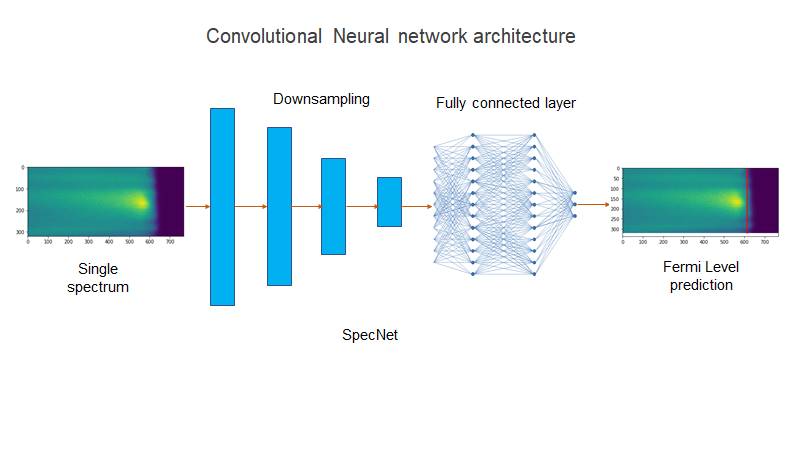
Preprocessing of dataset:-

1. The dataset contains of 40 data files. These files contain around 1500 spectras which were extracted from the dataset.
2. All the spectras were cropped randomly to remove upto 10% of the channels at the starting and ending of the spectras so that the neural network doesn’t learn to predict some constant values.
3. The spectras were resized to a size of (320,768) using BiCubic interpolation to be feeded into a Convolutional neural network(CNN). It was further scaled and normalized to a range of -1 to 1.
4. The spectrums were finally split in 70:20:10 ratio for training, validation and testing respectively.

Architecture of CNN(It is similar to a Unet ([https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47#:~:text=UNET%20Architecture%20and%20Training,for%20Bio%20Medical%20Image%20Segmentation.&text=Thus%20it%20is%20an%20end,accept%20image%20of%20any%20size](https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47#:~:text=UNET%20Architecture%20and%20Training,for%20Bio%20Medical%20Image%20Segmentation.&text=Thus%20it%20is%20an%20end,accept%20image%20of%20any%20size.)) encoding followed by a fully connected layer).



The full architecture of the neural network is:-

Spec\_unet(

(conv\_downsample): Sequential(

(0): Conv2d(1, 8, kernel\_size=(7, 7), stride=(1, 1), padding=(3, 3), bias=False)

(1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(2): LeakyReLU(negative\_slope=0.1)

(3): Conv2d(8, 16, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(4): Dropout(p=0.2)

(5): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(6): LeakyReLU(negative\_slope=0.1)

(7): Conv2d(16, 32, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(8): Dropout(p=0.2)

(9): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(10): LeakyReLU(negative\_slope=0.1)

(11): Conv2d(32, 64, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(12): Dropout(p=0.2)

(13): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(14): LeakyReLU(negative\_slope=0.1)

(15): Conv2d(64, 64, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(16): Dropout(p=0.2)

(17): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(18): LeakyReLU(negative\_slope=0.1)

(19): Conv2d(64, 64, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(20): Dropout(p=0.2)

(21): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(22): LeakyReLU(negative\_slope=0.1)

(23): Conv2d(64, 64, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)

(24): Dropout(p=0.2)

(25): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(26): LeakyReLU(negative\_slope=0.1)

(27): Conv2d(64, 8, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)

(28): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(29): LeakyReLU(negative\_slope=0.1)

)

(linear): Sequential(

(0): Linear(in\_features=480, out\_features=32, bias=True)

(1): Dropout(p=0.2)

(2): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

(3): LeakyReLU(negative\_slope=0.1)

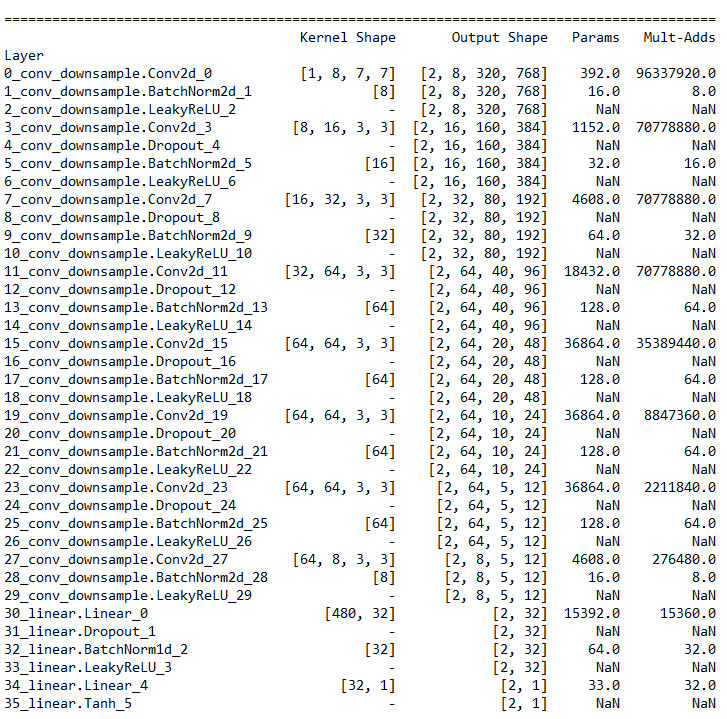
(4): Linear(in\_features=32, out\_features=1, bias=True)

(5): Tanh()

)

)

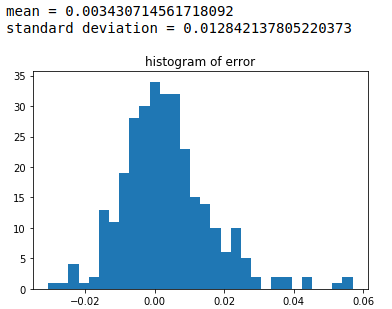
To get a more clear picture of the working, you can also see a forward pass of the network. The input is a spectrum of size (2, 1, 320, 768) and output is a single number which corresponds to the fermi\_level in the spectrum of shape (2,1). You can ignore the starting index 2 as it corresponds to the batch size.



The network was finally trained with a L1 norm between the predicted and the actual calculated value.

Results:-

The histogram shows the difference of actual energy and predicted energy in the x axis. As you can see in the histogram, the mean of errors is around 0.003eV with a standard deviation of  0.012eV.



Assuming the data follows the same error distribution(gaussian), you can assume the predicted value to have an error of:- ()milleV with 68% confidence, ()milleV with 95% confidence, and so on, where N denotes the number of spectrums in your data file.