# A Universal Text Encoder: Architecture and Datacenter Deployment for Rich Embeddings in Multimodal AI

Rohith Garapati
Independent Researcher
https://github.com/INFINITYone22

August 1, 2025

**Abstract**

I introduce a single universal text encoder designed as a standard for supplying rich language embeddings to multimodal AI applications, such as text-to-image and text-to-video generation. The encoder employs character-level tokenization with a comprehensive vocabulary of 4,096 tokens covering all global languages, enabling detailed semantic capture without subword merging artifacts. Built upon a deep Transformer architecture with 32 layers and 4,096-dimensional hidden states, the model processes letter-level sequences to generate consistent, high-quality embeddings across diverse linguistic inputs. To achieve computational efficiency, the model operates on large datacenters managed by service providers, offloading the encoding burden from user edge devices. During inference, service providers process user prompts, generate embeddings using the universal encoder, and transfer compact representations via SafeTensors format through secure protocols. These embeddings are subsequently fed into local AI models on edge devices for text-to-image or text-to-video generation. This approach establishes a bias-free, consistent standard for multimodal embeddings while ensuring optimal resource utilization across the AI pipeline.

## 1 Introduction

The rapid advancement of multimodal AI systems, particularly in text-to-image and text-to-video generation, has highlighted the critical importance of high-quality text representations. Current approaches often rely on domain-specific encoders or subword tokenization schemes that may introduce biases or lose fine-grained semantic information. I propose a paradigm shift towards a single, universal text encoder that serves as a global standard for generating rich language embeddings across all applications requiring text understanding.

The motivation for this work stems from the observation that existing text encoders, while effective in their respective domains, lack the universality and depth required for truly comprehensive language understanding. Subword tokenization methods, though computationally efficient, often merge meaningful character sequences, potentially losing important semantic nuances that could enhance multimodal alignment. Furthermore, the computational burden of running large encoding models on edge devices limits the accessibility and scalability of advanced AI applications.

My approach addresses these limitations through a character-level tokenization strategy combined with a deep Transformer architecture optimized for datacenter deployment. The resulting system enables service providers to offer encoding-as-a-service, where user prompts are processed centrally and rich embeddings are delivered to edge devices for local AI model conditioning. This architecture

ensures consistent, high-quality representations while optimizing computational resource allocation across the entire AI ecosystem.

## 2 Related Work

The foundation of modern text encoding lies in the Transformer architecture, which revolutionized natural language processing through its self-attention mechanism. The ability to capture long-range dependencies and parallel processing capabilities made Transformers the backbone of contemporary language models. In the context of multimodal AI, models like CLIP demonstrated the effectiveness of contrastive learning for aligning text and image representations, establishing important precedents for cross-modal understanding.

Character-level tokenization, while less common than subword approaches, offers distinct advantages for comprehensive language modeling. Unlike byte-pair encoding (BPE) or SentencePiece tokenization, character-level approaches preserve the fundamental building blocks of language, enabling models to learn compositional patterns from the ground up. This granularity is particularly valuable for handling diverse scripts, rare words, and maintaining consistency across different languages and domains.

The concept of distributed AI computation, where heavy processing occurs in datacenters while edge devices handle lightweight inference, has gained traction with the proliferation of cloud-based AI services. This paradigm allows for optimal resource utilization while maintaining responsiveness and reducing the computational requirements for end-user devices.

My work builds upon these established concepts, combining the depth of Transformer architectures with character-level granularity and distributed deployment strategies to create a comprehensive solution for universal text encoding.

## 3 Model Architecture

### 3.1 Character-Level Tokenization Strategy

The cornerstone of the universal text encoder is its character-level tokenization approach, which treats each individual character, symbol, and special token as a discrete unit. This strategy fundamentally differs from subword tokenization methods by preserving the atomic elements of language, enabling the model to learn compositional patterns and handle diverse linguistic structures without predetermined vocabulary limitations.

The tokenization vocabulary encompasses 4,096 unique tokens, designed to support global language coverage:

- Basic Latin characters: uppercase (A–Z), lowercase (a–z), digits (0–9)
- Extended Latin: accented characters, diacritics, and regional variants
- Major script systems: Cyrillic, Greek, Arabic, Hebrew, Devanagari, and other Indic scripts
- East Asian components: Hangul jamo, Hiragana, Katakana, and fundamental CJK radicals
- Symbols and punctuation: mathematical symbols, currency symbols, emojis, and standard punctuation
- Special tokens: beginning-of-sequence `[BOS]`, end-of-sequence `[EOS]`, padding `[PAD]`, and control tokens

Whitespace handling is critical at the character level. Spaces convey semantic structure but multiple consecutive spaces inflate sequence lengths without proportional value. I implement preprocessing that normalizes multiple consecutive spaces to single spaces, preserving structure while optimizing efficiency. The maximum sequence length is 4,096 tokens.

## 3.2 Transformer Architecture Design

The universal encoder employs a deep Transformer architecture optimized for character-level processing and rich embedding generation. The model consists of 32 identical encoder layers with multi-head self-attention and position-wise feed-forward networks, residual connections, and layer normalization.

### 3.2.1 Multi-Head Self-Attention

Self-attention captures dependencies between characters regardless of positional distance. For layer $\ell$:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\, W^O \tag{1}$$

$$\text{head}_i = \text{Attention}(QW_i^Q,\ KW_i^K,\ VW_i^V) \tag{2}$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \tag{3}$$

The model uses $h = 32$ heads, with per-head dimensions $d_k = d_v = 128$ (total hidden size 4,096), enabling specialization across local orthography, morphology, and global semantics.

### 3.2.2 Position-wise Feed-Forward Networks

Each layer applies a position-wise feed-forward network:

$$\text{FFN}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2 \tag{4}$$

with feed-forward dimension 16,384 (4× hidden size). GELU supports smooth, stable optimization in deep networks.

### 3.2.3 Layer Architecture and Depth

Using pre-normalization, each encoder block computes:

$$h_\ell^{\text{norm}} = \text{LayerNorm}(h_{\ell-1}) \tag{5}$$

$$\tilde{h}_\ell = h_{\ell-1} + \text{MultiHead}(h_\ell^{\text{norm}},\, h_\ell^{\text{norm}},\, h_\ell^{\text{norm}}) \tag{6}$$

$$\tilde{h}_\ell^{\text{norm}} = \text{LayerNorm}(\tilde{h}_\ell) \tag{7}$$

$$h_\ell = \tilde{h}_\ell + \text{FFN}(\tilde{h}_\ell^{\text{norm}}) \tag{8}$$

A depth of 32 layers supports hierarchical feature learning, with early layers modeling local character patterns and deeper layers encoding global semantics.

### 3.3 Model Configuration and Specifications

Listing 1 specifies the configuration:

Listing 1: Universal Text Encoder Configuration

```
model:
  type: transformer_encoder
  precision: fp8 # fallback to bf16 for stability if needed
  vocab_size: 4096 # global character-level coverage
  max_sequence_length: 4096
  layers: 32
  attention_heads: 32
  hidden_dim: 4096
  feedforward_dim: 16384
  activation: gelu
  dropout: 0.1
  normalization: layer_norm
  positional_encoding: sinusoidal

input_preprocessing:
  space_handling: normalize_multiple_to_single

output:
  pooling: eos_token # fixed-size output via EOS pooling
  embedding_dim: 4096
  compression: optional_pca

training:
  pretrain_objective: [masked_language_modeling, contrastive]
  batch_size: 128
  learning_rate: 3e-5
  optimizer: AdamW
  weight_decay: 0.01
  scheduler: cosine_annealing
  epochs: 300
  gradient_clipping: 1.0

inference:
  output_format: safetensors
  compression_level: optional
  transfer_protocol: https_api
```

### 3.4 Precision and Efficiency Considerations

The model employs FP8 precision to reduce memory and accelerate inference while maintaining embedding quality. This enables efficient datacenter deployment for a model with approximately 8.6 billion parameters. Where needed, BF16 can be used to ensure numerical stability during training or sensitive inference regimes.

# 4 Training and Optimization

## 4.1 Pre-training Objectives

Unsupervised pre-training uses masked language modeling (MLM) on diverse multilingual corpora. At the character level, 15% of input characters are masked and predicted from context:

$$\mathcal{L}_{\text{MLM}} = -\mathbb{E}_{x\sim\mathcal{D}}\left[\sum_{i\in M}\log P\big(x_i \mid x_{\setminus M}\big)\right] \tag{9}$$

where $M$ are masked positions and $x_{\setminus M}$ denotes the unmasked context.

## 4.2 Contrastive Fine-tuning

Following pre-training, contrastive fine-tuning with text–image and text–video pairs optimizes cross-modal alignment:

$$\mathcal{L}_{\text{contrastive}} = -\log\frac{\exp\left(\text{sim}(t, v^+)/\tau\right)}{\sum_{v\in\mathcal{V}}\exp\left(\text{sim}(t, v)/\tau\right)} \tag{10}$$

where $\text{sim}(\cdot,\cdot)$ is cosine similarity, $v^+$ is the positive pair, $\mathcal{V}$ is the batch of visual embeddings, and $\tau$ is temperature.

## 4.3 Optimization Strategy

Training uses AdamW with learning rate $3 \times 10^{-5}$, cosine annealing, and gradient clipping. The model trains in distributed settings across multiple GPUs using data parallelism and gradient accumulation to achieve effective batch sizes within memory constraints.

# 5 Deployment Strategy

## 5.1 Datacenter-Centric Architecture

The encoder is deployed in large-scale datacenters for:

- Resource optimization via high-performance compute utilization
- Model consistency across users and applications
- Maintenance efficiency via centralized updates
- Cost effectiveness through shared infrastructure

## 5.2 Service Provider Integration

Providers host the encoder and expose standardized APIs:

1. User applications submit prompts via secure HTTPS endpoints
2. The datacenter generates 4,096-dimensional embeddings
3. Embeddings are serialized using SafeTensors
4. Optional compression is applied and results are transmitted back
5. Local AI models on edge devices consume embeddings for generation

### 5.3 Embedding Transfer and Format

SafeTensors is used for serialization due to:

- Security: validation against malicious payloads
- Efficiency: compact, fast-loading binary format
- Compatibility: cross-platform and framework support
- Metadata: versioning and compatibility tracking

Optional compression (e.g., PCA or quantization) can reduce transfer costs while preserving quality within thresholds.

### 5.4 Edge Device Integration

Edge devices:

1. Load SafeTensors embeddings into local model memory
2. Validate dimensions and metadata
3. Feed embeddings into conditioning pathways
4. Generate images or videos locally

## 6 Technical Considerations and Implementation

### 6.1 Scalability and Performance

FP8 implementation and the 32-layer architecture can be parallelized across multiple GPUs to achieve low-latency inference for complex prompts in production.

### 6.2 Quality Assurance and Consistency

Character-level tokenization ensures consistent handling across languages and scripts, mitigating biases introduced by subword heuristics and improving robustness on rare or domain-specific tokens.

### 6.3 Security and Privacy

Standard security practices include encrypted transport, authenticated access, and audit logging. Prompts are processed ephemerally without persistent storage.

## 7 Conclusion

I presented a universal text encoder that delivers consistent, high-quality language embeddings for multimodal AI. Character-level tokenization, a deep 32-layer Transformer, and a datacenter-centric deployment model together establish a bias-resistant, scalable standard for text conditioning. Future directions include domain-adapted variants, tighter integration with emerging multimodal architectures, and optimization for new hardware. Open-source configuration details and ongoing development are available at INFINITYone22/Universal-Text-Encoder.

## 8 Acknowledgments

This work represents the independent research and development efforts of Rohith Garapati.