PDF of the Most Important Information for HW2:

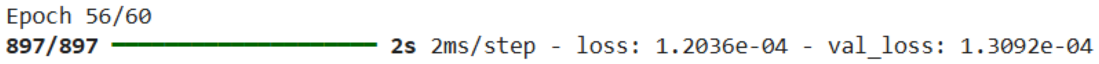
Answers to Questions:

1. Increasing the number of units per layer helped bring the error down in general. This was especially true for orders 2 and 3.
2. Increasing the number of units per layer increasing the complexity and thereby the computational abilities of the network. This is why the error decreased as the model is able to learn more complex patterns.
3. We use sigmoid activation instead of ReLu because we know that we are modelling a Volterra Series which is a differentiable, smooth function. Thus, we will use sigmoid which is also a smooth differentiable function so that we stay in the same class of functions (we are approximating a differentiable, smooth function with a differentiable, smooth function) which will have better accuracy than using ReLu. The other activation function we could have used would be the Tanh function (hyperbolic tangent function) which is a shifted version of the sigmoid. Thus, either sigma or tanh would be good functions to use as they are smooth and differentiable. We don't want to use ReLu because it is not differentiable at all values.

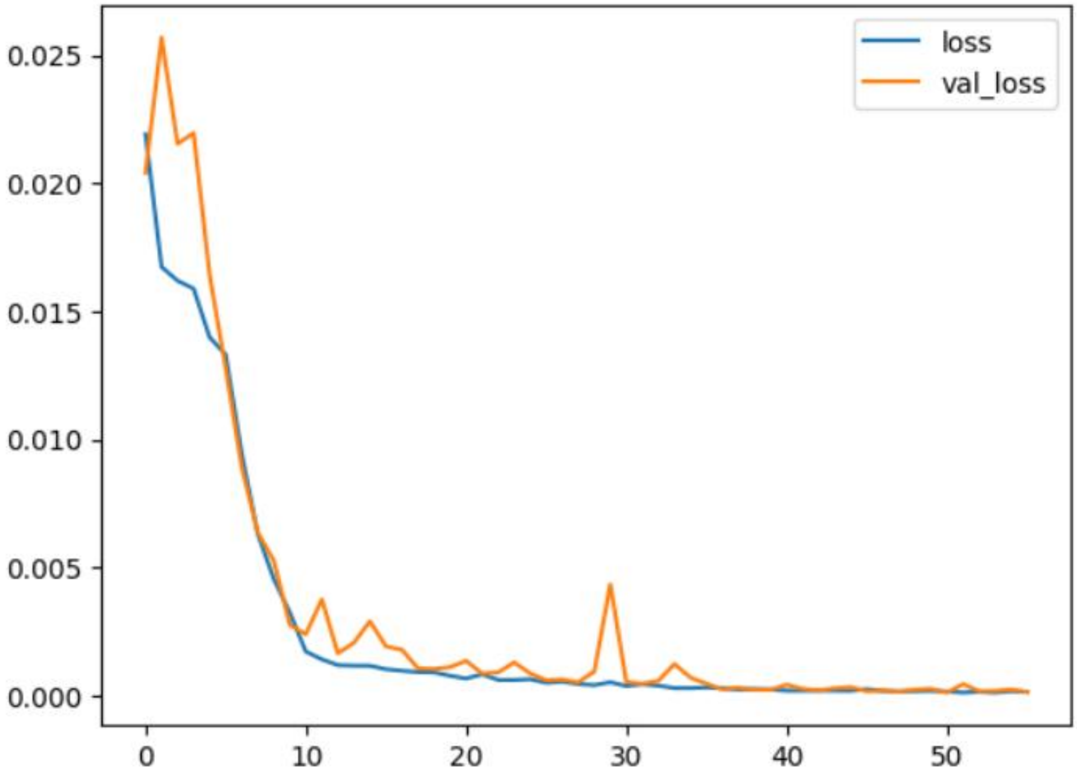
Another parameter to analyze in the future would be the patience parameter. Would increasing the patience parameter past 5 be useful for the situations where the model bounded around, or would the model continue to bounce around and never decrease even if the patience parameter were increased? In general, as the order of Volterra Series increased the model began to suffer from oscillating val error which lead to the early stopping criteria being activated during the first 20 epochs. Thus, the network suffered from overfitting more often as the order increased.

Order = 2	Best Val loss	Best Loss	Epoch #
20, 20	0.0021	0.0011	25
20, 40	0.0019	0.0011	25
40, 20	4.0203e-04	3.7454e-04	45
40, 40	**1.3092e-04**	**1.2036e-04**	**60**

Order 2: 40, 40



```
Epoch 56/60
897/897 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 1.2036e-04 - val_loss: 1.3092e-04
```



After running several iterations, I found that 60 epochs was best, as past 60 epochs the model did not improve.

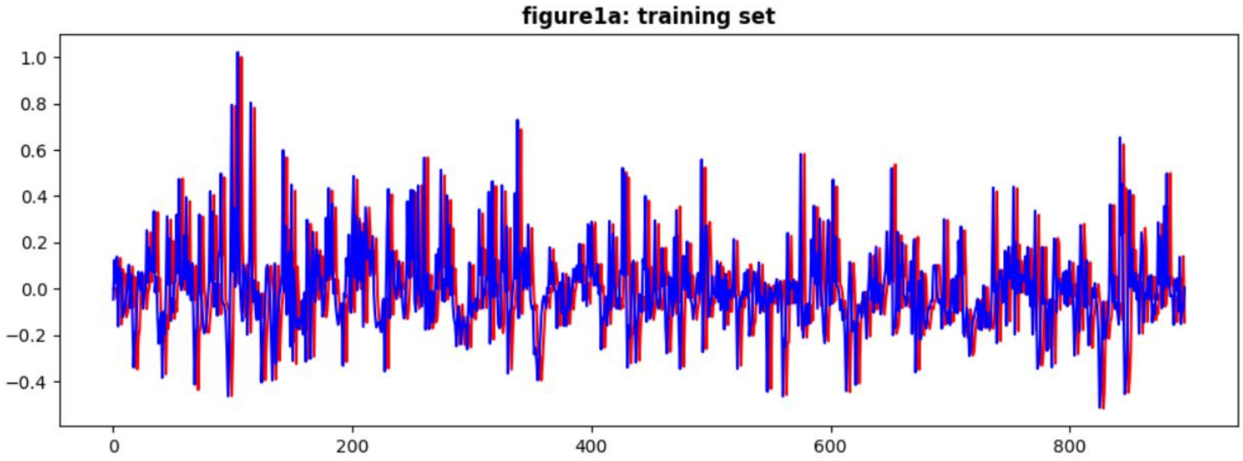


figure1a: training set

figure1b: test set

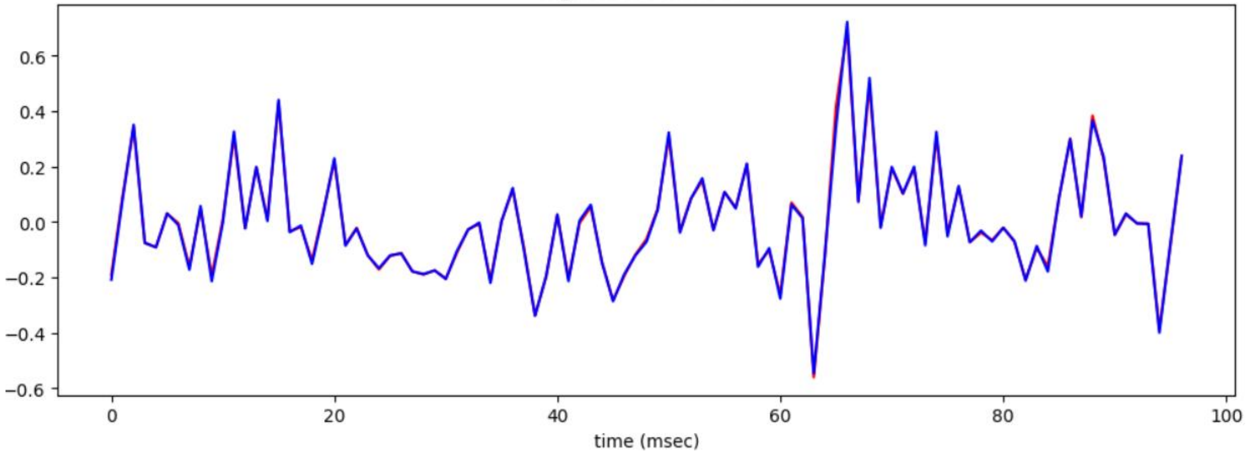
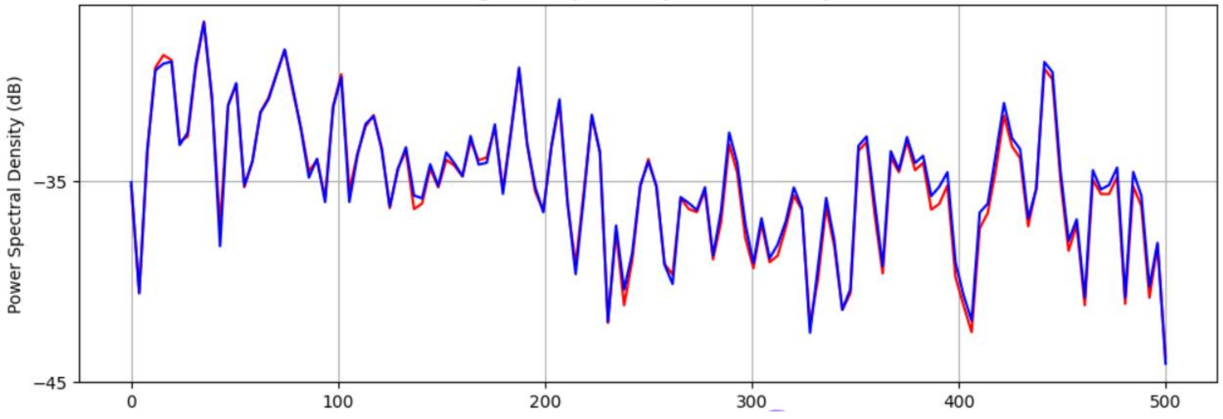


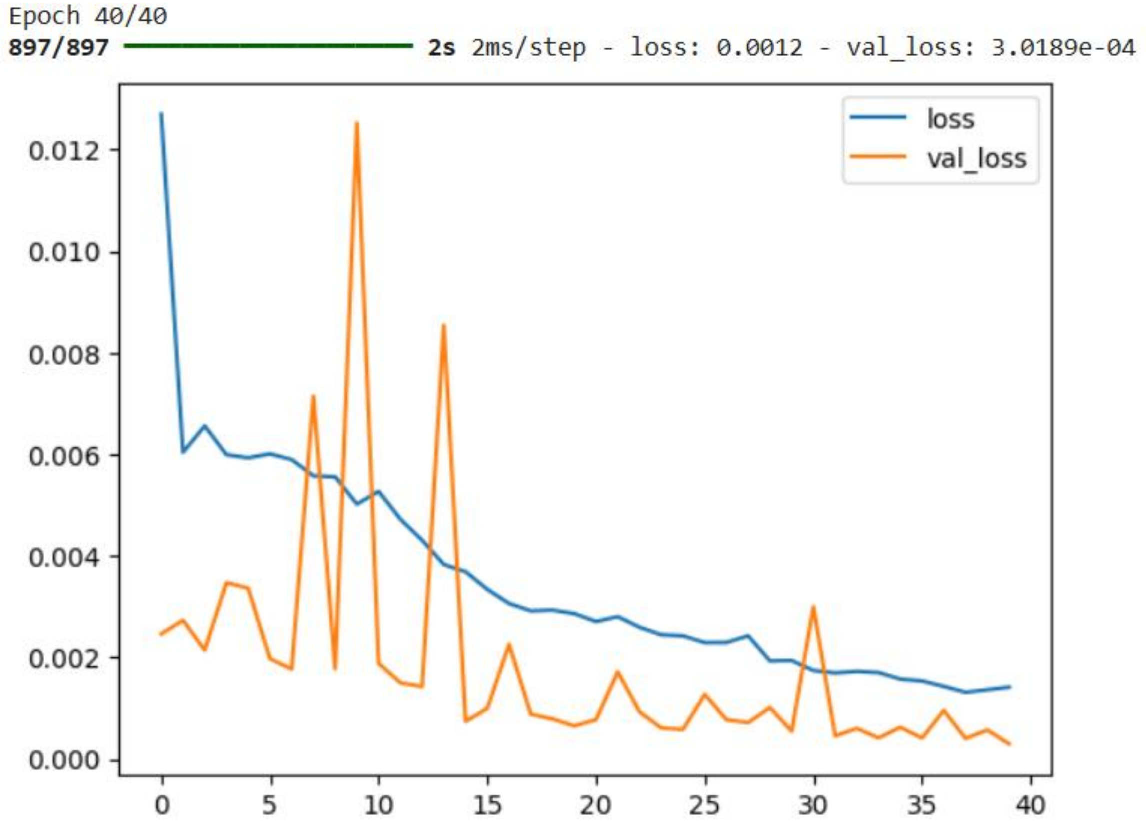
figure1c: power spectral density



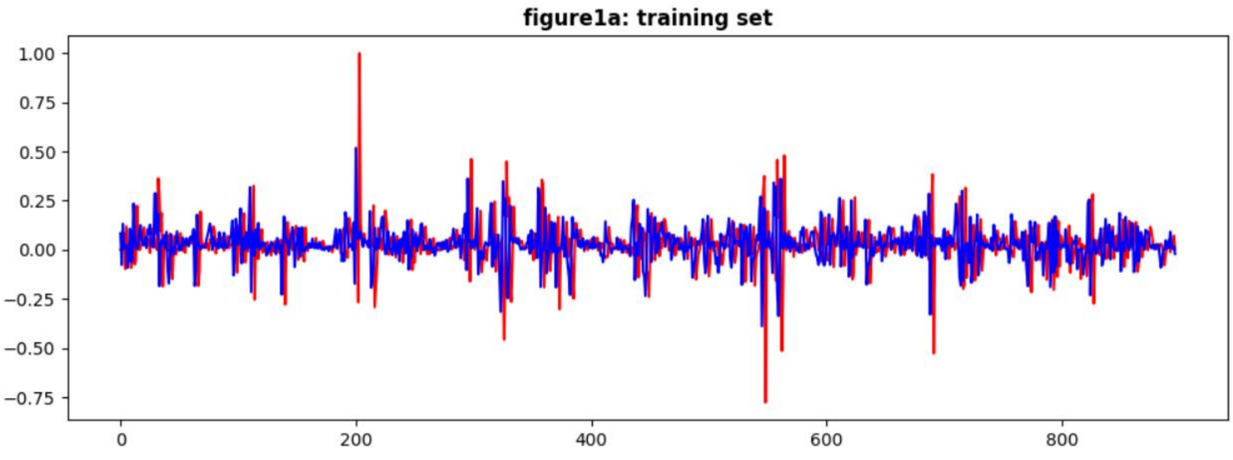
Order = 3	Best Val loss	Best Loss	Epoch #
20,20	7.0094e-04	0.0011	30
20,40	4.28573-04	0.0014	30
40, 20	0.0028	0.0049	40
40, 40	**3.0189e-04**	**0.0012**	**40**

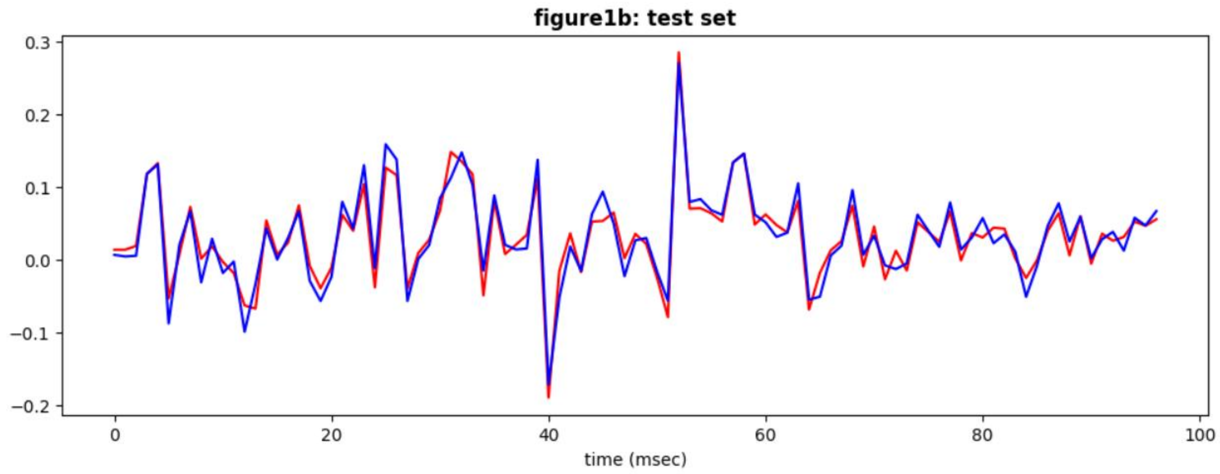
Order 3: 40, 40

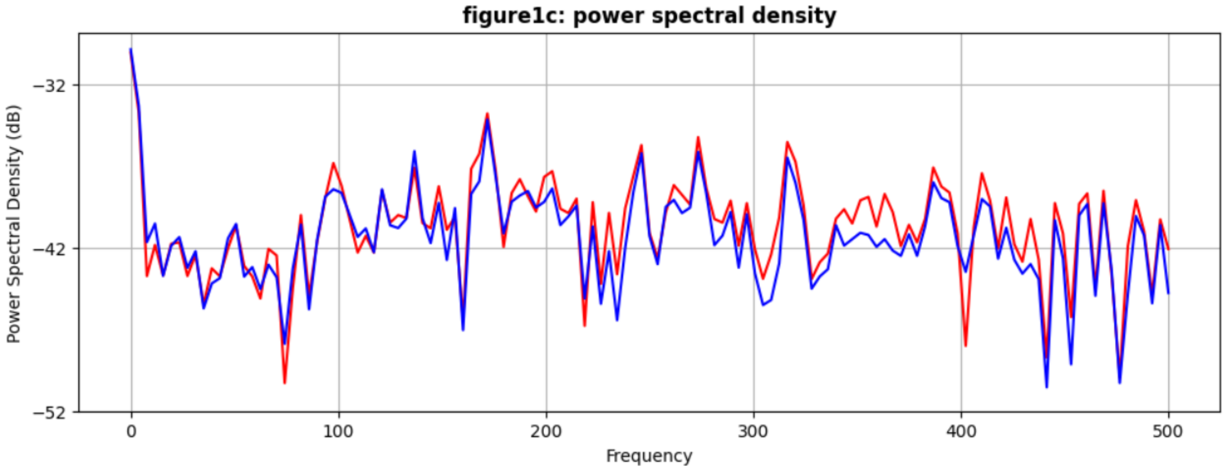
It took many runs of the model with these conditions to get an iteration which did not get stopped early on.



```
Epoch 40/40
897/897 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.0012 - val_loss: 3.0189e-04
```



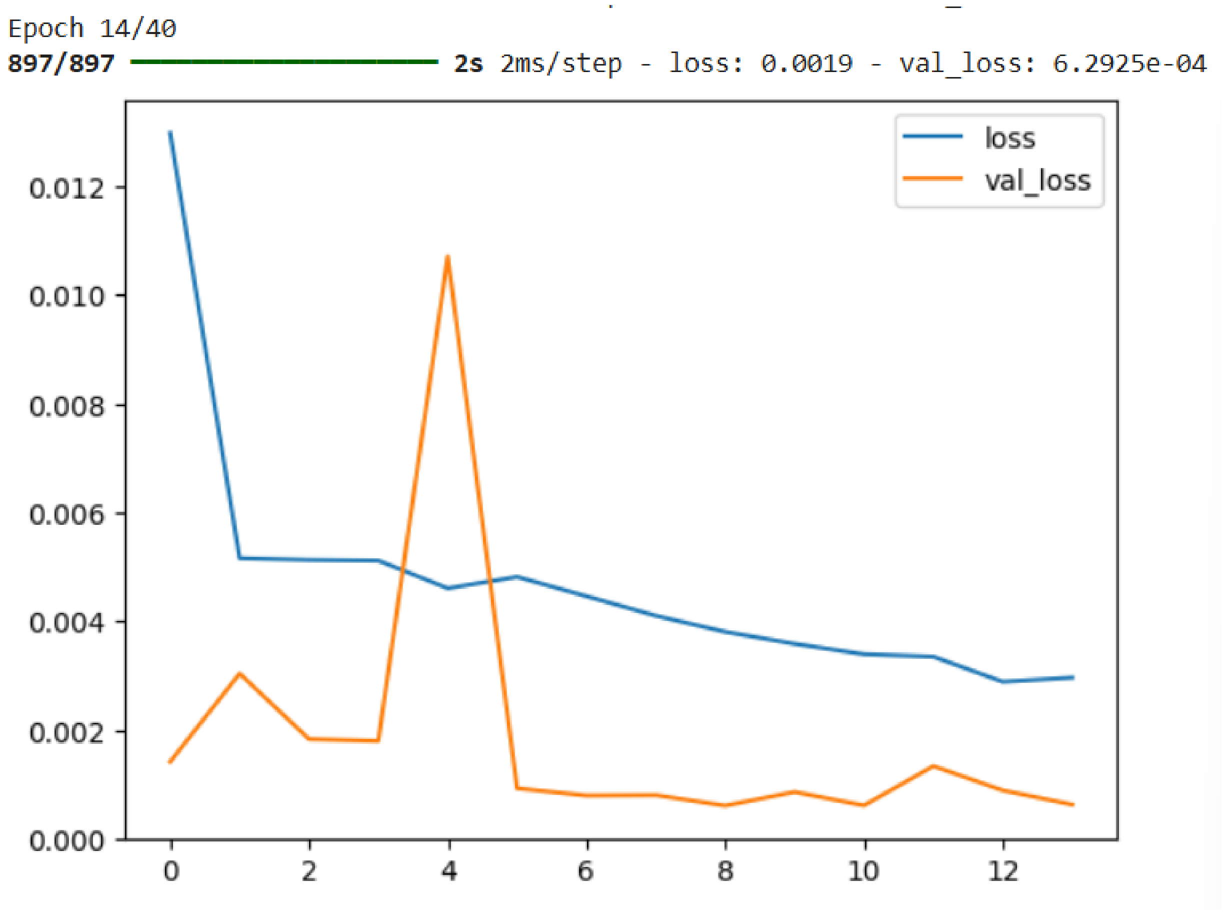




Order = 4	Best Val loss	Best Loss	Epoch #
20,20	**6.2925e-04**	**0.0019**	**40**
20,40	6.7304e-04	0.0025	40
40, 20	5.3026e-04	0.0023	40
40, 40	0.0015	0.0032	80

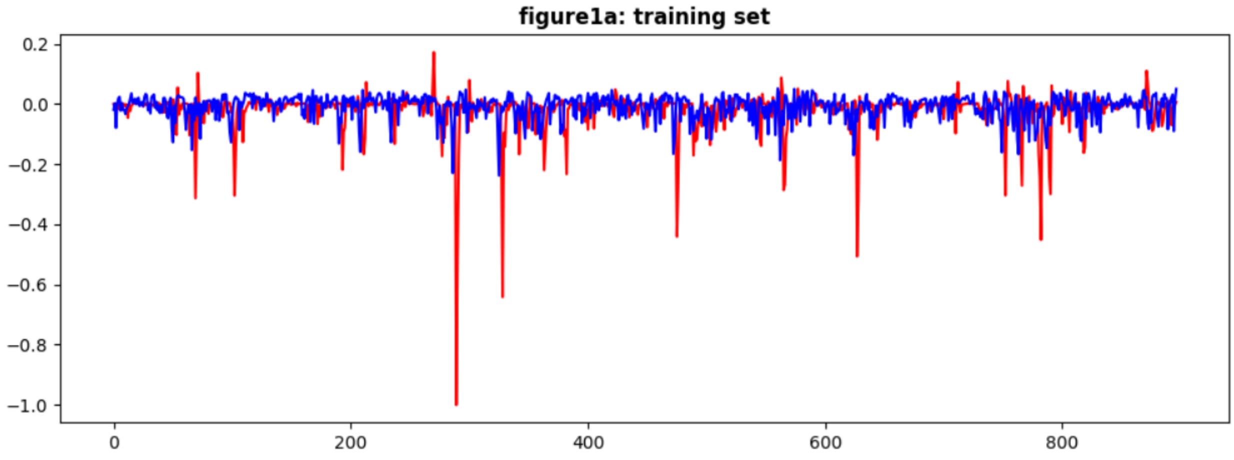
Order 4: 20, 20

After over 10 different runs of the model, it never made it past 15 epochs due to it bouncing around too much.



```
Epoch 14/40
897/897 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.0019 - val_loss: 6.2925e-04
```

Order 4: 20, 20 figure 1abc plots:



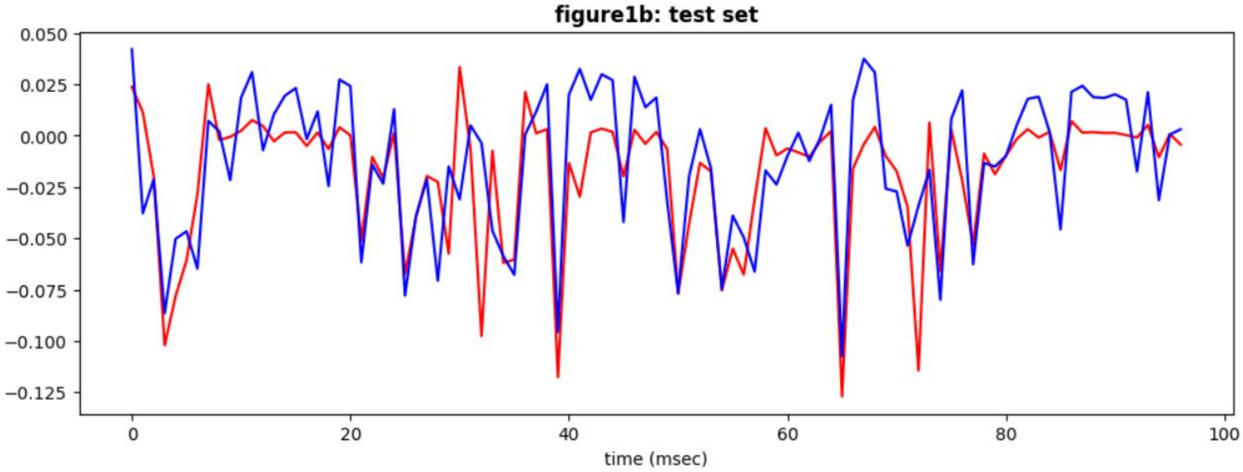


figure1b: test set

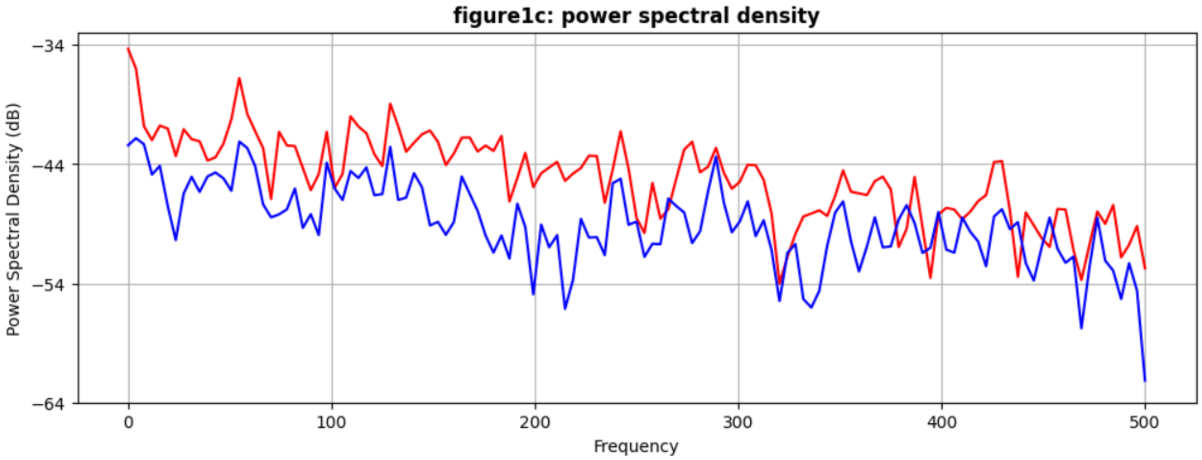


figure1c: power spectral density