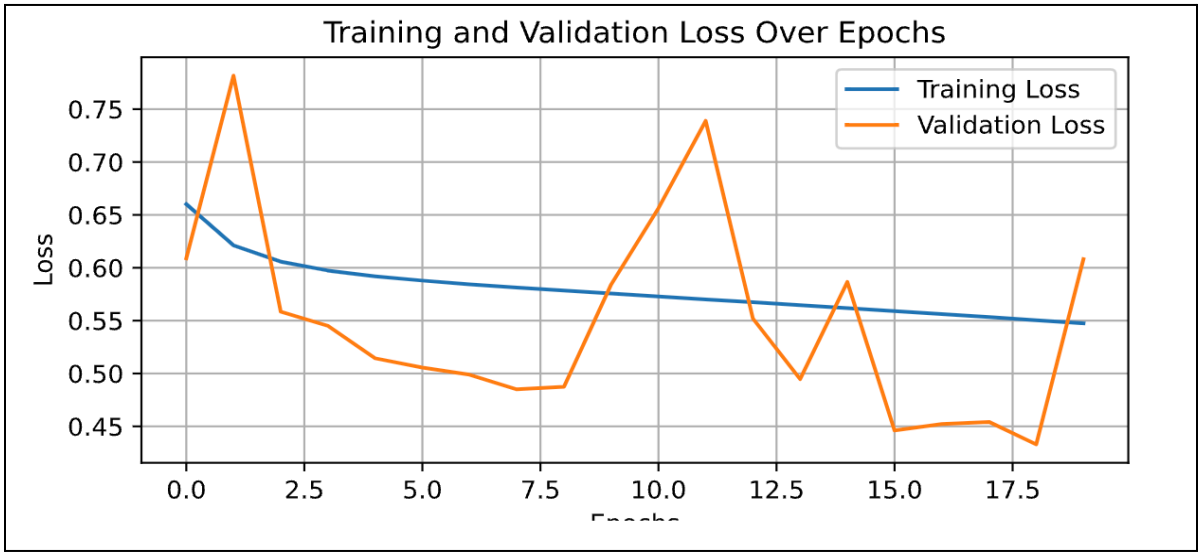
I. Modify Model Architecture To Increase Performance

1. **Initial Neural Network** (Class name: Simplemodel)
   Model Architecture:

```python
class Simplemodel(nn.Module):

    def __init__(self, num_bits_per_symbol, H=64, W=512):
        super(Simplemodel, self).__init__()
        scale = 8
        self.linear1=nn.Linear(in_features=2, out_features=scale*num_bits_per_symbol)
        self.linear2=nn.Linear(in_features=scale*num_bits_per_symbol, out_features=num_bits_per_symbol)
        self.activation = nn.ReLU()

    def forward(self, inputs):
        y = inputs #[64, 64, 512]

        # Stack the tensors along a new dimension (axis 0)
        z = torch.stack([y.real, y.imag], dim=0) #[2, 64, 64, 512]
        z = z.permute(1, 2, 3, 0) #[64, 64, 512, 2]
        z = self.linear1(z)
        z = self.activation(z)
        z = self.linear2(z)
        z = nn.Sigmoid()(z) #[64, 64, 512, num_bits_per_symbol]
        z = z.flatten(-2, -1) #combine last two dimension => [64, 64, 512*num_bits_per_symbol]
        return z
```

Number of epochs: 20
Training Loss after 20 epochs: 0.5475
Validation Loss after 20 epochs: 0.6081

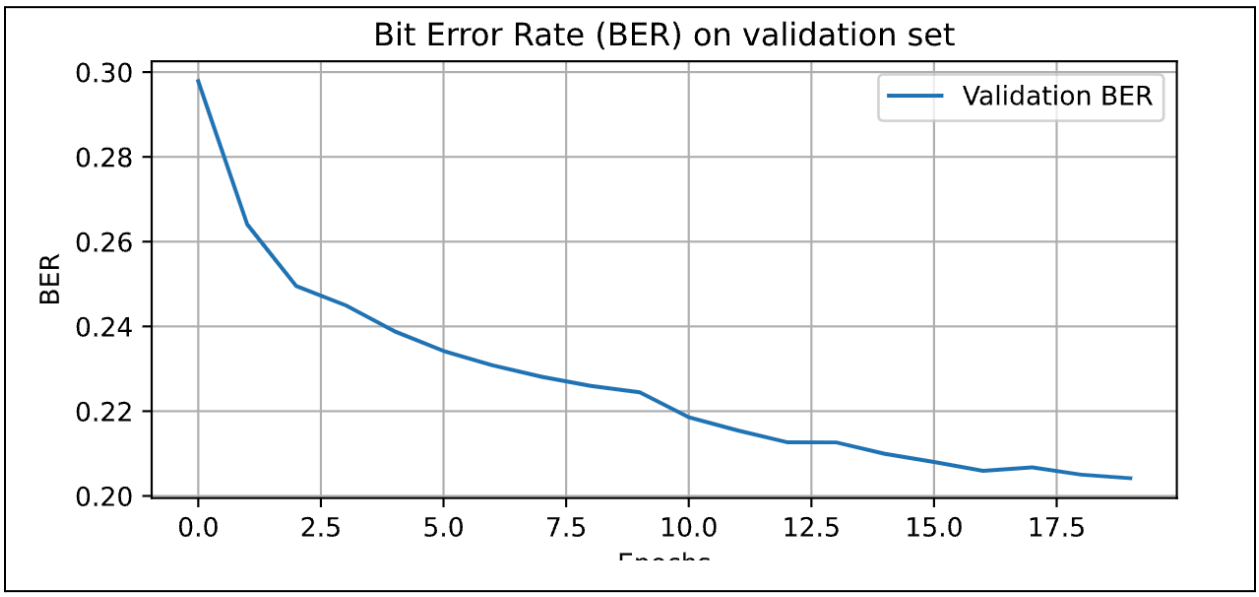
Plot of Training Loss and Validation Loss over 20 epochs:



https://github.com/sahaj-totla/Deep-Learning/blob/main/loss_graphs/simple_model_loss_graph.svg

Optimizer: torch.optim.SGD(model.parameters(), lr=0.1)
Validation BER after 20 epochs: 0.2042

Plot of BER on Validation set:



https://github.com/sahaj-totla/Deep-Learning/blob/main/BER_graphs/simple_model_BER_graph.svg

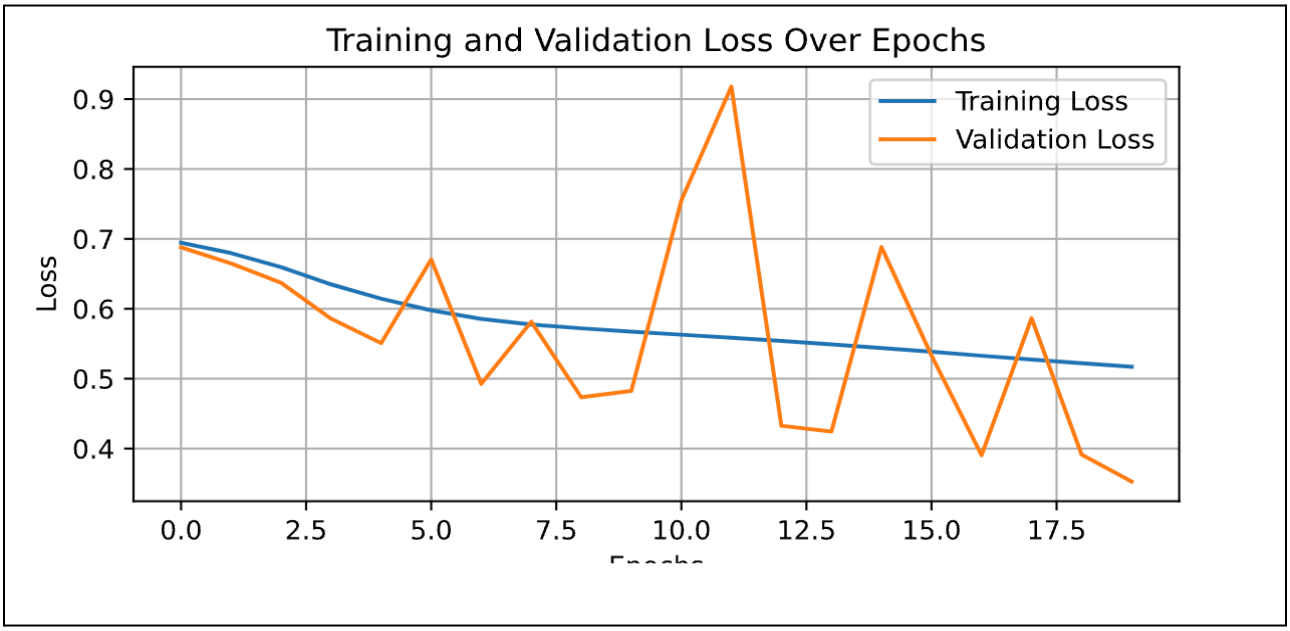
2. **Enhanced Neural Network** (Class name: EnhancedSimpleModel) - Increased the number of hidden layers

   Model Architecture:

```python
class EnhancedSimpleModel(nn.Module):

    def __init__(self, num_bits_per_symbol, H=64, W=512):
        super(EnhancedSimpleModel, self).__init__()
        scale = 8
        self.linear1 = nn.Linear(in_features=2, out_features=scale * num_bits_per_symbol)
        self.linear2 = nn.Linear(in_features=scale * num_bits_per_symbol, out_features=scale * num_bits_per_symbol // 2)
        self.linear3 = nn.Linear(in_features=scale * num_bits_per_symbol // 2, out_features=scale * num_bits_per_symbol // 4)
        self.linear4 = nn.Linear(in_features=scale * num_bits_per_symbol // 4, out_features=num_bits_per_symbol)
        self.activation = nn.ReLU()

    def forward(self, inputs):
        y = inputs
        z = torch.stack([y.real, y.imag], dim=0)
        z = z.permute(1, 2, 3, 0)
        z = self.linear1(z)
        z = self.activation(z)
        z = self.linear2(z)
        z = self.activation(z)
        z = self.linear3(z)
        z = self.activation(z)
        z = self.linear4(z)
        z = nn.Sigmoid()(z)
        z = z.flatten(-2, -1)
        return z
```

   Number of epochs: 20
   Training Loss after 20 epochs: 0.5169
   Validation Loss after 20 epochs: 0.3529

   Plot of Training Loss and Validation Loss over 20 epochs:

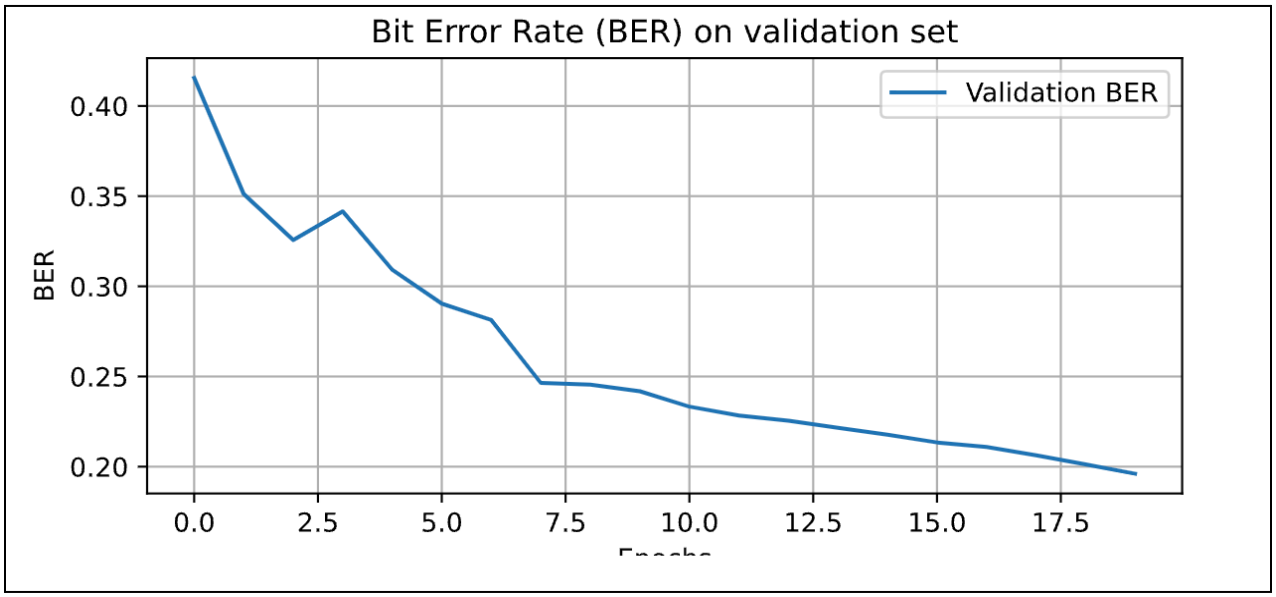


https://github.com/sahaj-totla/Deep-Learning/blob/main/loss_graphs/enhanced_model_loss_graph.svg

Optimizer: torch.optim.SGD(model.parameters(), lr=0.1)

Validation BER after 20 epochs: 0.1961

Plot of BER on Validation set:



https://github.com/sahaj-totla/Deep-Learning/blob/main/BER_graphs/enhanced_model_BER_graph.svg

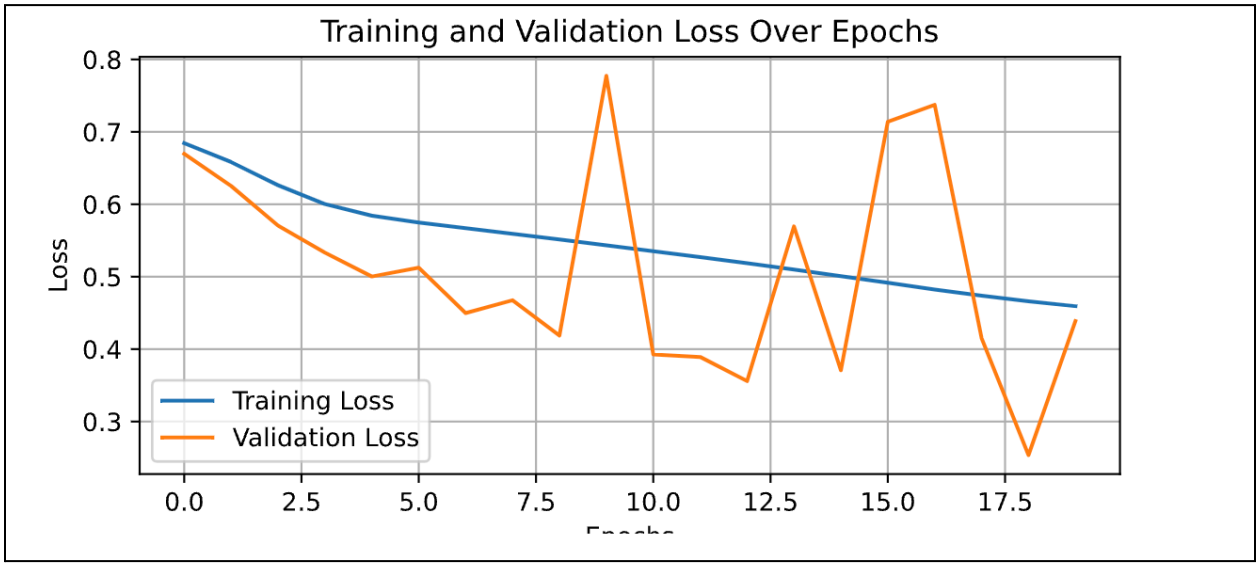
3. **Enhanced Neural Network** (class name: EnhancedSimpleModel2) - Increased the number of neurons in each hidden layer

   Model Architecture:

```
[24] class EnhancedSimpleModel2(nn.Module):

        def __init__(self, num_bits_per_symbol, H=64, W=512):
            super(EnhancedSimpleModel2, self).__init__()
            scale = 8
            self.linear1 = nn.Linear(in_features=2, out_features=scale * num_bits_per_symbol)
            self.linear2 = nn.Linear(in_features=scale * num_bits_per_symbol, out_features=scale * num_bits_per_symbol * 2)
            self.linear3 = nn.Linear(in_features=scale * num_bits_per_symbol * 2, out_features=scale * num_bits_per_symbol * 2)
            self.linear4 = nn.Linear(in_features=scale * num_bits_per_symbol * 2, out_features=num_bits_per_symbol)
            self.activation = nn.ReLU()

        def forward(self, inputs):
            y = inputs
            z = torch.stack([y.real, y.imag], dim=0)
            z = z.permute(1, 2, 3, 0)
            z = self.linear1(z)
            z = self.activation(z)
            z = self.linear2(z)
            z = self.activation(z)
            z = self.linear3(z)
            z = self.activation(z)
            z = self.linear4(z)
            z = nn.Sigmoid()(z)
            z = z.flatten(-2, -1)
            return z
        z = z.flatten(-2, -1)
        return z
```

   Number of epochs: 20
   Training Loss after 20 epochs: 0.4593
   Validation Loss after 20 epochs: 0.4387

   Plot of Training Loss and Validation Loss over 20 epochs:

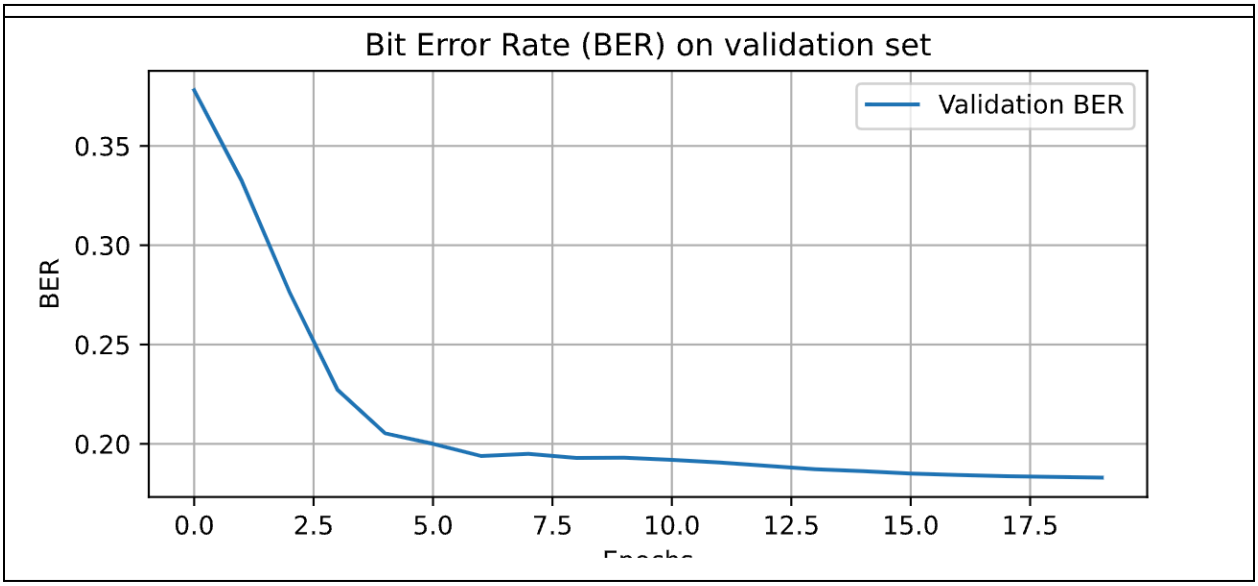


https://github.com/sahaj-totla/Deep-Learning/blob/main/loss_graphs/enhanced_model_2_loss_graph.svg

Optimizer: torch.optim.SGD(model.parameters(), lr=0.1)

Validation BER after 20 epochs: 0.1830

Plot of BER on Validation set:



https://github.com/sahaj-totla/Deep-Learning/blob/main/BER_graphs/enhanced_model_2_BER_graph.svg

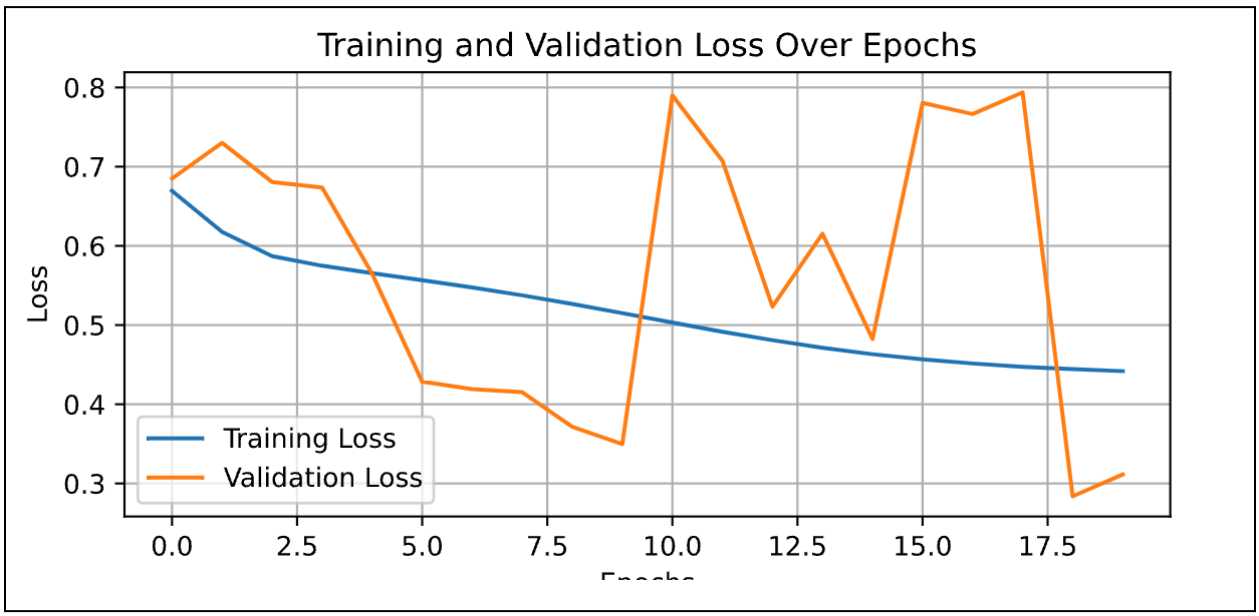
4. **Enhanced Neural Network** (class name: EnhancedSimpleModel3) - Increased scale factor to 16

   Model Architecture:

```python
class EnhancedSimpleModel3(nn.Module):

    def __init__(self, num_bits_per_symbol, H=64, W=512):
        super(EnhancedSimpleModel3, self).__init__()
        scale = 16
        self.linear1 = nn.Linear(in_features=2, out_features=scale * num_bits_per_symbol)
        self.linear2 = nn.Linear(in_features=scale * num_bits_per_symbol, out_features=scale * num_bits_per_symbol * 2)
        self.linear3 = nn.Linear(in_features=scale * num_bits_per_symbol * 2, out_features=scale * num_bits_per_symbol * 2)
        self.linear4 = nn.Linear(in_features=scale * num_bits_per_symbol * 2, out_features=num_bits_per_symbol)
        self.activation = nn.ReLU()

    def forward(self, inputs):
        y = inputs
        z = torch.stack([y.real, y.imag], dim=0)
        z = z.permute(1, 2, 3, 0)
        z = self.linear1(z)
        z = self.activation(z)
        z = self.linear2(z)
        z = self.activation(z)
        z = self.linear3(z)
        z = self.activation(z)
        z = self.linear4(z)
        z = nn.Sigmoid()(z)
        z = z.flatten(-2, -1)
        return z
```

   Number of epochs: 20
   Training Loss after 20 epochs: 0.4418
   Validation Loss after 20 epochs: 0.3113

   Plot of Training Loss and Validation Loss over 20 epochs:

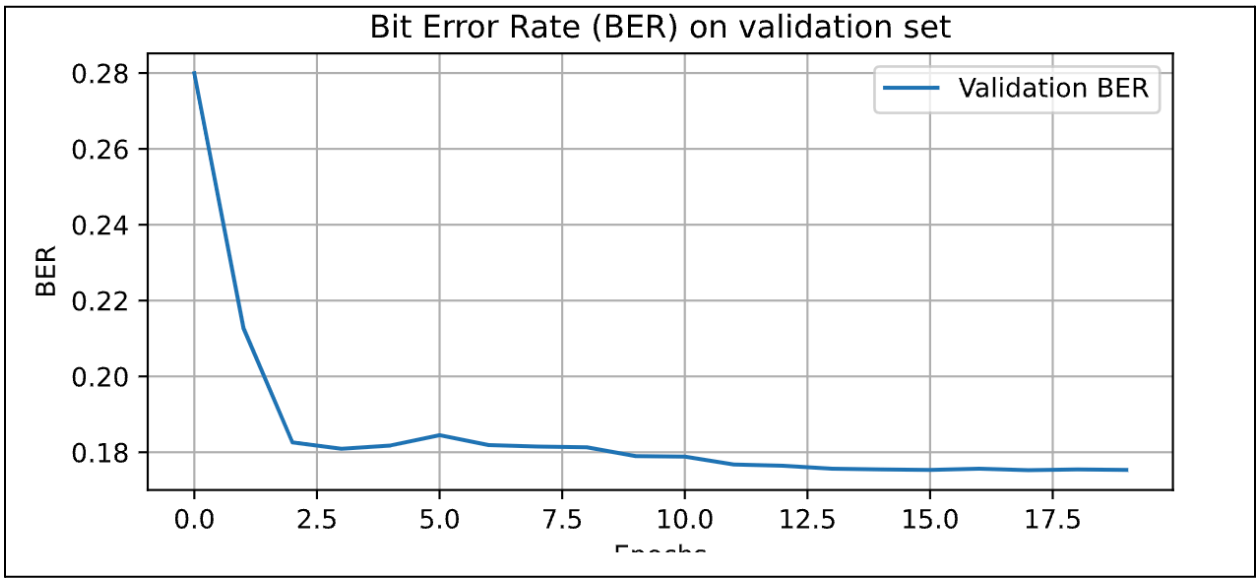


https://github.com/sahaj-totla/Deep-Learning/blob/main/loss_graphs/enhanced_model_3_loss_graph.svg

Optimizer: torch.optim.SGD(model.parameters(), lr=0.1)

Validation BER after 20 epochs: 0.1754

Plot of BER on Validation set:



https://github.com/sahaj-totla/Deep-Learning/blob/main/BER_graphs/enhanced_model_3_BER_graph.svg

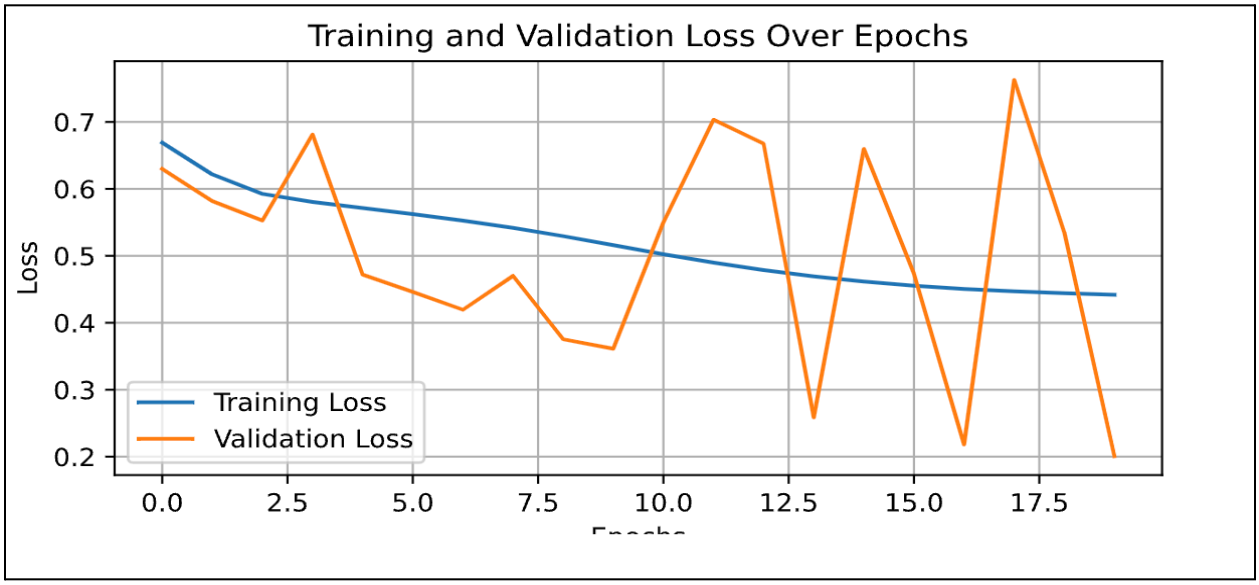
5. **Enhanced Neural Network** (class name: EnhancedSimpleModel4) - increased scale factor to 20, changed optimizer to ASGD, decreased output features in hidden layer 2

   Model Architecture:

```python
class EnhancedSimpleModel4(nn.Module):

    def __init__(self, num_bits_per_symbol, H=64, W=512):
        super(EnhancedSimpleModel4, self).__init__()
        scale = 20
        self.linear1 = nn.Linear(in_features=2, out_features=scale * num_bits_per_symbol)
        self.linear2 = nn.Linear(in_features=scale * num_bits_per_symbol, out_features=scale * num_bits_per_symbol * 2)
        self.linear3 = nn.Linear(in_features=scale * num_bits_per_symbol * 2, out_features=scale * num_bits_per_symbol)
        self.linear4 = nn.Linear(in_features=scale * num_bits_per_symbol, out_features=num_bits_per_symbol)
        self.activation = nn.LeakyReLU()

    def forward(self, inputs):
        y = inputs
        z = torch.stack([y.real, y.imag], dim=0)
        z = z.permute(1, 2, 3, 0)
        z = self.linear1(z)
        z = self.activation(z)
        z = self.linear2(z)
        z = self.activation(z)
        z = self.linear3(z)
        z = self.activation(z)
        z = self.linear4(z)
        z = nn.Sigmoid()(z)
        z = z.flatten(-2, -1)
        return z
```

   Number of epochs: 20
   Training Loss after 20 epochs: 0.4418
   Validation Loss after 20 epochs: 0.2006

   Plot of Training Loss and Validation Loss over 20 epochs:

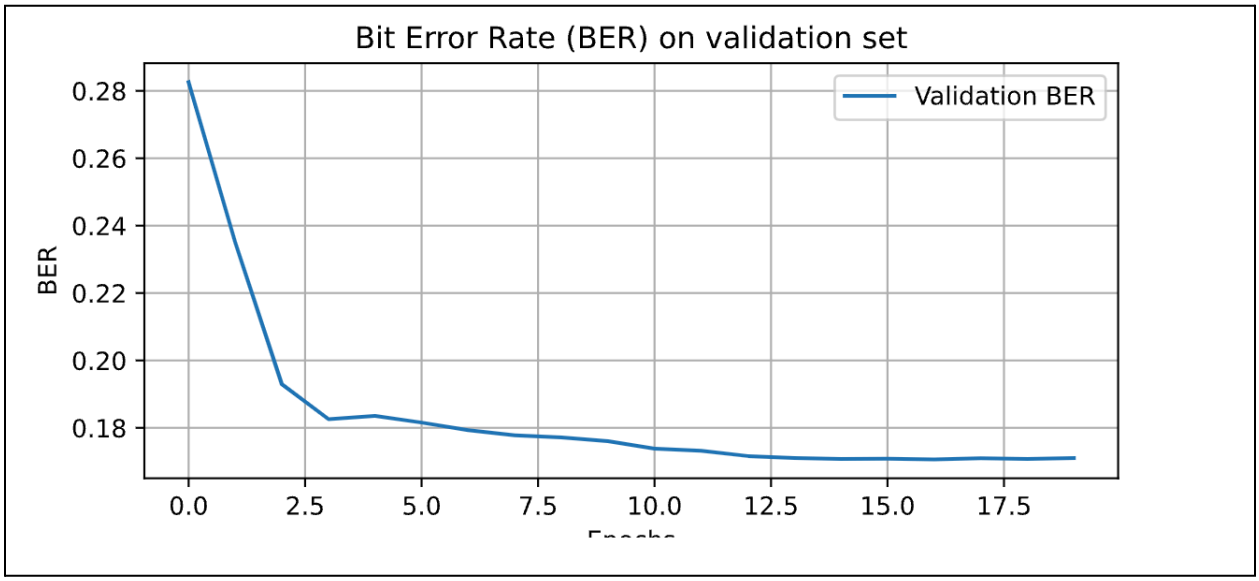


https://github.com/sahaj-totla/Deep-Learning/blob/main/loss_graphs/enhanced_model_4_loss_graph.svg

Optimizer: torch.optim.ASGD(model.parameters(), lr=0.1)

Validation BER after 20 epochs: 0.1710

Plot of BER on Validation set:



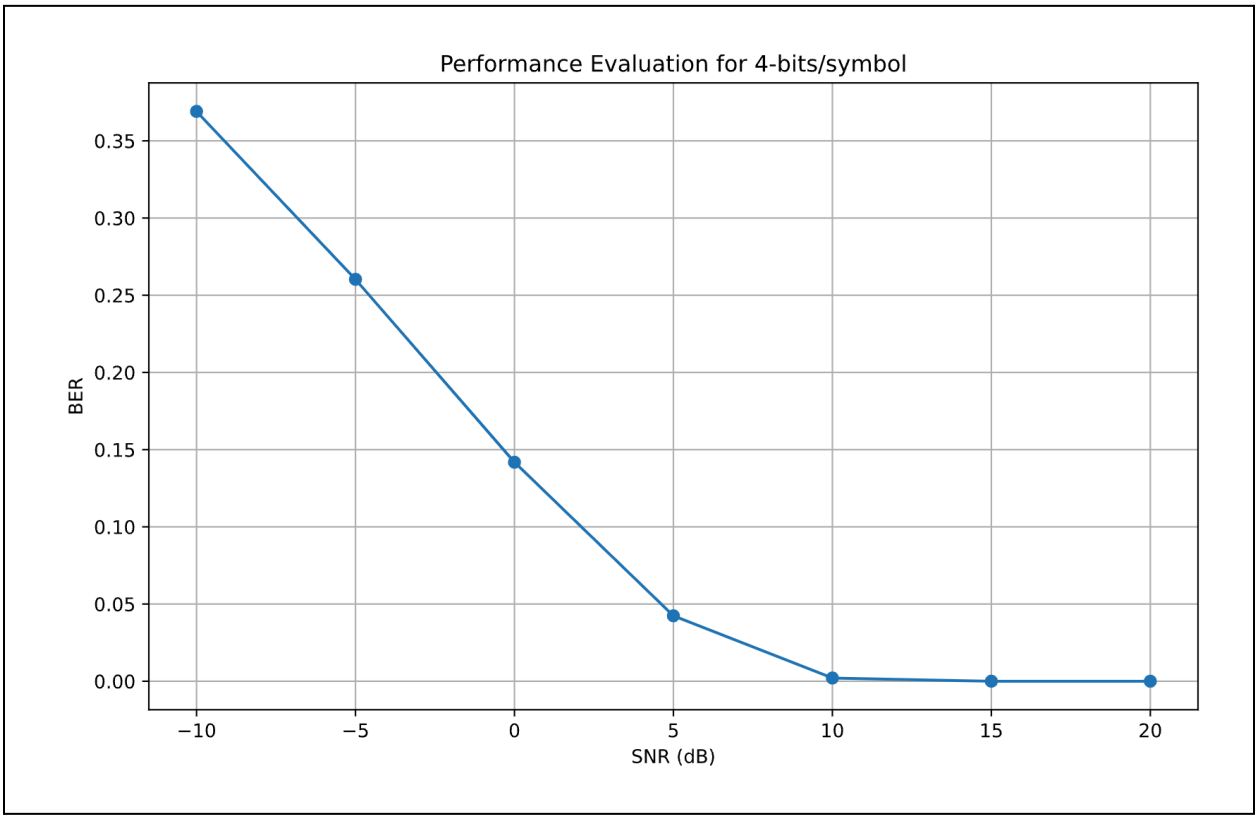
https://github.com/sahaj-totla/Deep-Learning/blob/main/BER_graphs/enhanced_model_4_BER_graph.svg

EnhancedSimpleModel4 came out to be the best model with Validation BER at 0.1710 with moderate training and validation loss.

II. Evaluating Performance across System Configurations like Signal to Noise Ratio

Model: EnhancedSimpleModel4
Optimizer: ASGD

BER vs SNR graph for performance evaluation for 4 bits per symbol



https://github.com/sahaj-totla/Deep-Learning/blob/main/SNR_graph/enhanced_model_4_BER_vs_SNR_graph.svg