# Anomaly Detection Using AutoEncoders

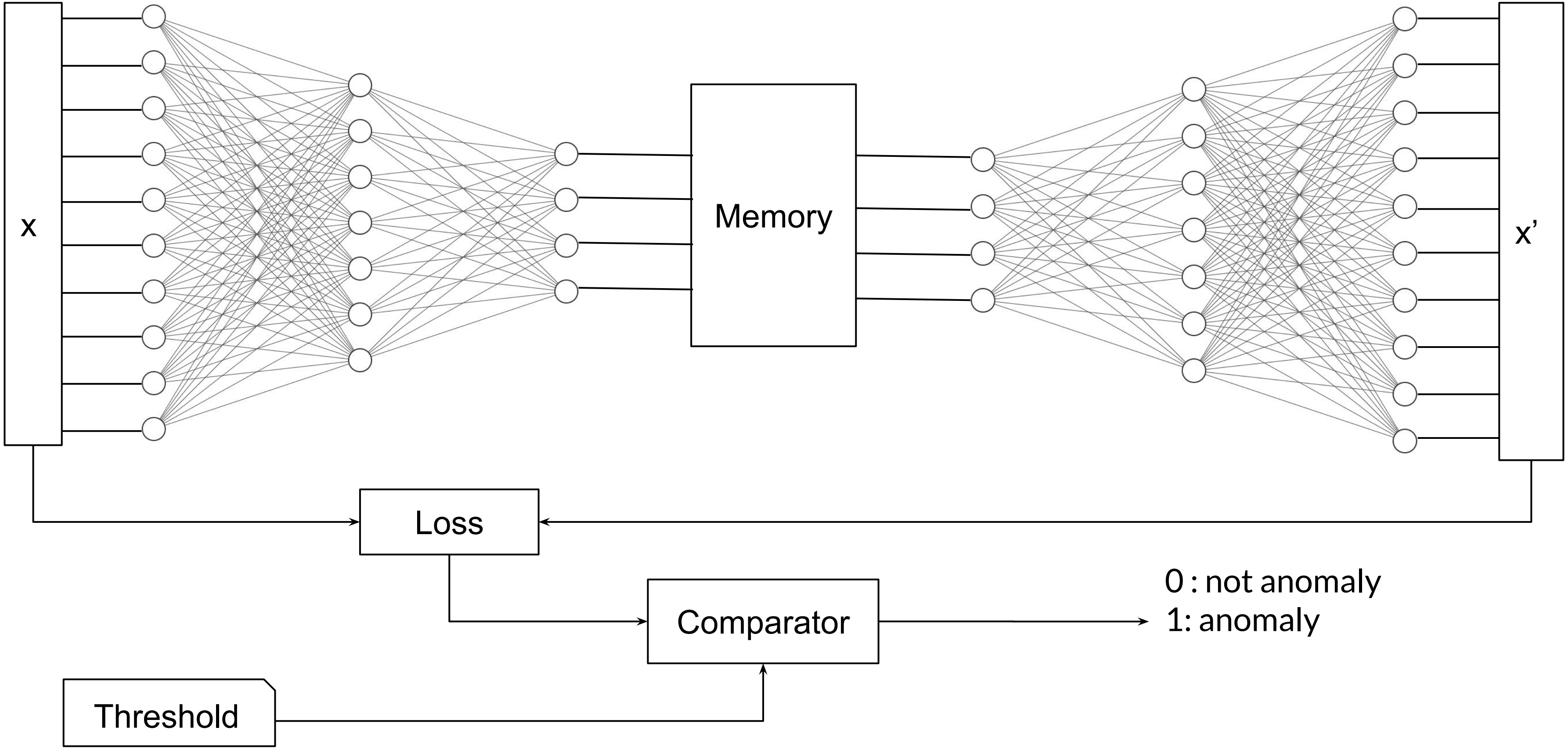
# Problem Description

- Detecting anomalies using autoencoders
- Key Concept:
  - Reconstruction vector of anomalies is different from the original vector
  - Reconstruction vector of non anomalies is close to the original vector
- Hyperparameters:
  - Memory Size
  - Optimizer
  - Comparator threshold

# Related Work

- Normal AutoEncoders
- Variational AutoEncoders
- DevNet (deviation loss function)
- Memory Augmented AutoEncoders (MemAE)

- Selection:
    - MemAE + DevNet loss function



x
Memory
x’
Loss
Comparator
Threshold
0 : not anomaly
1: anomaly

# Timeline

## Tasks:

- Simple AE Model Building
- Data Preprocessing
- Building Memory Layer
- Training + Tuning Memory to minimize MSE
- Investigate Deviation Loss function
- Freezing weights and building Deviation Loss Model
- Training on Deviation Loss
- Running on test data and submit on Kaggle

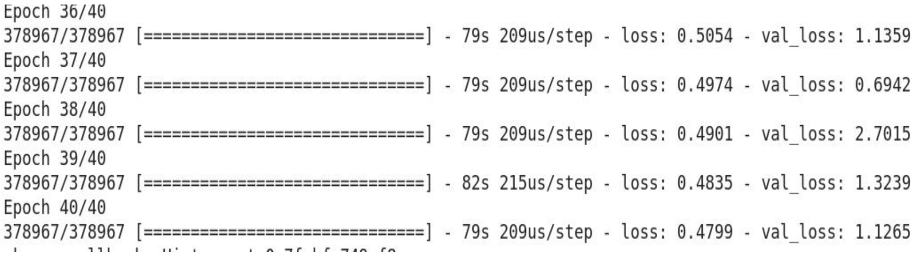
# Progress

| Task | Progress |
|---|---|
| Simple AE Model Building | 100% |
| Data Preprocessing | 100% |
| Building Memory Layer | 100% |
| Training + Tuning Memory to minimize MSE | 50% |
| Investigate Deviation Loss function | 0% |
| Freezing weights and building Deviation Loss Model | 0% |
| Training on Deviation Loss | 0% |
| Running on test data and submit on Kaggle | 0% |

# Results

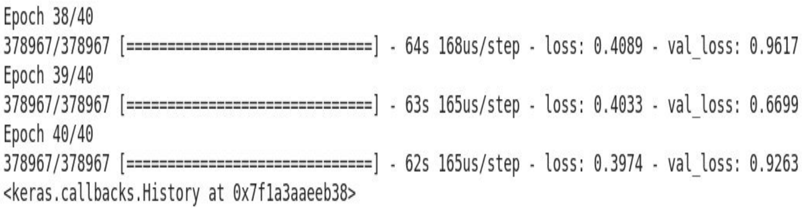
# Best Losses

Memory size = 40



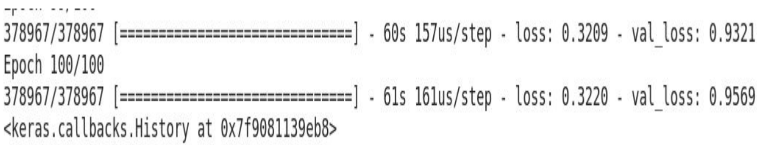
```
Epoch 36/40
378967/378967 [==============================] - 79s 209us/step - loss: 0.5054 - val_loss: 1.1359
Epoch 37/40
378967/378967 [==============================] - 79s 209us/step - loss: 0.4974 - val_loss: 0.6942
Epoch 38/40
378967/378967 [==============================] - 79s 209us/step - loss: 0.4901 - val_loss: 2.7015
Epoch 39/40
378967/378967 [==============================] - 82s 215us/step - loss: 0.4835 - val_loss: 1.3239
Epoch 40/40
378967/378967 [==============================] - 79s 209us/step - loss: 0.4799 - val_loss: 1.1265
```

Memory size = 20



```
Epoch 38/40
378967/378967 [==============================] - 64s 168us/step - loss: 0.4089 - val_loss: 0.9617
Epoch 39/40
378967/378967 [==============================] - 63s 165us/step - loss: 0.4033 - val_loss: 0.6699
Epoch 40/40
378967/378967 [==============================] - 62s 165us/step - loss: 0.3974 - val_loss: 0.9263
<keras.callbacks.History at 0x7f1a3aaeeb38>
```

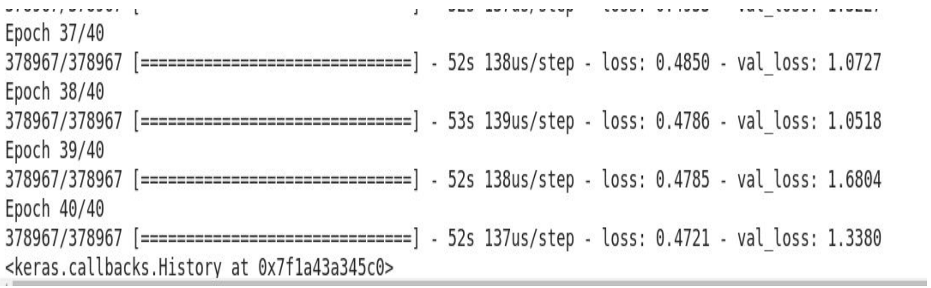
Memory size = 10



```
378967/378967 [==============================] - 60s 157us/step - loss: 0.3209 - val_loss: 0.9321
Epoch 100/100
378967/378967 [==============================] - 61s 161us/step - loss: 0.3220 - val_loss: 0.9569
<keras.callbacks.History at 0x7f9081139eb8>
```

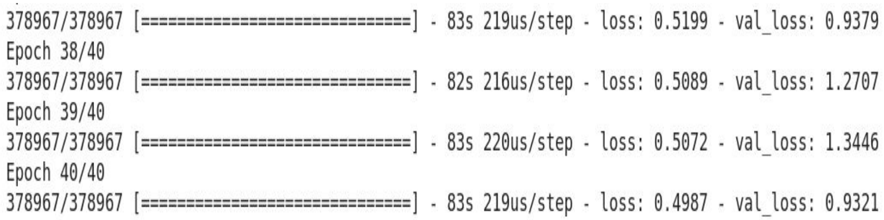
# Best Losses cont.

Memory size = 50



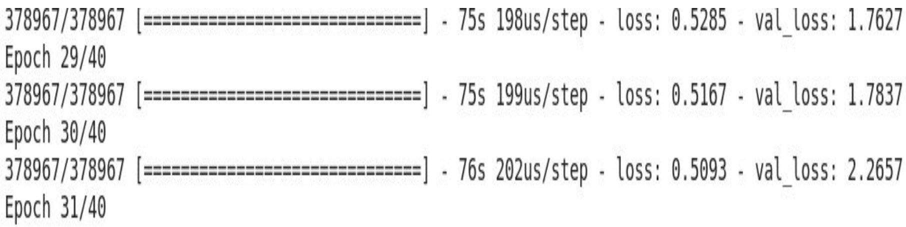
```
Epoch 37/40
378967/378967 [==============================] - 52s 138us/step - loss: 0.4850 - val_loss: 1.0727
Epoch 38/40
378967/378967 [==============================] - 53s 139us/step - loss: 0.4786 - val_loss: 1.0518
Epoch 39/40
378967/378967 [==============================] - 52s 138us/step - loss: 0.4785 - val_loss: 1.6804
Epoch 40/40
378967/378967 [==============================] - 52s 137us/step - loss: 0.4721 - val_loss: 1.3380
<keras.callbacks.History at 0x7f1a43a345c0>
```

Memory size = 100



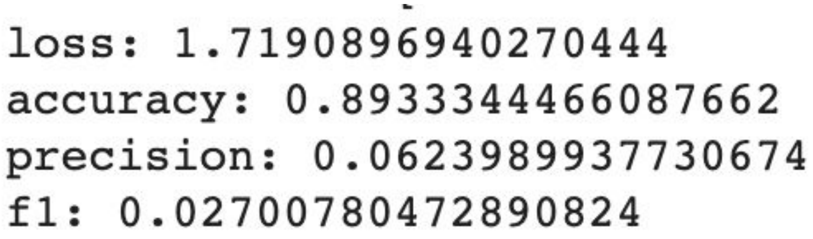
```
378967/378967 [==============================] - 83s 219us/step - loss: 0.5199 - val_loss: 0.9379
Epoch 38/40
378967/378967 [==============================] - 82s 216us/step - loss: 0.5089 - val_loss: 1.2707
Epoch 39/40
378967/378967 [==============================] - 83s 220us/step - loss: 0.5072 - val_loss: 1.3446
Epoch 40/40
378967/378967 [==============================] - 83s 219us/step - loss: 0.4987 - val_loss: 0.9321
```

Memory size = 200



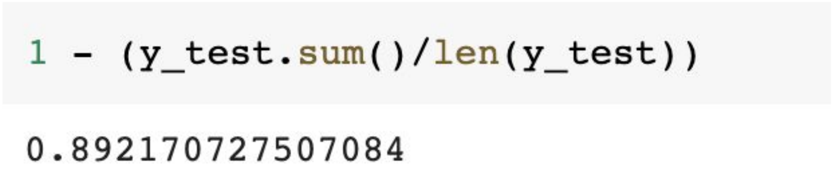
```
378967/378967 [==============================] - 75s 198us/step - loss: 0.5285 - val_loss: 1.7627
Epoch 29/40
378967/378967 [==============================] - 75s 199us/step - loss: 0.5167 - val_loss: 1.7837
Epoch 30/40
378967/378967 [==============================] - 76s 202us/step - loss: 0.5093 - val_loss: 2.2657
Epoch 31/40
```

# Best Accuracy



```
loss: 1.7190896940270444
accuracy: 0.8933344466087662
precision: 0.0623989937730674
f1: 0.0270078047289082 4
```

Using Adadelta and
Memory size 100



```
1 - (y_test.sum()/len(y_test))
```

```
0.892170727507084
```

Actual ratio of non-anomaly

Conclusion: Model is NOT doing a good job