ANN-Meeting

5/25/2021

Zulkaida

Propose Method & Procedure

Step	Procedure
1	Established local-fit NN code with reasonable results
2	Established global-fit NN code with reasonable results
3	Established comparison with benchmark (for example least square method) & Truth values
4	Play around to win over benchmark
5	Established the generic-hyperparameter search tools
6	Established the good performance of NN over truth evaluation and validation/testing set
7	Finishing: Optimize the job-submission code

The modification of the new functions?

1. Overall interference function
2. The jacobian
3. A factor of 2 in BHUU
4. Minus sign in tmin
5. Transverse vector-product definiton

```
// Defurne's Jacobian
jcob = 2. * PI ;
```

```
// Convert Unpolarized Coefficients to nano-barn and use Defurne's Jacobian
con_AUUBH = AUUBH * GeV2nb * jcob;
con_BUUBH = BUUBH * GeV2nb * jcob;
```

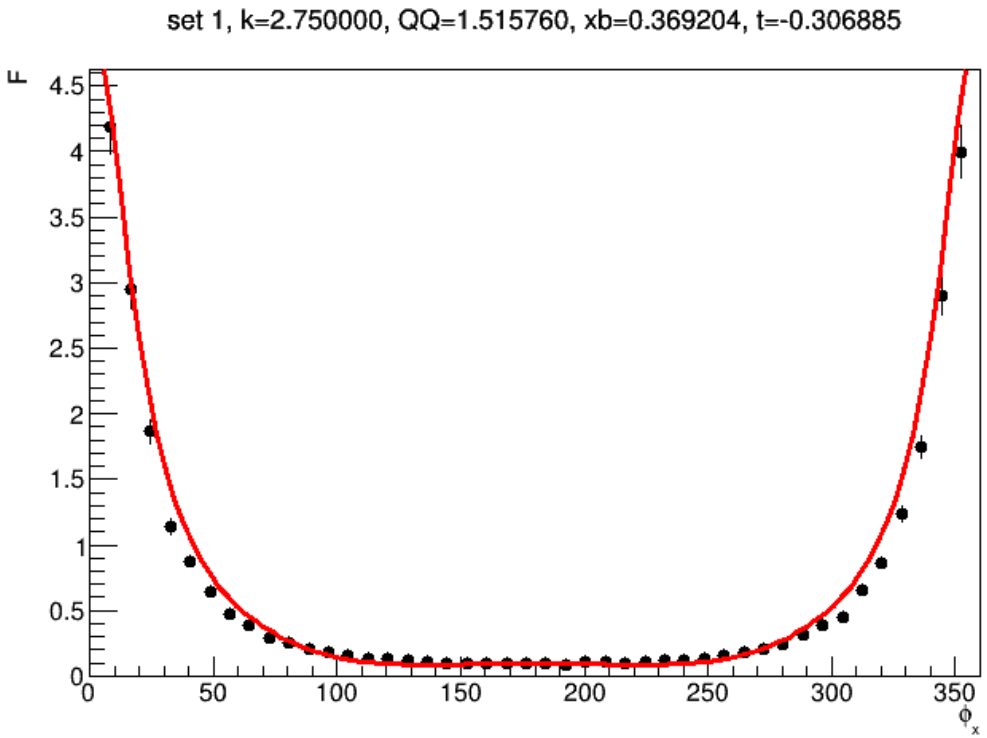
```
//Minimum t value
tmin = -( QQ * ( 1. - sqrt( 1. + gg ) + gg / 2. ) ) / ( x * ( 1. - sqrt( 1. + gg ) + gg / ( 2.* x ) ) );
```

```
kk_T    = TProduct(K,K);
kkp_T   = kk_T;
kqp_T   = TProduct(K,QP);
kd_T    = -1.* kqp_T;
dd_T    = TProduct(D,D);
kpqp_T  = TProduct(KP,QP);
kP_T    = TProduct(K,P);
kpP_T   = TProduct(KP,P);
qpP_T   = TProduct(QP,P);
kpd_T   = TProduct(KP,D);
qpd_T   = TProduct(QP,D);
```

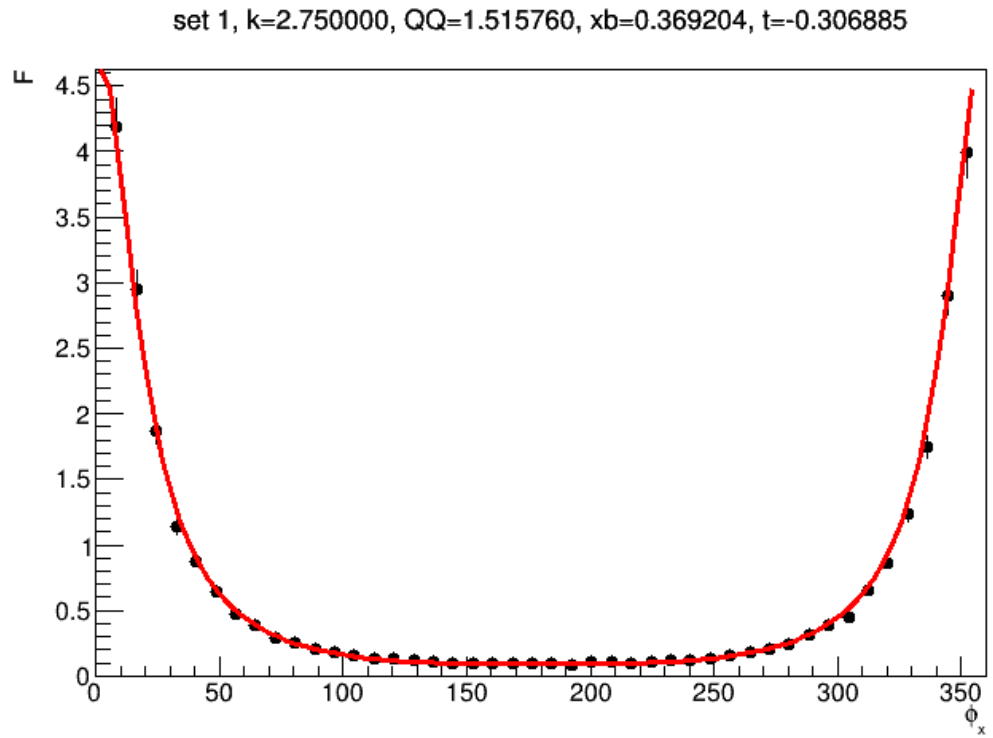
```
// Transverse vector products
kk_T    = TProduct(K,K);
kkp_T   = kk_T;
kqp_T   = TProduct(K,QP);
kd_T    = -1.* kqp_T;
dd_T    = TProduct(D,D);
kpqp_T  = kqp_T;
kP_T    = TProduct(K,P);
kpP_T   = TProduct(KP,P);
qpP_T   = TProduct(QP,P);
kpd_T   = -1.* kqp_T;
qpd_T   = -1. * dd_T;
```

First trial on 2 architectures

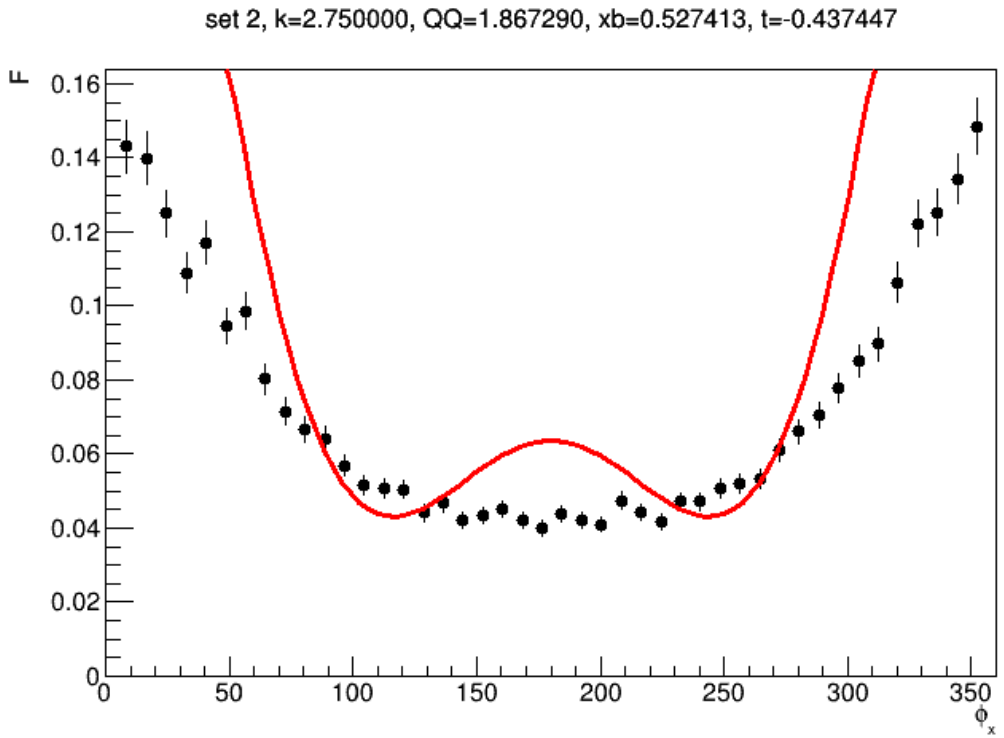
Architecture	1st	2nd
Hidden Layers	3	1
Neurons	100,100,80	512
DropOut	No	No
Epoch	4000	4000
Replica	1000	1000
Loss function	MSE	MSE
Optimized	Adam	Adam
Learning-rate	0.02	0.02
Data Normalization	No	No
Activation function	Tanh/ReLU	ReLU



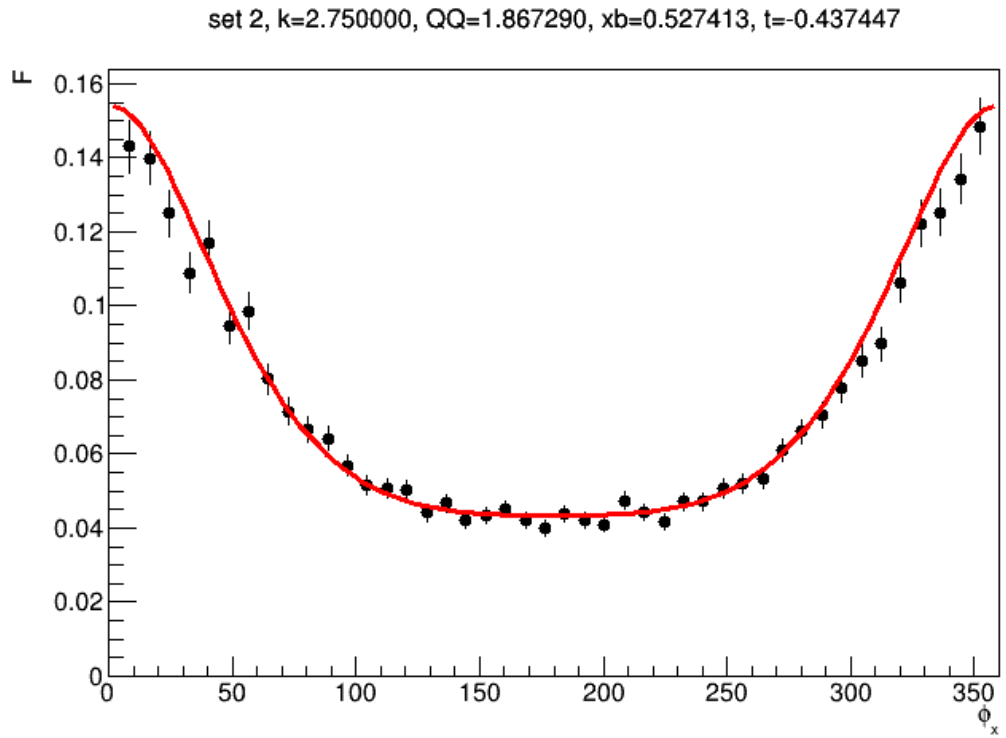
1st architecture



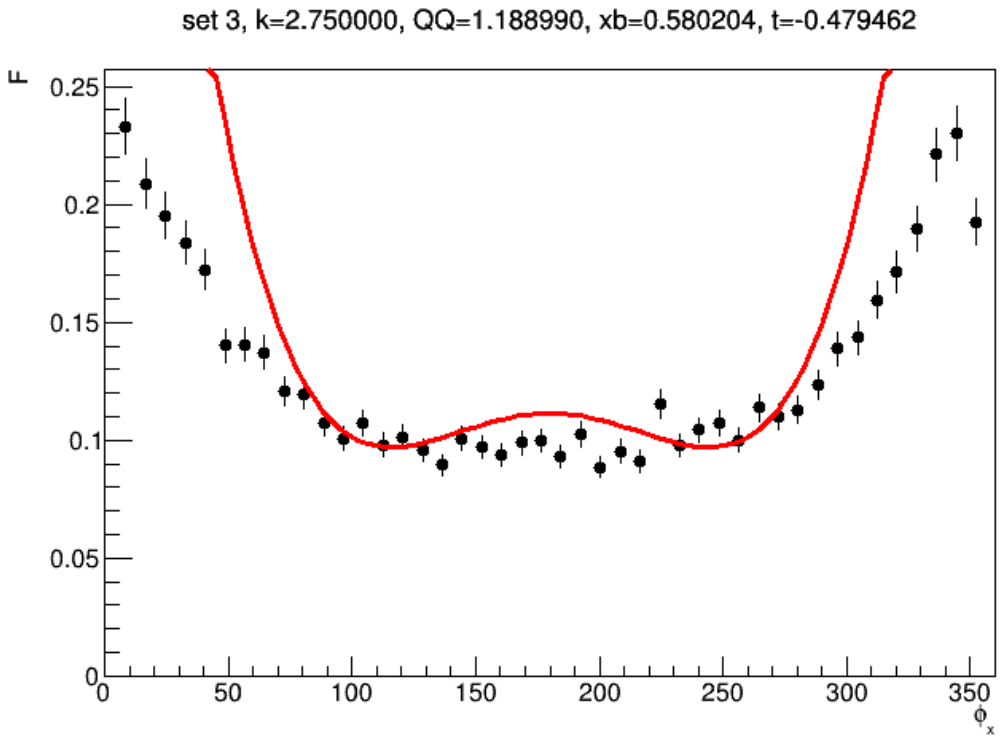
2nd architecture



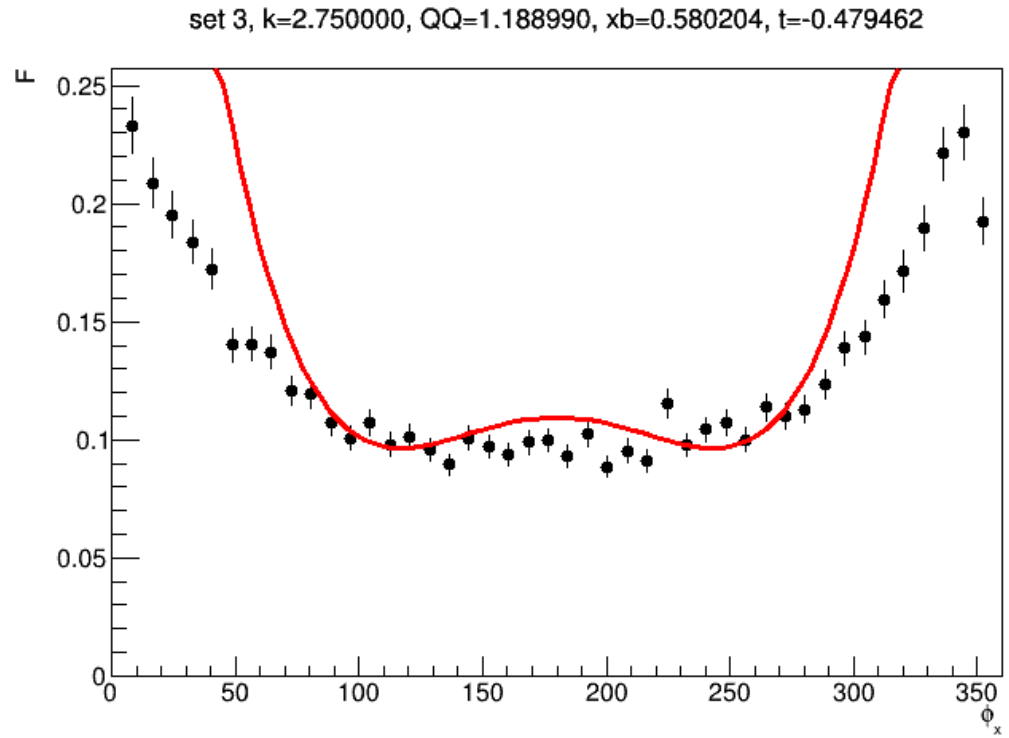
1st architecture



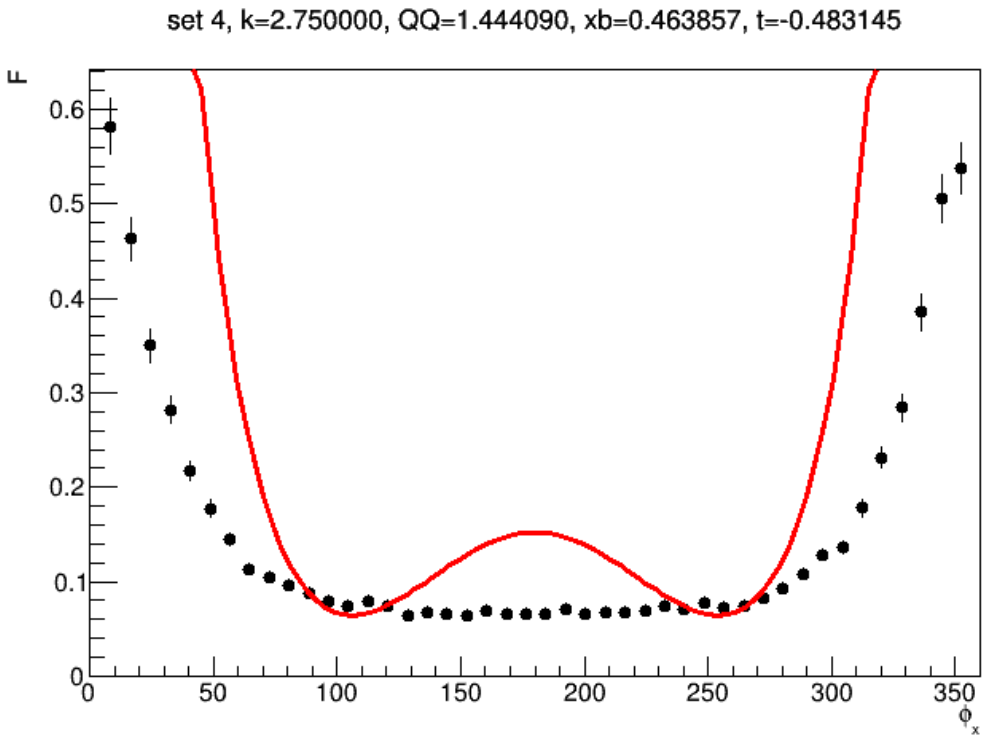
2nd architecture



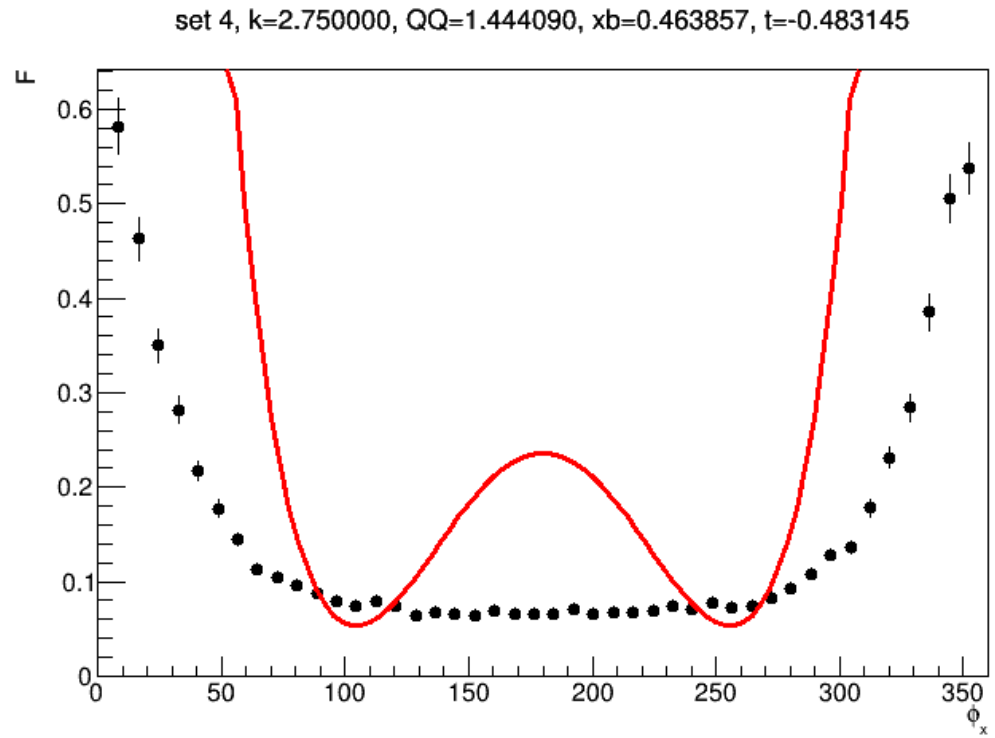
1st architecture



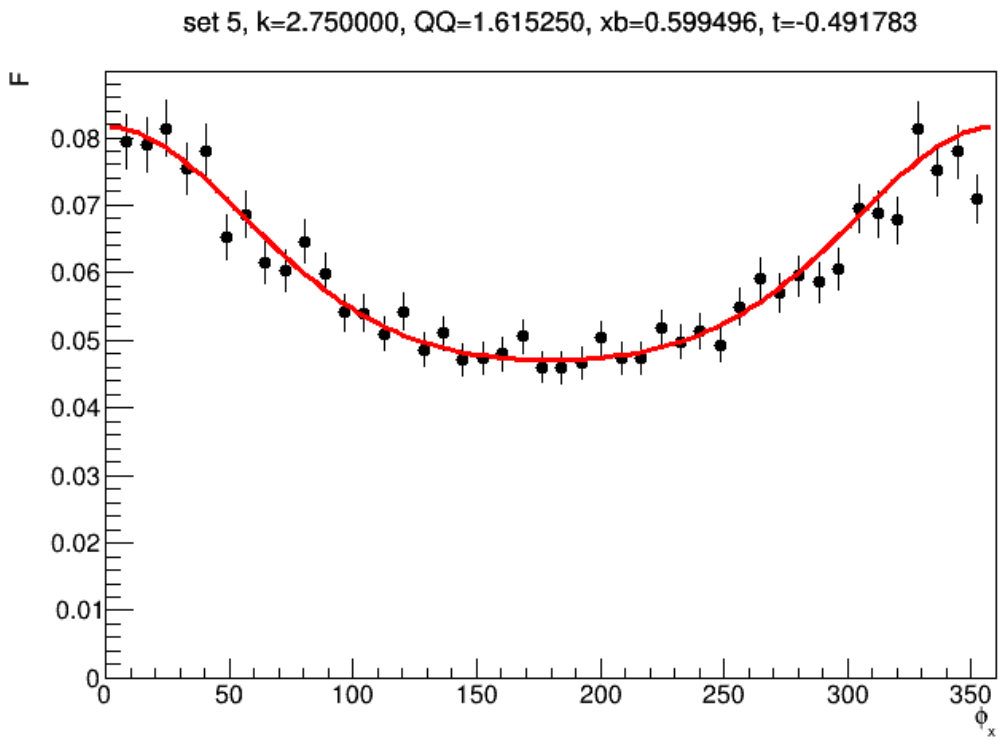
2nd architecture



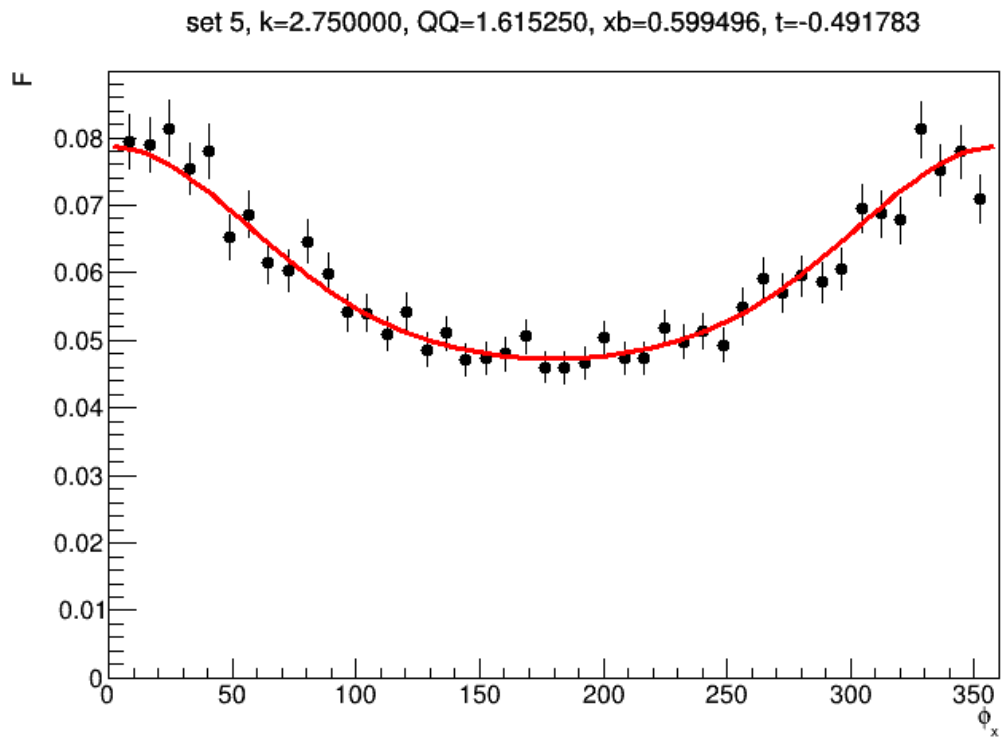
1st architecture



2nd architecture



1st architecture



2nd architecture

Conclusion

- 1st architecture perform reasonable on set 1 and 5
- 2nd architecture (less layer but more neurons) perform reasonable on set 1,2 and 5
- Both architecture overfitting on set 3 & 4
- Reducing the number of neurons in 2nd architecture from 512 to 256 make it worse
- See github to run the code