DL - A2 : Team 101

Dhaaneshwar - dshrisai | Suman - sumanman

Part 1-1 :

Autoencoder Architecture:

- Input Layer: 1000 neurons
- Hidden Layer 1: FC, 512 neurons, ReLU
- Bottleneck Layer: FC, 32 neurons
- Hidden Layer 2: FC, 512 neurons, ReLU
- Output Layer: FC, 1000 neurons, Sigmoid
- Loss Function: MSE
- Regularization: L2 Regularization on all weight matrices with a hyperparameter λ

Task 1) Trainable Parameters :

Layer	Parameters
Input	0
Hidden Layer 1 (FC 512)	$1000 \times 512 + 512$ (Bias) $= 512\,512$
Relu	0
Bottleneck Layer	$512 \times 32 + 32 = 16384$
Hidden Layer 2 (FC 512)	$512 \times 32 + 512 = 16896$
Relu	0
Output (FC 1000)	$512 \times 1000 + 1000 = 513000$

Total = 1,058,824

Task 2) Loss with L_2 Regularization:

$$\text{Loss} = \text{mse Loss} + \lambda \sum |w|^2$$

λ: regularization hyperparameter

$\sum |w|^2$: L_2 L_2 Term.

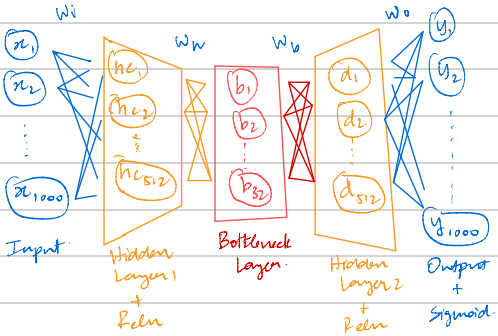
When $w \to 0$, the regularization term tends to zero which results in low loss. This can help prevent overfitting as smaller weights reduce model's complexity and its ability to fit noise.

During Backpropagation,

$$W \leftarrow W - \alpha \left(\frac{\partial \text{loss}}{\partial w} + \lambda w \right)$$

→ This additional term (λw) during backpropagation, encourages smaller weights and smoothens the model to fit better.

Task 3) Computational Graph:



Task 4)

1. Autoencoders are trained on unlabelled data, which can lead to choosing the wrong threshold for loss.
2. New materials need additional training period causing degrade.
3. Imbalanced Dataset can cause generalization failing to flag subtle defects
4. Not all defects affect reconstruction significantly which is an assumption.
5. High Dimensional data may cause inference lag in real time usage.

Task 5)

1. We can use Variational Encoder to introduce probalistic reasoning.
2. Incoporation of Temporal context in LSTM Autoencoder to capture time dependency.
3. Instead of static threshold we can introduce dynamic threshold.
4. Data augmentation to generate synthetic anomalies using simulation or GANS
5. Deploying lightweight models for faster real time quality control on site.

- Sumanan
- dshrisai

} Team 101

PART I. II

(1)

a) Linear Transformations

Each input vector x_i is linearly transformed into three different vectors

Query $q_i = W_q x_i + b_q$

Key $K_i = W_K x_i + b_K$

Value $V_i = W_V x_i + b_V$

where,

→ $W_q, W_K, W_V \in R^{d_{model} \times d_K}$ are learned weight matrices

→ $b_q, b_K, b_V \in R^{d_K}$ are bias vectors

→ d_{model} is the input embedding size; d_K is the dimension of queries, keys & values

q_i → Represents what the token is 'looking for'

K_j → Represents what each token "offers"

V_j → Represents the actual content to use in the weighted sum.

b) Scaled Dot-Product Attention :-

for each query vector q_i, attention scores are computed against all key vector K_j using the dot product:-

$$Score_{ij} = \frac{q_i \cdot K_j}{\sqrt{d_K}}$$

the dot product measures similarity between query q_i & Key K_j

role of $\sqrt{d_K}$:- to prevent large values when d_K is large, which could cause softmax to have extremely small gradients.

Apply softmax over all j to get normalized attention weights:

$$a_{ij} = \frac{\exp\left(\frac{q_i \cdot K_j^T}{\sqrt{d_K}}\right)}{\sum_{j=1}^{N} \exp\left(\frac{q_i \cdot K_j}{\sqrt{d_K}}\right)}$$

c) The attention scores α_{ij} are used to weight the value vectors v_j using the following formula.

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

where,

- α_{ij} is the normalized attention weight between query q_i & k_j
- v_j is the value vector for token x_j
- z_i is the output for token x_i, a weighted sum of all value vectors based on their relevance to x_i

d) Optional Output transformation :-

After computing z_i, we can often apply one more linear transformation :-

$$z_i^{final} = W_o z_i + b_o$$

where, $W_o \in R^{d_k \times d_{model}}$

$b_o \in R^{d_{model}}$

Impact the final representation :-

1) projects the attention output back to the original embedding space
2) Enables stacking of transformer layers
3) Learns a better latent representation by mixing attended information.

2) Computational Graph