# Assignment 4

due December 9

CSMC 352: Machine Learning
Prof. Rose Sloan
Nathan Cho
nc5123@bard.edu

## 1. MNIST

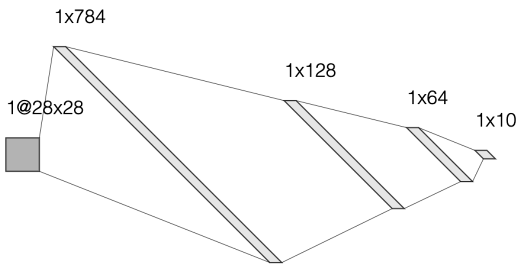
### Implementation

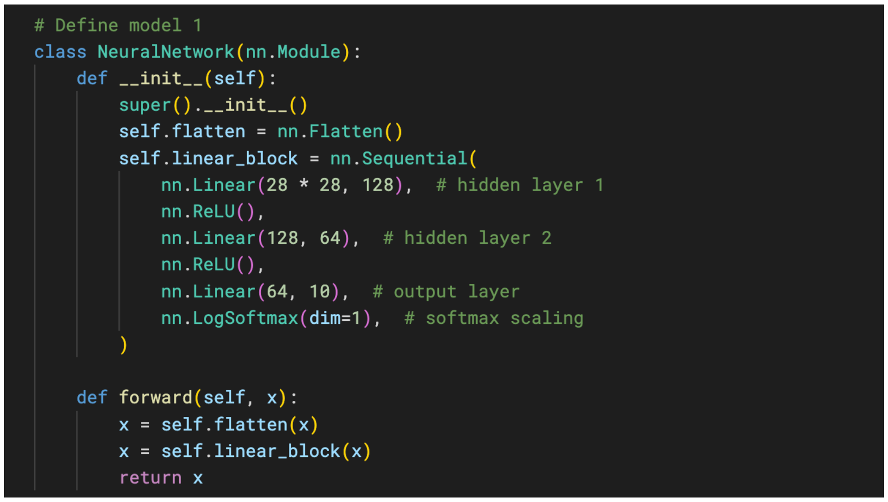
Based on PyTorch Tutorials, two neural networks were created, trained and tested.

Both models were trained and tested on **minibatch with size of 64.**
Both models' output layer was implemented using **softmax.**
Both models used **negative log likelihood loss** as loss function.
Both models were optimized with **stochastic gradient descent.**
Both models were **trained for 20 epochs** each.

#### Model 1: Basic Neural Network

The following is the architecture and definition of the Basic Neural Network model.

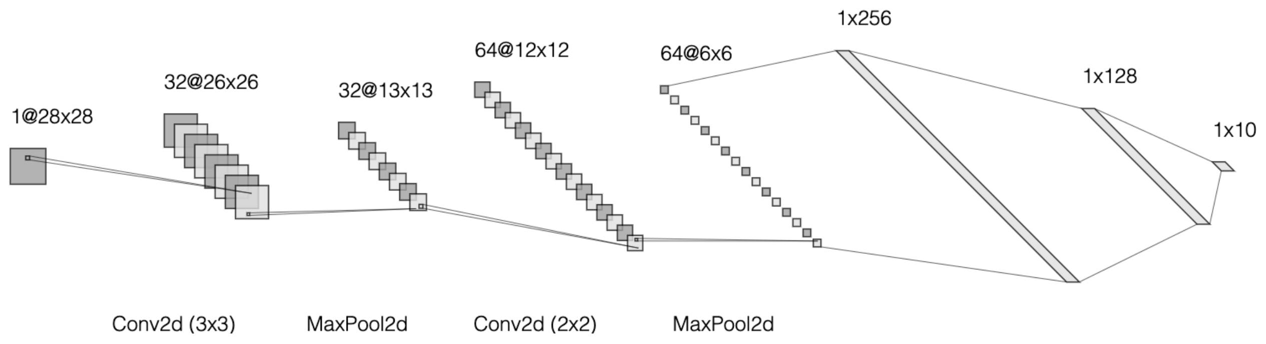


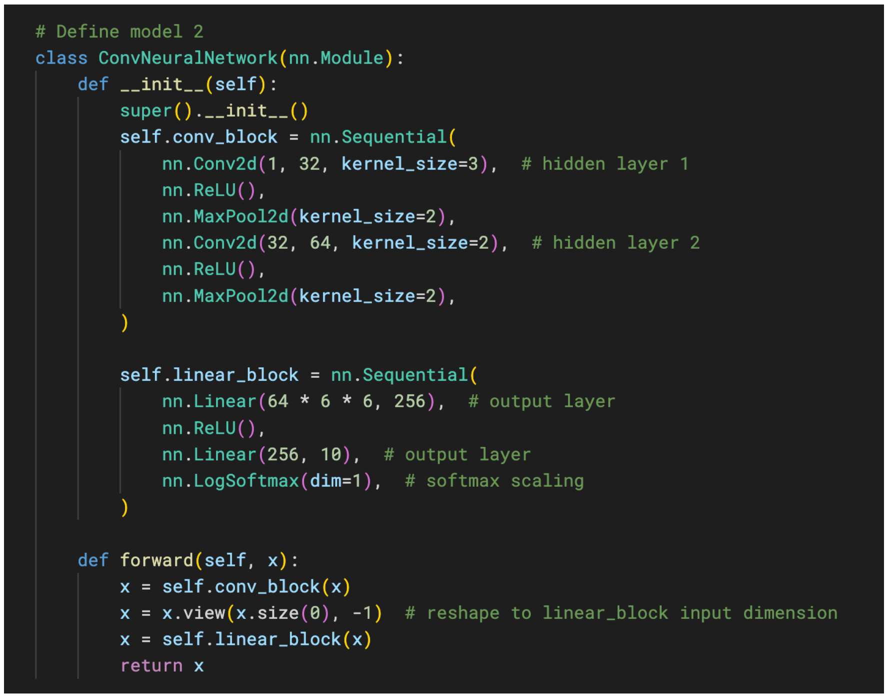


```
# Define model 1
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear_block = nn.Sequential(
            nn.Linear(28 * 28, 128),  # hidden layer 1
            nn.ReLU(),
            nn.Linear(128, 64),  # hidden layer 2
            nn.ReLU(),
            nn.Linear(64, 10),  # output layer
            nn.LogSoftmax(dim=1),  # softmax scaling
        )

    def forward(self, x):
        x = self.flatten(x)
        x = self.linear_block(x)
        return x
```

## Model 2: Convolutional Neural Network

The following is the architecture and definition of the Convolutional Neural Network model.





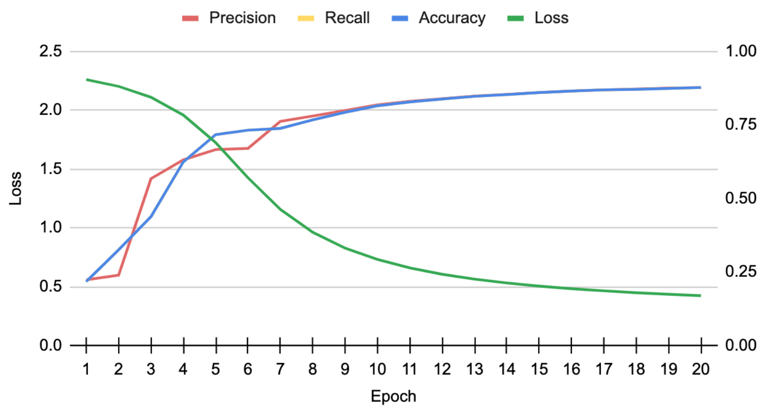
```
# Define model 2
class ConvNeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv_block = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3),  # hidden layer 1
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2),
            nn.Conv2d(32, 64, kernel_size=2),  # hidden layer 2
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2),
        )

        self.linear_block = nn.Sequential(
            nn.Linear(64 * 6 * 6, 256),  # output layer
            nn.ReLU(),
            nn.Linear(256, 10),  # output layer
            nn.LogSoftmax(dim=1),  # softmax scaling
        )

    def forward(self, x):
        x = self.conv_block(x)
        x = x.view(x.size(0), -1)  # reshape to linear_block input dimension
        x = self.linear_block(x)
        return x
```

## Results

Raw terminal outputs are available at ***problem1_results.txt***.

### Model 1: Basic Neural Network

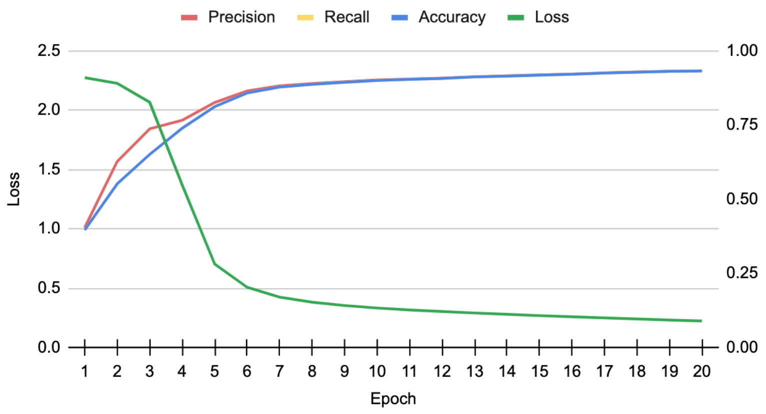
After 20 epochs, the Basic Neural Network model reached **0.877** in accuracy, precision, recall metrics. The following are metrics at each epoch and confusion matrix at last epoch.



| BasicNN | | Classified digits | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **Σ** |
| Actual digits | **0** | 949 | 0 | 5 | 1 | 0 | 14 | 7 | 2 | 2 | 0 | **31** |
| | **1** | 0 | 1101 | 1 | 7 | 0 | 2 | 3 | 1 | 20 | 0 | **34** |
| | **2** | 13 | 18 | 876 | 21 | 20 | 2 | 28 | 11 | 36 | 7 | **156** |
| | **3** | 4 | 2 | 21 | 890 | 1 | 35 | 2 | 23 | 27 | 5 | **120** |
| | **4** | 2 | 2 | 2 | 1 | 885 | 0 | 15 | 0 | 15 | 60 | **97** |
| | **5** | 19 | 4 | 9 | 68 | 12 | 682 | 28 | 9 | 44 | 17 | **210** |
| | **6** | 19 | 3 | 13 | 1 | 23 | 19 | 872 | 1 | 7 | 0 | **86** |
| | **7** | 5 | 22 | 29 | 1 | 7 | 0 | 0 | 904 | 3 | 57 | **124** |
| | **8** | 5 | 7 | 11 | 33 | 12 | 52 | 19 | 6 | 805 | 24 | **169** |
| | **9** | 9 | 4 | 6 | 11 | 77 | 14 | 0 | 44 | 15 | 829 | **180** |

### Model 2: Convolutional Neural Network

After 20 epochs, the Convolutional Neural Network model reached **0.932** in accuracy, precision, recall metrics. The following are metrics at each epoch and confusion matrix at last epoch.



| CNN | | Classified digits | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **Σ** |
| Actual digits | **0** | 965 | 0 | 0 | 3 | 0 | 4 | 6 | 1 | 1 | 0 | **15** |
| | **1** | 0 | 1114 | 3 | 2 | 0 | 1 | 4 | 0 | 11 | 0 | **21** |
| | **2** | 13 | 4 | 935 | 26 | 6 | 2 | 9 | 16 | 18 | 3 | **97** |
| | **3** | 4 | 2 | 8 | 940 | 0 | 25 | 2 | 9 | 12 | 8 | **70** |
| | **4** | 1 | 2 | 6 | 0 | 900 | 0 | 13 | 1 | 6 | 53 | **82** |
| | **5** | 10 | 4 | 0 | 20 | 5 | 828 | 11 | 1 | 8 | 5 | **64** |
| | **6** | 13 | 5 | 2 | 2 | 7 | 21 | 905 | 2 | 1 | 0 | **53** |
| | **7** | 3 | 9 | 20 | 5 | 5 | 0 | 0 | 933 | 4 | 49 | **95** |
| | **8** | 8 | 3 | 3 | 28 | 9 | 26 | 11 | 3 | 862 | 21 | **112** |
| | **9** | 11 | 6 | 2 | 10 | 16 | 7 | 0 | 7 | 6 | 944 | **65** |

### Conclusion

After the same number of training epochs, Convolutional Neural Network performed much better than Basic Neural Network. This difference comes from CNN's ability to learn high level features such as shape and edges of the digits. Extracting this spatial data is crucial in digit recognition and image recognition problems as the model learns the components in the image instead of exact position of pixels. These advantages make CNN more efficient and effective compared to the Basic Neural Network.

# 2. HR Dataset

## Dataset

The goal of this task is **predicting the employee turnover risk with HR data**. The dataset is provided by Giri Pujar on Kaggle (licensed CC0: Public Domain). It consists of 14,999 rows with 10 columns of features which includes the following:

Employment status (current or previous employee)
Satisfaction level
Evaluation score
Number of projects assigned
Number of years employed
Average monthly working hours
Accident history
Promotion history for last 5 years
Current department
Salary level

Employment status (turnover) is the label which the models will try to predict from other features. The dataset was imbalanced with only 23.8% of labels being positive.

## Implementation

### Data Preprocessing

#### Feature Extraction and Vectorization

8 new features were extracted from the original dataset:

Mean deviation of

Number of projects assigned
Average monthly working hours
Promotion history for last 5 years
Salary level

compared to

Employees in the same department
Employees with same years employed

These features were extracted because poor working conditions (number of projects, working hours) and lack of compensation (promotion, salary) are one of the big reasons why employees leave. By comparing these metrics in their perspective such as employees in their department and employees who started working at the same time, these features can give a deeper insight of how employees are feeling about their current position.

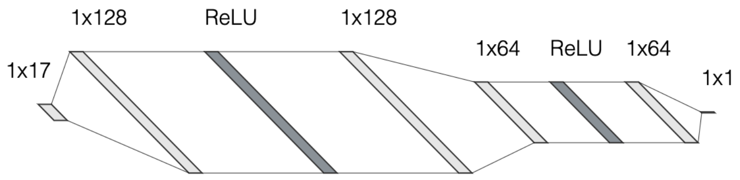
In the original dataset, department and salary level were encoded in string. These were vectorized into integer ID values using **pandas.Categorical**. Salary level “low” was converted into 0, “medium” into 1, “high” into 2.

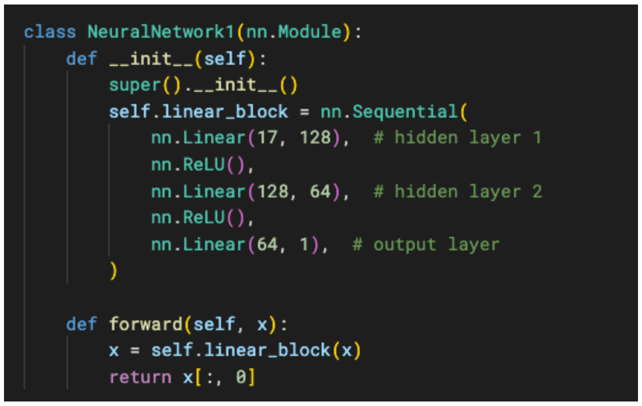
## Neural Network Architecture

The following are the architectures and definitions of each Neural Network model.

All models were trained and tested on **minibatch with size of 64** selected from **stratified subset.**
All models used **binary cross entropy loss with sigmoid** as loss function.
All models were optimized with **stochastic gradient descent.**
All models were **trained for 750 epochs** each.

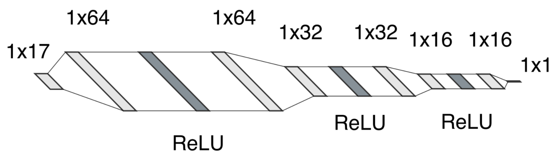
### Model 1: 2 Hidden Layers with ReLU

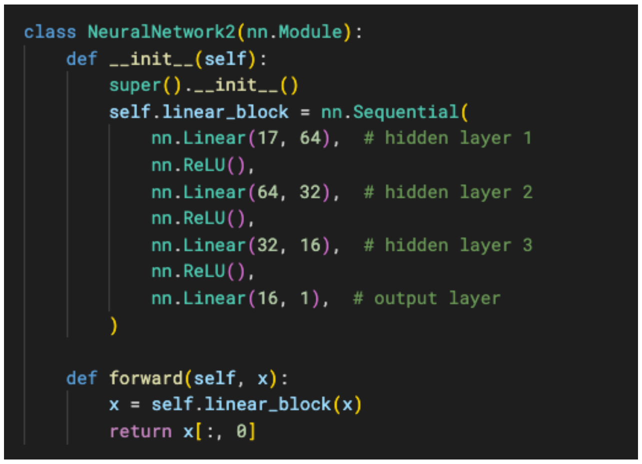




```python
class NeuralNetwork1(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear_block = nn.Sequential(
            nn.Linear(17, 128),  # hidden layer 1
            nn.ReLU(),
            nn.Linear(128, 64),  # hidden layer 2
            nn.ReLU(),
            nn.Linear(64, 1),  # output layer
        )

    def forward(self, x):
        x = self.linear_block(x)
        return x[:, 0]
```

### Model 2: 3 Hidden Layers with ReLU

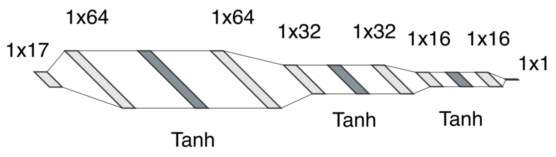


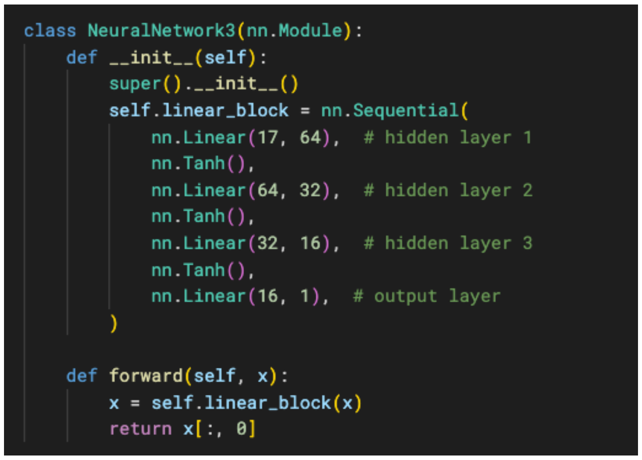


```python
class NeuralNetwork2(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear_block = nn.Sequential(
            nn.Linear(17, 64),  # hidden layer 1
            nn.ReLU(),
            nn.Linear(64, 32),  # hidden layer 2
            nn.ReLU(),
            nn.Linear(32, 16),  # hidden layer 3
            nn.ReLU(),
            nn.Linear(16, 1),  # output layer
        )

    def forward(self, x):
        x = self.linear_block(x)
        return x[:, 0]
```

In order to learn more complex patterns in the data, one additional layer was added. The first layer is scaled down to 64 units to reduce computational complexity.

## Model 3: 3 Hidden Layers with Tanh

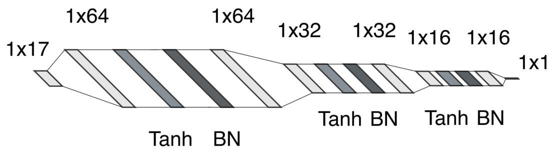


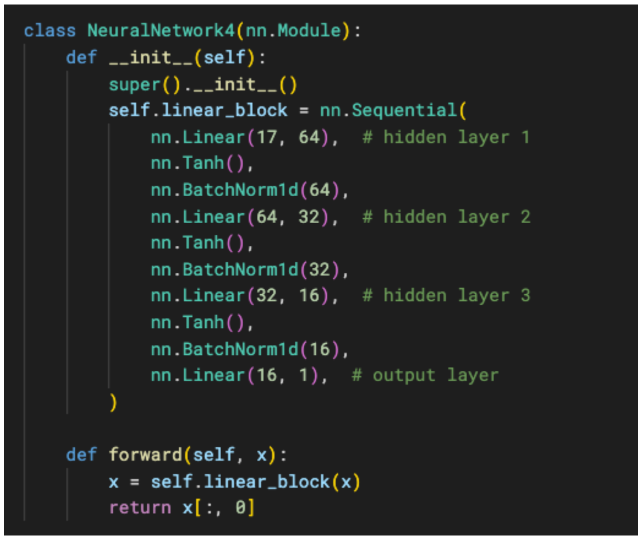


```
class NeuralNetwork3(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear_block = nn.Sequential(
            nn.Linear(17, 64),   # hidden layer 1
            nn.Tanh(),
            nn.Linear(64, 32),   # hidden layer 2
            nn.Tanh(),
            nn.Linear(32, 16),   # hidden layer 3
            nn.Tanh(),
            nn.Linear(16, 1),  # output layer
        )

    def forward(self, x):
        x = self.linear_block(x)
        return x[:, 0]
```

In order to measure which activation function performs better, ReLU was replaced to Tanh function.

## Model 4: 3 Hidden Layers with Tanh and BatchNorm





```
class NeuralNetwork4(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear_block = nn.Sequential(
            nn.Linear(17, 64),   # hidden layer 1
            nn.Tanh(),
            nn.BatchNorm1d(64),
            nn.Linear(64, 32),   # hidden layer 2
            nn.Tanh(),
            nn.BatchNorm1d(32),
            nn.Linear(32, 16),   # hidden layer 3
            nn.Tanh(),
            nn.BatchNorm1d(16),
            nn.Linear(16, 1),  # output layer
        )

    def forward(self, x):
        x = self.linear_block(x)
        return x[:, 0]
```

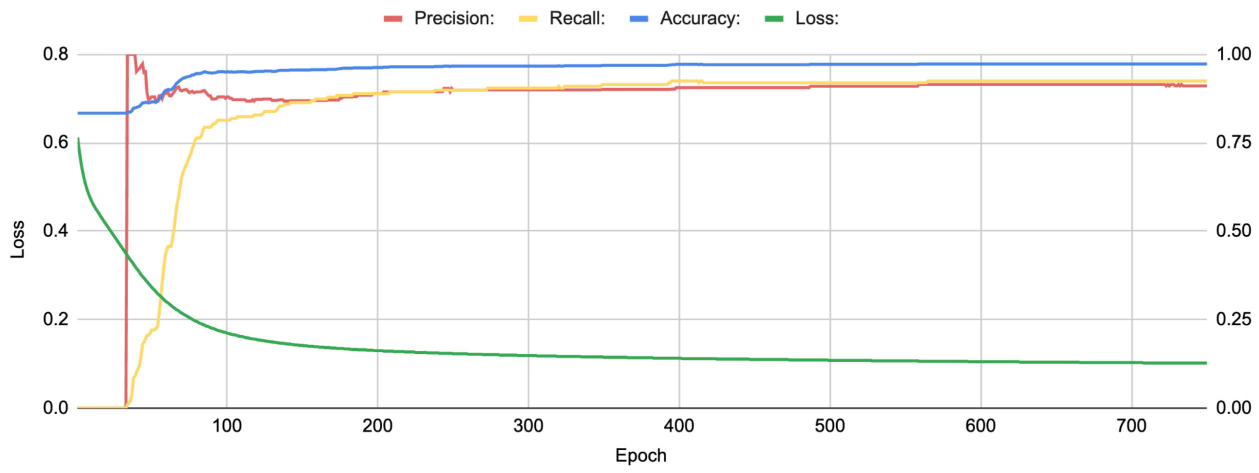
In order to speed up the training process, batch normalization was added in between layers.

## Results

### Individual Model Performance

#### Model 1: 2 Hidden Layers with ReLU

After 750 epochs, Model 1 reached **0.973** in accuracy, 0.914 in precision, 0.925 in recall and 0.103 in loss.

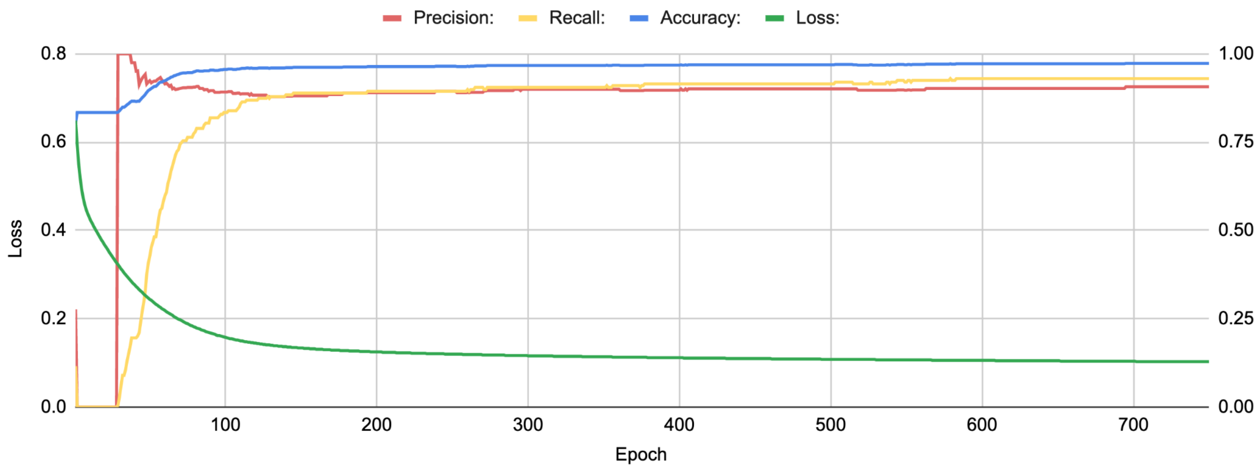


Observing the training progress shows that the model started predicting correct values after 40 epochs as loss continuously decreased.

#### Feature extraction

In order to see the significance of the extracted features, Model 1 was tested **without** the 8 extra features. After 750 epochs, model 1 reached **0.973** in accuracy, 0.905 in precision, 0.930 in recall and 0.104 in loss.

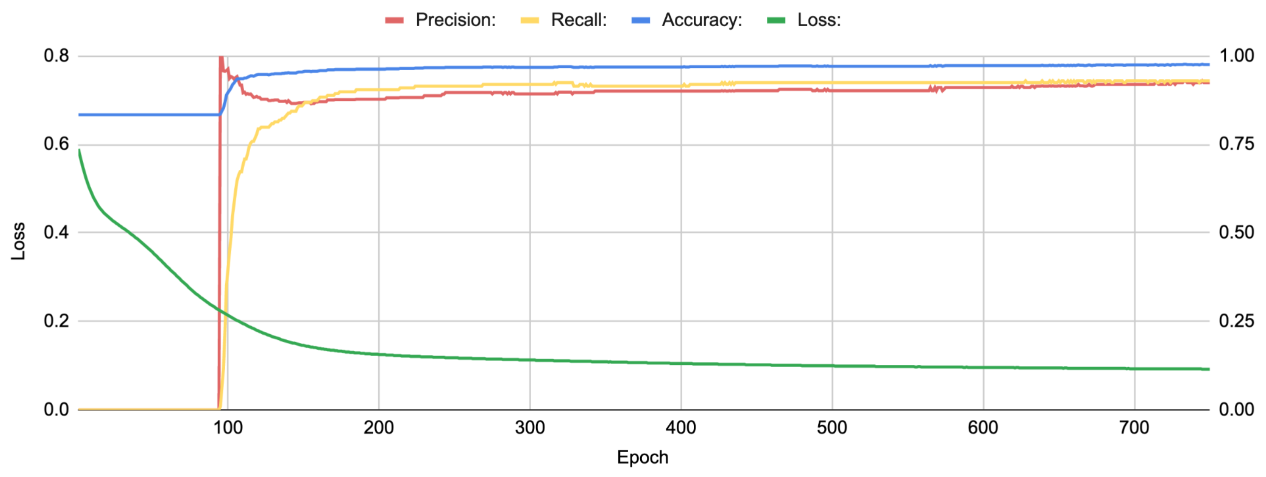
As it showed lower precision score without feature extraction, we can infer that the new features help the model make less false positive predictions.



Observing the training progress shows that the model started learning the data approximately 10 epochs earlier than the model with extra features. This can be because there are less parameters to adjust during the training progress with a smaller set of features.

## Model 2: 3 Hidden Layers with ReLU

After 750 epochs, Model 2 reached **0.975** in accuracy, 0.921 in precision, 0.929 in recall and 0.093 in loss.



Observing the training progress shows that the model started predicting the correct value around epoch 90 and the loss value took much longer to converge. This can be due to having more parameters to adjust as the extra layer was added.

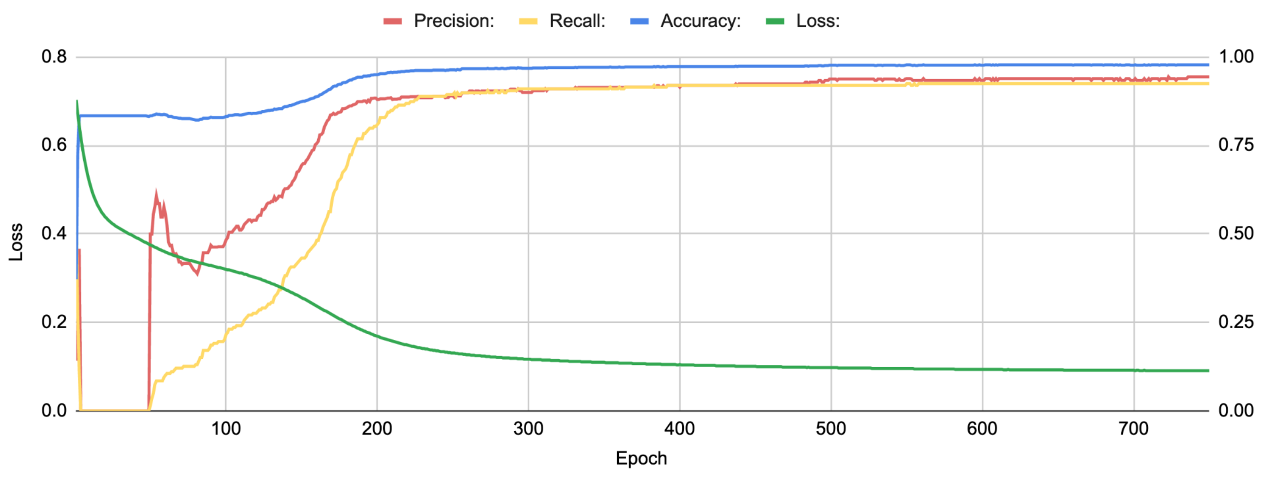
## Comparison Between Model 1 and Model 2

Although the difference in metrics aren't significant, we can see Model 2 performed better in all metrics, especially in loss. Key architectural differences between Model 1 and Model 2 is the number of units in one layer and number of layers.

From the metrics, we can infer that Model 2 learned more complex patterns in the data better than Model 1 with the help of an extra layer. But from the training progress, we can see that the extra layer made the training progress much slower.

## Model 3: 3 Hidden Layers with Tanh

After 750 epochs, Model 3 reached **0.978** in accuracy, 0.939 in precision, 0.925 in recall and 0.092 in loss.



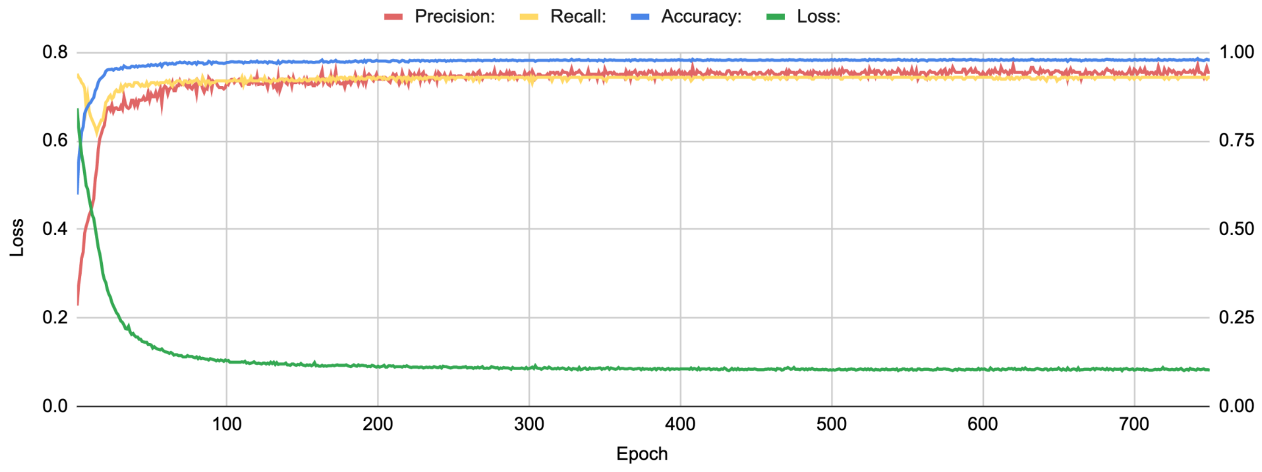
Observing the training progress shows that the model took much longer to converge compared to ReLU.

### Comparison Between Model 2 and Model 3

Changing the activation function showed significant changes on how the model learns the data. Although using Tanh function instead of ReLU led to longer training time, it showed higher accuracy and precision. One possible reason for this outcome can be the symmetry of the Tanh function, which could have helped the model fit better to the dataset which was scaled to mean 0.

### Model 4: 3 Hidden Layers with Tanh and BatchNorm

After 750 epochs, Model 4 reached **0.979** in accuracy, 0.944 in precision, 0.929 in recall and 0.084 in loss.



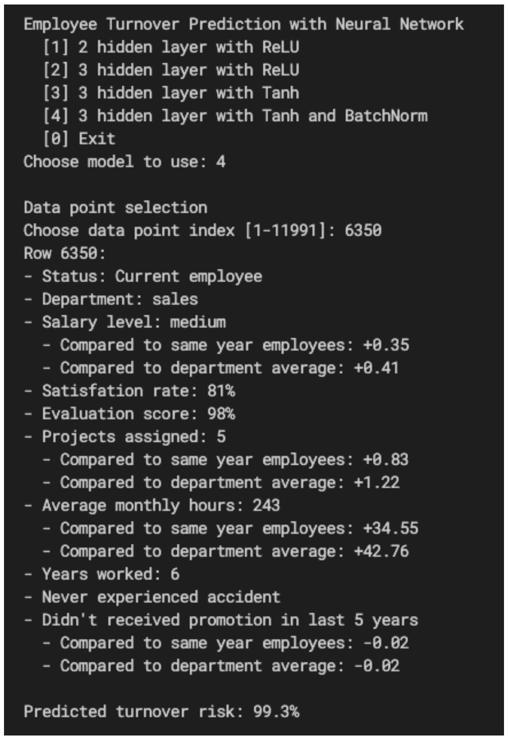
Observing the training progress shows that the model converged faster than any other models using batch normalization.

### Comparison Between Model 3 and Model 4

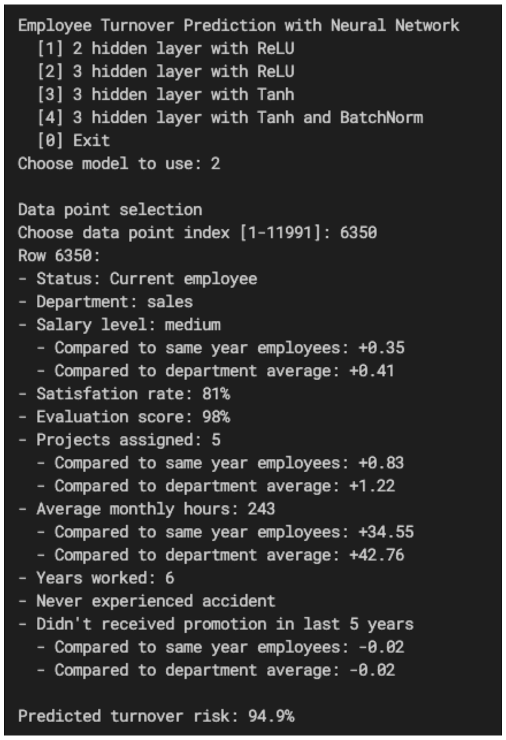
Applying batch normalization technique in Model 4 showed better performance as it trained for more epochs after convergence. The model converged around epoch 150, but it took much longer training every epoch compared to other models due to the computational complexity. The reason for faster convergence is because batch normalization reduces the amount of internal covariate shift within the network by normalizing the input distribution every epoch.

## Interactive Demo

Running ***problem2.py*** will export all 4 trained models into a PyTorch file in the same directory. You can use ***problem2_interactive.py*** to load these exported models and interactively test each data point in the dataset. The following are example use cases:

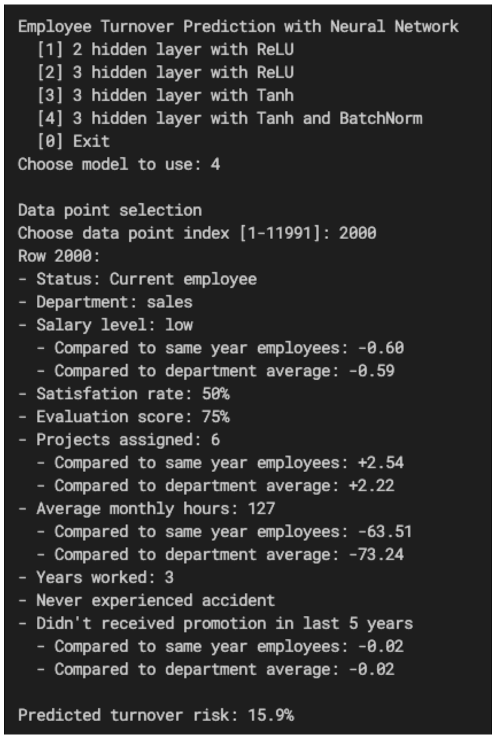


```
Employee Turnover Prediction with Neural Network
  [1] 2 hidden layer with ReLU
  [2] 3 hidden layer with ReLU
  [3] 3 hidden layer with Tanh
  [4] 3 hidden layer with Tanh and BatchNorm
  [0] Exit
Choose model to use: 4

Data point selection
Choose data point index [1-11991]: 6350
Row 6350:
- Status: Current employee
- Department: sales
- Salary level: medium
  - Compared to same year employees: +0.35
  - Compared to department average: +0.41
- Satisfation rate: 81%
- Evaluation score: 98%
- Projects assigned: 5
  - Compared to same year employees: +0.83
  - Compared to department average: +1.22
- Average monthly hours: 243
  - Compared to same year employees: +34.55
  - Compared to department average: +42.76
- Years worked: 6
- Never experienced accident
- Didn't received promotion in last 5 years
  - Compared to same year employees: -0.02
  - Compared to department average: -0.02

Predicted turnover risk: 99.3%
```

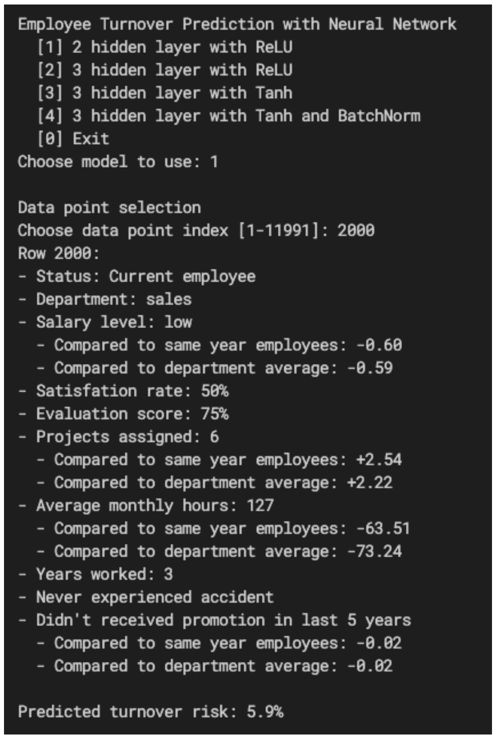


```
Employee Turnover Prediction with Neural Network
  [1] 2 hidden layer with ReLU
  [2] 3 hidden layer with ReLU
  [3] 3 hidden layer with Tanh
  [4] 3 hidden layer with Tanh and BatchNorm
  [0] Exit
Choose model to use: 2

Data point selection
Choose data point index [1-11991]: 6350
Row 6350:
- Status: Current employee
- Department: sales
- Salary level: medium
  - Compared to same year employees: +0.35
  - Compared to department average: +0.41
- Satisfation rate: 81%
- Evaluation score: 98%
- Projects assigned: 5
  - Compared to same year employees: +0.83
  - Compared to department average: +1.22
- Average monthly hours: 243
  - Compared to same year employees: +34.55
  - Compared to department average: +42.76
- Years worked: 6
- Never experienced accident
- Didn't received promotion in last 5 years
  - Compared to same year employees: -0.02
  - Compared to department average: -0.02

Predicted turnover risk: 94.9%
```

Testing with Employee 6350 shows that this employee is currently employed but has extremely high turnover risk, possibly due to above average working hours and no promotion compared to years worked. Using Model 4 and Model 2 on the same employee shows different predicted risks.

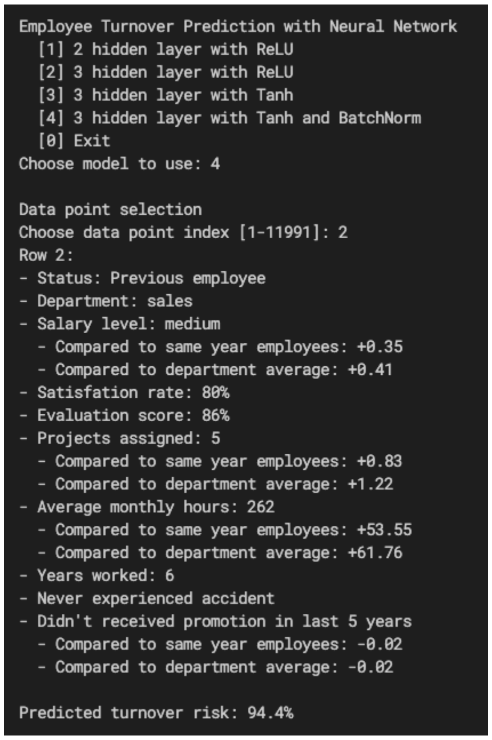


```
Employee Turnover Prediction with Neural Network
  [1] 2 hidden layer with ReLU
  [2] 3 hidden layer with ReLU
  [3] 3 hidden layer with Tanh
  [4] 3 hidden layer with Tanh and BatchNorm
  [0] Exit
Choose model to use: 4

Data point selection
Choose data point index [1-11991]: 2000
Row 2000:
- Status: Current employee
- Department: sales
- Salary level: low
  - Compared to same year employees: -0.60
  - Compared to department average: -0.59
- Satisfation rate: 50%
- Evaluation score: 75%
- Projects assigned: 6
  - Compared to same year employees: +2.54
  - Compared to department average: +2.22
- Average monthly hours: 127
  - Compared to same year employees: -63.51
  - Compared to department average: -73.24
- Years worked: 3
- Never experienced accident
- Didn't received promotion in last 5 years
  - Compared to same year employees: -0.02
  - Compared to department average: -0.02

Predicted turnover risk: 15.9%
```

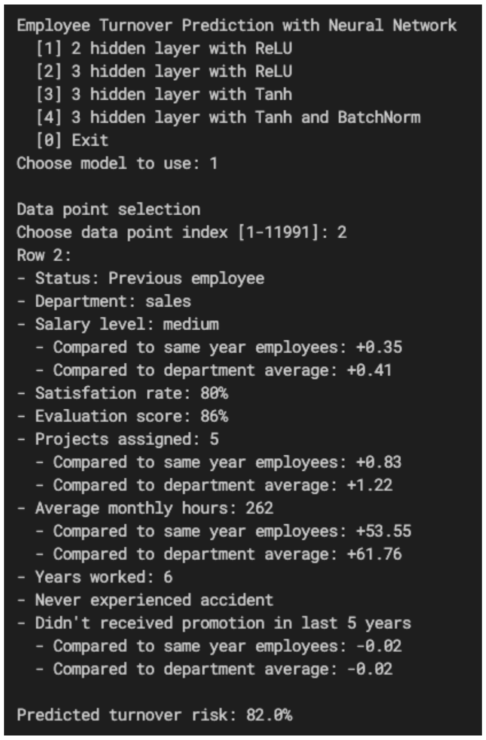


```
Employee Turnover Prediction with Neural Network
  [1] 2 hidden layer with ReLU
  [2] 3 hidden layer with ReLU
  [3] 3 hidden layer with Tanh
  [4] 3 hidden layer with Tanh and BatchNorm
  [0] Exit
Choose model to use: 1

Data point selection
Choose data point index [1-11991]: 2000
Row 2000:
- Status: Current employee
- Department: sales
- Salary level: low
  - Compared to same year employees: -0.60
  - Compared to department average: -0.59
- Satisfation rate: 50%
- Evaluation score: 75%
- Projects assigned: 6
  - Compared to same year employees: +2.54
  - Compared to department average: +2.22
- Average monthly hours: 127
  - Compared to same year employees: -63.51
  - Compared to department average: -73.24
- Years worked: 3
- Never experienced accident
- Didn't received promotion in last 5 years
  - Compared to same year employees: -0.02
  - Compared to department average: -0.02

Predicted turnover risk: 5.9%
```

Testing with Employee 2000 shows that this employee is currently employed with low turnover risk. Using Model 4 and Model 1 on the same employee shows different predicted risks.



```
Employee Turnover Prediction with Neural Network
  [1] 2 hidden layer with ReLU
  [2] 3 hidden layer with ReLU
  [3] 3 hidden layer with Tanh
  [4] 3 hidden layer with Tanh and BatchNorm
  [0] Exit
Choose model to use: 4

Data point selection
Choose data point index [1-11991]: 2
Row 2:
- Status: Previous employee
- Department: sales
- Salary level: medium
  - Compared to same year employees: +0.35
  - Compared to department average: +0.41
- Satisfation rate: 80%
- Evaluation score: 86%
- Projects assigned: 5
  - Compared to same year employees: +0.83
  - Compared to department average: +1.22
- Average monthly hours: 262
  - Compared to same year employees: +53.55
  - Compared to department average: +61.76
- Years worked: 6
- Never experienced accident
- Didn't received promotion in last 5 years
  - Compared to same year employees: -0.02
  - Compared to department average: -0.02

Predicted turnover risk: 94.4%
```



```
Employee Turnover Prediction with Neural Network
  [1] 2 hidden layer with ReLU
  [2] 3 hidden layer with ReLU
  [3] 3 hidden layer with Tanh
  [4] 3 hidden layer with Tanh and BatchNorm
  [0] Exit
Choose model to use: 1

Data point selection
Choose data point index [1-11991]: 2
Row 2:
- Status: Previous employee
- Department: sales
- Salary level: medium
  - Compared to same year employees: +0.35
  - Compared to department average: +0.41
- Satisfation rate: 80%
- Evaluation score: 86%
- Projects assigned: 5
  - Compared to same year employees: +0.83
  - Compared to department average: +1.22
- Average monthly hours: 262
  - Compared to same year employees: +53.55
  - Compared to department average: +61.76
- Years worked: 6
- Never experienced accident
- Didn't received promotion in last 5 years
  - Compared to same year employees: -0.02
  - Compared to department average: -0.02

Predicted turnover risk: 82.0%
```

Testing with Employee 2 shows that this employee already left the company. Using Model 4 correctly predicts the turnover risk as very high (94%), but Model 1 only predicts as high (82%).