# Artificial Intelligence (AI)-based Semantic Communications with Multimodal Data: Framework and Implementation

Jean-Luc A. DeRieux

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Walid Saad, Chair

Harpreet S. Dhillon

Narendran Ramakrishnan

September 10, 2025

Blacksburg, Virginia

Keywords: Semantics, Contextual Learning, Multimodal, Microservice, Matching

# Artificial Intelligence (AI)-based Semantic Communications with Multimodal Data: Framework and Implementation

Jean-Luc A. DeRieux

(ABSTRACT)

Semantic communication (SC) has emerged as an effective paradigm for reducing the bandwidth needs of wireless services by exploiting the so-called "semantics" or meaning behind the data. To date, existing works in this area either focus on multimodal approaches only and omit context-aware recovery or embed it in cross-modal settings, such as audio-to-video, rather than providing a unified, modality-agnostic method. These works also impose substantial architecture redesigns for additional modalities support and are not easily extensible. In contrast to prior work, in this thesis, a novel semantic framework called the semantic context-aware framework for adaptive multimodal reasoning (SCE-FOAM) is proposed. SCE-FOAM is a multimodal semantic framework that enables compact transmission, efficient reconstruction, and contextually-aware predictions using a unique microservice-based architecture. This unique design simultaneously offers an extensible and modular platform for incorporating new modalities and enabling scalable deployment strategies. Experimental results show that SCE-FOAM can achieve data reductions up to 50% for text, 94.56% for audio, and 98.70% for images, respectively. Lastly, the proposed contextual-prediction model achieves an average accuracy of 90% across all modalities. In addition, a heuristic-based extension of the deferred acceptance (DA) matching algorithm is proposed. The extension enables network node matches that incorporate exploration, coverage, and diversification heuristics. In summary, this thesis presents a unified, extensible, context-aware, multimodal SC framework and a heuristic extension to the DA matching algorithm.

# Artificial Intelligence (AI)-based Semantic Communications with Multimodal Data: Framework and Implementation

Jean-Luc A. DeRieux

(GENERAL AUDIENCE ABSTRACT)

Semantic communication is a method for a device to talk with other devices on a wireless network using their own unique and dynamic language, rather than traditional methods that rely on information represented as binary 1's and 0's. By representing information in a traditional binary form, the devices are prevented from understanding the meaning of the message, which can be both inefficient and error-prone when information is being sent or received. By using semantic communication, the devices are endowed with understanding the underlying meaning, or semantics, of the information. This means devices will be able to represent information in their unique language, which can be more efficient than traditional methods and can allow devices to understand context when information is missing. Conversely, in this thesis, a novel semantic framework called the semantic context-aware framework for adaptive multimodal reasoning (SCE-FOAM) is proposed. This framework supports common media types such as text, audio, and images, and is able to efficiently represent information in minimized forms. Additionally, the framework allows for context recovery, which enables devices to predict missing information when their connection is lost. Lastly, it is designed to be easily used by software developers and researchers, which will encourage widespread adoption of SCE-FOAM. In summary, this thesis contributed a semantic framework that supports multimedia (text, audio, image) messages, context recovery, and is designed to be easily used for widespread adoption in the semantic communication field.

# Dedication

*Soli Deo Gloria.*

# Acknowledgments

First, I would like to express my deepest gratitude to my advisor, Dr. Walid Saad, for his unwavering support, guidance, and leadership throughout my research. His consistent mentorship guided me in my graduate academic journey and taught me to always strive to be the best version of myself. Thank you again for your continued encouragement and the opportunity to pursue my graduate degree at Virginia Tech.

I am also grateful to the members of my M.S. advisory committee, Dr. Harpreet Dhillon and Dr. Naren Ramakrishnan, for their valuable feedback and careful review of my thesis.

My sincere thanks go to my colleagues at The Network intElligence, Wireless, and Security laboratory at Virginia Tech (NEWS@VT) for their support and constructive input during my research. In particular, I would like to thank Dr. Christo Thomas, whose weekly meetings and insightful feedback were instrumental in refining this work, and my brother, Alexander DeRieux, for his constant encouragement and valuable recommendations.

And most of all, I am deeply grateful for my faith in Christ, which has guided and sustained me, and for my family: my parents, David and Melinda, and my siblings Alexander, Tatiana, Sophia, Analiesa, Michaila, and Isabelle, for their unwavering love and support. Thank you.

This work was supported in part by the US National Science Foundation under Grant 2201641, and in part by the Center for Assured and Resilient Navigation in Advanced TransportatION Systems (CARNATIONS) under the US Department of Transportation (USDOT)'s University Transportation Center (UTC) program (Grant No. 69A3552348324).

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

AI Artificial Intelligence

API Application Programming Interface

BIN Binary (file)

BLEU Bilingual Evaluation Understudy

CLI Command Line Interface

CPU Central Processing Unit

DA Deferred Acceptance (Gale-Shapley matching algorithm)

DeepJSCC Deep Joint Source-Channel Coding

GPU Graphical Processing Unit

HEX Hexadecimal (base-16 number)

HPC High Performance Compute

HTTP Hypertext Transfer Protocol

IoT Internet of things

JEST Javascript Testing Framework

JIT Just-In-Time

KB Knowledge Base

LPIPS Learned Perceptual Image Patch Similarity

ML Machine Learning

MSE Mean Squared Error

NLP Natural Language Processing

NPM Node Package Manager

PID Process Identification

PSNR Peak Signal to Noise Ratio

QoS Quality of Service

SC Semantic Communication

SCE-FOAM Semantic Context-aware Framework for Adaptive Multimodal Reasoning

SD-VAE Stable Diffusion Variational Autoencoder

SH Shell Script

SNR Signal to Noise Ratio

SONAR Sentence-Level Multimodal and Language-Agnostic Representations

TCP Transmission Control Protocol

UML Unified Modeling Language

VAE Variational Autoencoder

VRAM Video Random Access Memory

WSL Windows Subsystem for Linux

# Chapter 1

# Introduction

Wireless network traffic is projected to grow 85% by 2030, growing at an annual rate of 17% [1]. These demands are being driven by multiple factors, such as 5G adoption [2], increased device density with the Internet of Things (IoT) [3], artificial intelligence (AI)[4], and higher-resolution digital content [5]. Emerging applications in next-generation networks impose stringent quality-of-service (QoS) requirements and require significantly lower latency, which can only be achieved through improved bandwidth efficiency. However, the prevailing communication paradigm, such as in Shannon's information theory [6], focuses on how fast and how reliably a stochastic symbol source can be transmitted over a noisy channel, as illustrated in Fig. 1.1. This approach — also referred to as source coding — can require large encoded representations that arbitrarily convey no semantic interpretation and are purely used to recover the original data with a low probability of error.

Traditional source coding techniques based on Shannon's theory, like H.264 [7], MP3 [8], and JPEG [9], have long enabled efficient bit-level compression for video, audio, and image data, respectively. Nevertheless, these conventional source coding techniques require exact coded information to decode the underlying meaning and are unable to prevent reoccurring transmissions of the same piece of information (e.g., requesting the same image), which increases network bandwidth. There have been recent works to replace traditional source-coding with semantics variants such as deep joint source-channel coding (JSCC) that embed high-level information into deep neural networks, as done in [10–14]. These methods improve

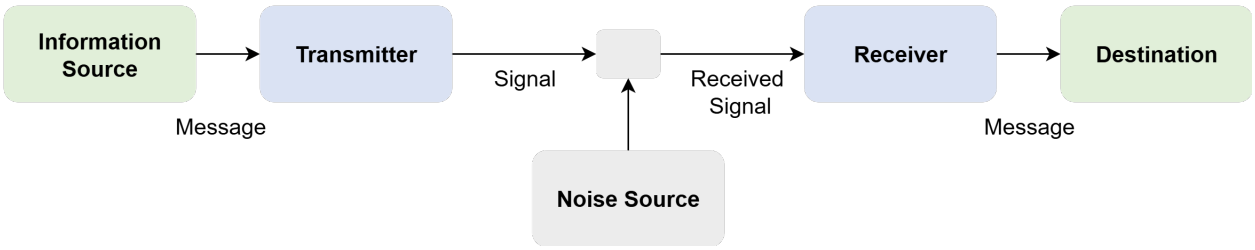



Figure 1.1: A general communication system from Shannon's information theory. Information is passed from a source to a transmitter, which encodes and sends the corresponding signal over a noisy channel to a receiver. The receiver then retrieves the signal, decodes the message, and relays it to the destination source.

bandwidth efficiency by tightly coupling source and channel coding processes around task-relevant semantics rather than transmitting raw bitstreams that redundantly encode low-level data. However, these approaches *do not include modality extensibility*, which prohibits an architecture from including new modality types and *lack context-aware mechanisms* that would allow the receiver to infer or reconstruct missing contextual information.

Semantic communication (SC), on the other hand, is a newer paradigm that richly encodes information into meaning-driven symbols and allows for context recovery from a communication stream [15–18]. Unlike the previous paradigm that encodes information into coded pieces of information that don't convey meaning, semantic symbols represent the meaning of the underlying information. SC can also leverage shared context between a sender and receiver, which enables SC systems to predict missing context. This idea builds upon a growing body of work in semantics as discussed in the next section.

## 1.1 Related Works

There have been multiple recent works on multimodal frameworks and context-aware SC systems, such as [19–27], to name only a few. These works utilize pretrained AI models, multimodal support(natural language, audio, and image), and unified semantic representations

to transmit and receive SC streams.

In [19], the authors leverage large pre-trained multi-modal AI models supporting text, audio, and video to dynamically allocate encoding and decoding resources across different modalities in a unified semantic representation, while a knowledge base is used to resolve semantic ambiguities by incorporating a user-specific context and intent. Their methods showed higher semantic fidelity and greater resiliency to channel noise compared to other models.

In [20], the authors constructed a dynamic feature selection module that adaptively prunes and sends only the most relevant fused semantic symbols to the receiver. The authors also utilize a compact shared codebook dictionary to send the indices corresponding to features to a receiver; these combined methods allow for bandwidth reduction during transmission.

In [21], the authors present a generative, audio-driven video-conference framework that sends continuous audio alongside brief image snippets and leverages a generative adversarial network (GAN) based synthesis to generate frames based on audio cues during a stream. Their proposed method showed an 83% reduction in required transmission bandwidth by transmitting bandwidth without sacrificing perceptual video quality.

The prior works, along with the three that were mentioned. Enabled us to create a list of limitations that formulated our problem statement, which is as follows:

- **Limited context recovery:** These prior works either omit *context-aware* recovery [19, 20, 22, 23, 27] which allows for the receiver to predict missing information, or embed it in cross-modal settings [21, 24].
- **Limited modality extensibility:** These works [19–27] impose substantial architecture redesigns for additional modalities support due to their monolithic structure, and they lack well-defined interfaces that would enable integration for optimized widespread deployment.

These two main limitations of context recovery and modality extensibility ultimately led us to formulate our contributions, which can be seen in the next section 1.2.

## 1.2 Contributions

The main contributions of this thesis are `SCE-FOAM`. `SCE-FOAM` is a semantic context-aware framework for adaptive multimodal reasoning, which introduces a microservice-based semantic translator architecture that integrates encoding, decoding, and contextual predictions. Our unique approach compared to previous methods allows for modality-agnostic APIs, which enable the framework to be easily extensible and maintainable. Another key contribution of this thesis is an extension to the deferred acceptance (DA) [28] algorithm. The extension to the DA algorithm supports a heuristic approach to matching network nodes for exploration, coverage, and diverse matches. Our main contributions include:

- **Multimodal encoding, decoding, and contextualization:** We develop a unified multimodal semantic framework that enables encoding, decoding, and context-aware prediction of semantic symbols and their corresponding modalities (natural language, audio, and images) to adapt dynamically to varying communication conditions.
- **Modularization via microservices:** We design a modular application programming interface (API) microservice-based controller and translator architecture that allows for quick extensibility for integrating new modalities of any kind without the need to integrate each modality into a monolithic software architecture. Ensuring that `SCE-FOAM` framework is network scalable depending on need and deployable across the network for the best practices of software applications.
- **Semantic lexicon:** We introduce a comprehensive semantic lexicon knowledge base

(KB) designed to efficiently store, manage, and transmit semantic symbol references that are frequently sent over the channel to reduce redundancy.

- **Heuristic extension of the deferred acceptance matching algorithm:** We introduce an extended DA algorithm to support a many-to-many solution for network node recommendations using a heuristic to achieve exploratory, coverage, and diverse matching among semantic `SCE-FOAM` network nodes.

Through our evaluation, `SCE-FOAM` achieved uncompressed data reductions up to 50% for text, 94.56% for audio, and 98.70% for images. It further achieves data reconstruction accuracy of nearly 84% using BERTScore for text, 98% using normalized cross-correlation for audio, and a LPIPS score of 0.037 for images, respectively. The proposed contextual-prediction model achieves an average accuracy of 90% across all modalities. Furthermore, the extended DA algorithm showed promising results for encouraging exploration, coverage, and diversification of semantic nodes. Finally, `SCE-FOAM` was submitted to an Institute of Electrical and Electronics Engineers (IEEE) conference for contributions to the SC field. [1]

In the future, we plan to investigate integrating new modalities (virtual reality and augmented reality), new text encoding techniques, and integrating newer pre-trained prediction models for each modality. Finally, we plan to investigate ways to enable globally stable [29] matching between semantic network nodes for the extended DA algorithm.

The rest of this thesis is organized as follows. Chapter 2 discusses the theoretical foundation needed for this work. In chapter 3, we discuss the technical details of the framework. In chapter 4, we show our experiments and results along with a context-recovery example. In chapter 5, we illustrate our heuristic approach to DA, and finally, conclusions and future work are drawn in chapter 6.

[1] J.L. DeRieux, C. Thomas, W. Saad, "SCE-FOAM: Semantic Context-aware Framework for Adaptive Multimodal Reasoning," submitted to *IEEE INFOCOM*, 2026.

# Chapter 2

# Foundations of Semantic Representation and Attention

In this chapter, we provide the theoretical foundations for semantics in section 2.1, establishing the link between raw data and meaning. We then explore attention mechanisms for contextualized understanding in section 2.2, which forms the backbone of modern sequence models. Next, we examine various embedding techniques which can be used for semantic symbols in section 2.3, followed by an introduction to variational autoencoders (VAEs) in subsection 2.4.1 and their role in stable diffusion in subsection 2.4.2. We then conclude with a summary in section 2.5 that highlights key insights across these topics.

## 2.1 From Bits to Meaning

### 2.1.1 Shannon's Insight on Semantic Irrelevance

In Claude Shannon's 1948 paper on communication [6], he states, "These semantic aspects of communication are irrelevant to the engineering problem." This statement was made because communication systems at the time were unable to extract the semantic aspects of information. And so he focused on the core engineering problem of communication, which relied purely on coding and noise-resilience, and with this, he laid the foundations for reliable

transmission through transmitters and receivers as illustrated in Fig. 1.1.

### 2.1.2 Modern Semantic-Information Extensions

With Shannon's focus on the syntactic aspects of information, disregarding semantic content, other researchers were motivated to reincorporate meaning. Two of these notable works are "An Outline of a Theory of Semantic Information" by Rudolf Carnap & Yehoshua Bar-Hillel [30] and "Outline of a Theory of Strongly Semantic Information" by Luciano Floridi [31]. Together, these works explore the formal foundations of extracting meaning from information and the philosophical principles that govern such extraction.

In their paper [30], published in the 1950$s$, Carnap and Bar-Hillel proposed a semantic entropy that learns to assign propositions to unlike and meaningfully distinct information within a given language. Their work aimed to measure the amount of uncertainty removed when a statement is learned, laying the groundwork for a quantitative treatment of meaning in communication.

Decades later, in another paper [31], Floridi offered a contrasting view to Shannon's "weak" information theory, which primarily addressed the syntactic transmission of bitstreams. He argued that for a message to be classified and quantified as semantic, it must satisfy the dual criteria of truthfulness and epistemic validity (i.e., degree of validation). Floridi's account shifted the study of information from a purely mathematical treatment to one rooted in epistemology and philosophy of meaning, enabling a richer conceptual understanding of abstract meaning.

Both of these works laid the foundation for modern semantic communication systems by emphasizing the importance of measuring the uncertainty of meaning as it grows more abstract, as well as the epistemic and philosophical criteria that determine the validity. In doing so,

they paved the way for communication systems to evolve from transmitting raw bitstreams to conveying meaning-centric, information-rich semantic representations.

### 2.1.3 Compression vs. Semantics

In the field of semantic communication, there is often confusion between compression and semantics. At first glance, the two concepts appear similar, as both involve representing information in more compact forms. However, their objectives, scope, and underlying principles are fundamentally different. Compression primarily focuses on how to encode and transmit information using the fewest bits possible, as opposed to semantics, which focuses on what the underlying information's meaning is and how it should be interpreted.

Compression can come in two types: lossless and lossy. Some examples of lossless algorithms are Huffman encoding[32], and LZ77[33]. Some other algorithms that are considered lossy include MP3[8], JPEG[9], and H. 264[7]. The major difference is that in lossless algorithms, the information can be reconstructed without any degradation. In lossy algorithms, a reconstruction will cause the quality of the information to get noisier. The quality degradation and distortion of lossy schemes are often calculated through the use of mean squared error (MSE) [34] and peak signal-to-noise ratio (PSNR) [35] compared to the original data.

In contrast, semantic communication abstracts information at the level of meaning. It discards irrelevant details while retaining task-relevant information, which preserves the informational intent. This enables AI and machine learning (ML) models to reason about each symbol as it holds relevant information about the communication channel. By focusing on meaning rather than form, semantics can substantially reduce the number of bytes required to represent the information while preserving the underlying semantic meaning.

These two concepts of compression and semantics are what form the basis for classical in-

formation theory and semantic communication. Understanding these two concepts is key to understanding the importance of semantics in applications, as this new method outshines current compression methods and will pave the way for a new communication system. In the next section 2.2, we will learn about the Transformer and FastFormer architectures, which form the basis for our contextual prediction model and context embeddings talked about in the subsection 2.3.2.

## 2.2 Transformer and FastFormer

### 2.2.1 Transformer & FastFormer Architecture

The FastFormer was introduced by Wu *et al.* in [36]. Its design was an improvement on the Transformer architecture introduced in [37]. Some of the main differences between these two similar architectures are that Transformers use scaled dot-product self-attention, computing a $N \times N$ similarity matrix. The FastFormer replaces that with a global additive attention that aggregates all the queries and keys into two global summaries, then updates each token with additive fusion with the summaries.

Fig. 2.1 shows the original Transformer architecture that uses the scaled dot-product self-attention. The architecture comprises an encoder and decoder designed for sequence-to-sequence tasks such as translation. It works by first taking in embeddings and using positional encoding to allow for contextual relationships between embeddings to be understood as the transformer takes information in as a single sequence. The positional encoding is then passed into the multi-headed attention, which allows the model to attend to different positions sequence simultaneously learning the relationship. Multiple heads let the model capture different types of relationships in parallel. Normalization is then added to the output

of the multi-headed attention through a residual connection to stabilize training, preserving the gradients. A feed-forward neural network is then used to project the features into a new representation, which is the output of the encoder block. For the decoder block, it follows almost the same process; however, it uses masked multi-headed attention. The mask is to prevent the decoder from looking at future predictions. Then the following processes take place for the remainder of the decoder block: Normalization, multi-headed attention, feedforward, another normalization, a linear layer, and then softmax.

Fig. 2.2 illustrates the architecture of FastFormer, which utilizes global additive attention. By summarizing global query and key vectors and using them to guide value transformations, the FastFormer reduces the computational complexity of each attention layer from $O(N^2d)$ in Transformers to $O(Nd)$ per layer, enabling faster processing of longer sequences.

### 2.2.2 Attention in Transformer

The attention mechanism is commonly used in natural language processing (NLP), such as Bidirectional Encoder Representations from Transformers (BERT) [38] and Generative Pre-trained Transformer (GPT) [39], to learn word correlations in a sequence. Attention mechanisms allow for any sequence to attend to embeddings depending on their correlation and relationship in a sequential stream. Before going into attention, we will first discuss positional encoders within the architecture. The problem is that the transformer processes tokens in parallel and not sequentially. A way to account for order is through positional encoding, as shown in the sinusoidal positional encoder equations (2.1), (2.2), and (2.3).

$$\boldsymbol{x}_i = \boldsymbol{E}_{\text{token},i} + \boldsymbol{P}_i, \tag{2.1}$$

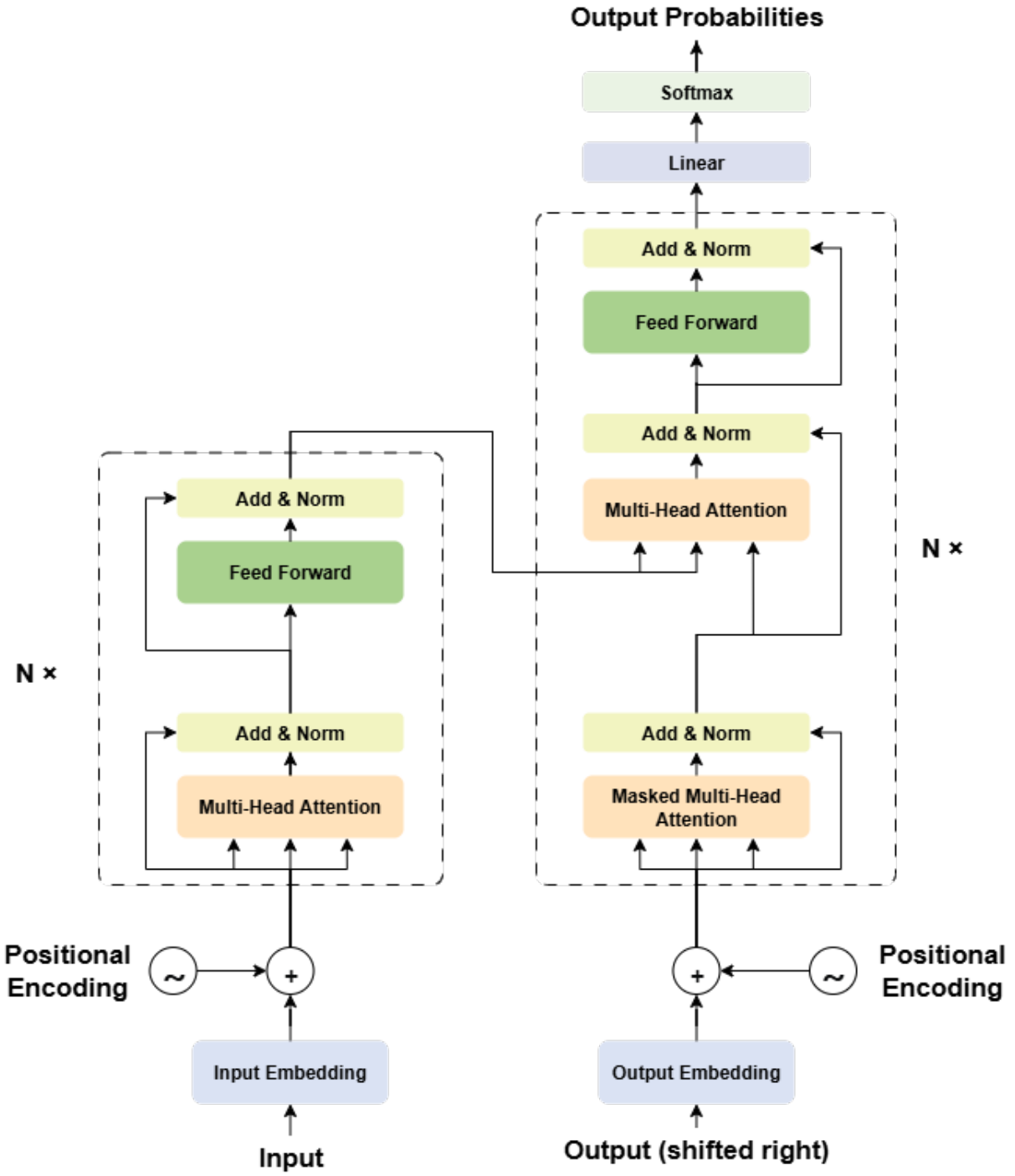



Figure 2.1: Transformer architecture showing positional encoders, multi-headed attentions, masked multi-headed attentions, residual with normalization layers, feed-forward networks, linear layers, and softmax layers.

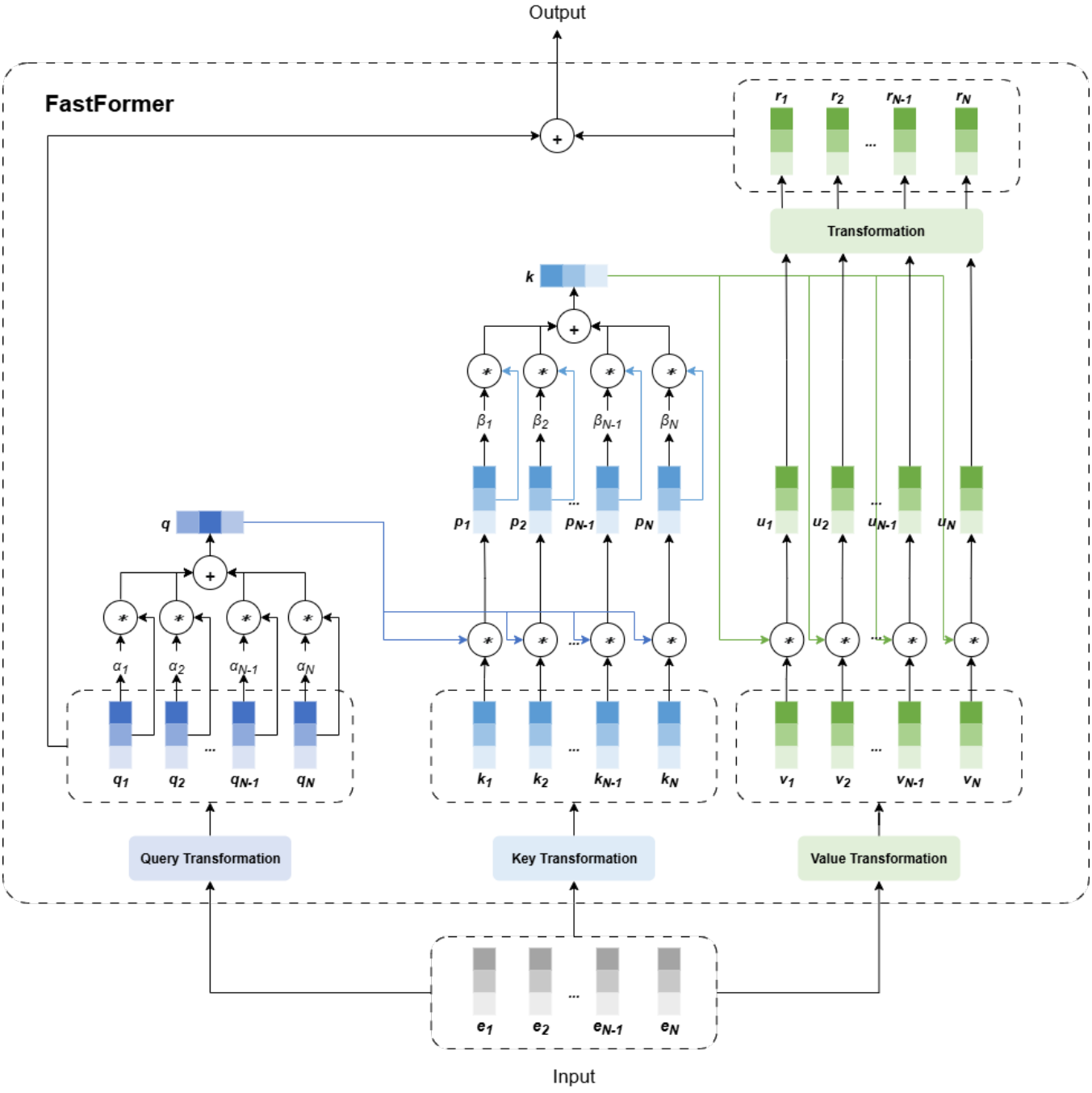



Figure 2.2: Fast former architecture showing query, key, and value transformation blocks operate within a global additive attention mechanism, where global query and key vectors guide the interaction with values.

$$\boldsymbol{P}_{i,2k} = \sin\Big(\frac{i}{10000^{2k/d_{\text{model}}}}\Big), \tag{2.2}$$

$$\boldsymbol{P}_{i,2k+1} = \cos\Big(\frac{i}{10000^{2k/d_{\text{model}}}}\Big) \tag{2.3}$$

Where $\boldsymbol{x}_i$ is the input embedding, $i$ is the token position (starting at 0), $k$ is the index of the embedding dimension, and $d_{\text{model}}$ is the total hidden dimension size. $\boldsymbol{E}_{\text{token},i}$ is the token embedding, $\boldsymbol{P}_{i,2k}$ and $\boldsymbol{P}_{i,2k+1}$ are the positional encoding vectors that are interleaved into each dimension with even $(2k)$, odd $(2k+1)$ pairing to allow for unique vector positions.

After computing positional embeddings, the embedding vectors are then split into $\boldsymbol{Q}$uery, $\boldsymbol{K}$eys, and $\boldsymbol{V}$alues, where self-attention is then performed with each query on all $\boldsymbol{K}$eys. A weighted sum is then computed over the value vectors, producing the self-attention as in (2.4).

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\Big(\frac{\boldsymbol{Q}\,\boldsymbol{K}^\top}{\sqrt{d_k}}\Big)\,\boldsymbol{V} \tag{2.4}$$

The steps for this process are first to project each token embedding into $\boldsymbol{q}_i, \boldsymbol{k}_i, \boldsymbol{v}_i$. Compute attention scores $a_{ij} = \exp\big(\boldsymbol{q}_i \cdot \boldsymbol{k}_j / \sqrt{d_k}\big)$ and normalize the values $\sum_j a_{ij} = 1$, and lastly for each token $i$, the output is $\sum_j a_{ij}\,\boldsymbol{v}_j$, aggregate context from the whole sequence.

By following the positional encoding and self-attention methods. The embeddings retain their sequential positioning while also obtaining attention vectors that allow a model to understand how semantic symbols or words relate to each other sequentially.

### 2.2.3 Global Additive Attention in FastFormer

In global additive attention, a global query vector is used to compute interactions with keys to produce a global key vector, which is then used to interact with the value vectors to produce the output. The first step is to project the input $\boldsymbol{X}$ into queries $\boldsymbol{W}_Q$, keys $\boldsymbol{W}_K$, and values $\boldsymbol{W}_V$ where $\boldsymbol{X} \in \mathbb{R}^{N \times d}$ in ( 2.5):

$$\boldsymbol{Q} = \boldsymbol{X}\,\boldsymbol{W}_Q, \quad \boldsymbol{K} = \boldsymbol{X}\,\boldsymbol{W}_K, \quad \boldsymbol{V} = \boldsymbol{X}\,\boldsymbol{W}_V. \tag{2.5}$$

$$\boldsymbol{q}_{\text{glob}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{Q}_i \qquad \boldsymbol{k}_{\text{glob}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{K}_i \tag{2.6}$$

$$\widetilde{\boldsymbol{X}}_i = \text{ReLU}\big(\boldsymbol{Q}_i + \boldsymbol{k}_{\text{glob}}\big)\,\boldsymbol{W}_1 + \text{ReLU}\big(\boldsymbol{K}_i + \boldsymbol{q}_{\text{glob}}\big)\,\boldsymbol{W}_2. \tag{2.7}$$

Secondly, we then compute the global summaries each in (2.6). Lastly, we fuse each token via additive attention in (2.7) with the global context using additive attention where $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{d \times d}$ are projection matrices. The output $\widetilde{\boldsymbol{X}}_i$ is then combined with $\boldsymbol{x}_i$ via a residual and layer norm before passing into a feed-forward network, which concludes global additive attention.

## 2.3 Embeddings

### 2.3.1 Static Embeddings

In static embeddings like Word2Vec [40] or GloVe [41], each word is assigned a unique vector. These vectors, in the case of Word2Vec (300 dimensions) and GloVe (50, 100, 200, and 300 dimensions), are often significantly large and not easy to comprehend. However, for example, imagine a $2D$ map where the axes are $x$ (left, right) and $y$ (up, down). We then define three classes of items, such as cars, cats, and dogs. On this $2D$ map, when car-like items get placed on the map, these items would be placed somewhat far away from cats and dogs, as this class would relate to vehicles and not animals, indicating low semantic alignment. But if cats and dogs were plotted, they would occur more closely together as they are the same type of class (i.e., animal), indicating high semantic alignment. Each word has a higher-dimensional coordinate that is correlated in space to words of similar meaning.

### 2.3.2 Contextualized Embeddings

Unlike static embeddings, which are fixed vectors. Contextualized word embedding vectors can change depending on their position in a sequence. A clear example is provided by revisiting natural language. Some natural language words can have different interpretations depending on their context. For example, take a word like *Bank*; this specific natural language word can be interpreted in two ways. Here's an example: "She went to the bank to withdraw cash," and "He sat by the bank of the river." This one word can convey several meanings depending on the context and word placement. This can be mitigated by using positional embeddings and an attention mechanism to facilitate sequential word alignment.

### 2.3.3 Multimodal Embeddings

Having examined the static and contextualized embeddings, we proceed with evaluating multimodal embeddings. Multimodal is an ML term that means a combination of multiple different modalities, like text and images, or text and speech, etc. For example, in OpenAI's CLIP embeddings [42], the model was trained on both text and images to compare the semantically rich NLP captions of images while fusing the image data into the embedding. A similar approach is done with sentence-level multimodal and language-agnostic representations (SONAR) [43]. SONAR was trained with text sentence-level embeddings along with a speech encoder (one per language). The model was trained to place both sentences and speech into the same vector space. Allowing for richer context in data fusion.

### 2.3.4 Codec vs. EnCodec

A codec is defined as a coder-decoder compression technique that allows for representing data in a smaller form and then reconstructing it from that representation. There are a mentioned before, there are multiple different techniques, like [44] for text, H.264 [7] and MP3 [8] for audio, and finally JPEG [9] for images. These multimedia codecs aren't an exhaustive list; however, they highlight the most commonly used codecs in the industry.

In contrast, an EnCodec, as introduced by *Meta* [45], is a neural codec. It encompasses an encoder and decoder neural networks that can take in an audio waveform and output a latent tensor representing the semantic meaning. The representation can also be converted back into the waveform by passing the latent tensor into the EnCodec decoder. The EnCodec encoder-decoder model used by *Meta* is learned end-to-end using gradient descent, which enables the networks to learn complex relationships between different features that help it discover efficient, high-fidelity compression strategies.

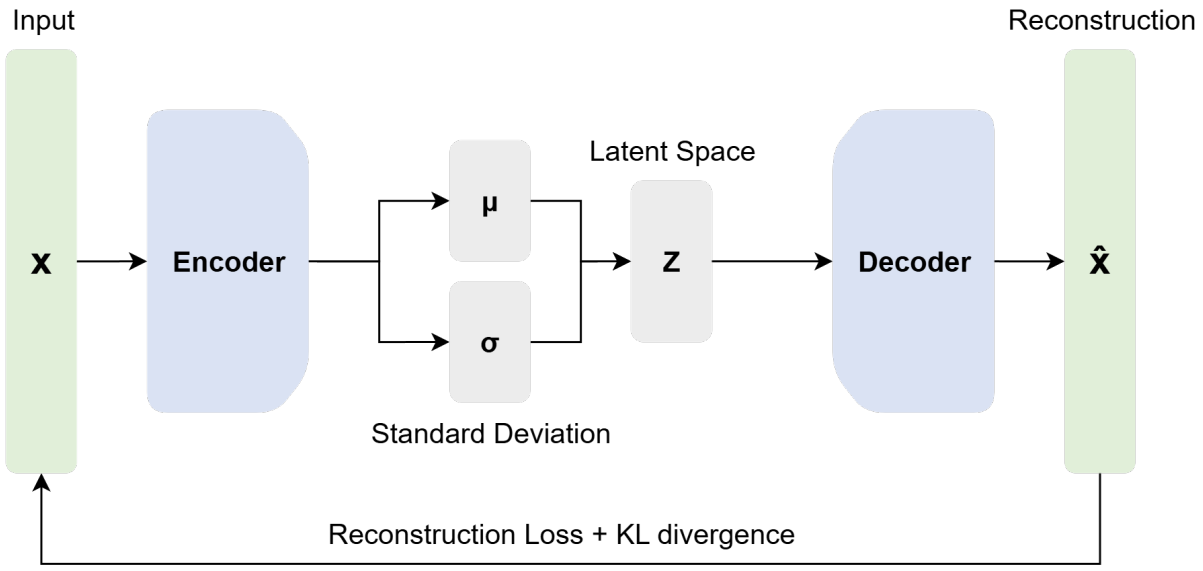



Figure 2.3: VAE architecture showing an input that is fed into an encoder to produce a mean ($\mu$) and standard deviation ($\sigma$) of the approximate posterior distribution. A latent vector $z$ is sampled from the distribution, and then passed through a decoder to produce a reconstruction from the original input. The model then takes the loss and Kullback-Leibler divergence to perform backpropagation.

## 2.4 Variational Autoencoders and Stable Diffusion

### 2.4.1 Variational Autoencoders

Variational autoencoders (VAEs) were introduced by the authors in [46]. A VAE is known as a probabilistic generative model that learns to encode data ($x$) into a latent space ($z$) using standard deviation, then take a vector from that latent space $\hat{z}$, and reconstruct back into $\hat{x}$. Fig. 2.3 represents the VAE architecture with encoder-decoder blocks. Mathematically, the first step used in VAEs is the encoder's approximate posterior as in (2.8):

$$q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) = \mathcal{N}\big(\boldsymbol{z};\, \boldsymbol{\mu}_\phi(\boldsymbol{x}),\, \mathrm{diag}(\boldsymbol{\sigma}^2_\phi(\boldsymbol{x}))\big), \qquad p_\theta(\boldsymbol{x} \mid \boldsymbol{z}), \tag{2.8}$$

Which can be interpreted as the encoder's approximate posterior $q_\phi$, whereas $\boldsymbol{z}$ given $\boldsymbol{x}$ produces a mean vector $\mu_\phi(\boldsymbol{x})$ and a standard-deviation vector. Lastly, we assume that $\boldsymbol{z}$ is a multivariate normal with a mean and diagonal covariance $\text{diag}(\boldsymbol{\sigma}^2_\phi(\boldsymbol{x}))\big)$. The term $p_\theta$, is the decoder likelihood parametrized by $\theta$; it specifies how to reconstruct or generate an observation of $\boldsymbol{x}$ when given a latent $\boldsymbol{z}$ and in the next equation we discus the VAE loss in (2.9) which is used in training to refine the encoder and decoder blocks.

$$\mathcal{L}(\theta, \phi; \boldsymbol{x}) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})\big] \; - \; \mathrm{D}_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p(\boldsymbol{z})\big). \tag{2.9}$$

The VAE loss is then trained with the evidence lower bound (ELBO) [47] objective, and can be interpreted such that $\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\big[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})\big]$ is the reconstruction ($\theta$ decoder parameters) and it encourages the decoder to assign high probabilities to observed $\boldsymbol{x}$ when drawn from the encoder. While the term $\mathrm{D}_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \,\|\, p(\boldsymbol{z})\big)$ is the regularization ($\phi$ encoder parameters) and the goal is to keep the learned posterior close to the prior which prevents $\boldsymbol{z}$ from straying too far from $\boldsymbol{x}$ particular data point being evaluated. Lastly, the VAE updates the model weights by propagating the error as seen in (2.10).

$$\boldsymbol{z} = \boldsymbol{\mu}_\phi(\boldsymbol{x}) \; + \; \boldsymbol{\sigma}_\phi(\boldsymbol{x}) \odot \boldsymbol{\epsilon}, \quad \forall \epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{2.10}$$

Backpropagation is then performed by taking the original $q_\phi(\boldsymbol{z} \mid \boldsymbol{x})$ equation, which is a stochastic draw, and making it deterministic for back propagation. The equation can be interpreted such that $\boldsymbol{z}$ is equal to the mean $\boldsymbol{\mu}_\phi(\boldsymbol{x})$ plus the standard deviation $\boldsymbol{\sigma}_\phi(\boldsymbol{x})$ multiplied element-wise with the noise $\boldsymbol{\epsilon}$ which is sampled once per forward pass from the standard normal. Backpropagation is then performed on the neural network to refine the weights.

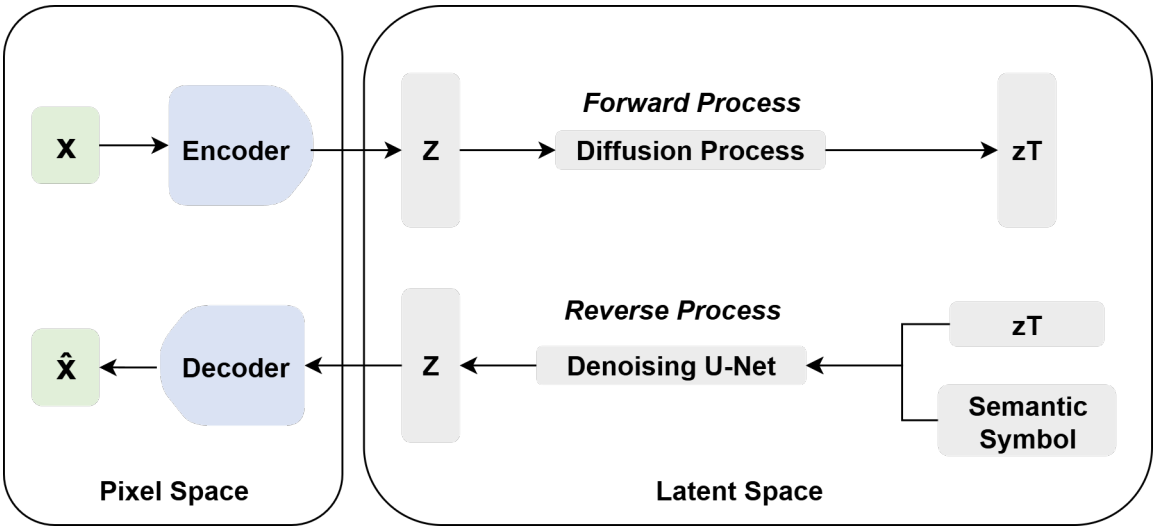



Figure 2.4: SD-VAE architecture showing an input that is fed into a VAE encoder from the pixel space to produce a latent vector $z$. The forward diffusion process is then performed on $z$ to produce $zT$. A condition variable along with $zT$ is then used with reverse diffusion to produce the original $z$. A VAE decoder is then used to reconstruct the original image.

These three equations are used to construct VAEs and are a powerful generative tool for image generation, like Stability AI's stable diffusion VAE [48]. In our next section, we will discuss stable diffusion and how it is applied to VAEs.

### 2.4.2 Stable Diffusion Variational Autoencoders

Stable diffusion VAEs (SD-VAEs), as illustrated in Fig. 2.4, were introduced by the authors in [49] and use the VAE latent space for running the diffusion process instead of on the pixel space. The process enables SD-VAEs to learn directly on a semantically rich feature space, reducing the complexity of modeling high-dimensional pixel space.

SD-VAE diffusion models work by first encoding an image from pixel space into a latent space using a VAE-encoder. The latent vector $z$ and then passed through a forward diffusion process pipeline where Gaussian noise [50] is slowly added until the entire image is transformed into Gaussian noise, which is represented as $zT$ in the Fig. 2.4. The open and

closed form forward diffusion equations are represented in (2.11) and (2.12):

$$q(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}) = \mathcal{N}\Big(\boldsymbol{z}_t; \sqrt{1-\beta_t}\,\boldsymbol{z}_{t-1}, \beta_t \boldsymbol{I}\Big)\,, \text{ where } t = 1, \ldots, T. \tag{2.11}$$

$$q(\boldsymbol{z}_t \mid \boldsymbol{z}_0) = \mathcal{N}\big(\boldsymbol{z}_t; \sqrt{\bar{\alpha}_t}\,\boldsymbol{z}_0, (1-\bar{\alpha}_t)\,\boldsymbol{I}\big)\,, \text{ where } \bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s). \tag{2.12}$$

Where $z_t$ is the noisy sample, $\beta_t$ is the variance at each step. After the noise has been added through forward diffusion, a reverse diffusion process is performed using a U-net that predicts the noise $\epsilon$ present in $z_t$, which allows recovery of the original VAE latent variable $z_0$. The prediction network for the noise represented in (2.13) and (2.14):

$$p_\theta(\boldsymbol{z}_{t-1} \mid \boldsymbol{z}_t, \boldsymbol{c}) = \mathcal{N}\big(\boldsymbol{z}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{z}_t, t, \boldsymbol{c}), \sigma_t^2\,\boldsymbol{I}\big)\,, \tag{2.13}$$

$$\boldsymbol{\mu}_\theta(\boldsymbol{z}_t, t, \boldsymbol{c}) = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{z}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(\boldsymbol{z}_t, t, \boldsymbol{c})\right), \tag{2.14}$$

Where $\boldsymbol{c}$ is an optional conditioning signal as illustrated by the semantic symbol in Fig. 2.4, $\epsilon_\theta$ is the U-net (i.e. neural-network) that predicts the added noise, and $\mu_\theta$ is used to compute the mean of the Gaussian distribution for the reverse diffusion step. We can then calculate the loss for the neural network by using the equation represented in (2.15):

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{t,\,\boldsymbol{z}_0,\,\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\big(\sqrt{\bar{\alpha}_t}\,\boldsymbol{z}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon},\ t\big)\right\|_2^2\right], \tag{2.15}$$

Where $\bar{\alpha}_t$ is the cumulative product of the noise schedule, $\boldsymbol{z}_0$ is the original VAE latent representation of the image, $\epsilon$ is the Gaussian noise, and $\epsilon_\theta$ is the predicted noise at step $t$. Through these processes of forward diffusion, reverse diffusion, and optimizing the loss, the

SD-VAE learns to encode and decode photorealistic images.

## 2.5 Summary

In this chapter, we have discussed Shannon's insight on semantics in subsection 2.1.1, some modern literature addressing semantic information in subsection 2.1.2, and described the difference between compression and semantics in subsection 2.1.3. We then talked about the sequence-to-sequence architectures called the Transformer and FastFormer in section 2.2, along with the different attention mechanisms they use, such as self-attention in subsection 2.2.2 and global attention in subsection 2.2.3. We then discussed different embedding techniques in section 2.3, such as in the subsections static 2.3.1, contextualized 2.3.2, multimodal 2.3.3, and EnCodecs 2.3.4. Finally, we covered generative models such as VAEs and SD-VAEs in section 2.4, illustrating their architectures and fundamental equations. In the next chapter 3, we will discuss the new framework and what this thesis contributes to the field of semantic communication.

# Chapter 3

# System Architecture and Implementation

## 3.1 Overview

Existing semantic communication systems [19–27] exhibit three fundamental limitations that constrain their applicability. *First*, most prior works are tied to fixed or single modalities implementations and lack a unified extensibility for new modalities (e.g., natural language, audio, and vision, etc.). *Second*, they typically do not explicitly model shared context or provide mechanisms for contextual recovery. *Thirdly and Fourthly*, the architectures are often monolithic or compute-intensive, impeding lightweight, scalable deployment across diverse network conditions and edge platforms. This brings forth the central research questions: How can we create a unified semantic communication framework (i) that supports modality extensibility (ii), leverages context-awareness (iii), and is designed to be modular and practical for deployment (iv)? To address this, we propose `SCE-FOAM`, a unified, multimodal, and context-aware prediction framework that can efficiently scale to meet the throughput requirement of network traffic. Fig. 3.1 illustrates our concept of operation diagram showing a bidirectional image-transfer scenario for `SCE-FOAM`. Although text and audio are also supported, this example focuses on images.

In Fig. 3.1, two `SCE-FOAM` systems communicate with each other either wirelessly or locally

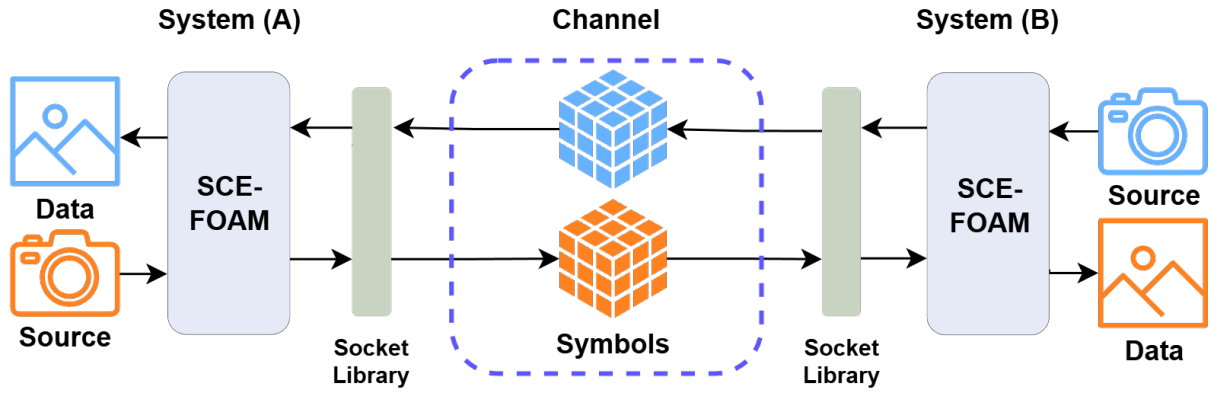



Figure 3.1: Concept of operation diagram using SCE-FOAM in bidirectional image transmission scenario. Systems A and B each act as both a transmitter and receiver: at each SCE-FOAM system, their respective camera sources collect raw images and extract semantic meaning in the form of semantic symbols. The systems then pass the symbols through a socket library, which transmits them to their receiving system. At the receiving system, the socket library passes the symbols to SCE-FOAM, which then collects and decodes these symbols into the underlying meaning.

to share images taken from a camera source. Images are captured directly from a camera source and then fed into SCE-FOAM. The semantic *meaning* is then extracted from the image, represented as a semantic symbol — the blue/orange tensor cube in the diagram — and then sent to the receiving system's SCE-FOAM framework. System B then decodes the symbol and relays the underlying meaning, which is represented as an image in this diagram.

The concept of an operation diagram showing a possible application of sending images semantically with our SCE-FOAM framework; SCE-FOAM supports text, audio, and image, and is designed as a unified approach to access varying modalities by integrating an extensible interface, which is designed to be agnostic to the underlying modality structure. In addition, we leverage continuous contextual learning between each system to contextual predict missing or inadequate data. Finally, our microservice-based architecture allows for scalability as new translators can be deployed to easily adapt to the network demands. In the following sections, we delve into the framework's internal architecture, such as core methods, lexicon,

controllers, and translators.

## 3.2 High-Level Architecture

The architecture for `SCE-FOAM` is illustrated in Fig. 3.2, which describes the semantic framework (SF), supporting methods (i.e., functions), controller classes (i.e., objects), the lexicon, and the translator microservices, which are accessible through APIs. To exchange semantic symbols, our framework would integrate into an existing stack's transmit (TX) and receive (RX) local methods. The SF handles semantic encoding, decoding, and prediction but delegates all socket-level send/receive operations to the desired networking stack on the application layer, such as Python's `socket` library or Node.js's `net` module, without requiring any custom channel-management code.

Within the SC framework, there are three core methods: `send`, `receive`, and `predict`. Each core method can have a controller class that is designed to reside within the SF and access the corresponding modality — text, audio, image — translator microservice for encoding, decoding, and predicting semantic symbols. The `send` method takes in raw information, encodes it, adds the symbols to the KB, and presents the information that needs to be sent. The `receive` method takes in semantic symbols and references and decodes the corresponding information to be relayed from the framework while also training a live continuous model on the data. For the `predict` method, contextual information gets passed continuously through the model, which trains it to understand sequential contextual information. A piece of contextual information can then be inputted into the `predict` method, and what is returned is a prediction based on previously trained context.

The `SCE-FOAM` KB is designed to act as a dynamic lexicon of semantic symbol values with corresponding hexadecimal (HEX) keys, resembling a dictionary of key-value pairs to encom-

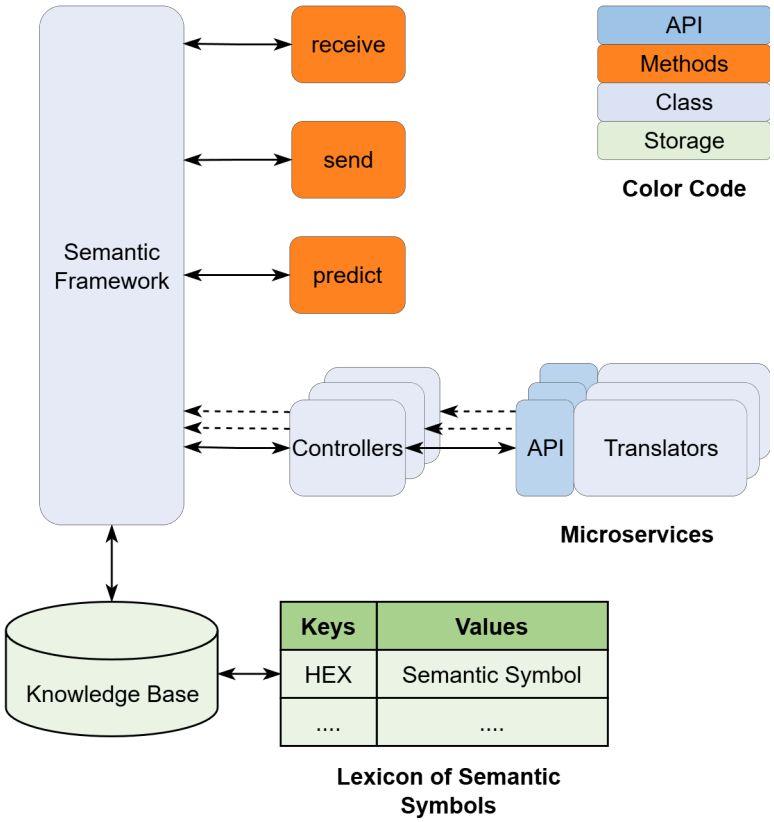



Figure 3.2: Proposed architecture using `SCE-FOAM`. The semantic framework supports three methods `send`, `receive`, `predict`, a local KB lexicon, and connects to translator microservices through controller classes for each modality. To exchange semantic symbols, `SCE-FOAM` can plug into any existing networking stack on the application layer (e.g. Python's `socket` library or Node.js's `net` module).

pass the entirety of the communicated symbols. This allows for the framework to reference previously communicated symbols to prevent redundant bandwidth-heavy transmissions.

A *microservice* is defined as a lightweight, server-based components — deployed locally or in the cloud — that expose a language-agnostic API. This architecture lets developers treat each service as an independent "black-box," calling the functionality without worrying about the internal implementation. In `SCE-FOAM`, the microservices allow for each translator to spin up, scale, or retire each modality service and provide a unified API interface across all modalities, which enables `SCE-FOAM` to be easily deployed. This is different from previous

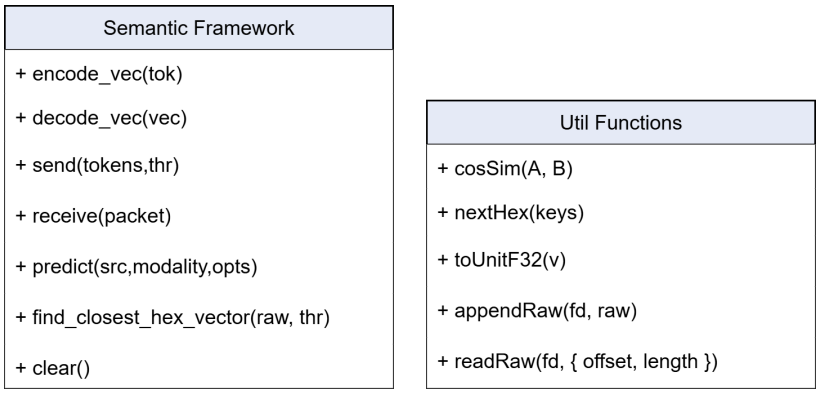



Figure 3.3: `SCE-FOAM` semantic framework UML diagram showing core methods `send`, `receive`, and `predict` along with helper and utility methods.

implementations [19–21], where their frameworks were designed as monolithic software applications, instead of efficient, modular service-based applications. In the following sections, we will discuss the core methods, lexicon, modality controllers, and microservice translators.

## 3.3 Send, Receive, and Predict Methods

In SC, the basic operations can be defined as `send`, `receive`, and `predict` [15]; these operations handle encoding and decoding of semantic symbols, enable dynamic KB lexicon additions, and lookups for minimal and efficient transmission. Lastly, they support training and querying the prediction model for context-aware predictions.

The `send` method takes as input raw information (e.g., text, audio, images, etc.) and a threshold within the range $t \in [0, 1]$. The raw information is first inspected to determine modality type: text, audio, or images. Once the modality is determined, a specific controller designed for the modality is selected. The `send` method then shares the raw information with the controller, and what is returned is an encoded vector $\boldsymbol{v} \in \mathbb{R}^d$. A comparison check is then performed by computing the cosine similarity ($\text{cos\_sim}(\boldsymbol{A}, \boldsymbol{B}) = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\|\boldsymbol{A}\| \, \|\boldsymbol{B}\|}$, where $\boldsymbol{A}$

and $\boldsymbol{B}$ are vectors in the same feature space.) as this allows comparison of vectors which are stored in the KB. What is returned from the cosine similarity is a percentage within the range of $[0, 1]$, which is then compared against the given threshold $t \in [0, 1]$ to determine if a symbol should be added to the KB or reference retrieved. Depending on whether the vector is present in the KB, it will either create a new HEX and vector pair or it will find the vector's key reference, then send either the key or the key-value pair to the `receive` method of the intended `SCE-FOAM` system to mitigate redundant meaning being transmitted.

The `receive` method of the other `SCE-FOAM` system then obtains the symbol from the sender and checks if a key or key-value pair was sent. If given just a key, the method will retrieve the corresponding vector; if given a key-value pair, it first stores that pair and then retrieves the corresponding vector. The algorithm then decodes the semantic vector from the corresponding modality controller, and finally feeds the interpreted information to the interface accessing `SCE-FOAM`.

The `predict` method within `SCE-FOAM` is designed to train on data that has passed through the `receive` method. The `receive` method accesses the controllers/translators decode method, which has a built-in training cycle (i.e., one cycle is a set amount of epochs for a pair depending on modality, see section 3.6 for more details) to learn contextual information. The prediction method can then be requested to provide a prediction vector based on previous vector examples.

When these three methods are combined into the `SCE-FOAM` framework, information being passed through the SF dynamically gets encoded into semantic symbols, decoded back into the original modality from a semantic symbol, stored or referenced for efficient bandwidth transmissions, and predicted when data is missing or is unusable. These methods, when integrated into this framework, allow for seamless integration into existing systems and architectures that want to utilize the benefits of SC, without having to build everything

from scratch into a researcher's or developer's existing software stack. Researchers can easily access the methods from the `SCE-FOAM` framework, and everything else is handled automatically without any custom method additions. In the next section, we will discuss in detail the technical aspects of the KB lexicon, how it operates, and how it is accessible.

## 3.4 Knowledge Base Lexicon

The KB lexicon is designed as a dictionary structure with pairs of 32-*bit* HEX keys and vector embedding values of varying sizes depending on the modality. This structure was inspired by ASCII [44] tables, which have a HEX key and natural language character value pairs. The vector embeddings that are stored in the KB are the semantic symbols that relay meaning, and the size of the KB can grow dynamically depending on how many semantic symbols are being transmitted over the communication channel and what threshold as a percentage ranging from $[0, 1]$ is set by the developer when using accessing the `send` method.

The `send` method's threshold uses the cosine similarity, which determines how alike an incoming vector is compared to those present in the KB lexicon. Higher thresholds for the cosine similarity between vectors mean the algorithm favors *uniqueness of meaning*, rather than a lower threshold that allows for *overlapping meaning* to pass through. By setting the right threshold, symbols will have just the right uniqueness and occurrence compared to if the threshold is set to an extremely high or significantly low value. We use the Basic Linear Algebra Subprograms (BLAS) `sdot` [51] cosine similarity method to compare, where BLAS time complexity is $O(n)$, and $n$ is the dot product of two length vectors.

Each KB is initialized on creation of the corresponding `SCE-FOAM`. Corresponding *knowledge_base.json* and *knowledge_base_latents.bin*, which contain the dictionary key-value pairs for referencing semantic symbols, and the binary file holds the $float32$ vectors for the seman-

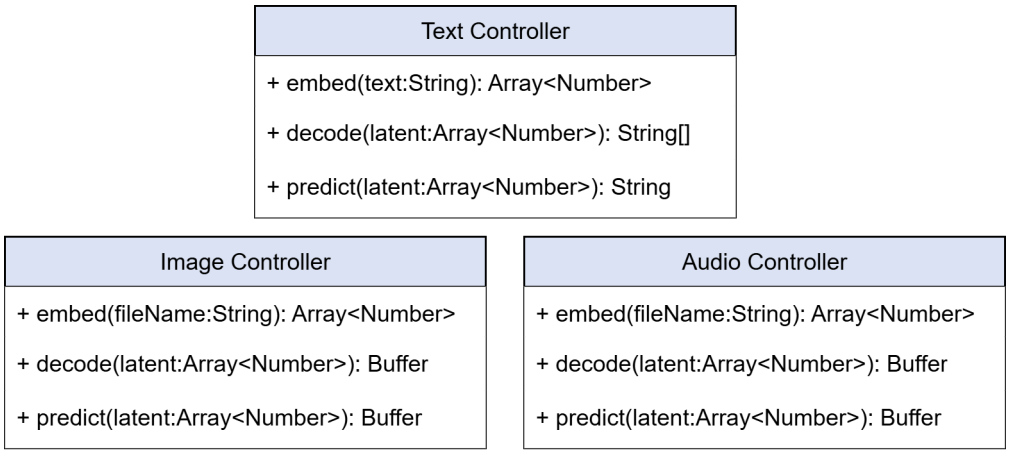



Figure 3.4: `SCE-FOAM` controller UML diagrams showing embed, decode, and predict methods for text, audio, and image, respectively.

tic symbol (values). The key-value pairs use a HEX as the keys and use a {offset, length, mod, [rows, cols]} for accessing the $float32$ vectors directly in the *knowledge_base_latents.bin* file.

By maintaining an active KB lexicon of semantic symbols, compared to sending redundant symbols, our SF recognizes and references previously transmitted symbols, boosting bandwidth efficiency during recurring transmission feeds.

## 3.5 Modality Controllers

Each modality controller is designed for accessing their corresponding translator microservices to access the supported modalities such as text, audio, and images. While the `SCE-FOAM` framework accesses the controllers directly for making those operations. Fig. 3.4 shows each modality controller interface with embed, decode, and predict controller methods.

For embedding, the text controller takes as input a string directly, while audio and image modality controllers reference a file from a directory on the sender's host machine (i.e., the machine running `SCE-FOAM`). These three modalities output a vector array $\boldsymbol{V} =$

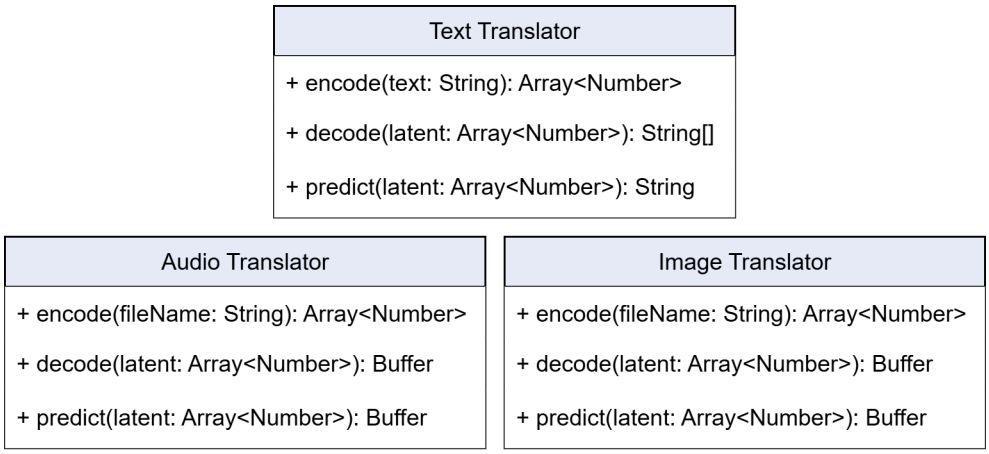



Figure 3.5: `SCE-FOAM` translator UML diagrams showing encode, decode, and predict methods for text, audio, and image, respectively.

$[\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N] \in \mathbb{R}^{d \times N}$ of $float32$ corresponding to their embeddings.

For decoding, the text controller inputs an array of vectors — the output from the controller's embedding method — and returns an array of strings that corresponds to each text that was received sequentially as an input. For audio and image, an array of vectors is passed in, and what is returned is a buffer corresponding to the common WAV or JPEG file format.

Finally, for prediction, the `predict` method is fed an input of a contextual vector and returns a prediction vector $f : \mathbb{R}^d \to \mathbb{R}^k, \quad \hat{\boldsymbol{y}} = f(\boldsymbol{x}_{\text{ctx}})$ corresponding to data been trained on.

## 3.6 Modality Translator

We design modality translators to accomplish three separate tasks. The first is to encode and decode a modality into a vector, then translate it back into the original form, and lastly, predict vectors based on previous contextual vectors. With this design, we created three separate microservice translators dedicated to text, audio, and images. Fig. 3.5 shows each modality translator with the corresponding encode, decode, and predict interface methods.

For the text translator, we use Meta's SONAR sentence embeddings [43], which maps word tokens into $d = 1024$ dimensions. SONAR's time complexity is $O(n^2d + nd^2)$ and $O(md^2)$ for encoder/decoder, where $n$ is the token length, and $d$ is the model dimension, respectively. A key challenge in text-based SC is finding an embedding space that is more information dense than ASCII [44], as most common word embedding spaces are 512-dimensions, such as GloVe [41] and Word2Vec [40]. With this in mind, we chose to do SONAR as it is a sentence-level embedding and it avoids using the 512-dimensions per word, which far exceeds the underlying ASCII byte usage. The encoding and decoding modules take in text and output semantic symbols. To enable effective prediction, we incorporated a history queue $\mathcal{H}_q = \{(\boldsymbol{v}_i, \boldsymbol{w}_i)\}_{i=1}^{k},\ \boldsymbol{v}_i, \boldsymbol{w}_i \in [-1, 1]^n$ that updates with each decoding step. Once the history contains more than two entries, a FastFormer [36] is trained for a set number of epochs to predict the next symbol in the sequence $f : (\mathbb{R}^d)^n \to (\mathbb{R}^d)^n,\ f([\boldsymbol{v}_{t-n+1}, \ldots, \boldsymbol{v}_t]) = [\boldsymbol{w}_{t-n+1}, \ldots, \boldsymbol{w}_t],\ \boldsymbol{v}_i, \boldsymbol{w}_i \in \mathbb{R}^d$ (e.g., the model trains for a set number of epochs, called *cycle*, to predict $\boldsymbol{w}_i$ when given $\boldsymbol{v}_i$). After each training cycle, the corresponding instances are removed from the history. Training the model on the history allows the receiver to understand the context in a communication stream. The FastFormer's time complexity is $O(nd + nd^2)$ where $n$ is the token length, and $d$ is the model dimension, respectively.

For the audio translator, it uses Meta's audio encoder and decoder *EnCodec24kHz* [45], which takes in an audio clip and encodes it into a semantic symbol of varying code-block sizes. EnCodec's time complexity is $O(L \cdot C \cdot k \cdot D) \approx O(L)$ where $L$ is the audio length sequence, $C$ fixed channel, $k$ kernel, and $d$ depth constant. Similar to how a text translator works, the audio server trains the prediction method on a history queue larger than 2. Each prediction is designed to output 30-second windows regardless of the input sequence.

The last supported modality translator is image. It utilizes an SD-VAE from Stability AI [48] and works by taking in frames of any size, which it then resizes to $512 \times 512$. The resized

frame is then fed into the SD-VAE for encoding. Decoding also works in the reverse process. The SD-VAE's time complexity is $O(HW \cdot C^2 \cdot L)$ where spatial size $H \times W$; $C$ channel width, $L$ depth. For predictions, the model will train on a history queue greater than 2 and, after a training cycle, will pop off the queue and wait until new information has been received to be added to the queue. This allows the contextual prediction model to continuously learn.

All three translator microservices were designed to have the same embed, decode, and predict interface and corresponding controllers to allow for researchers and software developers to easily integrate more modalities depending on the required application.

## 3.7 Summary

In this chapter, we introduced the `SCE-FOAM` semantic framework including concept of operations scenario in section 3.1, the system architecture in section 3.2 overview showing how each module works, the three core methods within the framework `send`, `receive`, and `predict` in section 3.3, the KB lexicon in section 3.4 with hex and symbol pairs, the controllers in section 3.5 designed for accessing the microservices, and the modality translators in section 3.6 for running directly the multimodal models for encoding, decoding, and predicting semantic symbols from sequential streams. In the next chapter 4, we present experimental results that validate `SCE-FOAM` architecture.

# Chapter 4

# Experimental Results and Analysis

## 4.1 Datasets and Experimental Setup

To evaluate `SCE-FOAM`, we selected a diverse set of test data across different modalities. We use varying text corpus lengths, codecs, and image resolutions to measure the semantic attributes of compression, latency, accuracy, and prediction capabilities.

For our codecs, we used FLAC [52], MP3 [8], OGG[53], Opus[54], and WAV [55] for audio. For image, we used 720p, 1080p, 1440p, and 4k resolutions with AVIF [56], JPEG [9], JXL [57], PNG [58], and WebP [59] codecs, respectively. For text, we used plain *txt* files [44].

For our latent models, we used word two vector (Word2Vec) [60], global vectors for word representation (GloVe) [41], and fast text (FastText) [61] for text. For audio, we used EnCodec (48 kHz, 96 kbps) [62], EnCodec (48 kHz, 128 kbps) [62], Descript Audio Codec (96 kbps) [63], and SoundStream (192 kbps) [64]. Lastly, we used Efficient Learned Image Compression [65], Hyperprior models (BMShj, MSH) [66, 67], and Cheng et al. 2020 (cheng20) [68] for images.

For our accuracy and timing evaluation, we used three various text sources: the Declaration of Independence [69], the Magna Carta [70], and Abraham Lincoln's Gettysburg Address [71]. For audio, we used three separate audio tracks: "Resonance" by HOME [72], "Wave" by Forhill [73], and John F. Kennedy's inaugural address [74]. For images, we used NASA's

EVA-92 photograph of an astronaut on the ISS [75], a photo of a golden retriever dog at the park [76], and an image of Sáttítla Highlands from a NASA satellite [77].

Finally, for our prediction evaluation, we used three sequential video feed sources: NASA ISS feed [78] showing an example of a loss packet, New York Central Park video feed [79] with an unusable frame, and Disney's Steamboat Willie animation [80] with a damaged frame. Each of these sources was selected to validate the versatility of the `SCE-FOAM` framework.

## 4.2 Hardware, Software Environment, and Scripts

The experiments were run on a workstation equipped with an NVIDIA RTX 4070 Ti GPU, an Intel Core i5-13600K CPU, and 32GB of DDR5 RAM. The chosen GPU's 12GB of V-RAM was sufficient to load all three modality models simultaneously, which allowed the microservices to make fast inferences without having to load a model when a query is requested, and could be set up for scaling of multiple microservices based on the required demand.

We ran the communication experiments locally between two `SCE-FOAM` instances on the same host machine. To account for a real-world mobile network delay, a approximate channel-specific delta of 25–50 ms can be added(i.e., 50–100 ms round-trip time (RTT)) [81] to the measured local latencies to account for the real-world mobile network delay.

The software operating system we used was Ubuntu [82] via Windows Subsystem for Linux (WSL) [83]. We then used the Node.js [84] runtime environment and the `npm` package manager [85] within WSL to run each script command. The scripts were written using the Jest [86] framework, which allowed for efficient and controlled experiments.

Each *npm* script command was designed to validate the varying aspects of evaluation, such as latent size, latency, reconstruction and prediction accuracy, and communication feeds.

Table 4.1: A table of the commands and descriptions of the Jest test scripts used to evaluate and validate each functionality within the `SCE-FOAM` architecture framework.

| Command | Description |
|---|---|
| `npm run test` | Run all tests. |
| `npm run test:text` | Run all text tests. |
| `npm run test:audio` | Run all audio tests. |
| `npm run test:image` | Run all image tests. |
| **Size Tests** | |
| `npm run test:size` | Run size tests for all modalities. |
| `npm run test:size:text` | Run text size tests only. |
| `npm run test:size:image` | Run image size tests only. |
| `npm run test:size:audio` | Run audio size tests only. |
| **Prediction Tests** | |
| `npm run test:pred` | Run all prediction tests. |
| `npm run test:pred:text` | Run text prediction tests only. |
| `npm run test:pred:image` | Run image prediction tests only. |
| `npm run test:pred:audio` | Run audio prediction tests only. |
| **Prediction Feed Tests** | |
| `npm run test:feed` | Run all feed prediction tests. |
| `npm run test:feed:nasa` | Run NASA video feed prediction tests only. |
| `npm run test:feed:park` | Run park video feed prediction tests only. |
| `npm run test:feed:film` | Run film video feed prediction tests only. |
| **Timing Tests** | |
| `npm run test:time` | Run all timing tests. |
| `npm run test:time:text` | Run text timing tests only. |
| `npm run test:time:image` | Run image timing tests only. |
| `npm run test:time:audio` | Run audio timing tests only. |
| **Communication Tests** | |
| `npm run test:comm` | Run all communication tests. |
| `npm run test:comm:text` | Run text communication tests only. |
| `npm run test:comm:image` | Run image communication tests only. |
| `npm run test:comm:audio` | Run audio communication tests only. |
| **Accuracy Tests** | |
| `npm run test:acc` | Run all accuracy tests. |
| `npm run test:acc:text` | Run text accuracy tests only. |
| `npm run test:acc:image` | Run image accuracy tests only. |
| `npm run test:acc:audio` | Run audio accuracy tests only. |

Table 4.2: Evaluation metrics by modality.

| **Modality** | **Metric** |
|---|---|
| Text | Bidirectional Encoder Representations from Transformers Score (BERTScore) |
| Audio | Normalized Signal Cross-Correlation |
| Image | Learned Perceptual Image Patch Similarity (LPIPS) |
| Embedding | Cosine Similarity |

Table 4.1 shows the list of *npm* commands that can be used to validate the test results shown in this thesis. This list also allows for our framework to be easily validated for future CI/CD workflows, enabling automated and controlled testing before deploying any official framework updates or releases to the public.

All of the experiments and the evaluation scripts, along with the source code for `SCE-FOAM`, are publicly available on the NEWS@VT GitHub repository [1]

## 4.3 Evaluation Metrics

We assessed perceptual quality, semantic alignment, and reconstruction performance using a collection of established metrics across several different modalities, such as natural language, audio, and visual. Table 4.2 shows the evaluation metrics used for each modality.

Text reconstructions were evaluated with the Bidirectional Encoder Representations from Transformer Score (BERTScore) [87] to measure the semantic correlation and similarity between source and reconstructed texts. The BERTScore metric can produce a range between 0 and 1, where 1 is a perfect semantic reconstruction of the original text.

Audio normalized signal cross-correlation [88] is used to measure the structural fidelity and signal alignment of the reconstructed waveform with the original for audio. The output from

[1] https://github.com/news-vt/scefoam

Table 4.3: Timing metrics (milliseconds) for encode, decode, and predict.

| Modality | Encode | Decode | Predict |
|---|---|---|---|
| Text | 142.00 | 3594.79 | 1333.87 |
| Audio | 2519.90 | 3076.85 | 3532.70 |
| Image | 281.66 | 349.35 | 225.41 |

this metric will produce a range between -1 and +1, where +1 is a perfect reconstruction.

Learned Perceptual Image Patch Similarity (LPIPS) [89] approximates perceptual similarity compared to methods that measure pixel-wise errors (e.g., Mean Squared Error (MSE) [34]) and produces a range between 1 to 0, where 0 is a perceptually identical image.

We evaluated the prediction accuracy of each modality by computing the cosine similarity between the forecasted and target embeddings, which directly quantifies how the higher-dimensional semantic attributes align. Together, these metrics ensure our evaluation captures the perceptual quality and semantic alignment required for SC.

## 4.4 Inference Time

The inference latencies for each modality are summarized in table 4.3. These latencies are also after optimizing each model by using mixed precision variables, efficient resizing and interpolation using Torch [90], and allowing each model to stay within GPU memory, mitigating reload times for faster inferences.

We found that image inference speeds were fastest, averaging 285 milliseconds for encoding, decoding, and predictions. For text, the inference speeds started to grow exponentially from 142 milliseconds all the way to 3594.79 milliseconds. Lastly, we found that audio performed the worst with an average 3-second inference speed. Although these latencies are relatively high, they are consistent with our hardware's (RTX 4070 Ti) performance limits, which can

handle 376.4 tokens/sec.

In the future, purpose-built hardware designed for Transformer models should enable sub-millisecond or even nanosecond latencies. One promising example is Sohu by Etched [91], which is an ASIC-based hardware accelerator optimized to run the Transformer architecture. According to an inference graph they released of Sohu [91], they were able to obtain $500,000$ tokens/sec, though not specifying which model they used. Assuming their metric is based on a similar model size, it would enable real-time semantic communication. As a comparison, the RTX 4070 Ti achieved 376.4 tokens/sec according to our internal testing of the GPT-2 model [92]. That is about $1329\times$ speedup in token throughput on Sohu versus the RTX 4070 Ti.

## 4.5 Comparison of Codecs, Latent Models, and Uncompressed Sizes

Comparison evaluations were performed with varying sizes of lengths and resolutions for each modality (i.e., text, audio, image). We empirically solved size comparisons for different codecs (See table 4.4) and then analytically solved different size comparisons for latent variables (See table 4.4).

For text, we used excerpts from the Declaration of Independence [69] in 100, 500, 1000, and 1335-word increments, highlighting the smallest representation in bold. Here we notice that in the table 4.4a for 100 and 500-word files, the original text is more minimal. However, for 1000 and 1335 words, the latent sizes were smaller than their text counterpart. Beyond approximately 700 words, the latent variable outperforms ASCII, demonstrating that longer passages benefit from the semantic representation. For our text latent model comparison,

Table 4.4: Size metrics for different text, audio, and image codecs compared to the latent models we selected in *bytes*.

(a) Text

| **Filename** | **Raw** | **SONAR** |
|---|---|---|
| words_100.txt | **578** | 4,096 |
| words_500.txt | **3,042** | 4,096 |
| words_1000.txt | 6,110 | **4,096** |
| words_1335.txt | 8,117 | **4,096** |

(b) Audio

| **Filename** | **Raw** | **Mp3** | **Ogg** | **Opus** | **Flac** | **EnC24-48k** |
|---|---|---|---|---|---|---|
| test_audio_1 | 50,274,474 | 4,332,332 | 3,519,010 | 4,777,345 | 22,250,775 | **2,733,789** |
| test_audio_2 | 37,572,686 | 3,409,012 | 3,298,365 | 3,640,074 | 31,727,174 | **2,041,158** |
| test_audio_3 | 25,808,974 | 2,341,962 | 1,468,258 | 1,714,750 | 8,522,615 | **1,409,619** |

(c) Image

| **Resolution** | **Raw** | **Jpg** | **Webp** | **Jxl** | **Avif** | **SD-VAE** |
|---|---|---|---|---|---|---|
| 4k | 10,642,635 | 2,383,140 | 593,688 | 641,561 | 282,542 | **140,219** |
| 1440p | 5,838,923 | 1,351,930 | 375,716 | 342,884 | 192,680 | **140,351** |
| 1080p | 3,707,337 | 887,774 | 271,582 | 238,832 | 145,810 | **140,041** |
| 720p | 1,839,057 | 454,142 | 153,162 | 130,725 | **89,329** | 140,137 |

our method using SONAR exceeded the overall performance metrics of other models.

For audio, we compare the byte sizes of different codecs against the latent variable in table 4.4b. Audio file 1 was based on the 4:45-minute song "Wave" by Forhill. The test revealed that the semantic symbol was the smallest representation at 94.56% reduction over WAV and 36.90% compared to MP3. Moving towards our audio latent model, our selected EnCodec outperforms all other methods, such as EnC48, DAC, and ST.

Finally, for images, we compared different sizes and codecs of a NASA Astronaut image [75] in table 4.4c. The experiment showed that for almost all the file formats and resolutions, our method of using SC outperformed almost all the other codecs. With almost a 98.70% reduction in size with 4k image PNG over latent, and the smallest improvement of 4% reduction compared to JXL. We found that 720p AVIF was the only format smaller than the latent-based. This was due to the dimensionality of our latent space compared to the

Table 4.5: Size metrics for different text, audio, and image latent models compared to the latent models we selected in *bytes*.

(a) Text

| **Filename** | **Raw** | **SONAR** | **Word2Vec** | **GloVe** | **FastText** |
|---|---|---|---|---|---|
| words_100.txt | 578 | **4,096** | 120,000 | 120,000 | 120,000 |
| words_500.txt | 3,042 | **4,096** | 600,000 | 600,000 | 600,000 |
| words_1000.txt | 6,110 | **4,096** | 1,200,000 | 1,200,000 | 1,200,000 |
| words_1335.txt | 8,117 | **4,096** | 1,602,000 | 1,602,000 | 1,602,000 |

(b) Audio

| **Filename** | **Raw** | **EnC24-48k** | **EnC48-96k** | **EnC48-128k** | **DAC-96k** | **ST-192k** |
|---|---|---|---|---|---|---|
| test_audio_1 | 50,274,474 | **855,000** | 3,420,000 | 4,560,000 | 3,420,000 | 6,840,000 |
| test_audio_2 | 37,572,686 | **639,000** | 2,556,000 | 3,408,000 | 2,556,000 | 5,112,000 |
| test_audio_3 | 25,808,974 | **438,000** | 1,752,000 | 2,336,000 | 1,752,000 | 3,504,000 |

(c) Image

| **Res.** | **Raw** | **SD-VAE** | **ELIC** | **BMShj** | **Cheng20** | **MSH** |
|---|---|---|---|---|---|---|
| 4k | 10,642,635 | 1,036,800 | **829,440** | 1,036,800 | 1,555,200 | 2,073,600 |
| 1440p | 5,838,923 | 460,800 | **368,640** | 460,800 | 691,200 | 921,600 |
| 1080p | 3,707,337 | 259,200 | **207,360** | 259,200 | 388,800 | 518,400 |
| 720p | 1,839,057 | 115,200 | **92,160** | 115,200 | 172,800 | 230,400 |

number of pixels representing the original image. Finally, for the images, we noticed that our method exceeded the size comparison of ELIC, but we believe this is the case because our method of SD-VAE holds more information compared to the other methods.

We attribute our performance gains, which average 29.08% across all modalities for codecs, and 64.37% for all latent models, to the dimensionality used for each modality. Each modality utilizes a low-dimensional space to preserve the underlying semantic meaning, thereby driving the major improvements compared to the information-dense codec and uncompressed methods.

Table 4.6: Comparison of original and reconstructed text excerpts: Declaration of Independence, Magna Carta, and Gettysburg Address. (A–C) originals; (D–F) reconstructions.

| **Original — Declaration of Indep.** | **Original — Magna Carta** | **Original — Gettysburg Address** |
|---|---|---|
| In Congress, July 4 1776 the unanimous Declaration of the States of America, when in the course of human events, it … | JOHN, by the grace of God King of England, Lord of Ireland, Normandy and Aquitaine, and Count of Anjou, let it be known to… | Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in Liberty, and dedicated… |
| **Reconstruction — Declaration of Indep.** | **Reconstruction — Magna Carta** | **Reconstruction — Gettysburg Address** |
| In this year's Declaration of the Fourteenth Amendment of the States of America, July 4, 1776, the unanimous Declaration of the… | God, Lord of England, by the grace of King Henry, King of England, King of Ireland, Duke of Normandy and of Aquitaine… | Four hundred and seven years ago, our fathers gave birth to a new nation on this continent, dedicated to the idea that all men are… |

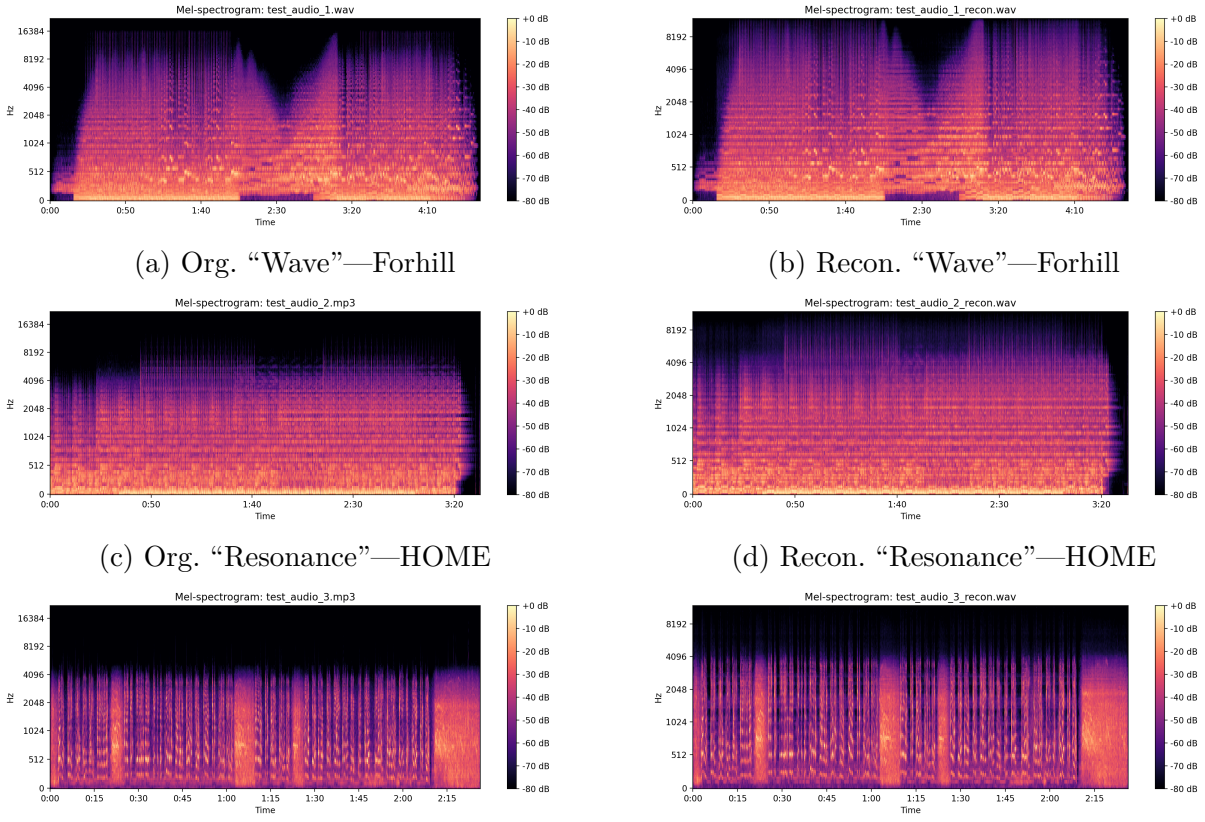
## 4.6 Reconstruction Accuracy

`SCE-FOAM` reconstruction accuracy for each modality was calculated using three text examples, three audio clips, and six image clips to calculate the average reconstruction accuracy for each modality. Example outputs appear side-by-side in table 4.6 for text, and figures 4.1 and 4.2 for audio and image, respectively.

Table 4.6 shows three examples of original and reconstruction comparisons of text. We scored an 84% BERTScore semantic similarity accuracy compared to the original texts. The model was able to capture major semantic cues (e.g., dates and proper names) and high-level meaning, but smaller semantic details proved challenging in syntactic dependencies among words and maintaining coherent logical flow.

In Fig. 4.1, we computed the Mel Spectrogram [93] for three clips varying in different lengths that were semantically represented and then reconstructed. The reconstruction accuracy we achieved scored 97% using normalized signal cross correlation. We notice that the reconstructed signals occupy fewer frequencies than the original clip, which results in reduced audio quality, with more noise and a less full-bodied sound due to the frequency drop.

In Fig. 4.2, we computed the LPIPS test to compare the originals with the reconstructed images. We averaged 0.0327, which means that there are significantly fewer semantic dif-




(a) Org. "Wave"—Forhill

(b) Recon. "Wave"—Forhill

(c) Org. "Resonance"—HOME

(d) Recon. "Resonance"—HOME

(e) Org. JFK inaugural speech

(f) Recon. JFK inaugural speech

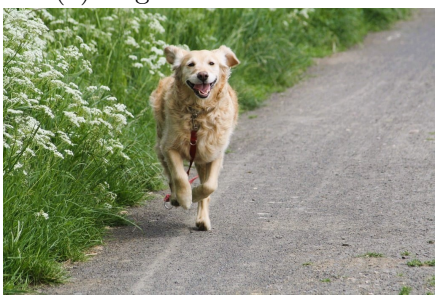
Figure 4.1: Comparison of Mel-spectrograms for original and reconstructed audio: (a, c, e) originals, "Wave" by Forhill, "Resonance" by HOME, and JFK's inaugural address; (b, d, f) their corresponding reconstructions.

ferences between the original and the reconstructed. We did notice that for finer details — similar to text — the model has a hard time relaying these concepts. This results in the reconstructions showing a blur effect that affects all the images. A sharpening of an image could be performed to mitigate this effect, such as the method discussed in [94], but it would result in more computation and latency, which might prohibit some real-time systems.

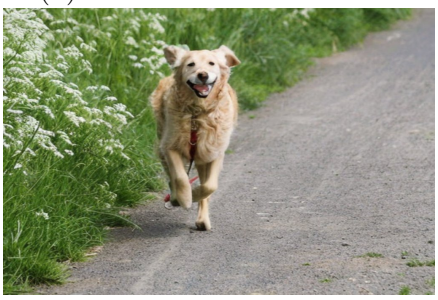
We achieve high reconstruction accuracy for each modality as the neural networks — especially the multi-head attention modules — learn nonlinear mappings that model both intra-modality structure and cross-modal dependencies, allowing for precise decoding back into the information space with low distortion and high fidelity.

(a) Org. NASA astronaut

(b) Recon. NASA astronaut



(c) Org. golden retriever



(d) Recon. golden retriever

(e) Org. Sáttítla Highlands

(f) Recon. Sáttítla Highlands

Figure 4.2: Comparison of original and reconstructed images: NASA astronaut spacewalk, golden retriever in the park, and an image of Sáttítla Highlands from the perspective of a satellite. (a, c, e) original images; (b, d, f) corresponding reconstructions.

## 4.7 Knowledge Base Lookup Latency

For our KB lookup latency experiment, we compiled different modalities and sent them sequentially through the framework's `send` method. We then queried *findClosestHexByVector* and calculated the time for the transmitter to look through the KB. As data is sent, our framework compares embedding vectors to see if they match and will store the results, which will cause the KB to grow in size. We decided to test different modalities separately for the

sake of this control experiment, but cross-modality is supported as well.

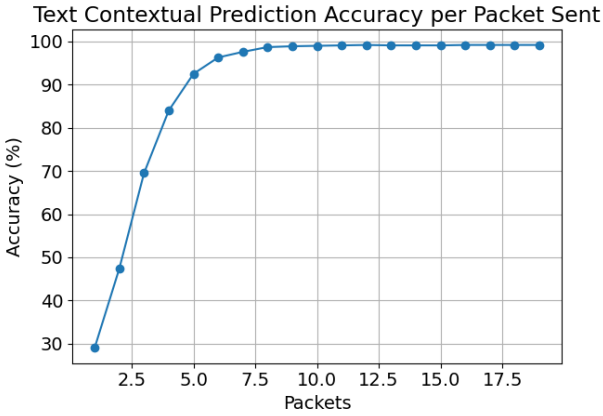
The evaluation from the KB lookup latency experiment shows a relatively linear trend as symbols get stored in the KB. The results indicate that our text, audio, and image lookup latencies averaged around $80\mu s$, $3840\mu s$, and $60\mu s$, respectively. The audio embedding demonstrated a rise in lookup latency compared to text and image modalities; this is most likely a result of the dynamic size code blocks for varying-sized audio clips.

The scaling trend was estimated by fitting a linear model to lookup latency as KB size increased. The fitted sloped approximated a rate per item added of $30\mu s$, $365\mu s$, and $2.5\mu s$ for text, audio, and image, respectively. These measurements were performed for single-modality KBs only, and due to the non-consistent nature of the code blocks, adding new modalities or varying the sequential sizes of semantic data will cause the KB to deviate from this linear trend because modality has varying vector dimensions and computational costs.

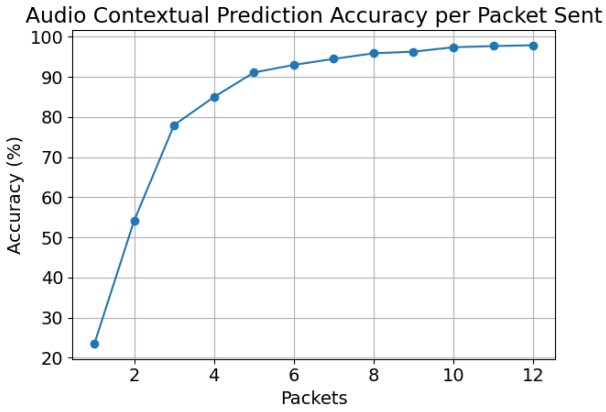
Some possible future research implementations for improving the latency would be to have an algorithm that, instead of computing the entirety of the embedding, only calculates the threshold specified. I.e., if the threshold is 90%, then checking only that percentage of the embedding would save computing time and possible lookup times, as the model wouldn't need to compute the entire cosine similarity between vectors. This would mean a new cosine BLAS function would be needed to make this possible.

## 4.8 Prediction Accuracy and Context-Recovery Example

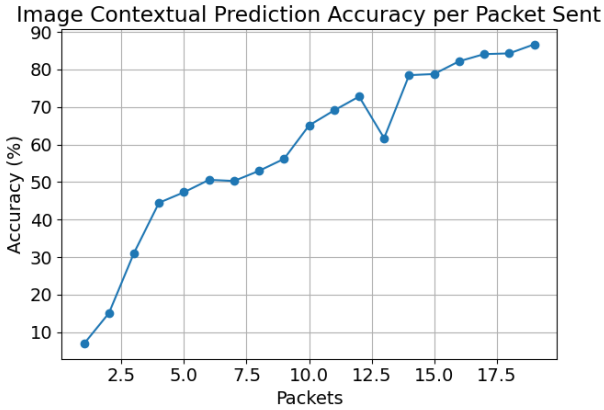
For our contextual predictions, we begin by running an evaluation that plots the accuracy of the prediction model while learning from a continuous stream of packets. We then investigate




(a) Text prediction accuracy




(b) Audio prediction accuracy




(c) Image prediction accuracy

Figure 4.3: Contextual prediction accuracies for text, audio, and image modalities.

different applications for the SCE-FOAM architecture with packet loss and frame restoration.

The contextual prediction accuracies are shown in Fig. 4.3. We plotted the prediction accuracy over successive packets sent. The packets were designed to be sent sequentially over several iterations, and then the cosine similarity between each predicted embedding and the true target was computed. Our model works on associational logic, which allows the model to learn a statistical relationship between contextual information.

With text and audio predictions, our model was able to achieve 95% accuracy after sending 8 sequential packets. For images, this increased by 18 packets and achieved a prediction

accuracy rate of 87%. Our results are significantly promising with average accuracies of 92.33% for all prediction models, which are not pretrained and rely purely on the information flow from each `SCE-FOAM` system. We also noticed that though images took twice as long to predict, they are also commonly transmitted at higher rates, such as 24, 30, 60 frames per second, which would easily make up for the extra packets needed to learn and predict.

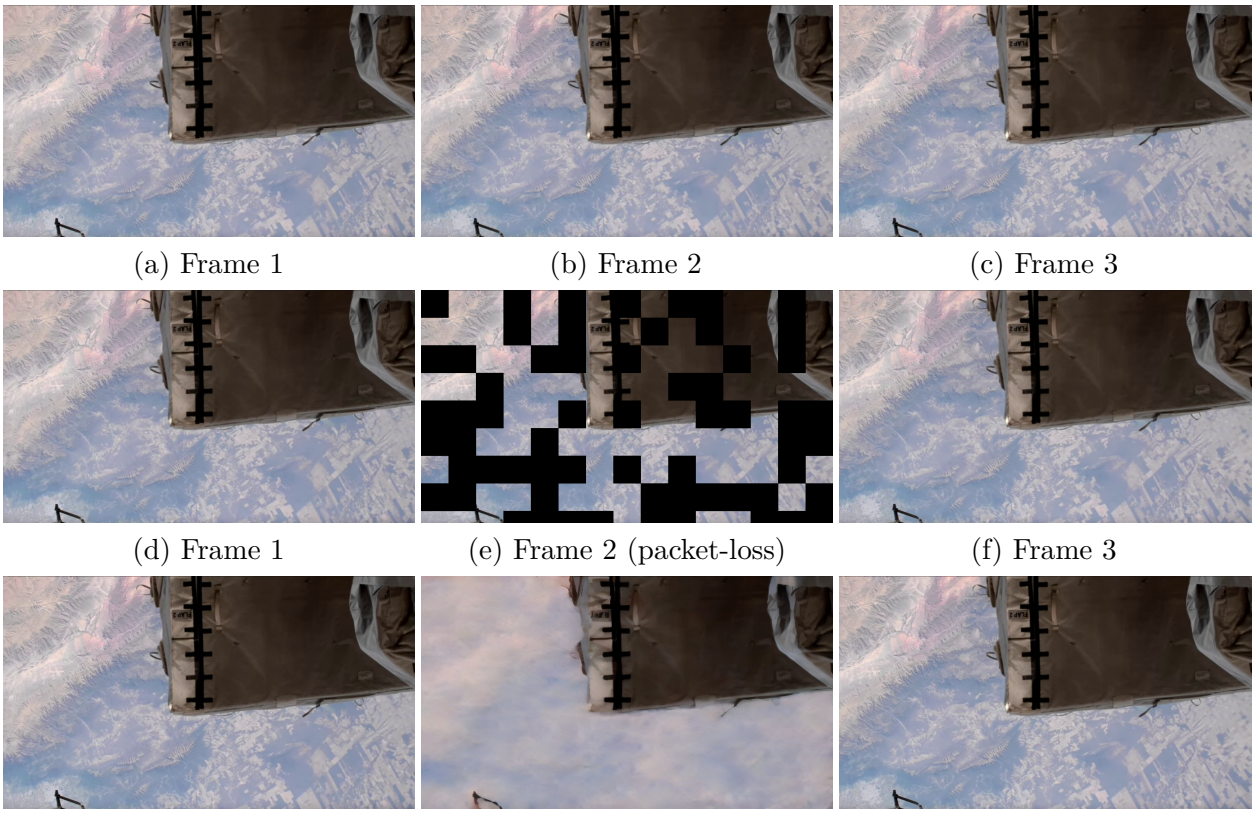
To visualize the prediction model and different applications, we compiled different scenarios, such as packet loss and frame restoration, that our framework can solve.

In Fig. 4.4, we demonstrate a possible application in which packet loss is introduced (e.g., due to a communication link outage or other factors). Our framework can then generate the missing or corrupted video frame from a contextual frame.

In our example, we used a video stream from NASA's International Space Station [78] and played the video feed 20 times with `sends/receives` and then had the model predict frame 2 based on frame 1 as context. By using this method, our model was able to predict frame 2 with relative clarity. Having a dedicated contextual image prediction model trained on the relationships would allow for zero-shot predictions in the future.

In Fig. 4.5, we illustrate an example of using our framework on a frame whose quality has been degraded by camera dynamics (e.g., motion blur from rapid panning or sudden exposure shifts [95]); in particular, example frame 2 is overexposed. By integrating a frame checker within the user device, our framework would be able to predict a contextual frame that would match previous frames, restoring the semantic quality sequentially. The video frames are from the New York Central Park [79].

These experiments used a FastFormer [36] model. For the forecasting, which leverages associational logic, predicts the statistical pattern within the data. This method requires a tremendous amount of data and training time to learn the statistical patterns between each



(a) Frame 1 (b) Frame 2 (c) Frame 3

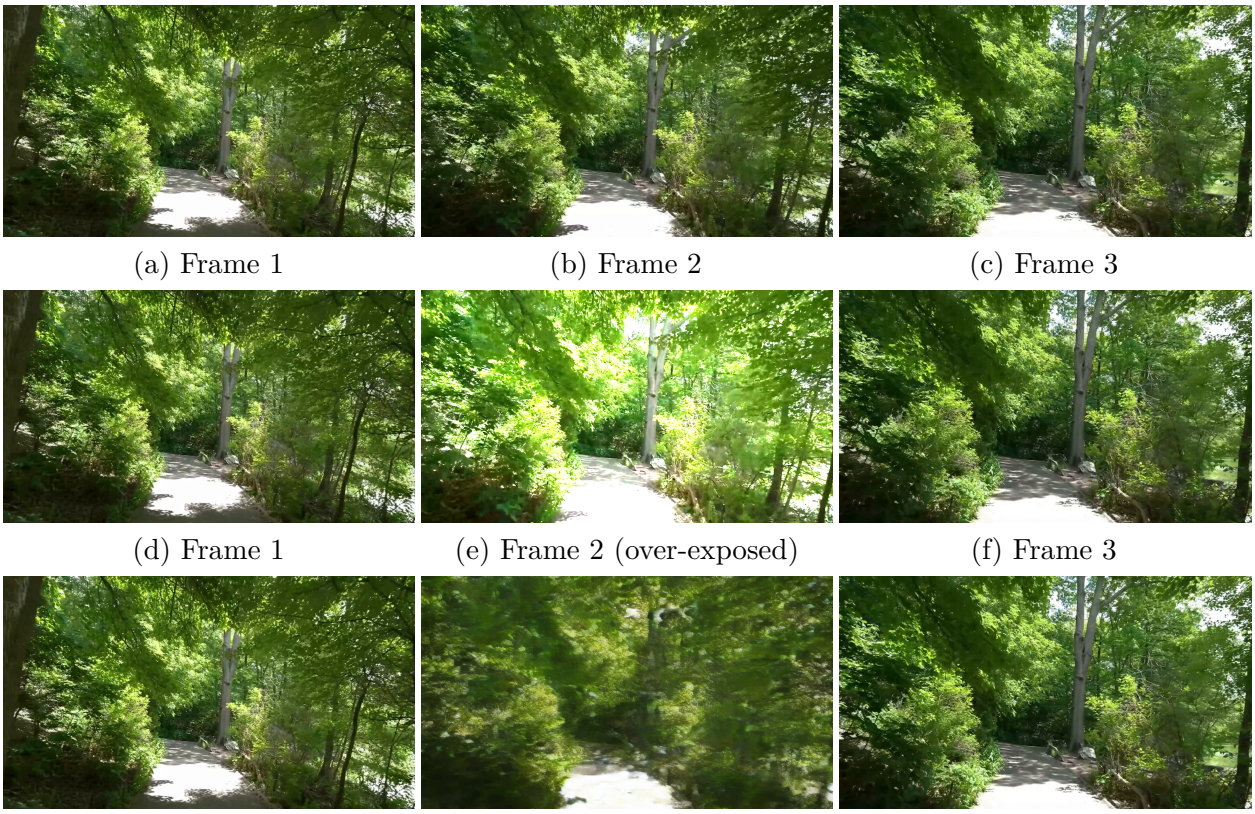
(d) Frame 1 (e) Frame 2 (packet-loss) (f) Frame 3

(g) Frame 1 (h) Frame 2 (predicted) (i) Frame 3

Figure 4.4: Illustration of using the prediction model to reconstruct a packet-lost frame in a NASA ISS feed: (a–c) raw control frames; (d–f) packet loss in frame 2; (g–i) predicted recovery of frame 2 using context from frame 1.

modality domain. This is due to the law of large numbers [96]; as the model obtains more data points, the statistical relationships will eventually come out.

In the future, we would like to implement a causal model into the translator predictor to understand the causal relationships within the data without relying purely on statistical methods. We anticipate that in the near future, there will be model architectures that will incorporate mechanisms for combined causal and associative inferences that will benefit the future iterations of this framework.



(a) Frame 1 (b) Frame 2 (c) Frame 3

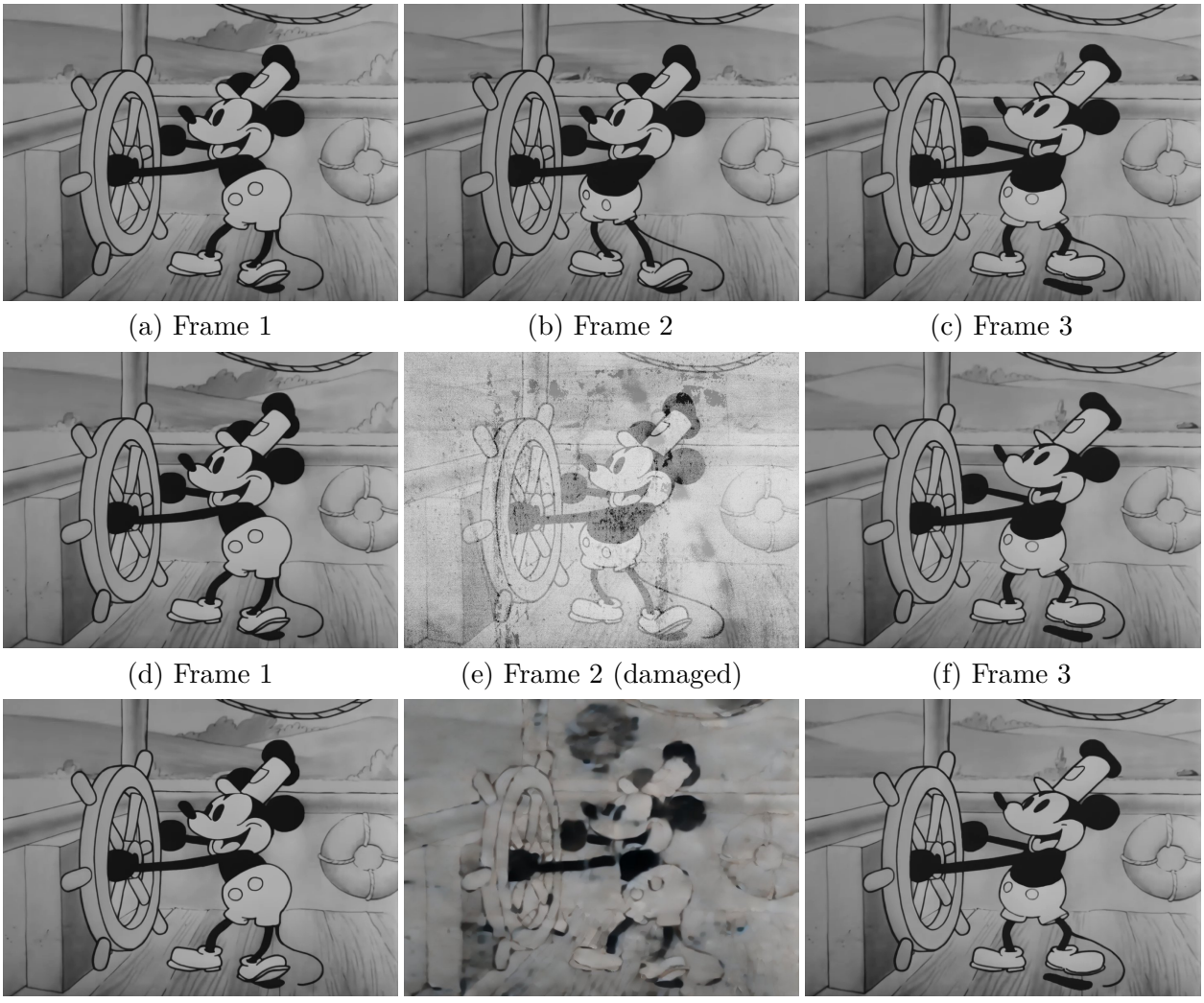
(d) Frame 1 (e) Frame 2 (over-exposed) (f) Frame 3

(g) Frame 1 (h) Frame 2 (predicted) (i) Frame 3

Figure 4.5: Illustration of the prediction model restoring an overexposed frame in a New York Central Park feed: (a–c) raw control frames; (d–f) frame 2 overexposed; (g–i) frame 2 reconstructed from frame 1 context.

## 4.9 Summary

In this chapter, we discussed the test data in section 4.1, the hardware and Jest scripts for each evaluation in section 4.2, each metric in section 4.3, the inference timing in section 4.4, the size comparisons of latent and codecs in section 4.5, reconstruction accuracies of each modality in section 4.6, KB lookup timing in section 4.7, and finally the prediction accuracies and demos in section 4.8 showing varying scenarios for `SCE-FOAM` framework.



(a) Frame 1 (b) Frame 2 (c) Frame 3

(d) Frame 1 (e) Frame 2 (damaged) (f) Frame 3

(g) Frame 1 (h) Frame 2 (predicted) (i) Frame 3

Figure 4.6: Illustration of the prediction model restoring a damaged frame in *Steamboat Willie*: (a–c) control frames; (d–f) frame 2 damaged; (g–i) frame 2 reconstructed from frame 1 as context.

# Chapter 5

# Heuristic Extension of Deferred Acceptance Algorithm

## 5.1 Overview

In addition to the `SCE-FOAM` framework, we also propose an extension to the DA algorithm [28] introduced by Gale and Shapley, whose original form is designed to find stable matching between two sets of participants, this matching structure guarantees a stable match and under certain games these matches are considered the Nash equilibrium [29]. The DA algorithm works by having one side, "Proposers," make proposals to the other side, the "Acceptors". This process is a one-to-one matching, meaning that it considers all the user's preferences, but each member of each group is only paired with one match. There have been extensions that allow for many-to-many matches, such as the Roth and Sotomayor algorithm [97]; however, this algorithm is not designed to adapt to dynamic networks.

Matching algorithms for wireless networks are important as they can find the best node depending on varying network constraints. The constraints can be defined as, efficiency (finding a lower latency), reliability (stability of the link), scalability (bandwidth capability of the node), energy saving (finding closer nodes rather then ones farther away), and fault tolerance (ability to find the best node when a network link degrades) [98–100]. With the constraints in mind, in our extension to the DA algorithm, we aimed to incorporate methods

to mitigate these constraints.

In our extension of the DA algorithm, we incorporate exploration, coverage, and diversification heuristics that are designed to extend the nodes matching capabilities. Exploration prevents the algorithm from getting matched with the same node by suggesting alternatives. Coverage ensures that all nodes are involved in the distribution of the load each round, without relying on just a few. Finally, diversification encourages nodes to match with new nodes, potentially increasing bandwidth. It is also essential to note that our internal tests indicate the algorithm does not converge to a global stable pairing; however, future work can be planned to address this issue while exploring a heuristic approach to the DA algorithm, which has not been previously explored. In the next chapter, we will explain how the DA algorithm works, what we contribute to the extension, while showing some empirical results from some experiments we performed.

## 5.2 Extension of the Algorithm

In the DA algorithm 1, a list of preferences is first compiled. In our example 5.1, we have two tables that show the preferences of each group, which are wireless nodes. Each proposor who is free proposes to their most preferred acceptor. Each acceptor tentatively accepts the most preferred proposal and rejects any less-preferred proposals. This process repeats until no proposor can make a new proposal. What is left is a stable and optimal matching for proposors. The resulting algorithm is said to have a time complexity of $O(n^2)$, which means that for every node $n$ that is added, the computational resources grow quadratically. This is similar to our extension as it adds some more steps but stays within the same time complexity.

In our extension, which can be seen in 2 and 5.2. The extension first breaks up node

Table 5.1: One-to-one DA algorithm before and after.

**Before**

**Proposer**

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 4 |
| 2 | 4 | 1 | 2 | 3 |
| 3 | 1 | 3 | 2 | 4 |
| 4 | 2 | 3 | 1 | 4 |

**Acceptor**

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 3 | 2 | 4 |
| 2 | 3 | 4 | 1 | 2 |
| 3 | 4 | 2 | 3 | 1 |
| 4 | 3 | 2 | 1 | 4 |

⇒

**After**

**Proposer**

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 4 |
| 2 | 4 | 1 | 2 | 3 |
| 3 | 1 | 3 | 2 | 4 |
| 4 | 2 | 3 | 1 | 4 |

**Acceptor**

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 3 | 2 | 4 |
| 2 | 3 | 4 | 1 | 2 |
| 3 | 4 | 2 | 3 | 1 |
| 4 | 3 | 2 | 1 | 4 |

**Algorithm 1** DA algorithm.

**Require:** Finite sets of proposers $P$ and acceptors $A$.
For each proposer $p \in P$, a strict preference order $\succ_p$ over a subset of $A$.
For each acceptor $a \in A$, a strict preference order $\succ_a$ over a subset of $P$.
**Ensure:** Stable matching exist between $P$ and $A$
$M \leftarrow \emptyset$ ▷ $M$ are the matches (P,A)
Mark every $p \in P$ as FREE; for each $p$, keep a list of partners not yet proposed to.
**while exists** free $p \in P$ with someone left to propose to **do**
    $a \leftarrow$ highest-ranked unproposed partner of $p$ in $\succ_p$
    $p$ **proposes** to $a$
    **if** $a$ is unmatched in $M$ **then**
        $M \leftarrow M \cup \{(p, a)\}$ ▷ $a$ tentatively accepts i.e. match
    **else**
        Let $p$ be the current partner ▷ $p'$ is the potential new partner
        **if** $p' \succ_a p$ **then** ▷ $a$ prefers $p'$ over current partner
            $M \leftarrow \big(M \setminus \{(p, a)\}\big) \cup \{(p', a)\}$ ▷ Removes $(p, a)$, unions $(p', a)$ with $M$
            Mark $p$ as FREE ▷ $a$ rejects $p$
        **else**
            $p'$ remains FREE ▷ $a$ rejects $p'$
        **end if**
    **end if**
**end while**
**return** $M$

preferences into columns and treats each column as a round. Then, the DA algorithm is run for the entire table. In each consecutive round, the first column is popped from the table. This process repeats until no more columns are left. There are also constraints on what values can be chosen when iterating through columns. First, if a preference was chosen

Table 5.2: Many-to-many extension of DA algorithm before and after.

**Before**

**Proposer**

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 4 |
| 2 | 4 | 1 | 2 | 3 |
| 3 | 1 | 3 | 2 | 4 |
| 4 | 2 | 3 | 1 | 4 |

**Acceptor**

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 3 | 2 | 4 |
| 2 | 3 | 4 | 1 | 2 |
| 3 | 4 | 2 | 3 | 1 |
| 4 | 3 | 2 | 1 | 4 |

⇒

**After**

**Proposer**

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 4 | 3 |
| 2 | 4 | 3 | 2 | 2 |
| 3 | 3 | 2 | 3 | 4 |
| 4 | 2 | 1 | 1 | 1 |

**Acceptor**

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 4 | 4 | 4 |
| 2 | 4 | 3 | 2 | 2 |
| 3 | 3 | 2 | 3 | 1 |
| 4 | 2 | 1 | 1 | 3 |

in a previous round, it can't be chosen again(diversification). It secondly records what choices were taken and enforces per-round uniqueness (coverage). Thirdly, if the algorithm encounters a stalemate in a selection, then it will choose randomly from acceptors that have not been used in the round and have not been used by the current proponent(exploration).

In the table 5.2, showcasing the before and after of running the custom heuristic-driven DA algorithm on the preference tables for proposers and acceptors. In the case of this table, the green and grey suggestions are all colored as they've been recalculated and placed for the algorithm's results, showing they meet the criteria of a heuristic.

Empirical testing with a $4 \times 4$ table showed that our extension explored 0.41 more newer matches compared to DA, indicating balanced exploration, which was neither too constrained (>0.25), nor too exploitative (>0.75). We also showed perfect coverage at 100% as our algorithm purposely matched each round. Lastly, our diversification scored 1.19, lower than DA's 1.345, which were both calculated using Shannon entropy [6]. When computing the combined scores using $CD = \alpha C + (1 - \alpha)D$, where $\alpha$ is the weighting parameter, we outperform DA, getting 0.93 for our heuristic-driven extension while vanilla DA gets 0.81.

**Algorithm 2** Heuristic-driven extension of the DA algorithm.

**Require:** Finite sets of proposers $P$ and acceptors $A$.
For each $p \in P$, a strict preference order $\succ_p$ over a subset of $A$.
For each $a \in A$, a strict preference order $\succ_a$ over a subset of $P$.
**Ensure:** Stable matching exists between $P$ and $A$
Split preferences of each $p$ and $a$ into $c \in C$ **columns** (rounds).
$M \leftarrow \emptyset$ ▷ $M$ are the matches $(P, A)$ for all rounds
Mark every $p \in P$ as FREE; for each $p$, keep a list of partners not yet proposed to.
**while exists** $c \in C$ **do**
    *used_this_round* $\leftarrow \emptyset$
    **while exists** free $p \in P$ with someone left to propose to **do**
        $a \leftarrow$ highest-ranked partner of $p$ not yet in $p$'s history ▷ row-wise `diversification`; fallback to last-ranked or random for `exploration`
        $p$ **proposes** to $a$
        **if** $a$ is unmatched in $M$ **then**
            $M \leftarrow M \cup \{(p, a)\}$ ▷ $a$ tentatively accepts i.e. match
        **else**
            Let $p$ be the current partner; $p'$ the potential new partner
            **if** $p' \succ_a p$ **then** ▷ $a$ prefers $p'$ over current partner
                $M \leftarrow \big(M \setminus \{(p, a)\}\big) \cup \{(p', a)\}$ ▷ Removes $(p, a)$, unions $(p', a)$ with $M$
                Mark $p$ as FREE ▷ $a$ rejects $p$
            **else**
                $p'$ remains FREE ▷ $a$ rejects $p'$
            **end if**
        **end if**
        Record $a$ in $p$'s history and mark $a$ as used in *used_this_round* ▷ enforces row-wise `diversification` and column-wise `coverage` within this round
    **end while**
    Remove current column $c$ from each $P$ and $A$ ▷ Advance to next round
**end while**
**return** $M$

## 5.3 Summary

In this chapter, we discussed the problem with wireless node matching algorithms in section 5.1. The DA algorithm, the heuristic extension we proposed, some examples illustrating each algorithm, and concluded with empirical results in section 5.2. In the next chapter 6, we will conclude with final thoughts for our work and future work.

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

In this thesis, we developed `SCE-FOAM`, a semantic context-aware framework for adaptive multimodal reasoning that achieves up to 46.86% reduction of transmitted bits compared to conventional codecs and various different semantic latent models. We have shown that our contextual prediction models can predict and restore missing information with accuracies up to 92.33%. The modular design of our microservice-based architecture enables researchers and software developers to easily integrate and maintain their research contributions. The microservice-based architecture also allows for efficiency, as modality translators can be spun up or down depending on the semantic communication needs. The lexicon knowledge base efficiently stores and transmits references for semantic symbols that are frequently exchanged over a communication channel. This reduces the bandwidth on the communication channel and allows for orders of magnitude smaller transmission compared to conventional redundant methods. We also introduced a heuristic extension to the DA algorithm that achieves exploration, coverage, and diversification in matching semantic network nodes. These contributions enable multimodal reasoning, bandwidth reduction, modularity, and efficient network node matching, thereby facilitating the widespread adoption of semantic communication. In the next section 6.2, we will discuss future work.

## 6.2 Future Work

Looking ahead, we created a list of future semantic works for `SCE-FOAM`, which includes:

- **New modalities support:** The integration of more modalities, such as virtual reality (VR) and augmented reality (AR), to support the growing demand for new media.
- **New text model:** Investigation of a new text model that can be bandwidth efficient with small and large text corpora compared to SONAR.
- **Using pretrained prediction models:** The integration of a pretrained model would generalize better for domain-specific applications, which may further boost prediction fidelity and reduce training overhead.
- **Implementing multi-user networking:** Moving away from a single link implementation of `SCE-FOAM` and updating the framework for multi-user applications.
- **Integrating architecture privacy:** Researching ways to integrate privacy features within the semantic knowledge base and symbol creation to prevent misuse.
- **Globally stable matching:** Researching mathematical revisions for our heuristic extension of DA to support globally stable matches for multi-user applications.

These future works are the next steps in our semantic communication framework. Implementing new modalities allows for more applications. Finding better text models will enable better bandwidth efficiency. Using pretrained prediction models will mitigate the need for heavy training. Multi-user semantics will facilitate widespread adoption. Architecture privacy will prevent misuse in multi-user applications, and finally, globally stable matching will allow for more efficient network node pairings. We believe these future works will contribute to the next phase in our semantic communication framework.

# Bibliography


[1] Ericsson. *Ericsson Mobility Report: June 2025*. Tech. rep. Forecast includes global mobile network data traffic growth, projecting a 17% CAGR (2024–2030) with annual growth slowing to 15% by 2030. Ericsson, June 2025. URL: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast.

[2] Ericsson. *Ericsson Mobility Report – Mobile Data Traffic Forecast*. June 2025.

[3] Cisco Systems. *Cisco Annual Internet Report (2018–2023) White Paper*. 2020.

[4] Diane Mievis. *AI Is Scaling Fast, and So Must Our Networks and Policies*. Technical Blog, Cisco Government and Education. June 2025. URL: https://blogs.cisco.com/gov/ai-scaling-networks-connectivity-eu-policy.

[5] Cisco Systems. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020*. White Paper. Feb. 2016.

[6] C. E. Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3 (July 1948), pp. 379–423.

[7] ITU-T and ISO/IEC JTC1. *Advanced Video Coding for Generic Audiovisual Services*. ITU-T Recommendation H.264 and ISO/IEC 14496-10. ITU-T and ISO/IEC JTC1, May 2003.

[8] ISO/IEC. *Information Technology—Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s—Part 3: Audio*. Tech. rep. ISO/IEC 11172-3:1993. ISO/IEC, Dec. 1993.

[9] ISO/IEC. *Digital Compression and Coding of Continuous-Tone Still Images—Part 1: Requirements and Guidelines*. Tech. rep. ISO/IEC 10918-1:1994. ISO/IEC, Feb. 1994.

[10] Jialong Xu et al. "Deep Joint Source–Channel Coding for Semantic Communications". In: *IEEE Communications Magazine* 61.11 (Nov. 2023), pp. 42–48.

[11] Zhonghao Lyu et al. "Semantic Communications for Image Recovery and Classification via Deep Joint Source and Channel Coding". In: *IEEE Transactions on Wireless Communications* 23.8 (Apr. 2023), pp. 8388–8404.

[12] Guangyi Zhang et al. "Learned Image Transmission with Hierarchical Variational Autoencoder". In: *Proceedings of the 39th AAAI Conference on Artificial Intelligence*. 2025, pp. 13215–13223. DOI: 10.1609/aaai.v39i12.33442.

[13] Joohyuk Park et al. "Joint Source-Channel Coding for Channel-Adaptive Digital Semantic Communications". In: *arXiv preprint arXiv:2311.08146* (2024).

[14] Ecenaz Erdemir et al. "Generative Joint Source-Channel Coding for Semantic Image Transmission". In: *IEEE Journal on Selected Areas in Communications* 41.8 (Nov. 2022), pp. 2645–2657. DOI: 10.1109/JSAC.2023.3288243.

[15] Christina Chaccour et al. "Less Data, More Knowledge: Building Next Generation Semantic Communication Networks". In: *arXiv* (Nov. 2022). DOI: 10.48550/arXiv.2211.14343. arXiv: 2211.14343 [cs.IT].

[16] Christina Chaccour and Walid Saad. "Disentangling Learnable and Memorizable Data via Contrastive Learning for Semantic Communications". In: *arXiv* (Dec. 2022). DOI: 10.48550/arXiv.2212.09071. arXiv: 2212.09071 [cs.IT].

[17] Shaolong Guo and et al. "A Survey on Semantic Communication Networks". In: *IEEE Access* (2024). eprint: 2405.01221.

[18] Dylan Wheeler and Balasubramaniam Natarajan. "Engineering Semantic Communication: A Survey". In: *IEEE Access* 11 (2022), pp. 13965–13995. DOI: 10.1109/ACCESS.2023.3243065.

[19] Feibo Jiang et al. "Large AI Model Empowered Multimodal Semantic Communications". In: *IEEE Communications Magazine* 62.4 (Apr. 2024), pp. 56–62.

[20] Guangyi Zhang et al. "A Unified Multi-Task Semantic Communication System for Multimodal Data". In: *IEEE Transactions on Communications* 72.7 (July 2024), pp. 4101–4116.

[21] Haonan Tong et al. "Multimodal Semantic Communication for Generative Audio-Driven Video Conferencing". In: *IEEE Wireless Communications Letters* 13.2 (Feb. 2024), pp. 410–414.

[22] Guangyuan Liu et al. "Context-Aware Semantic Communication for the Wireless Networks". In: *arXiv preprint arXiv:2505.23249* (May 2025).

[23] Yusong Zhang et al. "Multimodal LLM Integrated Semantic Communications for 6G Immersive Experiences". In: *arXiv preprint arXiv:2507.04621* (July 2025).

[24] Mingkai Chen et al. "Cross-Modal Graph Semantic Communication Assisted by Generative AI in the Metaverse for 6G". In: *Research* 7 (Apr. 2024), p. 0342.

[25] Chaowei Wang et al. "Multimodal Semantic Communication Accelerated Bidirectional Caching for 6G MEC". In: *Future Generation Computer Systems* 140 (Mar. 2023), pp. 225–237.

[26] Jiayi Lu et al. "Generative Artificial Intelligence-Enhanced Multimodal Semantic Communication in Internet of Vehicles: System Design and Methodologies". In: *IEEE Vehicular Technology Magazine* 20.2 (June 2025), pp. 71–82.

[27] Fuhui Zhou et al. “Cognitive Semantic Communication Systems Driven by Knowledge Graph: Principle, Implementation, and Performance Evaluation”. In: *IEEE Transactions on Communications* 72.1 (Jan. 2023), pp. 193–208.

[28] David Gale and Lloyd S. Shapley. “College Admissions and the Stability of Marriage”. In: *The American Mathematical Monthly* 69.1 (1962), pp. 9–15. DOI: 10.2307/2312726.

[29] John Nash. “Equilibrium points in n-person games”. In: *Proceedings of the National Academy of Sciences* 36.1 (1950), pp. 48–49. DOI: 10.1073/pnas.36.1.48.

[30] Rudolf Carnap and Yehoshua Bar-Hillel. “An Outline of a Theory of Semantic Information”. In: *Journal of Symbolic Logic* 19.3 (Sept. 1952), pp. 230–232.

[31] Luciano Floridi. “Outline of a Theory of Strongly Semantic Information”. In: *Minds and Machines* 14.2 (May 2004), pp. 197–221.

[32] David A. Huffman. “A Method for the Construction of Minimum-Redundancy Codes”. In: *Proceedings of the IRE* 40.9 (1952), pp. 1098–1101. DOI: 10.1109/JRPROC.1952.273898.

[33] Jacob Ziv and Abraham Lempel. “A Universal Algorithm for Sequential Data Compression”. In: *IEEE Transactions on Information Theory* 23.3 (1977), pp. 337–343. DOI: 10.1109/TIT.1977.1055714.

[34] Adrien-Marie Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. Firmin Didot, 1805.

[35] William K. Pratt. *Peak Signal-to-Noise Ratio (PSNR)*. Standard image quality metric, widely used in image processing and compression research. 1974.

[36] Chuhan Wu et al. “Fastformer: Additive Attention Can Be All You Need”. In: *arXiv* (Aug. 2021). DOI: 10.48550/arXiv.2108.09084. arXiv: 2108.09084 [cs.CL].

[37] Ashish Vaswani et al. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 5998–6008.

[38] Jacob Devlin et al. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding”. In: *Proc. NAACL-HLT 2019*. 2019, pp. 4171–4186. DOI: 10.48550/arXiv.1810.04805.

[39] Alec Radford et al. *Improving Language Understanding by Generative Pre-Training*. Technical Report. OpenAI, 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[40] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *arXiv* (Jan. 2013). DOI: 10.48550/arXiv.1301.3781. eprint: 1301.3781.

[41] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “GloVe: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543.

[42] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*. Vol. 139. PMLR. 2021, pp. 8748–8763. URL: https://proceedings.mlr.press/v139/radford21a.html.

[43] Paul-Ambroise Duquenne, Holger Schwenk, and Benoît Sagot. “SONAR: Sentence-Level Multimodal and Language-Agnostic Representations”. In: *arXiv* (Aug. 2023). DOI: 10.48550/arXiv.2308.11466. eprint: 2308.11466.

[44] *American National Standard for Information Systems—Coded Character Set—7-Bit American National Standard Code for Information Interchange*. ANSI X3.4–1986 (R1997). New York, NY: American National Standards Institute, 1986.

[45] Alexandre Défossez et al. “High Fidelity Neural Audio Compression”. In: *arXiv* (Oct. 2022). eprint: 2210.13438. URL: https://arxiv.org/abs/2210.13438.

[46] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *arXiv* (Dec. 2013). DOI: 10.48550/arXiv.1312.6114. arXiv: 1312.6114 [stat.ML].

[47] Michael I. Jordan et al. “An Introduction to Variational Methods for Graphical Models”. In: *Machine Learning* 37.2 (1999), pp. 183–233. DOI: 10.1023/A:1007665907178.

[48] Stability AI. *sd-vae-ft-ema*. Hugging Face Model Card. 2022. URL: https://huggingface.co/stabilityai/sd-vae-ft-ema.

[49] Robin Rombach and et al. “High-Resolution Image Synthesis with Latent Diffusion Models”. In: *arXiv:2112.10752* (2022).

[50] John G. Proakis and Masoud Salehi. *Digital Communications*. 5th. McGraw-Hill, 2007. ISBN: 978-0072957167.

[51] Charles L. Lawson et al. “Basic Linear Algebra Subprograms for Fortran Usage”. In: *ACM Transactions on Mathematical Software* 5.3 (1979), pp. 308–323. DOI: 10.1145/355841.355847.

[52] S. Nicolson et al. *Free Lossless Audio Codec (FLAC)*. 2024.

[53] I. Gonçalves et al. *Ogg Media Types*. RFC 5334, Internet Engineering Task Force. 2008.

[54] IETF Codec Working Group. *Opus Audio Codec*. 2012.

[55] IBM Corporation and Microsoft Corporation. *Multimedia Programming Interface and Data Specifications 1.0*. Tech. rep. IBM and Microsoft, 1991. URL: https://www.mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/Docs/riffmci.pdf.

[56] Alliance for Open Media (AOMedia) and ISO/IEC. *Image File Format for AV1 (AVIF), ISO/IEC 23000-22:2020*. 2020.

[57] Joint Photographic Experts Group. *JPEG XL Image Coding System*. https://jpeg.org/jpegxl/. 2021.

[58] World Wide Web Consortium. *Portable Network Graphics (PNG)*. 2003.

[59] Google Inc. *WebP Image Format*. 2023.

[60] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv preprint arXiv:1301.3781* (2013).

[61] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 135–146.

[62] Alexandre Défossez et al. "High Fidelity Neural Audio Compression". In: *arXiv preprint*. 2022. URL: https://arxiv.org/abs/2210.13438.

[63] Descript Inc. *Descript Audio Codec (DAC)*. 2022.

[64] Neil Zeghidour et al. "SoundStream: An End-to-End Neural Audio Codec". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2022), pp. 495–507. DOI: 10.1109/TASLP.2021.3129994.

[65] Dailan He et al. "ELIC: Efficient Learned Image Compression with Unevenly Grouped Space-Channel Contextual Adaptive Coding". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 5718–5727. URL: https://arxiv.org/abs/2203.10886.

[66] Johannes Ballé et al. "Variational Image Compression with a Scale Hyperprior". In: *International Conference on Learning Representations (ICLR)*. 2018. URL: https://arxiv.org/abs/1802.01436.

[67] David Minnen, Johannes Ballé, and George Toderici. "Joint Autoregressive and Hierarchical Priors for Learned Image Compression". In: *Advances in Neural Information Processing Systems*. 2018.

[68] Zhengxue Cheng et al. “Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 7939–7948. URL: https://arxiv.org/abs/2001.01568.

[69] Continental Congress. *The Declaration of Independence*. National Archives, United States. 1776. URL: https://www.archives.gov/founding-docs/declaration-transcript.

[70] King of England John. *Magna Carta*. Issued at Runnymede, 15 June 1215. 1215. URL: https://www.bl.uk/collection-items/magna-carta-1215.

[71] Library of Congress. *The Gettysburg Address*. https://hdl.loc.gov/loc.gdc/gdcwebcasts.140106ipo0730. Mar. 2014.

[72] HOME. *Resonance*. SoundCloud. 2014. URL: https://soundcloud.com/home/resonance.

[73] Forhill. *Wave*. SoundCloud. Mar. 2018. URL: https://soundcloud.com/forhill/wave.

[74] CBS. *JFK's Famous Inaugural Address Passage*. https://www.youtube.com/watch?v=mxa4HDgfWFs. 2025.

[75] NASA. *GMT031_08_56_Butch-Wilmore_EVA-92-D5 Camera Download 3*. NASA Image and Video Library. 2025. URL: https://www.nasa.gov/image-detail/gmt031_08_56_butch-wilmore_eva-92-d5-camera-download-3/.

[76] Itoldya420 Archive. *Golden Retriever on White Background*. https://itoldya420.getarchive.net/amp/media/dog-golden-retriever-animals-a6bb20. 2025.

[77] NASA Earth Observatory / Visible Earth. *Lava Flows of the Sáttítla Highlands*. https://visibleearth.nasa.gov/images/153845/lava-flows-of-the-sattitla-highlands/1538471. Jan. 2025.

[78] NASA. *Live High-Definition Views from the International Space Station*. https://www.youtube.com/live/H999s0P1Er0?si=3JIQuobrzBQK59by. Mar. 2025.

[79] Prowalk Tours. *Central Park, New York Walking Tour 4K60fps with Captions | Over 2.8 Million Views!* https://youtu.be/0AU9_H47w7k?si=wK976yozdoD7megK. July 2023.

[80] Walt Disney and Ub Iwerks of Walt Disney Films. *Steamboat Willie*. 1928.

[81] Mads Lauridsen et al. "From LTE to 5G for Connected Mobility". In: *IEEE Communications Magazine* 55.3 (Mar. 2017), pp. 78–85. DOI: 10.1109/MCOM.2017.1600778CM.

[82] Canonical Ltd. *Ubuntu – Linux for Everyone*. https://ubuntu.com/. 2004.

[83] Microsoft Corporation. *Windows Subsystem for Linux (WSL)*. https://docs.microsoft.com/windows/wsl/. 2016.

[84] OpenJS Foundation. *Node.js*. https://nodejs.org/. 2009.

[85] npm, Inc. *npm – Node.js Package Manager*. https://www.npmjs.com/. 2010.

[86] Meta Platforms, Inc. *Jest – Delightful JavaScript Testing*. https://jestjs.io/. 2020.

[87] Tianyi Zhang et al. *BERTScore: Evaluating Text Generation with BERT*. Apr. 2019. URL: https://arxiv.org/abs/1904.09675.

[88] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. 2nd. Upper Saddle River, NJ: Prentice Hall, 1999. ISBN: 0-13-754920-2.

[89] Richard Zhang et al. “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. June 2018, pp. 586–595.

[90] PyTorch Contributors. *PyTorch*. 2016. URL: https://pytorch.org/.

[91] Sohu Corporation and Etched. *Etched*. https://www.etched.com/. 2024.

[92] Alec Radford et al. *Language Models are Unsupervised Multitask Learners*. OpenAI Blog. Feb. 2019. URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[93] Brian McFee et al. “librosa: Audio and Music Signal Analysis in Python”. In: *Proceedings of the 14th Python in Science Conference*. 2015, pp. 18–25. URL: https://conference.scipy.org/proceedings/scipy2015/pdfs/mcfee.pdf.

[94] Xintao Wang et al. “Real-ESRGAN: Training Real-World Blind Super-Resolution With Pure Synthetic Data”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. 2021.

[95] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 2nd. Springer, 2022. DOI: 10.1007/978-3-030-34372-9. URL: https://szeliski.org/Book/.

[96] Jakob Bernoulli. *Ars Conjectandi*. Thurneysen Brothers, 1713.

[97] Alvin E. Roth and Marilda A. Oliveira Sotomayor. *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*. Econometric Society Monographs. Cambridge, UK: Cambridge University Press, 1990. ISBN: 9780521437882.

[98] Usman Mahmood Malik et al. “Stable Matching-Based Resource Allocation in Fog Computing for IoT Applications”. In: *Mathematics* 11.17 (2023), p. 3798. DOI: 10.3390/math11173798.

[99] Monowar Hasan and Ekram Hossain. "Distributed Resource Allocation for Relay-Aided Device-to-Device Communication: A Stable Matching Approach". In: *IEEE Transactions on Wireless Communications* 13.10 (2015), pp. 5275–5288. DOI: 10.1109/TWC.2014.2334636.

[100] Yunan Gu et al. "Matching theory for future wireless networks: fundamentals and applications". In: *IEEE Communications Magazine* 53.5 (2015), pp. 52–59. DOI: 10.1109/MCOM.2015.7105644.