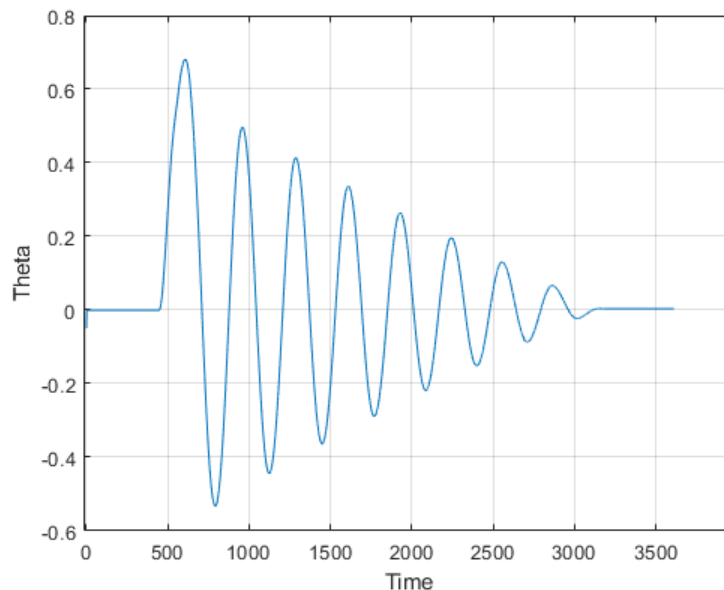


Sprawozdanie ICS – Mańka Lisek, Lab2

1. Feedforwardnet – Dane 1

Dane wykorzystane do trenowania sieci neuronowej są pobrane z Reaction Pendulum. Stanowi to rozszerzenie zadania wykonywanego na Laboratorium Problemowym.

```
theta = dane_NN.signals(1).values + 1.58;  
  
time = dane_NN.time;  
  
figure  
plot(theta)  
xlabel('Time')  
ylabel('Theta')  
grid on
```



Dane zostały podzielone zgodnie z teorią przedstawioną na laboratoriach. Początkowo sieć neuronowa była trenowana dla 2 warstw.

```
y_begin_sample = 602;  
y_end_sample = 2875;  
training_data_input_1 = theta(y_begin_sample-2:y_end_sample-2)
```

```
training_data_input_1 = 2274×1  
0.6777
```

```

0.6770
0.6782
0.6797
0.6798
0.6796
0.6807
0.6804
0.6800
0.6808
:

```

```

training_data_input_2 = theta(y_begin_sample-1:y_end_sample-1)
training_data_output = theta(y_begin_sample:y_end_sample)      :
x_train = [training_data_input_1'; training_data_input_2']

```

```

x_train = 2×2274
    0.6777    0.6770    0.6782    0.6797    0.6798    0.6796    0.6807 ...
    0.6770    0.6782    0.6797    0.6798    0.6796    0.6807    0.6804

```

```

% time = time(600:2873)
y_train = training_data_output'

```

```

y_train = 1×2274
    0.6782    0.6797    0.6798    0.6796    0.6807    0.6804    0.6800 ...

```

Następnie została wytrenowana sieć neuronowa z użyciem funkcji feedforwardnet.

```

net = feedforwardnet(2, 'trainlm');
net = train(net, x_train, y_train);
view(net);
y_net = net(x_train);
perf = perform(net,y_train, y_net)
net = feedforwardnet(2, 'trainlm');
net = train(net, x_train, y_train);

```

```

y_net = net(x_train);

```

```

sample_time = 0.01;
perf = perform(net,y_train, y_net)

```

```

perf = 3.8247e-06

```

```

gensim(net, sample_time)

```

```

ans = 'untitled1'

```

```

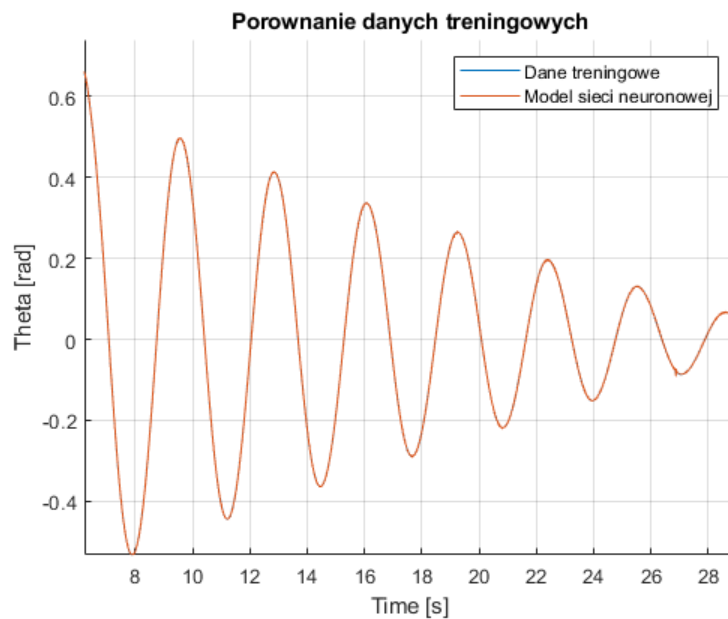
figure();
hold on
plot(time, y_train)

```

```

plot(time, y_net)
hold off
legend("Dane treningowe", 'Model sieci neuronowej')
xlabel('Time [s]')
ylabel('Theta [rad]')
title('Porownanie danych treningowych')
grid on

```



```
err = immse(y_train, y_net)
```

```
err = 3.8455e-06
```

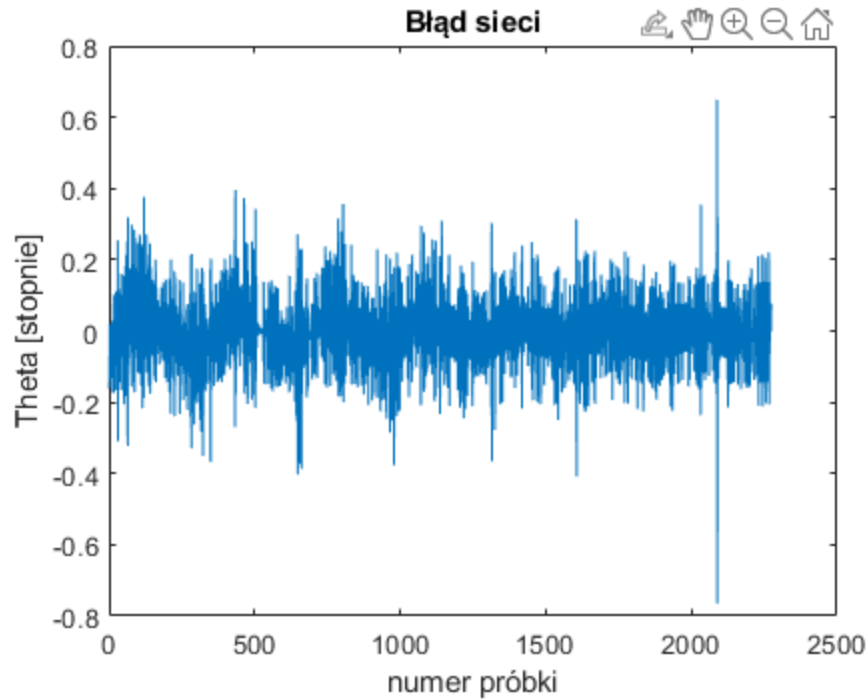
```
err = y_net-y_train
```

```
err = 1×2274
    -0.0018    -0.0005     0.0013     0.0003    -0.0013     0.0013     0.0002 ...
```

```
err_degree = err * 180/pi
```

```
err_degree = 1×2274
    -0.1040    -0.0299     0.0722     0.0167    -0.0750     0.0744     0.0108 ...
```

```
figure;
plot(err)
```



```

layers = 1:4;
layers_test = zeros(length(layers), length(training_data_output));
for i=layers
    net = feedforwardnet(i, 'trainlm');
    net = train(net, x_train, y_train);

    y_net = net(x_train);
    layers_test(i, :) = y_net;
    perf = perform(net,y_train, y_net)
end

```

```

perf = 3.8506e-06
perf = 3.8419e-06
perf = 3.8511e-06
perf = 3.8247e-06

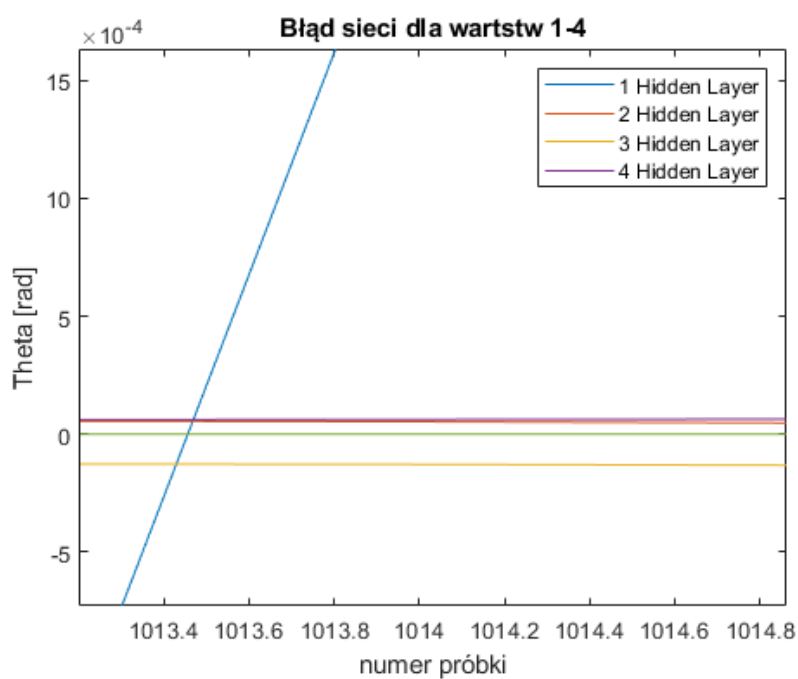
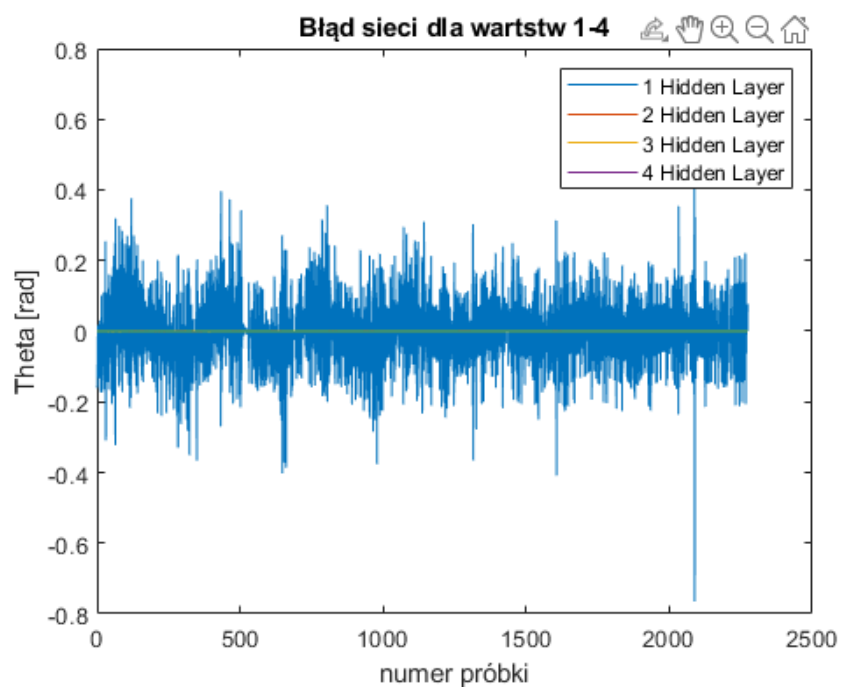
```

```

err = layers_test - y_net;
hold on
plot(err(1, :))
plot(err(2, :))
plot(err(3, :))
plot(err(4, :))
hold off
legend('1 Hidden Layer', '2 Hidden Layer', '3 Hidden Layer', '4 Hidden Layer')

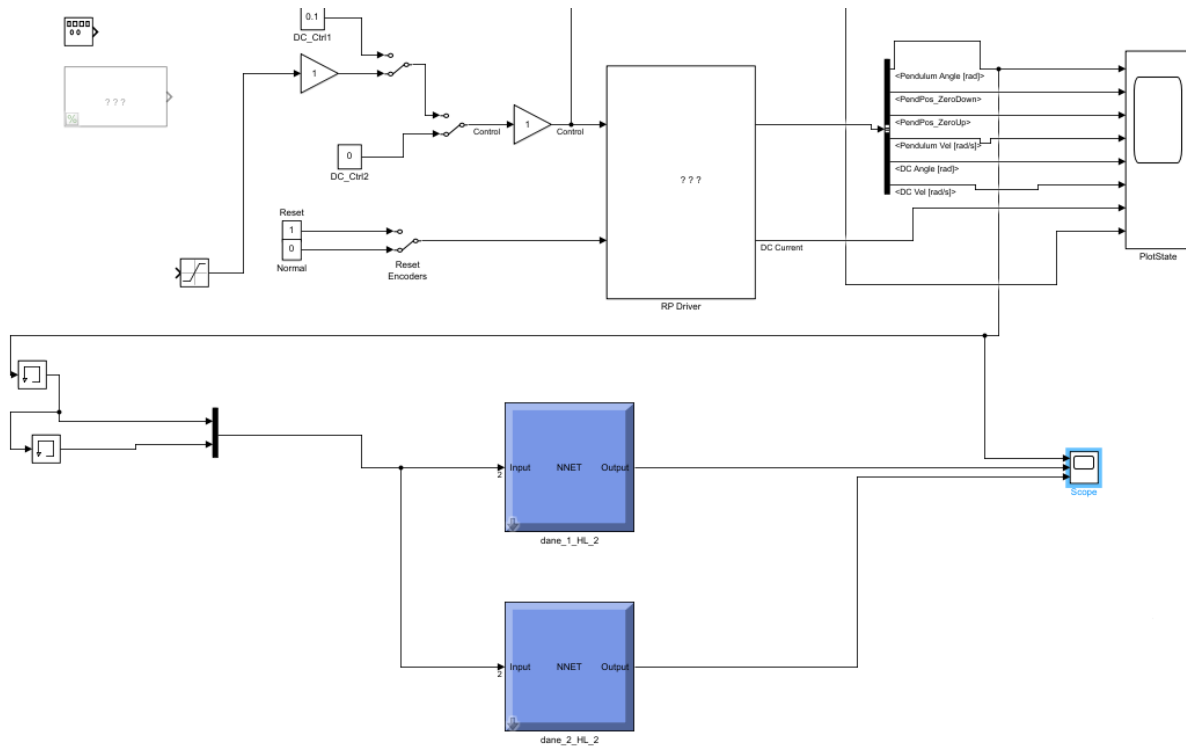
```

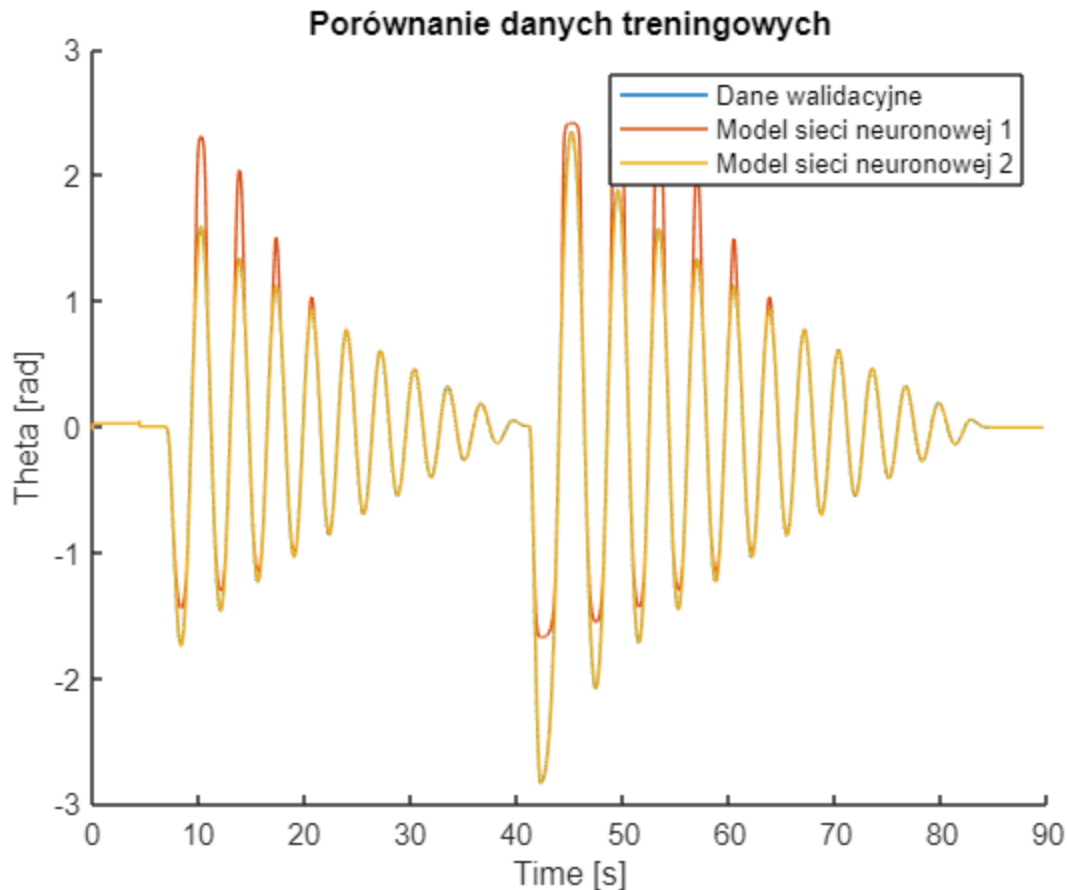
Analiza błędów sieci



Porównanie dla danych o większym zakresie ruchu wahadła.

Rysunek poniżej przedstawia walidację dwóch sieci neuronowych sprawdzonych na labolatoriach. Druga sieć neuronowa jest przetrenowana na większej ilości danych.





Narxnet

Narxnet w środowisku Matlab to zaawansowane narzędzie, które profesjonalnie implementuje opisane procedury. Aby skorzystać z tej funkcji w pełni efektywnie, konieczne jest przekształcenie formatu danych na typ 'cell'. Następnie należy dostarczyć jako wejście zmienną θ , reprezentującą wynikową zmienną sieci. Kluczowym elementem jest precyzyjne określenie parametrów działania sieci, takich jak opóźnienia w danych, które mają być zaimplementowane w celu generowania danych treningowych wewnątrz sieci.

Narxnet w Matlab to narzędzie o dużej elastyczności, umożliwiające skuteczne tworzenie modeli neuronowych. Dostosowywanie opóźnień danych pozwala na dokładne dostosowanie sieci do specyfiki danego zadania. Ten proces staje się szczególnie kluczowy w przypadku danych zawierających złożone wzorce czasowe. Eksperymentowanie z różnymi konfiguracjami staje się niezbędne w celu osiągnięcia optymalnej wydajności modelu.

```
%wyjscie jest wejściem  
x_train = training_data_output'
```

```
x_train = 1×4101  
    0.1725    0.2099    0.2472    0.2846    0.3220    0.3588    0.3958 ...
```

```
inputSeries = num2cell(x_train)
```

```
inputSeries = 1×4101 cell
```

	1	2	3	4	5	6	7	...
1	0.1725	0.2099	0.2472	0.2846	0.322	0.3588	0.3958	

```
targetSeries = num2cell(x_train)
```


```
targetSeries = 1×4101 cell
```

	1	2	3	4	5	6	7	...
1	0.1725	0.2099	0.2472	0.2846	0.322	0.3588	0.3958	

```
net = narxnet(1:2, 1, 2);  
[Xs,Xi,Ai,Ts] = preparets(net,targetSeries,{},inputSeries)  
net = train(net,Xs,Ts,Xi,Ai);
```


Network Diagram

Training Results

Training finished: Stopped manually 

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	321	1000
Elapsed Time	-	00:07:28	-
Performance	0.811	8.17e-08	0
Gradient	3.45	9.71e-06	1e-07
Mu	0.001	1e-07	1e+10
Validation Checks	0	0	6

Training Algorithms

Data Division: Random dividerand

Training: Levenberg-Marquardt trainlm

Performance: Mean Squared Error mse

Calculations: MEX

Training Plots

Performance

Training State

Error Histogram

Regression

Time-Series Response

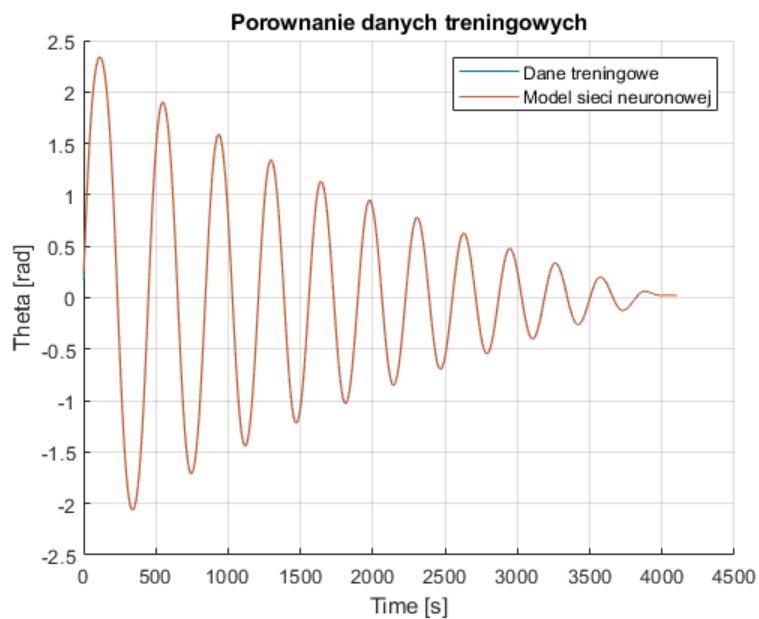
Error

Input-Error Cross-correlation

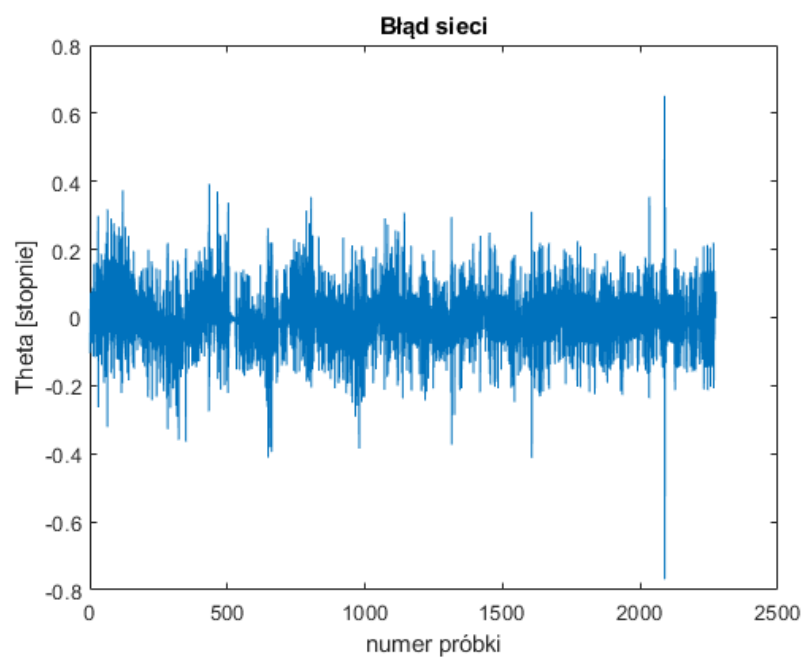
```
[y_net,Xf,Af] = net(Xs,Xi,Ai);  
y_net = cell2mat(y_net)
```

```
y_net = 1x4099  
        0.2471    0.2845    0.3218    0.3592    0.3953    0.4327    0.4685 ...
```

```
figure();  
hold on  
plot(training_data_output)  
plot(y_net)  
hold off  
legend("Dane treningowe", 'Model sieci neuronowej')  
xlabel('Time [s]')  
ylabel('Theta [rad]')  
title('Porównanie danych treningowych')  
grid on
```



```
figure;  
plot(err_degree)  
xlabel('numer próbki')  
ylabel('Theta [stopnie]')  
title('Błąd sieci')
```



Wnioski i spostrzeżenia

Dla sieci feedforward dane zostały dopasowane z błędem (pomiędzy próbkami) poniżej 0.005 radiana dla 1 warstwowej sieci neuronowej.

Dla większej liczby warstw od 2 – 4 błąd jest na poziomie e^{-5} .

Problemem okazał się fakt, iż sieć została przetrenowana dla danych w zakresie -0.6 do 0.6 radianów. Po zaaplikowaniu danych z zakresu -3.14 do 3.14 radianów okazało się, że błąd dla wartości spoza danych treningowych osiąga wartości 1 radiana (pomiędzy próbkami). Z tego powodu przetrenowano sieć na nowych danych, w pełnym zakresie wahań, co poskutkowało znaczącą poprawą.

Dzięki użyciu bloczka 'gensim' i 'scope' możliwa była obserwacja działania modelu w czasie rzeczywistym.

