Recurrent Auto-encoder Model for Sensor Signal Analysis

TIMOTHY WONG

SENIOR DATA SCIENTIST, CENTRICA PLC

About Us

We supply energy and services to over 27 million customer accounts

Supported by around 12,000 engineers and technicians

Our areas of focus are Energy Supply & Services, Connected Home, Distributed Energy & Power, Energy Marketing & Trading

Bord Gáis Energy

British Gas

centrica

Direct Energy

DYNO

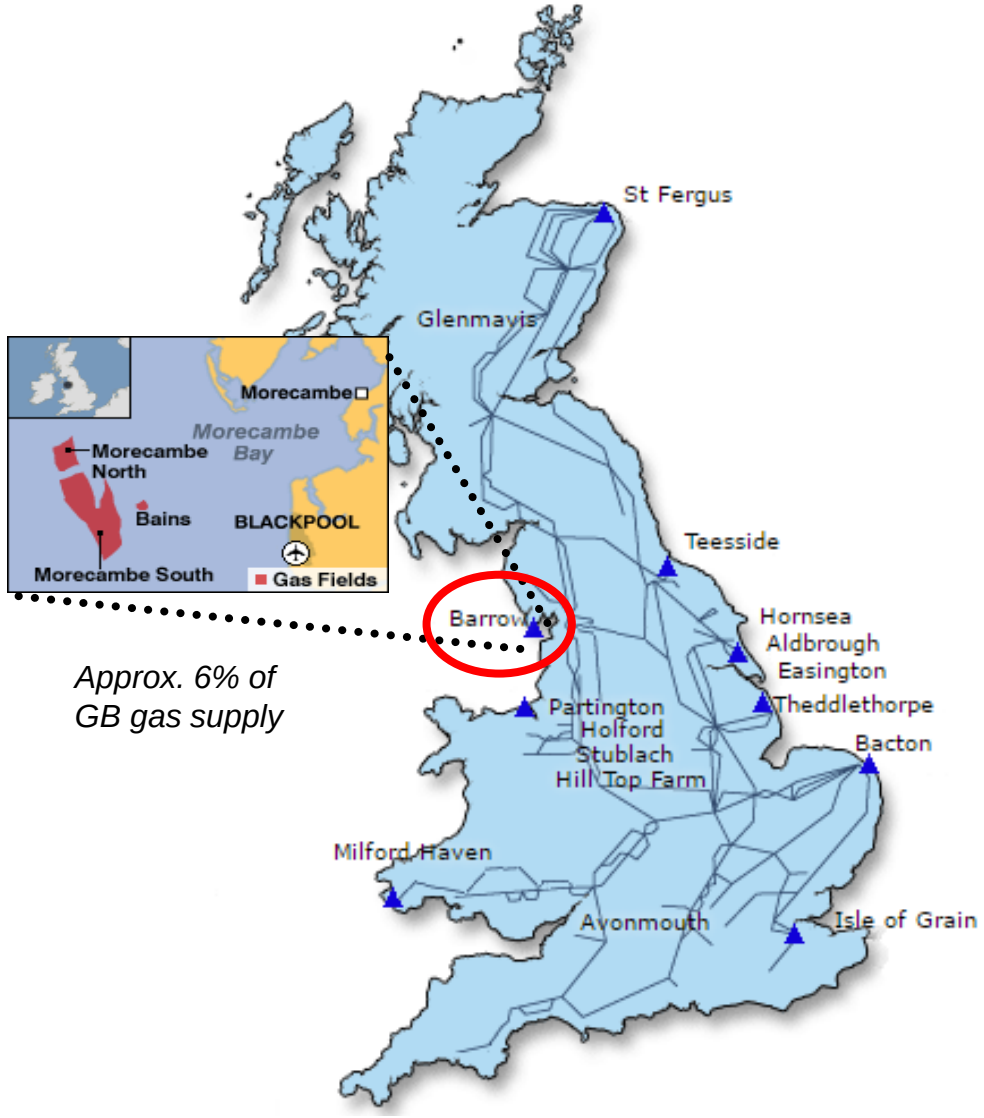
HIVE

Local Heroes

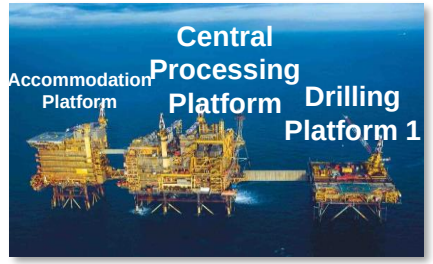
Gas Terminal

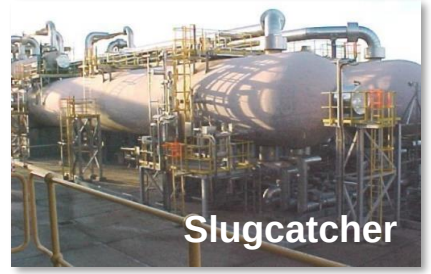
East Irish Sea

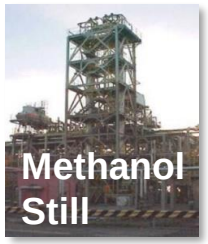
- Morecambe (Barrow) gas terminal
- Produces roughly 6% of GB gas supply
- Onshore and offshore operations
- Comprised of many subsystems



Approx. 6% of GB gas supply



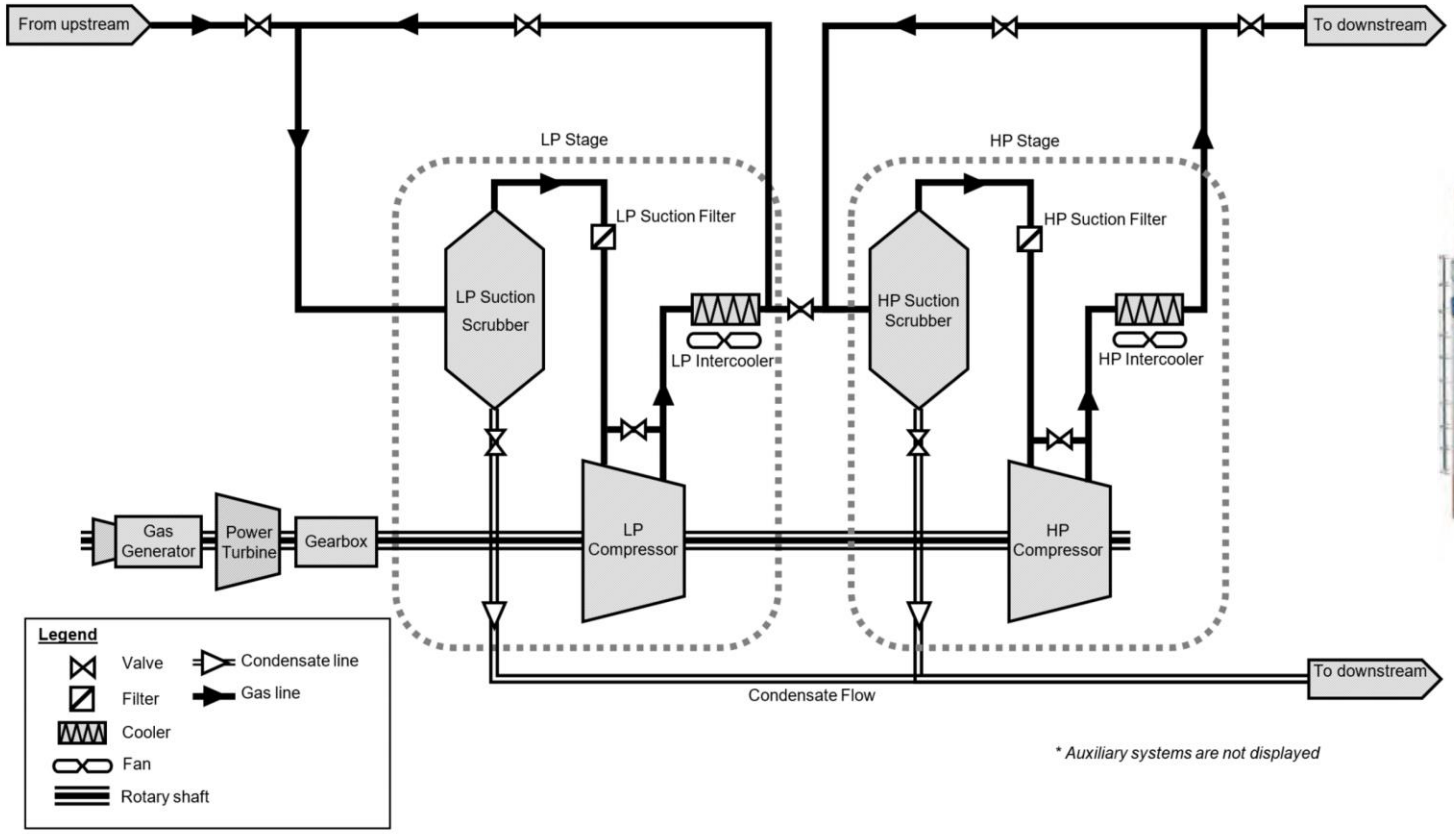


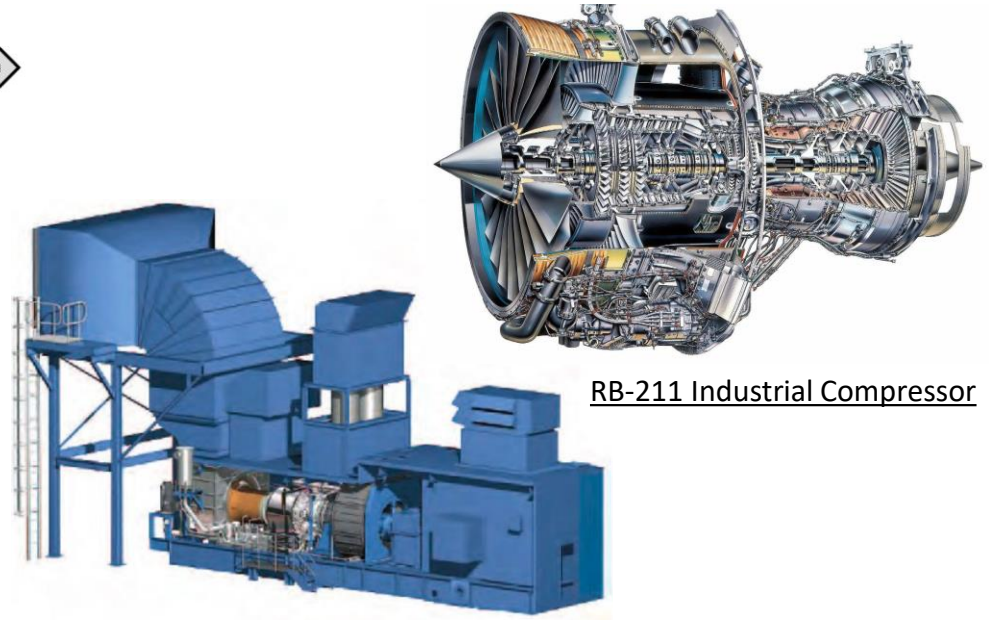


Impeller

Suction scrubber

Compression Sub-system

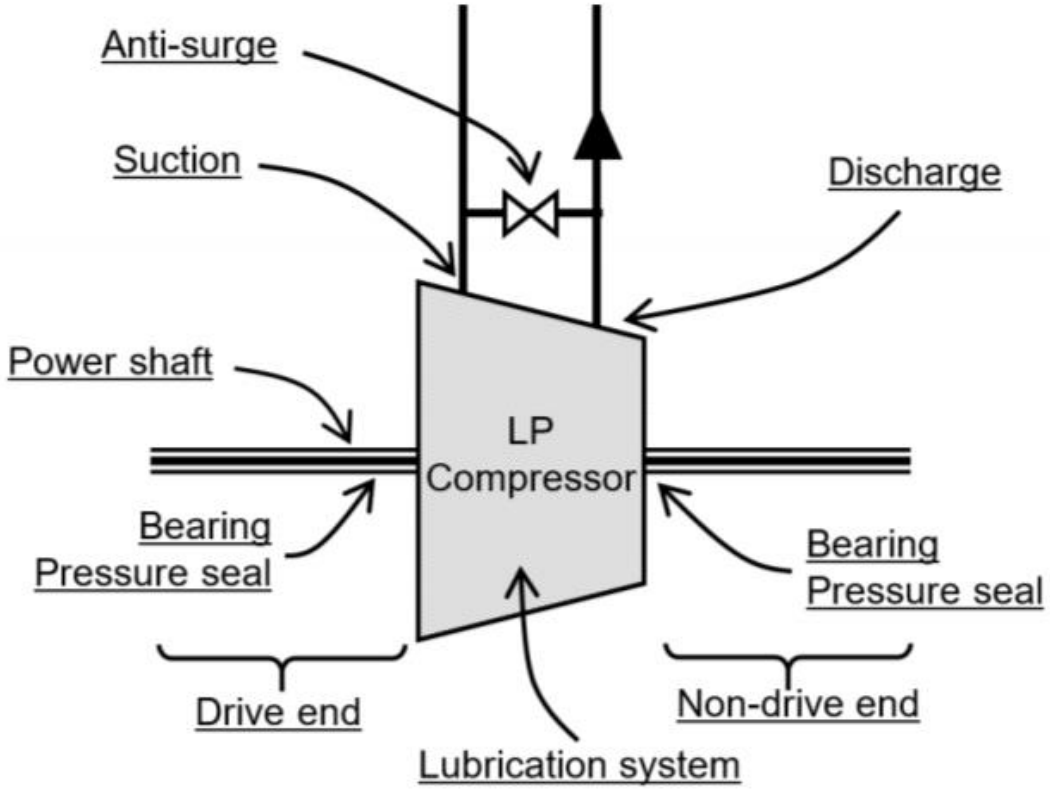




RB-211 Industrial Compressor

- Two centrifugal compressors driven on a single shaft
- Powered by aeroderivative gas turbine
- Increases and regulate gas pressure at a certain level

Compressor


Anti-surge
Suction
Discharge
Power shaft
LP
Compressor
Bearing
Pressure seal
Bearing
Pressure seal
Drive end
Non-drive end
Lubrication system

Problem

Large-scale industrial processes contains many sensors

- Sensor records continuous stream of real-valued measurements (*e.g. temperature, pressure, rotary speed… etc.*)

Can we determine the underlying operating states of the process?



Data Preprocessing

Real-valued measurements recorded by sensor are just time series data (Unbounded & unlabelled)

- Unevenly-spaced time series (inconsistent interval)
- Can be convert into regularly-spaced time series through downsampling.
- P sensors can form a P-dimensional multivariate time series:

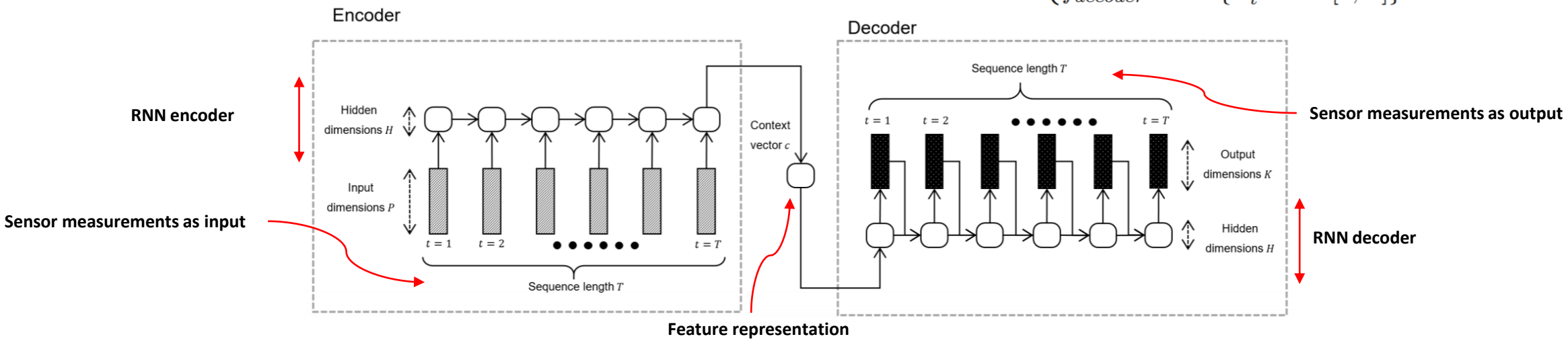
$$\{\mathbb{R}_t^P : t \in [1, T]\}$$

Recurrent Auto-encoder

RNN Encoder-decoder model (‘Seq2seq’, 2014)

- Can be converted into recurrent auto-encoder by aligning input/output time steps
- Performs partial reconstruction on the decoder side *(Decoder dims smaller than Encoder dims)*
- Deep structure learns abstract temporal patterns
  - (Train parameters by gradient descent. Choice of optimiser is important!)

$$\begin{cases} f_{encoder} : \{\mathbb{R}_t^P : t \in [1,T]\} \rightarrow c \\ f_{decoder} : c \rightarrow \{\mathbb{R}_t^K : t \in [1,T]\} \end{cases} \quad K \leqslant P$$



Long-short term memory (LSTM, 1997)

Forget gate:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

Input gate:

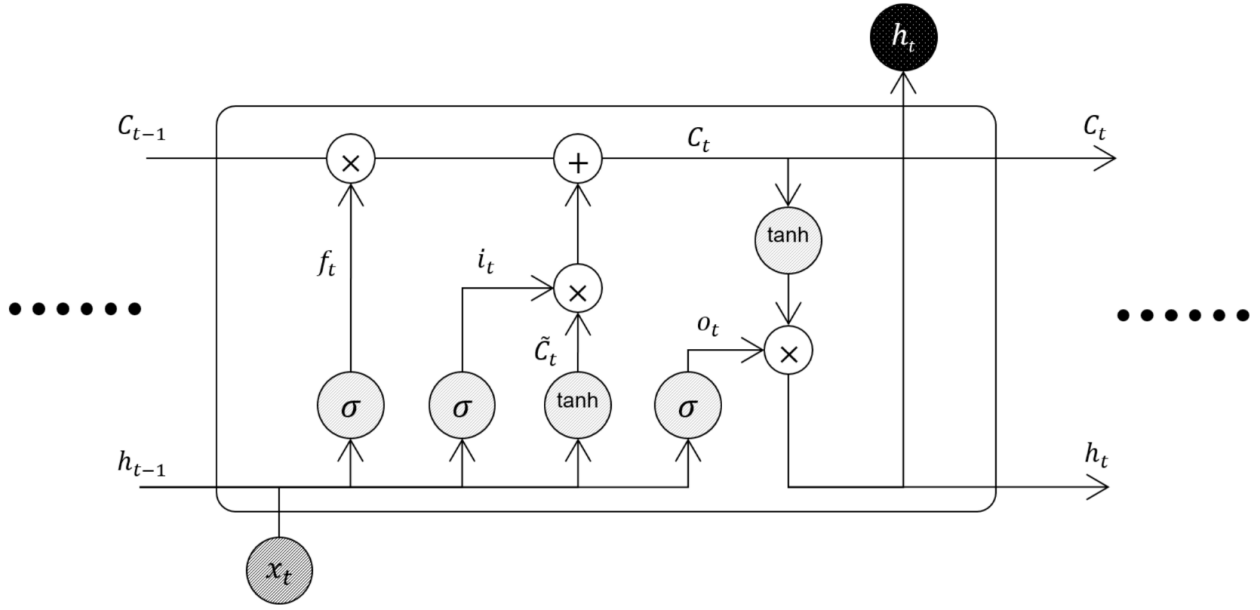
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = tanh(W_c[h_{t-1}, x_t] + b_C)$$

Update recurrent state:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$$

Output gate:

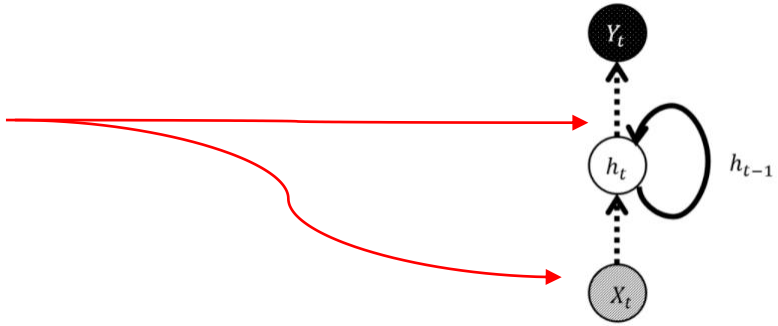
$$O_t = \sigma(W_i[h_{t-1}, x_t] + b_o)$$
$$h_t = O_t \times tanhC_t$$



Regularisation

Prevents network overfitting

- Dropout wrapper (2014)
  - Masks neuron inputs
  - Non-recurrent connections only
- L1 / L2 regularisation



Sampling

Samples of length T can be generated from any dataset with total size $T' \geq T$

- Generate sequences with fixed length T
- Samples can be generated recursively by shifting time step by fixed number of unit
- $T' - T$ samples can be generated

Algorithm 1: Drawing samples consecutively from the original dataset

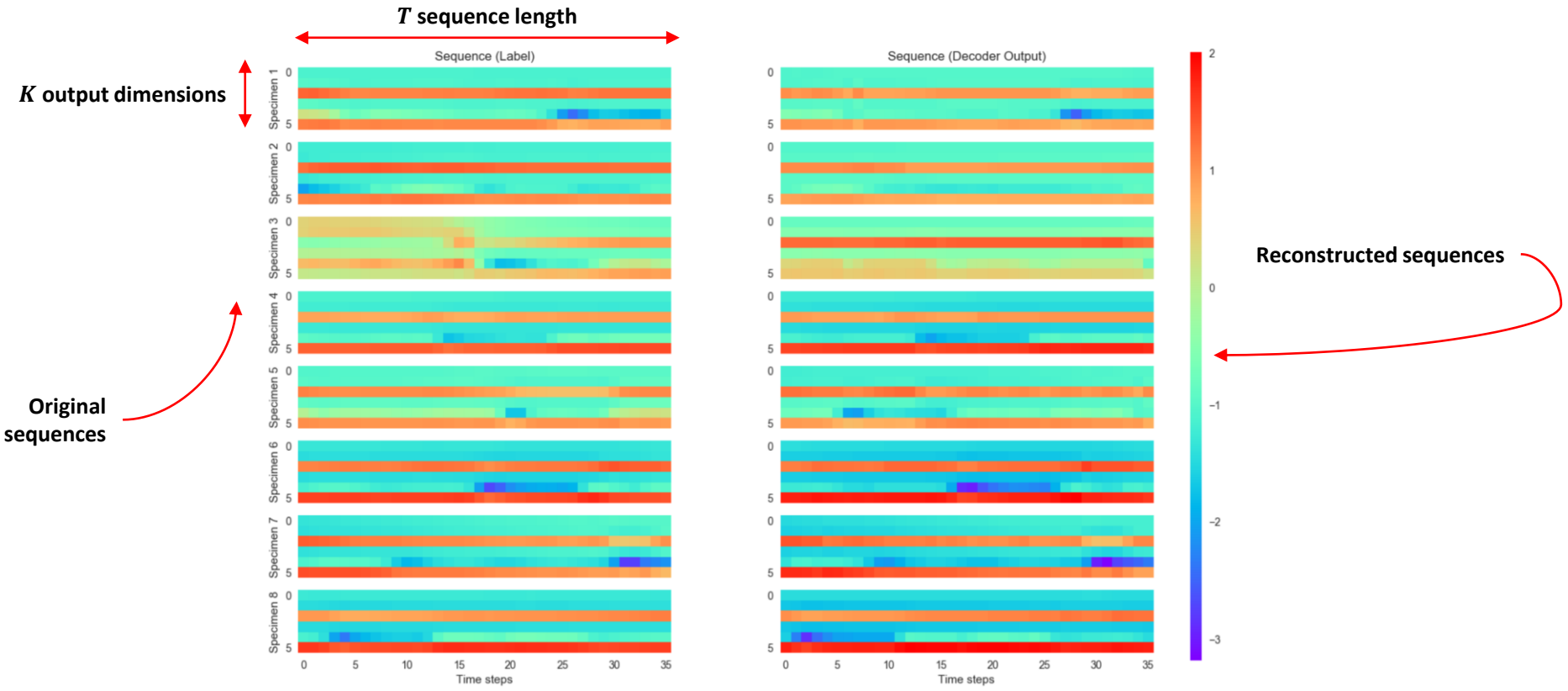
Input: Dataset length T'
Input: Sample length T
$i \leftarrow 0$;
while $i \leqslant i + T$ **do**
  Generate sample sequence $(i, i + T]$ from the dataset;
  $i \leftarrow i + 1$;
end

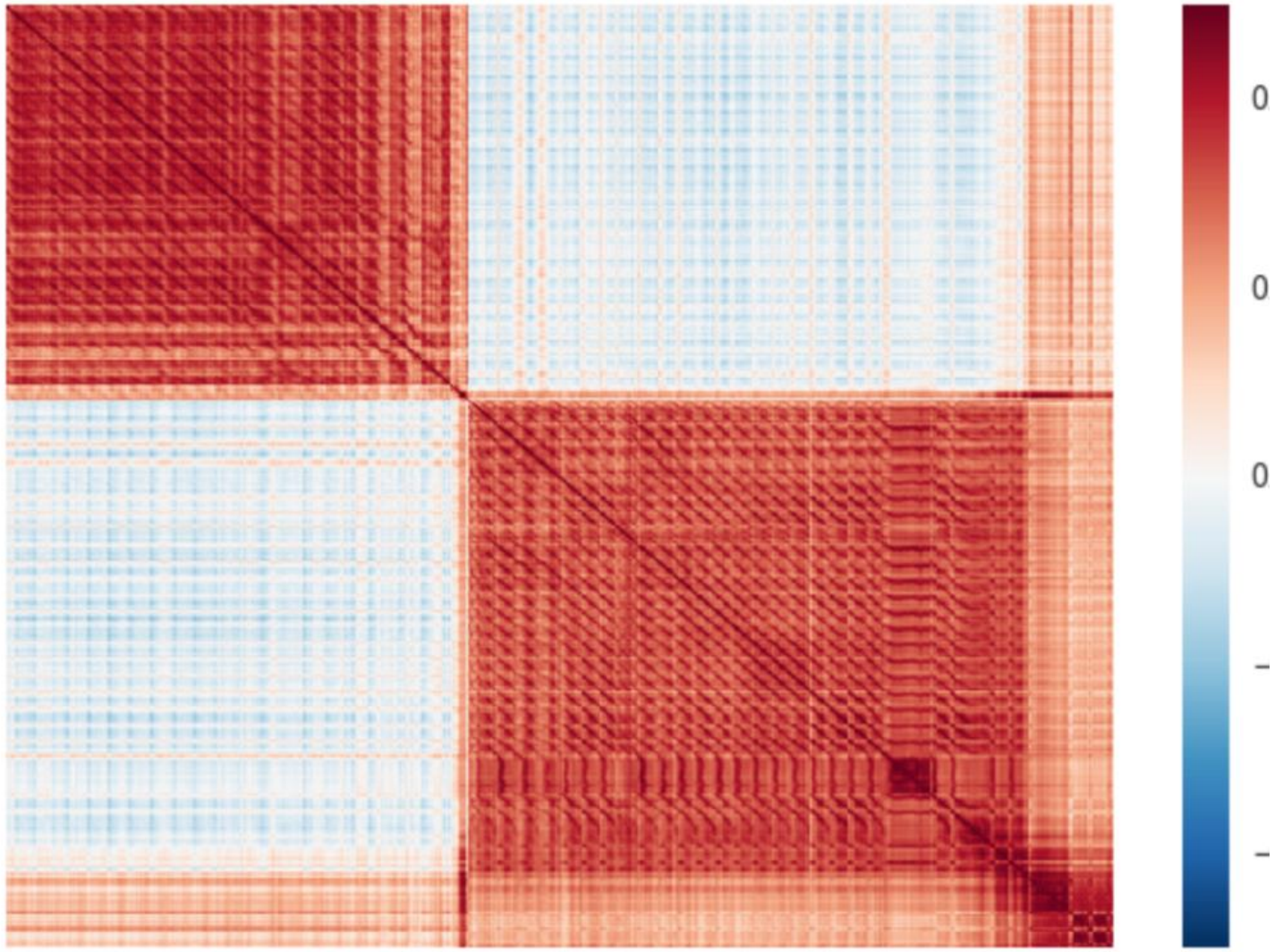
Standardising dataset using z-score

$$z_p = \frac{x_p - \bar{x}_p}{\sigma_p}$$

Sequence Reconstruction

Specimens were selected randomly for qualitative appreciation





Context vector

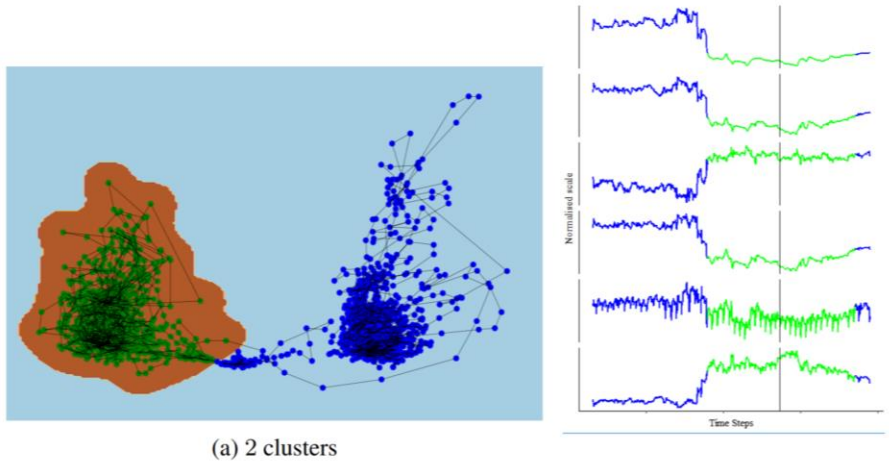
Pairwise Pearson correlation

- Diagonal shows *strong correlation* among successive context vectors
- i.e. Operating state drifting gradually

Context Vectors (Dimensional Reduction)

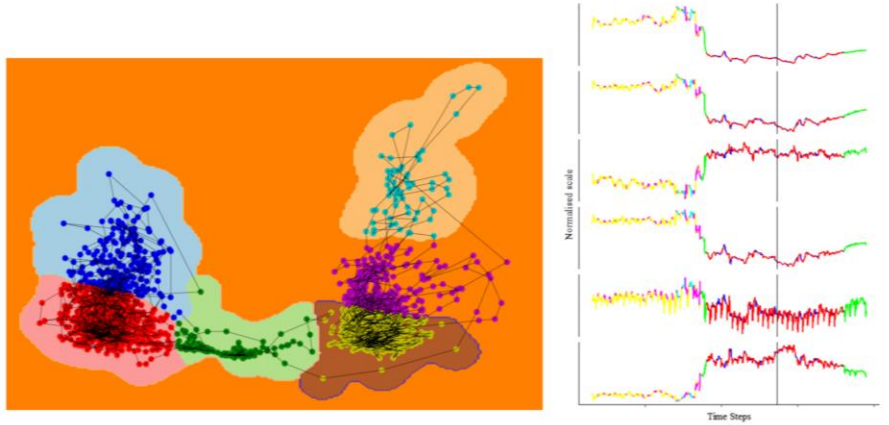
Dimensionality reduction (PCA) to 2D

- Additional clustering algorithms can be applied (e.g. K-means)
- Decision boundary calculated using Support Vector Machine (+RBF)
- Can be applied to on-line setting for operational state recognition
  - e.g. `IF cluster1 → cluster2 THEN TRIGGER ALERT`



(a) 2 clusters

Neighbourhoods reflect similar operating states

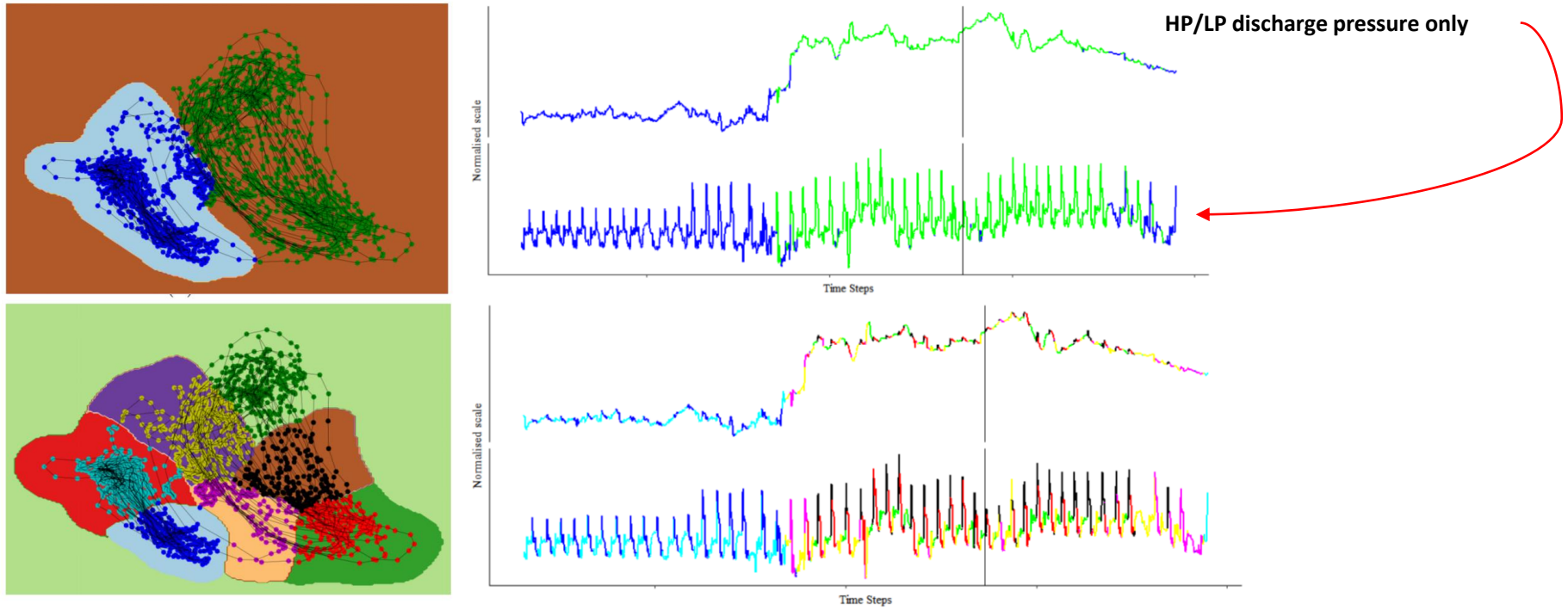


(b) 6 clusters

Context Vectors (Dimensional Reduction)

Alternative example

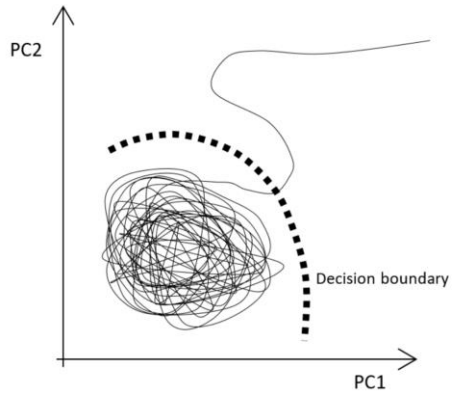
- Drifting context vector (i.e. the operating state)



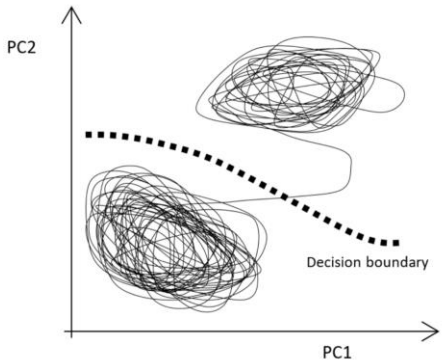
Summary

Unsupervised clustering method for multidimensional time series

- Neural network-based time series feature selection
- Works with any real-valued temporal measurement (e.g. temperature, pressure… etc.)
- Identifies underlying operating states
  - Drifts around same neighbourhood → No change
  - Drift across boundary → changed



(a) Process with a single healthy state.



(b) Multi-state process with transition between states.

```
@article{anonymous,
       title={Recurrent Auto-Encoder Model for Multidimensional Time Series Representation},
       author={Anonymous},
       journal={International Conference on Learning Representations},
       year={2018},
       url={https://openreview.net/forum?id=r1cLblgCZ}
}
```

The technical method described in this presentation is the subject of British patent application GB1717651.2.

Tools Used

python™

TensorFlow

spyder

NVIDIA CUDA

cuDNN

Keras

scikit learn

Q&A

BRISTOL DATA SCIENTISTS

Timothy Wong

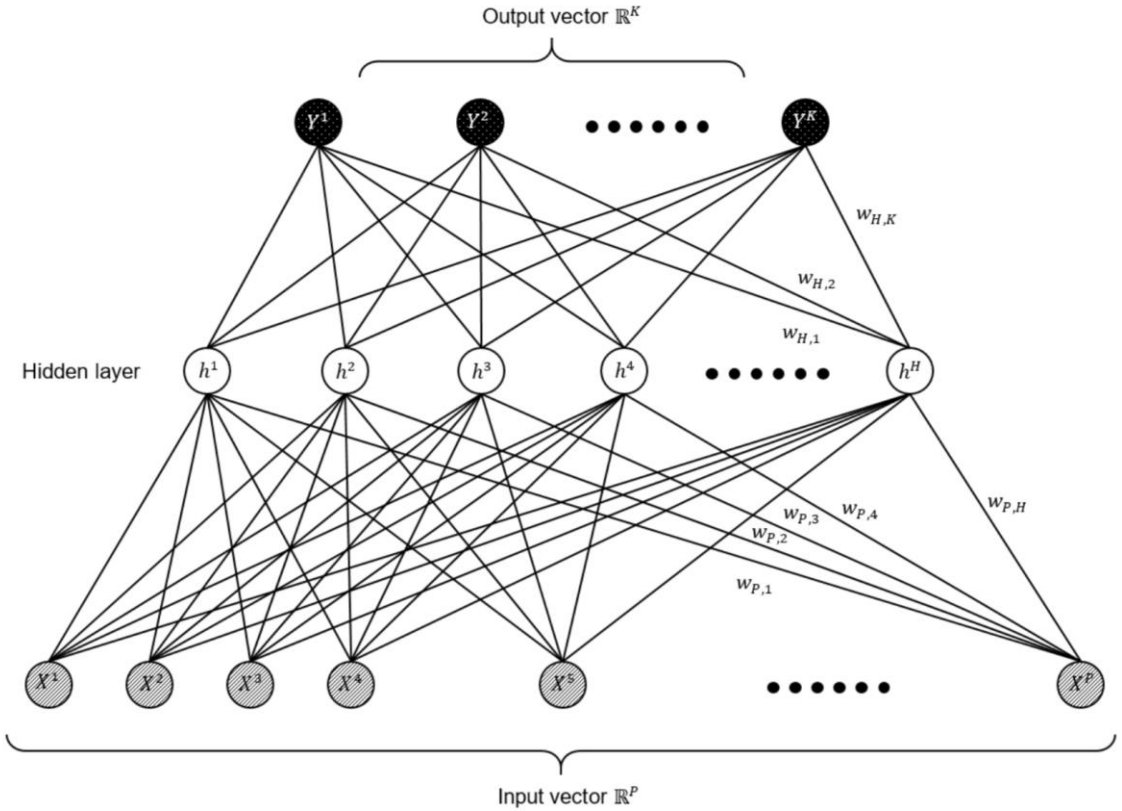
Senior Data Scientist (Centrica plc)

timothy.wong@hotmail.co.uk

Bord Gáis Energy · British Gas · centrica

Direct Energy · DYNO · HIVE · Local Heroes

Neural Nets

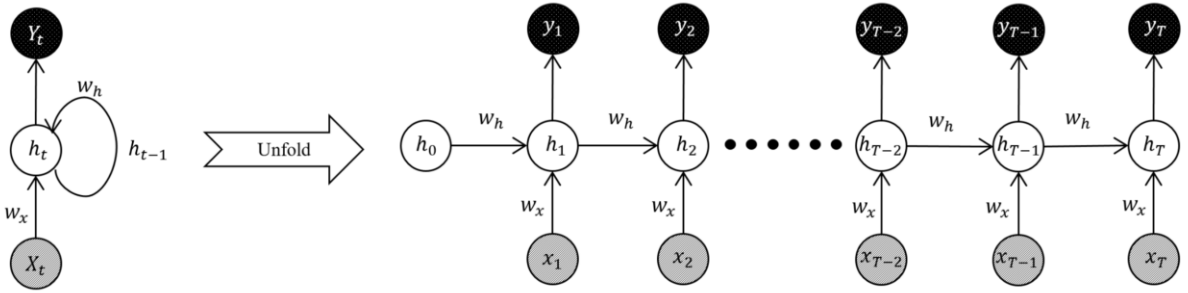

Output vector $\mathbb{R}^K$
Y^1
Y^2
Y^K
$w_{H,K}$
$w_{H,2}$
$w_{H,1}$
Hidden layer
h^1
h^2
h^3
h^4
h^H
$w_{P,H}$
$w_{P,4}$
$w_{P,3}$
$w_{P,2}$
$w_{P,1}$
X^1
X^2
X^3
X^4
X^5
X^P
Input vector $\mathbb{R}^P$

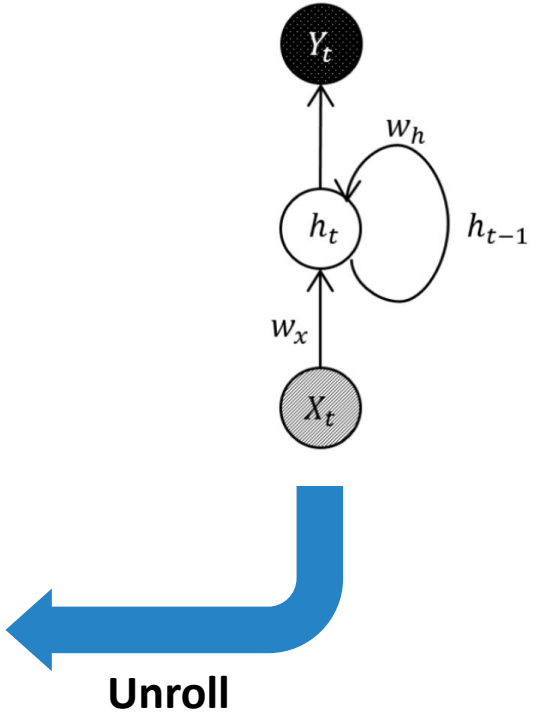
Recurrent Neural Nets

- Powerful tool for handling temporal data
- Learns past information in a given sequence

$$h_t = f(h_{t-1}, x_t)$$

- Back-propagation Through Time (BPTT)
  - RNNs can be unrolled into deep forward-feeding network (FNN)
  - Yet prone to vanishing gradient problem





Non-linear Activation Functions

Sigmoid activation

$$f(x) = \frac{1}{1 + e^{-x}}$$

Hyperbolic tangent activation

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Softplus activation

$$f(x) = \ln(1 + e^{x})$$

Rectified linear unit (ReLU)

$$f(x) = \max(0, x)$$

Leaky ReLU

$$f(x) = \begin{cases} 0.001x \\ x \end{cases}$$

Parametrised ReLU (PReLU)

$$f(x) = \begin{cases} ax \\ x \end{cases}$$

Commonly-used activation functions

SGD Optimiser

Evaluate gradient

$$\nabla \mathcal{L}_w = \lim_{h \to 0} \frac{f(w+h) - f(w)}{h}$$

Update gradient

$$w' = w - \mu \nabla \mathcal{L}_w$$

Algorithm 2: Batch gradient descent

Input: Sample size N

```
while true do
    Evaluate gradient with entire sample N;
    Update gradient;
end
```

Batch SGD

Minibatch SGD

Algorithm 4: Minibatch gradient descent

Input: Sample size N

Input: Minibatch size B

```
while true do
    Shuffle sample N;
    i ← 0
    while i ⩽ N/B do
        Draw a subset (iB, min(N, (i + 1)B)] from the sample;
        Evaluate gradient with subset elements;
        Update gradient;
        i ← i + 1
    end
end
```

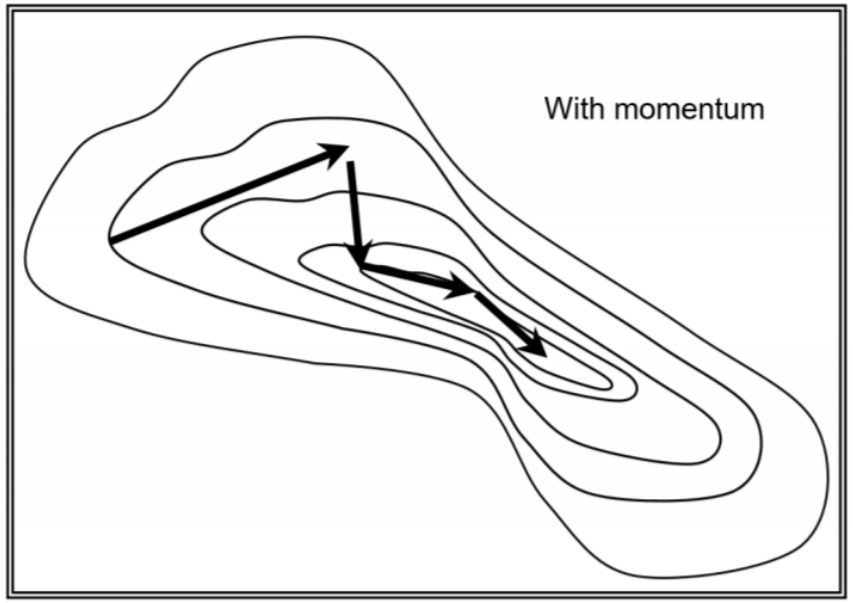
SGD with Momentum

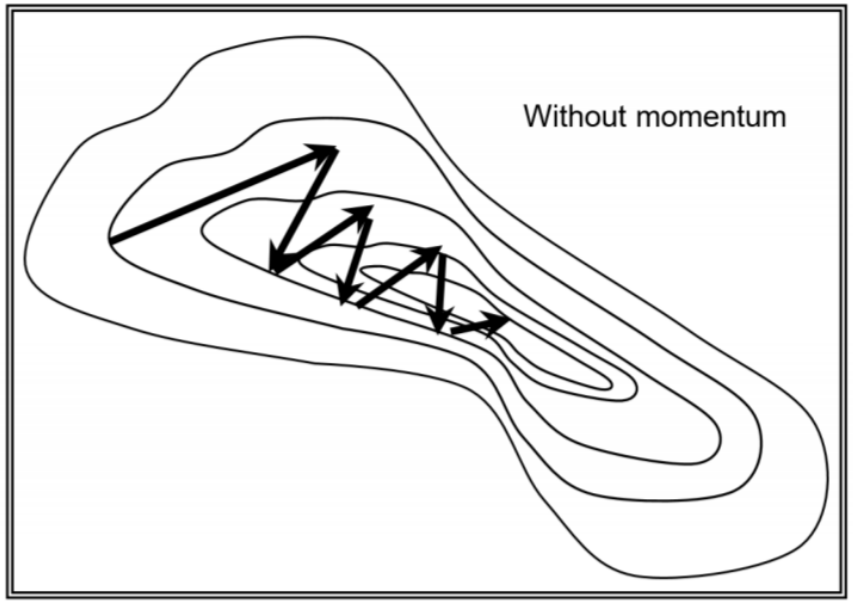
Evaluate gradient

$$v_t = \alpha v_{t-1} + \mu \nabla \mathcal{L}_w$$

Update gradient

$$w' = w - v_t$$





Commonly-used Optimisers

Adagrad

Sum of square term

$$g_t = \sum_{t'=1}^{t} (\nabla \mathcal{L}_{w,t'})^2$$

Update gradient

$$w' = w - \frac{\mu}{\sqrt{g_t} + \epsilon} \nabla \mathcal{L}_{w,t}$$

RMSProp

Sum of square term

$$E[g_t] = \rho E[g_{t-1}] + (1 - \rho)(\nabla \mathcal{L}_{w,t})^2$$

Update gradient

$$w' = w - \frac{\mu}{\sqrt{g_t} + \epsilon} \nabla \mathcal{L}_{w,t}$$

Adam

Decaying past gradient

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla \mathcal{L}_{w(t,i)}$$

Decaying past squared gradient

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla \mathcal{L}_{w(t,i)})^2$$

Bias adjustment

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Gradient update

$$w' = w - \frac{\mu}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

Effects of Learning Rate

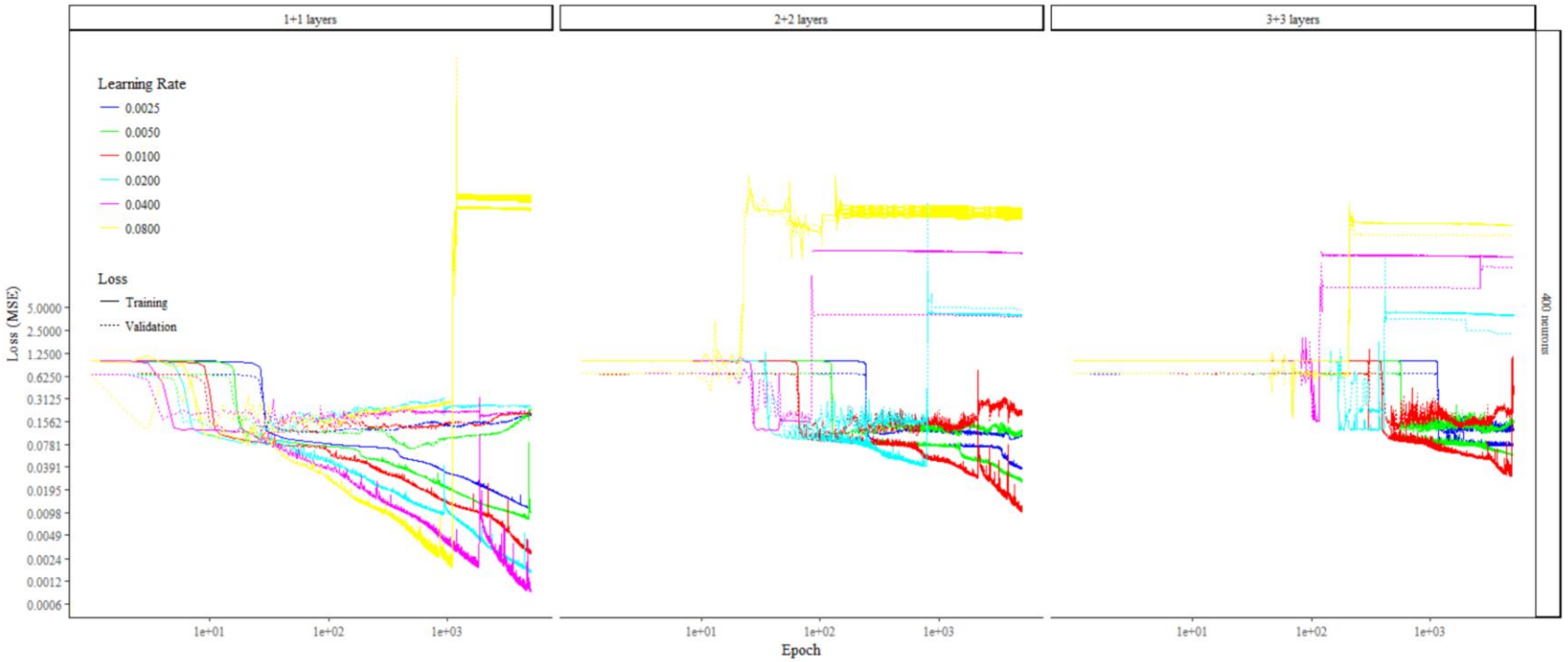


Figure 12: Training and validation losses of minibatch gradient descent optimisers. Three sets of models containing different number of hidden layers were trained. All models have 400) neurons in each hidden layer.

Sequence Reversal

original input sequence $\{\mathbb{R}_1^P, \mathbb{R}_1^P, \mathbb{R}_3^P, ..., \mathbb{R}_{T-1}^P, \mathbb{R}_T^P\}$ → reverse order $\{\mathbb{R}_T^P, \mathbb{R}_{T-1}^P, \mathbb{R}_{T-2}^P, ..., \mathbb{R}_2^P, \mathbb{R}_1^P\}$

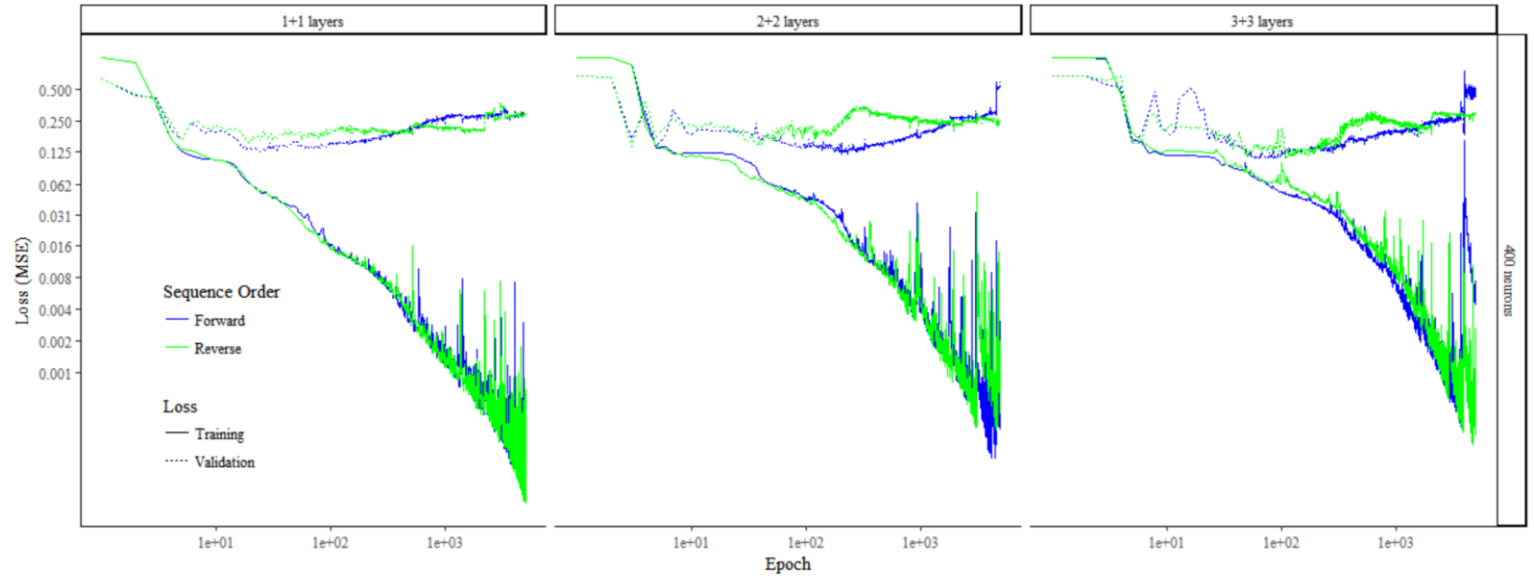


Figure 17: Effects of sequence reversal on the training and validation losses. Three sets of models were trained, each has different number of hidden layer in the encoder-decoder structure. All models have 400 neurons in each hidden layer. Models were trained with Adam optimiser at $B = 256$ with no dropout.

Effects of Sequence Length

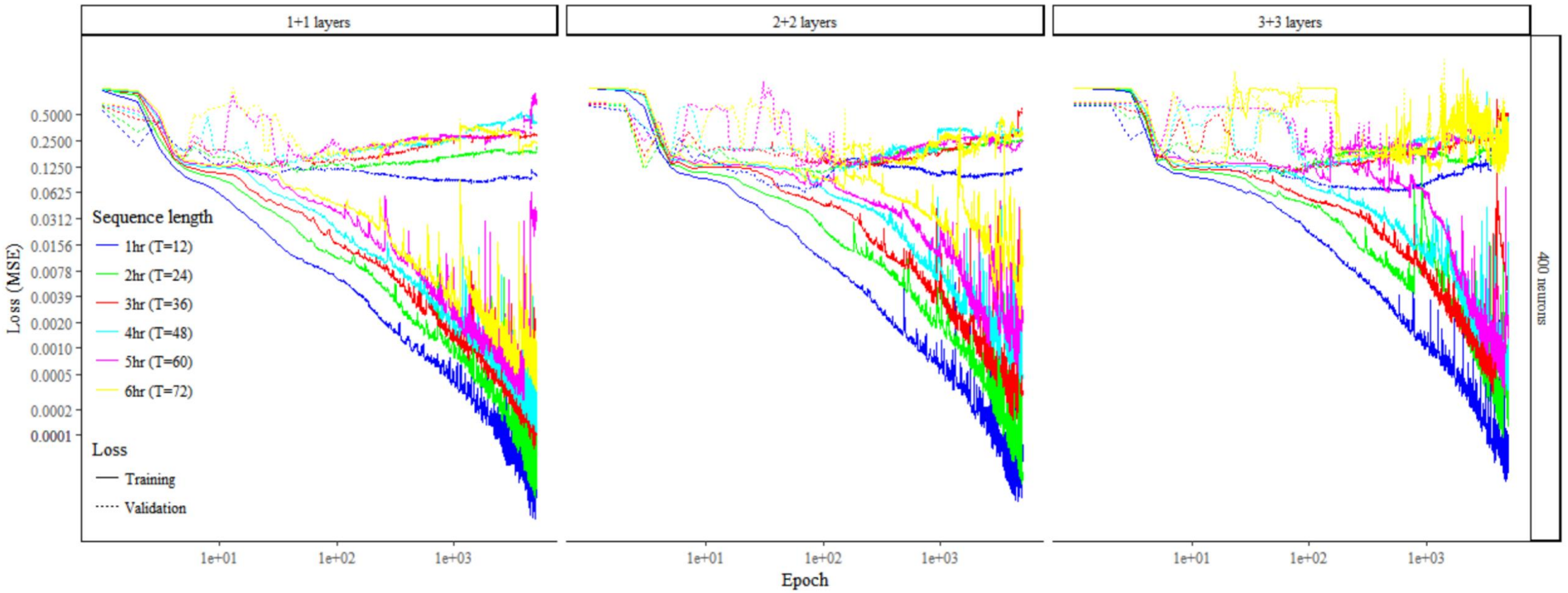


Figure 18: Effect of sequence length on the training and validation MSE loss of three sets of models. All models were trained with Adam optimiser at $B = 256$ with no dropout.

Effects of Various Optimisers

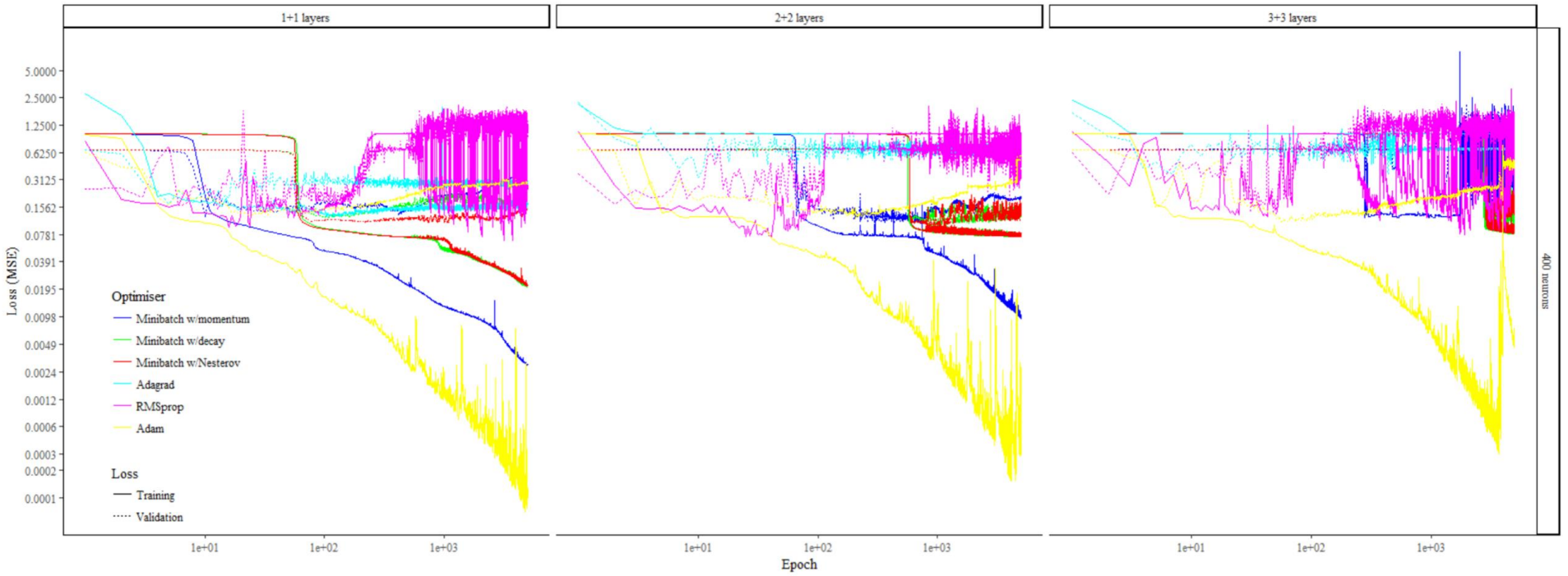


Figure 13: Training and validation losses of various optimisers. All models were trained with same batch size $B = 256$.

Effects of Network Topology

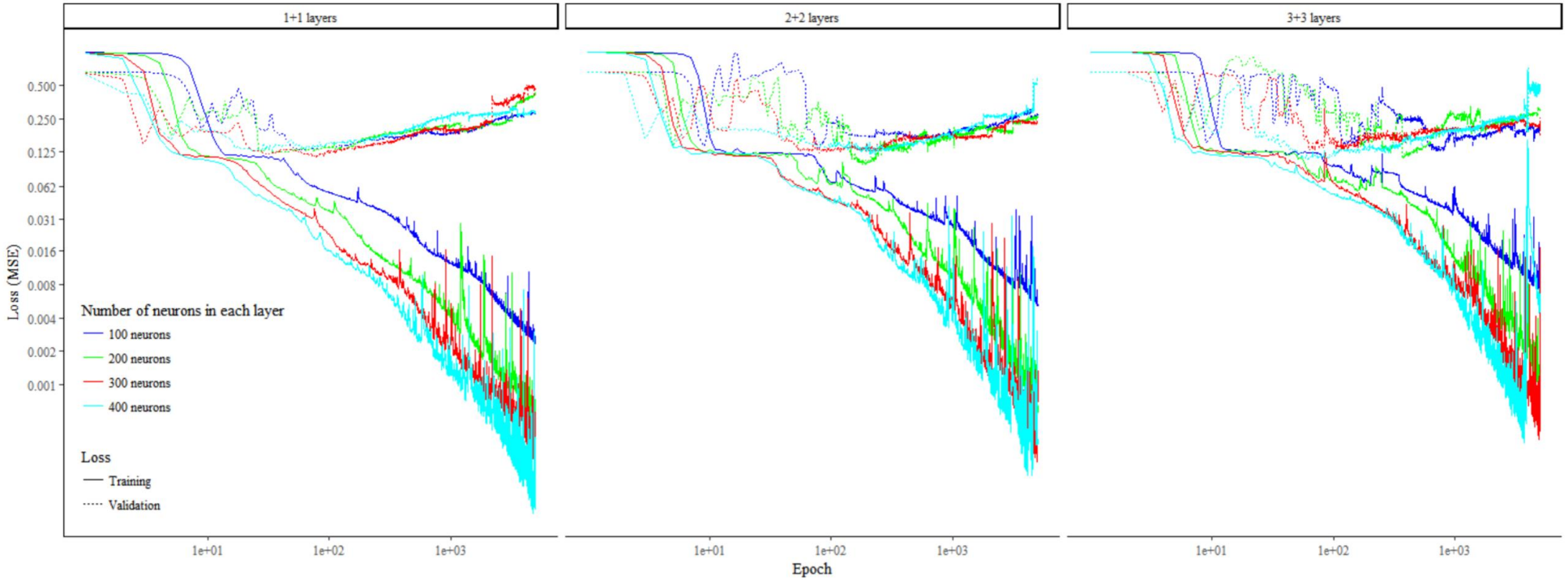


Figure 14: Effect of different number of neurons and hidden layers on training and validation loss. All models were trained with Adam optimiser ($B = 256$).

Effects of Dropout

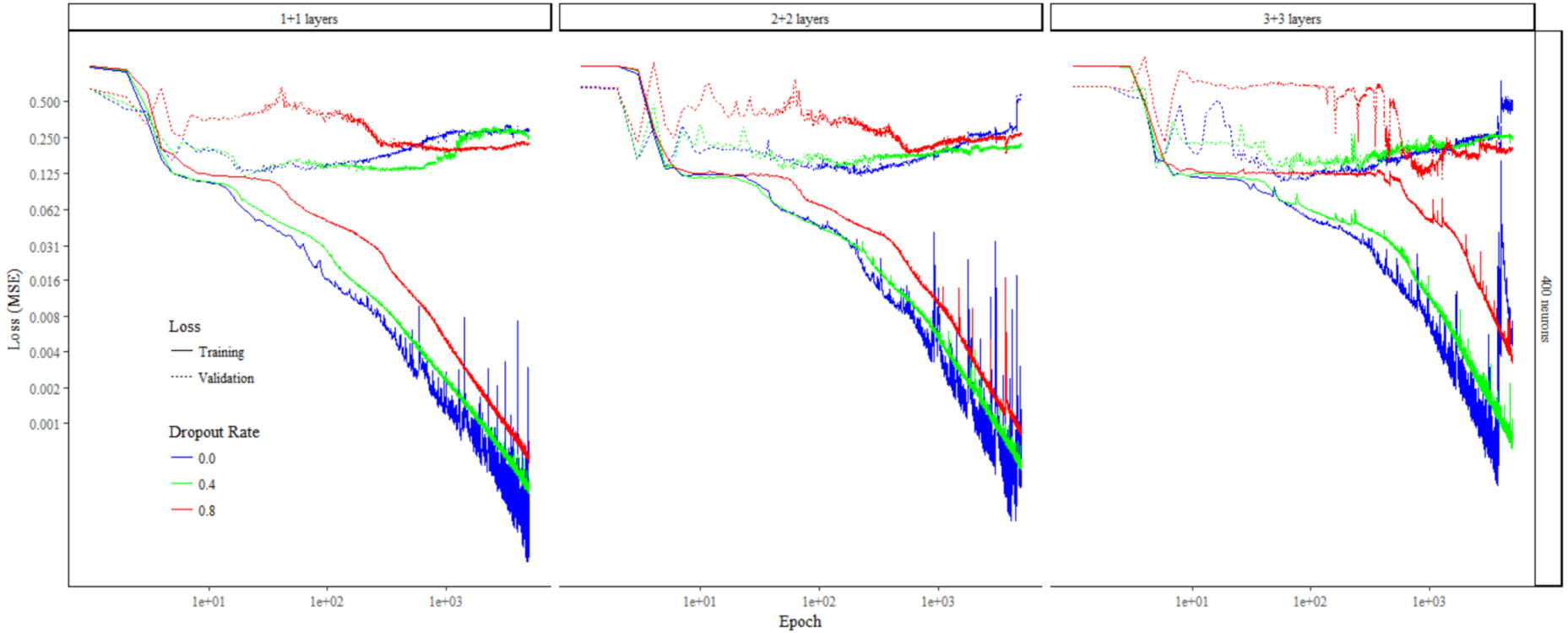


Figure 15: Chart showing the effects of various dropout rate.

Effects of Minibatch size

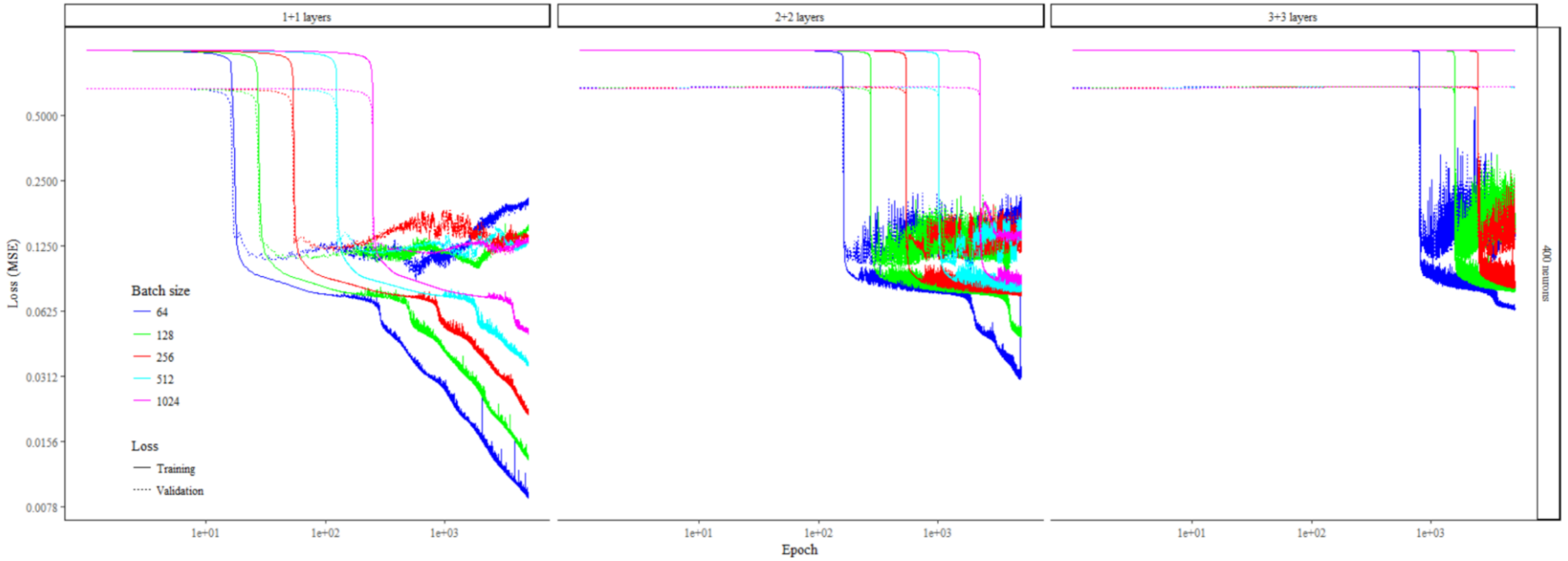


Figure 11: Several sets of auutoencoder models were trained using minibatch gradient descent optimiser with different batch sizes. They contain $1+1$ layers, $2+2$ layers and $3+3$ layers in the encoder-decoder structures respectively. All hidden layers contain 400 neurons. Both x- and y-axes are in logarithmic scale.