Batch Normalization in PyTorch

```
class Net(nn.Module):
    def __init__(self, use_batch_norm, insize = 784,
                                hid_dim=256, outsize=10):
        super(Net, self).__init__()
```

usual stuff

```
        self.insize = insize
        self.hid_dim = hid-dim
        self.outsize = outsize
```

```
        self.use_batch_norm = use_batch_norm
```

→ True / False

first hidden layer with optional b_norm

```
        if use_batch_norm:
            self.fc1 = nn.Linear(insize, hid-dim*2, bias= False)
            self.batch_norm1 = nn.BatchNorm1d(hid-dim*2)
        else:
            self.fc1 = nn.Linear(insize, hid-dim * 2)
```

second hidden layer with optional b_norm

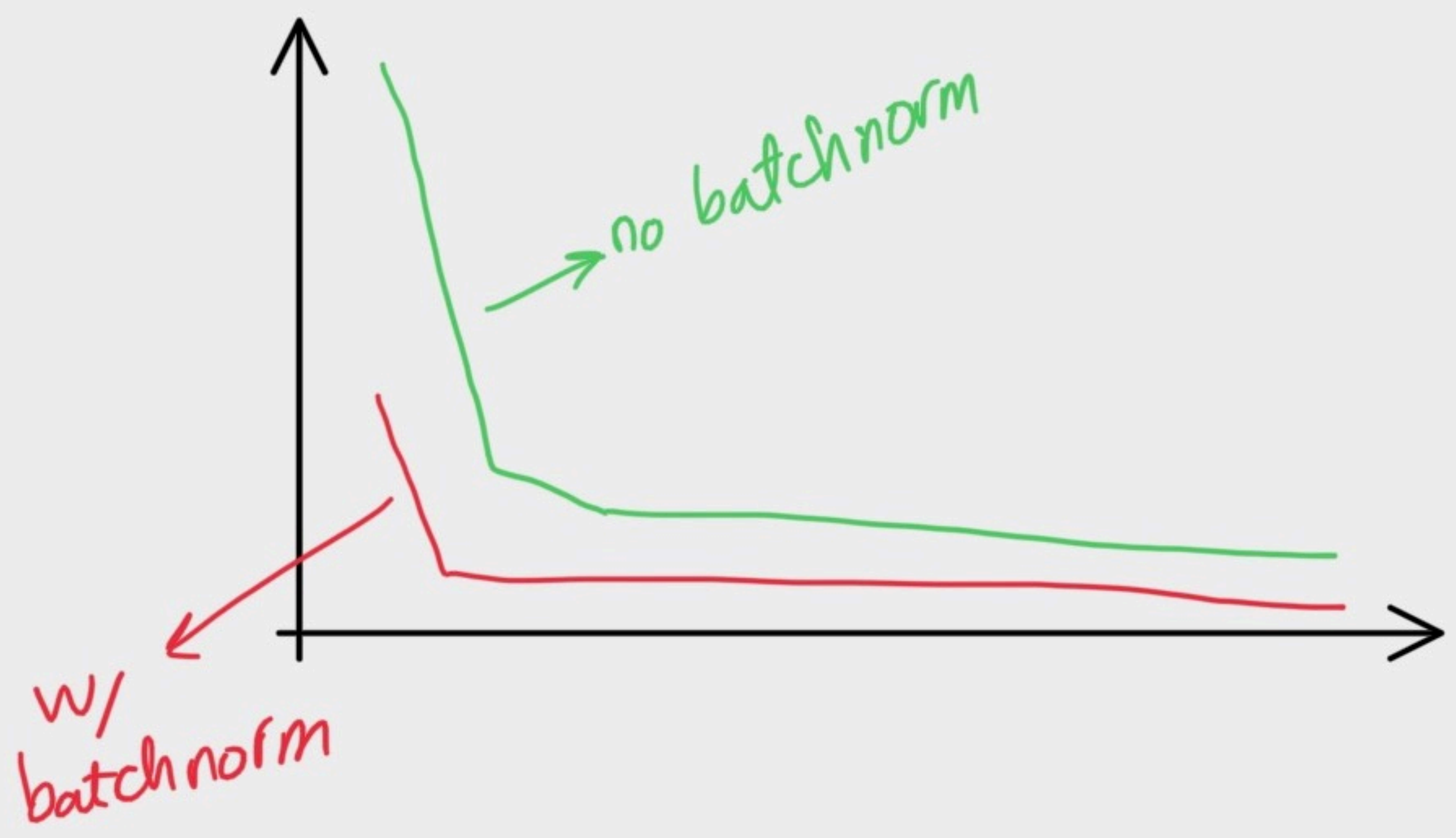
```
        if use_batch_norm:
            self.fc2 = nn.Linear(hid_dim*2, hid_dim, bias= False)
            self.batch_norm2 = nn.BatchNorm1d(hid-dim)
        else:
            self.fc2 = nn.Linear(hid_dim*2, hid_dim)
```

```
    self.fc3 = nn.Linear(hid_dim, outsize)  → final fC


def forward(self, x):
    x = x.view(-1, 28*28)  → flatten
first  ┌ x = self.fc1(x)
layer  │ if self.use_batch_norm:
       │     x = self.batch_norm1(x)
       └ x = F.relu(x)
second ┌ x = self.fc2(x)
layer  │ if self.use_batch_norm:
       │     x = self.batch_norm2(x)
       └ x = F.relu(x)


    x = self.fc3(x)  → final fC, no batch norm or activation
    return x
```

How does it affect the training?



Summary

- Layers with batchnorm: bias = False
- BatchNorm1d, BatchNorm2d in PyTorch
- batchnorm layer: before activation function