****

**SINDH MADRESSATUL ISLAM UNIVERSITY, KARACHI**

**FACULTY OF INFORMATION TECHNOLOGY**

**DEPARTMENT OF SOFTWARE ENGINEERING**

**NATURAL LANGUAGE PROCESSING**

**QUIZ 1: CHART ATTENTION TRANSFORMER (CAT)**

**SUBMITTED BY:**

MUHAMMAD SAAD NASEEM (BSE-21F-097)

BSSE 8C (EVENING)

**SUBMITTED TO:**

SYED MUHAMMAD HASSAN ZAIDI

### **Following are details of model design, your task is to describe in step by step. Model Name is CAT: Chart Attention Transformer**

### **Chart Attention Transformer**

### **CAT Architecture:**

### **Lightweight design choices (e.g., MobileViT, DistilBERT, etc.)**

### **Input encoding (chart image, OCR tokens, question)**

### **Reasoning module (if any):**

### **Chain of Thought (CoT)**

### **Attention flow**

### **Multimodal graph**

**Step-by-Step Breakdown of the CAT (Chart Attention Transformer) Model**

Step 1: **Overview of CAT**

The Chart Attention Transformer (CAT) is a specialized transformer-based model designed to process and reason over chart-based data, combining visual and textual inputs in a lightweight and efficient manner.

Step 2: **CAT Architecture:**

It integrates chart images, OCR-extracted tokens, and user questions to generate meaningful answers, leveraging a multimodal approach. The architecture emphasizes efficiency through lightweight components (inspired by models like MobileViT and DistilBERT) and incorporates a reasoning module that uses Chain of Thought (CoT), attention flow, and a multimodal graph.

Step 3: **Lightweight Design Choices**

CAT adopts a lightweight design to ensure computational efficiency while maintaining performance, drawing inspiration from models like MobileViT (a mobile-friendly Vision Transformer) and DistilBERT (a distilled version of BERT). Here’s how this is implemented:

1. **Vision Component (Inspired by MobileViT):**

* Instead of a full-scale Vision Transformer (ViT), CAT uses a lightweight vision backbone similar to MobileViT. This involves breaking down the chart image into patches (e.g., 16x16 pixel grids) and processing them with a hybrid of convolutional layers and transformer blocks.
* The convolutional layers capture local spatial features (e.g., lines, bars, or text regions in the chart), while lightweight transformer layers model global relationships across patches. This reduces the parameter count and computational overhead compared to standard ViTs.

1. **Language Component (Inspired by DistilBERT):**

* For textual inputs (OCR tokens and questions), CAT employs a distilled transformer architecture akin to DistilBERT. This involves fewer layers (e.g., 6 instead of 12 in BERT) and a reduced hidden size, making it faster and less resource-intensive.
* DistilBERT’s pretraining knowledge is leveraged to encode textual data effectively, ensuring CAT can handle language inputs with minimal computational cost.

Step 4: **Input Encoding**

CAT processes three distinct types of inputs—chart images, OCR tokens, and questions—encoding them into a format suitable for the transformer architecture. Here’s how each is handled:

1. **Chart Image Encoding:**

* The chart image is divided into a grid of patches (e.g., 16x16 or 32x32 pixels), similar to Vision Transformer approaches.
* Each patch is passed through the lightweight vision backbone (e.g., MobileViT-inspired), which generates a sequence of patch embeddings. These embeddings capture visual features like shapes, colors, and spatial layouts of chart elements (e.g., bars, lines, or axes).
* Positional encodings are added to preserve the spatial arrangement of patches, enabling the model to understand the chart’s structure.

1. **OCR Tokens Encoding:**

* Text extracted from the chart (e.g., labels, titles, or numerical values) via Optical Character Recognition (OCR) is tokenized into a sequence of tokens.
* These tokens are fed into the lightweight language module (e.g., DistilBERT-inspired), which converts them into dense embeddings. Special tokens (e.g., [CLS], [SEP]) may be added to mark the beginning and end of the sequence.
* Positional encodings are applied to maintain the order of the OCR tokens, which is critical for interpreting chart annotations correctly.

1. **Question Encoding:**

* The user-provided question (e.g., “What is the highest value in the bar chart?”) is tokenized and processed by the same language module as the OCR tokens.
* The question is encoded into a sequence of embeddings, with positional encodings added to preserve word order. A special [CLS] token embedding may serve as a summary representation of the question.

1. **Unified Input Sequence:**
   * The encoded chart patches, OCR tokens, and question tokens are concatenated into a single input sequence. Modal-specific type embeddings (e.g., vision, OCR, question) are added to distinguish between the three input types, allowing the transformer to process them cohesively.

Step 4: **Transformer Core with Attention Flow**

The encoded input sequence is fed into the core transformer layers of CAT, which process the multimodal data using an attention mechanism tailored for chart understanding:

1. **Multi-Head Self-Attention:**
   * CAT employs multi-head self-attention to model relationships within and across modalities. For example, attention can link a question token (e.g., “highest value”) to relevant OCR tokens (e.g., numerical labels) and chart patches (e.g., the tallest bar).
   * The lightweight design ensures fewer attention heads or a smaller attention dimension, reducing computational complexity.
2. **Attention Flow:**

* The attention mechanism prioritizes information flow based on relevance. For instance, when answering a question about a specific chart element, attention weights focus on the corresponding patches and OCR tokens, filtering out irrelevant parts of the input.
* This dynamic attention flow enables CAT to adaptively emphasize key chart features (e.g., peaks, trends) based on the question.

1. **Layer Processing:**

* The transformer consists of multiple lightweight layers (e.g., 4–6 layers), each with self-attention and feed-forward networks. Layer normalization and residual connections stabilize training, while the reduced depth keeps the model efficient.

Step 5: **Reasoning Module**

CAT incorporates a reasoning module to enhance its ability to answer complex chart-related questions. This module leverages Chain of Thought (CoT) reasoning, attention flow, and a multimodal graph:

1. **Chain of Thought (CoT):**

CAT breaks down the reasoning process into intermediate steps, mimicking human-like problem-solving.

1. Identify all bar heights (via OCR tokens and chart patches).
2. Compare the values step by step.
3. Select the maximum value.

During training, CAT may be fine-tuned with CoT-style prompts (e.g., annotated reasoning steps) to generate rationales before producing the final answer. At inference, it implicitly follows this step-wise logic.

1. **Attention Flow in Reasoning:**

The attention mechanism guides the reasoning process by iteratively refining focus. In each CoT step, attention weights shift to relevant parts of the input (e.g., from all bars to the tallest bar), ensuring the reasoning is grounded in the data.

1. **Multimodal Graph:**

* CAT constructs a graph representation where nodes represent chart patches, OCR tokens, and question tokens, and edges capture relationships (e.g., spatial proximity between a bar and its label, semantic links between question words and chart elements).
* A Graph Attention Network (GAT) or similar mechanism processes this graph, allowing CAT to reason over structured relationships. For example, the graph might connect the question token “highest” to OCR tokens representing numerical values, weighted by their visual prominence in the chart.

Step 6: **Output Generation**

After processing through the transformer layers and reasoning module,

CAT generates an output:

* The final layer’s [CLS] token embedding (or a similar summary representation) is passed through a lightweight feed-forward head. For classification tasks (e.g., multiple-choice questions), it outputs logits over possible answers. For open-ended questions, it may generate a sequence of tokens via a decoder-like mechanism.
* The reasoning module’s CoT output (e.g., intermediate steps) can optionally be provided alongside the final answer for transparency.