FC Layer Mapping - Detailed Explanation

1. Gated Attention Mechanism (Controlled Influence of Each Encoder)

Instead of directly concatenating **E_lab** (Lab Report Embedding) and **E_conv** (Conversation Embedding), introduce a **gating mechanism**:

Step 1: Compute Attention Scores for Each Encoder

- Define learnable **gating weights** for **Lab vs. Conversation**:
-

  $\alpha\_lab = \sigma(W\_lab \cdot E\_lab + b\_lab)$

  $\alpha\_conv = \sigma(W\_conv \cdot E\_conv + b\_conv)$

  - σ is a **sigmoid function** (keeps values between 0 and 1).
  - W_lab, W_conv are **learnable weights**.
  - α_lab, α_conv represent how much each encoder contributes.

Step 2: Weighted Fusion of Embeddings

$E\_fused = \alpha\_lab \cdot E\_lab + \alpha\_conv \cdot E\_conv$

🔹 **Impact**:

- If the lab report is **more reliable**, the model **learns to assign a higher weight to E_lab**
- If conversation symptoms provide **critical missing details**, the model adjusts accordingly.
- If **conflicting information** exists (lab suggests Malaria, but symptoms don't fully match), the model **reduces attention on conversation embeddings** to avoid misclassification.

2. Feature Attribution (Understanding Which Encoder Contributed More to a Prediction)

Once the model has made a prediction, you can analyze **how much each encoder influenced the final decision**.

Method 1: Gradient-Based Attribution

Compute gradients of the output probability pjp_jpj **(for a disease j) w.r.t. each embedding:**

$\partial pj / \partial Elab, \partial pj / \partial Econv$

Interpretation:

- If **$\partial p_j / \partial E_{lab}$** is high, the neuron learned mostly from lab reports.
- If **$\partial p_j / \partial E_{conv}$** is high, the neuron relied more on conversation symptoms.

Method 2 : Attention Weights Analysis (If Using Attention Mechanism)

🔹 **How it Works:**

- If you use **self-attention** or **cross-attention** in your model, you can extract the **attention scores** for each encoder.
- The **higher the attention score for Lab vs. Conversation**, the more influence that encoder had.

🔹 **How to Implement:**

Extract the attention weight matrix after fusion:
$\alpha_{lab}, \alpha_{conv} = \text{softmax}(W_{att} \cdot [E_{lab}, E_{conv}])$

- Higher **α_{lab}** → More reliance on lab reports.
- Higher **α_{conv}** → More reliance on symptoms.
-

🔹 **Impact:**
✅ Gives a **direct interpretable score** of **encoder contribution** for each prediction.
✅ More reliable than gradient-based methods like **Saliency Maps**, which can be noisy.

Method 3 : Cosine Similarity Between Neuron Weights and Encoders

🔹 **How it Works:**

- Measures **how aligned** each neuron's learned weights are with **Lab vs. Conversation embeddings**.

🔹 **How to Implement:**

- Compute **cosine similarity** between **ICD neuron weights** and each encoder embedding:

Similarity calculations:

- **Similarity_lab** = $(W_j \cdot E_{lab}) / (\|W_j\| \|E_{lab}\|)$
- **Similarity_conv** = $(W_j \cdot E_{conv}) / (\|W_j\| \|E_{conv}\|)$

- Higher similarity → The **ICD neuron relies more on that encoder**.

🔹 **Impact:**
✅ Helps quantify **which encoder's features contribute most** to the disease prediction.
✅ **Easy to implement**, works even without attention layers.

---

3. Weight-Based Analysis (Neuron Activation Per Encoder)

- Analyze how strongly each neuron **activates** in response to **E_lab vs. E_conv**.
- Compute **cosine similarity** between **final neuron weights** and each encoder embedding:

Here is the properly formatted version in plain text:

Similarity_lab = (Wj · Elab) / (||Wj|| ||Elab||)
Similarity_conv = (Wj · Econv) / (||Wj|| ||Econv||)

🔹 **Interpretation**:

- If **Similarity_lab > Similarity_conv**, neuron relies **more on lab reports**.
- If **Similarity_conv > Similarity_lab**, neuron relies **more on symptoms**.

You can **tune the training loss** to **increase weight on lab reports when needed**.

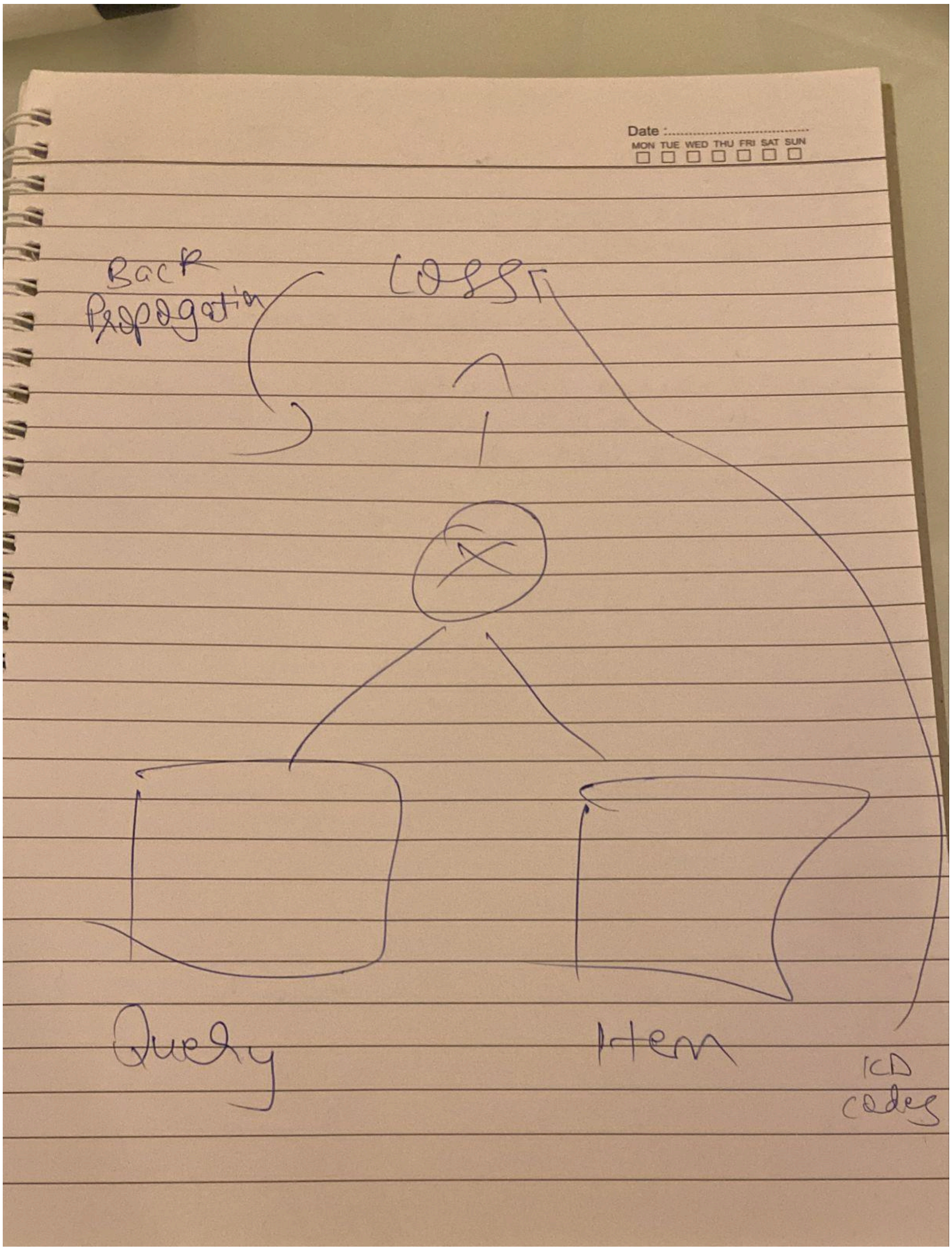
4. Preventing Vague Predictions (Thresholding & Confidence Calibration)

- **Use Probability Thresholding**: Ensure that **diseases are predicted only if confidence is above a threshold** (e.g., 0.7).
- **Calibration**: Train the model to **output uncertainty scores** to detect **low-confidence cases**.

🔹 **Impact**:

- If conversation symptoms **contradict the lab report**, the model should **decrease prediction confidence** and **raise a flag for doctor review** instead of making a vague prediction.

How the Conventional Method Works (Directly Sending Codes to Loss Function for Multilabel Classification)



This approach **directly maps ICD codes to the final layer neurons** and optimizes prediction using **multilabel classification**.

---

1. Model Structure

- **Encoders**

  - **Lab Report Encoder (E_lab, 768-dim)**
  - **Conversation Encoder (E_conv, 1024-dim)**
  - **Fused Embedding: EfusedE_{\text{fused}}** (concatenated or dimensionally reduced to 1536-dim or 1792-dim)
- **Fully Connected (FC) Layer**

  - **Number of neurons = Number of ICD codes** (e.g., 2000 neurons for 2000 diseases)
  - **Each neuron corresponds to one disease.**
  - **Output = Logits for each ICD code**

---

2. Forward Pass (How the Model Predicts)

Computing Logits

- **z_j** = W_j · E_fused + b_j
  - **W_j** → Weights for disease **j**
  - **E_fused** → Input embedding
  - **b_j** → Bias term
  - **z_j** → Logit (raw prediction score)

Applying Activation Function (Sigmoid for Multilabel Classification)

- **p_j** = 1 / (1 + e^(-z_j))

Each neuron independently predicts a probability for its corresponding disease.

---

3. Backpropagation (How the Model Learns)

Loss Function: Binary Cross-Entropy (BCE)

L = - ∑ (from j=1 to C) **[y_j log(p_j) + (1 - y_j) log(1 - p_j)]**

- **C** → Total number of ICD codes
- **y_j = 1** if the disease is present, otherwise **0**

- **Weight Updates:**
  If the model predicts Malaria but the patient actually has Typhoid, it reduces W_malaria and increases W_typhoid.
- **With each training iteration, the predictions are refined.**

---

4. Why Does This Work for Fast Deployment?

🔹 **Simple to implement** (Standard classification model with sigmoid + BCE loss).

🔹 **Trains quickly** as it doesn't require attention mechanisms.

🔹 **Best for deployment**

🔹 **Control over Conversation and Lab Reports Embedding**

🔹 **Could alter their influence on the model easily**

🔹 **Multi-disease support** (predicts multiple ICD codes independently).