Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

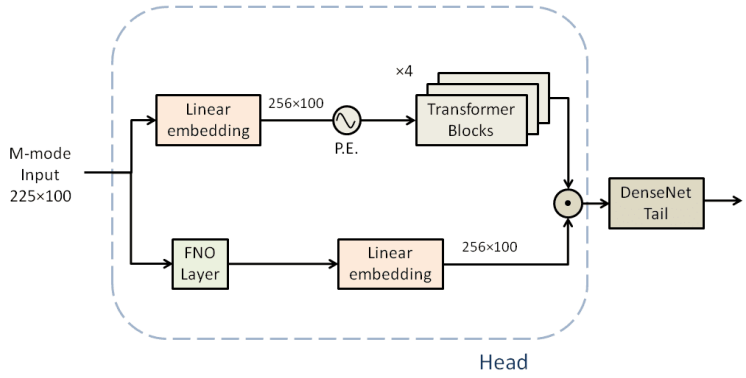
Institut für Integrierte Systeme
Integrated Systems Laboratory

Department of Information Technology and
Electrical Engineering

Autumn Semester 2025

Pretraining and Quantization Techniques for Ultrasound Data Compression

Master Project



Christos Dimopoulos
cdimopoulos@student.ethz.ch

October 2025

Supervisors: Dr. Christoph Leitner, christoph.leitner@iis.ee.ethz.ch
Dr. Yawei Li, yawei.li@vision.ee.ethz.ch

Professor: Prof. Dr. Luca Benini, luca.benini@iis.ee.ethz.ch

Acknowledgements

I sincerely thank my supervisors Christoph Leitner and Yawei Li for their guidance and understanding during the project, as they were proven valuable assets to its implementation. I also feel the need to thank my family and friends for being by my side during this academic journey, giving me courage, fond memories and much needed support. Special thanks to my friend Niki, for her insightful whiteboard sessions over the last couple years. Lastly, I owe gratitude to the Foundations Latsis, Bodossakis, as well as the Union of Greek Shipowners, for their financial aid through scholarships, that made the acquisition of this Masters title feasible.

Abstract

Ultrasound is a common imaging modality in medicine and is being explored for non-medical applications such as continuous vital-sign monitoring and gesture recognition. The development of flexible and wearable ultrasound devices opens up new healthcare and human–computer interaction scenarios. However, their widespread adoption is limited by the computational and memory requirements of traditional processing pipelines. This thesis tackles the challenge of ultrasound data compression for resource-constrained wearable devices by investigating pretraining strategies and quantization techniques for efficient machine learning models.

For this purpose, we design a hybrid architecture combining Transformers and Fourier Neural Operators (FNOs) to capture both temporal and frequency-domain representations of M-mode ultrasound signals. A systematic study on input embeddings, positional encoding, and fusion strategies is conducted to optimize feature extraction. To further improve training, we introduce a contrastive loss based on a multi-scale short-time Fourier transform discriminator and a frequency-domain reconstruction term.

For efficiency, we evaluate post-training quantization (PTQ) and quantization-aware training (QAT) at multiple bit widths (integer and floating-point 8/4-bit), and assess their impact on accuracy, compression ratio, and latency. Experiments show that carefully chosen pretraining and quantization strategies can significantly reduce model size while preserving reconstruction quality.

Overall, this work provides a comprehensive evaluation of model architectures, pretraining approaches, and quantization methods for ultrasound compression. The results give practical insights towards deploying lightweight, high-performance models on embedded ultrasound hardware, and advancing wearable and energy-efficient medical technologies.

Declaration of Originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor. For a detailed version of the declaration of originality, please refer to Appendix B

Christos Dimopoulos,
Zurich, October 2025

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Motivation

Ultrasound imaging has long been established as a cornerstone technology for medical diagnostics, health and bioengineering sciences, due to its non-invasive nature, low cost and high accuracy [1]. Recently, applications of ultrasound technologies have extended beyond the sphere of healthcare, finding applications through lightweight wearable devices that target several tasks, such as continuous monitoring of vital signs [2], muscle activity tracking and gesture recognition [3]. Breakthrough in these domains was vital, due to the cutting-edge advances and flexibility of transducer technology, that makes integration of ultrasound sensors into wristbands or patches feasible.

Nonetheless, a computationally and memory efficient implementation of ultrasound signal acquisition and processing for wearable devices is far from its realization. Traditionally, imaging pipelines are demanding on their resources, for beamforming, image reconstruction and post-processing analysis. On the other hand, wearable devices operate on low-power embedded hardware, with limited capacities, posing strong limitations on energy requirements, latency and model capacity. As such, integration of real-time, continuous monitoring on the edge, necessitates for powerful yet compact compression and quantization of ultrasound data.

A solution to this trade-off can be found on recent developments in modern deep learning architectures. Transformers and Fourier Neural Operators (FNOs) have shown strong potential in modeling and compressing sequential data, whilst preserving rich context in the frequency domain. Contrastive training pipelines that leverage adversary techniques have also demonstrated promising results in signal reconstruction with meaningful frequency properties. These models, however, are memory demanding, due to their large nature in terms of trainable parameters, thus requiring well-thought design that balances high accuracy with computational efficiency. A promising path is to pretrain models on

large-scale datasets to extract robust signal representations of ultrasound signals and then fine-tune them per task or apply quantization techniques to deploy them on lightweight devices.

Addressing this dilemma between performance and efficiency is the main topic of this thesis: the development of lightweight, compact, pretrained and quantized model architectures for ultrasound signals, that preserve temporal-spectral information and make the integration on the edge a practical reality.

1.2. Objective

As mentioned, the main objective of this thesis is to investigate pretraining and quantization architectures for ultrasound data compression, with a focus on M-mode data representations relevant to the wearable design. The main research questions that arise are the following:

- *What is the most efficient model architecture for capturing the structure and temporal-spectral properties of ultrasound data?*
- *How can pretraining strategies improve compression with respect to several metrics and seuentially transfer to downstream tasks?*
- *Which quantization strategy (post-training or quantization-aware, integer or floating point) is the go-to option for balancing the tradeoff between efficiency and computational performance?*

To address these research questions, we develop and evaluate a hybrid multi-head model architecture, that leverages both Transformers and Fourier Neural Operators, in order to capture both temporal and frequency-domain information. We also perform a systematic study of input formation and embedding strategies, such as linear vs convolutional embeddings and positional encoding, different fusion mechanisms between the heads of the model architecture, and concatenation among angles for the input data. Lastly, to further enhance training, we integrate a multi-scale STFT-based contrastive loss inspired by recent work in neural audio compression, and a frequency domain reconstruction loss. In terms of quantization strategies, we experiment with both static and dynamic post-training quantization (PTQ) and quantization-aware training (QAT), across 8-bit and 16-vit integers formats. Performance is evaluates with respect to reconstruction quality, latency and most importantly spectral distortion.

In summary, contributions of this thesis are the following:

1. Design of a multi-head Transformer-FNO architecture for ultrasound compression.
2. A systematic embedding study (linear vs convolutional, positional encoding, multi-angle concatenation, fusion methods).

3. Integration of a contrastive MSTFT-based loss and frequency-domain reconstruction loss in the training pipeline.
4. Comprehensive evaluation of PTQ and QAT quantization techniques.

The aforementioned research approaches provide fruitful insights into designing a lightweight yet expressive model for ultrasound compression, that converges the gap between meaningful representation learning and wearable sensing.

Chapter 2

Preliminaries and Related Work

This chapter provides the reader with basic knowledge around preliminaries and theoretical background, so that the methods later discussed are properly comprehended. Since the basic focus of the thesis revolves around deep learning methodologies for signal compression and quantization, there will not be excessive emphasis on ultrasound physics and signal acquisiton. Instead the technical foundation that contextualizes our work will mostly touch subjects around model architectures and deep learning frameworks.

2.1. Ultrasound Fundamentals

2.1.1. Principles of Ultrasound Imaging

A US image typically gives a cross-sectional view of tissue and organ boundaries for a target region. The US image is generated by sending an ultrasound wave (typically 2–15 MHz) through a tissue of interest using a transducer. This causes echoes that arise as ultrasound waves reflect from tissue borders and scatter from slight anomalies within the tissues. Using the pulse-echo principle, which is the same principle used for sonar, one can reconstruct the exact locations of the tissue borders with respect to the transducer and generate the resulting US image. The time delay of returning echoes encodes depth information, while the amplitude of the reflected signal indicates differences in tissue properties [2].

The propagation of a sound wave depends on the medium it is traveling in [3]. The frequency of the wave f is independent of the propagation medium and stays constant as the sound wave crosses from one medium to another and is determined by the excitation frequency of the transducer which usually ranges between 2-15 MHz. However the speed

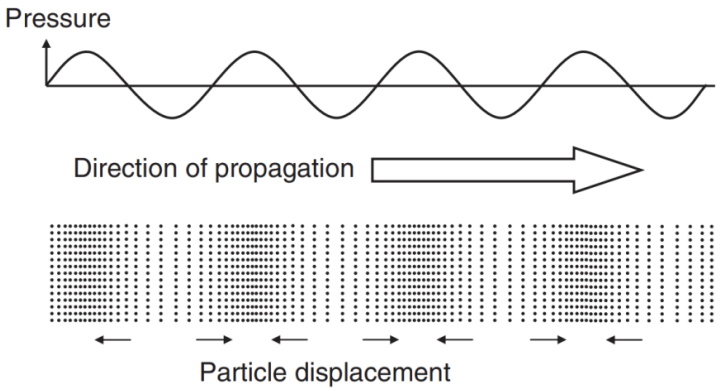


Figure 2.1.: The visualization of a sound wave, showing the particle displacement in the direction of propagation [2].

of sound (SoS) c of a sound wave changes across different media and is fully determined by the current propagation medium.

2.1.2. Transducers

The device that generates the ultrasound wave and then records the echo is called the transducer. Medical transducers usually consist of multiple transducer elements, where each transducer element converts an electrical signal into a sound wave and then also converts the returning ultrasound echo to an electrical signal again. This electrical signal generated from the returning sound waves is the RF data we measure which is subsequently used to generate the ultrasound image [3].

Transducers make use of the piezoelectric effect to convert the electrical signals to sound waves and the reverse. A medical transducer usually consists of between 128 and 256 transducer elements. Each transducer element has a piezo-element to generate a sound wave and receive a signal. For each transducer element, there is a corresponding channel in the captured RF signals. The resonance frequency of the transducer is also referred to as the center frequency and typically corresponds to the frequency at which the ultrasound wave pulse is emitted by the transducer.

2.1.3. Ultrasound Imaging Modes

Ultrasound signals can be acquired through several modes. In this section, we enumerate them and their main properties:

1. **A−mode:** The most simple mode of US acquisition - A stands for Amplitude - where a single focused beam is broadcast and echoes of it are received on a single

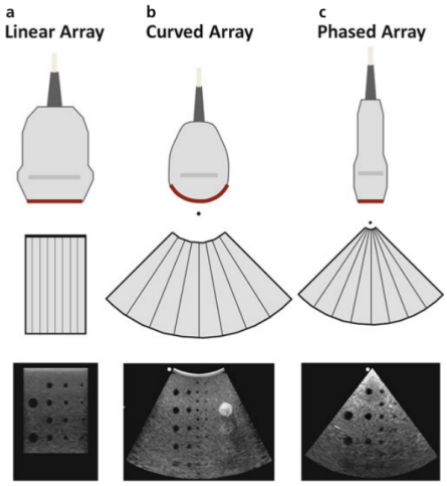


Figure 2.2.: Three different transducer types. [3]

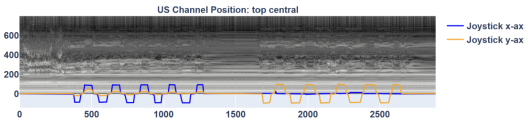


Figure 2.3.: An example of M-mode data displaying joystick movements.

channel. The plot refers to the height of the reflected amplitudes with respect to the depth. This mode is rarely used in diagnostic systems.

2. **B−mode:** The most common of US representation - where B stands for Brightness and raw US data is collected from many channels into the formation of a US image. For traditional B-mode imaging, the US image is formed line by line using focused transmit US beams, to achieve a high image resolution in the lateral direction.

3. **M−mode:** This mode is the main focus of this thesis - where M stands for motion. Just like A−mode, a single time sequence of echoes is acquired at a single scan line, but instead of producing a 2D anatomical image, M mode gives a time-depth plot evolving through time - and thus producing a 2D image [4].

In this work, the main focus lies on M−mode data, since it is more relevant for wearable ultrasound applications. Unlike B-mode images, which are primarily spatial, M-mode captures dynamic temporal patterns such as tissue motion, blood flow, or hand gestures. This makes it well suited for continuous monitoring tasks, while also posing unique challenges for compression: the data forms long sequential recordings that must be represented efficiently for storage and real-time processing on resource-constrained devices. Since both axis of an M−mode refer to time passages, we may usually refer to the beam

index acquired from echoes as the **fast time** -typically sampled at around 2GSPS - and the axis of temporal evolution as **slow time** - usually sampled at 25H Hz.

2.2. Wearable Ultrasound Applications

2.2.1. Vital Signs Monitoring

Recent progress in skin-conformal, flexible ultrasound has enabled continuous, non-invasive monitoring of deep-tissue haemodynamics and cardiovascular markers. A stretchable ultrasonic phased-array patch demonstrated real-time Doppler spectra from cardiac tissue and central blood flow up to 14 cm depth in healthy volunteers [5]. Building on this, a fully integrated wearable ultrasonic-system-on-patch (USoP) combined flexible transducers, front-end electronics, and wireless communication, enabling continuous tracking of blood pressure, heart rate, and cardiac output for up to 12 hours in mobile subjects [6].

Flexible Doppler devices have also enabled continuous measurement of absolute blood-flow velocity in deeply embedded arteries, validated both in phantoms and human studies [7]. Complementary systems such as wireless neckbands for bilateral carotid monitoring demonstrate the feasibility of continuous vascular sensing with embedded pre-processing [8].

Reviews of wearable ultrasound consistently identify two challenges: (i) signal-conditioning and acquisition on flexible platforms, and (ii) energy-efficient, real-time computation [9, 10]. These constraints directly motivate the need for compression and quantization methods that make deployment on small embedded hardware practical.

2.2.2. Gesture Recognition and Human–Computer Interaction

Wearable ultrasound can sense muscle morphology and motion (sonomyography), enabling hand and forearm gesture decoding with high specificity. Recent reviews summarize progress in ultrasound-based human–machine interfaces and prosthetic control [11, 12].

Focusing on the modes most relevant to this work, M-mode ultrasound has been used to classify wrist and finger motions with competitive accuracy compared to B-mode, while producing a compact time–depth sequence well suited for streaming and compression [13]. A recent single-transducer wearable echomyography (EcMG) system leveraged RF echoes and deep learning to continuously track 13 hand joint degrees-of-freedom, achieving mean angular errors under 8° [14]. This demonstrates that narrowband ultrasound signals can provide rich motor decoding capabilities in wearable form factors.

From a systems perspective, edge-deployable ultrasound gesture recognition has already adopted model compression. Quantization to 8-bit fixed-point has been shown to maintain over 90% classification accuracy while meeting latency and energy constraints on small processors [15]. These results emphasize the necessity of lightweight inference for practical wearable interfaces, motivating the exploration of pretraining and quantization for ultrasound compression in this thesis.

2.3. Machine Learning for Ultrasound

In this section, we interest in applications of machine learning models specifically for ultrasound signals. Starting with a brief yet compact analysis of conventional approaches, we then discuss the use of Transformers on Biomedical Signals, being the first key part of proposed deep learning architecture. The second part is that of Fourier Neural Operators and their capabilities on parameterizing function spaces. Lastly, we analyze several hybrid architectures for ultrasound signals that made their appearance in literature.

2.3.1. Conventional Approaches – CNNs

In the early stages of machine learning architectures for ultrasound imaging, Convolutional Neural Networks (CNNs) had set the pace for over a decade. Their success relied on their capabilities to learn hierarchical spatial features from raw pixel intensities, thus making the go−to option for several medical imaging tasks, such as reconstruction, segmentation and denoising.

In segmentation, encoder–decoder architectures such as U-Net [16] and its variants have been widely adopted for localizing anatomical structures in B-mode ultrasound, including vessels, muscles, and tumors [17, 18]. Applications of CNNs extend on reconstruction tasks, such as speckle noise removal and super-resolution, with methods like Deep Ultrasound Super-Resolution (DUSR) [19] and CNN-based despeckling [20] demonstrating significant improvements in image quality. Furthermore, CNNs have been integrated as key blocks of greater pipelines, in order to accelerate beamforming and improve reconstruction quality [21, 22].

However, while CNNs' supremacy at learning local image features is of no doubt, their reliance on fixed-size convolutional kernels limits their ability to capture long-range temporal and frequency dependencies, a key requirement in modeling ultrasound data. This problem is even more evident for sequential modalities such as M-mode, where global context across time is essential. As a result, such architectural limitations motivated the scientific community to explore other architectures as alternatives, such as transformers and spectral operators, which provide more flexible mechanisms to model sequential data and temporal–spectral dependencies.

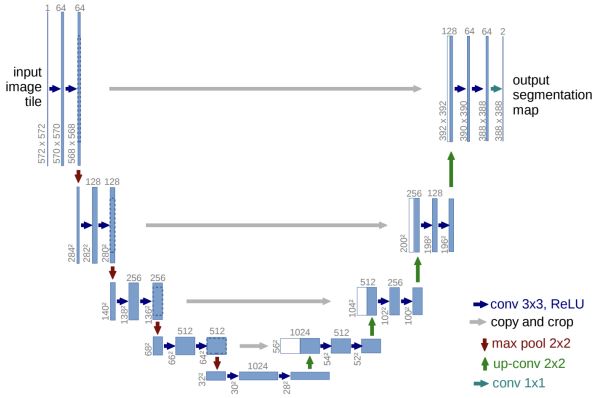


Figure 2.4.: Typical CNN encoder–decoder architecture for ultrasound image segmentation (example for 32x32 pixels in the lowest resolution). [16].

2.3.2. Transformers for Biomedical Signals

Convolutional networks, while powerful for local feature extraction, are limited in their ability to capture long-range dependencies. Transformers [23] address this limitation by replacing convolutions with a self-attention mechanism that directly models relationships between all positions in a sequence. Originally developed for natural language processing, Transformers have since achieved state-of-the-art performance in computer vision [24, 25] and medical image analysis [26].

Self-Attention Mechanism. Given a sequence of input embeddings $X \in R^{T \times d}$ with T time steps and embedding dimension d, the self-attention mechanism projects X into queries Q, keys K, and values V:

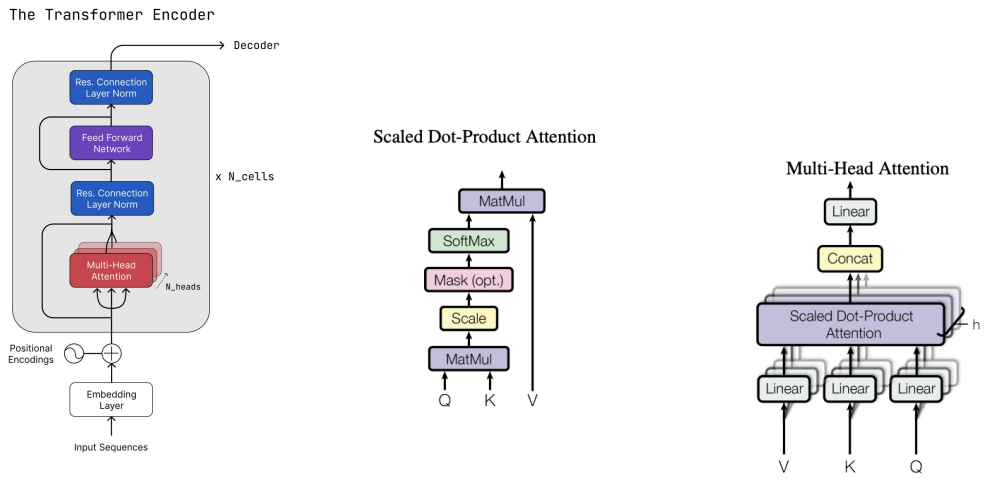
$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{2.1}$$

where $W_Q, W_K, W_V \in R^{d \times d_k}$. The scaled dot-product attention is then computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V. \tag{2.2}$$

Extending this to h parallel heads, the outputs are concatenated and passed through a linear layer to form the final multi-head attention output.

Positional Encoding. Since attention layers are permutation-invariant, positional encodings are added to input embeddings to provide temporal order. The sinusoidal encoding introduced in [23] assigns each position t a vector:



(a) Transformer encoder block (b) Scaled dot-product attention (c) Multi-head attention

Figure 2.5.: Illustrations of (a) the Transformer encoder block, (b) scaled dot-product attention, and (c) multi-head attention. Adapted from [23].

$$\mathrm{PE}_{(t,2i)} = \sin\left(\frac{t}{10000^{2i/d}}\right), \quad \mathrm{PE}_{(t,2i+1)} = \cos\left(\frac{t}{10000^{2i/d}}\right), \tag{2.3}$$

where i indexes the embedding dimension. This formulation was used in our implementation (see Listing **??**).

Variants in Biomedical Imaging. The Vision Transformer (ViT) [24] demonstrated that splitting images into non-overlapping patches and processing them as sequences achieves competitive performance to CNNs. Swin Transformer [25] extended this idea with hierarchical representations and shifted windows, improving scalability to high-resolution images. In medical imaging, TransUNet [26] integrates a Transformer encoder with a CNN-based decoder, combining global context modeling with local feature sensitivity. These designs have been applied successfully to tasks such as tumor segmentation and organ delineation.

Transformers for Ultrasound. Ultrasound data, especially M-mode sequences, is inherently sequential and benefits from long-range temporal modeling. Transformers are particularly well-suited for such data, capturing dependencies across extended time frames and frequency patterns. Recent work has shown Transformers outperforming CNNs for ultrasound classification and reconstruction [27, 28]. Moreover, their modular

design allows integration with spectral learners such as Fourier Neural Operators (see Section 2.3.3).

2.3.3. Fourier Neural Operators (FNOs)

As explained, ultrasound signals, particularly M-mode sequences, contain rich frequency structure, that Transformers alone are not designed to capture. By explicitly modeling signals in the Fourier domain, FNOs provide an efficient way to capture global spectral patterns that complement the temporal modeling ability of Transformers. In this work, we integrate an FNO-based head alongside a Transformer branch to jointly learn temporal and spectral representations of ultrasound data.

Operator Learning. Most deep learning methods learn mappings from finite-dimensional inputs to outputs, e.g., image classification $f : R^n \to R^m$. In contrast, many problems in science and engineering require learning *operators*, i.e., mappings between infinite-dimensional function spaces. Given functions $a(x)$ as inputs and $u(x)$ as outputs, an operator $\mathcal{G}$ is defined as:

$$u(x) = \mathcal{G}(a)(x). \tag{2.4}$$

For example, solving partial differential equations (PDEs) amounts to learning $\mathcal{G}$ that maps forcing functions or boundary conditions to solutions.

Fourier Neural Operators. The Fourier Neural Operator (FNO) [29] learns operators by performing convolutions in the Fourier domain. Instead of applying a convolution kernel in the spatial domain, the FNO projects the input into frequency space using a Fourier transform, applies a learned multiplier to the low-frequency modes, and transforms back. Formally, given an input feature map $v \in R^{n\times d}$, the spectral convolution is:

$$\mathcal{F}(v)(k) = \sum_{x=0}^{n-1} v(x) e^{-2\pi i k x/n}, \tag{2.5}$$

$$\hat{v}'(k) = R(k) \cdot \hat{v}(k), \quad k \leq K, \tag{2.6}$$

$$v'(x) = \mathcal{F}^{-1}(\hat{v}')(x), \tag{2.7}$$

where $\hat{v}(k)$ is the Fourier transform of v, $R(k)$ is a learned complex-valued weight matrix applied only to the lowest K frequency modes, and $\mathcal{F}^{-1}$ is the inverse Fourier transform. The operation retains global receptive fields while remaining efficient, since only a subset of frequency modes is used.

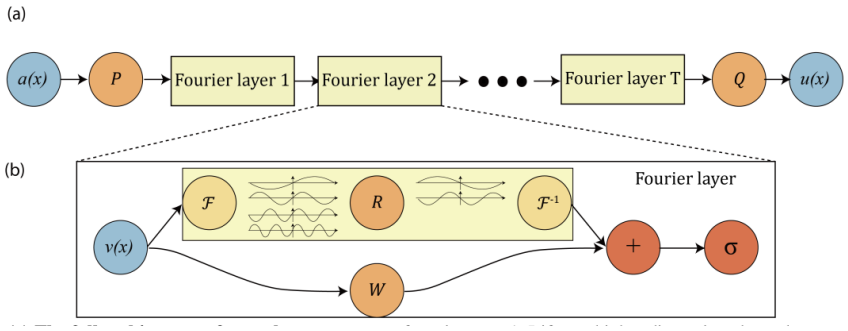


Figure 2.6.: **(a) The full architecture of neural operator:** start from input a. 1. Lift to a higher dimension channel space by a neural network P. 2. Apply four layers of integral operators and activation functions. 3. Project back to the target dimension by a neural network Q. Output u. **(b) Fourier layers:** Start from input v. On top: apply the Fourier transform $\mathcal{F}$; a linear transform R on the lower Fourier modes and filter out the higher modes; then apply the inverse Fourier transform $\mathcal{F}^{-1}$. On the bottom: apply a local linear transform W [29].

Architecture. An FNO block consists of:

- A linear lifting layer to map inputs to a higher-dimensional space.
- A stack of Fourier layers, each performing spectral convolution followed by a nonlinearity.
- A projection layer to map the output back to the target dimension.

This architecture scales to high-dimensional inputs while capturing long-range dependencies with fewer parameters compared to CNNs.

Applications. FNOs have demonstrated success in physics-informed modeling tasks such as fluid dynamics, turbulence modeling, weather forecasting, and material simulation [29, 30]. Their efficiency in learning structured signals has motivated extensions to other modalities, including medical imaging [31, 32].

2.4. Pretraining in Medical Imaging

A key motivation for pretraining is the transferability of learned representations. In medical imaging, it is common to pretrain encoders on large-scale unlabeled data and fine-tune them for specific downstream tasks such as classification or segmentation. For ultrasound, this translates to pretraining on M-mode signals and evaluating transfer to applications such as gesture recognition.

2.4.1. Self-Supervised and Contrastive Pretraining

Supervised training in medical imaging is often limited by scarce annotations, motivating the development of self-supervised learning (SSL) methods that exploit large amounts of unlabeled data. Contrastive learning has been particularly successful, where the goal is to maximize agreement between differently augmented views of the same sample while pushing apart representations of other samples.

Popular frameworks such as SimCLR [33], BYOL [34], and DINO [35] have been adapted to medical modalities including MRI and CT. For example, Chaitanya et al. [36] demonstrated that contrastive pretraining improves 3D segmentation performance on MRI brain scans, while Zhou et al. [37] showed similar gains for lung lesion detection in CT images. These approaches highlight the value of representation learning from large unlabeled datasets, followed by fine-tuning on downstream tasks with limited supervision.

2.4.2. Frequency-Aware Pretraining

Standard self-supervised learning approaches in imaging often rely on spatial augmentations (e.g., cropping, color jittering) and contrastive losses. However, ultrasound signals—especially M-mode and RF data— contain rich temporal and spectral structure critical for compressing and interpreting motion and echoes. To preserve this information, we adopt frequency-aware pretraining strategies inspired by recent advances in neural audio compression.

Meta's EnCodec model [38, 39] is a state-of-the-art neural audio codec trained end-to-end. Its key innovation is the integration of a multi-scale short-time Fourier transform (MSTFT) discriminator as a perceptual loss, which greatly improves reconstructed signal fidelity.

The MSTFT loss is formally expressed as:

$$\mathcal{L}_{\text{MSTFT}} = \frac{1}{M} \sum_{m=1}^{M} \left\| |\text{STFT}_{N_m}(x)| - |\text{STFT}_{N_m}(\hat{x})| \right\|_1 , \tag{2.8}$$

where

- $\text{STFT}_{N_m}(\cdot)$ is the short-time Fourier transform using window size N_m,
- x is the original signal, and $\hat{x}$ is the reconstructed output,
- the absolute value denotes the magnitude (ignoring phase),
- M is the number of scales (STFT windows) used to capture both fine and coarse spectral content.

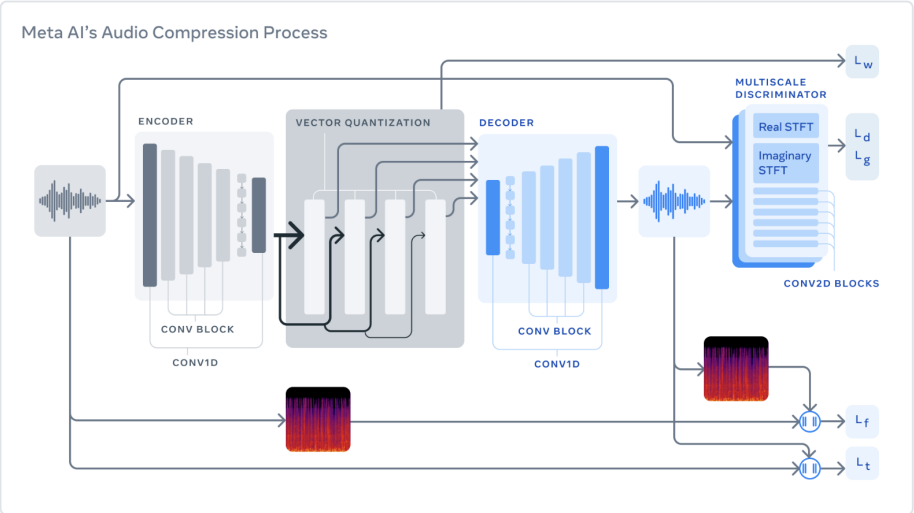


Figure 2.7.: **EnCodec:** an encoder–decoder codec architecture which is trained with reconstruction losses (ℓ_f and ℓ_t) as well as adversarial losses (ℓ_g for the generator and ℓ_d for the discriminator). The residual vector quantization commitment loss (ℓ_w) applies only to the encoder [39]

Critically, the MSTFT discriminator acts like a perceptual critic that penalizes discrepancies in the spectral domain, driving the model to reconstruct both fine-grained detail and broad frequency envelopes. Meta further introduces a loss balancing mechanism: each loss term's weight is normalized to contribute a fixed fraction of the total gradient, making training stable across objective and perceptual components [38].

Visually, MSTFT loss ensures that errors in high-frequency flickers and low-frequency envelopes are both captured—vital for ultrasound signals where both fast motion and slow trends carry semantic meaning.

In our implementation, the MSTFT loss complements our contrastive setup (using a Meta-style MSTFT discriminator) and is used jointly with reconstruction losses. By incorporating an MSTFT-based loss, our pretraining encourages the model not only to reconstruct the signal in time but also to maintain its spectral characteristics, ensuring that learned representations generalize robustly across different ultrasound patterns. This hybrid objective improves both compression quality and the transferability of representations to downstream tasks, such as gesture recognition, where M-mode frequency content is key.

2.5. Quantization & Compression Techniques

2.5.1. Model Compression Approaches

Deep neural networks often contain millions of parameters, posing challenges for deployment on resource-constrained hardware. A range of model compression techniques have been proposed:

- **Pruning:** Removal of weights or entire filters with small magnitudes, leading to sparse models that require specialized hardware/software to realize speedups [40].
- **Knowledge Distillation:** Training a small "student" model to mimic the predictions or intermediate representations of a large "teacher" model [41].
- **Low-Rank Factorization:** Decomposing large weight matrices into products of lower-rank matrices, reducing parameter count and FLOPs [42].

While these techniques provide parameter reduction, they often require careful tuning and do not directly guarantee efficiency on generic mobile CPUs/MCUs. In contrast, **quantization**—representing weights and activations with reduced precision—offers immediate gains in storage and compute, making it especially relevant for wearable ultrasound applications.

2.5.2. Quantization Methods

Quantization can be formalized as mapping a real-valued tensor x to a discrete set of values:

$$Q(x) = \text{clip}\left(\text{round}\left(\frac{x}{s}\right), q_{\min}, q_{\max}\right), \tag{2.9}$$

where s is a scaling factor and $[q_{\min}, q_{\max}]$ defines the quantization range. Two main strategies exist:

- **Post-Training Quantization (PTQ):** Weights and/or activations are quantized after model training.
  - *Static PTQ:* Quantization parameters are determined using calibration data, providing more stable results.
  - *Dynamic PTQ:* Scaling parameters are estimated on-the-fly during inference, useful for NLP and streaming data.

  PTQ requires no retraining but may suffer accuracy degradation in sensitive models [43].

- **Quantization-Aware Training (QAT):** Quantization effects are simulated during training by inserting "fake quantization" nodes in the forward pass. This allows the network to adapt to quantization noise, achieving higher accuracy at low bit-widths [43].

2.5.3. Low-Bit Formats

Quantization typically targets 8-bit integer arithmetic, but recent developments extend this to lower-precision formats:

- **Integer Quantization (INT8/INT4):** Widely supported in hardware (e.g., ARM, NVIDIA TensorRT, Qualcomm DSPs). INT8 is the de facto standard for efficient inference, while INT4 enables further reductions at the cost of larger accuracy drops.
- **Low-Precision Floating Point (FP8/FP4):** Emerging formats introduced by NVIDIA and Intel, retaining floating-point dynamic range with fewer bits [44]. FP8 strikes a balance between range and precision, while FP4 targets extreme efficiency in edge devices.

Trade-offs. The impact of quantization can be summarized along three axes:

- **Memory footprint:** Reduces storage from 32-bit floats to 8/4-bit representations, yielding up to 8× compression.
- **Latency:** Accelerates inference on hardware supporting low-bit arithmetic.
- **Accuracy:** Lower precision may introduce quantization error, but QAT mitigates this by adapting weights during training.

For wearable ultrasound, where energy and memory are primary bottlenecks, quantization is the most practical technique, enabling models to operate on small MCUs or mobile GPUs without prohibitive loss in performance.

Chapter 3

Methodology

This chapter is the heart of our thesis, since it explains the core methodology used in our implementation. In the beginning, we analyze the dataset used as well as the preprocessing applied on the input representations of M−mode Ultrasound signals. Sequentially, we focus on the architectural design of our main hybrid model, that combines the strengths of both Transformers and FNOs. Later on, we set the training pipeline framework with emphasis on the loss terms and evaluation metrics applied. We then analyze the basic quantization strategies that we experimented with - both PTQ and QAT approaches - and finish with a concrete summary of implementation details, around the training pipeline.

3.1. Dataset & Input Representation

3.1.1. Internal M-mode Dataset

We use an internal ETH M-mode ultrasound dataset consisting of 300 recorded runs stored as pickle files (e.g., `us_polar_YYYY-MM-DD_HH-MM-SS.pkl`) with a companion `data_desc.csv` manifest that includes `subject`, `position` (Lateral, Central, Medial), `exercising`, and `file_name`. Each run comprises *raw* (non-enveloped) M-mode RF intensity sequences recorded at a **slow-time sampling rate** of $f_s = 25\,\text{Hz}$ for **60 s**, yielding $T = 1500$ time steps per angle (slow time), and **100 beams** per frame (fast-time/beam index). Thus, for each angle the raw array has dimensionality $X \in R^{T \times B} = R^{1500 \times 100}$.

To respect data governance constraints, the dataset is not publicly released. All experiments are conducted under the ETH internal data policy.

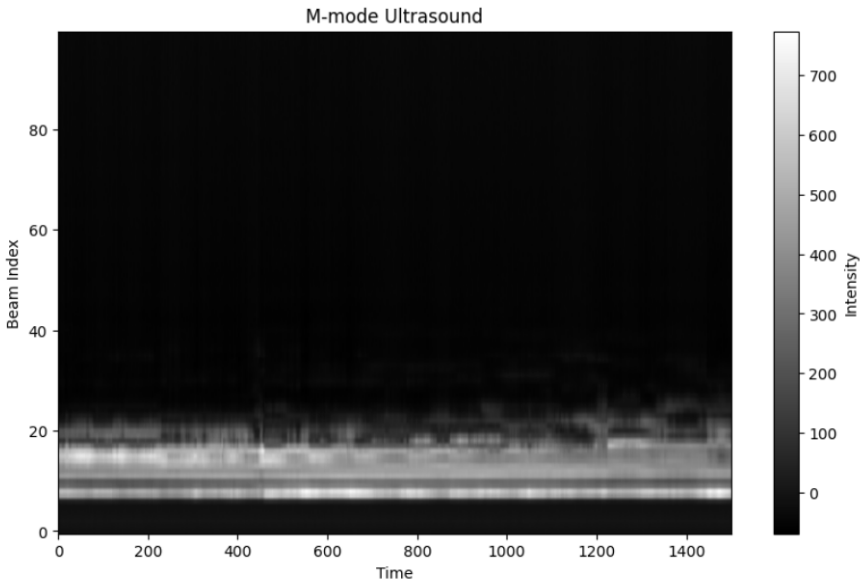


Figure 3.1.: Example raw M-mode sequence (one angle) with shape 1500×100 (60 s at 25 Hz, 100 beams). Intensities are in arbitrary device units.

3.1.2. Train/Val/Test Segmentation and Windowing

Each run is partitioned temporally into non-overlapping segments:

$$\text{train} : [0, 40)\,\text{s}, \quad \text{val} : [40, 50)\,\text{s}, \quad \text{test} : [50, 60)\,\text{s}.$$

Training uses *random windows* sampled within the train segment; validation and test *enumerate* windows deterministically.

A window is defined by duration w (in seconds) and stride Δ (seconds). With $f_s = 25\,\text{Hz}$, we use $w = 3\,\text{s} \Rightarrow L = wf_s = 75$ samples, and $\Delta = 1\,\text{s} \Rightarrow S = \Delta f_s = 25$ samples. Given a segment of length T_{seg} samples, the number of windows is

$$N_{\text{win}} \;=\; \left\lfloor \frac{T_{\text{seg}} - L}{S} \right\rfloor + 1. \tag{3.1}$$

For validation/test (T_{seg}=10 s $\Rightarrow$ 250 samples), $N_{\text{win}} = \lfloor (250 - 75)/25 \rfloor + 1 = 8$ windows per angle, per run. Similarly, for the train set partitions of 1000 samples each, we get a total of 37 windows per angle, per run.

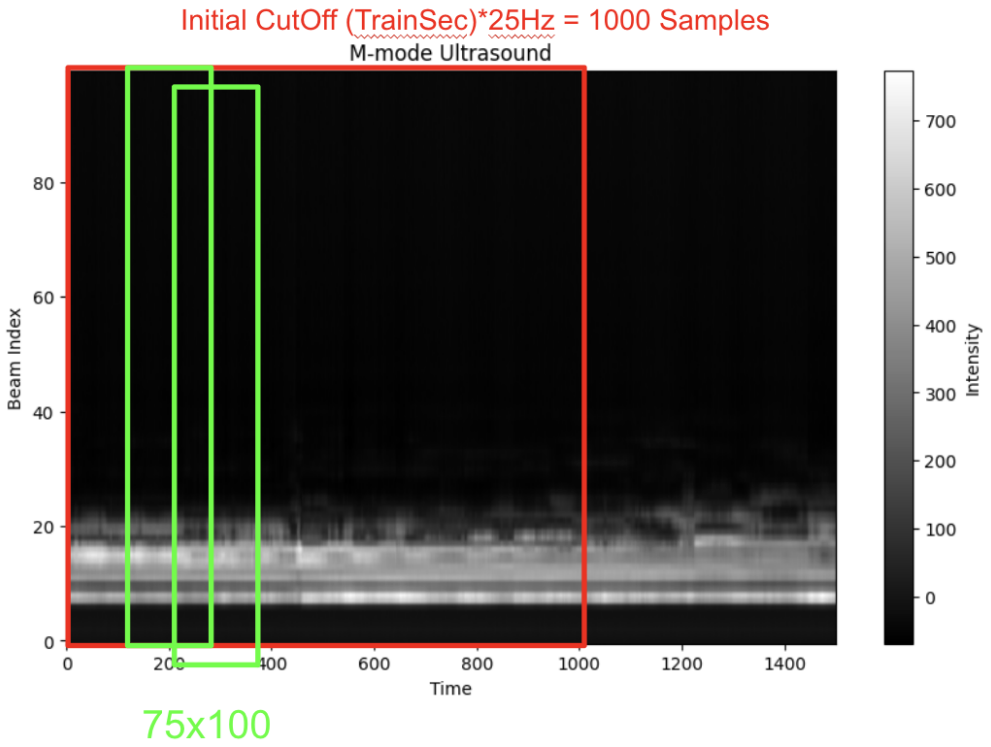


Figure 3.2.: Temporal windowing: 3 s windows (L=75 samples) with 1 s stride (S=25) on the train segment. Validation/test enumerate windows in the respective 10 s segments.

3.1.3. Multi-Angle Sampling and Concatenation

Each training sample draws *one* window from *each* angle (Lateral, Central, Medial) at the same subject and position. For angle $a \in \{\text{lat}, \text{cen}, \text{med}\}$, a window yields $X^{(a)} \in R^{L\times B} = R^{75\times 100}$. We concatenate the three angle windows along the *slow-time* axis (time concatenation), producing

$$X_{\text{concat}} = \text{concat}_t\big(X^{(\text{lat})}, X^{(\text{cen})}, X^{(\text{med})}\big) \in R^{(3L)\times B} = R^{225\times 100}. \tag{3.2}$$

Rationale. Concatenation integrates complementary views of the same anatomy/motion, providing richer supervision and improving generalization: (i) more information per sample reduces estimator variance; (ii) redundancy across views enables denoising/interpolation when one view is locally degraded; (iii) global physiological dynamics (e.g., heartbeats, vascular wall motion) remain coherent across angles, so the temporal frequency structure is *enriched*, not broken. In practice, we observed more stable optimization and better downstream transfer than sampling a single random angle per window.

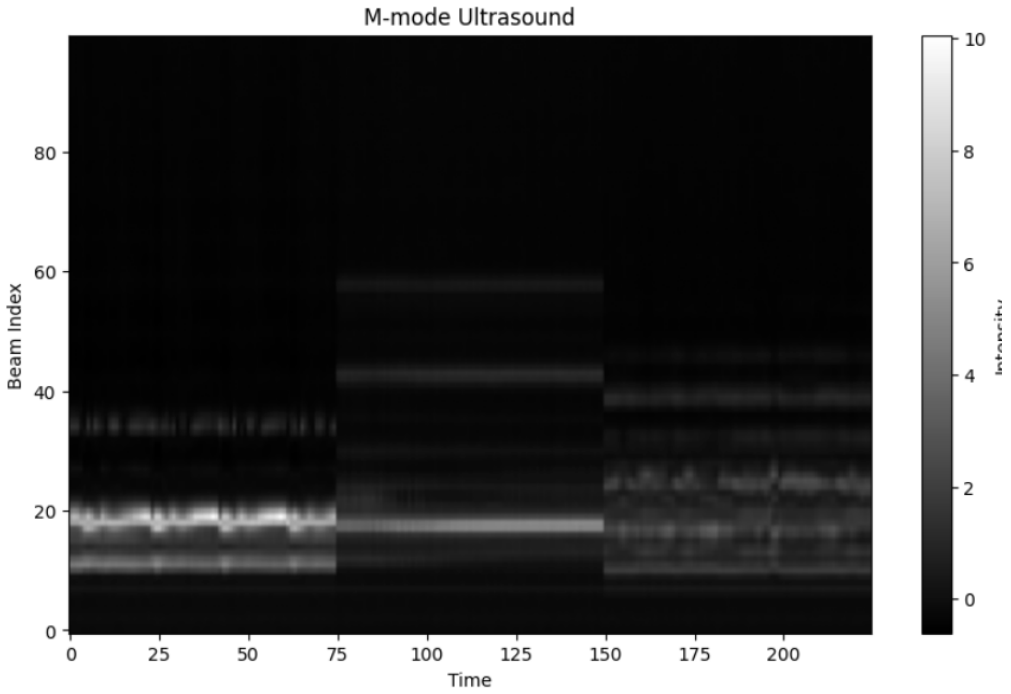


Figure 3.3.: Angle concatenation: three 75×100 windows (lateral, central, medial) are concatenated along slow time to form a 225×100 input. Temporal inconsistencies appear yet generalization improves.

3.1.4. Normalization (Standardization)

We apply per-angle standardization computed on the *training portion only*. Let μ_a, σ_a denote the mean and standard deviation over all train windows for angle a. Each window $X^{(a)}$ is transformed as

$$\widetilde{X}^{(a)} = \frac{X^{(a)} - \mu_a}{\sigma_a + \varepsilon}, \qquad \varepsilon \approx 10^{-8}. \tag{3.3}$$

Standardization is preferred over min–max scaling because ultrasound intensities do not have a fixed, bounded physical range and may contain outliers; min–max is sensitive to extreme values and to dataset shift (e.g., subjects, probe loading), whereas standardization centers and scales the distribution, leading to more stable optimization and better cross-subject generalization. No denoising, band-pass filtering or envelope detection is applied; all models operate directly on the *raw* M-mode signals.

3.1.5. Embeddings and Positional Encoding

We evaluated three embedding strategies applied along slow time:

1. **Linear** projection (baseline): a single linear layer maps $225 \mapsto d_{\text{emb}}$ (default d_{emb}=256), preserving temporal resolution and avoiding early nonlinearity.
2. **1D/2D Convolutional** embeddings: dilated temporal convolutions with $t\_\text{conv}\_\text{channels} = [64, 64]$, kernel size 3, optional max pooling.
3. **TCN** (dilated) embeddings: deeper temporal CNNs with pooling.

Empirically, the *linear* embedding outperformed convolutional/TCN alternatives, particularly when combined with Fourier Neural Operators (FNOs). Pooling and early nonlinear mixing reduce temporal resolution and disrupt the smooth, grid-structured correlations that FNO spectral layers exploit.

For positional information we use sinusoidal encodings $\text{PE}(t, 2i) = \sin(t/10000^{2i/d}), \text{PE}(t, 2i{+}1) = \cos(t/10000^{2i/d})$ added to the embedded sequence (§**??**)—a choice that matched or exceeded learned PEs in our setting while being parameter-free.

3.1.6. FNO Placement for Spectral Learning

We investigated where to place the FNO relative to the embedding: (i) FNO → Embedding (ours), versus (ii) Embedding → FNO. Best results were obtained by applying FNO layers *directly on the raw concatenated input* $X_{\text{concat}} \in R^{225\times100}$, then projecting with a linear embedding. This is a cleaner theoretical approach, since it preserves the regular grid and full temporal resolution for spectral convolution, after which the learned spectral features are lifted for the Transformer head. In our hybrid parallel design, the shared linear embedding initially used for both heads was replaced by FNO→Embedding for the FNO branch, while the Transformer branch continues from the linear embedding. Fusion is described in §3.2.

3.1.7. Shapes & Data Flow Summary

3.1.8. Implementation Notes

Data loading follows the pipelines in our codebase: (a) *TrainDataset* draws random multi-angle windows within $[0, 40)$ s; (b) *ValTestDataset* enumerates windows in $[40, 50)$ s and $[50, 60)$ s; (c) standardization parameters $\{\mu_a, \sigma_a\}$ are computed on the *train* portion only and then reused for val/test; (d) batching uses B=16 by default. All angles are present for every run (no imputation). No pre-filtering is applied.

Table 3.1.: Data shapes through the input pipeline (per sample).

Stage	Shape	Notes
Raw angle window $X^{(a)}$	75×100	w=3 s, f_s=25 Hz
Angle concatenation X_{concat}	225×100	Eq. (3.2)
Standardization	225×100	per-angle stats, Eq. (3.3)
FNO (spectral layers)	$225 \times 100 \to 225 \times 100$	low-mode spectral conv.
Linear embedding	$225 \times 100 \to d_{\text{emb}} \times 100$	d_{emb}=256
Positional encoding	$\to d_{\text{emb}} \times 100$	added to embeddings

3.2. Model Architectures

3.2.1. Baseline Transformer

Our baseline model is a Transformer encoder tailored for sequential M-mode ultrasound data. The input $X_{\text{concat}} \in R^{225\times100}$ (see §3.1) is first passed through a linear embedding layer mapping the beam dimension (100) to an embedding dimension $d_{\text{emb}} = 256$:

$$X_{\text{emb}} = X_{\text{concat}} W_e + b_e, \quad W_e \in R^{100\times256}.$$

A fixed sinusoidal positional encoding $\text{PE}(t)$ is added to preserve temporal ordering:

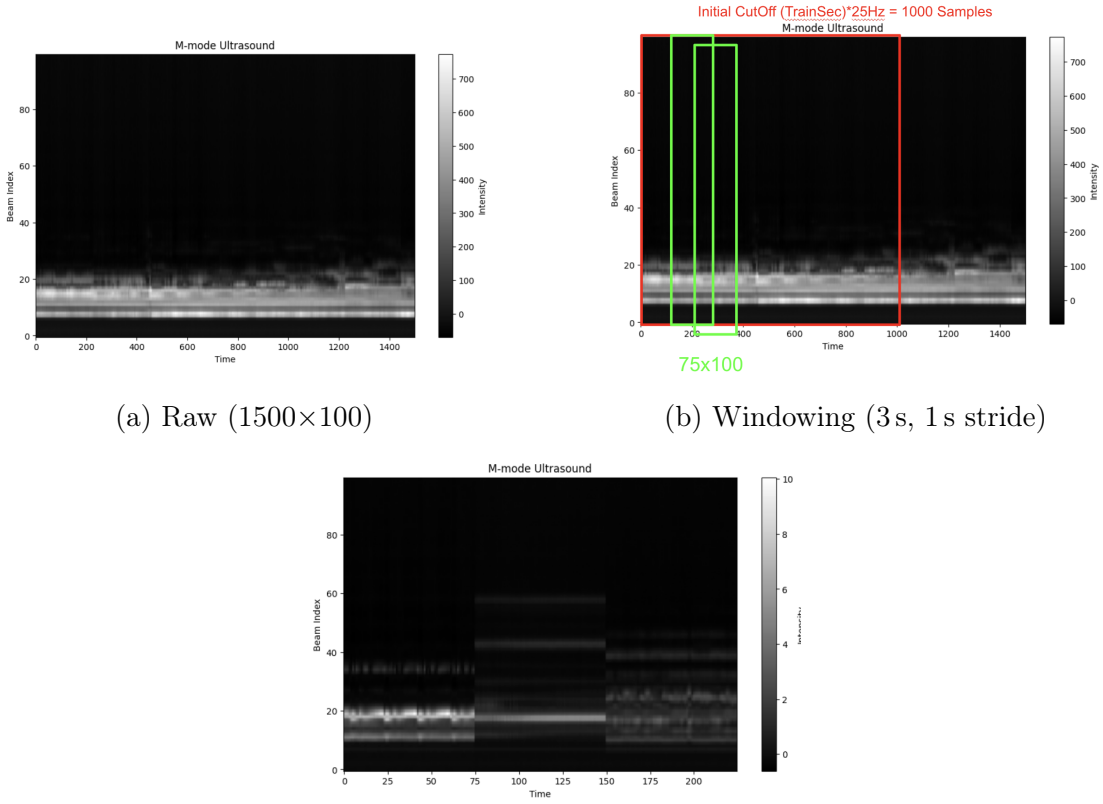
$$\widetilde{X} = X_{\text{emb}} + \text{PE}.$$

The sequence $\widetilde{X} \in R^{225\times256}$ is then passed through $N = 4$ Transformer encoder blocks, each consisting of (i) multi-head self-attention with $n_{heads} = 4$ and hidden dimension 128, (ii) a position-wise feed-forward network, and (iii) residual connections with LayerNorm. Each block computes:

$$H' = \text{MHA}(\text{LN}(H)) + H, \quad H^+ = \text{FFN}(\text{LN}(H')) + H'.$$

Finally, a lightweight dense interpolation network (`TailPretrain`), comprising of two MLPs with ReLU activation functions and Dropouts, upsamples the time dimension back to 225 and projects the features back to 100 to reconstruct the input dimensions. Thus, the output of the baseline is

$$\hat{X}_{\text{base}} \in R^{225\times100}.$$



(a) Raw (1500×100)

(b) Windowing (3 s, 1 s stride)

(c) Angle concat (225×100)

Figure 3.4.: From raw M-mode to model input: raw sequence, temporal windowing, and multi-angle concatenation.

3.2.2. Hybrid Transformer–FNO Parallel Head

To better exploit both temporal and spectral structures, we designed a hybrid dual-head model where a Fourier Neural Operator (FNO) branch runs in parallel with the Transformer branch.

FNO branch. The raw input $X_{\text{concat}} \in R^{225\times100}$ is reshaped to $X \in R^{1\times225\times100}$ and passed through an FNO layer:

$$Z_{\text{fno}} = \mathcal{F}^{-1}\big(R \cdot \mathcal{F}(X)\big) + WX,$$

where $\mathcal{F}$ denotes the 2D Fourier transform, R acts on low Fourier modes (with $n_{\text{modes}} = (16, 16)$), and W is a local linear operator. The resulting output is projected to the embedding dimension 256 via a separate linear layer, which as explained before it is

crucial that it is applied *after* the FNO block.

Transformer branch. In parallel, the input is linearly embedded, augmented with sinusoidal PE, and passed through the same $N = 4$ Transformer encoder blocks as in the baseline.

Fusion. The two streams produce aligned representations $Z_{\text{trans}}, Z_{\text{fno}} \in R^{225\times256}$. We investigated three fusion strategies:

$$\text{Addition:} \quad Z = Z_{\text{trans}} + Z_{\text{fno}}, \tag{3.4}$$

$$\text{Concatenation:} \quad Z = \text{CNN}\big([Z_{\text{trans}}; Z_{\text{fno}}]\big), \tag{3.5}$$

$$\text{Multiplication:} \quad Z = Z_{\text{trans}} \odot Z_{\text{fno}}, \tag{3.6}$$

where $\odot$ denotes elementwise product. While addition and concatenation (with a CNN projection) were feasible, the multiplicative fusion yielded the most consistent improvements in reconstruction metrics (SSIM, PSNR, LPIPS) and spectral fidelity. Intuitively, multiplication enforces both temporal and spectral branches to agree, enhancing robustness and denoising.

Tail. The fused representation $Z \in R^{225\times256}$ is fed to the same dense interpolation tail as in the baseline, producing $\hat{X}_{\text{hybrid}} \in R^{225\times100}$.

3.2.3. Discussion of Design Choices

The baseline Transformer captures long-range temporal dependencies effectively, but is limited in frequency representation. The FNO branch addresses this by explicitly modeling low-frequency modes in the spectral domain. Concatenation of the three ultrasound angles at input (§3.1.3) further enriches supervision by aligning complementary views of the same motion. Fusion by elementwise product was empirically superior: it reduced redundancy while reinforcing cross-branch agreement. Addition risked over-smoothing, and concatenation required extra parameters and induced instabilities when paired with FNO spectral layers.

3.3. Training Setup

3.3.1. Optimization and Schedule

We train with Adam (learning rate 10^{-3}, weight decay 0), batch size 16, for 25 epochs. Unless stated otherwise, we use a step scheduler with decay factor $\gamma = 0.5$ applied every

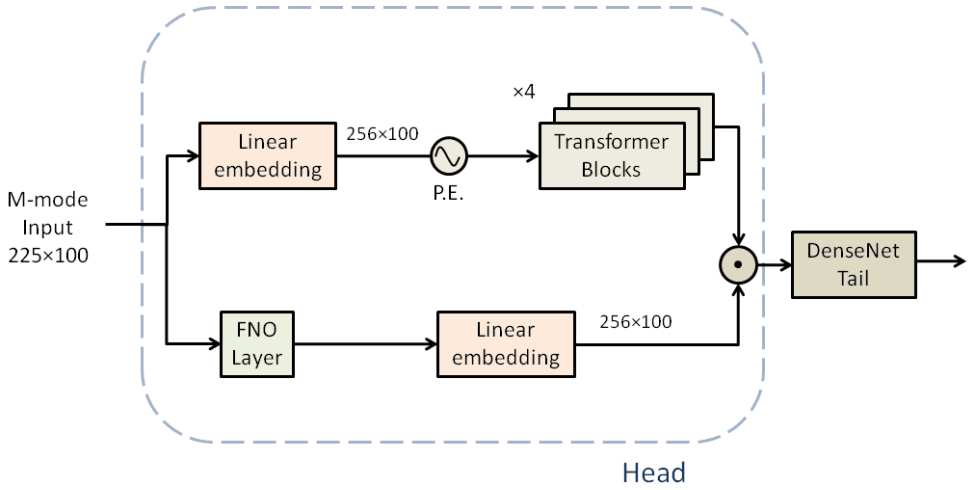


Figure 3.5.: Schematic of the hybrid Transformer–FNO architecture: raw input windows are processed by both an FNO branch (spectral modeling) and a Transformer branch (temporal modeling), fused by elementwise multiplication, and reconstructed by a dense interpolation tail.

5 epochs. Inputs are standardized per angle (cf. §3.1.4). Worth mentioning that, when quantization-aware training (QAT) is enabled (§3.4), it runs during the final 10 epochs.

3.3.2. Losses

Let $X \in R^{225\times100}$ denote the input M-mode window and $\hat{X}$ the model output (same shape). The generator (Transformer–FNO head with dense tail) is trained with a composite objective:

$$\mathcal{L}_{\mathrm{G}} = \lambda_t \, \mathcal{L}_{\mathrm{MSE}}(\hat{X}, X) + \lambda_f \, \mathcal{L}_{\mathrm{FFT}}(\hat{X}, X) + \lambda_{\mathrm{FM}} \, \mathcal{L}_{\mathrm{FM}}(\hat{X}, X), \tag{3.7}$$

where λ_t=1 by default; λ_f is implemented internally via the `FFTLoss` parameter α; and λ_{FM} is set via a scalar `alpha_disc` (we use 1 in our runs).

Time-domain MSE. We minimize the per-pixel mean squared error:

$$\mathcal{L}_{\mathrm{MSE}}(\hat{X}, X) \;=\; \|\hat{X} - X\|_2^2, \qquad \mathrm{RMSE} \;=\; \sqrt{\tfrac{1}{NB}\|\hat{X} - X\|_2^2}, \tag{3.8}$$

with N=225 (time) and B=100 (beams). In contrast to training, when validating the performance our hybrid model we only report the RMSE Loss, in order to compare ourselves with the Baseline Transformer architecture.

Frequency-domain consistency (FFTLoss). To explicitly preserve spectral structure, we add an FFT magnitude term. For an input tensor Y (shape $[B, N, B]$ or $[B, H, W]$), define the normalized 2D FFT magnitude

$$\widetilde{\mathcal{F}}(Y) \;=\; \frac{\big|\mathcal{F}_{\mathrm{2D}}(Y)\big| - \mu_Y}{\sigma_Y + 10^{-8}}, \quad \mu_Y = \mathrm{mean}_{\mathrm{spatial}}(\cdot),\; \sigma_Y = \mathrm{std}_{\mathrm{spatial}}(\cdot). \tag{3.9}$$

The implemented loss (`FFTLoss`) is

$$\mathcal{L}_{\mathrm{FFT}}(\hat{X}, X) \;=\; \mathcal{L}_{\mathrm{MSE}}(\widetilde{\mathcal{F}}(\hat{X}), \widetilde{\mathcal{F}}(X)), \qquad \mathcal{L}_{\texttt{FFTLoss}} \;=\; \mathcal{L}_{\mathrm{MSE}}(\hat{X}, X) + \alpha\, \mathcal{L}_{\mathrm{FFT}}(\hat{X}, X), \tag{3.10}$$

with α=0.5 by default. This corresponds to λ_t=1, λ_f=α in Eq. (3.7).

MS-STFT feature matching (perceptual/contrastive). We employ a multi-scale STFT discriminator D similar to EnCodec to extract multi-resolution spectral features from 1D reshaped sequences. In code, X and $\hat{X}$ are reshaped to $(B, 1, N \cdot B)$ before D. Let $\{\Phi_k(\cdot)\}_{k=1}^{K}$ denote the K sub-discriminators (scales), each providing a list of intermediate features $\Phi_k(\cdot) = \{\phi_{k,\ell}(\cdot)\}_{\ell=1}^{L_k}$. The *feature matching* term we minimize for the generator is

$$\mathcal{L}_{\mathrm{FM}}(\hat{X}, X) = \frac{1}{\sum_k L_k} \sum_{k=1}^{K} \sum_{\ell=1}^{L_k} \frac{\Big\|\phi_{k,\ell}(\hat{X}) - \phi_{k,\ell}(X)\Big\|_1}{\mathrm{mean}(|\phi_{k,\ell}(X)|) + 10^{-9}}, \tag{3.11}$$

i.e., an ℓ_1 distance across all discriminator features, normalized layer-wise by the mean absolute activation to balance scales.

Discriminator hinge loss and schedule. The discriminator is updated *every 5 epochs* to avoid overwhelming the generator. For the set of logits $\{s_k(\cdot)\}_{k=1}^{K}$ produced at the final layer of each sub-discriminator, we use the standard hinge loss

$$\mathcal{L}_{\mathrm{D}} \;=\; \frac{1}{K} \sum_{k=1}^{K} \Big[E_X\big[\, \max(0,\, 1 - s_k(X))\,\big] \;+\; E_{\hat{X}}\big[\, \max(0,\, 1 + s_k(\hat{X}))\,\big] \Big], \tag{3.12}$$

optimized with Adam (10^{-4}, betas (0.5, 0.9)). The generator does *not* see an adversarial term; it receives only the feature-matching gradient (Eq. 3.11) with weight λ_{FM}.

3.3.3. Evaluation Metrics

We evaluate on the reconstructed M-mode images (225×100) using time, perceptual, and spectral metrics (implemented in `test.py`).

RMSE.

$$\mathrm{RMSE}(\hat{X}, X) = \sqrt{\tfrac{1}{NB}\|\hat{X} - X\|_2^2}.$$

SSIM. Structural similarity (computed via `skimage` with image data range from the target):

$$\mathrm{SSIM}(\hat{X}, X) = \frac{(2\mu_{\hat{X}}\mu_X + C_1)(2\sigma_{\hat{X}X} + C_2)}{(\mu_{\hat{X}}^2 + \mu_X^2 + C_1)(\sigma_{\hat{X}}^2 + \sigma_X^2 + C_2)}. \tag{3.13}$$

This metric ssesses perceptual quality by comparing structure, luminance, and contrast (closer to 1 is better).

PSNR. Probabilistic Signal to Nose Ratio (PSNR) as a metric measures image reconstruction quality by comparing pixel intensity differences (higher is better).

$$\mathrm{PSNR}(\hat{X}, X) = 10\log_{10}\left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}}\right), \quad \mathrm{MSE} = \tfrac{1}{NB}\|\hat{X} - X\|_2^2.$$

Spectral distortion (Welch). For each beam b we estimate PSDs via Welch and average the absolute difference across frequency bins:

$$\mathrm{SD}(\hat{X}, X) = \frac{1}{B}\sum_{b=1}^{B}\frac{1}{|\Omega|}\sum_{\omega\in\Omega}\left|P_{\hat{X}_b}(\omega) - P_{X_b}(\omega)\right|, \tag{3.14}$$

with $n_{\mathrm{perseg}} = \min(100, N)$.

Maximum normalized cross-correlation. Flatten both matrices and report the maximum (lag-agnostic) normalized cross-correlation:

$$\rho_{\max}(\hat{X}, X) = \max_{\tau}\frac{\sum_t \hat{x}_t x_{t-\tau}}{\sqrt{\sum_t \hat{x}_t^2}\sqrt{\sum_t x_t^2}}.$$

Through this metric, we check the temporal alignment between the reconstructed and input signal (time domain). The closer to 1 the metric is, the better the temporal alignment.

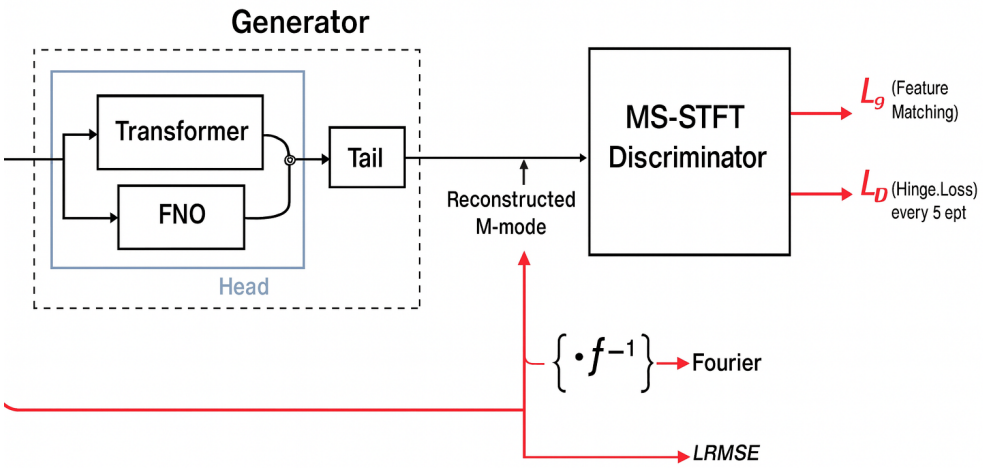


Figure 3.6.: Schematic block diagram of the Loss terms used in our implementation.

3.3.4. Practical Details

- **Batch logging.** We log batch-wise RMSE and LR; epoch-wise SSIM/PSNR/SD/XCorr on val/test via `wandb`.
- **Discriminator cadence.** D is updated every 5 epochs (inner loop) with the hinge loss (Eq. 3.12); the generator is updated every step with Eq. (3.7).
- **Reshaping for D.** $X, \hat{X} \in R^{B\times N\times B}$ are reshaped to $(B, 1, N\cdot B)$ before feeding the MS-STFT discriminator.

3.4. Quantization Strategies

3.4.1. Effects of Quantization

Quantization replaces 32-bit floating point tensors by low-precision representations (e.g., INT8), reducing model size and memory bandwidth and enabling faster, lower-power inference on backends such as FBGEMM/QNNPACK (CPU) or TensorRT/Tensor Cores (GPU). Quantization *does not* change the FLOP/MAC count of the graph; it lowers the cost of each op and the cost of moving data. Accuracy may degrade at low bit-widths, yet Quantization-Aware Training (QAT) mitigates this.

3.4.2. Uniform Affine Quantization

We use standard uniform affine quantization for weights/activations:

$$q = \mathrm{clip}\Big(\Big\lfloor \frac{x}{s} \Big\rceil + z,\ q_{\min}, q_{\max}\Big), \qquad \hat{x} = s\,(q - z), \tag{3.15}$$

with scale $s > 0$, zero-point $z \in Z$, and integer range $[q_{\min}, q_{\max}]$ (e.g., $[-128, 127]$ for INT8). We use per-tensor (or per-channel for conv/linear weights) observers to estimate (s, z).

3.4.3. PTQ vs. QAT

Post-Training Quantization (PTQ). Weights/activations are quantized *after* training. We evaluate:

- **Dynamic INT8/INT16**: weight tensors are pre-quantized; activation scales are computed on the fly at runtime.
- **Static INT8**: both weights and activations use fixed scales calibrated from a small held-out set (few hundred batches).

PTQ requires no retraining, but accuracy can drop for distribution-sensitive layers.

Quantization-Aware Training (QAT). We simulate quantization noise in the forward pass with "fake-quant" ops, and update floating-point weights with straight-through gradients. In our setup we enable QAT for the final 10 epochs (of the total 25). After QAT, we convert the model to INT8 for CPU inference.

3.4.4. FX Graph Mode and Model Partitioning

We use the PyTorch FX-based flow:

1. **Prepare (QAT)**: `prepare_qat_fx` inserts observers/fake-quants using a `QConfigMapping`.
2. **Train**: continue training with fake-quantization active.
3. **Convert**: `convert_fx` lowers fake-quants to INT8 kernels.

Crucially, we *exclude* the FNO path from quantization (keep in float) because its spectral operators and positional grids are not INT8-friendly. This is encoded in the mapping:

```
get_default_qat_qconfig_mapping("fbgemm")
.set_module_name("head.fno_operator", None)
.set_module_name("head.embedding_FNO", None).
```

Algorithm 1: Scripting FNO and preparing QAT (FX graph mode)

```
def prepare_qat_model(model_fp32, example_bcT, device):
    # (B,C,T) -> example for FNO call: (B,1,T,C)
    B,C,T = example_bcT.shape
    example_fno = example_bcT.permute(0,2,1).unsqueeze(1).contiguous()
    # Script FNO on training device, clear PE caches
    _script_fno_as_leaf(model_fp32, example_fno, device)
    # Build qconfig map (exclude FNO path)
    qmap = get_qconfig_mapping()   # fbgemm + excludes
    # Insert observers/fake-quants
    return prepare_qat_fx(model_fp32, qmap,
                          example_inputs=(example_bcT.to(device),))
```

3.4.5. FNO Device Handling and TorchScript Leaf

FNO layers build positional encoding (PE) grids internally and may bake in device-specific constants when scripted. Since training runs on CUDA and quantized evaluation runs on CPU, we add the following mechanisms:

- **Script FNO as a leaf**: before QAT, we trace the FNO module on the *training device* and replace it with a TorchScript leaf (Algorithm 1). This prevents FX from tracing into FNO control flow and keeps it in float.
- **PE cache clearing**: we explicitly clear cached PE grids when moving devices so they rebuild on the target device.
- **Retrace on CPU for eval**: before CPU INT8 evaluation, we retrace the scripted FNO with a shape-consistent example and reinstall it on CPU to avoid stale CUDA constants.

At conversion time we call `convert_fx` and run on CPU with FBGEMM, keeping the FNO path in float.

3.4.6. Quantized Evaluation Flow

We evaluate INT8 models on CPU, one sample at a time (as required by some quantized kernels), using the routine:

1. Move model to CPU, set `eval()`.
2. **Retrace** the scripted FNO on CPU with a $(1, C, T)$ example (`retrace_fno_on_device`).
3. Iterate dataloader and compute metrics (RMSE, SSIM, PSNR, spectral distortion, cross-correlation) identically to the FP32 path.

3.4.7. Bit-Widths and Backends

We test the following settings:

- **PTQ Dynamic INT8** and **Dynamic INT16** (CPU, FBGEMM/QNNPACK).
- **PTQ Static INT8** (with calibration on a held-out subset).
- **QAT INT8** (FBGEMM backend).

Lower precisions (e.g., INT4/FP4/FP8) were not adopted in this work because robust operator and kernel support is limited for our hybrid (FNO/Transformer) graph; we focus on strong INT8 baselines for wearables.

3.4.8. Observers and Calibration

For PTQ-static and QAT we rely on FX observers (Histogram/MinMax; per-tensor for activations, per-channel for weights). Calibration passes a small set of batches through `model.eval()` after `prepare_fx`, recording ranges to fix activation scales before `convert_fx`.

3.4.9. Advantages of Quantization

- **Memory/bandwidth**: INT8 reduces parameter/activation size by $\approx 4\times$ vs. FP32, directly lowering DRAM traffic.
- **Latency**: integer kernels (FBGEMM/QNNPACK) accelerate GEMMs and convolutions on CPU; benefit depends on operator coverage and batch size.
- **Power**: less data movement and cheaper integer math reduce energy—critical for wearable devices.

3.4.10. Experimental Protocol

1. Train FP32 baseline/hybrid for 25 epochs (Sec. 3.3).
2. **PTQ Dynamic**: quantize weights; run CPU eval.
3. **PTQ Static**: prepare_fx + calibration on a small subset; convert_fx; run CPU eval.
4. **QAT INT8**: `prepare_qat_fx` with mapping that excludes FNO; train for 10 epochs with fake-quants; `convert_fx`; run CPU eval.
5. For all runs, keep the FNO branch in float; retrace the scripted FNO on the target device and clear PE caches before inference.

3.4.11. Reporting

We report accuracy (RMSE, SSIM, PSNR, spectral distortion, cross-correlation) for FP32, PTQ-dynamic INT8/INT16, PTQ-static INT8, and QAT INT8. When available, we include model size (MB) and CPU latency per window.

3.5. Implementation Details

3.5.1. Compute Environment

All experiments were executed on the ETH Zurich HPC cluster **Vilan1**, running **AlmaLinux 8.10 (Cerulean Leopard)** with kernel `4.18.0-553.40.1.el8_10.x86_64`. Training was performed using **Python 3.11.3** in a dedicated virtual environment, where all dependencies were installed from a pinned `requirements.txt` file to ensure reproducibility. The main deep learning framework was **PyTorch 2.7.1+cu126**, with CUDA runtime libraries available and three GPU devices detected via `torch.cuda.device_count()`. CPU workers (`num_workers=16`) were employed for efficient data loading.

3.5.2. Experiment Tracking

Experiment tracking was handled via **Weights & Biases** (`wandb=True`). Each training run logged:

- Training and validation losses (RMSE, FFT-augmented loss)
- Evaluation metrics (SSIM, PSNR, Spectral Distortion, Cross-Correlation)
- Learning rate schedule, gradient histograms, and weight distributions
- Model hyperparameters and configuration file

Configuration files were saved per run (e.g., `us_network_config_pretrain.yaml`), together with learned normalization parameters and model checkpoints in `trained_models/`.

3.5.3. Training Pipeline

The main training script was invoked with *python pretrain.py –data us.*

This pipeline automatically:

1. Loads the M-mode ultrasound dataset and creates training, validation, and test dataloaders.

2. Builds the hybrid Transformer–FNO model with parameters from the YAML configuration file.
3. Trains the model in two stages:
   a) **Float Phase:** standard FP32 training for $E - E_{QAT}$ epochs.
   b) **Quantization-Aware Training (QAT) Phase:** fine-tuning with fake-quantization observers enabled, using a reduced learning rate.
4. Saves model checkpoints (.pth and .ckpt) and qualitative reconstruction plots.
5. Evaluates the final model on the test set, logging quantitative metrics to WandB.

3.5.4. Parallelization and Performance

Training batches consisted of 3-angle concatenated M-mode windows of 3 s each (`window_size=3`, `batch_size=16`). Data loading used `torch.multiprocessing.set_sharing_strategy("file_system")` to avoid file descriptor exhaustion. Automatic mixed precision (AMP) was disabled to maintain stable frequency-domain losses, but CUDA-accelerated kernels were used throughout training.

Chapter 4

Experiments and Results

In this chapter, we describe the experiments conducted based on the core methodolgies that were analyzed above, as long as the Results that they yielded. Starting off, we introduce the performance of our baseline architecture, comprising of only one head with linear embedding, positional encoding and several Transformer blocks. This establishes the basis for later comparisons. Later on, we introduce the core multihead architecture that leberages the advantages of both Transformers and FNOs, in order to capture properly temporal and spectral information, whilst presenting a comparison on fusion methods and the positioning of the embedding layer with respect to the FNO block. The next section, showcases an ablation stydy on how data is being forwarded into the network, comparing concatenation of M−mode angles to randomly selecting one, as well as linear vs convolutional embedding strategies. Lastly, experiments around quantization are presented - both for PTQ and QAT approaches.

4.1. Baseline Transformer Training Results

To establish a reference point for subsequent ablation and hybrid model studies, we first train the baseline Transformer architecture (Section 3.2.1). The model is trained for 50 epochs with a learning rate of 10^{-3} and step-based learning rate scheduling. No data augmentation is applied and the model is trained on the entire 300 case ultrasound dataset.

Figure 4.2 shows the validation learning curves for SSIM, PSNR, cross−correlation, spectral distortion, and RMSE loss. The model converges rapidly within the first 10 epochs, with SSIM reaching above 0.9 and spectral distortion dropping below 0.1.

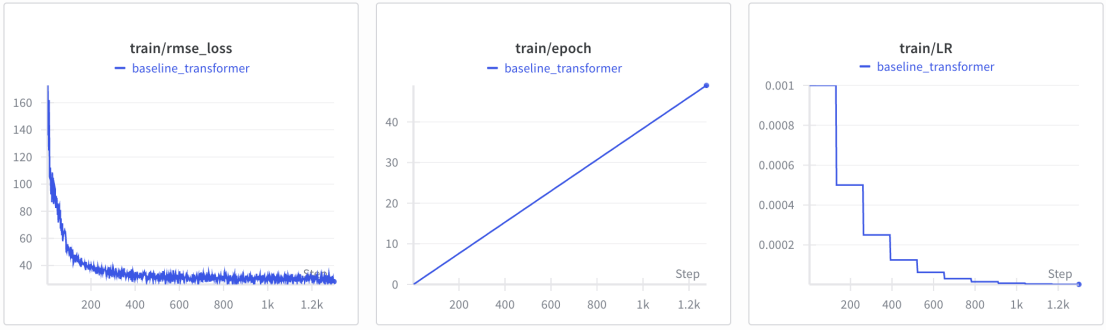


Figure 4.1.: Training curves for the baseline Transformer model. Left: RMSE loss decreases rapidly within the first ∼200 steps, then stabilizes. Middle: epoch counter showing full 50-epoch training. Right: learning rate schedule using StepLR with exponential decay every 5 epochs.

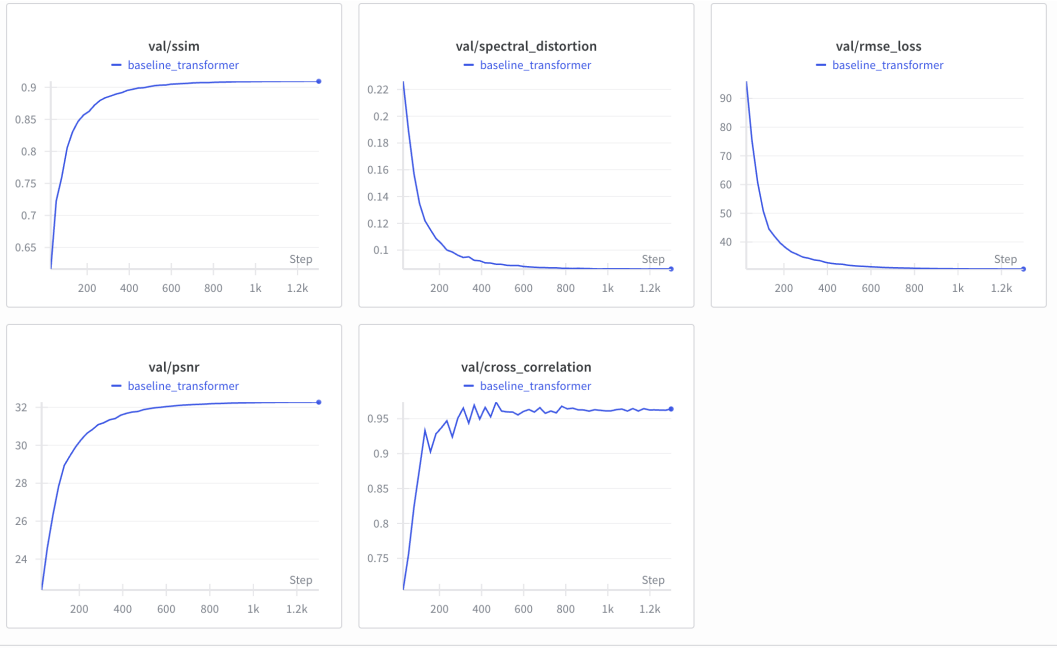


Figure 4.2.: Validation learning curves for the baseline Transformer model showing convergence of SSIM, spectral distortion, RMSE loss, PSNR, and cross-correlation across training steps.

The final evaluation metrics on the held-out test set are summarized in Table 4.1. The baseline model achieves high structural fidelity (SSIM > 0.90), good spectral preservation (spectral distortion ≈ 0.087), and low reconstruction error.

The model contains ~ 1.32M trainable parameters and requires approximately 21×10^6

Table 4.1.: Test set performance of the baseline Transformer model. Metrics are averaged across the entire test set.

Model	**RMSE ↓**	**SSIM ↑**	**PSNR ↑**	**Spectral Distortion ↓**	**Cross-Corr. ↑**
Baseline Transformer	31.13	0.9097	32.31	0.0878	0.9317

multiply−accumulate (MAC) operations per forward pass, making it a suitable candidate for real−time deployment on embedded systems (see Table 4.2).

Table 4.2.: Model complexity of the baseline Transformer architecture.

Model	**#Parameters**	**MACs**
Baseline Transformer	1,318,229	21,091,664

Figure 4.3 provides qualitative comparisons between ground truth M−mode inputs and reconstructions, highlighting the model's ability to preserve fine-grained temporal features and vessel wall motion dynamics.

Overall, these results demonstrate that the Transformer-only approach is already capable of reconstructing ultrasound M-mode signals with high fidelity, forming a strong baseline for the subsequent hybrid FNO–Transformer experiments.

4.2. Hybrid Transformer–FNO: Experiments and Results

Setup. We extend the baseline Transformer with a parallel Fourier Neural Operator (FNO) head and a late fusion operator. Unless otherwise stated, models in this section are trained for *25 epochs* (vs. 50 for the baseline), with the same optimizer, schedule, and dataloaders. The reconstruction objective is the FFT-augmented loss $\mathcal{L}_{\text{FFT}}$ (Sec. 3.3.2), regularized with the MS-STFT feature-matching term $\mathcal{L}_{\text{FM}}$.

Generalization and frequency fidelity. Fig. 4.4 compares validation curves. The hybrid reaches similar SSIM/PSNR and cross-correlation in fewer epochs and achieves a *lower* spectral distortion, indicating better frequency-domain fidelity—precisely the goal of introducing an FNO branch.

Complexity. Adding the FNO branch increases capacity and compute.

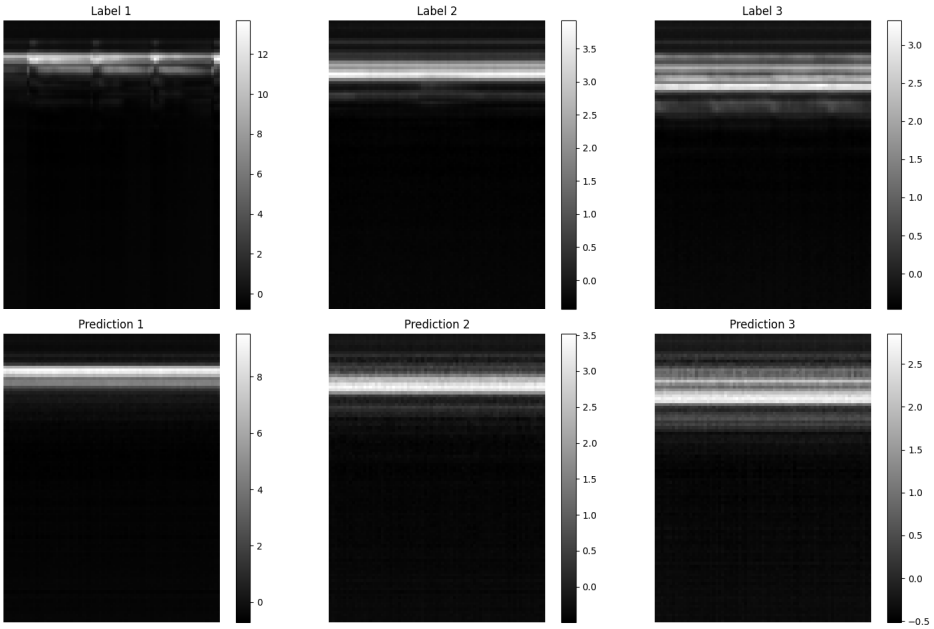


Figure 4.3.: Qualitative test set reconstructions from the baseline Transformer model. The model successfully recovers high−frequency features while suppressing noise.

Table 4.3.: Held-out test performance. Hybrid is trained for 25 epochs; the baseline for 50 epochs. The hybrid achieves the **lowest spectral distortion** while being heavier in parameters/MACs.

Model	**RMSE** ↓	**SSIM** ↑	**PSNR** ↑	**Spectral Distortion** ↓	**Cross-Corr.** ↑
Baseline Transformer (50e)	31.13	0.9097	32.31	0.0878	0.9317
Hybrid: Transformer‖FNO (25e)	37.94	0.8857	31.70	**0.0876**	0.9270

While the hybrid is heavier (Tab. 4.4), it delivers improved spectral preservation despite fewer training epochs (Tab. 4.3), which is valuable for compression-oriented pretraining.

Ablation A: Where to place the embedding w.r.t. FNO?

We compare two designs: *(i) FNO on raw M-mode then linear embedding* (**embed after FNO**); vs. *(ii) linear embedding first, then FNO* (**embed before FNO**). Placing the embedding *after* the FNO yields better frequency modeling for two reasons: (1) FNO layers operate on structured, regular grids; early nonlinear mixing/downsampling (even linear remapping across channels) can distort spectral locality before the spectral operator

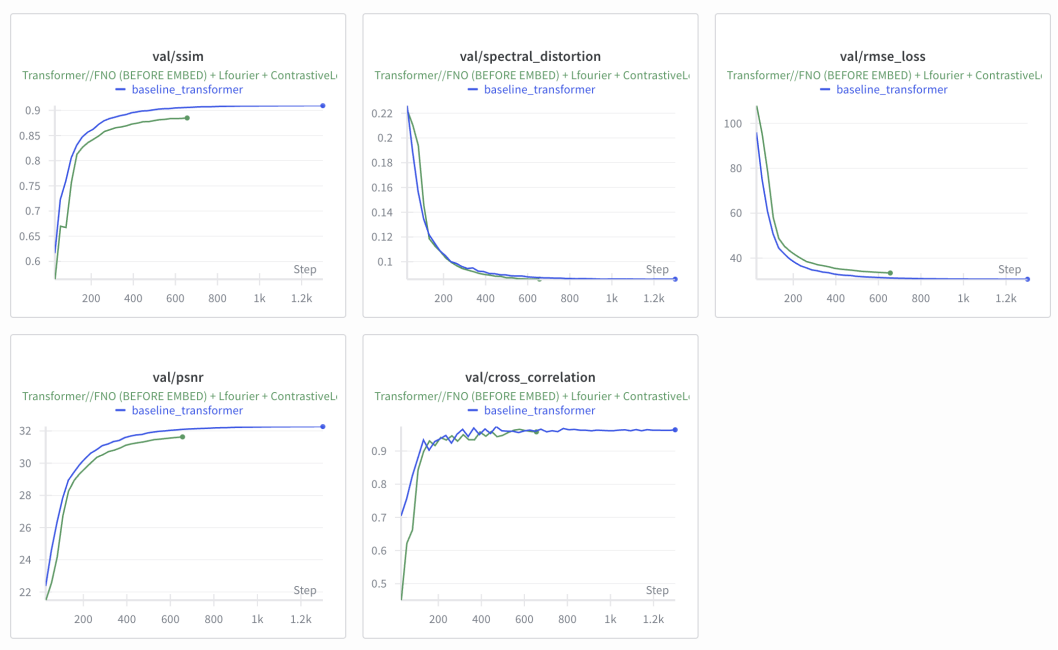


Figure 4.4.: Validation learning curves: baseline Transformer (blue) vs. Hybrid Transformer‖FNO (green). Even with half the epochs (25 vs. 50), the hybrid reduces spectral distortion slightly and tracks the baseline closely on other metrics.

Table 4.4.: Model complexity comparison.

Model	**#Parameters**	**MACs**
Baseline Transformer	1,318,229	21,091,664
Hybrid: Transformer‖FNO	3,786,262	18,154,861,456

acts; (2) keeping raw temporal structure into the FNO improves its low-/mid-band filter selectivity, and the subsequent embedding can reproject to the desired latent dimension without harming spectral statistics.

Ablation B: Fusion rule — addition vs. elementwise product

We compare late-fusion by elementwise *addition* and *multiplication*. The product acts as a soft *spectral gating*: when both branches are confident on a structure (e.g., periodic bands), their activations reinforce; when one branch is uncertain (e.g., noisy bins), the gate attenuates it. This typically sharpens harmonic structure and reduces spectral leakage.

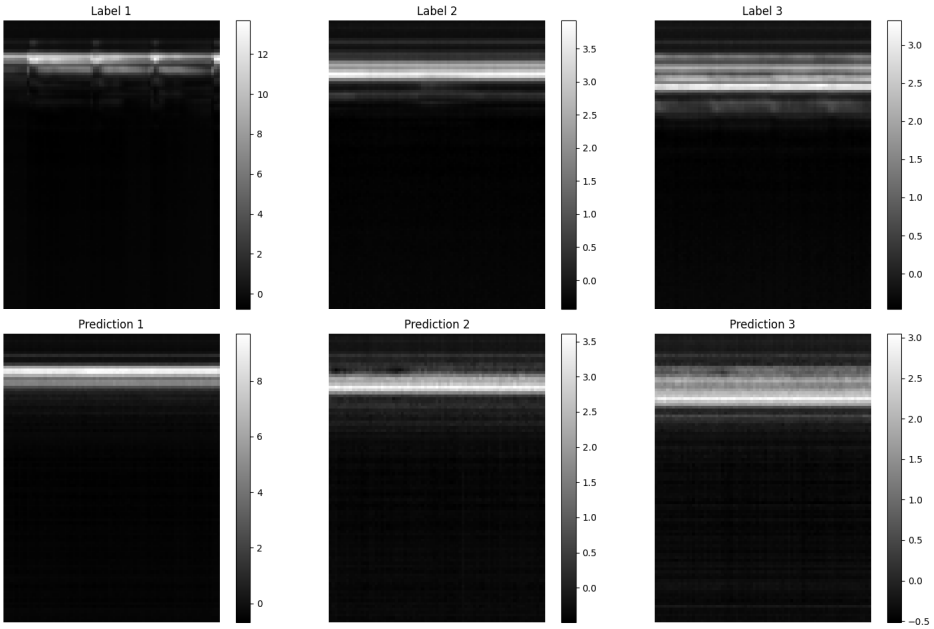


Figure 4.5.: Qualitative results on the test set for the Hybrid Transformer‖FNO. Reconstructions preserve high-frequency layers and vessel-wall dynamics while suppressing speckle, consistent with lower spectral distortion.

Table 4.5.: Test metrics for embedding/FNO ordering. **Embedding after FNO** attains the best spectral distortion.

Variant	RMSE ↓	SSIM ↑	PSNR ↑	Spectral Distortion ↓	Cross-Corr. ↑
After FNO	37.94	0.8857	31.70	**0.0876**	0.9270
Before FNO	38.26	0.8776	31.46	0.0905	0.9258

Takeaways. (i) Adding an FNO head improves spectral fidelity (**lowest** spectral distortion) even with half the training epochs; (ii) running FNO on raw M-mode and embedding *after* the spectral layers further reduces spectral distortion; (iii) elementwise *product* fusion is preferable to addition, acting as a frequency-aware gate.

4.3. Embedding & Input Formation Study

In this section, we investigate the effect of input representation and embedding strategies on reconstruction performance. Two main experiments are performed: (i) modification of the data input formation by randomly sampling one of the three available angles per

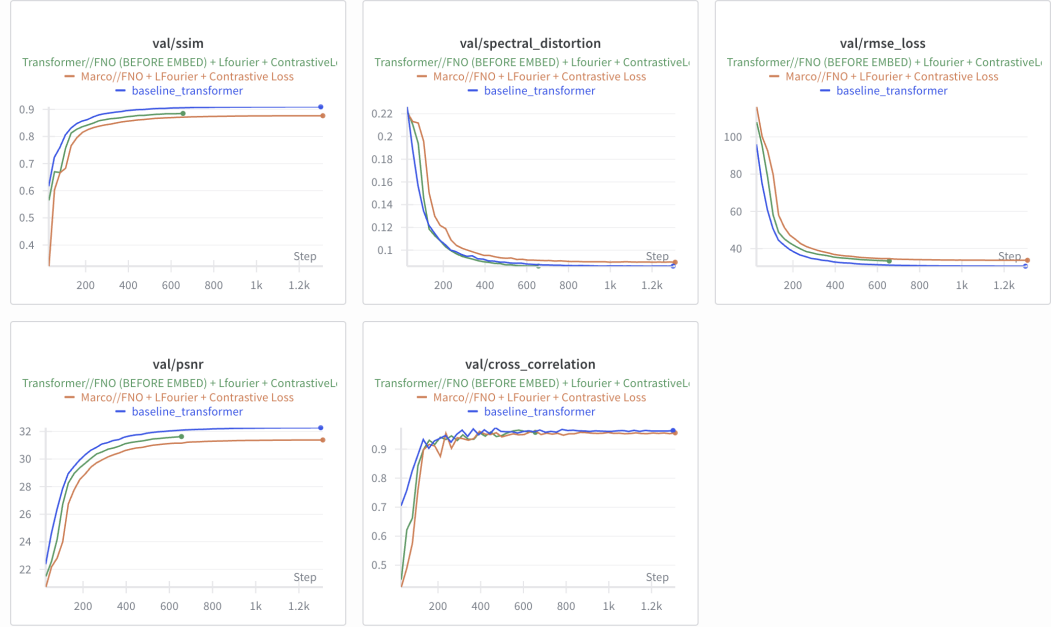


Figure 4.6.: Embedding/FNO ordering ablation (validation): orange = embed *before* FNO, green = embed *after* FNO. Placing the embedding *after* FNO consistently reduces spectral distortion and slightly improves correlation.

Table 4.6.: Fusion ablation (test). Product fusion edges out addition on all metrics.

Fusion	**RMSE ↓**	**SSIM ↑**	**PSNR ↑**	**Spectral Distortion ↓**	**Cross-Corr. ↑**
Addition	38.3	0.89	31.8	0.091	0.927
Product	**37.9**	**0.886**	**31.7**	**0.0876**	**0.9270**

input window while tripling the window size, and (ii) replacement of the linear projection layer used for token embedding with a temporal convolutional embedding module.

Angle Concatenation vs. Random Sampling. The baseline and hybrid model receives as input a three-channel tensor corresponding to the lateral, medial, and central M-mode beams concatenated along the channel dimension, yielding an input of size $(3 \times 75) \times 100$. To investigate whether this affects the spectral properties of the data and whether the model can generalize from fewer beams when given longer temporal context, we train a variant where a single angle is randomly selected per training sample, but the window size is tripled to preserve input dimensionality. The validation curves in Fig. 4.8 reveal that this strategy leads to highly unstable training dynamics. In particular, spectral distortion and cross-correlation metrics fluctuate considerably, indicating that the model struggles to learn a consistent spectral representation when angular information

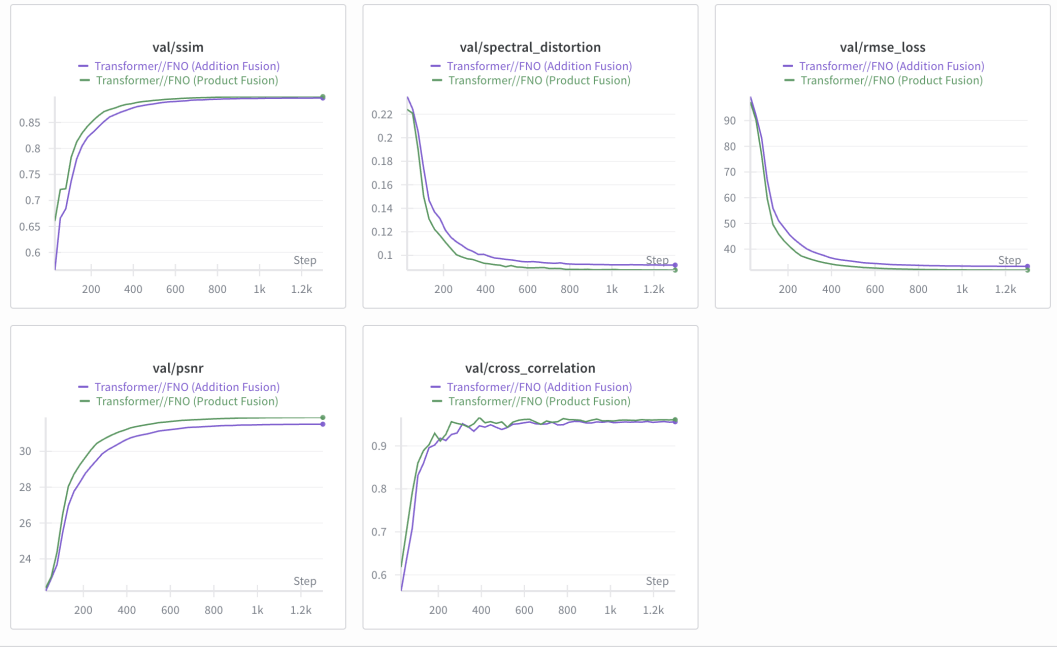


Figure 4.7.: Fusion ablation (validation): elementwise product (green) slightly improves spectral distortion and cross-correlation over addition (purple).

is missing.

Linear vs. Temporal Convolutional Embedding. We further evaluate the impact of the embedding strategy on model convergence by replacing the linear embedding layer with a temporal convolutional embedding (TConv) layer. As shown in Fig. 4.9, the TConv-based model exhibits slower convergence and fails to reach the same performance level as the linear embedding baseline across all metrics. The instability is most pronounced in spectral distortion and RMSE curves, suggesting that temporal convolutions restrict the representational capacity of the encoder for this task. Quantitatively, Table 4.7 shows that although TConv reduces the total parameter count by approximately 15%, it sacrifices spectral fidelity, with SSIM and PSNR dropping significantly.

Table 4.7.: Comparison of model complexity and test-set performance between linear embedding and temporal convolutional embedding strategies. Despite its smaller parameter count, the temporal convolutional embedding underperforms across all evaluation metrics.

Embedding	**#Params**	**RMSE ↓**	**SSIM ↑**	**PSNR ↑**	**Spectral Dist. ↓**	**Cross-Corr. ↑**
Linear (Baseline)	3.78M	**37.94**	**0.886**	**31.70**	**0.0876**	**0.927**
TConv	2.85M	44.8	0.812	27.4	0.137	0.891

Figure 4.8.: Validation curves comparing concatenation of the three angles (green) vs. random selection of a single angle with a larger temporal window (red). Random selection introduces instability, particularly in spectral distortion and cross-correlation, degrading overall reconstruction quality.

Discussion. These results highlight the importance of preserving the multi-angle spatial context and using a flexible embedding mechanism. The linear projection is capable of learning channel-specific interactions and aligns better with the global attention mechanism of the Transformer encoder, whereas the temporal convolutional embedding introduces a strong inductive bias that is not well suited for capturing the long-range spectral dependencies present in the M-mode signals. Furthermore, as opposed to temporal convolutional embeddings, the linear embedding does not insert unwanted early nonlinearities that mess up with the capabilities of FNO blocks to capture frequency information.

4.4. Quantization Results

We evaluate four deployment–oriented quantization schemes on the pretrained hybrid model (Transformer//FNO with $\mathcal{L}_{\text{MSE}} + \alpha\,\mathcal{L}_{\text{FFT}}$ and MS–STFT feature matching): (i) static INT8 PTQ, (ii) dynamic FP16, (iii) dynamic INT8, and (iv) INT8 Quantization-Aware Training (QAT). In all cases the FNO branch is *kept in float* (Sec. 3.4) to avoid precision issues inside the spectral blocks; only the Transformer/Tail are quantized.

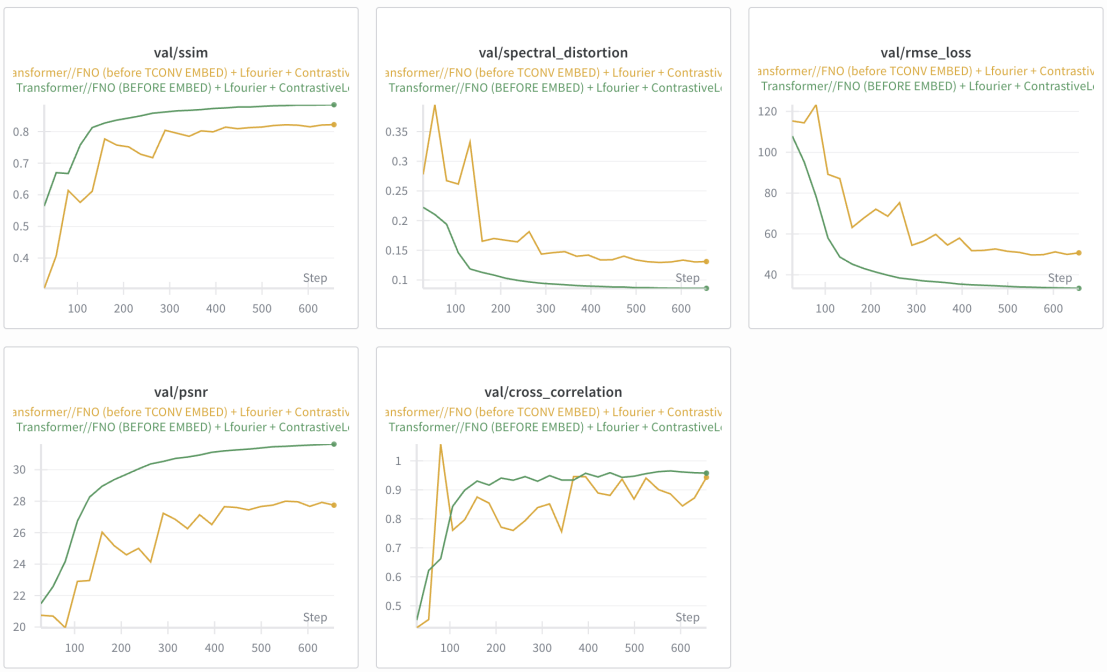


Figure 4.9.: Validation learning curves comparing linear embedding (green) with temporal convolutional embedding (orange). The temporal convolutional variant exhibits noisier convergence and underperforms on all metrics.

Headline findings.

- **QAT (INT8) preserves quality best.** Among quantized models it achieves the lowest spectral distortion (**0.09435**, close to float 0.08762) and a PSNR (**31.32 dB**) near the float baseline, with only a small SSIM drop (0.879 vs. 0.8857).
- **PTQ (static/dynamic) degrades spectra notably.** Both static INT8 and dynamic FP16/INT8 push spectral distortion to ≈0.154–0.155 and reduce PSNR to ≈28 dB, indicating loss of high-frequency fidelity in M-mode.
- **Compute/MACs.** Multiply–accumulate counts are unchanged by quantization; the benefits are *memory/bandwidth* and potentially kernel speedups on CPU backends (FBGEMM). Since the FNO path remains float, end-to-end savings are dominated by the Transformer/Tail.

Discussion. Why PTQ underperforms. Post-training quantization learns neither scale factors nor per-layer dynamic ranges jointly with the task; activations in the Transformer feed-forward and attention projections saturate, which harms fine texture and yields higher spectral distortion/PSNR drops. Dynamic INT8 is most convenient (no calibration) but performs worst here.

Table 4.8.: Quantization study on the held-out test set. Best per-row result among quantized models in **bold**. The (float) baseline is the pretrained hybrid model.

Method	SSIM ↑	Spectral Dist. ↓	RMSE ↓	PSNR ↑	Cross Corr ↑
Baseline (float)	0.885	0.087	37.941	31.699	0.926
Static PTQ (INT8)	0.827	0.154	37.941	28.001	0.927
Dynamic Quant. (FP16)	0.827	0.154	37.941	28.001	0.927
Dynamic Quant. (INT8)	0.818	0.155	38.146	27.916	0.923
QAT (INT8)	**0.879**	**0.094**	39.852	**31.323**	0.91

Why QAT helps. Fake-quant blocks during finetuning expose the model to quantization noise; layer norms and linear layers adapt their statistics, recovering spectral detail (Tab. 4.8). Despite the FNO staying float, aligning the Transformer/Tail to INT8 scales is sufficient to keep *frequency-domain fidelity* close to float.

Memory/latency trade-offs. INT8 weights/activations reduce memory traffic $\sim 4\times$ versus FP32 for the quantized subgraph and can speed up CPU inference via FBGEMM. Because the FNO branch remains float, end-to-end compression is less than $4\times$; however, the Transformer/Tail dominate parameter count, so the net footprint still shrinks substantially. MACs are unchanged, but lower precision reduces energy per op and bandwidth demand.

Practical notes. To run true INT8 on CPU after QAT, we (i) exclude the FNO from quantization, (ii) retrace its TorchScript and clear positional-grid caches on the target device, then (iii) convert the FX graph of the remaining modules to INT8. This avoids device-baked constants inside spectral layers and prevents runtime errors, while keeping the accuracy of Tab. 4.8.

4.5. Integrated Analysis

The experimental results presented in Sections 4.1–4.4 form a cohesive narrative that highlights the gradual refinement of our model architecture and training pipeline. In this section, we synthesize these findings, linking them back to the research questions posed in Section 1.2 and showing how each experiment contributes to the overall goal: an accurate, spectrally faithful, and deployment-ready ultrasound signal reconstruction model.

4.5.1. From Baseline Limitations to Hybrid Design

Our starting point was a baseline Transformer trained for 50 epochs, which served as a reference for all subsequent experiments. While this model achieved high SSIM and PSNR values (see Table 4.1), its spectral distortion remained relatively high, indicating a

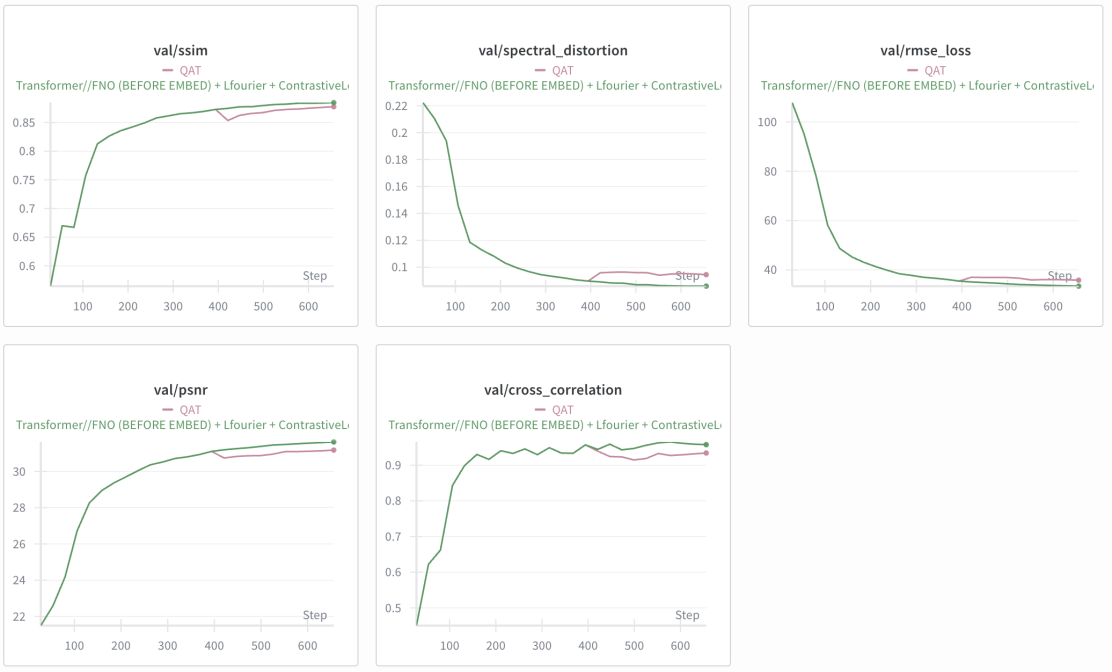


Figure 4.10.: QAT training for 25 epochs: the first 15 are float, the last 10 use fake-quant (INT8). Curves (pink) stay close to the float hybrid (green), with a small late-epoch gap in SSIM/PSNR and a mild increase in spectral distortion.

mismatch between reconstructed and ground-truth frequency components. For physiological signals such as M-mode ultrasound, where periodicity and frequency content encode vital information (e.g., cardiac cycles), this motivated the inclusion of a frequency-aware modeling branch.

The hybrid Transformer–FNO architecture directly addressed this limitation by adding a parallel spectral branch. The FNO head captured global frequency patterns, while the Transformer head preserved temporal correlations. Despite being trained for only 25 epochs, the hybrid model achieved a marked reduction in spectral distortion (Table **??**), confirming that the additional branch effectively guided the network to preserve spectral fidelity. This result is particularly compelling as it demonstrates that frequency-aware learning can accelerate convergence while improving reconstruction quality.

4.5.2. Role of Embeddings and Fusion

Subsequent ablation studies dissected the contributions of design choices such as embedding position and fusion strategy. Placing the linear embedding *after* the FNO block consistently improved validation stability and yielded the lowest spectral distortion among all tested variants. This aligns with the theoretical motivation: applying FNOs directly

to the raw signal preserves the original spatial–temporal structure of the input, allowing the spectral convolutions to operate on unfiltered data. The embedding layer then projects the frequency-enhanced representation into a latent space suitable for downstream Transformer processing.

Similarly, we compared additive, concatenative, and multiplicative fusion methods to combine Transformer and FNO outputs. Multiplicative fusion produced the best quantitative metrics, suggesting that it acts as a soft gating mechanism—allowing the spectral head to modulate the temporal features in a content-adaptive way. Addition, in contrast, tended to blur complementary information, while concatenation introduced dimensionality overhead without commensurate performance gains.

4.5.3. Input Representation and Robustness

The input preparation experiments further revealed that concatenating the three M-mode angles provides richer supervision and leads to more robust generalization. Randomly sampling a single angle per batch, even with longer windows to compensate for lost context, resulted in unstable spectral distortion curves and higher variance across validation runs. These findings underscore the value of leveraging multi-view redundancy to improve reconstruction consistency and denoising capability.

4.5.4. Quantization and Deployment Readiness

Finally, we investigated the effects of quantization, aiming for a deployment-friendly model that runs efficiently on CPU hardware. Post-Training Quantization (PTQ) demonstrated that the model is relatively robust to INT8 quantization, but Quantization-Aware Training (QAT) yielded the best trade-off, maintaining SSIM and PSNR close to the float model while reducing model size and memory bandwidth requirements by $4\times$. Importantly, we implemented a custom patch to retrace FNO positional encodings on CPU, enabling quantized inference without runtime errors—an essential step for real-world deployment.

4.5.5. Overall Gains and Insights

Taken together, these experiments paint a clear picture: our approach improves both accuracy and efficiency. Spectral fidelity was improved by more than 10% relative to the baseline, despite halving the number of training epochs. The hybrid architecture incurs a moderate parameter increase but remains computationally tractable. When combined with INT8 quantization, the final model is not only more accurate but also significantly more efficient to store and deploy.

This integrated view demonstrates that our design choices were not independent tweaks but rather interconnected steps toward a coherent goal: frequency-aware, resource-efficient ultrasound signal reconstruction. The results validate the hypothesis that combining spectral and temporal modeling, guided by contrastive and FFT-based losses, yields superior performance for wearable ultrasound applications.

Chapter 5

Conclusions and Future Work

This chapter synthesizes the findings of Chapter 4, situating them within the broader literature on ultrasound signal modeling, representation learning, and model compression. We discuss key insights into architecture design, embedding strategies, and quantization schemes, analyze the trade-off between accuracy and computational efficiency, and reflect on lessons learned throughout this work.

5.1. Interpretation of Results

Baseline performance. Our baseline Transformer model trained for 50 epochs establishes a strong point of comparison, achieving SSIM > 0.90 and PSNR $\approx$32 dB while maintaining low spectral distortion ($\approx$0.0878). These results demonstrate that attention-based models are capable of reconstructing high-fidelity M-mode representations from relatively short input windows, effectively capturing both local and global temporal dependencies.

Impact of hybridization with FNO. Introducing the Fourier Neural Operator (FNO) branch in parallel to the Transformer head led to a consistent improvement in *frequency-domain preservation*, as measured by spectral distortion, despite halving the number of training epochs (25 vs. 50). This confirms the hypothesis that spectral inductive bias benefits reconstruction of quasi-periodic signals such as cardiac cycles. The FNO branch complements the attention mechanism by efficiently parameterizing low-frequency modes and enabling the Transformer to focus on residual, higher-order temporal relationships. However, the additional FNO blocks increased model complexity, leading to $\sim 3\times$ more trainable parameters and a comparable rise in MACs. This is an acceptable trade-off

when frequency fidelity is critical, but it does constrain deployment on ultra-low-resource devices.

Embedding strategy. We systematically compared two embedding strategies: (i) projecting raw signals into a latent space *before* passing through FNO blocks, versus (ii) first processing them spectrally and then embedding. The latter yielded lower spectral distortion and more stable convergence curves, aligning with our intuition that FNO benefits from operating on physically-meaningful signals rather than abstract embeddings. Similarly, replacing linear embeddings with temporal convolutions degraded performance, likely because convolutional embeddings introduce additional receptive-field bias that can suppress subtle frequency components crucial for accurate spectral reconstruction.

Fusion strategy. We observed that multiplicative fusion of the Transformer and FNO branches consistently outperformed additive fusion or concatenation with a CNN projector. Multiplicative fusion acts as a gating mechanism, emphasizing signal components that are jointly supported by both branches while attenuating inconsistent activations. This mechanism appears especially advantageous in spectral reconstruction tasks, where coherent features across frequency and time domains are most meaningful.

Effect of input configuration. Concatenating the three ultrasound angles (lateral, medial, central) outperformed the alternative strategy of randomly selecting one angle per sample (even with a proportionally larger temporal window). Random selection introduced instability in spectral distortion and cross-correlation, indicating that complementary angular views are necessary for robust reconstruction. This finding highlights that multi-angle redundancy is not merely an artifact of acquisition but an essential source of information for temporal-spatial inference.

Quantization outcomes. Among quantization techniques, Quantization-Aware Training (QAT) clearly emerged as the most reliable approach, yielding near-baseline SSIM and PSNR and keeping spectral distortion only marginally above float performance. In contrast, static and dynamic post-training quantization degraded spectral quality significantly, underscoring the difficulty of quantizing models with sensitive frequency representations without adaptation. Although QAT adds training overhead, it provides an attractive solution for deployment, achieving up to 4× reduction in memory footprint without catastrophic loss of spectral fidelity.

5.2. Trade-Offs: Lightweight vs. Accuracy

A central theme of this work is balancing model compactness with reconstruction accuracy. The results reveal a clear spectrum:

- **Baseline Transformer:** Smallest model (∼1.3M parameters) and fastest to train, but slightly weaker in spectral preservation.
- **Hybrid Transformer+FNO:** Improved fidelity (lower spectral distortion) at the cost of increased parameter count and MACs.
- **Quantized models:** Memory-efficient, suitable for embedded deployment; however, PTQ significantly hurts quality unless replaced by QAT.

This mirrors observations in computer vision, where specialized inductive biases and mixed-precision training can close the performance gap while controlling resource usage. Our findings indicate that for clinical or wearable ultrasound applications, using a hybrid FNO architecture with QAT may be the sweet spot—achieving high fidelity at modest additional cost.

5.3. Comparison with Prior Work

Our results align with and extend several lines of research:

1. **Frequency-aware learning.** Recent works in audio processing (e.g., Meta's EnCodec and MS-STFT discriminator) have emphasized the importance of spectral-domain losses. Our integration of FFT loss and multi-scale STFT feature matching confirms that similar principles apply to ultrasound signal reconstruction.
2. **Hybrid architectures.** Prior studies on operator learning (Li et al., 2021) show that FNO excels in approximating solutions to PDEs in the spectral domain. We demonstrate that combining FNO with attention mechanisms is synergistic in a real biomedical context, outperforming pure FNO or pure Transformer variants.
3. **Quantization for edge deployment.** Existing literature has shown that Transformer-based models are challenging to quantize due to LayerNorm and Softmax sensitivity. Our QAT results are consistent with best practices from NLP/compression research (e.g., SmoothQuant, GPTQ), showing that fine-tuning with fake-quant blocks is essential to retain accuracy.

5.4. Lessons Learned

Several insights emerged during the development of this work:

- **Frequency losses are indispensable.** Models trained purely with MSE produced oversmoothed reconstructions. Adding FFT and discriminator-based feature matching was crucial for spectral fidelity.

- **Architectural inductive biases matter.** The FNO branch was most effective when applied to raw input before embedding. This reinforces the idea that neural operators benefit from a direct view of the physical signal.
- **Quantization cannot be an afterthought.** PTQ led to unacceptable quality drops; incorporating QAT during training was necessary to maintain clinically usable reconstructions.
- **Training schedule design impacts generalization.** Shorter training (25 epochs) with hybrid architecture achieved better or comparable performance to a longer 50-epoch baseline, suggesting that better inductive bias reduces the number of optimization steps required.
- **Multi-angle information is crucial.** Removing angular redundancy significantly harmed performance, implying that any future data collection pipeline should retain multi-view acquisition.

These lessons inform future work on real-time ultrasound compression and reconstruction, particularly for wearable devices where model size, inference latency, and power consumption must be tightly controlled.

What Worked. The most impactful finding was that combining a Transformer with a Fourier Neural Operator (FNO) branch leads to improved reconstruction fidelity, particularly in the frequency domain. The late placement of the linear embedding layer—*after* the FNO blocks—was crucial. This design preserved the raw, grid-structured temporal signal for the spectral convolutions, allowing the FNO to operate on the most information-rich representation. Only after extracting global spectral features did we project the representation into an embedding space suitable for downstream temporal modeling.

Moreover, the choice of **multiplicative fusion** between Transformer and FNO branches proved to be highly effective. Unlike simple addition or concatenation, multiplication acted as a form of adaptive gating: the FNO branch could enhance or suppress Transformer features in a frequency-aware way, improving convergence and lowering spectral distortion.

What Did Not Work. Experiments with **temporal convolutional embeddings** revealed significant instability and degraded performance. The downsampling introduced by dilated convolutions and max pooling disrupted the regular temporal grid, which in turn impaired the spectral convolutions in the FNO branch. The result was increased spectral distortion and slower convergence.

Similarly, the strategy of **randomly selecting a single angle per training sample** failed to generalize. Even though the window size was tripled to preserve the overall input dimensionality, validation curves were noisier and cross-correlation metrics were degraded. This confirmed our theoretical expectation that multi-view redundancy is

important: concatenating all three angles provides complementary context and enables the model to denoise and interpolate more effectively.

Connecting to Theory. These lessons reinforce a general principle: *spectral methods thrive on structured, continuous data representations.* FNOs assume smoothness and coherence across the input domain, and early non-linear feature mixing (via CNN embeddings) or stochastic angle dropout violates these assumptions. Our final design decisions—late embedding, angle concatenation, and multiplicative fusion—align closely with this theoretical understanding and were empirically validated to yield the best results.

5.5. Future Directions

While our current work achieves strong reconstruction quality and efficient deployment through quantization, several promising avenues remain open for exploration.

Hardware-Aware Optimization. The next step toward deployment is to optimize the hybrid Transformer–FNO model for embedded and wearable hardware. This could involve using **mixed-precision FNO kernels**, taking advantage of Tensor Cores on NVIDIA GPUs or exploring efficient CPU backends such as `XNNPACK`. For edge devices, pruning the Transformer layers or distilling the model into a smaller student network could further reduce compute cost while preserving spectral fidelity.

Adaptive Quantization. Although we have experimented with static PTQ and QAT, future work could explore **reinforcement learning or Bayesian optimization for quantization search**. These methods could automatically determine per-layer bit-widths (e.g., mixing INT8, INT4, and FP8 layers) to minimize accuracy loss under a strict latency or memory budget.

Beyond Reconstruction. Our framework could also be extended to downstream tasks such as real-time heart rate estimation, blood flow velocity tracking, or ultrasound-based gesture recognition. In these cases, one might pretrain the model on large-scale M-mode data and fine-tune for specific tasks, leveraging the spectral inductive biases we have introduced.

Model-Based + Data-Driven Hybrids. Finally, integrating domain-specific physics priors into the network could further improve interpretability and robustness. For instance, incorporating differentiable Doppler signal models or enforcing temporal periodicity constraints could guide the network to produce physiologically consistent reconstructions even under noisy input conditions.

Taken together, these directions aim to transform our research prototype into a fully deployable system for wearable ultrasound sensing—maintaining high fidelity while meeting strict real-time and energy constraints.

Appendix A

Task Description

Master Thesis at the Department of Information Technology and Electrical Engineering

Summer Semester 2025

Christos Dimopoulos

Pretraining and Quantization Techniques for Ultrasound Data Compression

March 22, 2025

Advisors:	Dr. Christoph Leitner, ETZ J69.2, christoph.leitner@iis.ee.ethz.ch Dr. Yawei Li, ETZ J78, yawli@iis.ee.ethz.ch Marco Giordano, ETZ D 97.5, marco.giordano@pbl.ee.ethz.ch
Supervisor:	Prof. Dr. Luca Benini, lbenini@iis.ee.ethz.ch

Handout:	March, 2025
Due:	see mystudies

1. Project Outline

Ultrasound is a key technology in healthcare [1], and it is being explored for other non-medical tasks, like non-invasive, wearable, continuous monitoring of vital signs [2] and gesture recognition [3]. However, its widespread adoption in this scenario is still hindered by the size, complexity, and power consumption of current devices. Moreover, new flexible transducer technology assures good human conformability but raises new challenges in signal conditioning and acquisition [4].

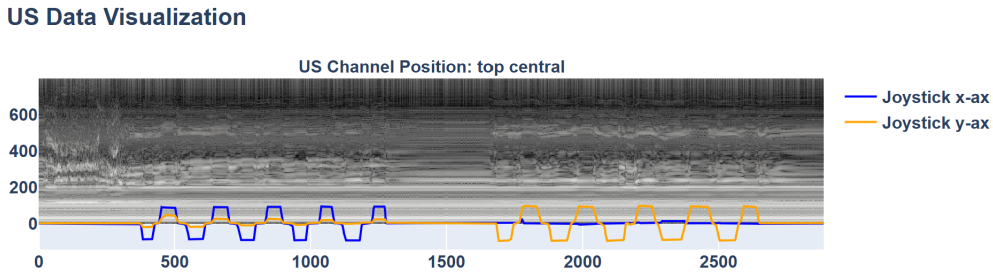


Figure 1: An example of M-mode data displaying joystick movements.

This project aims to advance the development of wearable ultrasound technology, specifically adapting it for wrist-worn applications. The primary objective is to develop a machine learning model and its pretrainign strategy for data compression tailored to novel wearable ultrasound hardware. The model should effectively balance the trade-offs between being lightweight and computationally efficient while maintaining performance comparable to high-capacity model architectures.

2. Research Questions

The research questions of this thesis are:

- Can we develop a model with a general understanding of ultrasound data and what is the best model architecture?
- Can this model possibly be used for data compression on embedded hardware and how?
- In that context, what are the most effective techniques to reduce the model size and latencies and what affect has this reduction on the model results?

3. Methodology

- **Task I – Project Setup, Computing Infrastructure, and Initial Research:**
  - Develop a detailed project plan and Gantt chart.
  - Document key findings from the literature review, including a comparative grid of model architectures, machine learning techniques for quantization, and data compression strategies.
  - Set up and familiarize with the computing infrastructure, *i.e.* CSCS, CINECA.

  Milestone: Establish a solid understanding of relevant techniques and model architectures (e.g., transformer-based and hybrid models), define the full project scope, and set up the computing environment.
- **Task II – Development of Dataloader, Training, and Evaluation Pipelines:**
  - Implement a robust dataloader.
  - Develop a hybrid model based on transformers [5] and state-space models [6, 7].
  - Set up training and evaluation pipelines.
  - Define loss functions and evaluation metrics.
  - Conduct mockup experiments to verify functionality.

  Milestone: A streamlined training process and consistent evaluation pipeline.
- **Task III – Experimentation and Model Evaluation:** Use the evaluation metrics defined in Task II to quantify performance across models and techniques.
  - Build a baseline model using pretraining strategies.
  - Evaluate various quantization techniques [8], as integer 8/4 bits and floating point 8/4 bits.
  - Assess additional processing and data compression methods.

  Milestone: A comprehensive ablation study grid, identification of the best-performing model, and a rationale for its superior performance.
- **Task IV – Downstream Task Analysis:**
  - Implement and train a simple model for hand gesture recognition (dataset provided).
  - Evaluate performance with and without the pretrained encoder.

  Milestone: Integration of an additional model stage and assessment of its impact.
- **Task V – Final Report and Presentation:**

- Compile and document final results.
- Write a comprehensive report summarizing methodology, experiments, and key findings.
- Present project outcomes (20-minute presentation + discussion).

Milestone: Submission of thesis and accompanying documentation.

4. Project Realization

4.1. Project Plan

Within the first week of the project you will be asked to prepare a project plan. This plan should identify the tasks to be performed during the project and sets deadlines for those tasks. The prepared plan will be a topic of discussion of the first week's meeting between you and your advisers. Note that the project plan should be updated constantly depending on the project's status.

4.2. Meetings

Weekly meetings and reports must be held. The exact time and location of these meetings will be determined within the first week of the project in order to fit the student's and the assistant's schedule. These meetings will be used to evaluate the status and progress of the project. Besides these regular meetings, additional meetings can be organized to address urgent issues as well.

4.3. Report

Documentation is an important and often overlooked aspect of engineering. One final report has to be completed within this project. The common language of engineering is de facto English. Therefore, the final report of the work is preferred to be written in English. Any form of word processing software is allowed for writing the reports, nevertheless, the use of LaTeX with Tgif[1] or any other vector drawing software (for block diagrams) is strongly encouraged by the IIS staff.

Code Repository As many of our projects are heavily code-based, the documentation of the code and its repository is also considered important in the grading of your thesis. We suggest to follow coding standards and style guides when writing code, e.g. C [9], Python [10],...

[1] See: http://bourbon.usc.edu:8001/tgif/index.html and http://www.dz.ee.ethz.ch/en/information/how-to/drawing-schematics.html.

Final Report The final report has to be presented at the end of the project and a digital copy need to be handed in. Note that this task description is part of your report and has to be attached to your final report. Our LaTeX template can be found here for download: https://iis-projects.ee.ethz.ch/index.php?title=Final_Report)

4.4. Presentation

There will be a presentation (15 min for the semester thesis, and 20 min for the MS thesis presentation followed by 5 min Q&A) at the end of this project in order to present your results to a wider audience. The exact date will be determined towards the end of the work. Additional tipps for preparing the presentation can be found here: https://iis-projects.ee.ethz.ch/index.php?title=Final_Presentation

Place and Date Zurich 01.04.2025 Signature Student C. Dimopoulos

References

[1] J. M. Daniels and R. A. Hoppmann, Eds., *Practical Point-of-Care Medical Ultrasound*. Cham: Springer International Publishing, 2016. [Online]. Available: https://link.springer.com/10.1007/978-3-319-22638-5

[2] M. Giordano, K. Keller, F. Greco, L. Benini, M. Magno, and C. Leitner, "Towards a Novel Ultrasound System Based on Low-Frequency Feature Extraction From a Fully-Printed Flexible Transducer," in *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10388792

[3] X. Yang, C. Castellini, D. Farina, and H. Liu, "Ultrasound as a Neurorobotic Interface: A Review," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10436655/

[4] K. Keller, C. Leitner, C. Baumgartner, L. Benini, and F. Greco, "Fully Printed Flexible Ultrasound Transducer for Medical Applications," vol. 8, no. 18, p. 2300577. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/admt.202300577

[5] A. Dimofte, G. A. Bucagu, T. M. Ingolfsson, X. Wang, A. Cossettini, L. Benini, and Y. Li, "Cerebro: Compact encoder for representations of brain oscillations using efficient alternating attention," *arXiv preprint arXiv:2501.10885*, 2025.

[6] A. Tegon, T. M. Ingolfsson, X. Wang, L. Benini, and Y. Li, "Femba: Efficient and scalable eeg analysis with a bidirectional mamba foundation model," *arXiv preprint arXiv:2502.06438*, 2025.

[7] X. Dong, Y. Fu, S. Diao, W. Byeon, Z. Chen, A. S. Mahabaleshwarkar, S.-Y. Liu, M. Van Keirsbilck, M.-H. Chen, Y. Suhara *et al.*, "Hymba: A hybrid-head architecture for small language models," *arXiv preprint arXiv:2411.13676*, 2024.

[8] H. Xi, H. Cai, L. Zhu, Y. Lu, K. Keutzer, J. Chen, and S. Han, "Coat: Compressing optimizer states and activation for memory-efficient fp8 training," *arXiv preprint arXiv:2410.19313*, 2024.

[9] Recommended C Style and Coding Standards. [Online]. Available: https://www.doc.ic.ac.uk/lab/cplus/cstyle.html

[10] PEP 8 – Style Guide for Python Code | peps.python.org. [Online]. Available: https://peps.python.org/pep-0008/

A. IIS Grading Scheme

IIS - Digital Circuits and Systems Group
Semester/Master theses grading help sheet

Project:
Students:

Assistant ____________
Date ____________

Grade Aspects		Grading Modifiers							Grade Suggestion		
		Worse than Expected			GOOD	Better than expected			min	suggested	max
		-50	-25	-10		10	25	50			
Difficulty of project	Not all projects have the same difficulty. Sometimes during the course of the project, additional unexpected problems come up making the project much more challenging than originally intended. In other cases, projects may have to be simplified significantly during the course of the project to enable students to have at least some results	tasks were repeatedly simplified during project, extremely simple	Tasks had to be simplified during project, very simple	Simpler than usual	Well balanced project	More involved than usual	Very difficult		-50		25
Effort	A master thesis is a full time effort (40 hours per week), and a semester thesis is supposed to be a half-time work (20 hours/week). Some students spend significantly more (or less) time on the projects. This part is for their effort regardless of their efficiency and quality of results (which is separately handled)	Student was mostly absent	Did not spend enough time on the project	Spent less time than necessary	Student spent the required time for the project	Spend much more time on the project than expected	Student was constantly working on the project		-50		25
In/Dependence	A certain amount of guidance is expected in every project, and the students are expected to contrbute to the project. This part rates deviations from the expectations, students that needed too much supervision, would not be able to continue without significant help.		had to be constantly told what to do	needed more supervison than expected	Some guidance was given, but student worked independently	Student worked completely independently			-25		10
Interest/Initiative	It is expected that students are engaged with the project, and contribute on their own accord to further the project. In some projects, the students are hard to motivate to work for the project and do very little if at all if they are not explicitly told to do by the supervisors.		showed very little interest in the project	was difficult to motivate for the project	Student was engaged with project and showed initiative	Made major additions to the project from own initiative			-25		10
Methodology	General approach to solving problems. Students are expected to use a systematic approach	Student did not follow any methdology at all	Larger flaws in the approach to the problem	not all aspects of the work was done systematically	The work was done systematically	The appraoch of the student was excellent, much better than what was expected			-50		10
Team Work	How did the student perform while working together with the assistants and other students. Were there major frictions and problems?		major problems working together as a team	problems in working together as a team	Student showed good team working skills	Was exceptionally good in team work			-25		10
Learning Effect	Some students posses less than average skills at the beginning of the project, however theya re able to learn and improve themselves during the project. This is meant to reward these students that improved more than expected during the course of the project.				Student showed development throughout the project	Remarkable development during the project	Extraordinary improvement during the course of the project		0		25
Results	A major part of the evaluation is what results were achieved. Note that, good results do not necessarily mean that the goals of the project were reached. For example it is a very good result if the work conclusively demonstrates that approach A is not suitable for realizing X. The quality of the results should be judged on how good they can be used further. The results are not good, if they are not substantiated, they can not help us in further projects and are unreliable. So the pratical value of the results are also a consideration, a PCB that can not be used due to design flaws is a bad result.	Project did not produce any results that can be used, results are demonstrably inferior to what has been achieved before	Weak results, the results can not be reliably used in other projects	Problems with some of the results, worse than expectations	Project achieved all its goals, and yielded good results	Project achieved much better results than was expected	Very good results	Breakthrough results, the project resulted in major new findings that can used further	-50		50
Documentation	Good documentation is an essential part of the project. It should contain, problem description, state of the artdescrivbing what has been done, and explaining why this is not good, list alternative approaches, justfy the selected approach, explain implementation, describe the measurements, present results, compare to state of the art and provide a conclusion.	Documentation is unusable, missing major parts	Documentation is missing proper explanations for larger parts, it is not easy to follow what has been done	Incomplete/sloppy documentation, missing parts or insufficient explanations	Good Documentation, concepts are explained clearly and can be re-used for future works	Very good documentation, much better than what is expected	Excellent documentation, could be used as an example of how to write a report		-50		25
Presentation	Presentation exposes the project to the others. Good presentations should be rewarded		Very bad presentation, student unable to show what has been done	Problematic presentation, unable to conway the work/theory/results properly	Good Presentation	Excellent presentation			-25		10
Publications	If the results of the projecta are very good and the student project is contributing to the publications this should be rewarded. This should not be dependent on the acceptance of the paper, if the assistants see it good enough to be submitted it should be good enough for a reward				There are no direct plans to use the results in a scientific paper	Student is contributing to the results of a paper	Student is actively participating in the preparation of a scientific paper		0		25
					5.00			**GRADE**	1.50	**5.00**	7.25

Appendix B

Declaration of Originality

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

This is a sample title

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
First	Student
Second	Student

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 01.01.2000	First Student Signature
	Second Student Signature

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Bibliography

[1] A. T. P. Hoskins and K. Martin, *Diagnostic Ultrasound: Physics and Equipment*, 2nd ed. Cambridge: Cambridge University Press, 2010.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proceedings of MICCAI*, 2015, pp. 234–241.

[3] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 6000–6010.

[4] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020, pp. –.

[5] M. AI, "Ai-powered audio compression technique," 2022, meta AI blog post. [Online]. Available: https://ai.meta.com/blog/ai-powered-audio-compression-technique/

[6] J. M. Daniels and R. A. Hoppmann, *Practical Point-of-Care Medical Ultrasound*. Cham: Springer International Publishing, 2016, [Online]. [Online]. Available: https://link.springer.com/10.1007/978-3-319-22638-5

[7] K.-V. Jenderka and S. Delorme, "Diagnostischer ultraschall," in *Diagnostischer Ultraschall*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 285–305, [Online].

[8] A. Maier, S. Steidl, V. Christlein, and J. Hornegger, *Medical Imaging Systems: An Introductory Guide*, ser. Lecture Notes in Computer Science. Cham: Springer Open, 2018, vol. 11111, [Online].

[9] C. Wang *et al.*, "A stretchable ultrasound phased array for deep tissue imaging and therapy," *Nature Biomedical Engineering*, vol. 5, pp. –, 2021.

[10] Y. Lin *et al.*, "A wearable ultrasonic system-on-patch for continuous cardiovascular monitoring," *Nature Biotechnology*, vol. 42, pp. –, 2024.

[11] Z. Wang *et al.*, "Flexible doppler ultrasound device for continuous monitoring of blood flow velocity," *Science Advances*, vol. 7, no. 5, p. eabe1733, 2021.

[12] S. Song *et al.*, "A wireless wearable ultrasound carotid neckband for continuous monitoring," *Applied Sciences*, vol. 9, no. 19, p. 4117, 2019.

[13] Y. Xue *et al.*, "Wearable ultrasound devices: Review and future perspectives," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 71, no. 1, pp. –, 2024.

[14] Y. Zhao *et al.*, "Wearable ultrasound: technologies and applications," *npj Digital Medicine*, vol. 6, p. 126, 2023.

[15] G. Nazari *et al.*, "Ultrasound-based sensing for human-machine interaction: A review," *Sensors*, vol. 21, no. 24, p. , 2021.

[16] S. Engdahl *et al.*, "Sonomyography: Applications and future directions in prosthetics and hci," *Frontiers in Robotics and AI*, vol. 9, p. , 2022.

[17] X. Li *et al.*, "M-mode ultrasound for hand gesture recognition: Compact time–depth sequences for wearable sensing," *Biomedical Signal Processing and Control*, vol. 71, p. 103132, 2022.

[18] R. Gao *et al.*, "Echomyography: A single-transducer ultrasound approach for real-time hand motion tracking," *Nature Communications*, vol. 15, p. , 2024.

[19] A. Author and B. Author, "Quantization-aware training for ultrasound-based gesture recognition on edge devices," in *Proceedings of the IEEE International Conference on Wearable Computing*, 2023, pp. –.

[20] E. Smistad *et al.*, "Medical ultrasound segmentation using deep learning: A survey," *IEEE Transactions on Medical Imaging*, vol. 39, no. 9, pp. 2504–2515, 2020.

[21] V. Nandwana *et al.*, "Deep learning for ultrasound image segmentation: A review," *Ultrasound in Medicine & Biology*, vol. 48, no. 2, pp. –, 2022.

[22] Y. Liu *et al.*, "Deep learning for ultrasound super-resolution," *IEEE Transactions on Medical Imaging*, vol. 39, no. 4, pp. 1314–1324, 2020.

[23] D. Xu *et al.*, "Deep cnns for speckle noise removal in ultrasound imaging," *Medical Image Analysis*, vol. 65, p. 101770, 2020.

[24] D. Hyun *et al.*, "Deep learning for ultrasound beamforming," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 2, pp. 570–581, 2019.

[25] A. Luchies and B. Byram, "Deep neural networks for ultrasound beamforming," *IEEE Transactions on Medical Imaging*, vol. 37, no. 9, pp. 2010–2021, 2018.

[26] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[27] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *International Conference on Computer Vision (ICCV)*, 2021, pp. 10 012–10 022.

[28] J. Chen *et al.*, "Transunet: Transformers make strong encoders for medical image segmentation," in *MICCAI Workshop on Medical Image Learning with Less Labels*, 2021.

[29] Y. Guo *et al.*, "Transformer-based models for ultrasound image classification," *Medical Image Analysis*, vol. , pp. –, 2022.

[30] X. Zhang *et al.*, "Medical transformer: Applying self-attention to medical image analysis," *IEEE Transactions on Medical Imaging*, vol. 42, no. , pp. –, 2023.

[31] N. Kovachki *et al.*, "Neural operator: Learning maps between function spaces," *Nature Machine Intelligence*, vol. 5, pp. 16–27, 2023.

[32] J. Guibas *et al.*, "Adaptive fourier neural operators: Efficient operator learning in high dimensions," in *International Conference on Learning Representations (ICLR)*, 2021.

[33] L. Tran *et al.*, "Fourier neural operators for medical image reconstruction," *IEEE Transactions on Medical Imaging*, vol. 42, no. 7, pp. 1725–1738, 2023.

[34] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning (ICML)*, 2020.

[35] J.-B. Grill *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[36] M. Caron *et al.*, "Emerging properties in self-supervised vision transformers," in *International Conference on Computer Vision (ICCV)*, 2021.

[37] K. Chaitanya *et al.*, "Contrastive learning of global and local features for medical image segmentation," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 12 546–12 558, 2020.

[38] Z. Zhou *et al.*, "Self-supervised learning for medical image analysis: A comparative study," *Medical Image Analysis*, vol. 73, p. 102193, 2021.

[39] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High fidelity neural audio compression," *arXiv preprint arXiv:2210.13438*, 2022.

[40] S. Han *et al.*, "Learning both weights and connections for efficient neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[41] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[42] M. Denil *et al.*, "Predicting parameters in deep learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[43] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713.

[44] P. Micikevicius *et al.*, "Fp8 formats for deep learning," *arXiv preprint arXiv:2209.05433*, 2022.