Assignment 2
Autoencoder and Transformer Architectures

Part I: Theoretical Part

Part I.I: Autoencoders for Anomaly Detection

Task 1: Calculate the total number of trainable parameters in the autoencoder, including weights and biases. Show your detailed steps for each layer.

The autoencoder consists of multiple fully connected (FC) layers. Each layer has weights and biases.

Input Layer → Hidden Layer 1 (512 neurons):

$$\text{Weights} = 1000 \times 512$$
$$\text{Biases} = 512$$
$$\text{Total parameters} = 1000 \times 512 + 512 = 512512$$

Hidden Layer 1 → Bottleneck Layer (32 neurons):

$$\text{Weights} = 512 \times 32$$
$$\text{Biases} = 32$$
$$\text{Total parameters} = 512 \times 32 + 32 = 16416$$

Bottleneck Layer → Hidden Layer 2 (512 neurons):

$$\text{Weights} = 32 \times 512$$
$$\text{Biases} = 512$$
$$\text{Total parameters} = 32 \times 512 + 512 = 16896$$

Hidden Layer 2 → Output Layer (1000 neurons):

$$\text{Weights} = 512 \times 1000$$
$$\text{Biases} = 1000$$
$$\text{Total parameters} = 512 \times 1000 + 1000 = 513000$$

Total trainable parameters:

$$512512 + 16416 + 16896 + 513000 = 1058824$$

Task 2: Explain how the L2 regularization term will be incorporated into the training process with the MSE loss. Describe its impact on the weight updates during back-propagation.

The L2 regularization term is added to the loss function to prevent overfitting by penalizing large weight values.

MSE Loss Function:

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

where N is the number of samples, $\hat{y}_i$ is the predicted output, and y_i is the actual output.

Regularized Loss with L2 Regularization:

$$\mathcal{L} = \mathcal{L}_{MSE} + \lambda \sum_{l} ||W_l||^2$$

where λ is the regularization strength, and $||W_l||^2$ is the squared sum of the weights.

During backpropagation, the weight updates follow:

$$W_l \leftarrow W_l - \eta\left(\frac{\partial \mathcal{L}_{MSE}}{\partial W_l} + 2\lambda W_l\right)$$

where η is the learning rate. The additional term $2\lambda W_l$ pulls weights toward zero, preventing overfitting.

Impact of L2 Regularization on Weight Updates:

- The term $-2\lambda W_l$ acts like a friction force, pushing the weights toward zero.
- This discourages large weight values, making the model simpler and more generalizable to unseen data.
- It does **not** force weights to exactly zero (unlike L1 regularization, which induces sparsity), but it reduces their magnitude gradually.
- The impact depends on λ:
  - If λ is too small, regularization is weak, and overfitting may still occur.
  - If λ is too large, regularization is too strong, and the model may **underfit** (failing to learn meaningful patterns).

Task 3: Draw a computational graph (examples: 1, 2, 3) for the autoencoder. Automated computational graph generators are not allowed.

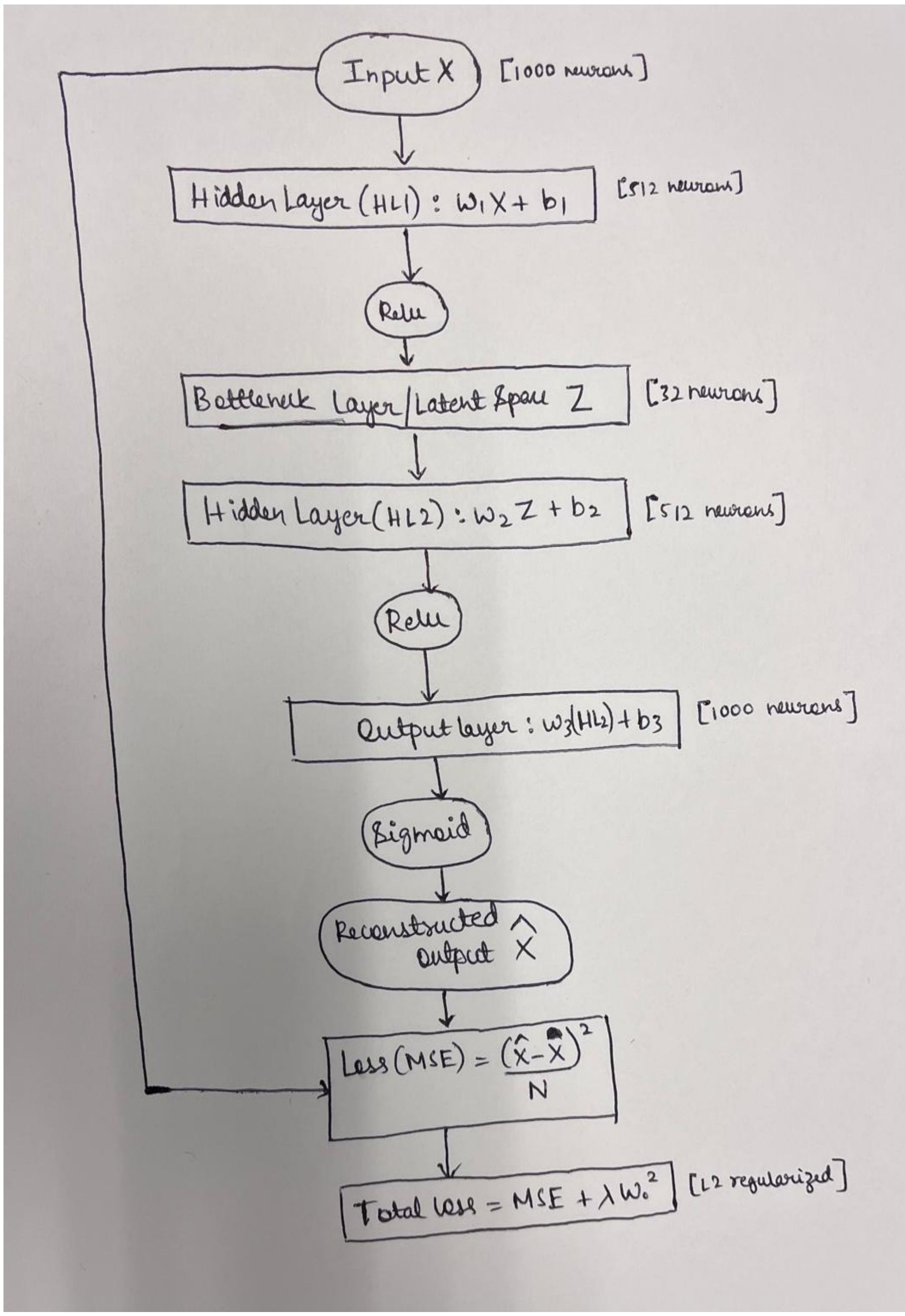


Figure 1: Computational graph of the autoencoder

Task 4: Discuss potential challenges and limitations (at least 4) of using autoencoders for anomaly detection in manufacturing. Consider data, anomaly definition, model assumptions, and deployment.

1. **Defining Anomalies:** Anomalies are often rare, making it difficult to train the model effectively. The model may struggle to generalize if the definition of anomalies is unclear or changes over time.

2. **Data Quality and Preprocessing:** The autoencoder assumes that normal data patterns are well-represented in training. Poor data quality (sensor noise, missing values) can lead to inaccurate reconstructions and false positives.

3. **Overfitting to Normal Data:** The model may learn to reconstruct normal data too well, leading to low reconstruction errors even for slightly anomalous data.

4. **Deployment and Real-time Performance:** Real-time anomaly detection requires low latency and integration with existing systems which can be challenging.

Task 5: Propose potential improvements or extensions (at least 4) to the system to enhance its effectiveness in detecting anomalies. Consider architecture, loss, preprocessing, integration.

1. **Architectural Enhancements:** Using Variational Autoencoders (VAEs) or Convolutional Autoencoders to improve feature extraction.

2. **Improved Loss Function:** Instead of plain MSE, use contrastive loss or weighted loss functions that emphasize anomalies, making the model more sensitive to rare defects.

3. **Preprocessing and Feature Engineering:** Apply dimensionality reduction (PCA, t-SNE) and denoising techniques to remove sensor noise before feeding data into the autoencoder.

4. **Integration with Other Methods:** Combine autoencoders with statistical methods (Z-score, IQR) or machine learning classifiers (SVM, Isolation Forest) to enhance anomaly detection accuracy.

Part I.11 Transformers & Self-Attention

Task 1:

1. Mathematical Operations in Self-Attention:

Given an input sequence,

$$x = (x_1, x_2, \ldots, x_n);$$

the self-attention mechanism proceeds in the following stages:

(a) Linear Transformations

1. Query (q): Queries represent the vector that "asks" about relationships with other elements. Each input x_i vector is mapped to a query vector q_i.

$$q_i = W_q x_i + b_q,$$

where W_q is the query weight matrix and b_q is the corresponding bias vector.

2. Key (k): Keys represent the vector that "answers" the queries, signalling relevance to each element. Each x_i is also mapped to a key vector k_i.

$$k_i = W_k x_i + b_k,$$

where W_k is the key weight matrix and b_k is its bias.

3. Value (v): Values carry the actual content information of each element that will be weighted by attention scores.

$$v_i = W_v x_i + b_v,$$

Each W_v is the value weight matrix and b_v is its bias.

Typically, W_q, W_k and W_v are all (input dim) × d matrices (where d might be the embedding dimension or a choosen "attention dimentions"), and the biases are d-dimensional vectors. These learnable parameter let the model transform each input query, key & value subspaces tailored for attention.

(b) Scaled Dot - Product Attention

Once the queries q_i and keys k_j are computed, the attention score between element i and element j is obtained via the dot product, scaled by $\sqrt{d_k}$ (the square root of k dim) :

$$\text{score}(i,j) = \frac{q_i \cdot k_j}{\sqrt{d_k}}$$

- without scaling, the dot product $q_i \cdot k_j$ could grow large in magnitude as d_k increases. leading to large gradients & making the softmax distribution overly peaked. Dividing by $\sqrt{d_k}$ normalizes these scores, stabalizing training & helping the model distribute attention more effectively.

- Normalization : after computing the raw scores score (i,j), we typically apply a softmax accross j (for each fixed i) to obtain attention weights :

$$a_{ij} = \text{softmax}_j(\text{score}(i,j)) = \frac{\exp(\text{score}(i,j))}{\sum_{j'} \exp(\text{score}(i,j'))}$$

(c) Weighted Sum :

Using the attention weights α_{ij}, we compute a weighted sum of the value vectors V_j. For each position i ;

$$z_i = \sum_{j=1}^{N} \alpha_{ij} V_j$$

Intuitively, the model "looks at" all the values in the sequence (through x_{ij}), but places greater weight on the most relevant elements (i.e, those with higher attention scores).

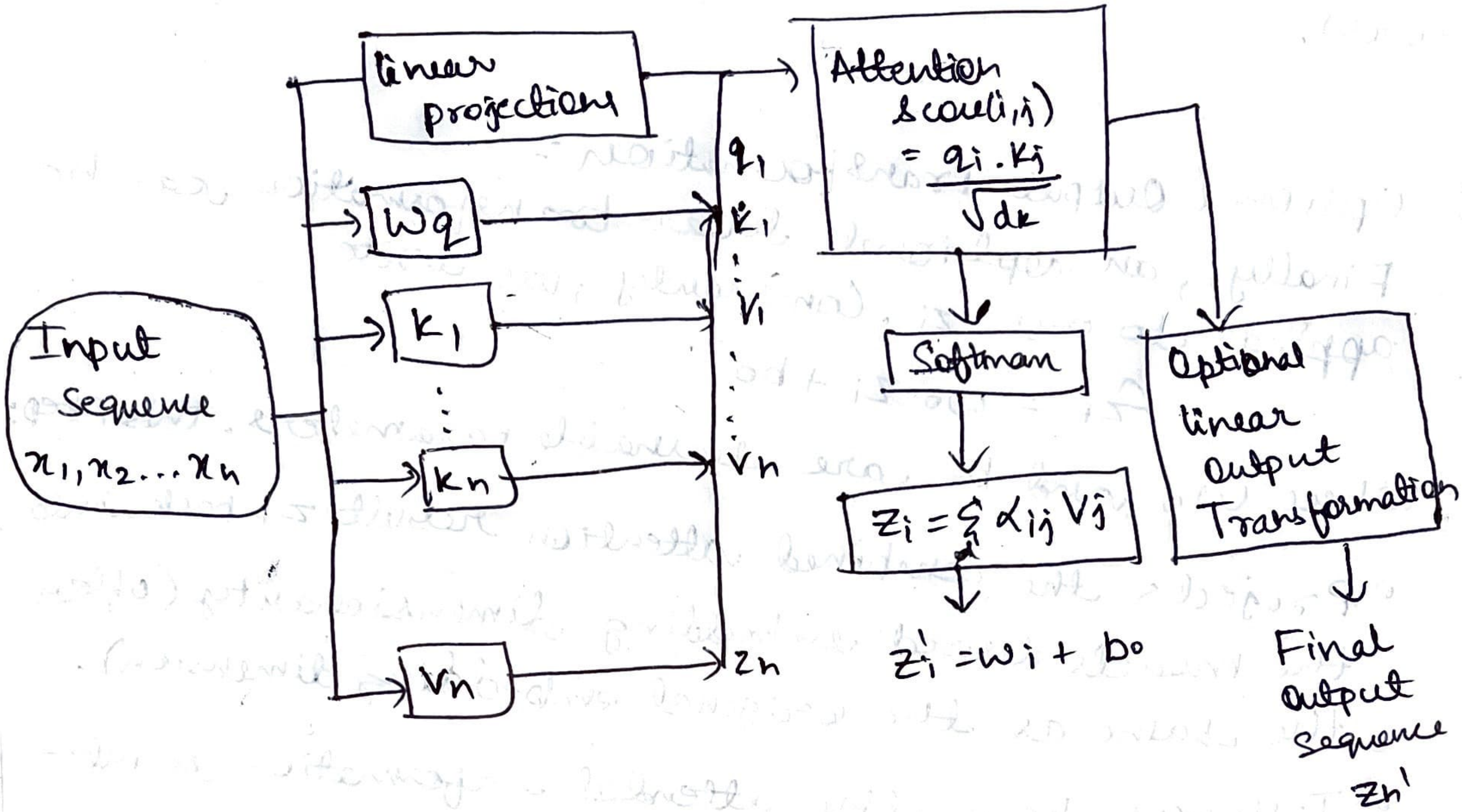
(d) Optional Output Transformation:-
Finally, an optional linear transformation can be applied to each z_i. Commonly, we write:

$$\hat{z}_i = W_o z_i + b_o,$$

where W_o and b_o are learnable parameters. this step:

- Projects the combined attention result z_i back into the model's desired embedding dimensionality (often the same as the original embedding dimension).
- Influences how the attended information is integrated into the broader model's representation. If this projection is omitted, z_i itself can be used directly, but typically the model includes this learnable output matrix to enhance expressive power.
- Including this step leads to better overall performance and flexibility within Transformer architectures.

2. Computational Graph :



Contribution Table:

Team Member	Assignment Part	Contribution (%)
Ruthvik Vasantha Kumar	1,2,3,4, all bonus	50%
Shreyas Bellary Manjunath	1,2,3,4, all bonus	50%