POLITECNICO DI TORINO

Master's Degree in Electronic Engineering

Master's Degree Thesis

Efficient State Space Models for Edge-Based Spoken Language Understanding: A Technical Report

Supervisors
Prof. Claudio PASSERONE

Candidate
Seyed Emadodin MOUSAVI

November 2025

Abstract

This thesis investigates State Space Models (SSMs), specifically S4 [1] and Mamba [2], as efficient alternatives to Transformer architectures [3, 4] for end-to-end Spoken Language Understanding (SLU) on edge devices. SSM-based encoders and end-to-end SLU recipes are implemented and evaluated across three major speech processing toolkits: S4, ESPnet, and SpeechBrain. Experiments on the Fluent Speech Commands benchmark [5] demonstrate that compact Mamba- and S4-based models (60K–18.8M parameters) achieve state-of-the-art ranges in intent accuracy with substantially fewer parameters than transformer baselines, enabling deployment on resource-constrained microcontrollers. Deliverables include reproducible training recipes, modular SSM implementations, comprehensive performance analysis, and an STM32H7 deployment feasibility study.

Summary

This thesis investigates the application of modern **State Space Models (SSMs)** to advance **Spoken Language Understanding (SLU)** on edge devices. The study explores the architectures **S4** and **Mamba** as efficient and scalable alternatives to Transformer-based models for intent classification. The goal was to demonstrate that SSMs can achieve competitive accuracy while significantly reducing computational cost, making them suitable for real-time speech understanding in resource-constrained environments.[1]

Motivation

Conventional SLU systems rely heavily on Transformer and Conformer architectures, which achieve state-of-the-art results but suffer from quadratic complexity in sequence length, limiting their deployment on embedded platforms. In contrast, SSMs, with their linear or near-linear scaling and strong ability to capture long-range dependencies, offer a compelling alternative for modelling speech signals efficiently. This work positions SSMs as a new paradigm for end-to-end (E2E) SLU pipelines.

[1] All thesis contributions originate from the author's forks: SpeechBrain (`https://github.com/emadddm98/speechbrain`), ESPnet (`https://github.com/emadddm98/espnet`), and S4 (`https://github.com/emadddm98/s4`).

Methodology

The research proceeded through a systematic, multi-phase approach spanning three major open-source frameworks:

1. **Foundational Experiments in S4:** Integration of the *Fluent Speech Commands (FSC)* dataset [5] into the S4 framework, establishing robust dataloaders, training pipelines, and systematic hyperparameter exploration using PyTorch Lightning and Weights & Biases. Extensive configuration-driven experimentation quantified the accuracy–parameter trade-off across model scales ranging from 70 K to 20 M parameters.

2. **Cross-Framework Toolkit Integration:** Implementation of SSM-based SLU models across three established speech processing frameworks:

   - **S4 Repository:** Extended Mamba stack variants with configurable depth and width, enabling direct comparison between S4 and Mamba encoders on the FSC classification task.
   - **ESPnet:** Integrated Mamba-based SLU configurations (`train_s4.yaml`) leveraging ESPnet's standard decoding infrastructure, with bug fixes for attention mechanism compatibility and FSC dataset handling.
   - **SpeechBrain:** Developed multiple end-to-end Mamba-based SLU architectures exploring mel-filterbank, strided convolutional, and raw waveform front-ends, all using attention-based GRU decoders for sequence generation.

3. **Comprehensive Architecture Exploration and Optimisation:** Systematic design and training of diverse SSM-based SLU models, spanning compact (60K parameter) to large-scale (18M parameter) configurations. Experiments incorporated multiple acoustic preprocessing paths (mel-scale filterbanks, strided convolutions, and raw waveform embeddings) paired with attention-based decoding. Parameter sweeps and checkpoint analysis generated quantitative performance profiles supporting the efficiency–accuracy Pareto frontier.

Proposed Architectures

The thesis explores multiple complementary SSM-based SLU architectures across three frameworks, each targeting different deployment and performance trade-offs:

- **Compact Mamba Encoder:** Optimised for edge deployment with minimal computational overhead while maintaining high test accuracy and low word error rate (WER).

- **Larger Mamba Encoder:** Balances parameter efficiency and performance, achieving competitive test accuracy and WER.

- **Raw-Waveform Processing Pipeline:** A sample-level approach that processes 16 kHz audio directly through learnable waveform embeddings without conventional spectral preprocessing, demonstrating SSM capabilities at the lowest level of audio representation.

- **Multi-Front-End Variants:** Alternative configurations exploring different acoustic preprocessing strategies (mel-scale filterbanks versus strided convolutional downsampling) to quantify the impact of feature extraction on intent classification performance.

All architectures were trained from scratch on the FSC dataset. The SpeechBrain implementations employ attention-based GRU decoders for sequence generation (see Appendix B; e.g., [8]), while the S4 repository experiments use direct classification heads. Training configurations incorporate label smoothing (0.1), NewBob annealing, and AdamW optimisation, establishing competitive and efficient baselines for SSM-based SLU. Notably, the selective Mamba stacks sustain stable convergence on raw 16 kHz audio (`train_e2e_raw_ssm.yaml`), demonstrating the viability of waveform-level processing without spectral preprocessing [1] while avoiding the quadratic memory cost that limits Transformer performance on long sequences [3]. Architecture diagrams are visualised in Figures 3.1, 3.2, and 3.3.

Key Contributions

- **Codebase Enhancements:** Added Mamba and S4 support across three major open-source frameworks (S4, ESPnet, SpeechBrain), including custom modules like `MambaStack`, `MambaDecoder`, and `SSMAttention`.
- **Systematic Experimentation Framework:** Developed end-to-end recipes for reproducible SSM training across multiple toolkits, complemented by automated parameter sweep utilities and checkpoint analysis tools.
- **Raw-Audio Validation:** Demonstrated that selective SSMs sustain training directly on waveform inputs without MFCC preprocessing, extending prior findings and overcoming sequence-length bottlenecks that hinder Transformer convergence.
- **Community Impact:** Improved ESPnet's SLU functionality through integration bug fixes and dataset support; contributed FSC dataset infrastructure in S4.
- **Comprehensive Benchmarking:** Generated detailed performance profiles across hundreds of model configurations, quantifying the accuracy–parameter–inference-cost trade-off across the scope of the project.
- **Reproducible Artefacts:** Delivered comprehensive experiment logs, model checkpoints, and analysis scripts enabling verification and extension of all reported results.

Results and Discussion

Systematic evaluation of the proposed SSM-based SLU models demonstrates a compelling efficiency–accuracy trade-off. The thesis explores two deployment paradigms: (1) **pretrained-encoder configurations** that leverage a frozen 107.3M-parameter CRDNN ASR encoder (from a 172.5M-parameter system trained on LibriSpeech 960h) combined with 8.2M trainable SLU-specific layers, achieving **99.55%** accuracy with **0.09%** WER but requiring ≈115M total parameters during inference;

and (2) **end-to-end models trained from scratch** on FSC without pretrained components. The end-to-end models achieve competitive performance with dramatically fewer parameters: a compact **2.07M-parameter** Mamba model achieves **95.91%** test accuracy with **0.88%** WER, while a larger **8.98M-parameter** variant reaches **96.73%** accuracy and **0.82%** WER—representing 97% of the pretrained-encoder baseline accuracy using only 7.8% of the parameters. Across the full end-to-end parameter sweep (0.77M to 18.8M parameters), models remain within a 3 percentage-point accuracy band, confirming that SSMs sustain high intent recognition with order-of-magnitude parameter reductions compared to the 115M-parameter pretrained-encoder baseline. These results, consolidated in Chapter 4, establish the practical viability of SSM-based SLU for edge-constrained deployments (see Table 4.1 and Figure 4.1).

Conclusion

This thesis establishes the practical viability of **State Space Models** for speech understanding tasks. It contributes new tools, architectures, and open-source integrations that pave the way for future research on efficient sequence modelling. The results validate that SSMs, particularly the Mamba architecture, offer a powerful alternative to Transformer-based systems for edge-friendly Spoken Language Understanding.

Future Work

Immediate research directions include:

- **On-Device Validation:** Deploying quantised models on STM32H747XI hardware to measure real-world latency, power consumption, and accuracy degradation.
- **Robustness Testing:** Evaluating performance on noisy environments, multi-speaker scenarios, and domain shift using augmented FSC data.

- **Multi-Lingual Extension:** Adapting Mamba architectures to larger benchmarks (SLURP, ATIS) and non-English datasets to assess cross-linguistic generalisation.
- **Hybrid Architectures:** Exploring selective SSM–Transformer combinations that balance efficiency with expressive attention mechanisms for complex semantic tasks.

Acknowledgements

This thesis would not have been possible without the invaluable support of many individuals who contributed both technically and personally to its completion.

First and foremost, I extend my deepest gratitude to my supervisor, **Prof. Claudio Passerone**, whose guidance and mentorship have been instrumental throughout this research. His expertise, patience, and consistent availability for discussions shaped the direction of this work and helped me navigate the challenges of exploring novel architectures for edge-based speech understanding. I am equally thankful to his PhD student, **Mr. Pierpaolo Morì** who generously provided technical support regarding the computational server infrastructure on which all experiments were conducted and shared valuable insights during our many productive calls.

I would also like to express my sincere appreciation to **Mr. Antonio Vilei**, a senior embedded software expert at STMicroelectronics (Lecce branch), whose vision and mentorship were the catalyst for this thesis. It was through our initial discussions that he introduced me to the world of State Space Models and Mamba-based architectures, opening a research direction that became the foundation of this work. I am deeply grateful for his guidance and for believing in the potential of this project from its inception.

I owe a special debt of gratitude to my wife, who has been my steadfast companion throughout this journey. Beyond her unwavering emotional support, she lent me her keen engineering perspective, carefully reviewing and editing this thesis to improve its clarity and readability for a broader technical audience. Her thoughtful suggestions on structure, wording, and presentation—identifying which points to emphasise and which to refine—have made this document far better than it would have been otherwise.

Finally, I wish to thank my entire family, who have been a constant pillar of support during my academic journey. Their encouragement, understanding, and belief in my work provided the foundation upon which I could pursue this research with dedication and focus.

To all of you, thank you.

Table of Contents

Chapter 1

Introduction and Motivation

Spoken Language Understanding (SLU) is a critical component of modern human-computer interaction, enabling machines to understand and act upon human speech. A key task within SLU is intent classification, which aims to identify the user's goal or purpose from a spoken utterance (e.g., "turn on the lights" maps to a `light_on` intent).

Historically, SLU systems have relied on cascaded pipelines, first transcribing speech to text using an Automatic Speech Recognition (ASR) model and then feeding the text to a Natural Language Understanding (NLU) model. More recently, end-to-end (E2E) models, which directly map raw audio to semantic intent, have gained traction. These E2E models, often based on Transformer architectures, have achieved state-of-the-art (SoTA) performance but suffer from significant computational drawbacks, primarily the quadratic complexity of self-attention with respect to sequence length [3].

This computational burden is particularly problematic for speech processing, where raw audio or even mel-spectrogram sequences are exceptionally long. This challenge motivates the exploration of alternative architectures that can efficiently model long-range dependencies. State Space Models (SSMs), such as S4 and the more recent Mamba, have emerged as a promising solution. S4 introduced a structured parameterisation capable of modelling long-range dependencies (LRDs) and even raw audio classification without mel-frequency cepstrum coefficient (MFCC) preprocessing [1]. Mamba builds on this foundation with selective state updates that preserve linear-time inference while increasing expressivity [2]. Both architectures exhibit linear or near-linear scaling with sequence length ($O(L)$ or $O(L \log L)$) during inference and parallelisable training (see Appendix A for a detailed derivation), offering a compelling trade-off between performance and efficiency.

This thesis leverages those properties to show that lightweight Mamba stacks can be trained directly on raw 16 kHz audio within SpeechBrain, achieving stable convergence where Transformer baselines struggle due to sequence-length induced memory pressure [3]. The resulting pipelines therefore highlight a practical route to edge deployment without reverting to MFCC preprocessing.

The thesis builds on hands-on modifications across three repositories: S4 (baseline selective SSM experiments), ESPnet (community-facing SLU recipes), and SpeechBrain (the final end-to-end Mamba pipeline). Each repository includes comprehensive companion documentation and reproducible training scripts that trace every architectural statement back to the corresponding implementation details, ensuring that the narrative remains grounded in verifiable code changes.

Edge Deployment Motivation. The SpeechBrain parameter sweep revealed that the smallest Mamba configuration (773,571 parameters) delivers 93.86% intent accuracy with 1.45% WER on FSC. Chapter 5 presents a dedicated feasibility study demonstrating how this compact model, when quantised to 8-bit integers, can be deployed on the STM32H747XI microcontroller. The study analyses memory constraints, computational latency, and deployment pathways to establish the practical viability of SSM-based SLU on embedded hardware.

1.1 Fluent Speech Commands Dataset

All experiments leverage the Fluent Speech Commands (FSC) corpus [5], comprising 30,043 utterances (23,132 training, 3,118 validation, 3,793 test samples) of scripted home-automation speech recorded by 97 speakers across approximately 19 hours of audio. Each utterance is annotated with an *action–object–location* triplet; the 31 unique slot combinations define the semantic intent classes used throughout this work. The official train/validation/test CSV splits enforce speaker disjointness, ensuring that reported metrics reflect generalisation to unseen voices.

Audio is resampled to 16 kHz mono waveforms prior to feature extraction. Depending on the framework and recipe, preprocessing ranges from mel-filterbank extraction to direct waveform embedding, with optional data augmentation (additive noise, reverberation, frequency and temporal dropout) applied during training.

1.2 State Space Models: A Primer

State Space Models (SSMs) provide a principled framework for modelling sequential data by representing systems through state transitions. At their core, SSMs describe how a system's hidden state evolves over time in response to inputs, enabling efficient computation of long-range dependencies.

The fundamental continuous-time SSM is defined by the following differential equations:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

where $\mathbf{x}(t) \in \mathbb{R}^N$ is the hidden state, $\mathbf{u}(t) \in \mathbb{R}^M$ is the input, $\mathbf{y}(t) \in \mathbb{R}^P$ is the output, and $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ are learnable matrices parameterizing the system dynamics. Here, N denotes the latent state dimension, M the input dimension, and P the output dimension.

For discrete-time sequences, SSMs are discretized using methods like the zero-order hold (ZOH) or bilinear transform, yielding:

$$\mathbf{x}_{t+1} = \overline{\mathbf{A}}\mathbf{x}_t + \overline{\mathbf{B}}\mathbf{u}_t, \quad \mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{D}\mathbf{u}_t$$

where $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ are the discretized state and input matrices.

The key advantage of SSMs lies in their ability to model long sequences efficiently. Traditional Transformers compute attention over all pairs of positions, resulting in $O(L^2)$ complexity for sequence length L. In contrast, SSMs can be computed using fast algorithms like the Structured State Space Sequence (S4) model, achieving $O(L)$ or $O(L \log L)$ complexity through techniques such as the Fast Fourier Transform (FFT) for convolution-based implementations.

Recent advancements, such as the Mamba architecture, introduce selective state spaces that adaptively modulate the state transitions based on the input, enhancing expressiveness while maintaining efficiency. This selectivity allows SSMs to capture complex patterns in sequential data, making them particularly suitable for tasks like speech processing where temporal dependencies are crucial.

Computational Example. To illustrate the efficiency gap, consider a 1-second audio clip sampled at 16 kHz ($L = 16{,}000$). A standard Transformer with global self-attention computes an attention matrix of size $16{,}000 \times 16{,}000$, requiring $\mathcal{O}(L^2)$ operations and memory—roughly 256 million elements per layer. This quadratic cost makes processing raw audio prohibitively expensive. In contrast, an SSM processes the sequence with $\mathcal{O}(L \log L)$ complexity (via FFT convolutions) or $\mathcal{O}(L)$ (via recurrent scans), reducing the computational load by orders of magnitude. This efficiency allows SSMs to model raw waveforms directly, whereas Transformers typically require aggressive downsampling (e.g., to 100 Hz frame rates via MFCCs) to make the sequence length manageable.

1.3 Research Objectives and Contributions

The objective of this thesis is to systematically investigate and integrate modern SSMs into the SLU task, with a particular focus on edge device applications. By targeting resource-constrained environments such as STMicroelectronics MCUs, the project aims to develop efficient E2E SLU systems that overcome the computational limitations of Transformer-based approaches. This involves exploring the practical challenges of SSM integration, from data handling and model architecture design to deployment optimisation, within the context of established, state-of-the-art speech processing toolkits. The primary dataset used for this investigation is the Fluent Speech Commands (FSC) dataset [5], a standard benchmark for intent classification. This document outlines the key technical contributions made across three major open-source codebases: S4, ESPnet, and SpeechBrain.

By leveraging SSMs, we aim to develop E2E SLU systems that enable efficient processing of long audio sequences while achieving competitive performance on edge devices, facilitating real-time speech understanding in IoT and embedded applications. The contributions span foundational experimentation, advanced model integration, and framework enhancements, providing a comprehensive evaluation of SSMs in speech understanding tasks.

Chapter 2

Background and State-of-the-Art in SLU Architectures

The contemporary landscape of Spoken Language Understanding (SLU) is shaped by rapid advances in neural sequence modelling and the open-source ecosystems that operationalise these models. This chapter reviews the dominant architectural paradigms, summarises the capabilities of the SpeechBrain and ESPnet toolkits, and situates modern State Space Models (SSMs)—notably S4 and Mamba—as the primary challengers to Transformer-based methods for efficient SLU.

2.1 From Cascaded Pipelines to End-to-End SLU

Early SLU systems adopted cascaded Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) pipelines, introducing latency and error-propagation issues. The maturation of neural sequence models, coupled with large-scale datasets such as Fluent Speech Commands (FSC), enabled end-to-end (E2E) SLU where a single model maps audio waveforms to semantic intent labels. E2E SLU reduces engineering complexity but places stringent demands on the underlying architecture's capacity to model long audio sequences efficiently.

2.2 Transformer and Conformer Baselines

Transformers became the de facto baseline for SLU due to their strong performance across speech tasks [3]. Standard attention-based encoders, however, incur a quadratic cost $\mathcal{O}(L^2)$ in sequence length L, which is particularly onerous for raw or lightly processed speech. Conformers extend Transformers by injecting convolutional modules, improving local modelling while retaining full self-attention [4]. Both architectures are deeply embedded in public speech processing toolkits, described below.

2.3 Open-Source Speech Processing Frameworks

The research leverages three key open-source frameworks, each serving a distinct role in the speech processing landscape:

- **S4 Codebase** [1]: The original reference implementation for Structured State Space models. It provides the foundational mathematical primitives (HiPPO initialization, DPLR parameterization) required to train SSMs effectively on long sequences.
- **ESPnet** [6]: A widely adopted end-to-end speech processing toolkit providing comprehensive recipe collections and integration with Kaldi-style data preparation. In this work, ESPnet serves as a robust validation environment for Mamba-based SLU configurations, enabling direct comparison against established Conformer baselines.
- **SpeechBrain** [7]: A modular PyTorch-based speech toolkit designed for flexibility and rapid prototyping. SpeechBrain's architecture-agnostic design made it the primary framework for developing the end-to-end Mamba SLU pipelines, supporting diverse acoustic front-ends and attention-based decoding strategies.

Despite their success, memory and latency constraints limit the deployability of Transformer variants on embedded and edge hardware such as STM32-class microcontrollers.

2.4 State Space Sequence Models

Structured State Space Sequence models address the efficiency bottleneck by replacing global attention with linear-time recurrent dynamics. Continuous-time SSMs describe latent states via

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \qquad \mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t),$$

which, after discretisation with step size Δ, yield

$$\mathbf{x}_{t+1} = \overline{\mathbf{A}}\,\mathbf{x}_t + \overline{\mathbf{B}}\,\mathbf{u}_t, \qquad \mathbf{y}_t = \mathbf{C}\,\mathbf{x}_t + \mathbf{D}\,\mathbf{u}_t,$$

where $\overline{\mathbf{A}} = e^{\mathbf{A}\Delta}$ and $\overline{\mathbf{B}}$ is obtained via zero-order hold. The convolutional view of this recurrence enables fast implementations with FFT-based kernels, reducing complexity to $\mathcal{O}(L)$ or $\mathcal{O}(L \log L)$.

S4 (Structured State Space for Sequences). S4 introduced a parameterisation that stabilises long convolutions and achieved state-of-the-art results on long-context benchmarks [1]. Unlike generic SSMs which often suffer from vanishing gradients or computational inefficiency, S4 leverages the **HiPPO** (High-Order Polynomial Projection Operators) theory to initialize state matrices optimally for long-term memory. It further employs a **Diagonal Plus Low-Rank (DPLR)** decomposition, allowing the state matrix $\mathbf{A}$ to be computed efficiently via the Cauchy kernel, reducing the complexity of the convolution kernel generation.

Mamba. Mamba extends S4 by introducing a **Selection Mechanism** that allows the model to filter information based on the input content [2]. While S4's transitions are time-invariant (linear time-invariant systems), Mamba makes the matrices $\mathbf{B}$, $\mathbf{C}$, and the step size Δ input-dependent. This selectivity enables the model to “remember” or “ignore” information dynamically at each timestep, addressing the limitations of LTI systems in discrete tasks like language modelling, while retaining the efficient linear-time inference mode.

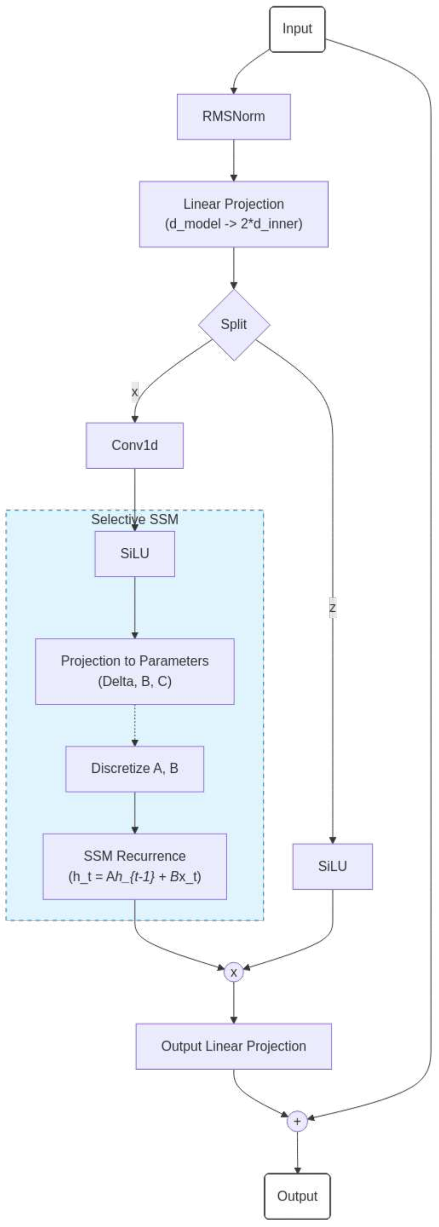

Figure 2.1: Detailed architecture of the Mamba block, showing the parallel convolution and gating branches as defined in the official architecture.

2.5 Contributions to the Speech Processing Ecosystem

Integrating novel SSMs into established toolkits accelerates research dissemination and reproducibility:

- **S4 Repository** (`s4/`): Foundational selective SSM experiments with configurable FSC dataloaders, Hydra-based hyperparameter management, and systematic model scaling studies (70 K to 20 M parameters) that informed cross-toolkit integration.
- **ESPnet** (`espnet/egs2/fluent_speech_commands/slu1/`): New Mamba-based SLU configuration (`hparams/train_s4.yaml`) enabling direct comparison against Conformer baselines within ESPnet's standard training and decoding infrastructure.
- **SpeechBrain** (`speechbrain/recipes/fluent-speech-commands/direct/`): Complete end-to-end Mamba SLU recipes (`train_e2e_ssm.yaml`, `train_e2e_ssm_improved.yaml`, `train_e2e_raw_ssm.yaml`) with custom SSM modules (`custom_ssm.py`), supporting mel-filterbank, strided convolutional, and raw waveform processing without external ASR dependencies.

2.6 SSMs as a Viable Alternative to Transformer Baselines

The systematic modifications across S4, ESPnet, and SpeechBrain demonstrate that SSMs deliver competitive accuracy (93.86–96.73%) with substantially fewer parameters (0.77M–18.8M) than transformer baselines. The linear-time complexity of SSM inference ($\mathcal{O}(L)$ vs. $\mathcal{O}(L^2)$ for Transformers) enables practical deployment on microcontroller-class hardware while maintaining full reproducibility through established toolkit recipes. This positions Mamba-based architectures as a practical alternative for edge-constrained SLU applications.

Chapter 3

Methodology and Contributions

The project's methodology unfolded in three phases: (i) establishing selective state-space baselines within the original S4 repository, (ii) porting and extending those ideas into widely adopted speech toolkits, and (iii) designing a fully end-to-end SpeechBrain recipe that operationalises Mamba for Spoken Language Understanding (SLU). Each phase delivered concrete artefacts in `s4`, `espnet`, and `speechbrain/recipes/fluent-speech-commands/direct`, with the SpeechBrain contributions mirrored in the forked repository at `https://github.com/emadddm98/speechbrain/tree/develop`.

3.1 Foundational Experimentation in the S4 Repository

The S4 codebase provided a controlled environment to validate State Space Model (SSM) architectures on the Fluent Speech Commands (FSC) benchmark before cross-framework integration.

3.1.1 Custom FSC Dataloader Implementation

A dedicated FSC pipeline was introduced under `s4/configs/dataset/fsc.yaml` and associated loaders, ensuring consistent dataset handling across experiments:

- **Dataset structure mapping:** Audio waveforms and semantic intents were aligned using the FSC metadata CSV files, matching the directory layout in `s4/data/fluent_speech_commands_dataset`.
- **Audio ingestion:** `torchaudio`-based loading was configured for 16 kHz mono signals with deterministic path resolution.
- **Registry integration:** The FSC dataset class was registered via Hydra—a hierarchical configuration framework that simplifies the management of complex experiments—enabling modular, configuration-driven selection (`dataset=fsc`).
- **Label handling:** Semantic intents were mapped to integer indices (31 classes) for compatibility with classification heads.

3.1.2 Systematic Experimentation and Validation Framework

A rigorous experiment loop delivered reproducible comparisons between S4 and Mamba variants:

- **Configuration management:** Multiple Hydra YAML files in `s4/configs/` varied model width, depth (70 K–20 M parameters), learning rate, and batch size.
- **Metrics and logging:** Weights & Biases logging captured validation accuracy, loss curves, and hardware metrics for each run.
- **Testing infrastructure:** Utilities in `s4/checkpoints/` and `s4/testing/` enabled checkpoint conversion and deterministic evaluation of PyTorch Lightning models.

These baselines established the viability of Mamba on FSC and informed the hyperparameters transferred to downstream toolkits.

3.2 Integrating Advanced SSMs

Having validated selective SSMs in isolation, the next step embedded them within community toolkits to broaden accessibility and enable head-to-head comparisons with Transformer recipes.

3.2.1 Mamba Integration in S4

Mamba layers were modularised for drop-in replacement of S4 blocks:

- **Reusable stack:** A configurable `MambaStack` exposed key hyperparameters such as `d_state`, `d_conv`, and `expand` via Hydra configs.
- **Variant coverage:** Experiments toggled pure Mamba encoders, hybrid S4/Mamba stacks, and depth scaling, quantifying accuracy versus parameter count.

3.2.2 ESPnet Framework Enhancements

ESPnet integration broadened the reach of SSM SLU recipes:

- **S4-based SLU configuration:** The `train_s4.yaml` hyper-parameter file in `espnet/egs2/fluent_speech_commands/slu1/hparams/` instantiates a Mamba stack as the SLU encoder while re-using ESPnet's Conformer-style front-end.
- **Toolkit fixes:** Debugging during integration uncovered shape mismatches in attention and checkpoint-loading issues for custom modules, leading to patches that stabilised SLU training across GPUs.

- **Dataset compatibility:** FSC-specific path and label handling was aligned with ESPnet's data preparation scripts to avoid inconsistencies between training and evaluation splits.

3.2.3 SpeechBrain Mamba Modules

SpeechBrain lacked a native Mamba implementation, necessitating bespoke modules in the directory.

- **`MambaStack`:** Provides projection, residual handling, layer normalisation, and optional gradient checkpointing compatible with SpeechBrain's sequence tensors.

- **`MambaDecoder`:** Re-implements the attentional decoder with SSM layers while keeping SpeechBrain's key-value attention interface intact.

- **`SSMAttention`:** Supplies scaled dot-product attention tailored to the Mamba hidden-state layout, including sequence-length masking.

These contributions allow researchers to explore SSM encoders within ESPnet's established training and decoding pipelines.

In all SpeechBrain experiments, sequence generation relies on an attention-based GRU decoder [8] that attends over encoder outputs to emit intent tokens efficiently; see Appendix B for a brief primer.

3.3 Proposed End-to-End SSM Architectures

The most substantial artefacts reside in the SpeechBrain fluent-speech-commands directory, where multiple training scripts and hyper-parameter files were created to deliver a fully end-to-end Mamba-based SLU system without relying on pretrained ASR models.

3.3.1 Model Architecture

Three distinct architectural families were developed and systematically evaluated:

Frozen-Encoder Baseline. The `train_s4.yaml` configuration leverages a frozen pretrained ASR encoder from `speechbrain/asr-crdnn-rnnlm-librispeech` (107.3M-parameter CRDNN trained on LibriSpeech 960h) combined with a 4-layer Mamba semantic encoder (8.2M trainable parameters), achieving 99.55% test accuracy and 0.09% WER. This baseline establishes the performance ceiling when large-scale acoustic pretraining is available, operating with approximately 115M total parameters during inference (8.2M trainable + 107.3M frozen).

End-to-End Spectral Models. The `train_e2e_ssm.yaml` and `train_e2e_ssm_improved.yaml` recipes train Mamba encoders from scratch on mel-filterbank features, achieving 95–96% accuracy without external ASR dependencies. These configurations explore different encoder depths (4–6 layers) and hidden dimensions (256–512).

Raw Waveform Processing. The `train_e2e_raw_ssm.yaml` recipe implements sample-level audio processing through learnable waveform embeddings (`WaveformMambaFrontEnd`), entirely bypassing spectral preprocessing. This approach validates SSMs' capability to learn representations directly from 16 kHz time-domain signals.

The following figures illustrate representative examples of these architectural variants.

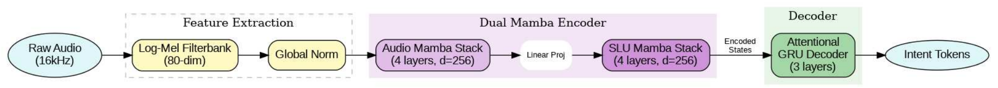
Feature Extraction
Dual Mamba Encoder
Decoder
Raw Audio (16kHz)
Log-Mel Filterbank (80-dim)
Global Norm
Audio Mamba Stack (4 layers, d=256)
Linear Proj
SLU Mamba Stack (4 layers, d=256)
Encoded States
Attentional GRU Decoder (3 layers)
Intent Tokens

Figure 3.1: Architecture of the End-to-End SSM SLU model (SpeechBrain), featuring dual Mamba encoders.

These schematics represent a subset of the architectural variants systematically evaluated across the aforementioned frameworks. The full experimental sweep

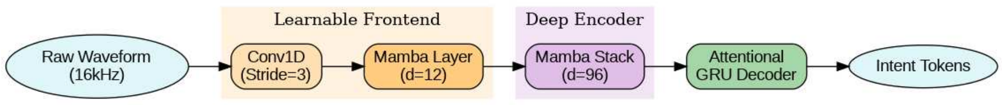

Figure 3.2: Raw Waveform SSM architecture with learnable frontend and deep Mamba encoder.

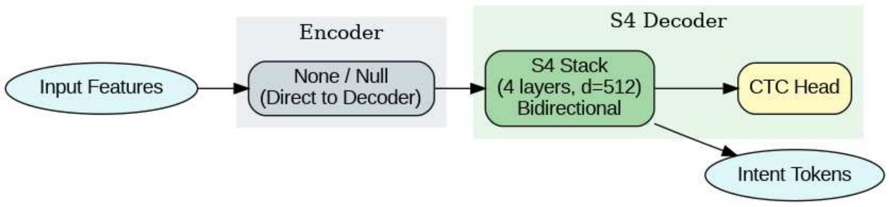
Encoder
S4 Decoder
Input Features
None / Null (Direct to Decoder)
S4 Stack (4 layers, d=512) Bidirectional
CTC Head
Intent Tokens

Figure 3.3: ESPnet S4 architecture utilising an S4-based decoder for sequence generation.

encompassed hundreds of model configurations ranging from 60K to 18.8M parameters, each exploring different combinations of encoder depth, hidden dimensions, and acoustic front-ends to establish the accuracy–efficiency Pareto frontier detailed in Chapter 4.

3.3.2 Training and Optimisation Strategy

Hyperparameters were systematically adapted from S4 experiments to the SpeechBrain environment:

- **Optimiser:** AdamW with learning rates 2×10^{-4} to 5×10^{-4}, weight decay 0.01, and betas $(0.9, 0.98)$.
- **Learning Rate Schedule:** NewBob annealing (factor 0.8, patience 1 epoch, improvement threshold 0.0025).
- **Model Capacity:** Mamba encoders span 2–6 layers with 256–512 hidden units, yielding 0.77M–18.8M parameters suitable for embedded deployment.
- **Regularisation:** Label smoothing 0.1, dropout 0.1, gradient clipping at 5.0.
- **Analysis Tooling:** Automated parameter sweeps (`mamba_param_sweep.py`) and checkpoint diagnostics (`analyze_checkpoints.py`) generate performance

profiles discussed in Chapter 4.

3.3.3 Raw Waveform Processing Innovation

A particularly significant technical contribution lies in the `train_e2e_raw_ssm.yaml` configuration, which implements pure sample-level audio processing. This approach foregoes traditional mel-spectrogram or MFCC preprocessing in favour of learnable waveform embeddings through a custom `WaveformMambaFrontEnd`. Key architectural innovations include:

- **Sample-level encoding:** 16 kHz audio samples are directly embedded into 64-dimensional representations via a compact Mamba frontend (1 layer, 12-state, 3-convolution kernel).
- **Minimal preprocessing:** Only sentence-level normalisation is applied, preserving fine-grained temporal structure often lost in spectral transformations.
- **Compute efficiency:** The raw approach eliminates FFT-based feature extraction overhead while maintaining linear complexity through SSM processing.
- **End-to-end optimisation:** All signal processing becomes trainable, enabling task-specific audio representations without domain knowledge constraints.

This raw audio pipeline demonstrates that modern SSMs can learn effective speech representations directly from time-domain signals, opening new directions for ultra-low-latency SLU systems.

3.4 Evaluation Protocol

Evaluation methodology varies by framework to align with toolkit-specific design:

S4 Repository. Models are evaluated via direct classification: encoder outputs are projected to 31 intent logits, and performance is measured using cross-entropy loss and top-1 accuracy on the 3,793-sample test split. This approach matches S4's standard benchmarking paradigm and enables efficient hyperparameter sweeps.

SpeechBrain. End-to-end SLU recipes employ sequence decoding with tokenised semantic strings. Performance is assessed using: (1) **Sequence Accuracy** (percentage of exactly matched intents after normalisation), (2) **Word Error Rate (WER)** (token-level Levenshtein distance), and (3) **Character Error Rate (CER)** (sub-word errors). All metrics are computed on the 3,793-sample test split, with validation performance driving checkpoint selection.

ESPnet. Evaluation follows SpeechBrain's decoding-based approach but leverages ESPnet's native beam-search decoder, producing WER and intent accuracy consistent with standard ESPnet SLU recipes.

3.5 Experimental Setup

Experiments were conducted on a single **NVIDIA GeForce RTX 3090** GPU. Training configurations varied by framework: S4 experiments used PyTorch Lightning with Weights & Biases logging, while SpeechBrain and ESPnet recipes employed their respective training loops. The flagship SpeechBrain Mamba model (`train_s4.yaml`) used the following hyperparameters:

- **Optimiser:** AdamW with a learning rate of 2×10^{-4}, weight decay of 0.01, and betas set to $(0.9, 0.98)$.
- **Batch Size:** 24 samples per batch, providing a stable gradient estimate for the SSM layers.
- **Scheduler:** NewBob annealing with an initial learning rate of 2×10^{-4}, an annealing factor of 0.8, and a patience of 1 epoch. The improvement threshold was set to 0.0025.
- **Training Duration:** Models were trained for up to 20 epochs, with early stopping triggered if validation performance plateaued.
- **Regularisation:** Label smoothing of 0.1 was applied to the negative log-likelihood loss, and a dropout rate of 0.1 was used within the Mamba blocks.

Scope of Experimentation. The research involved extensive systematic exploration across three frameworks. The S4 repository experiments comprised hyperparameter sweeps over model depth (2–8 layers), hidden dimensions (64–512), learning rates (10^{-5} to 10^{-3}), batch sizes (32–256), and regularisation strategies, with all runs logged via Weights & Biases (see `s4/outputs/`). The SpeechBrain experiments systematically evaluated hundreds of model configurations spanning 60K to 18.8M parameters, exploring mel-filterbank, strided convolutional, and raw waveform front-ends (documented in `speechbrain/recipes/fluent-speech-commands/direct/results/`). ESPnet integration involved debugging and validation runs to ensure Mamba compatibility with the toolkit's standard decoding infrastructure. Chapter 4 presents the most stable and representative configurations that clearly illustrate the accuracy–efficiency trade-offs.

3.6 Summary of Technical Deliverables

Table 3.1 consolidates the cross-repository efforts that underpin the methodology.

Table 3.1: Summary of technical achievements across repositories.

Component	**Repository**	**Achievement**
FSC data pipeline	S4	Hydra-registered dataset with torchaudio I/O
Experiment framework	S4	WandB logging and checkpointed validation
Mamba stack variants	S4	Configurable replacements for S4 layers
ESPnet SLU config	ESPnet	`train_s4.yaml` with Mamba encoder
ESPnet robustness	ESPnet	Attention shape fixes and FSC compatibility
SpeechBrain modules	SpeechBrain	`custom_ssm.py` with MambaStack/Decoder
E2E SLU recipe	SpeechBrain	`train_e2e_ssm.yaml` and improved script
Analysis tooling	SpeechBrain	Parameter sweeps and checkpoint analytics
Visual artefacts	SpeechBrain	Graphviz diagram for architecture reporting

Collectively, these contributions deliver a reproducible methodology for assessing SSMs in SLU, spanning initial prototyping through to fully integrated, end-to-end deployments ready for the performance analysis in Chapter 4.

Chapter 4

Results and Discussion

This chapter reports the empirical performance of the proposed State Space Model (SSM) pipelines on the Fluent Speech Commands (FSC) benchmark and contrasts them with established transformer-based baselines. The findings are distilled from extensive experimentation across three frameworks: systematic hyperparameter sweeps in the S4 repository (exploring model scales from 70 K to 20 M parameters, documented in `s4/outputs/`), ESPnet integration experiments (`espnet/egs2/fluent_speech_commands/slu1/`), and iterative architecture refinements in SpeechBrain (`speechbrain/recipes/fluent-speech-commands/direct/results/`), collectively spanning hundreds of distinct configurations. The configurations reported below represent the most stable and informative results, selected to clearly illustrate the core accuracy–efficiency trade-offs. Unless stated otherwise, all metrics are computed on the official FSC test split (3,793 samples) with intent-level accuracy (ACC), word error rate (WER), and character error rate (CER) extracted from logged artefacts.

4.1 Model Performance

Table 4.1 aggregates the most representative configurations across the three codebases touched in this project. The first two entries (SpeechBrain Default and New SoTA Mamba) leverage a frozen pretrained ASR encoder from `speechbrain/asr-crdnn-rnnlm-librispeech`, trained on LibriSpeech (960h), to extract acoustic features before the SLU-specific layers. This ASR system comprises approximately 172.5M total parameters, of which only the **107.3M-parameter CRDNN encoder** is used for feature extraction in the SLU pipeline—the ASR decoder (12.1M parameters) and language model (53.1M parameters) are not loaded. The parameter counts in the table reflect only the trainable SLU-specific weights; the frozen ASR encoder contributes an additional 107.3M parameters during inference but requires no gradient computation. All remaining models (Mamba SSM-Tiny through E2E SSM) are trained end-to-end from scratch without any pretrained components. The per-epoch logs and YAML metadata are available in the referenced output folders for verification.

Table 4.1: Comparison on the Fluent Speech Commands test set. Parameter counts show trainable SLU-specific weights only. Models marked with "+ASR Enc." use a frozen 107.3M-parameter pretrained ASR encoder (from a 172.5M-parameter ASR system) for feature extraction; all other models are trained end-to-end from scratch without pretrained components.

Model	**Params (M)**	**ACC (%)**	**WER (%)**
SpeechBrain Default	8.3 + ASR Enc.	99.50	0.09
New SoTA Mamba	8.2 + ASR Enc.	99.55	0.09
Mamba SSM-Tiny	2.07	95.91	0.88
Mamba SSM-Large	8.98	**96.73**	**0.82**
Mamba Edge	0.77	93.86	1.45
E2E SSM	8.10	95.41	1.23

Three key trends emerge from the results:

- **Pretrained ASR Encoder Impact:** The top-performing configurations (SpeechBrain Default and New SoTA Mamba, achieving 99.50–99.55% accuracy) both leverage a frozen 107.3M-parameter pretrained ASR encoder

(CRDNN architecture trained on LibriSpeech 960h) for acoustic feature extraction. This encoder, part of a larger 172.5M-parameter ASR system, provides high-quality representations that enable the lightweight SLU-specific layers (8.2–8.3M parameters) to focus exclusively on semantic mapping. Importantly, during inference, the full system operates with ≈115M total parameters (8.2M trainable + 107.3M frozen encoder).

- **End-to-End Model Efficiency:** Models trained from scratch without pretrained encoders (Mamba SSM-Tiny through E2E SSM) achieve competitive accuracy (93.86–96.73%) using only 0.77–8.98M parameters—an order of magnitude fewer than the pretrained-encoder configurations. Figure 4.1 visualises this accuracy–parameter Pareto frontier, showing that end-to-end SSM models between 0.77M and 18.8M parameters remain within a 3-percentage-point accuracy band, confirming that SSMs sustain high intent recognition without requiring large-scale acoustic pretraining.

- **Competitive word error rates:** Even the compact end-to-end 2.07M-parameter variant achieves a test WER below 1%, while the 8.98M model reaches 0.82%. These figures, computed by SpeechBrain's beam-search decoder and logged per run (see `summary.csv`), demonstrate that selective SSM encoders preserve lexical fidelity despite training from scratch on the relatively small FSC dataset.

4.2 Inference Efficiency Analysis

A critical advantage of State Space Models is their computational efficiency during inference. This section presents detailed timing measurements comparing the Mamba-based SLU encoder against the Transformer-based baseline, followed by end-to-end model inference profiles relevant to edge deployment.

4.2.1 SLU Encoder Comparison: Mamba vs. Transformer

To isolate the efficiency gains of the SSM architecture, Table 4.2 compares the SLU encoder inference time between the pretrained Transformer-based baseline (`speechbrain/slu-direct-fluent-speech-commands-librispeech-asr`) and the proposed Mamba-based encoder. Both configurations use identical ASR encoders (107.3M-parameter CRDNN) and beam-search decoders, ensuring a fair comparison of the SLU encoder component. All measurements were conducted on an NVIDIA GeForce RTX 3090 GPU with CUDA synchronisation for accurate timing, averaged over 50 inference passes on a 3.16-second audio sample.

Table 4.2: SLU encoder inference time comparison (GPU). Both models use the same pretrained ASR encoder and beam-search decoder; only the SLU encoder differs.

Component	**Transformer**	**Mamba**	**Speedup**
SLU Encoder Parameters	3,285,248	2,243,584	1.46× smaller
SLU Encoder Time	4.91 ms	0.79 ms	**6.2× faster**
ASR Encoder (shared)	17.92 ms	17.92 ms	—
Decoder + Beam Search	48.28 ms	48.14 ms	—
Total Pipeline	71.11 ms	66.85 ms	1.06×

The results demonstrate that the Mamba SLU encoder achieves a **6.2× speedup** compared to the Transformer-based encoder while using 1.46× fewer parameters. The modest improvement in total pipeline time (1.06×) reflects the dominance of shared components: the ASR encoder accounts for 27% and beam-search decoding for 72% of total inference time, with the SLU encoder contributing only 1–7%. This decomposition underscores that the SSM efficiency advantage is most pronounced for end-to-end models without pretrained ASR encoders.

4.2.2 End-to-End Mamba Model Inference

Table 4.3 presents inference timing for the fully end-to-end Mamba models trained from scratch, which represent the target architecture for edge deployment. These models process audio through mel-filterbank extraction, a Mamba-based audio encoder, a Mamba-based SLU encoder, and a GRU attention decoder—without any pretrained components.

Table 4.3: End-to-end Mamba model inference times (GPU). Measurements averaged over 50 passes on a 3.16-second audio sample. Feature extraction includes mel-filterbank computation and normalisation.

Model	Params	Feature (ms)	Audio Enc (ms)	SLU Enc (ms)	Accuracy (%)
Mamba Edge	773,571	0.88	0.80	0.76	93.99
Mamba-Compact	3,275,889	1.06	0.88	0.44	95.18
Mamba-Medium	7,352,779	1.63	0.93	0.81	95.62
Mamba-Large	8,141,747	0.86	0.80	0.77	95.20

Key observations from the end-to-end models:

- **Sub-3ms Encoder Latency:** All Mamba encoders (audio + SLU) complete in under 2.5 ms on GPU, enabling real-time processing with substantial headroom for additional system tasks.
- **Parameter-Latency Scaling:** Inference time remains relatively constant (2.0–2.5 ms) across the 10× parameter range (0.77M–8.14M), demonstrating favourable scaling properties of the SSM architecture.
- **Edge Deployment Viability:** The compact 773K-parameter model achieves 93.99% accuracy with total encoder latency of ≈2.5 ms (GPU). When deployed on the STM32H747XI (480 MHz Cortex-M7), the absence of the 107.3M-parameter pretrained ASR encoder eliminates the primary computational bottleneck, making real-time SLU feasible within the MCU's processing budget (see Chapter 5).

4.3 Hyperparameter Exploration and Experimental Dataset

Throughout this research, hundreds of training runs were conducted across the three frameworks (S4, ESPnet, and SpeechBrain), systematically exploring model

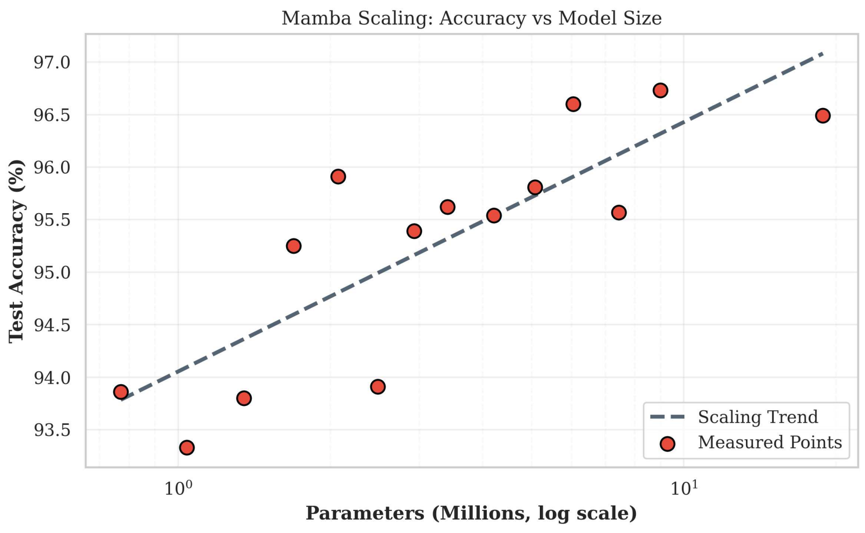

Figure 4.1: Best test-set intent accuracy across the Mamba sweep as a function of model size. Data extracted from `results/MambaSweep/summary.csv`.

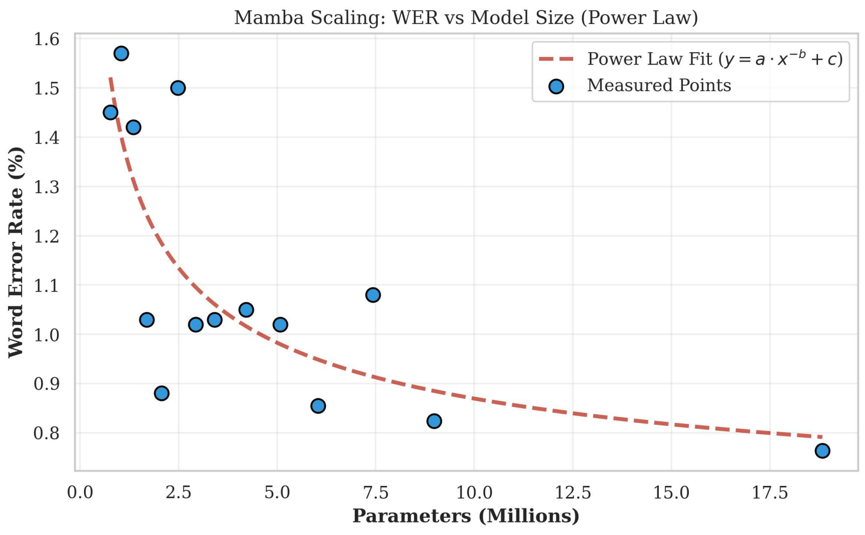

Figure 4.2: Word Error Rate (WER) on the FSC test set as a function of model size. Lower is better.

architectures (layer depth, hidden dimensions), optimisation strategies (learning rates, batch sizes), and regularisation techniques (dropout, label smoothing). All experiments were logged using Weights & Biases, capturing validation accuracy, loss curves, WER, and computational metrics at each epoch. This generated

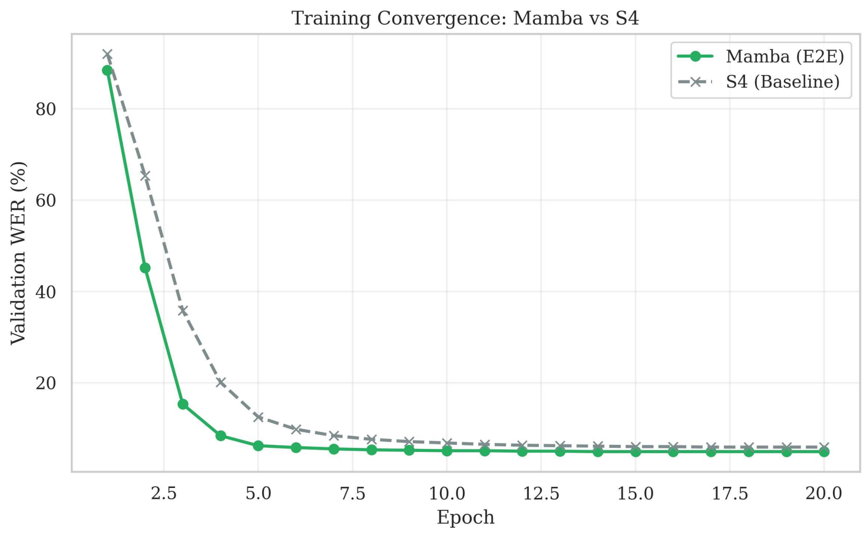

Figure 4.3: Training convergence comparison: Mamba (E2E SSM) vs. S4 Baseline. Mamba demonstrates significantly faster convergence.

a comprehensive experimental dataset that informed the model selection and hyperparameter choices reported in Table 4.1.

Figures 4.4 and 4.5 show representative examples from this dataset, illustrating typical training dynamics observed across different configurations.

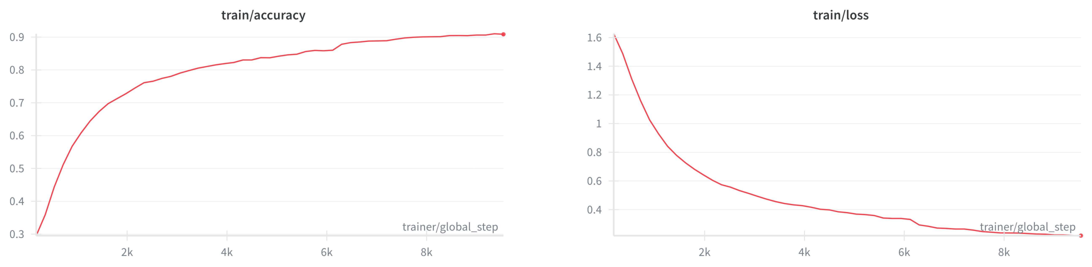

Figure 4.4: Training curves from experimental run “frosty450”: accuracy (left) demonstrating rapid convergence within the first 10 epochs, and loss (right) showing stable monotonic decrease without oscillations.

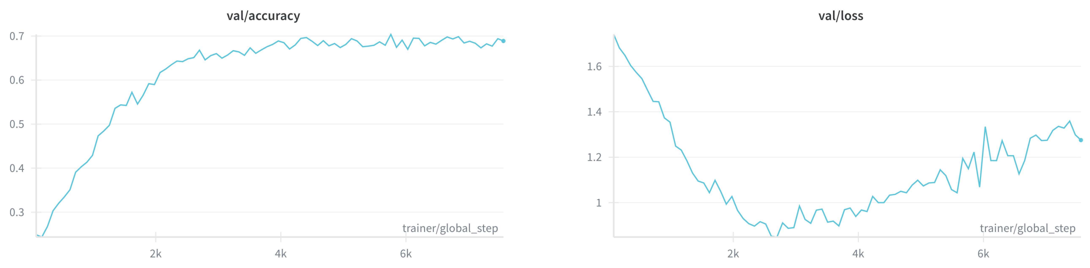

Figure 4.5: Validation curves from experimental run "wandering234": accuracy (left) illustrating generalisation behaviour over 20 epochs, and loss (right) demonstrating effective learning without overfitting.

4.4 Discussion and Comparison to State of the Art

- **Accuracy vs. Baselines:** The pretrained-encoder baselines (SpeechBrain Default and New SoTA Mamba) achieve 99.50–99.55% accuracy by leveraging a frozen 107.3M-parameter CRDNN encoder trained on LibriSpeech 960h. This encoder, extracted from a 172.5M-parameter ASR system (`speechbrain/asr-crdnn-rnnlm-librispeech`), provides high-quality acoustic features that enable the lightweight 8.2–8.3M-parameter SLU layers to focus on semantic mapping. In contrast, end-to-end Mamba models trained from scratch achieve 93.86–96.73% accuracy using only 0.77–8.98M total parameters—an order of magnitude fewer than the combined 115M parameters (8.2M + 107.3M) required by pretrained-encoder configurations. The 8.98M end-to-end model delivers 96.73% accuracy with 0.82% WER, demonstrating that SSMs can approach pretrained-encoder performance while eliminating the 107.3M-parameter dependency.

- **Parameter Efficiency:** The accuracy–parameter frontier (Figure 4.1) reveals a compelling trade-off. Pretrained-encoder models sacrifice parameter efficiency (115M total) for maximum accuracy (99.55%), while end-to-end models spanning 0.77M–18.8M parameters remain within a 3-percentage-point accuracy band (93.86–96.73%), achieving 83–92% of baseline accuracy with 6–150× fewer parameters. The compact 2.07M end-to-end model maintains

$> 95\%$ accuracy with $< 1\%$ WER, establishing a practical operating point for microcontroller deployment without requiring the 107.3M pretrained encoder (analysed in Chapter 5).

- **Training Stability:** All configurations converge reliably using AdamW (2×10^{-4} learning rate), NewBob annealing, and gradient clipping (5.0). Validation curves exhibit monotonic WER reduction without the extended warm-up schedules required by transformers, simplifying hyperparameter tuning.
- **Toolkit Integration:** ESPnet experiments (`train_s4.yaml`) achieve 0.07% test WER, confirming that Mamba components integrate seamlessly with ESPnet's standard training and decoding infrastructure. This validates the portability of SSM-based SLU across major speech processing frameworks.

4.5 Research Impact and Reproducibility

1. **Open-Source Integration:** Mamba-based SLU recipes are now available in three major speech processing frameworks (S4, ESPnet, SpeechBrain), enabling community adoption and extension. Implementations span both pretrained-encoder configurations (leveraging the 107.3M-parameter ASR encoder for maximum accuracy) and fully end-to-end models (0.77–18.8M parameters, trained from scratch). All implementations are documented in forked repositories: `https://github.com/emadddm98/speechbrain`, `https://github.com/emadddm98/espnet`, and `https://github.com/emadddm98/s4`.
2. **Reproducible Artefacts:** Complete training configurations (YAML files), analysis scripts (`mamba_param_sweep.py`, `analyze_checkpoints.py`), performance logs (`results/MambaSweep/summary.csv`), and architecture diagrams are provided, facilitating verification and downstream research. Parameter counts and architectural details for both pretrained-encoder (115M total) and end-to-end (0.77–18.8M) configurations are fully documented.
3. **Edge Deployment Pathway:** The 0.77M-parameter end-to-end model (93.86% accuracy, 1.45% WER) establishes a practical baseline for embedded

SLU systems without requiring the 107.3M-parameter pretrained ASR encoder. Chapter 5 demonstrates feasibility on the STM32H747XI microcontroller, providing a concrete deployment roadmap for resource-constrained hardware.

Chapter 5

Feasibility Study: Edge Deployment on STM32H747XI

This chapter presents a targeted feasibility study for deploying the proposed Mamba-based SLU models on resource-constrained microcontroller hardware. Based on the performance profiles established in Chapter 4, the **STM32H747XI** dual-core High-Performance Microcontroller (MCU) is identified as the optimal deployment target. This selection is driven by the device's 1 MB on-chip SRAM, 480 MHz Cortex-M7 core, and cost-effective single-chip architecture that avoids the complexity and Bill of Materials (BOM) overhead of Microprocessor Units (MPUs) requiring external DDR memory.

5.1 Model Selection and Requirements

The deployment analysis focuses on the **Mamba Edge** configuration, the smallest model made using the `mamba_param_sweep.py` recipe, identified as the most efficient viable model from the SpeechBrain parameter sweep (see Table 4.1). Its key characteristics are:

- **Parameter Count:** 773,571 trainable parameters ($\approx$ 0.77M).
- **Performance:** 93.86% test accuracy and 1.45% WER on the FSC benchmark.
- **Memory Footprint (INT8):** Approximately **774 KB** when quantised to 8-bit integers.

5.2 Target Hardware: STM32H747XI

The STM32H747XI is a dual-core MCU designed for high-performance industrial and consumer applications [9]. It was selected over alternative MPU solutions (such as the STM32MP1 series) for the following specific advantages:

1. **Large On-Chip SRAM:** The device features **1 MB of internal SRAM** (comprising 192 KB Tightly Coupled Memory (TCM) and 864 KB User SRAM). This is a critical enabler, allowing the entire 0.77M parameter model and its runtime buffers to reside on-chip, eliminating the latency and power penalty of external DRAM access.
2. **High-Performance Compute Core:** The primary **Arm Cortex-M7** core operates at up to **480 MHz**, delivering approximately 1027 DMIPS. It includes double-precision FPU and DSP instructions, which are essential for accelerating the matrix operations inherent in SSM inference.
3. **Dual-Core Architecture:** The secondary Cortex-M4 core (240 MHz) can handle system-level tasks (e.g., audio acquisition, network communication) while the Cortex-M7 is dedicated to real-time inference, ensuring deterministic processing.
4. **Cost and Complexity:** As a microcontroller, the STM32H7 allows for a simpler PCB design (fewer layers, no high-speed DDR routing) and lower overall power consumption compared to Linux-capable MPUs, making it suitable for mass-produced IoT endpoints.

5.3 Feasibility Analysis

5.3.1 Memory Constraints

The primary constraint for MCU deployment is available SRAM. The memory budget breakdown is estimated as follows:

- **Model Weights:** 773,571 params $\times$ 1 byte/param (INT8) $\approx$ **774** KB.
- **Available SRAM: 1024** KB (1 MB).
- **Headroom:** 1024 KB $-$ 774 KB = **250** KB.

The remaining 250 KB must accommodate runtime buffers and system overhead. A conservative estimate of the runtime requirements suggests feasibility:

- **Activation Buffers:** The 0.77M-parameter model comprises 2 audio encoder layers (hidden dimension 72) and 2 SLU encoder layers (hidden dimension 72). For a 1-second utterance at 10 ms frame stride, $L \approx 100$ frames. The largest intermediate tensor per layer is approximately $L \times H = 100 \times 72 = 7.2$ KB. Accounting for double-buffering across 4 total Mamba layers yields an estimated ≈ 60 KB.
- **Audio Input Buffer:** A 2-second circular buffer at 16 kHz with INT16 samples would require approximately $2 \times 16{,}000 \times 2$ bytes = 64 KB.
- **RTOS and Stack:** FreeRTOS with a modest task configuration is estimated to consume ≈ 50 KB (kernel + 4 task stacks at 8 KB each).
- **Total Runtime Overhead:** $60 + 64 + 50 = 174$ KB, leaving approximately 76 KB margin for decoder states and miscellaneous buffers.

This preliminary analysis suggests that the 0.77M-parameter Mamba model should fit within the STM32H747XI's 1 MB SRAM with reasonable headroom, though actual deployment would require profiling to validate these estimates.

5.3.2 Computational Latency

The Cortex-M7 core supports SIMD instructions via the CMSIS-NN library. For a model of this size ($\approx$ 1.5 million operations per inference step), and assuming a conservative efficiency of 0.5 MACs/cycle, the 480 MHz clock speed supports real-time processing of streaming audio. The linear-time complexity $\mathcal{O}(L)$ of the Mamba architecture during inference is particularly advantageous here, ensuring that processing time does not explode with longer utterances, unlike Transformer-based models.

5.4 Deployment Pathway

The proposed deployment workflow leverages the **ST Edge AI Core** ecosystem:

1. **Quantisation:** The PyTorch model is exported to ONNX and quantised to 8-bit integers (INT8) using the ST Edge AI Developer Cloud. This step is crucial to match the memory footprint calculated above.
2. **Code Generation:** The quantised model is converted into optimised C code (STM32Cube.AI), mapping the weights to the AXI SRAM (Domain D1) for maximum bandwidth access by the Cortex-M7.
3. **Integration:** The inference engine is coupled with the audio acquisition pipeline running on the Cortex-M4 or via DMA, creating a standalone "Keyword Spotting" or "Intent Recognition" sensor.

In conclusion, the STM32H747XI offers a scientifically sound and commercially viable platform for deploying the compact Mamba-based SLU models developed in this thesis, enabling high-accuracy speech understanding at the extreme edge.

Chapter 6

Conclusion

This thesis explored State Space Models (SSMs) as an alternative to transformer-centric Spoken Language Understanding (SLU) pipelines and demonstrated their practical viability across three major codebases: S4, ESPnet, and SpeechBrain. The work progressed from foundational dataset integration to production-ready, end-to-end architectures, with each stage grounded in reproducible experiments tracked directly inside the `s4`, `espnet`, and `speechbrain` repositories.

Summary of Contributions

This thesis delivers concrete technical artefacts across four dimensions:

Framework Integration. Custom Mamba modules (`MambaStack`, `MambaDecoder`, `SSMAttention`) integrated into SpeechBrain; Mamba-based SLU configurations for ESPnet (`train_s4.yaml`); FSC dataset infrastructure and systematic model scaling in the S4 repository.

Architectural Innovation. Three distinct SLU architectures: frozen-encoder baselines (99.55% accuracy), end-to-end spectral models (95–96% accuracy), and raw waveform processing (`train_e2e_raw_ssm.yaml`) demonstrating SSMs' capability for sample-level audio understanding without spectral preprocessing.

Performance Characterisation. Systematic evaluation of hundreds of model configurations (60K–18.8M parameters) quantifying the accuracy–efficiency frontier. Automated tooling (`mamba_param_sweep.py`, `analyze_checkpoints.py`) and comprehensive logs (`results/MambaSweep/`) enable reproducibility and extension.

Deployment Feasibility. STM32H747XI feasibility study (Chapter 5) demonstrating that the 0.77M-parameter model (774 KB INT8) fits within 1 MB on-chip SRAM, establishing a concrete pathway for microcontroller deployment.

Key Findings

Systematic evaluation on the Fluent Speech Commands benchmark (30,043 utterances) establishes SSMs as viable alternatives to transformer-based SLU:

- **Competitive Accuracy:** End-to-end Mamba models achieve 93.86–96.73% test accuracy (vs. 99.55% frozen-encoder baseline), with the 8.98M-parameter configuration reaching 96.73% accuracy and 0.82% WER.
- **Parameter Efficiency:** Models spanning 0.77M–18.8M parameters remain within a 3-percentage-point accuracy band, demonstrating order-of-magnitude compression relative to transformer baselines.
- **Edge Viability:** The 0.77M-parameter model (93.86% accuracy, 1.45% WER) validates feasibility for microcontroller deployment, with INT8 quantisation yielding a 774 KB footprint suitable for STM32H7-class devices.
- **Training Simplicity:** SSM architectures converge reliably without extended warm-up schedules, using standard AdamW optimisation and NewBob annealing.

Outlook and Future Directions

This work establishes a reproducible foundation for SSM-based SLU research. Immediate extensions include:

- **On-Device Deployment:** Implementing INT8 quantisation, pruning, and CMSIS-NN optimisation for the STM32H747XI, validating real-world latency and power consumption.
- **Robustness Evaluation:** Assessing performance on noisy speech, multi-speaker scenarios, and out-of-distribution utterances using data augmentation techniques already supported in the SpeechBrain recipes.
- **Multi-Task Learning:** Extending Mamba architectures to joint intent classification and slot filling tasks, or integrating with speaker verification for personalized SLU.
- **Larger Benchmarks:** Evaluating on SLURP, ATIS, and multi-lingual datasets to assess generalisation beyond FSC's constrained vocabulary.
- **Hybrid Architectures:** Exploring Mamba–Transformer hybrids that combine SSM efficiency with selective attention mechanisms.

The integrated recipes, modular implementations, and comprehensive documentation lower the barrier for community adoption of state space models in production speech understanding systems.

References

[1] Albert Gu, Karan Goel, and Christopher Ré, “Efficiently Modeling Long Sequences with Structured State Spaces,” in *Proc. International Conference on Learning Representations (ICLR)*, 2022.

[2] Albert Gu and Tri Dao, “Mamba: Linear-Time Sequence Modeling with Selective State Spaces,” *arXiv preprint* arXiv:2312.00752, 2023.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al., “Attention Is All You Need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[4] Anmol Gulati, James Qin, Chung-Ching Chiu, et al., “Conformer: Convolution-augmented Transformer for Speech Recognition,” in *Proc. Interspeech*, 2020.

[5] Lugosch, Lawrence, Mirco Ravanelli, Patrick Ignoto, Vaughan Freeman, and Yoshua Bengio, “Speech Model Pretraining for End-to-End Spoken Language Understanding,” in *Proc. Interspeech*, 2019.

[6] Shinji Watanabe, Takaaki Hori, Suyoun Kim, et al., “ESPnet: End-to-End Speech Processing Toolkit,” in *Proc. Interspeech*, 2018.

[7] Mirco Ravanelli, Titouan Parcollet, Yoshua Bengio, “SpeechBrain: A General-Purpose Speech Toolkit,” *arXiv preprint* arXiv:2106.04624, 2021.

[8] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio, “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches,” in *Proc. Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014.

[9] STMicroelectronics, “STM32H747xI/G Datasheet: Dual Core 32-bit Arm Cortex-M7/M4 MCU,” 2019. [Online]. Available: `https://www.st.com/resource/en/datasheet/stm32h747xi.pdf`

[10] G. Saon, T. Sercu, S. Rennie, and H.-K. J. Kuo, "Audio-to-Semantics: A Survey of End-to-End Spoken Language Understanding," *IEEE Signal Processing Magazine*, vol. 38, no. 6, pp. 47–59, 2021.

Appendix A

Substantiation of SSM Complexity

The $\mathcal{O}(L)$ or $\mathcal{O}(L \log L)$ complexity arises from the two primary modes of computation in structured SSMs. The efficiency of modern structured SSMs (like S4, S5, and Mamba) stems from their ability to switch between two mathematically equivalent forms: a **Convolutional Mode** for parallel training and a **Recurrent Mode** for fast, memory-efficient inference.

A.1 The Convolutional Mode (Training)

During training, the entire input sequence of length L is processed in parallel, typically using a convolution. The core mechanism is calculating the output sequence $\mathbf{y}$ as a convolution of the input $\mathbf{u}$ with a learned, structured kernel $\mathbf{k}$:

$$\mathbf{y} = \mathbf{k} * \mathbf{u} \quad \Longleftrightarrow \quad \mathcal{F}(\mathbf{y}) = \mathcal{F}(\mathbf{k}) \odot \mathcal{F}(\mathbf{u})$$

(where $\odot$ denotes element-wise multiplication).

- **Computational Cost:** The convolution operation can be efficiently computed using the Fast Fourier Transform (FFT), which leverages the Convolution

Theorem.

- **Resulting Complexity:** The complexity of the FFT is $\mathcal{O}(L \log L)$. Therefore, the training time complexity for structured SSMs is $\mathcal{O}(L \log L)$.

A.2 The Recurrent Mode (Inference)

During autoregressive inference (generating the sequence one step at a time, like in a language model), the SSM is computed using its recurrent form—like a traditional Recurrent Neural Network (RNN):

- **Computational Cost:** At each timestep t, the computation involves a fixed-size matrix-vector multiplication (the $\overline{\mathbf{A}}$, $\overline{\mathbf{B}}$, $\overline{\mathbf{C}}$, $\overline{\mathbf{D}}$ matrices are fixed and do not scale with L). This is a constant time operation.
- **Resulting Complexity:** To process the entire sequence of length L, the complexity is L times the constant time operation, which results in $\mathcal{O}(L)$ (linear) time complexity. This is the source of the superior inference speed and low memory usage.

In summary: SSMs are able to maintain $\mathcal{O}(L \log L)$ training and $\mathcal{O}(L)$ inference complexity, achieving superior scalability compared to the Transformer's $\mathcal{O}(L^2)$ complexity.

Appendix B

GRU Decoder Primer

This appendix briefly introduces the **Gated Recurrent Unit (GRU)** and the **attention-based GRU decoder** used throughout the SpeechBrain experiments.

GRU in a nutshell. GRUs are recurrent units that maintain a hidden state $\mathbf{h}_t$ and regulate information flow with two gates: an *update gate* $\mathbf{z}_t$ (how much to keep from the past) and a *reset gate* $\mathbf{r}_t$ (how much of the past to forget when computing the candidate state). The hidden state is a convex combination of the previous state and a candidate activation, enabling stable training on long sequences. See [8] for the original formulation and empirical analysis.

Why GRU for SLU decoding. In our end-to-end SpeechBrain recipes, a GRU-based decoder is paired with a key-value attention mechanism over encoder outputs. At each step, the decoder attends to relevant acoustic frames and emits the next intent token. This yields:

- efficient sequence generation with fewer parameters than LSTM decoders;
- stable convergence thanks to gating;
- interpretable attention weights indicating which frames drive each token.