Calculation of Parameters and Details Discussion of Neural Network Model

Md. Istiak Ahammed

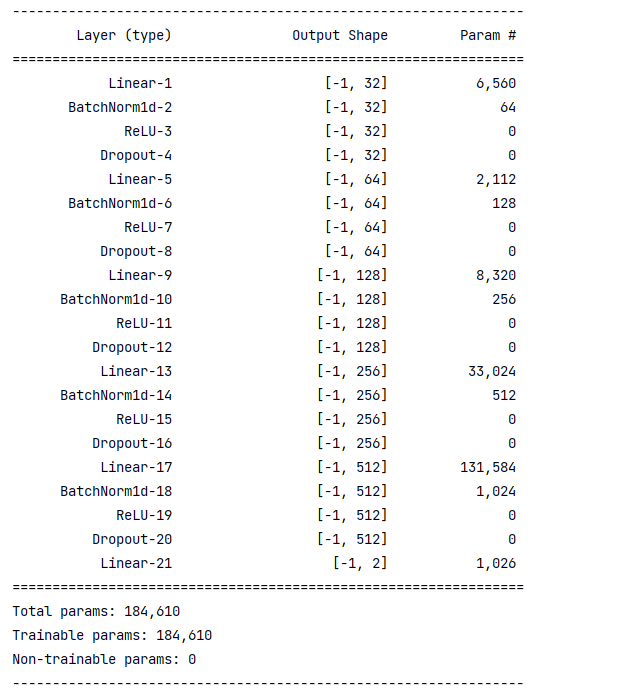
Example: Feed Forward Neural Network Architecture including

1. Fully connected layer/Dense layer/Linear layer
2. 1D Batch Normalization layer
3. ReLU activation function layer
4. Dropout layer

**Model’s Code:** Dimensionality increasing custom model

class CustomFFNv5(nn.Module):  
 def \_\_init\_\_(self, input\_dim, hidden\_dim, output\_dim, l2\_lambda):  
 super(CustomFFNv5, self).\_\_init\_\_()  
  
 self.fc1 = nn.Linear(input\_dim, hidden\_dim) *# 32* self.bn1 = nn.BatchNorm1d(hidden\_dim) *# Apply Batch Normalization* self.relu1 = nn.ReLU()  
 self.dropout1 = nn.Dropout(0.2)  
  
 self.fc2 = nn.Linear(hidden\_dim, 2 \* hidden\_dim) *# (32, 64)* self.bn2 = nn.BatchNorm1d(2 \* hidden\_dim) *# Apply Batch Normalization # 64* self.relu2 = nn.ReLU()  
 self.dropout2 = nn.Dropout(0.2)  
  
 self.fc3 = nn.Linear(2 \* hidden\_dim, 4 \* hidden\_dim) *# (64, 128)* self.bn3 = nn.BatchNorm1d(4 \* hidden\_dim) *# Apply Batch Normalization* self.relu3 = nn.ReLU()  
 self.dropout3 = nn.Dropout(0.2)  
  
 self.fc4 = nn.Linear(4 \* hidden\_dim, 8 \* hidden\_dim) *# (128, 256)* self.bn4 = nn.BatchNorm1d(8 \* hidden\_dim) *# Apply Batch Normalization* self.relu4 = nn.ReLU()  
 self.dropout4 = nn.Dropout(0.2)  
  
 self.fc5 = nn.Linear(8 \* hidden\_dim, 16 \* hidden\_dim) *# (256, 512)* self.bn5 = nn.BatchNorm1d(16 \* hidden\_dim) *# Apply Batch Normalization* self.relu5 = nn.ReLU()  
 self.dropout5 = nn.Dropout(0.2)  
  
 self.fc6 = nn.Linear(16 \* hidden\_dim, output\_dim) *# (512, 2=Two Class)* self.l2\_lambda = l2\_lambda  
  
 def forward(self, x):  
 x = self.fc1(x)  
 x = self.bn1(x) *# Apply Batch Normalization* x = self.relu1(x)  
 x = self.dropout1(x)  
  
 x = self.fc2(x)  
 x = self.bn2(x) *# Apply Batch Normalization* x = self.relu2(x)  
 x = self.dropout2(x)  
  
 x = self.fc3(x)  
 x = self.bn3(x) *# Apply Batch Normalization* x = self.relu3(x)  
 x = self.dropout3(x)  
  
 x = self.fc4(x)  
 x = self.bn4(x) *# Apply Batch Normalization* x = self.relu4(x)  
 x = self.dropout4(x)  
  
 x = self.fc5(x)  
 x = self.bn5(x) *# Apply Batch Normalization* x = self.relu5(x)  
 x = self.dropout5(x)  
  
 x = self.fc6(x)  
 return x  
  
 def l2\_regularization\_loss(self):  
 l2\_loss = 0.0  
 for param in self.parameters():  
 l2\_loss += torch.sum(torch.square(param))  
  
 return self.l2\_lambda \* l2\_loss

Model’s Summary:



Explanation of Parameters calculation in each layer

**Linear-1:** This is a fully connected layer with an input dimension of input\_dim (204) and an output dimension of hidden\_dim (32). The number of parameters in this layer is calculated as (input\_dim + 1) \* hidden\_dim = (204 + 1) \* 32 = 6,560.

**BatchNorm1d-1:** This is a batch normalization layer for 1-dimensional input with hidden\_dim (32) features. It has 2 \* hidden\_dim parameters, consisting of mean and variance for each feature. So the number of parameters in this layer is 2 \* hidden\_dim = 2 \* 32 = 64.

**Batch Normalization** is typically applied after the linear transformation in a neural network layer and before the non-linear activation function. This helps to normalize the inputs to the activation function, stabilizing the training process and improving the network's ability to learn.

**ReLU-1:** This is the activation function layer and doesn't have any parameters.

**Dropout-1:** This is the dropout layer with a dropout rate of 0.2. It doesn't have any parameters.

**Linear-2:** This is a fully connected layer with an input dimension of hidden\_dim (32) and an output dimension of 2 \* hidden\_dim (64). The number of parameters in this layer is (hidden\_dim + 1) \* (2 \* hidden\_dim) = (32 + 1) \* 64 = 2,112.

**BatchNorm1d-2:** Batch normalization layer for 1-dimensional input with 2 \* hidden\_dim (64) features. It has 2 \* (2 \* hidden\_dim) = 2 \* 64 = 128 parameters.

**ReLU-2:** Activation function layer.

**Dropout-2:** Dropout layer with a dropout rate of 0.2.

**Linear-3:** Fully connected layer with an input dimension of 2 \* hidden\_dim (64) and an output dimension of 4 \* hidden\_dim (128). The number of parameters in this layer is (2 \* hidden\_dim + 1) \* (4 \* hidden\_dim) = (2 \* 32 + 1) \* 128 = 8,320.

**BatchNorm1d-3:** Batch normalization layer for 1-dimensional input with 4 \* hidden\_dim (128) features. It has 2 \* (4 \* hidden\_dim) = 2 \* 128 = 256 parameters.

**ReLU-3:** Activation function layer.

**Dropout-3:** Dropout layer with a dropout rate of 0.2.

**Linear-4:** Fully connected layer with an input dimension of 4 \* hidden\_dim (128) and an output dimension of 8 \* hidden\_dim (256). The number of parameters in this layer is (4 \* hidden\_dim + 1) \* (8 \* hidden\_dim) = (4 \* 32 + 1) \* 256 = 33,024.

**BatchNorm1d-4:** Batch normalization layer for 1-dimensional input with 8 \* hidden\_dim (256) features. It has 2 \* (8 \* hidden\_dim) = 2 \* 256 = 512 parameters.

**ReLU-4:** Activation function layer.

**Dropout-4:** Dropout layer with a dropout rate of 0.2.

**Linear-5:** Fully connected layer with an input dimension of 8 \* hidden\_dim (256) and an output dimension of 16 \* hidden\_dim (512). The number of parameters in this layer is (8 \* hidden\_dim + 1) \* (16 \* hidden\_dim) = (8 \* 32 + 1) \* 512 = 131,584.

**BatchNorm1d-5:** Batch normalization layer for 1-dimensional input with 16 \* hidden\_dim (512) features. It has 2 \* (16 \* hidden\_dim) = 2 \* 512 = 1,024 parameters.

**ReLU-5:** Activation function layer.

**Dropout-5:** Dropout layer with a dropout rate of 0.2.

**Linear-6:** Fully connected layer with an input dimension of 16 \* hidden\_dim (512) and an output dimension of output\_dim (2). The number of parameters in this layer is (16 \* hidden\_dim + 1) \* output\_dim = (16 \* 32 + 1) \* 2 = 1,026.

To calculate the total number of parameters, you sum up the parameters from all the layers. In this case, you sum up the parameters from all the Linear layers.

Let's calculate the total parameters for your model using input\_dim = 204, hidden\_dim = 32, and output\_dim = 2

**Total parameters =** 6,560 + 64 + 2,112 + 128 + 8,320 + 256 + 33,024 + 512 + 131,584 + 1,024 + 1,026

= 184,610