NMSU Update

Dinupa

March 23, 2023

- We use Generative adversarial network (GAN) to extract the λ, μ, ν values from the reconstructed $\phi - cos\theta$ histograms. Following modifications are made to the GAN network.
- Generator is trained with regression step. Therefore criterion is MSELoss. Generator will try to extract the λ, μ, ν values from the input histogram.

```
Generator(
(conv): Sequential(
  (0): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1))
  (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (2): ReLU()
  (3): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): ReLU()
)
(fc): Sequential(
  (0): Linear(in_features=256, out_features=64, bias=True)
  (1): ReLU()
  (2): Linear(in_features=64, out_features=32, bias=True)
  (3): ReLU()
  (4): Linear(in_features=32, out_features=3, bias=True)
)
```

- Discriminator will try to classify whether the output from the Generator is real or fake. Therefore criterion is `BCELoss`.

```
Discriminator(
  (model): Sequential(
    (0): Linear(in_features=3, out_features=20, bias=True)
    (1): ReLU()
    (2): Linear(in_features=20, out_features=1, bias=True)
    (3): Sigmoid()
  )
)
```

- We can understand the behavior of GAN as a con artist trying to fake a painting (generator) while a detective trying to identify if the painting is fake or real (discriminator). Artist will succeed his work when he was able to fool the detective. This is what we try to achieve by training GAN.
- Our goal is to extract the angular coefficients from the new histograms using a trained GAN.

» Training histograms;

-> 1000 combinations of lambda, mu, nu were created in range [-1., 1.]
-> histograms were filled with 100k events.
-> costheta range [-0.6, 0.6]

» Test histograms;

-> 30 histograms were created with 20K different events.
-> 4 different combination of lambda, mu, nu is tested;

	lambda	mu	nu
test 1	0.3	-0.1	0.1
test 2	-0.1	0.3	0.5
test 3	0.5	0.1	-0.3
test 4	-0.01	-0.01	-0.01

-> These combinations of lambda, mu, nu is not used in training.

$$[\lambda, \mu, \nu] = [0.3, -0.1, 0.1]$$

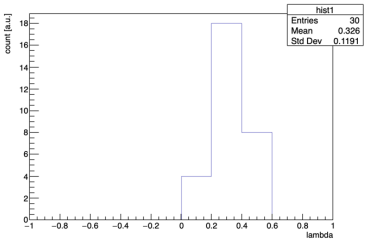

Figure 1: Extracted lambda.

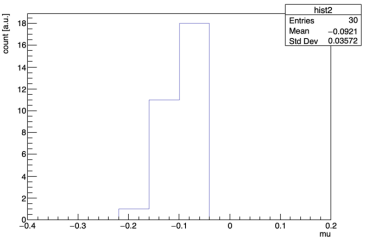

Figure 2: Extracted mu.

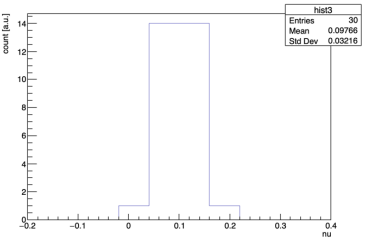

Figure 3: Extracted nu.

$$[\lambda, \mu, \nu] = [-0.1, 0.3, 0.5]$$

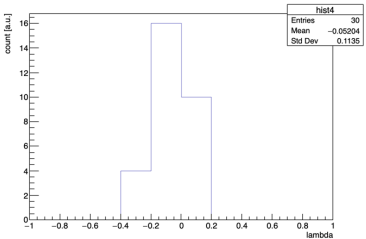

Figure 4: Extracted lambda.

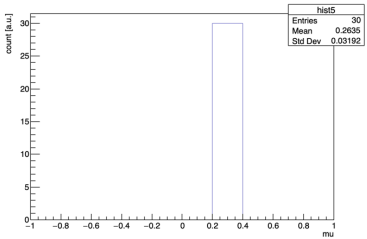

Figure 5: Extracted mu.

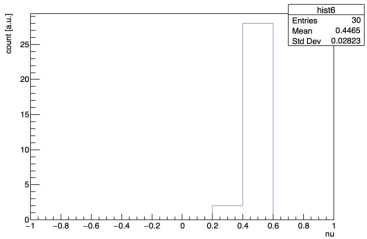

Figure 6: Extracted nu.

$$[\lambda, \mu, \nu] = [0.5, 0.1, -0.3]$$

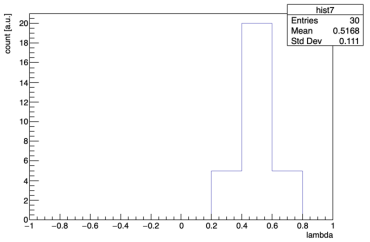

Figure 7: Extracted lambda.

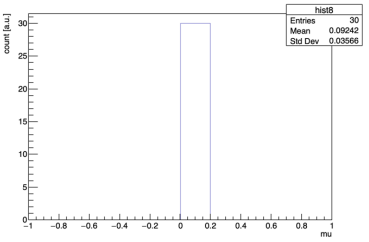

Figure 8: Extracted mu.

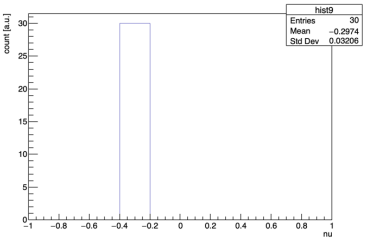

Figure 9: Extracted nu.

$$[\lambda, \mu, \nu] = [-0.01, -0.01, -0.01]$$

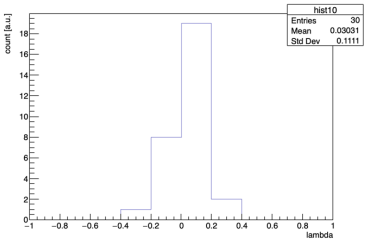

Figure 10: Extracted lambda.

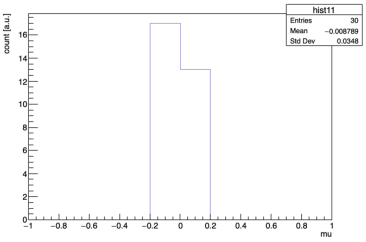

Figure 11: Extracted mu.

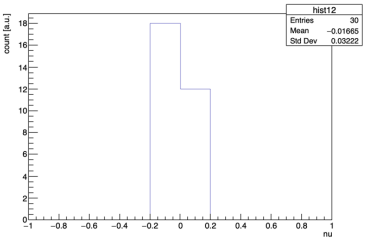

Figure 12: Extracted nu.

* [m, pT, x1, x2, xF] -> phi vs. costh -> lambda, mu, nu
* in reality phi and costh distribution -> lambda, nu, mu -> xsec
* if we have a method to get lambda, mu, nu -> profile histogram [m, pT, x1, x2, xF]
* this method (or any unbinned method) can be used with statistics
* in generator level can we generate lambda, mu, nu randomly ?
* since we have 1 + costh^2 distribution, is it possible to generate phi, costh for different lambda, mu, nu without weights ?