Comprehensive Mathematical Analysis of MGIE (MLLM-Guided Image Editing)

1 Introduction

This document presents a comprehensive mathematical analysis of the MGIE (Multimodal Large Language Model-Guided Image Editing) model architecture. This model integrates advanced techniques such as Progressive Feature Blending and Cross-Attention Masking into the standard workflow of image editing using Multimodal Large Language Models (MLLMs) and diffusion models. The following analysis covers the entire pipeline from instruction interpretation through to visual feature extraction and the final image editing process.

2 MGIE Model Architecture

2.1 Image Encoding

Let $\mathbf{I}$ represent the input image. The image encoding process transforms this image into a feature vector $\mathbf{v}$ using an image encoder function f_{enc}. We assume a simple convolutional neural network (CNN) encoder with a single convolutional layer followed by a fully connected layer.

$$\mathbf{v} = f_{\text{enc}}(\mathbf{I})$$

Convolutional Layer The first step is to apply a convolutional layer to the input image. Let $\mathbf{W}_{\text{conv}}$ be the convolutional filter weights and $\mathbf{b}_{\text{conv}}$ be the bias terms. The convolution operation can be described as:

$$\mathbf{H} = \text{ReLU}(\text{Conv2D}(\mathbf{I}, \mathbf{W}_{\text{conv}}) + \mathbf{b}_{\text{conv}})$$

Assume the input image $\mathbf{I}$ is a 4x4 grayscale image and the convolutional filter $\mathbf{W}_{\text{conv}}$ is a 2x2 filter with stride 1. For simplicity, let's assume:

$$\mathbf{I} = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 \\ 0.5 & 0.6 & 0.7 & 0.8 \\ 0.9 & 1.0 & 1.1 & 1.2 \\ 1.3 & 1.4 & 1.5 & 1.6 \end{bmatrix}$$

$$\mathbf{W}_{\text{conv}} = \begin{bmatrix} 0.2 & 0.2 \\ 0.2 & 0.2 \end{bmatrix}, \quad \mathbf{b}_{\text{conv}} = 0.1$$

Performing the convolution operation:

$$\begin{aligned}
\text{Conv2D}(\mathbf{I}, \mathbf{W}_{\text{conv}}) &= \begin{bmatrix} (0.1 \cdot 0.2 + 0.2 \cdot 0.2 + 0.5 \cdot 0.2 + 0.6 \cdot 0.2) & (0.2 \cdot 0.2 + 0.3 \cdot 0.2 + 0.6 \cdot 0.2 + 0.7 \cdot 0.2) \\ (0.5 \cdot 0.2 + 0.6 \cdot 0.2 + 0.9 \cdot 0.2 + 1.0 \cdot 0.2) & (0.6 \cdot 0.2 + 0.7 \cdot 0.2 + 1.0 \cdot 0.2 + 1.1 \cdot 0.2) \end{bmatrix} \\
&= \begin{bmatrix} (0.02 + 0.04 + 0.1 + 0.12) & (0.04 + 0.06 + 0.12 + 0.14) \\ (0.1 + 0.12 + 0.18 + 0.2) & (0.12 + 0.14 + 0.2 + 0.22) \end{bmatrix} \\
&= \begin{bmatrix} 0.38 & 0.36 \\ 0.60 & 0.68 \end{bmatrix}
\end{aligned}$$

Adding the bias term $\mathbf{b}_{\text{conv}}$ and applying the ReLU activation:

$$\mathbf{H} = \text{ReLU}\left(\begin{bmatrix} 0.38 + 0.1 & 0.36 + 0.1 \\ 0.60 + 0.1 & 0.68 + 0.1 \end{bmatrix}\right) = \text{ReLU}\left(\begin{bmatrix} 0.48 & 0.46 \\ 0.70 & 0.78 \end{bmatrix}\right) = \begin{bmatrix} 0.48 & 0.46 \\ 0.70 & 0.78 \end{bmatrix}$$

Fully Connected Layer Next, we flatten the output of the convolutional layer $\mathbf{H}$ and pass it through a fully connected (linear) layer to obtain the final feature vector $\mathbf{v}$. Let $\mathbf{W}_{\text{fc}}$ be the weights and $\mathbf{b}_{\text{fc}}$ be the bias terms of the fully connected layer.

Flattening $\mathbf{H}$:

$$\text{Flatten}(\mathbf{H}) = \begin{bmatrix} 0.48 & 0.46 & 0.70 & 0.78 \end{bmatrix}$$

Assume:

$$\mathbf{W}_{\text{fc}} = \begin{bmatrix} 0.2 & 0.3 & 0.4 & 0.5 \\ 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix}, \quad \mathbf{b}_{\text{fc}} = \begin{bmatrix} 0.1 & 0.2 \end{bmatrix}$$

Performing the linear transformation:

$$\mathbf{v} = \text{Flatten}(\mathbf{H}) \cdot \mathbf{W}_{\text{fc}} + \mathbf{b}_{\text{fc}}$$

Calculating the dot product:

$$\begin{aligned}
\mathbf{v} &= \begin{bmatrix} 0.48 & 0.46 & 0.70 & 0.78 \end{bmatrix} \cdot \begin{bmatrix} 0.2 & 0.3 & 0.4 & 0.5 \\ 0.1 & 0.2 & 0.3 & 0.4 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.2 \end{bmatrix} \\
&= \begin{bmatrix} (0.48 \cdot 0.2 + 0.46 \cdot 0.3 + 0.70 \cdot 0.4 + 0.78 \cdot 0.5) & (0.48 \cdot 0.1 + 0.46 \cdot 0.2 + 0.70 \cdot 0.3 + 0.78 \cdot 0.4) \end{bmatrix} + \left[0.1 \quad 0.2 \right. \\
&= \begin{bmatrix} (0.096 + 0.138 + 0.28 + 0.39) & (0.048 + 0.092 + 0.21 + 0.312) \end{bmatrix} + \begin{bmatrix} 0.1 & 0.2 \end{bmatrix} \\
&= \begin{bmatrix} 0.904 & 0.662 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.2 \end{bmatrix} \\
&= \begin{bmatrix} 0.4 & 0.6 \end{bmatrix}
\end{aligned}$$

Thus, the encoded feature vector $\mathbf{v}$ is:

$$\mathbf{v} = \begin{bmatrix} 0.4 & 0.6 \end{bmatrix}$$

2.2 Multimodal Large Language Model (MLLM)

2.2.1 Overview

The MLLM is at the heart of interpreting natural language instructions and generating expressive guidance for the image editing process. We use a transformer-based model for this purpose.

2.2.2 Tokenization and Embedding

Consider the instruction "Add a blue hat to the person in the image". The natural language instruction is first tokenized into a sequence of words/tokens, and then each token is embedded into a high-dimensional space.

Tokenization The instruction is broken down into the following tokens:

$$\text{Instruction} = [\text{"Add"}, \text{"a"}, \text{"blue"}, \text{"hat"}, \text{"to"}, \text{"the"}, \text{"person"}, \text{"in"}, \text{"the"}, \text{"image"}]$$

Embedding Each token is then converted into a vector. Let's assume we are using a simplified model where each word is embedded into a 2-dimensional space for demonstration purposes:

$$\begin{aligned}
\text{"Add"} &\rightarrow [0.1, 0.2], \\
\text{"a"} &\rightarrow [0.2, 0.1], \\
\text{"blue"} &\rightarrow [0.1, 0.3], \\
\text{"hat"} &\rightarrow [0.3, 0.2], \\
\text{"to"} &\rightarrow [0.1, 0.3], \\
\text{"the"} &\rightarrow [0.2, 0.2], \\
\text{"person"} &\rightarrow [0.3, 0.3], \\
\text{"in"} &\rightarrow [0.1, 0.2], \\
\text{"the"} &\rightarrow [0.2, 0.2], \\
\text{"image"} &\rightarrow [0.3, 0.1]
\end{aligned}$$

These embeddings are then arranged into a matrix $\mathbf{X}_{\text{embedded}}$:

$$\mathbf{X}_{\text{embedded}} = \begin{bmatrix} 0.1 & 0.2 \\ 0.2 & 0.1 \\ 0.1 & 0.3 \\ 0.3 & 0.2 \\ 0.1 & 0.3 \\ 0.2 & 0.2 \\ 0.3 & 0.3 \\ 0.1 & 0.2 \\ 0.2 & 0.2 \\ 0.3 & 0.1 \end{bmatrix}$$

2.2.3 Self-Attention Mechanism

The self-attention mechanism allows the model to focus on different parts of the input sequence, critical for understanding the context and relationships between words in the instructions.

Compute Query, Key, and Value Matrices The Query (Q), Key (K), and Value (V) matrices are computed using the embedding matrix. Assuming the use of identity matrices for simplicity in weights for Q, K, and V:

$$W^Q = W^K = W^V = \mathbf{I}_{2\times 2}$$

$$Q = K = V = \mathbf{X}_{\text{embedded}}$$

$$Q = K = V = \begin{bmatrix} 0.1 & 0.2 \\ 0.2 & 0.1 \\ 0.1 & 0.3 \\ 0.3 & 0.2 \\ 0.1 & 0.3 \\ 0.2 & 0.2 \\ 0.3 & 0.3 \\ 0.1 & 0.2 \\ 0.2 & 0.2 \\ 0.3 & 0.1 \end{bmatrix}$$

Calculate Attention Scores The attention scores are calculated by computing the dot product of Q and the transpose of K, then dividing by the square root of the dimension of the key vectors (in this case, 2) to normalize the scores. This is followed by applying the softmax function to obtain probabilities that sum to 1 across the rows.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

$$d_k = 2$$

$$QK^T = \begin{bmatrix} 0.05 & 0.04 & 0.09 & 0.09 & 0.09 & 0.08 & 0.12 & 0.05 & 0.08 & 0.09 \\ 0.04 & 0.05 & 0.06 & 0.09 & 0.06 & 0.08 & 0.09 & 0.04 & 0.08 & 0.07 \\ 0.09 & 0.06 & 0.10 & 0.12 & 0.18 & 0.08 & 0.18 & 0.08 & 0.08 & 0.12 \\ 0.09 & 0.09 & 0.12 & 0.13 & 0.18 & 0.10 & 0.18 & 0.08 & 0.10 & 0.12 \\ 0.09 & 0.06 & 0.18 & 0.18 & 0.18 & 0.08 & 0.18 & 0.08 & 0.08 & 0.12 \\ 0.08 & 0.08 & 0.08 & 0.10 & 0.08 & 0.08 & 0.12 & 0.06 & 0.08 & 0.07 \\ 0.12 & 0.09 & 0.18 & 0.18 & 0.18 & 0.12 & 0.18 & 0.10 & 0.12 & 0.12 \\ 0.05 & 0.04 & 0.08 & 0.08 & 0.08 & 0.06 & 0.10 & 0.05 & 0.06 & 0.07 \\ 0.08 & 0.08 & 0.08 & 0.10 & 0.08 & 0.08 & 0.12 & 0.06 & 0.08 & 0.07 \\ 0.09 & 0.07 & 0.12 & 0.12 & 0.12 & 0.07 & 0.12 & 0.07 & 0.07 & 0.10 \end{bmatrix}$$

$$\frac{QK^T}{\sqrt{d_k}} = \frac{QK^T}{1.414} = \begin{bmatrix} 0.035 & 0.028 & 0.063 & 0.063 & 0.063 & 0.056 & 0.084 & 0.035 & 0.056 & 0.063 \\ 0.028 & 0.035 & 0.042 & 0.063 & 0.042 & 0.056 & 0.063 & 0.028 & 0.056 & 0.049 \\ 0.063 & 0.042 & 0.070 & 0.084 & 0.127 & 0.056 & 0.127 & 0.056 & 0.056 & 0.084 \\ 0.063 & 0.063 & 0.084 & 0.092 & 0.127 & 0.070 & 0.127 & 0.056 & 0.070 & 0.084 \\ 0.063 & 0.042 & 0.127 & 0.127 & 0.127 & 0.056 & 0.127 & 0.056 & 0.056 & 0.084 \\ 0.056 & 0.056 & 0.056 & 0.070 & 0.056 & 0.056 & 0.084 & 0.042 & 0.056 & 0.049 \\ 0.084 & 0.063 & 0.127 & 0.127 & 0.127 & 0.084 & 0.127 & 0.070 & 0.084 & 0.084 \\ 0.035 & 0.028 & 0.056 & 0.056 & 0.056 & 0.042 & 0.070 & 0.035 & 0.042 & 0.049 \\ 0.056 & 0.056 & 0.056 & 0.070 & 0.056 & 0.056 & 0.084 & 0.042 & 0.056 & 0.049 \\ 0.063 & 0.049 & 0.084 & 0.084 & 0.084 & 0.049 & 0.084 & 0.049 & 0.049 & 0.070 \end{bmatrix}$$

Apply softmax across each row to convert scores to probabilities:

$$\text{softmax}\left(\frac{QK^T}{1.414}\right) = \begin{bmatrix} 0.076 & 0.070 & 0.104 & 0.104 & 0.104 & 0.091 & 0.149 & 0.076 & 0.091 & 0.104 \\ 0.070 & 0.076 & 0.091 & 0.104 & 0.091 & 0.104 & 0.104 & 0.070 & 0.104 & 0.091 \\ 0.091 & 0.076 & 0.104 & 0.149 & 0.194 & 0.091 & 0.194 & 0.091 & 0.091 & 0.149 \\ 0.091 & 0.091 & 0.149 & 0.163 & 0.194 & 0.104 & 0.194 & 0.091 & 0.104 & 0.149 \\ 0.091 & 0.076 & 0.194 & 0.194 & 0.194 & 0.091 & 0.194 & 0.091 & 0.091 & 0.149 \\ 0.091 & 0.091 & 0.091 & 0.104 & 0.091 & 0.091 & 0.149 & 0.076 & 0.091 & 0.091 \\ 0.104 & 0.091 & 0.194 & 0.194 & 0.194 & 0.104 & 0.194 & 0.104 & 0.104 & 0.104 \\ 0.076 & 0.070 & 0.091 & 0.091 & 0.091 & 0.076 & 0.104 & 0.076 & 0.076 & 0.091 \\ 0.091 & 0.091 & 0.091 & 0.104 & 0.091 & 0.091 & 0.149 & 0.076 & 0.091 & 0.091 \\ 0.091 & 0.091 & 0.149 & 0.149 & 0.149 & 0.091 & 0.149 & 0.091 & 0.091 & 0.104 \end{bmatrix}$$

The final attention matrix is then multiplied by the value matrix V to update the representation of each word based on the attention paid to other words:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{1.414}\right) \cdot V$$

$$\text{Attention}(Q, K, V) = \begin{bmatrix} 0.076 & 0.070 & 0.104 & 0.104 & 0.104 & 0.091 & 0.149 & 0.076 & 0.091 & 0.104 \\ 0.070 & 0.076 & 0.091 & 0.104 & 0.091 & 0.104 & 0.104 & 0.070 & 0.104 & 0.091 \\ 0.091 & 0.076 & 0.104 & 0.149 & 0.194 & 0.091 & 0.194 & 0.091 & 0.091 & 0.149 \\ 0.091 & 0.091 & 0.149 & 0.163 & 0.194 & 0.104 & 0.194 & 0.091 & 0.104 & 0.149 \\ 0.091 & 0.076 & 0.194 & 0.194 & 0.194 & 0.091 & 0.194 & 0.091 & 0.091 & 0.149 \\ 0.091 & 0.091 & 0.091 & 0.104 & 0.091 & 0.091 & 0.149 & 0.076 & 0.091 & 0.091 \\ 0.104 & 0.091 & 0.194 & 0.194 & 0.194 & 0.104 & 0.194 & 0.104 & 0.104 & 0.104 \\ 0.076 & 0.070 & 0.091 & 0.091 & 0.091 & 0.076 & 0.104 & 0.076 & 0.076 & 0.091 \\ 0.091 & 0.091 & 0.091 & 0.104 & 0.091 & 0.091 & 0.149 & 0.076 & 0.091 & 0.091 \\ 0.091 & 0.091 & 0.149 & 0.149 & 0.149 & 0.091 & 0.149 & 0.091 & 0.091 & 0.104 \end{bmatrix} \cdot \begin{bmatrix} 0.1 & 0.2 \\ 0.2 & 0.1 \\ 0.1 & 0.3 \\ 0.3 & 0.2 \\ 0.1 & 0.3 \\ 0.2 & 0.2 \\ 0.3 & 0.3 \\ 0.1 & 0.2 \\ 0.2 & 0.2 \\ 0.3 & 0.1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.2 & 0.17 \\ 0.19 & 0.15 \\ 0.24 & 0.21 \\ 0.24 & 0.22 \\ 0.24 & 0.21 \\ 0.21 & 0.18 \\ 0.25 & 0.23 \\ 0.19 & 0.17 \\ 0.21 & 0.18 \\ 0.23 & 0.2 \end{bmatrix}$$

This updated matrix now contains attention-informed feature vectors for each word, incorporating context from the entire instruction set.

2.3 Progressive Feature Blending

Assume we have the language feature matrix $\mathbf{L}$ from the MLLM output and a visual feature matrix $\mathbf{V}$ representing initial or intermediate visual features of the image:

$$\mathbf{L} = \text{Attention}(Q, K, V)$$

$$\mathbf{V} = \begin{bmatrix} 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \end{bmatrix} \quad \text{(Assuming uniform initial visual features)}$$

We introduce blending weights β which change per layer or step. Let's apply blending in two stages for simplicity:

$$\beta_1 = 0.7, \quad \beta_2 = 0.3$$

$$\mathbf{B}_1 = \beta_1\mathbf{L} + (1 - \beta_1)\mathbf{V}$$

$$\mathbf{B}_2 = \beta_2\mathbf{L} + (1 - \beta_2)\mathbf{V}$$

Calculating the first blending stage:

$$\mathbf{B}_1 = 0.7 \cdot \begin{bmatrix} 0.2 & 0.17 \\ 0.19 & 0.15 \\ 0.24 & 0.21 \\ 0.24 & 0.22 \\ 0.24 & 0.21 \\ 0.21 & 0.18 \\ 0.25 & 0.23 \\ 0.19 & 0.17 \\ 0.21 & 0.18 \\ 0.23 & 0.2 \end{bmatrix} + 0.3 \cdot \begin{bmatrix} 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \end{bmatrix}$$

$$= \begin{bmatrix} 0.26 & 0.321 \\ 0.253 & 0.285 \\ 0.288 & 0.327 \\ 0.288 & 0.336 \\ 0.288 & 0.327 \\ 0.267 & 0.306 \\ 0.295 & 0.339 \\ 0.253 & 0.291 \\ 0.267 & 0.306 \\ 0.281 & 0.32 \end{bmatrix}$$

Calculating the second blending stage:

$$\mathbf{B}_2 = 0.3 \cdot \begin{bmatrix} 0.2 & 0.17 \\ 0.19 & 0.15 \\ 0.24 & 0.21 \\ 0.24 & 0.22 \\ 0.24 & 0.21 \\ 0.21 & 0.18 \\ 0.25 & 0.23 \\ 0.19 & 0.17 \\ 0.21 & 0.18 \\ 0.23 & 0.2 \end{bmatrix} + 0.7 \cdot \begin{bmatrix} 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \\ 0.4 & 0.6 \end{bmatrix}$$

$$= \begin{bmatrix} 0.34 & 0.471 \\ 0.337 & 0.465 \\ 0.352 & 0.483 \\ 0.352 & 0.486 \\ 0.352 & 0.483 \\ 0.343 & 0.474 \\ 0.355 & 0.489 \\ 0.337 & 0.471 \\ 0.343 & 0.474 \\ 0.349 & 0.48 \end{bmatrix}$$

These matrices $\mathbf{B}_1$ and $\mathbf{B}_2$ are then used in subsequent processing stages, such as the diffusion model or additional layers of the network, to provide a progressively more visually-informed guidance as the editing process advances.

2.4 Cross-Attention Masking

Let's assume a simple mask that emphasizes the central elements of the instructions, possibly corresponding to key features like "hat" or "blue":

$$\mathbf{M} = \begin{bmatrix} 1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 1 \end{bmatrix}$$

Apply the mask to the attention scores before the softmax step:

$$\text{Masked Attention}(Q, K, V) = \text{softmax}\left(\left(\frac{QK^T}{\sqrt{d_k}} \odot \mathbf{M}\right)\right) V$$

$$\text{Masked } QK^T = \frac{QK^T}{1.414} \odot \mathbf{M}$$

$$= \begin{bmatrix} 0.035 & 0.0028 & 0.0063 & 0.0063 & 0.0063 & 0.0056 & 0.0084 & 0.0035 & 0.0056 & 0.0063 \\ 0.0028 & 0.035 & 0.0042 & 0.0063 & 0.0042 & 0.0056 & 0.0063 & 0.0028 & 0.0056 & 0.0049 \\ 0.0063 & 0.0042 & 0.07 & 0.0084 & 0.0127 & 0.0056 & 0.0127 & 0.0056 & 0.0056 & 0.0084 \\ 0.0063 & 0.0063 & 0.0084 & 0.092 & 0.0127 & 0.007 & 0.0127 & 0.0056 & 0.007 & 0.0084 \\ 0.0063 & 0.0042 & 0.0127 & 0.0127 & 0.127 & 0.0056 & 0.0127 & 0.0056 & 0.0056 & 0.0084 \\ 0.0056 & 0.0056 & 0.0056 & 0.007 & 0.0056 & 0.056 & 0.0084 & 0.0042 & 0.0056 & 0.0049 \\ 0.0084 & 0.0063 & 0.0127 & 0.0127 & 0.0127 & 0.0084 & 0.127 & 0.007 & 0.0084 & 0.0084 \\ 0.0035 & 0.0028 & 0.0056 & 0.0056 & 0.0056 & 0.0042 & 0.007 & 0.035 & 0.0042 & 0.0049 \\ 0.0056 & 0.0056 & 0.0056 & 0.007 & 0.0056 & 0.0056 & 0.0084 & 0.0042 & 0.056 & 0.0049 \\ 0.0063 & 0.0049 & 0.0084 & 0.0084 & 0.0084 & 0.0049 & 0.0084 & 0.0049 & 0.0049 & 0.07 \end{bmatrix}$$

The softmax function is applied to the masked attention scores, focusing the model's attention more on specific elements according to the mask.

2.5 Feed-Forward Network (FFN)

After processing through the attention mechanism, each word's features are passed through a Feed-Forward Network (FFN) to further refine and enhance the features. This network consists of two linear transformations with a ReLU activation in between.

$$\text{FFN}(\mathbf{B}_2) = \text{ReLU}(\mathbf{B}_2 W_1 + b_1)W_2 + b_2$$

Assume the weight matrices W_1, W_2 and bias vectors b_1, b_2 are:

$$W_1 = \begin{bmatrix} 0.2 & 0.2 \\ 0.2 & 0.2 \end{bmatrix}, \quad b_1 = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0.3 & 0.3 \\ 0.3 & 0.3 \end{bmatrix}, \quad b_2 = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix}$$

Apply the first linear transformation and the ReLU activation:

$$\mathbf{B}_2 W_1 + b_1 = \begin{bmatrix} 0.34 & 0.471 \\ 0.337 & 0.465 \\ 0.352 & 0.483 \\ 0.352 & 0.486 \\ 0.352 & 0.483 \\ 0.343 & 0.474 \\ 0.355 & 0.489 \\ 0.337 & 0.471 \\ 0.343 & 0.474 \\ 0.349 & 0.48 \end{bmatrix} \cdot \begin{bmatrix} 0.2 & 0.2 \\ 0.2 & 0.2 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.268 & 0.268 \\ 0.2654 & 0.2654 \\ 0.2754 & 0.2754 \\ 0.2756 & 0.2756 \\ 0.2754 & 0.2754 \\ 0.2686 & 0.2686 \\ 0.277 & 0.277 \\ 0.2654 & 0.2654 \\ 0.2686 & 0.2686 \\ 0.2718 & 0.2718 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.368 & 0.368 \\ 0.3654 & 0.3654 \\ 0.3754 & 0.3754 \\ 0.3756 & 0.3756 \\ 0.3754 & 0.3754 \\ 0.3686 & 0.3686 \\ 0.377 & 0.377 \\ 0.3654 & 0.3654 \\ 0.3686 & 0.3686 \\ 0.3718 & 0.3718 \end{bmatrix}$$

Apply the ReLU activation, which retains only the positive part of each element:

$$\text{ReLU}\left(\begin{bmatrix} 0.368 & 0.368 \\ 0.3654 & 0.3654 \\ 0.3754 & 0.3754 \\ 0.3756 & 0.3756 \\ 0.3754 & 0.3754 \\ 0.3686 & 0.3686 \\ 0.377 & 0.377 \\ 0.3654 & 0.3654 \\ 0.3686 & 0.3686 \\ 0.3718 & 0.3718 \end{bmatrix}\right) = \begin{bmatrix} 0.368 & 0.368 \\ 0.3654 & 0.3654 \\ 0.3754 & 0.3754 \\ 0.3756 & 0.3756 \\ 0.3754 & 0.3754 \\ 0.3686 & 0.3686 \\ 0.377 & 0.377 \\ 0.3654 & 0.3654 \\ 0.3686 & 0.3686 \\ 0.3718 & 0.3718 \end{bmatrix}$$

Apply the second linear transformation:

$$\begin{bmatrix} 0.368 & 0.368 \\ 0.3654 & 0.3654 \\ 0.3754 & 0.3754 \\ 0.3756 & 0.3756 \\ 0.3754 & 0.3754 \\ 0.3686 & 0.3686 \\ 0.377 & 0.377 \\ 0.3654 & 0.3654 \\ 0.3686 & 0.3686 \\ 0.3718 & 0.3718 \end{bmatrix} \cdot \begin{bmatrix} 0.3 & 0.3 \\ 0.3 & 0.3 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.2208 & 0.2208 \\ 0.2196 & 0.2196 \\ 0.2226 & 0.2226 \\ 0.2227 & 0.2227 \\ 0.2226 & 0.2226 \\ 0.2206 & 0.2206 \\ 0.2231 & 0.2231 \\ 0.2196 & 0.2196 \\ 0.2206 & 0.2206 \\ 0.2215 & 0.2215 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.3208 & 0.3208 \\ 0.3196 & 0.3196 \\ 0.3226 & 0.3226 \\ 0.3227 & 0.3227 \\ 0.3226 & 0.3226 \\ 0.3206 & 0.3206 \\ 0.3231 & 0.3231 \\ 0.3196 & 0.3196 \\ 0.3206 & 0.3206 \\ 0.3215 & 0.3215 \end{bmatrix}$$

The output from the FFN gives the final, processed feature vectors for each word, enhanced through the network layers to better capture the nuances of the instruction in the context of the entire sentence.

2.6 Visual Token Generation

Convert the instruction embedding from the FFN into visual tokens, which will be used to guide the diffusion model in the image editing process. Assume that the tokenization model generates 4 visual tokens, each represented as a 2-dimensional vector:

$$\mathbf{u}_i = \text{Tokenize}(\mathbf{E}), \quad i = 1, \ldots, 4$$

For example:

$$\mathbf{u}_1 = [0.1, 0.1], \quad \mathbf{u}_2 = [0.2, 0.2], \quad \mathbf{u}_3 = [0.3, 0.3], \quad \mathbf{u}_4 = [0.4, 0.4]$$

3 Diffusion Model Processing

3.1 Starting with a Noisy Image

Begin the diffusion process with a noisy version of the input image. Let $\mathbf{I}$ represent the initial clean image, and ϵ a noise vector sampled from a normal distribution $\mathcal{N}(0, 0.1)$:

$$\mathbf{z}_0 = \mathbf{I} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.1)$$

Assuming $\mathbf{I}$ has a baseline value of 0.5 and $\epsilon = 0.1$, the initial noisy image $\mathbf{z}_0$ is:

$$\mathbf{z}_0 = 0.5 + 0.1 = 0.6$$

3.2 Iterative Denoising

Perform several steps of denoising, iterating through a predefined number T of steps. For simplicity, let's assume $T = 3$.

For each step t from T to 1, the process involves predicting the noise that was added at each step and then updating the image by subtracting a fraction of this predicted noise.

3.2.1 Predict Noise

Assume a simple linear model for noise prediction based on the step number:

$$\hat{\epsilon}_t = 0.2 \cdot t, \quad t = 3, 2, 1$$

$$\hat{\epsilon}_3 = 0.6, \quad \hat{\epsilon}_2 = 0.4, \quad \hat{\epsilon}_1 = 0.2$$

3.2.2 Update the Image

Assume a learning rate $\eta_t = 0.1$ for simplicity in updating the image based on the predicted noise:

$$\mathbf{z}_3 = 0.6, \quad \hat{\epsilon}_3 = 0.6$$

$$\mathbf{z}_2 = \mathbf{z}_3 - \eta_t \cdot \hat{\epsilon}_3 = 0.6 - 0.1 \cdot 0.6 = 0.54$$

$$\mathbf{z}_2 = 0.54, \quad \hat{\epsilon}_2 = 0.4$$

$$\mathbf{z}_1 = \mathbf{z}_2 - \eta_t \cdot \hat{\epsilon}_2 = 0.54 - 0.1 \cdot 0.4 = 0.50$$

$$\mathbf{z}_1 = 0.50, \quad \hat{\epsilon}_1 = 0.2$$

$$\mathbf{z}_0 = \mathbf{z}_1 - \eta_t \cdot \hat{\epsilon}_1 = 0.50 - 0.1 \cdot 0.2 = 0.48$$

3.3 Final Image

After completing the specified number T of denoising steps, the final image $\mathbf{I}'$ is obtained, representing the edited image based on the instructions provided:

$$\mathbf{I}' = \mathbf{z}_0 = 0.48$$

4 Loss Calculation

4.1 Instruction Loss ($\mathcal{L}_{\text{ins}}$)

Compute the cross-entropy loss between the generated and the true expressive instructions. Assume a simplified scenario where the predicted instruction matches the target, except for some slight variations.

$$\mathbf{y} = \begin{bmatrix} \text{"Add"} \\ \text{"a"} \\ \text{"blue"} \\ \text{"hat"} \\ \text{"to"} \\ \text{"the"} \\ \text{"person"} \\ \text{"in"} \\ \text{"the"} \\ \text{"image"} \end{bmatrix}, \quad \hat{\mathbf{y}} = \begin{bmatrix} \text{"Add"} \\ \text{"a"} \\ \text{"bright"} \\ \text{"blue"} \\ \text{"hat"} \\ \text{"to"} \\ \text{"the"} \\ \text{"person"} \\ \text{"in"} \\ \text{"the"} \\ \text{"image"} \end{bmatrix}$$

4.1.1 Cross-Entropy Loss Calculation

For simplicity, assume a constant cross-entropy loss of 0.2 for each token mismatch and sum over all tokens. The first six tokens match, and the rest have slight variations leading to an additional "bright" and shifted position for "blue":

$$\mathcal{L}_{\text{ins}} = 6 \times 0.2 + 5 \times 0.2 = 2.2$$

4.2 Editing Loss ($\mathcal{L}_{\textbf{edit}}$)

This loss measures the difference between the predicted and actual noise, quantified using the squared error:

$$\mathcal{L}_{\text{edit}} = \sum_{t=1}^{T} \|\epsilon_t - \hat{\epsilon}_t\|^2$$

$$\begin{aligned} \mathcal{L}_{\text{edit}} &= \|0.2 - 0.6\|^2 + \|0.2 - 0.4\|^2 + \|0.2 - 0.2\|^2 \\ &= 0.16 + 0.04 + 0 = 0.20 \end{aligned}$$

4.3 Combined Loss ($\mathcal{L}_{\textbf{all}}$)

The total loss is a weighted sum of the instruction and editing losses, assuming a weight $\alpha = 0.5$ for the editing loss:

$$\mathcal{L}_{\text{all}} = \mathcal{L}_{\text{ins}} + \alpha \cdot \mathcal{L}_{\text{edit}}$$

$$\mathcal{L}_{\text{all}} = 2.2 + 0.5 \cdot 0.20 = 2.3$$

5 Summary and Conclusion

This detailed mathematical analysis demonstrates the comprehensive step-by-step process of the MGIE architecture, including the integration of advanced techniques such as Progressive Feature Blending and Cross-Attention Masking. We have explored the:

- **Input and Preprocessing**: Encoding the input image and interpreting the instruction.
- **MLLM Processing**: Generating detailed instructions and visual tokens.
- **Diffusion Model Processing**: Iteratively denoising the image while applying edits.
- **Loss Calculation**: Computing the instruction loss, editing loss, and combined loss.

These steps illustrate how MGIE leverages both MLLMs and diffusion models to enhance instruction-based image editing. This comprehensive breakdown emphasizes the importance of each component and their collaborative roles in achieving effective image editing.

The integration of Progressive Feature Blending and Cross-Attention Masking further enhances the model's capacity to effectively merge and utilize both textual and visual information for image editing tasks, providing a more robust and context-aware editing capability.

Future work may involve exploring more sophisticated blending strategies, dynamic cross-attention masking based on the content of the instructions, and incorporating additional loss terms to improve the quality and fidelity of the edited images.