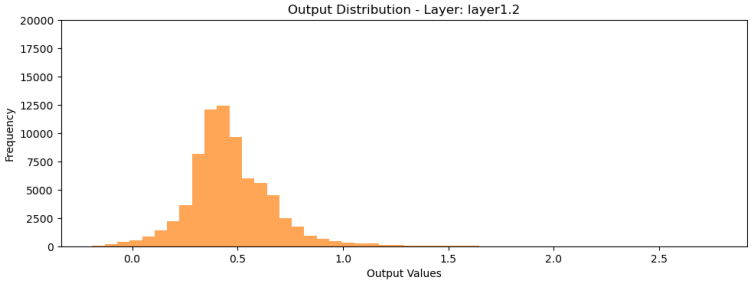

Input Distribution - Layer: layer1.1, Channel: 2
Frequency
Input Values - Channel 2


Input Distribution - Layer: layer1.1, Channel: 3
Frequency
Input Values - Channel 3

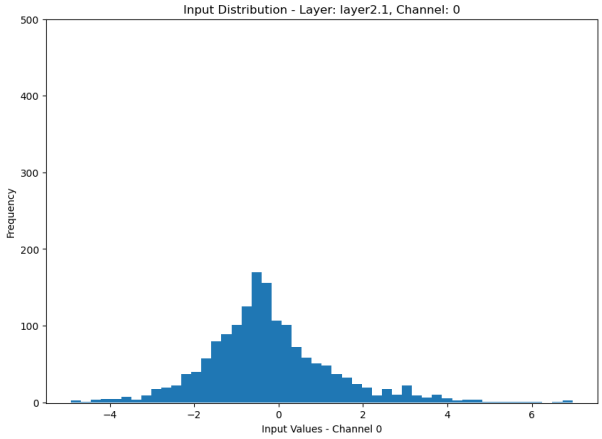

Input Distribution - Layer: layer1.1, Channel: 4
Frequency
Input Values - Channel 4

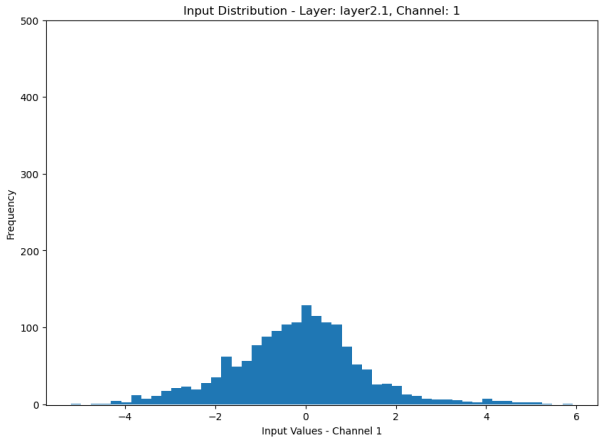


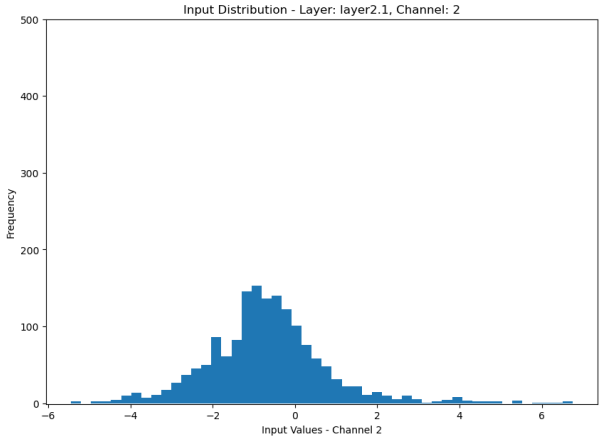
Output distribution of the first non-linear layer which is now approximated channel-by-channel with polynomial functions is as follows:

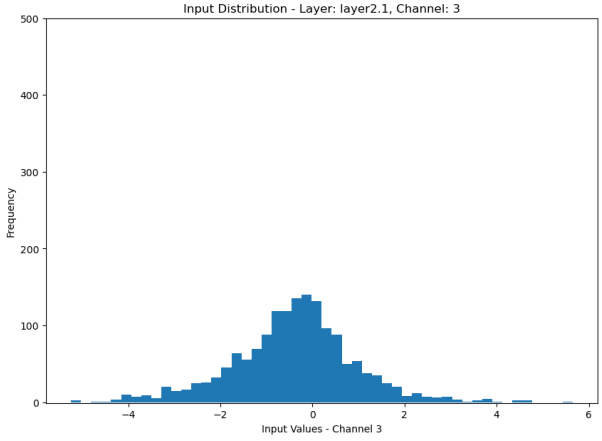


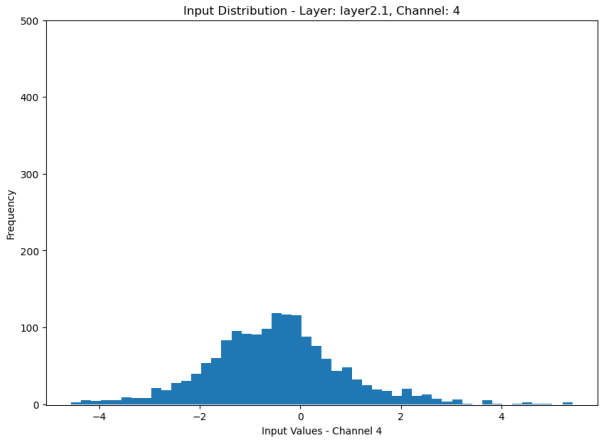
Input distributions of the second non-linear layer which is now approximated channel-by-channel with polynomial functions are as follows:

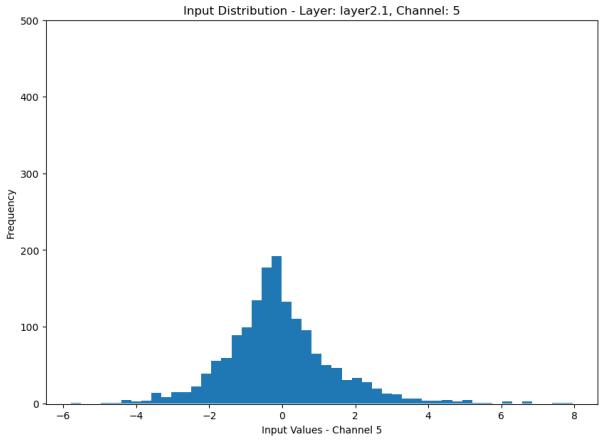


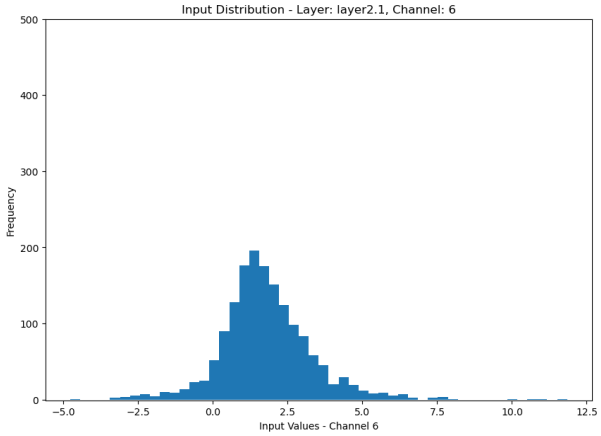


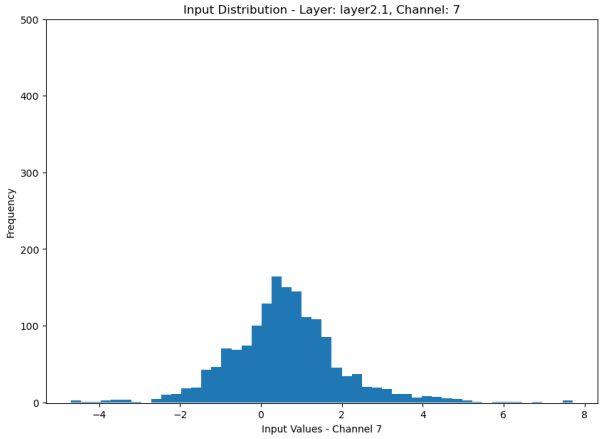

Input Distribution - Layer: layer2.1, Channel: 2
Frequency
Input Values - Channel 2
0
100
200
300
400
500
−6
−4
−2
0
2
4
6

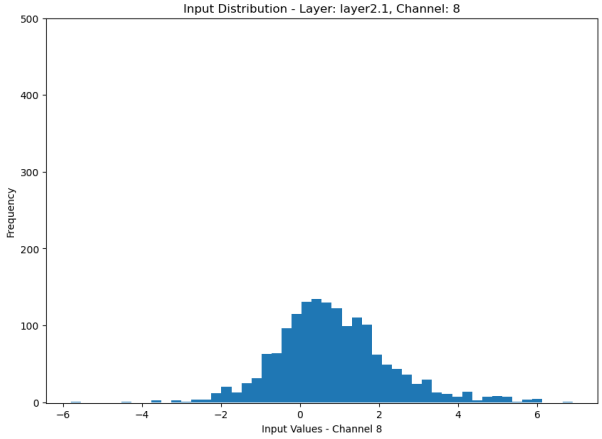

Input Distribution - Layer: layer2.1, Channel: 3
Frequency
Input Values - Channel 3
0
100
200
300
400
500
−4
−2
0
2
4
6

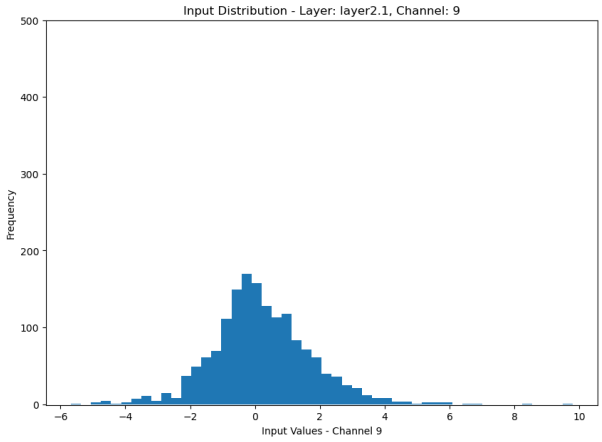

Input Distribution - Layer: layer2.1, Channel: 4
Frequency
Input Values - Channel 4

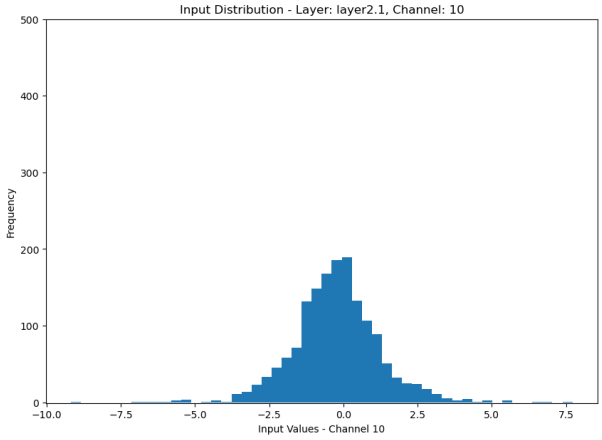

Input Distribution - Layer: layer2.1, Channel: 5
Frequency
Input Values - Channel 5

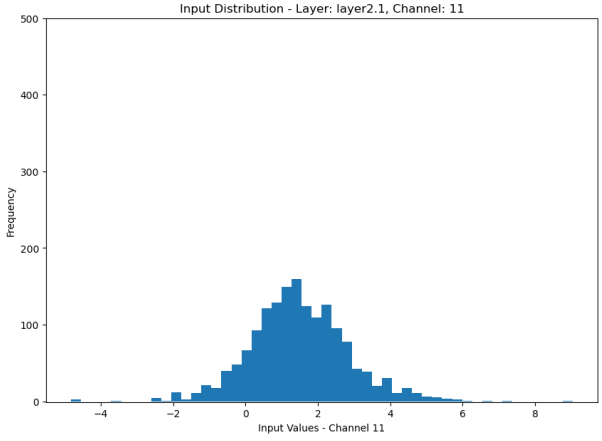

Input Distribution - Layer: layer2.1, Channel: 6
Frequency
Input Values - Channel 6

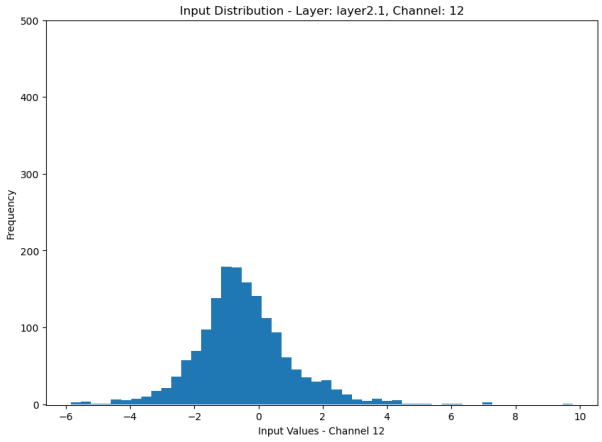

Input Distribution - Layer: layer2.1, Channel: 7
Frequency
Input Values - Channel 7

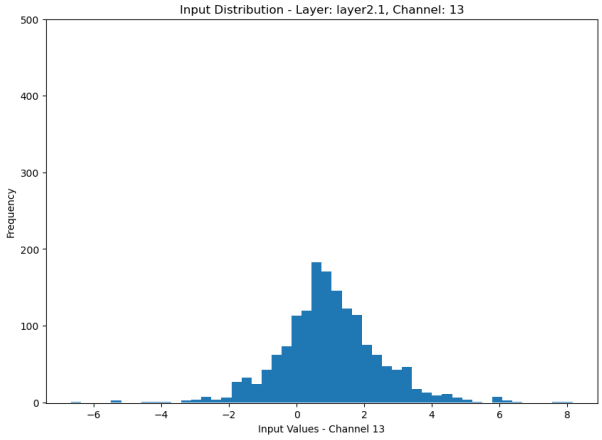

Input Distribution - Layer: layer2.1, Channel: 8
Frequency
Input Values - Channel 8

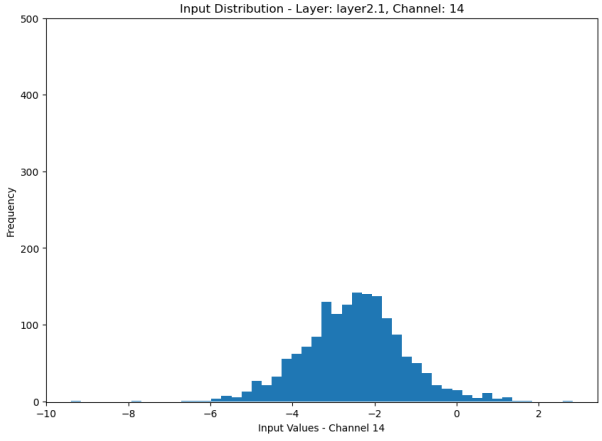

Input Distribution - Layer: layer2.1, Channel: 9
Frequency
Input Values - Channel 9

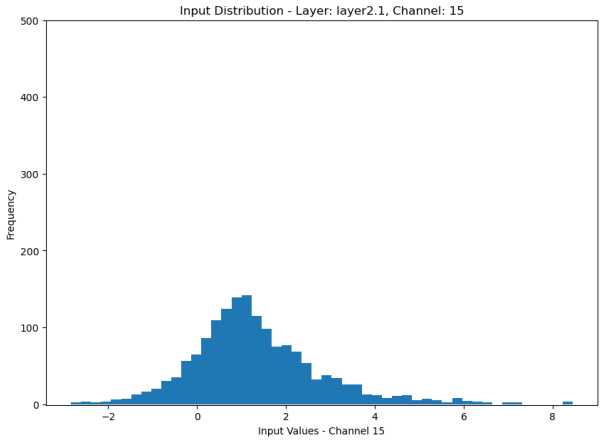

Input Distribution - Layer: layer2.1, Channel: 10
Frequency
Input Values - Channel 10


Input Distribution - Layer: layer2.1, Channel: 11
Frequency
Input Values - Channel 11

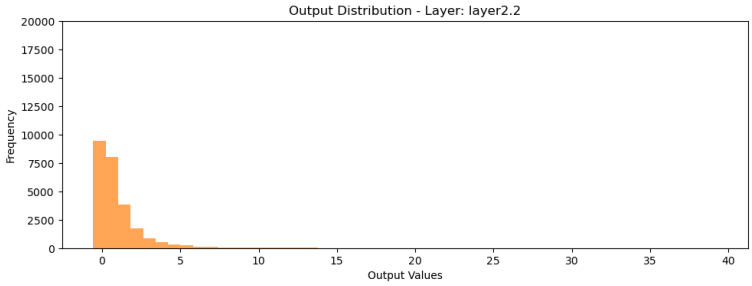

Input Distribution - Layer: layer2.1, Channel: 12
Frequency
Input Values - Channel 12


Input Distribution - Layer: layer2.1, Channel: 13
Frequency
Input Values - Channel 13


Input Distribution - Layer: layer2.1, Channel: 14
Frequency
Input Values - Channel 14



Output distribution of the second non-linear layer which is now approximated channel-by-channel with polynomial functions is as follows:



Input and output distribution of the third non-linear layer which is now approximated with a polynomial function is as follows:

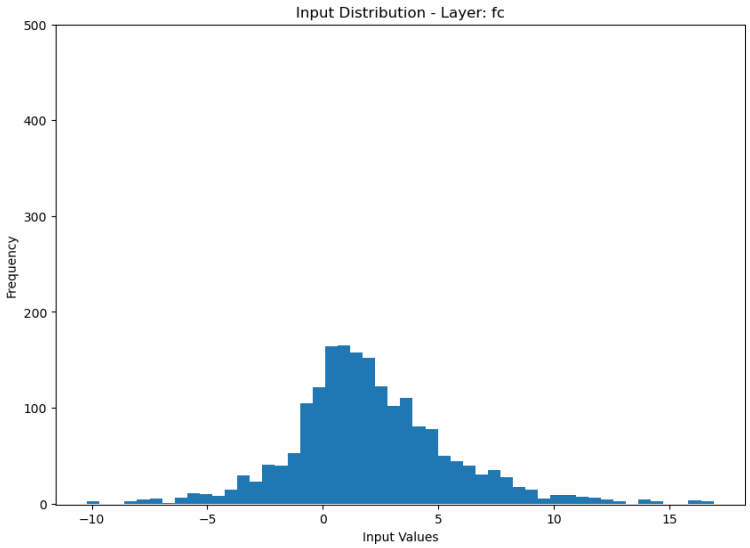


Figure 50: Input distribution for the 3rd non-linear layer after polynomial approximation

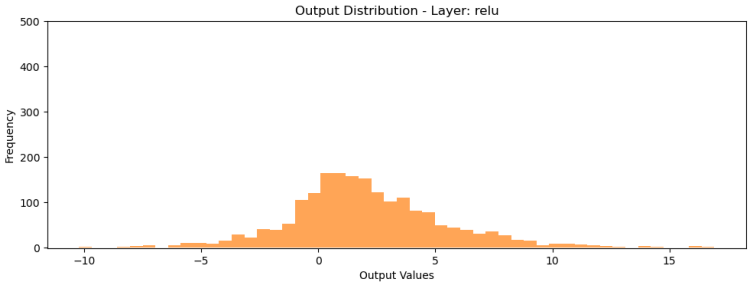


Figure 51: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and output distribution of the fourth non-linear layer which is now approximated with a polynomial function is as follows:

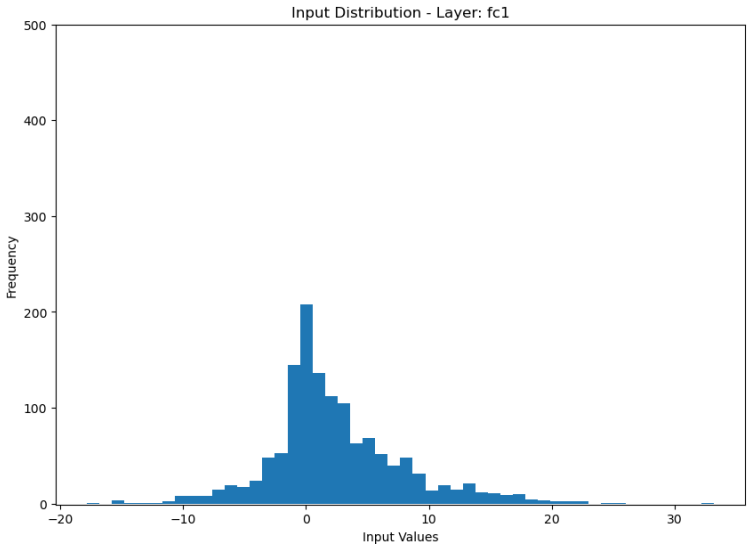


Figure 52: Input distribution for the 4th non-linear layer after polynomial approximation

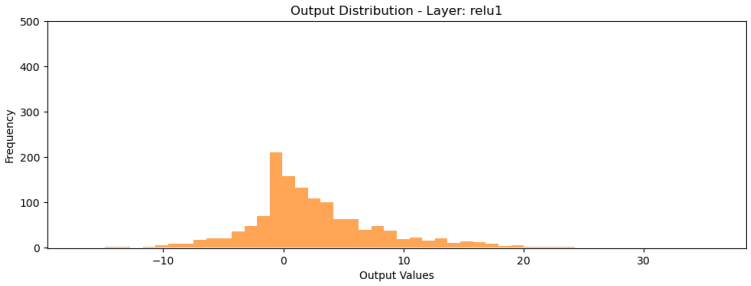


Figure 53: Output distribution for the 4th non-linear layer after polynomial approximation

4.3.3 Results

I compared the accuracy values for different polynomial orders in the range of 2-8 and different input ranges. The polynomial approximations mentioned above resulted in the closest-to-original accuracy I was able to obtain, with the accuracy of **39.862**.

4.4 Step-wise polynomial approximation without retraining

1. Approximate the first non-linear function based on the input/output distributions.
2. Re-generate the input/output distributions.
3. Approximate the following non-linear function based on the newly generated input/output distributions.
4. Return to step 2 and repeat these steps until all non-linear functions have been approximated.

4.4.1 Approximation functions

For the first non-linear layer, I used the polynomial approximations from the Section 4.3.1 because the input distribution for this layer didn't change.

For the second non-linear layer, I chose the below approximations based on the new input distribution using the Remez algorithm:

Channel 0:
- Polynomial coefficients:[0.3601975524558097, 0.4929417821162876, 0.12217708330514973]
- Polynomial order: 2
- Range: : [-3.67915803,3.9713484]
- Percentile: 99th

Channel 1:
- Polynomial coefficients: [0.24237299807969673, 0.49427383946206466, 0.18148236915522797]
- Polynomial order: 2
- Range: [-2.49811716,2.6574944]
- Percentile: 95th

Channel 2:
- Polynomial coefficients:[0.20554729258079685, 0.4850054420891712, 0.19442477973246983]
- Polynomial order: 2
- Range: [-2.40769711,1.0562396]
- Percentile: 90th

Channel 3:
- Polynomial coefficients:[0.23577154354588925, 0.5232796742117052, 0.18948183164729254]
- Polynomial order: 2
- Range: [-2.72273601,1.98166017]
- Percentile: 99th

Channel 4:
- Polynomial coefficients:[0.25006820232932914, 0.5207291449719289, 0.17746841353919388]

- Polynomial order: 2
- Range: [-2.88839861,1.6798993]
- Percentile: 95th

Channel 5:
- Polynomial coefficients:[0.403643054563497, 0.47748567575630163, 0.110548788003907]
- Polynomial order:2
- Range: [-3.45816347,4.65654997]
- Percentile: 99th

Channel 6:
- Polynomial coefficients:[-1.7318217192793877e-17, 1.0, 2.7689156241797638e-17]
- Polynomial order: 2
- Range: [0.08306662,3.53095756]
- Percentile: 90th

Channel 7:
- Polynomial coefficients:[0.25669742686700053, 0.4986577164176937, 0.1639025856010064]
- Polynomial order: 2
- Range: [-1.43709179,2.97672229]
- Percentile: 95th

Channel 8:
- Polynomial coefficients:[2.10804270092633e-17, 0.9999999999999999, 4.538352983269849e-17]
- Polynomial order: 2
- Range: [0.01773833,1.62276223]
- Percentile: 75th

Channel 9:
- Polynomial coefficients:[0.22132064836006876, 0.482276943986376, 0.2003969188204646]
- Polynomial order: 2
- Range: [-2.04932642,2.52735558]
- Percentile: 95th

Channel 10:
- Polynomial coefficients:[0.06897055035705656, 0.5227957805032868, 0.6465885589179922]
- Polynomial order: 2
- Range: [-0.79681678,0.48263964]
- Percentile: 70th

Channel 11:
- Polynomial coefficients:[-1.612465800162906e-18, 1.0, 2.039549239442568e-17]
- Polynomial order: 2
- Range:[0.05247274,3.19406693]

- Percentile: 90th

Channel 12:
- Polynomial coefficients:[0.2187040126155993, 0.5197330912310677, 0.20326512590645218]
- Polynomial order: 2
- Range: [-2.50863659,1.97047756]
- Percentile: 95th

Channel 13:
- Polynomial coefficients: [0.162856391680622, 0.7041877804207369, 0.10066139780324047]
- Polynomial order: 2
- Range: [-0.58846786,2.67275388]
- Percentile: 90th

Channel 14:
- Polynomial coefficients:[0.2627381714666811, 0.5774794673467237, 0.12037945087115477]
- Polynomial order: 2
- Range: [-1.11283011,3.30898166]
- Percentile: 95th

Channel 15:
- Polynomial coefficients:[0.05614261605192558, 0.9196646232204849, 0.023045622176184232]
- Polynomial order: 2
- Range: [-0.24936589,3.00988817]
- Percentile: 90th

The polynomial approximation for the third non-linear function ReLU was $0.012479090969812928x^2+0.9862126552402458x+0.0031307898016997007$ which was obtained by running the Remez algorithm on the range [-0.11103689,3.71812282] corresponding to 80th percentile of the input data distribution.

The final non-linear function was approximated with the polynomial $0.008547518696598477x^2+0.9910071550879954x+0.0019500843626132913$ which was obtained by running the Remez algorithm on the range [-0.09205175,3.88369029] corresponding to 70th percentile of the input data distribution.

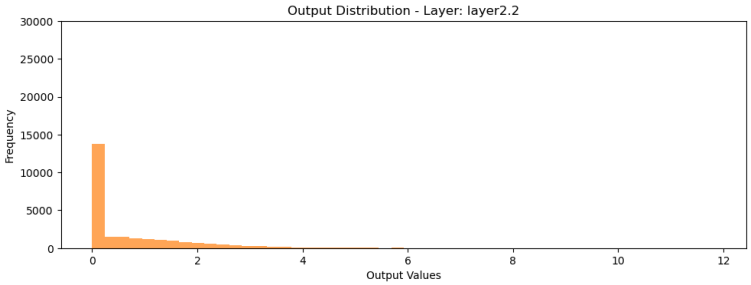
4.4.2 Input and output distribution of non-linear layers after approximation

Input and output distributions of the non-linear (now polynomial) functions of the approximated model for each iteration of the approximation are presented below:

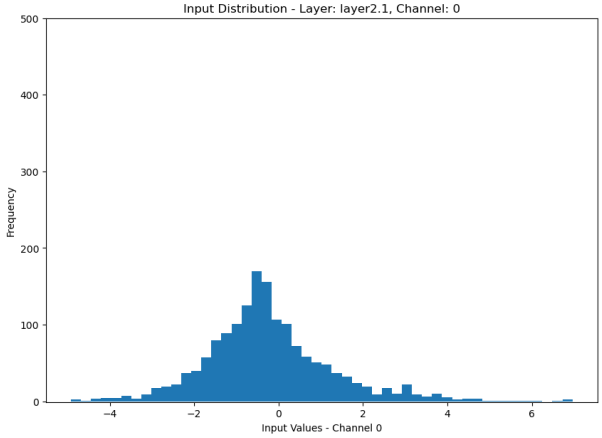
Iteration 0: Input/output distributions before any approximation were provided in the Section 3.2.1.

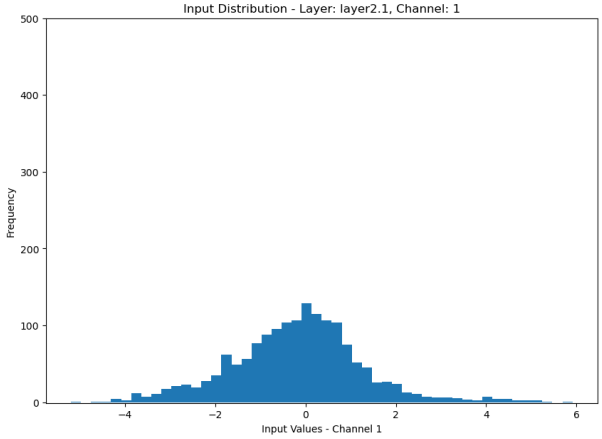
Iteration 1: Input/output distributions after the first approximation(first non-linear layer approximated):

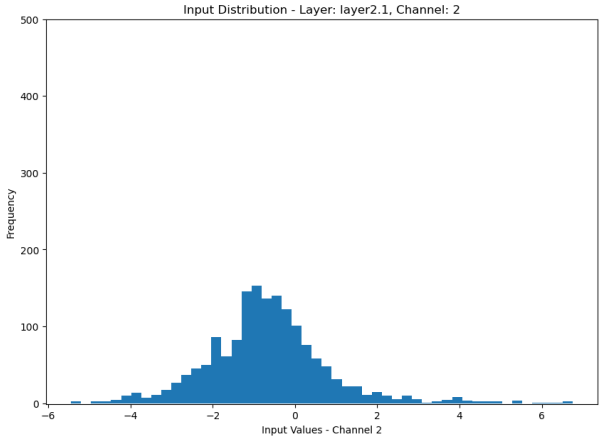
Input distribution of the first non-linear layer stays the same.
Output distribution of the first non-linear layer which is now approximated channel-by-channel with polynomial functions is as follows:

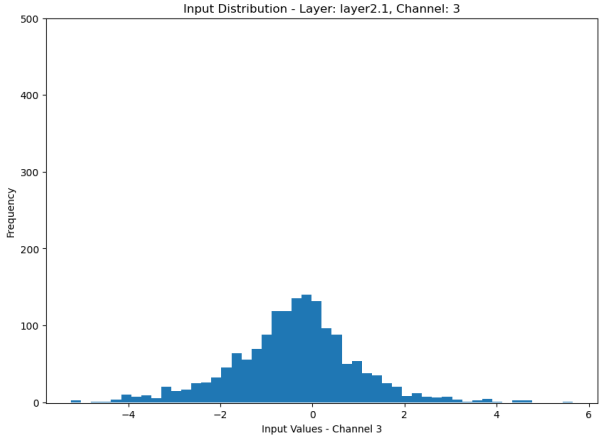


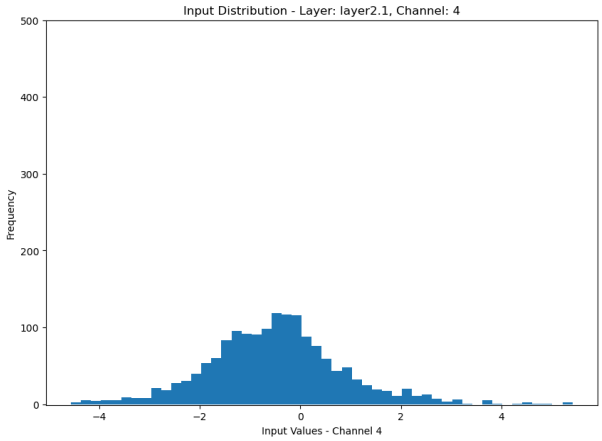
Input distributions of the second non-linear layer:

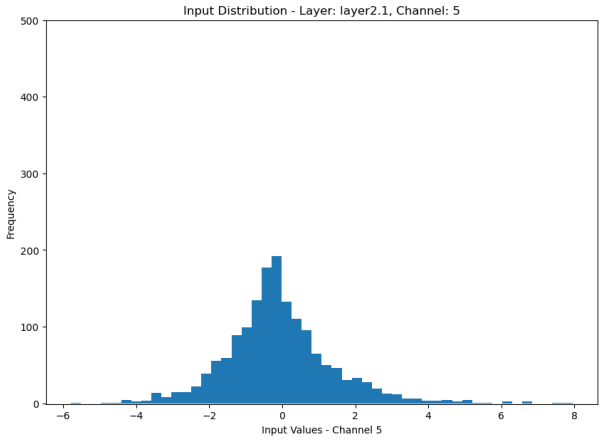


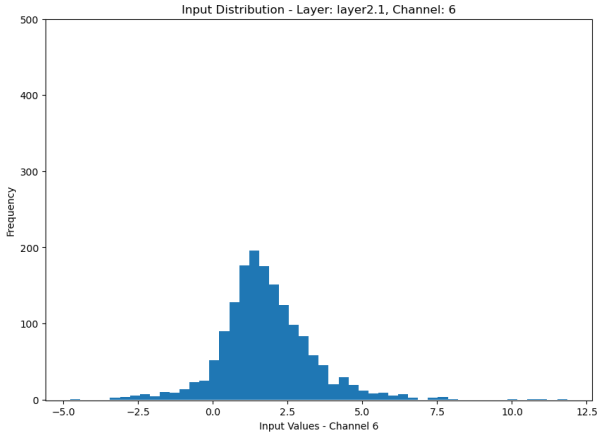


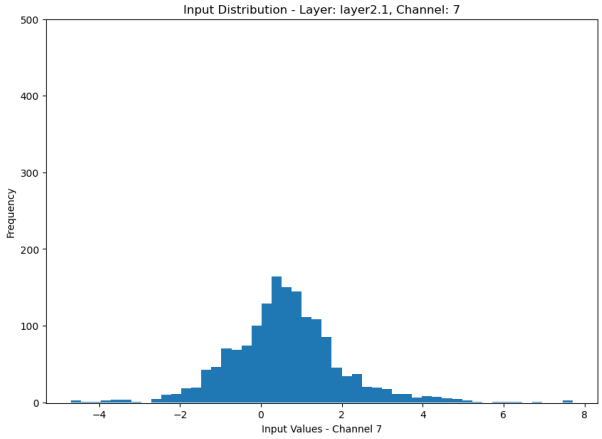

Input Distribution - Layer: layer2.1, Channel: 2
Frequency
Input Values - Channel 2
0
100
200
300
400
500
−6
−4
−2
0
2
4
6

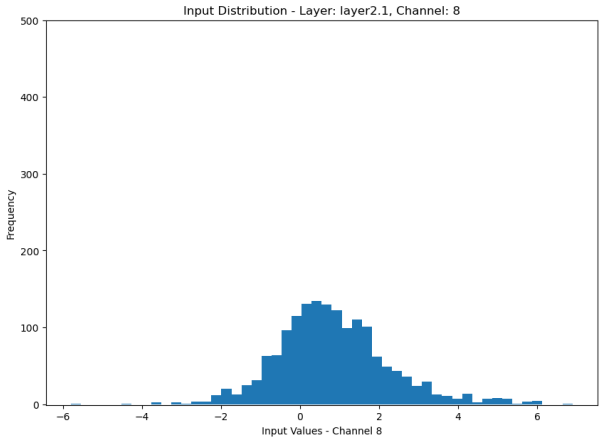

Input Distribution - Layer: layer2.1, Channel: 3
Frequency
Input Values - Channel 3
0
100
200
300
400
500
−4
−2
0
2
4
6

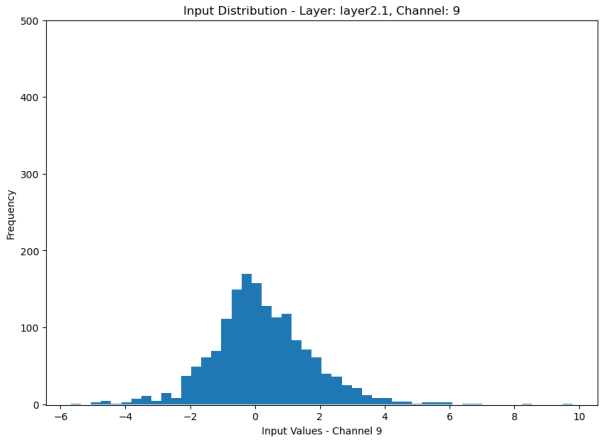

Input Distribution - Layer: layer2.1, Channel: 4
Frequency
Input Values - Channel 4
0
100
200
300
400
500
−4
−2
0
2
4

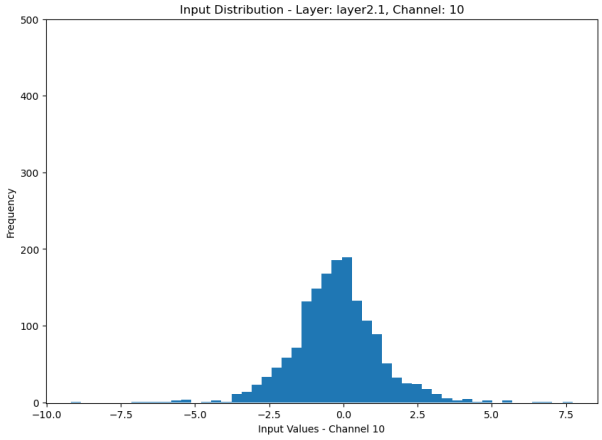

Input Distribution - Layer: layer2.1, Channel: 5
Frequency
Input Values - Channel 5
0
100
200
300
400
500
−6
−4
−2
0
2
4
6
8

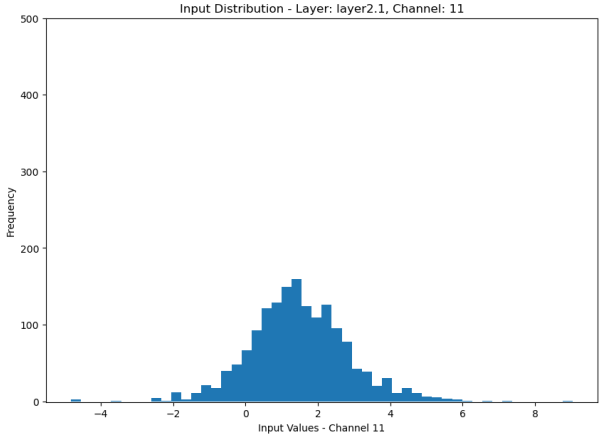

Input Distribution - Layer: layer2.1, Channel: 6
Frequency
Input Values - Channel 6

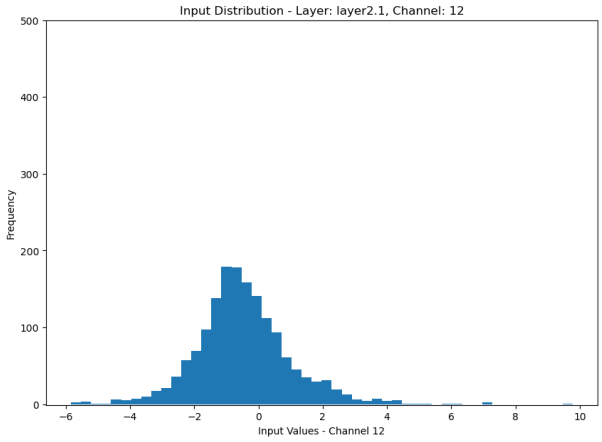

Input Distribution - Layer: layer2.1, Channel: 7
Frequency
Input Values - Channel 7

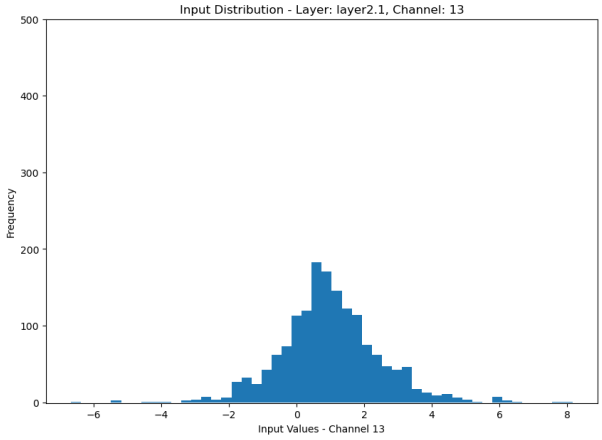

Input Distribution - Layer: layer2.1, Channel: 8
Frequency
Input Values - Channel 8

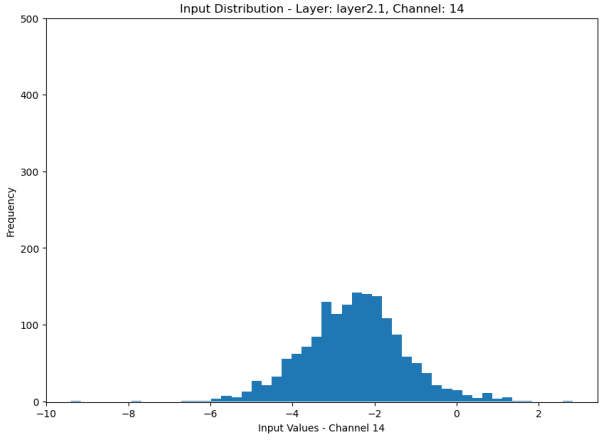

Input Distribution - Layer: layer2.1, Channel: 9
Frequency
Input Values - Channel 9

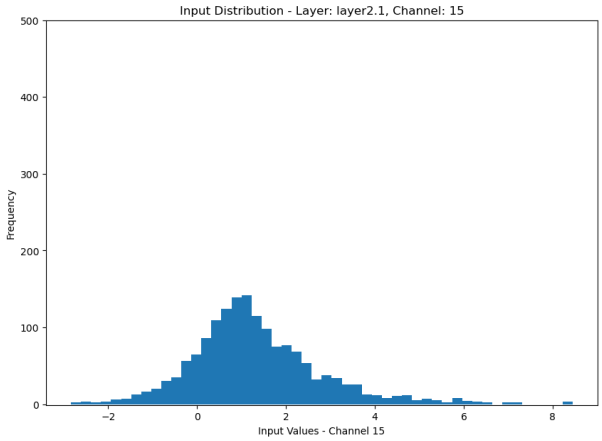

Input Distribution - Layer: layer2.1, Channel: 10
Frequency
Input Values - Channel 10


Input Distribution - Layer: layer2.1, Channel: 11
Frequency
Input Values - Channel 11

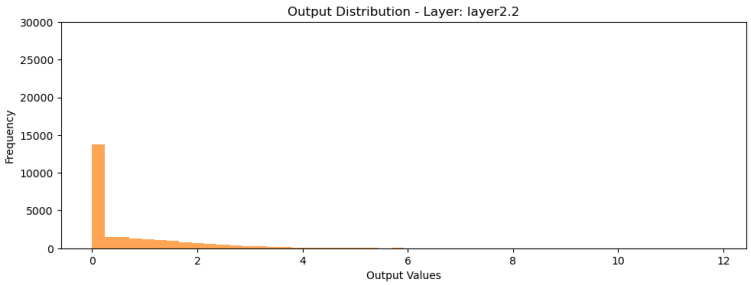

Input Distribution - Layer: layer2.1, Channel: 12
Frequency
Input Values - Channel 12


Input Distribution - Layer: layer2.1, Channel: 13
Frequency
Input Values - Channel 13


Input Distribution - Layer: layer2.1, Channel: 14
Frequency
Input Values - Channel 14



Output distribution of the second non-linear layer:



Input and output distribution of the third non-linear layer:

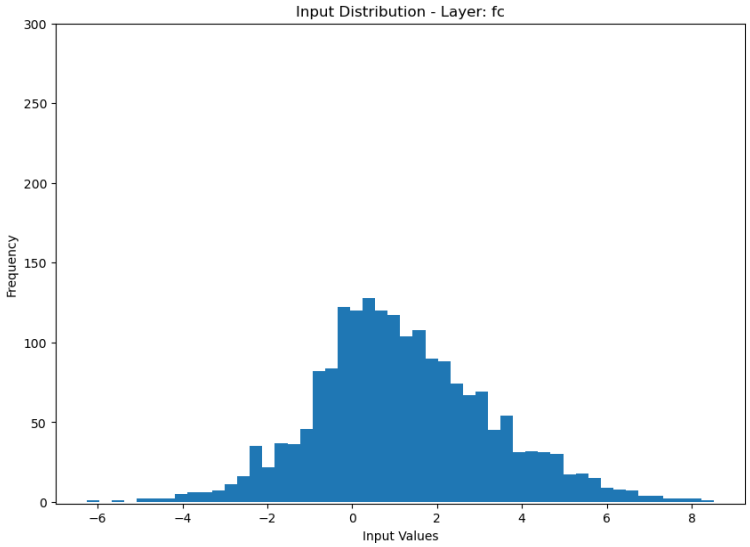


Figure 54: Input distribution for the 3rd non-linear layer after polynomial approximation

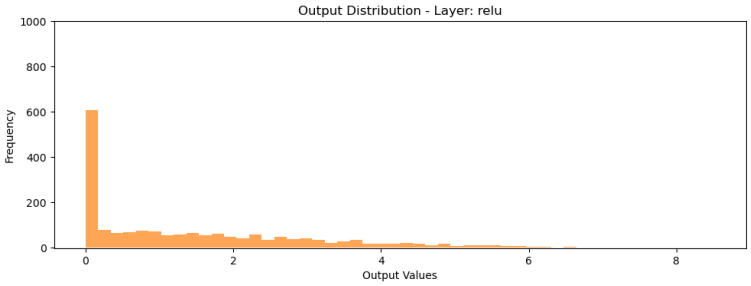


Figure 55: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and output distribution of the fourth non-linear layer:

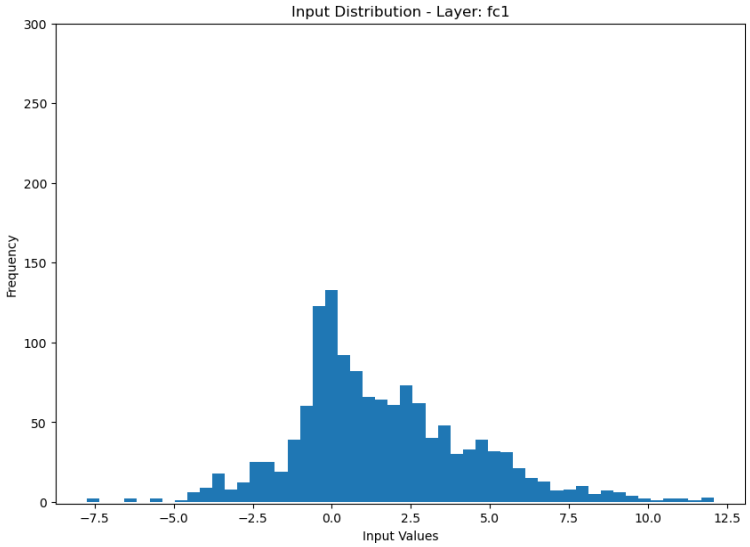


Figure 56: Input distribution for the 4th non-linear layer after polynomial approximation

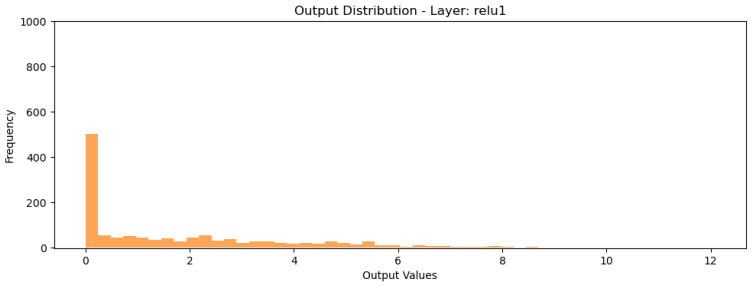


Figure 57: Output distribution for the 4th non-linear layer after polynomial approximation

Iteration 2: Input/output distributions after the second approximation (first and second non-linear layers approximated):

Input distribution of the second non-linear layer stays the same.
Output distribution of the second non-linear layer:

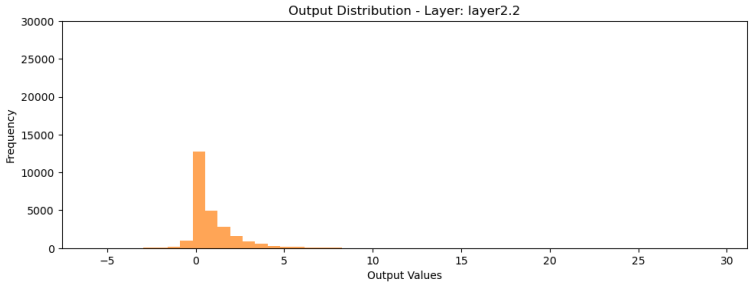


Figure 58: Input distribution for the 2nd non-linear layer after polynomial approximation

Input and output distribution of the third non-linear layer:

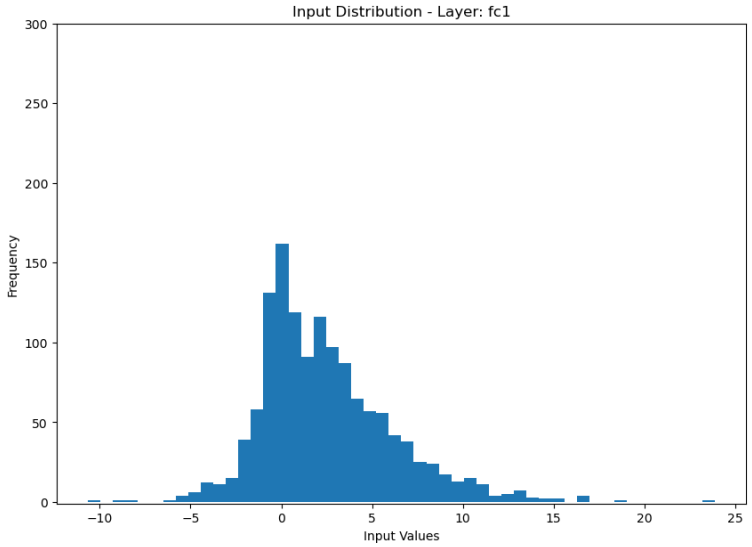


Figure 59: Input distribution for the 3rd non-linear layer after polynomial approximation

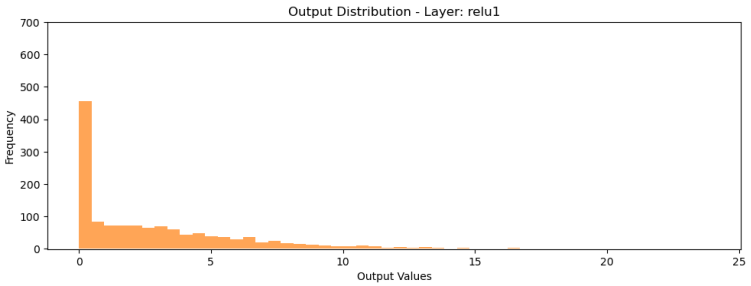


Figure 60: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and output distribution of the fourth non-linear layer:

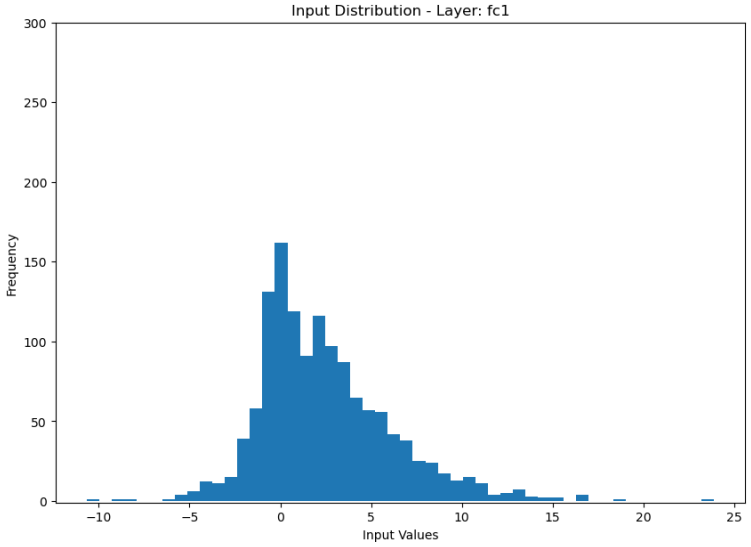


Figure 61: Input distribution for the 4th non-linear layer after polynomial approximation

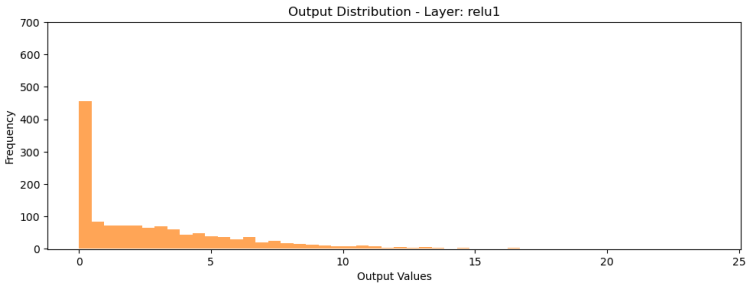


Figure 62: Output distribution for the 4th non-linear layer after polynomial approximation

Iteration 3: Input/output distributions after the third approximation (first, second, and third non-linear layers approximated):

Input distribution of the third non-linear layer stays the same.
Output distribution of the third non-linear layer:

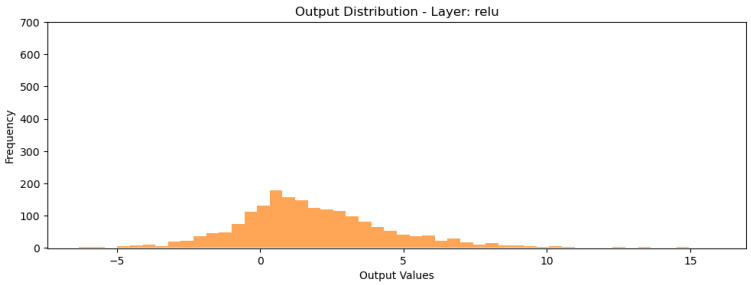


Figure 63: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and output distribution of the fourth non-linear layer:

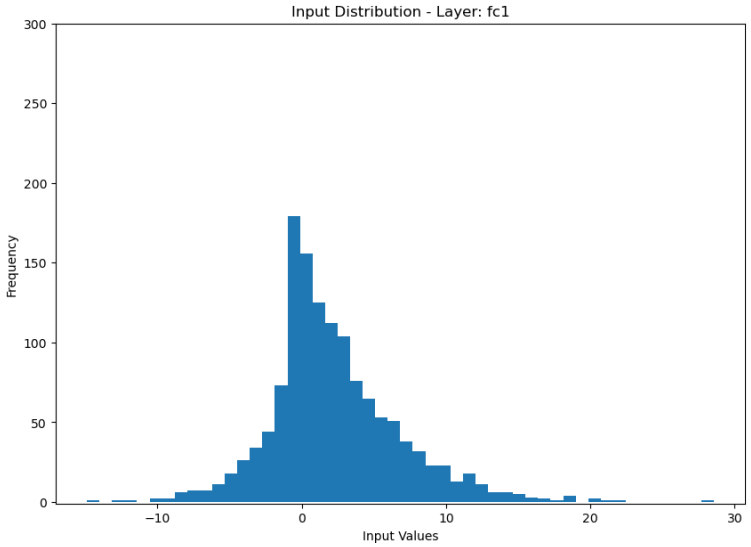


Figure 64: Input distribution for the 4th non-linear layer after polynomial approximation

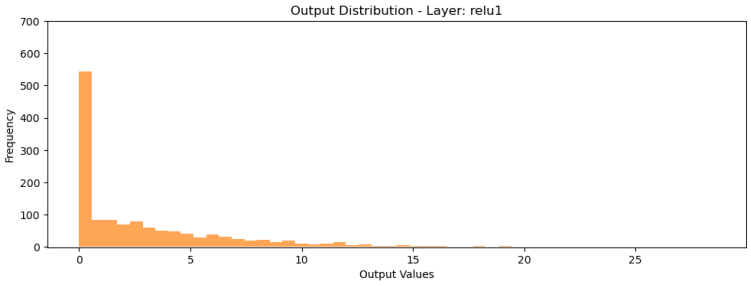


Figure 65: Output distribution for the 4th non-linear layer after polynomial approximation

Iteration 4: Input/output distributions after the fourth approximation (all non-linear layers approximated):

Input distribution of the fourth non-linear layer stays the same.
Output distribution of the fourth non-linear layer:

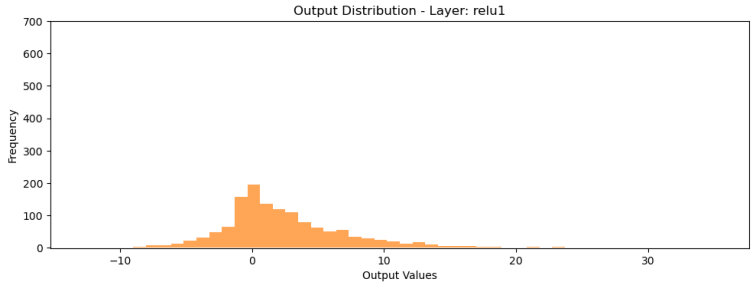


Figure 66: Ouput distribution for the 4th non-linear layer after polynomial approximation

4.4.3 Results

I compared the accuracy values for different polynomial orders in the range of 2-8 and different input ranges. The above approximations were the most optimal ones I could find and the test accuracy after all layers have been approximated with the polynomials listed above is **37.788**.

4.5 Step-wise polynomial approximation with retraining

1. Approximate the first non-linear function based on the input/output distributions.
2. Re-train the remaining model.
3. Re-generate the input/output distributions.
4. Approximate the following non-linear function based on the newly generated input/output distributions.
5. Return to step 2 and repeat these steps until all non-linear functions have been approximated.

How to re-train the model:
Consider that your model is $y = f_n(...f_2(f_1(x))...)$, where x is the input, y is the output, and f_i is a non-linear function. For simplicity, I omitted the linear parts.

First, approximate $f_1(\cdot)$ with a polynomial function $p_1(\cdot)$ based on the input/ouput distribution of the $f_1(\cdot)$. Then, create a model starting from the following layer, i.e., $y = f_n(...f_2(z)...)$, where $z = p_1(x)$, load the weights from the previous iteration (for the first iteration, load the weights of the original model), and re-train the model. Repeat the process until all layers have been approximated.

There is no retraining after the last layer has been approximated.

4.5.1 Approximation functions

For the first non-linear layer, I used the polynomial approximations from the Section 4.3.1 because the input distribution for this layer didn't change.

Then, I created a new model and trained it using 50 epochs(best validation accuracy was 56.970) and based on the new input distribution obtained, I approximated the second non-linear function (layer2) channel-by-channel with the following polynomials:

Channel 0:
- Polynomial coefficients:[0.01973892715823211, 0.9223482548216043, 0.06129323554336796]
- Polynomial order: 2
- Range: [-0.08832088,1.09286619]
- Percentile: 65th

Channel 1:
- Polynomial coefficients:[0.09342920956888064, 0.4656541151329062, 0.4002078881590805]
- Polynomial order: 2
- Range: [-1.11355584,0.4450728]

- Percentile: 65th

Channel 2:
- Polynomial coefficients:[0.0107410312530202, 0.8831661092247495, 0.2519647243624689]
- Polynomial order: 2
- Range: [-0.04233991,0.4043503]
- Percentile: 55th

Channel 3:
- Polynomial coefficients:[0.10988945022038751, 0.5238619127021986, 0.4067890204969612]
- Polynomial order: 2
- Range: [-1.27052701,0.79305671]
- Percentile: 75th

Channel 4:
- Polynomial coefficients:[7.784213263804564e-17, 0.9999999999999998, 1.641704170593912e-16]
- Polynomial order: 2
- Range: [0.15048368,1.08832796]
- Percentile: 60th

Channel 5:
- Polynomial coefficients:[0.08628411829835123, 0.513314138786741, 0.5119250198452968]
- Polynomial order:2
- Range: [-0.97254519,0.83820606]
- Percentile: 70th

Channel 6:
- Polynomial coefficients:[0.03996638453594949, 0.4759778885681728, 1.1191493891396345]
- Polynomial order: 2
- Range: [-0.32736983,0.4620404]
- Percentile: 65th

Channel 7:
- Polynomial coefficients:[0.07151102376628243, 0.5069314817621764, 0.5735567075038763]
- Polynomial order: 2
- Range: [-0.38145911,0.83269496]
- Percentile: 65th

Channel 8:
- Polynomial coefficients:[0.07102927675473336, 0.517129695736803, 0.5586751061870793]
- Polynomial order: 2
- Range: [-0.36091138,0.83363375]
- Percentile: 65th

Channel 9:

- Polynomial coefficients:[2.7580798755977298e-11, 0.9999999996400696, 7.928054879258184e-10]
- Polynomial order: 2
- Range: : [0.00690717,0.46333283]
- Percentile: 55th

Channel 10:
- Polynomial coefficients: [0.09845492828556106, 0.5144528453674345, 0.4435848531030707]
- Polynomial order: 2
- Range: [-1.13651649,0.61110998]
- Percentile: 70th

Channel 11:
- Polynomial coefficients:[0.10740035904501062, 0.5521507428065723, 0.32591653846760443]
- Polynomial order: 2
- Range: [-0.48372234,1.30677761]
- Percentile: 70th

Channel 12:
- Polynomial coefficients:[0.08418701514659278, 0.5025402707847245, 0.52209209135266]
- Polynomial order: 2
- Range: [-0.90917884,0.88522851]
- Percentile: 70th

Channel 13:
- Polynomial coefficients:[0.09079232429612061, 0.4958199737663953, 0.4842609669199901]
- Polynomial order: 2
- Range: [-0.9448482,0.98861812]
- Percentile: 70th

Channel 14:
- Polynomial coefficients:[0.07076202654330488, 0.6477607253203372, 0.3213270903444657]
- Polynomial order: 2
- Range: [-0.2676157,1.01174935]
- Percentile: 65th

Channel 15:
- Polynomial coefficients:[0.022329881848304037, 0.9056356723634531, 0.07960037167795865]
- Polynomial order: 2
- Range: [-0.09404435,1.02764484]
- Percentile: 65th

Then, I created a new model without the first and the second non-linear layers and trained it using 250 epochs (best validation accuracy was 57.430). Based on the new input distribution obtained, I approximated the third non-linear function (relu) with the polynomial $0.11035531961461492x^2+0.8938175727578919x+$

0.020320929911514796 which was obtained by running the Remez algorithm on the range [-0.45110965,4.54232235] corresponding to 85th percentile of the input data distribution.

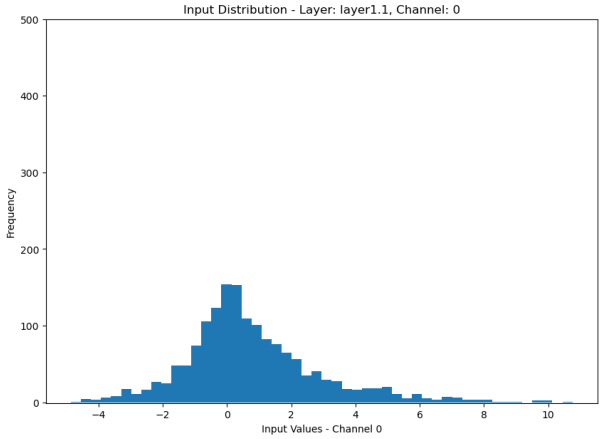
After that, I created another new model with only the third non-linear layer but there was no need to retrain this model because it now consisted of only ReLU and linear layers. Based on the new input distribution obtained, I approximated the fourth non-linear function (relu1) with the polynomial $(-4.0199156395916486e{-}17)x^2{+}1.0x{+}(-1.6716630433850442e{-}17)$ which was obtained by running the Remez algorithm on the range [0.32503663,3.29383993] corresponding to 60th percentile of the input data distribution.

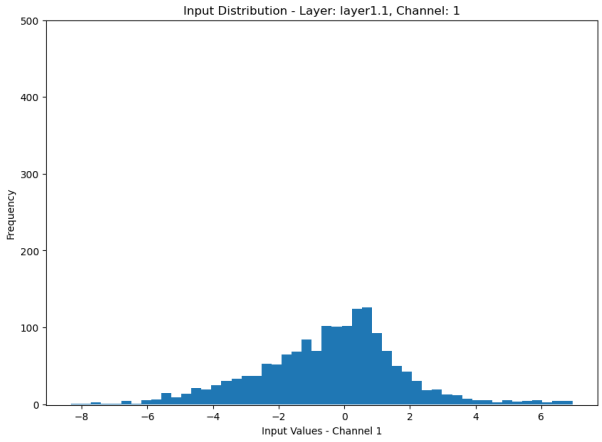
4.5.2 Input and output distribution of non-linear layers after approximation

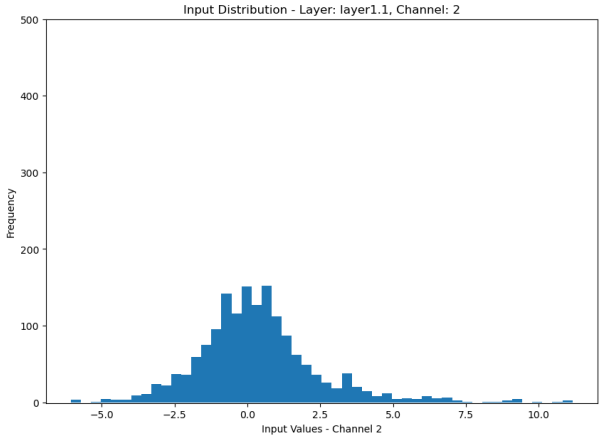
Iteration 0: Input/output distributions before any approximation were provided in the Section 4.3.1.

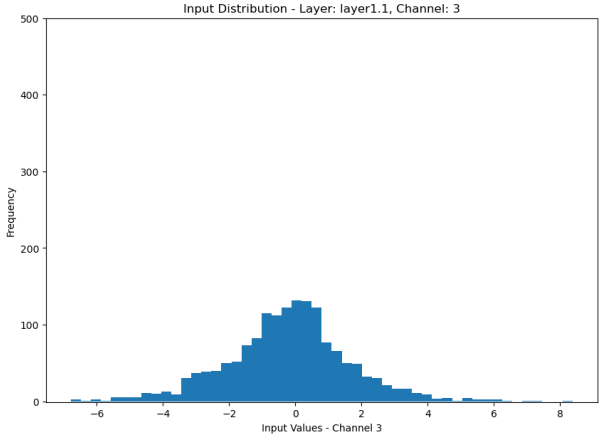
Iteration 1: Input/output distributions after the first approximation and after re-training.

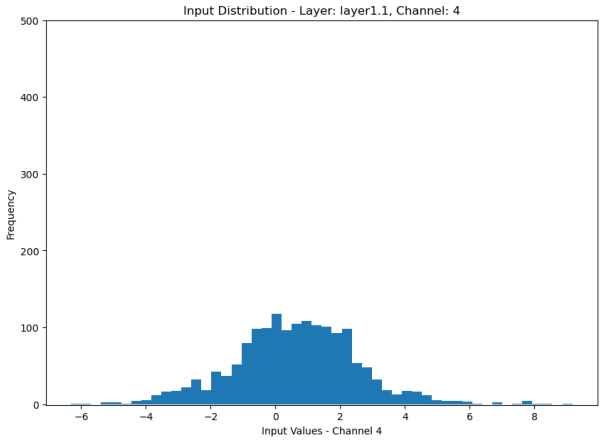
Input distributions of the second non-linear layer:

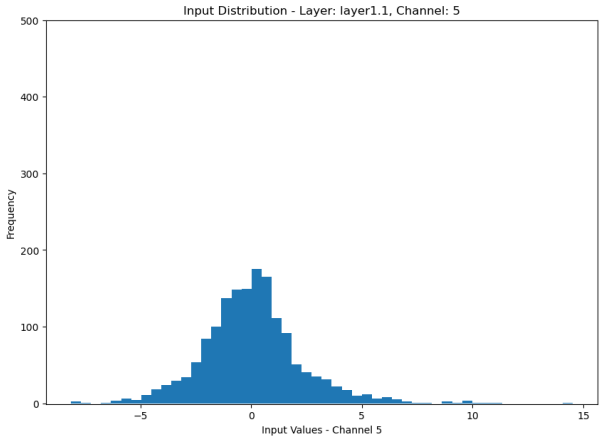


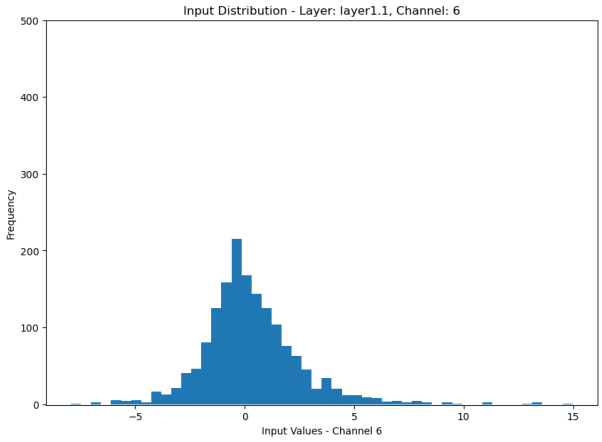

Input Distribution - Layer: layer1.1, Channel: 1
Frequency
Input Values - Channel 1

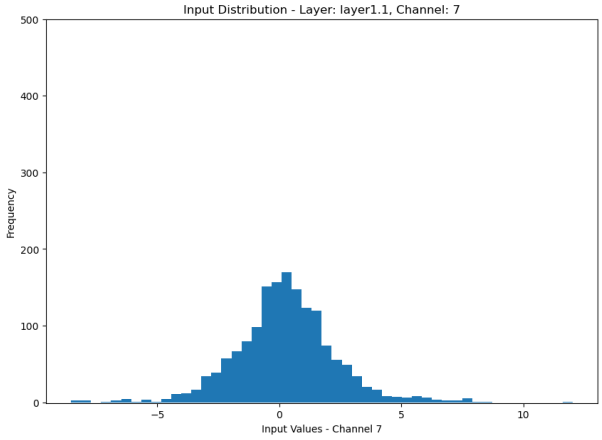

Input Distribution - Layer: layer1.1, Channel: 2
Frequency
Input Values - Channel 2

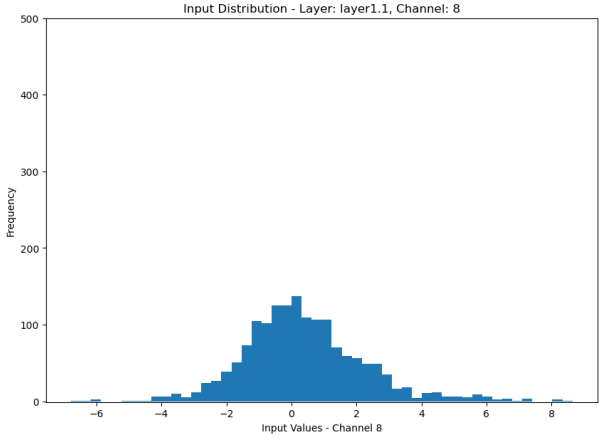

Input Distribution - Layer: layer1.1, Channel: 3
Frequency
Input Values - Channel 3

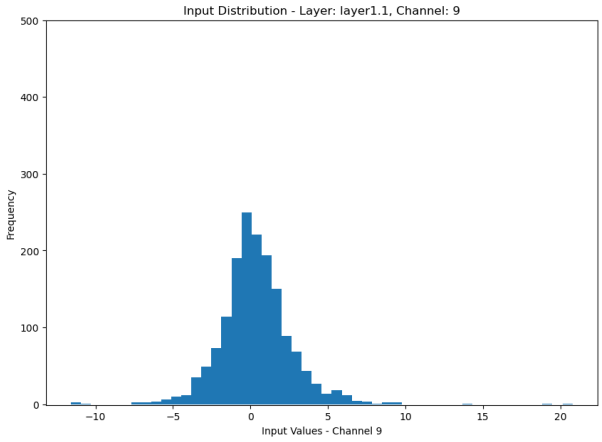

Input Distribution - Layer: layer1.1, Channel: 4
Frequency
Input Values - Channel 4
0
100
200
300
400
500
−6
−4
−2
0
2
4
6
8

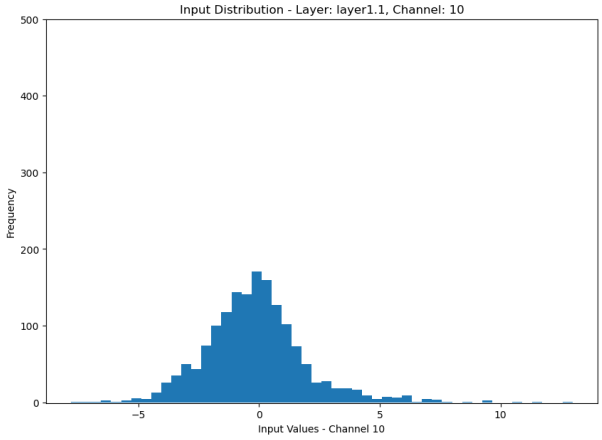

Input Distribution - Layer: layer1.1, Channel: 5
Frequency
Input Values - Channel 5
0
100
200
300
400
500
−5
0
5
10
15

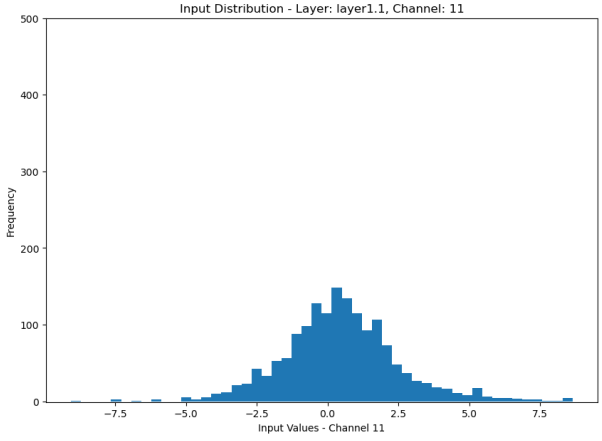

Input Distribution - Layer: layer1.1, Channel: 6
Frequency
Input Values - Channel 6

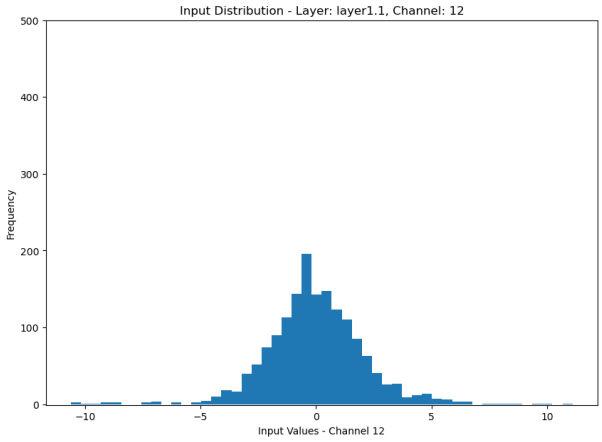

Input Distribution - Layer: layer1.1, Channel: 7
Frequency
Input Values - Channel 7

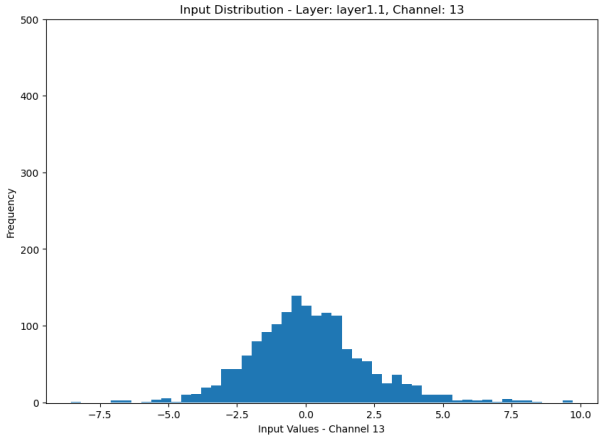

Input Distribution - Layer: layer1.1, Channel: 8
Frequency
Input Values - Channel 8

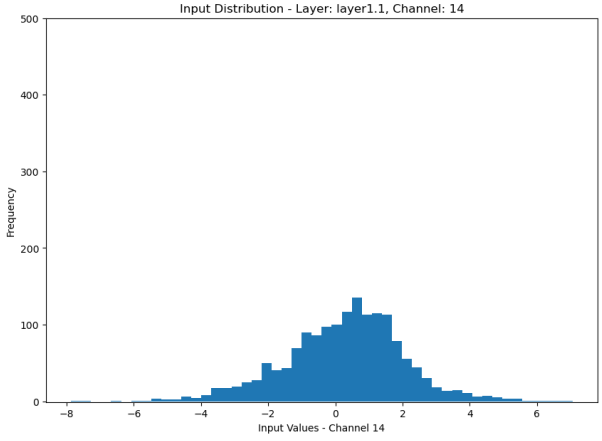

Input Distribution - Layer: layer1.1, Channel: 9
Frequency
Input Values - Channel 9

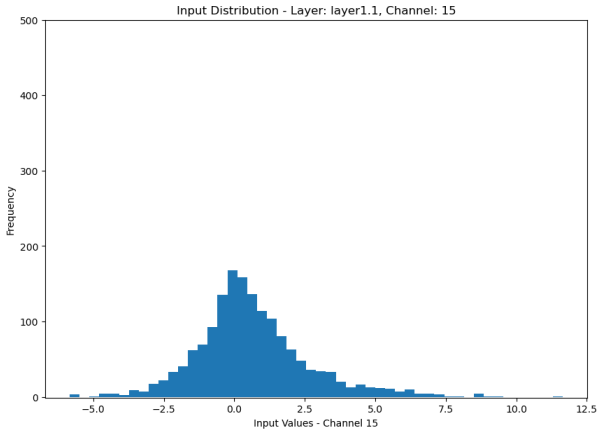

Input Distribution - Layer: layer1.1, Channel: 10
Frequency
Input Values - Channel 10


Input Distribution - Layer: layer1.1, Channel: 11
Frequency
Input Values - Channel 11


Input Distribution - Layer: layer1.1, Channel: 12
Frequency
Input Values - Channel 12


Input Distribution - Layer: layer1.1, Channel: 13
Frequency
Input Values - Channel 13


Input Distribution - Layer: layer1.1, Channel: 14
Frequency
Input Values - Channel 14



Input and output distribution of the third non-linear layer:

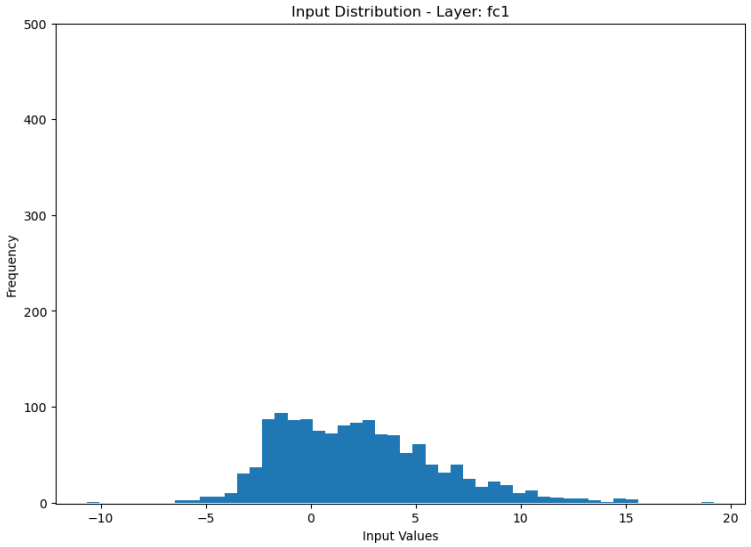


Figure 67: Input distribution for the 3rd non-linear layer after polynomial approximation

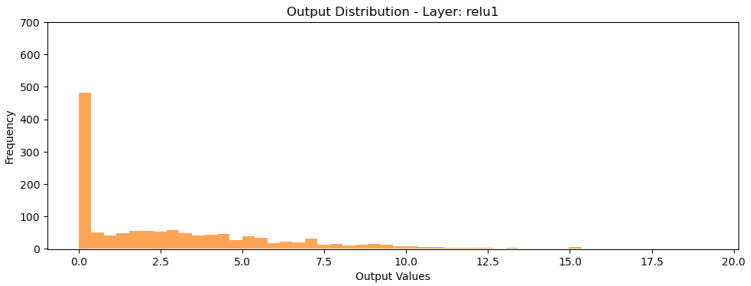


Figure 68: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and output distribution of the fourth non-linear layer:

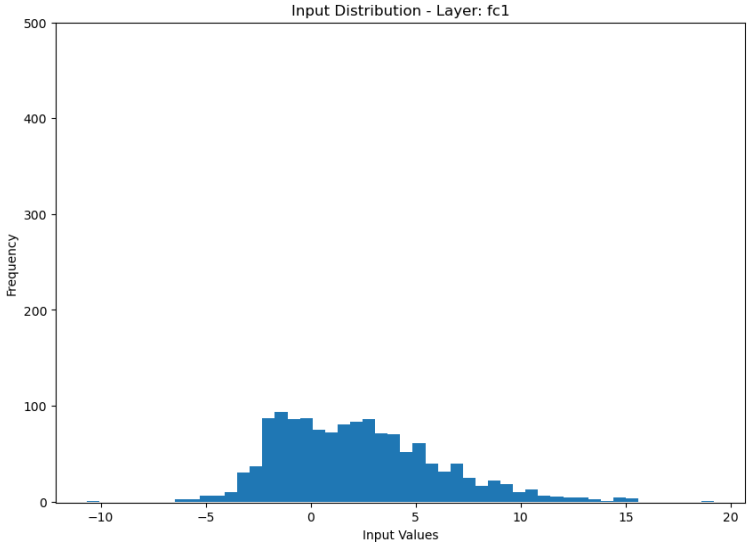


Figure 69: Input distribution for the 4th non-linear layer after polynomial approximation

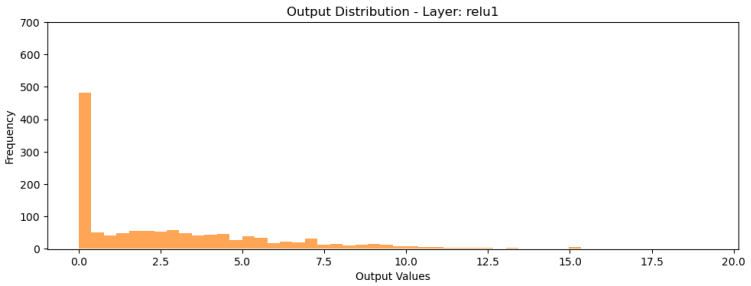


Figure 70: Output distribution for the 4th non-linear layer after polynomial approximation

Iteration 2: Input/output distributions after the second approximation and after re-training.

Input and distribution of the third non-linear layer:

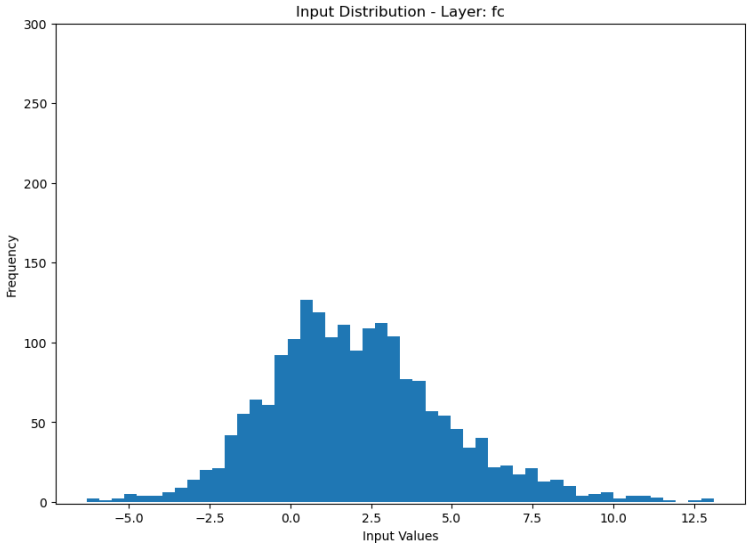


Figure 71: Input distribution for the 3rd non-linear layer after polynomial approximation

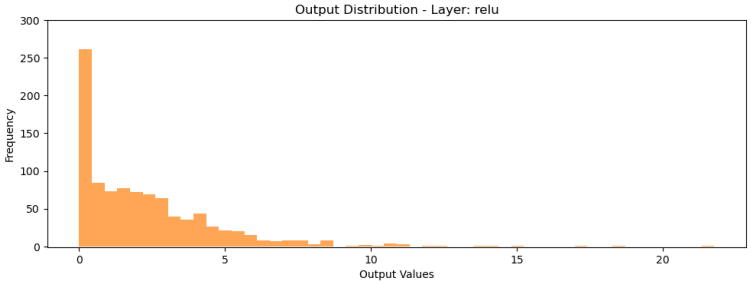


Figure 72: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and distribution of the fourth non-linear layer:

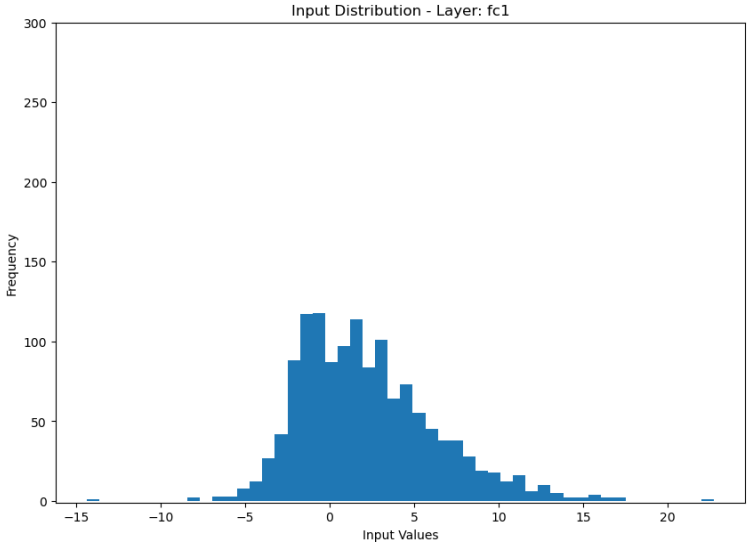


Figure 73: Input distribution for the 4th non-linear layer after polynomial approximation

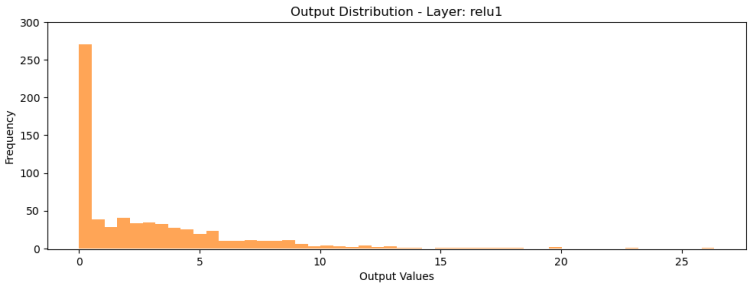


Figure 74: Output distribution for the 4th non-linear layer after polynomial approximation

Iteration 3: Input/output distributions after the third and fourth approximations (no need to re-train because there are only ReLU and linear layers in the new model):

Input and distribution of the third non-linear layer:

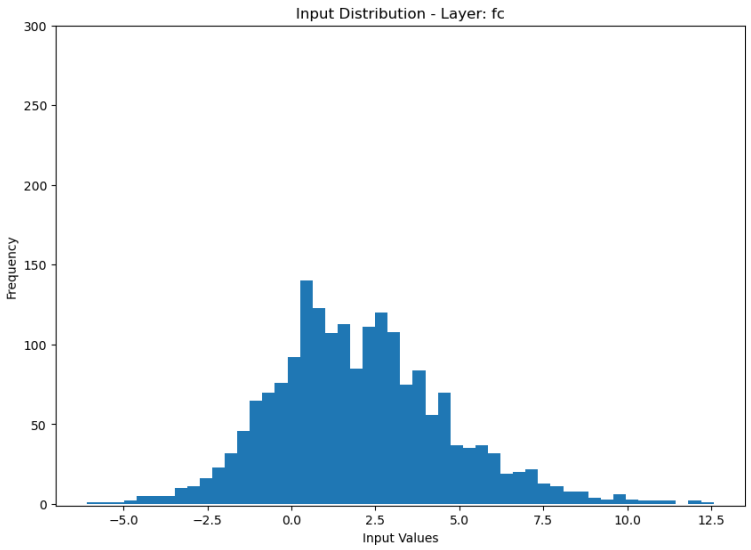


Figure 75: Input distribution for the 3rd non-linear layer after polynomial approximation

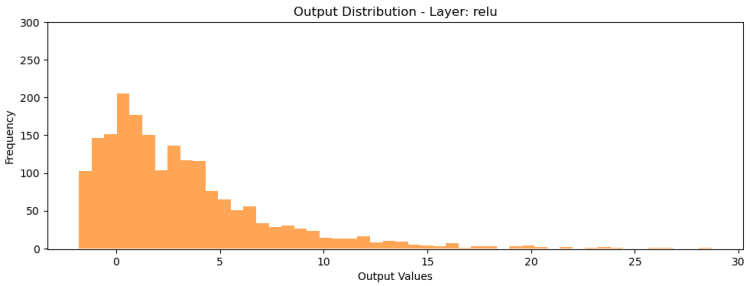


Figure 76: Output distribution for the 3rd non-linear layer after polynomial approximation

Input and distribution of the fourth non-linear layer:

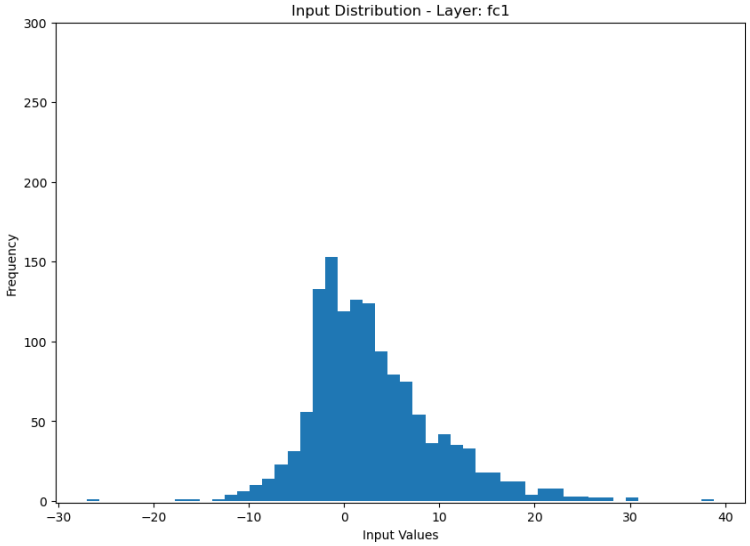


Figure 77: Input distribution for the 4th non-linear layer after polynomial approximation

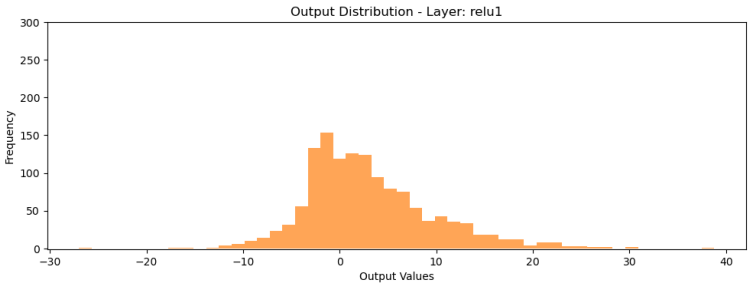


Figure 78: Output distribution for the 4th non-linear layer after polynomial approximation

4.5.3 Results

The test accuracy after all layers have been approximated is **56.567**.