



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Beyond Attention: Efficient Small Language Models for Domain- Specific Tasks

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE AND ENGINEERING

Author: **Valerio Ceccarelli**

Student ID: 11006753

Advisor: Prof. Matteo Matteucci

Co-advisors: Greta Corti, Paolo Cudrano, Sara Pidò

Academic Year: 2025-26

Abstract

For the past few years, language models have improved mainly by getting bigger, with each generation built with more parameters, more data, and more computation, and served from large data centers. This thesis follows the opposite direction, asking how little capability is lost as a model is made small enough to run on an edge device such as a wearable, a sensor, or a battery-powered board. At this scale memory and energy matter more than raw capability, and the attention mechanism that made the Transformer successful becomes ill-suited to the small accelerators found at the edge. The work therefore adopts Mamba, a state-space architecture whose computational cost grows sub-quadratically with the length of the input. Because a model with few parameters cannot hold broad world knowledge, it has to specialize on a narrow domain instead. To measure depth, four dedicated question-answering benchmarks are built, covering two narrow domains, medicine and finance, each cast into two task formats: a yes/no benchmark and a multiple-choice one. The existing small models BabyLlama, MobileLLM, and a 130M Mamba are adapted to these benchmarks and compared with BabyMamba, a Mamba model of 50 million parameters trained from scratch for this work through pretraining and offline distillation. The results are encouraging: the architecture can be pushed well below the smallest size of the original Mamba study and still be specialized on a narrow domain, remaining competitive with a model of equal size built on a different architecture.

Keywords: Natural Language Processing; Language Models; Small Language Models; Mamba; Edge Computing; Deep Learning; Neural Networks

Sommario

Negli ultimi anni i modelli linguistici sono migliorati principalmente diventando più grandi; ogni generazione è costruita con più parametri, più dati, più potenza di calcolo, ed è eseguita in grandi data center. Questa tesi segue la direzione opposta, chiedendosi quali capacità sopravvivono quando un modello viene ridotto fino a renderlo abbastanza piccolo da poter essere eseguito su un edge device, come un dispositivo indossabile, un sensore o una scheda embedded alimentata a batteria. A questa scala la memoria e l'energia contano più delle capacità pure, e il meccanismo di attention che ha decretato il successo del Transformer si rivela poco adatto ai piccoli acceleratori. Il lavoro adotta quindi Mamba, un'architettura state-space il cui costo computazionale cresce in modo sub-quadratico rispetto alla lunghezza dell'input. Poiché un modello con pochi parametri non può racchiudere un'ampia conoscenza del mondo, deve quindi specializzarsi su un dominio ristretto. Per misurare quanto a fondo un modello riesca a specializzarsi vengono costruiti quattro benchmark di domande e risposte, che coprono due domini ristretti, la medicina e la finanza, ciascuno articolato in due task: un benchmark a risposta sì/no e uno a scelta multipla. I modelli piccoli già esistenti BabyLlama, MobileLLM e Mamba da 130M vengono adattati a questi benchmark e confrontati con BabyMamba, un modello Mamba da 50 milioni di parametri addestrato da zero per questo lavoro tramite pretraining e distillazione. I risultati sono incoraggianti: l'architettura può essere ridotta ben al di sotto della dimensione più piccola dello studio originale su Mamba e può comunque essere specializzata su un dominio ristretto, tenendo testa a un modello di pari dimensione costruito su un'architettura diversa.

Parole chiave: Elaborazione del linguaggio naturale; Modelli linguistici; Piccoli modelli linguistici; Mamba; Edge Computing; Deep Learning; Reti neurali

Contents

Abstract	i
Sommario	iii
Contents	v
Introduction	1
1 Background	3
1.1 Language Models	3
1.1.1 The Transformer	4
1.1.2 From GPT-2 to Frontier Models	5
1.1.3 A Different Frontier	7
1.2 Hardware	8
1.2.1 Accelerating language models	8
1.2.2 Toward small devices	8
1.2.3 Attention limitations	9
1.3 Small Language Models	10
1.3.1 Transformers	10
1.3.2 Alternative Architectures	12
1.4 Benchmarks	14
1.4.1 Commonsense and Knowledge	15
1.4.2 Reasoning and Language Understanding	15
1.5 Language Model Lifecycle	16
1.5.1 Pretraining	16
1.5.2 Distillation	18
1.5.3 Supervised Fine-tuning	19
1.5.4 Evaluation	19

2	Data Collection	21
2.1	Datasets	21
2.1.1	WikiDoc	22
2.1.2	Investopedia	22
2.1.3	Wikipedia	23
2.2	Task Generation	23
2.2.1	Yes/No Benchmark	24
2.2.2	Multiple-choice Benchmark	24
3	Baseline Training	27
3.1	Fine-tuning	27
3.1.1	Methodology	27
3.1.2	Example pipeline	28
3.1.3	Experiments	29
3.2	Continual pretraining	31
3.2.1	Methodology	32
3.2.2	Continual pretraining experiments	32
3.2.3	Fine-tuning after continual pretraining	33
3.3	Results	34
4	BabyMamba	37
4.1	Architecture	37
4.2	Training methods	38
4.2.1	Pretraining	38
4.2.2	Offline distillation	38
4.2.3	Continual Pretraining and Fine-tuning	40
4.3	Experiments	40
4.3.1	Pretraining and Distillation	41
4.3.2	Continual pretraining and Fine-tuning	42
4.4	Results	44
5	Conclusion	45
5.1	Limitations and Future Developments	46
	Bibliography	49
	List of Figures	59

Introduction

For the past few years, language models have improved mainly by getting bigger [1, 2]. Each new generation has been larger than the one before it, built with more parameters, trained on more data, and driven by more computation. This has produced models of extraordinary generality, able to translate, summarize, write code, and answer questions across almost any subject. What it has cost is size: running a model close to the frontier means renting space in a data center, where clusters of specialized accelerators consume hundreds of watts and depend on gigabytes of fast memory [3].

Rather than focusing on further scaling, this thesis studies how far model size can be reduced before losing useful domain-specific behavior. The question it asks is not how much capability emerges when a model is made larger, but how little is lost when it is made smaller, and how far that shrinking can be carried before the model can no longer do anything useful. The target is a model small enough to run not in a data center but on the devices at the edge of the network: a wearable, a sensor, a battery-powered board. The reasons for moving in this direction are practical. A model that runs on the device itself keeps the data it processes local, responds without a round trip to a remote server, and continues to work where connectivity is slow or absent, all at a fraction of the energy and cost of a cloud deployment. These are the situations in which a small, self-contained language model is not a cheaper substitute for a large one but the only thing that works at all.

Shrinking a model to this point is not simply a matter of using fewer parameters. Below a certain size what limits the model is no longer raw capability but memory, energy, and the structure of the hardware itself, and at that point the very mechanism that made the Transformer [4] successful begins to work against it. Self-attention assumes abundant memory and a pattern of computation that the small, power-efficient accelerators found at the edge are not designed to provide. This is what motivates the turn toward alternative architectures, built to capture the same long-range dependencies as attention but at a cost that grows sub-quadratically with the length of the input. Among the available options this thesis adopts Mamba [5], which reaches competitive accuracy while relying on mature and widely supported implementations.

A second idea runs alongside the first. A model with few parameters cannot hold the broad world knowledge of a large one, so it cannot compete on breadth and must instead concentrate everything on a single, well-defined area, where a small model can perform that one task far better than its size alone would suggest. Yet small language models are usually judged on general-purpose benchmarks that reward the wide knowledge and multi-domain reasoning they were never meant to possess. Measured against that standard a specialized model can only look weak, and the quality that actually matters, namely how well it performs inside its own domain, is never measured at all. A fair assessment has to measure the opposite, testing depth in a narrow field rather than breadth across many.

From these two ideas the rest of the work follows. Two narrow domains, medicine and finance, are chosen, and a dedicated question-answering benchmark is built for each, since no existing benchmark measures the kind of focused competence at stake here. The source material is reshaped into two controlled formats, a balanced yes/no benchmark and a multiple-choice one, each with safeguards that keep a model from exploiting the form of a question instead of its content. Against these benchmarks the existing small models, BabyLlama, MobileLLM, and a 130M Mamba, are then adapted and measured, both by direct fine-tuning and by a two-stage procedure in which a phase of continual pretraining on the domain text precedes the fine-tuning, so that a fair point of comparison is in place before anything new is built.

The heart of the thesis is BabyMamba, a Mamba model of 50 million parameters trained from scratch for this work. Its size is chosen deliberately to fall below the smallest model released in the original Mamba study and to match that of BabyLlama, which makes two clean comparisons possible at once: between two architectures of the same size, and between two sizes of the same architecture. BabyMamba is produced both by pretraining from scratch and by offline distillation from a much larger teacher of the same Mamba family, and it is then specialized on the benchmarks with the same two-stage procedure used for the baselines, so that the new model and the existing ones are judged on equal terms.

What ties these strands together is a single question: how far can a capable architecture be compressed before specialization stops working? The evidence gathered here is encouraging. A Mamba model of 50 million parameters, trained from scratch, can be taught a narrow domain and remain competitive with a model of the same size built on a different architecture. The architecture can therefore be pushed well below the smallest size of the original work and still be specialized on a focused task, which is exactly the regime that matters for the constrained hardware where these models are meant to live.

1 | Background

This chapter presents the background needed to position the rest of the thesis. It begins with language models: the next-token prediction objective on which they are based, the Transformer architecture that allowed them to scale, and the progression from GPT-2 to current frontier systems. It then turns to the direction this work pursues, which is scaling models down rather than up.

Because this direction depends less on raw capability than on the hardware a model must run on, the chapter examines the constraints of edge devices and the structural reasons why attention is poorly suited to them. This motivates a survey of small language models, covering both scaled-down Transformers and alternative architectures such as linear attention, Mamba, and xLSTM, which aim for sub-quadratic complexity.

It then describes the benchmarks used to evaluate models at this scale. The chapter closes with the language model lifecycle, from pretraining and distillation to supervised fine-tuning and evaluation, which together provide the methodological vocabulary used throughout the thesis.

1.1. Language Models

A language model defines a probability distribution over sequences of symbols drawn from a fixed, finite vocabulary. The atomic units that make up this vocabulary are called *tokens*, and a raw string of text can be split into a sequence of tokens by a tokenizer using an algorithm such as byte-pair encoding [6]. One of the objectives of language models is next-token prediction, which tries to predict each token from those that precede it by approximating the joint probability distribution of natural language [7].

Training a model to predict the next token over a sufficiently large corpus makes it learn, as a byproduct, the ability to capture grammar, factual knowledge, and a degree of reasoning. Over the past decade the objective itself has remained the same; what has changed is the architecture used to model these probabilities and the scale at which it is trained.

1.1.1. The Transformer

For most of the history of Natural Language Processing (NLP), these probabilities were modeled by recurrent neural networks (RNNs) [8], which process one token at a time while carrying forward a hidden state [9]. Recurrence is well suited to sequential data, but it is inherently serial: the computation at each step depends on the result of the previous one. This makes training slow and weakens the propagation of information across distant positions [10].

The Transformer, introduced by Vaswani et al. [4] in *Attention Is All You Need*, removed recurrence entirely. Instead of passing a state from one step to the next, every token in the sequence gets compared directly to every other token through self-attention [11], forming a weighted combination of all tokens in the sequence.

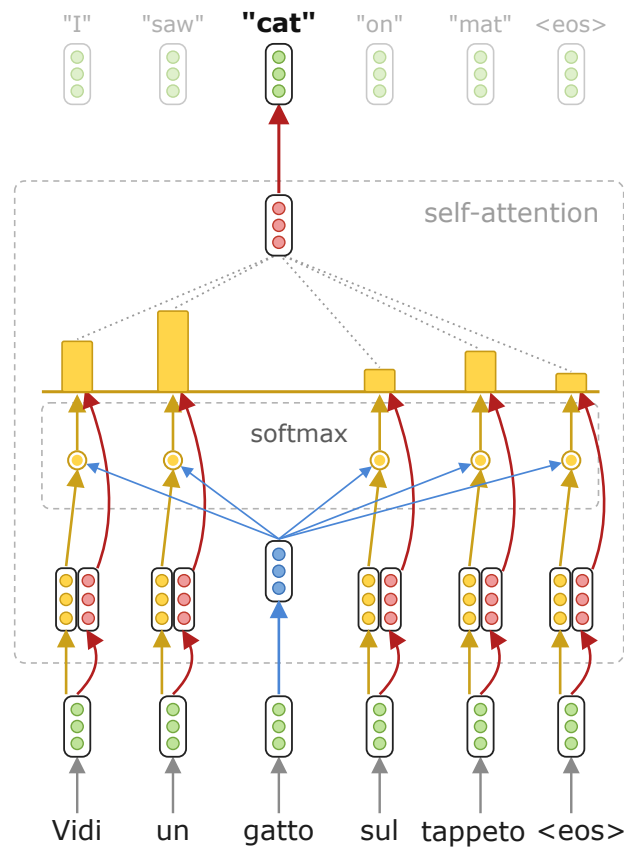


Figure 1.1: Self-attention mechanism diagram (inspired by the work of Elena Voita [12]).

The Transformer architecture is built from a stack of identical blocks, each containing a self-attention layer and a feed-forward network (FFN) layer, with residual connections and layer normalization applied to each.

Figure 1.1 shows the self-attention mechanism in detail for a single token. The main token is projected (with a matrix multiplication) into a query vector q in blue, while all the other tokens are projected into key vectors k in yellow and value vectors v in red. The similarity between the query and each key is computed as the scalar product qk^T , and the resulting scores are normalized with a softmax function to produce a probability distribution. The output for the main token is then a weighted combination of the value vectors, where the weights are given by the normalized scores.

Each of the resulting token representations is then passed through a feed-forward network with non-linearities to produce the final output of the block.

More formally, the attention mechanism is described by the following formula:

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

Here, Q , K , and V denote the query, key, and value matrices, and d_k is the key dimensionality.

Two properties explain why this architecture became dominant. First, any two positions interact in a single step regardless of the distance between them, so long-range dependencies are captured directly rather than through a long chain of recurrent updates. Second, the computation across positions is no longer serial: the attention weights and the resulting combinations are computed for all positions in parallel, as a small number of large matrix multiplications. As discussed in the next section, this maps naturally onto modern parallel accelerators.

1.1.2. From GPT-2 to Frontier Models

The Transformer was quickly adopted as the backbone of generative pretraining (see Section 1.5.1). GPT-2 [13] is the clearest early demonstration: a decoder-only Transformer trained on a large web corpus with the next-token objective described above, scaled up to 1.5 billion parameters. Its main contribution was not a new mechanism but an observation about scale: a model trained purely to predict the next token could perform a range of tasks, such as translation, summarization, and question answering, with no task-specific training and simply by being prompted. Capability emerged from scale and data, not from task-specific engineering.

This observation set the direction for the work that followed. Scaling laws [1] describe the cross-entropy loss as a power law in model size, dataset size, and training compute, a relationship that holds over many orders of magnitude and makes the gains from scale

predictable. GPT-3 [14] took the same architecture to 175 billion parameters and showed that, at sufficient scale, a model could learn a new task from a few examples given in the prompt alone, without any gradient updates. The Chinchilla analysis [2] later showed that these models had been trained on too little data: for a fixed compute budget, parameters and training tokens should grow together. This shifted the emphasis from raw size toward training smaller models on more data.

Acting on these scaling laws was only possible because the hardware kept pace. Every increase in model and dataset size demanded a comparable increase in computation, and over the same period the accelerators available for training grew at an extraordinary rate, with the compute used for the largest training runs rising by many orders of magnitude [15]. General-purpose graphics processing units (GPUs) gave way to chips with dedicated matrix-multiplication units [16], and these were assembled into progressively larger interconnected clusters. Architecture and hardware thus advanced together: the scaling laws showed that larger models were worth building, and the hardware made them feasible to train.

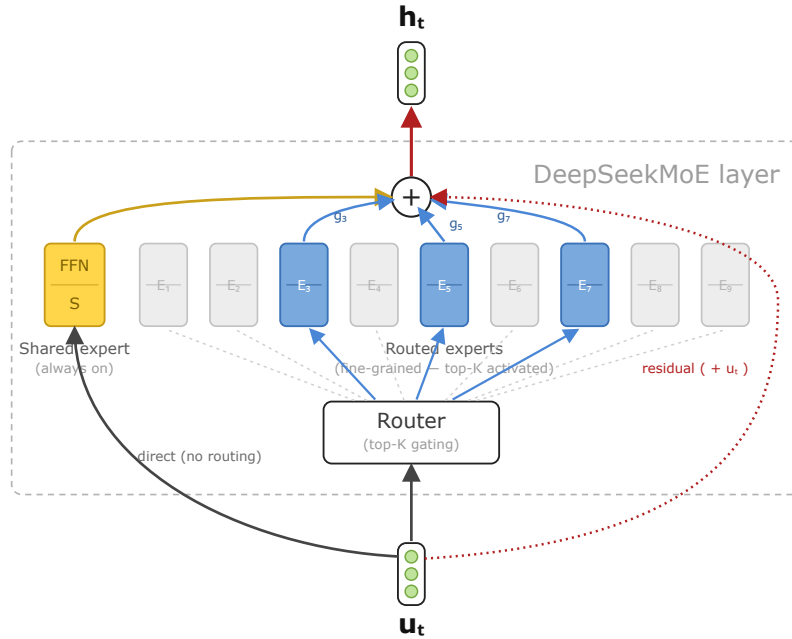


Figure 1.2: Example of a Mixture-of-Experts architecture, in which only a subset of the experts is activated to process a given token.

Scale of this kind eventually runs into an economic limit. In a dense Transformer every

parameter takes part in the processing of every token, so both training and inference cost grow in proportion to the parameter count. The Mixture-of-Experts (MoE) [17] architecture relaxes this constraint: it replaces the single feed-forward network in each block with many expert subnetworks and activates only a few of them for each token (Figure 1.2), so the total number of parameters can grow while the computation per token stays roughly constant. DeepSeek [18] later brought a refined, fine-grained version of the idea to the open frontier, reaching hundreds of billions of parameters while keeping both training and inference far cheaper than a dense model of comparable size. It is this decoupling of parameter count from per-token computation that first carried model sizes into the trillions of parameters [19].

This progress was not confined to a handful of closed laboratories. Alongside the proprietary systems, a parallel line of open-weight releases brought frontier-scale capability into the open, where it could be studied and built upon. Meta’s OPT [20] was an early example, reproducing a GPT-3-scale model of 175 billion parameters and releasing both its weights and a detailed account of the training process. The following year, LLaMA [21] took this idea further: it deliberately trained comparatively small models on even more tokens than compute-optimal training would call for, trading a higher training cost for cheaper inference. The result was a 13-billion-parameter model that matched GPT-3 despite being more than ten times smaller, released with open weights that seeded a large ecosystem of derivative models.

Contemporary frontier systems, such as GPT-5 [22] and the various open and proprietary model families, are the product of this regime: they have hundreds of billions of parameters, are trained on trillions of tokens, are then aligned to follow instructions, and are served from data centers equipped with clusters of specialized accelerators.

1.1.3. A Different Frontier

The trajectory described so far moves in a single direction: more parameters, more data, more compute, and a steadily larger physical footprint. Running a model close to the frontier means renting a data center. This thesis follows the opposite direction. Rather than asking how much capability emerges as models are scaled up, it asks how little has to be given up as they are scaled down, until a model is small enough to run not in a data center but on an edge device such as a wearable, a sensor, or a battery-powered embedded board.

This is not only a matter of using fewer parameters. Below a certain size what limits the model is no longer raw capability but memory, energy, and the structure of the hardware

itself, and even the attention mechanism that made the Transformer successful begins to strain. The next section turns to that hardware, since the constraints of the hardware are increasingly the constraints of the models meant to run on it.

1.2. Hardware

The progression from GPT-2 to contemporary frontier systems was not driven by algorithmic ideas alone. It was made possible, and in large part shaped, by the hardware available to train and serve these models [15]. The Transformer is, at its core, a sequence of large matrix multiplications, and its rise coincided with a generation of accelerators engineered precisely for dense linear algebra.

A discussion of small language models therefore has to begin with the hardware they are meant to run on. The models developed in this thesis do not target any specific chip or architecture, but they still move in the direction that the following sections set out for small, power-efficient accelerators.

1.2.1. Accelerating language models

General-purpose GPUs became the default platform for deep learning because their architecture, in which thousands of simple cores execute the same instruction over many data elements, is an almost perfect match for the operations that dominate neural networks. Training and inference consist overwhelmingly of matrix-matrix multiplications, and a GPU keeps its arithmetic units busy by streaming large tiles of weights and activations through them in parallel. Modern accelerators reinforce this with dedicated tensor cores [16], which perform fused multiply-accumulate operations over small matrix blocks in reduced precision, and with high-bandwidth memory fast enough to keep those units fed. Architecture and hardware co-evolved: attention and feed-forward layers are a good match for accelerators that reward large matrix multiplications, and the hardware in turn made ever-larger models economical. The cost of this regime is scale: such systems consume hundreds of watts and depend on gigabytes of fast memory [3], which is the opposite end of the spectrum from where this thesis is heading.

1.2.2. Toward small devices

This section follows the hardware in the opposite direction, toward devices that are progressively smaller, cheaper, and more power-efficient. Two classes of device mark this path: embedded GPUs and, below them, microcontrollers. Examining each in turn shows

where the Transformer begins to strain.

The least constrained of the two is the embedded GPU, which preserves a familiar GPU programming model within a much smaller power budget. A representative example is the NVIDIA Jetson Orin Nano [23], the smallest accelerator in NVIDIA’s lineup. In its entry-level 4 GB configuration it is a module the size of a memory stick that runs the full CUDA software stack within a single-digit-watt power envelope. It is still a general-purpose accelerator, so it can in principle execute everything a desktop GPU can, including standard Transformer inference. What it lacks is abundant memory. With only 4 GB of RAM shared between CPU and GPU, the limiting factor is no longer raw arithmetic but the footprint of the model weights and, for autoregressive generation, the key-value cache that grows with every token produced. This is the first point at which memory capacity, rather than compute, becomes the governing limit [24].

Microcontrollers sit further along this path, where memory is measured in a few megabytes of on-chip SRAM and power in tens of milliwatts. Two recent devices show how capable this class has become while remaining poorly suited to attention. The STM32N6 [25] is the first STM32 microcontroller to embed a dedicated neural processing unit (ST’s Neural-ART accelerator) next to its host CPU, while GreenWaves’ GAP9 pairs a parallel RISC-V compute cluster with a hardware neural accelerator (the NE16) [26, 27]. Both perform well on real-time convolutional inference for vision and audio at the edge, and both are squarely neural accelerators, yet neither is well suited to running a Transformer, for reasons that are structural rather than a matter of raw speed.

1.2.3. Attention limitations

The mismatch between these accelerators and the Transformer is specific and structural. Their designs rest on assumptions that hold for convolutional neural networks (CNNs) but break for self-attention. First, they expect statically known tensor shapes and compile the network graph ahead of time, whereas the attention score matrix has a dimension proportional to the sequence length and is therefore inherently dynamic [28]. Second, they assume a weight-stationary dataflow [29], in which one operand of each multiplication is a fixed, pre-loaded weight, whereas attention multiplies two tensors that are both produced at runtime (QK^T , then the product with V), so this assumption no longer holds. Third, they provide no native softmax, since integer accelerators have no hardware exponential unit, which leaves the normalization of attention scores as a known bottleneck. In addition, attention is memory-bound [30] and its key-value cache grows with the context length, which is difficult to accommodate within a budget of a few megabytes of on-chip

memory.

The result is a structural tension. The most capable small chips excel at the dense, static, weight-stationary linear algebra of convolutional networks, whereas the attention mechanism that defines the Transformer is dynamic in shape, multiplies pairs of activations, relies on a non-linearity these chips do not natively support, and requires a memory footprint that grows with the context length. Hardware acceleration has advanced significantly, but along a direction that does not align with attention. This is the gap that motivates the rest of the chapter: small language models that fit within these constraints by using alternative architectures.

1.3. Small Language Models

This thesis focuses on very small models. The following subsections survey notable examples with fewer than 150 million parameters, selected because the insights from their published work directly inform the research carried out in this thesis.

Several of these models are also used as baselines in the experimental evaluation.

1.3.1. Transformers

As introduced in Section 1.1.1, the Transformer is the backbone of most language models, and many small models are simply scaled-down versions of this architecture. The following examples are discussed for the insights they provide, but only BabyLlama and MobileLLM are later used in this thesis.

TinyStories

TinyStories originates from a Microsoft study [31] that investigates the lower bound on model size required to produce coherent English text. The authors train on a synthetic dataset of short stories restricted to a vocabulary that a three- to four-year-old child would understand, and show that models between 1M and 33M parameters can generate fluent text with near-perfect grammar. The restricted vocabulary (the 10k most common tokens from the GPT-Neo tokenizer [32]), however, limits the applicability of these models to real-world tasks.

Evaluation is performed not with standard benchmarks but by asking GPT-4 to rate the generated stories on grammar, creativity, and consistency on a scale from 1 to 10. The results show that small models trained on simple, task-specific data can outperform larger general-purpose models such as GPT-Neo 125M [33] on these synthetic tasks, although

they remain constrained by their limited vocabulary and narrow coverage.

MobileLLM

Meta’s MobileLLM paper [34] proposes an efficient architecture tailored for mobile deployment. The authors replace the standard feed-forward network with SwiGLU [35] and explore a range of hyperparameter configurations to identify an effective 125M design, finding that thin and deep architectures outperform wide and shallow ones. The paper also highlights that, for sub-billion-parameter models, the embedding layers represent a significant fraction of the total parameter budget: an embedding dimension of 512 with a 32k vocabulary, for example, yields input and output matrices of 16 million parameters each, which together account for over 20% of a 125M model.

MobileLLM is evaluated on BoolQ, PIQA, HellaSwag, and WinoGrande (see Section 1.4), where it outperforms comparable architectures such as GPT-Neo 125M [33] on most benchmarks.

BabyLlama

BabyLlama [36] won the first BabyLM Challenge [37], a competition focused on training small models from limited data. It uses knowledge distillation (see Section 1.5.2) from an ensemble of two teacher models, GPT-2 [13] (705M parameters) and LLaMA [21] (360M parameters), both trained from scratch for the challenge. The resulting student model has 58M parameters and outperforms both its teachers and other models of comparable size on the BLiMP benchmark (see Section 1.4), which demonstrates the effectiveness of distillation for compact models.

T5 (Text-to-Text Transfer Transformer)

T5 [38] is one of the first models to formalize and validate the pretraining and fine-tuning paradigm: a model is first pretrained on a large general-purpose corpus to acquire broad language understanding, and then fine-tuned on a smaller task-specific dataset to specialize its behavior (see more on Section 1.5). This two-stage approach is the same methodology adopted in this thesis. Unlike the other models discussed in this section, T5 uses an encoder-decoder architecture and is available in sizes down to approximately 60 million parameters.

1.3.2. Alternative Architectures

As described in Section 1.1.1, the Transformer relies on self-attention to capture long-range dependencies. This mechanism is computationally demanding (the complexity scales quadratically with respect to the input sequence length) and benefits from specialized hardware.

Several alternative architectures have since been proposed to reduce this cost, targeting linear or sub-quadratic complexity while preserving competitive performance. Among them, this thesis adopts Mamba because it reaches competitive performance with mature and widely supported implementations, whereas xLSTM depends on specialized hardware and custom software kernels that are not yet broadly available.

Linear Attention

Linear attention [39] removes the quadratic cost of self-attention by replacing the exponential similarity $\exp(q_i k_j^T)$ with a factorized kernel $\phi(q_i)^T \phi(k_j)$, where $\phi(\cdot)$ is a simple feature map. This factorization, combined with the associativity of matrix multiplication, allows attention to be rewritten as a recurrence with a fixed-size state matrix S_i :

$$S_i = S_{i-1} + \phi(k_i) v_i^T$$

The model therefore behaves like a recurrent network, with linear-time inference and constant memory with respect to sequence length. This efficiency comes at a cost. The update above is purely additive: each token is accumulated into S_i , and there is no way to remove information that has already been written, so linear attention cannot forget. Softmax attention avoids this problem because it recomputes a fresh weighting over the entire history at every step and can assign near-zero weight to irrelevant tokens.

In addition, compressing the entire sequence into a single fixed-size matrix limits memory capacity: once too many tokens are stored, they interfere with one another and retrieval becomes lossy [40]. For these reasons, plain linear attention underperforms softmax attention [41], and later architectures such as Mamba and xLSTM recover much of the gap by adding a data-dependent gate that lets the recurrent state selectively forget.

Mamba

Mamba [5] is a class of selective state-space models (SSMs [42]) designed to be more computationally efficient than Transformer-based architectures. Rather than interleaving attention and FFN blocks, it adopts a simplified homogeneous block structure that merges these components and approximates selective attention using only matrix multiplications, which makes the computation highly parallelizable.

At its core, Mamba is built on a discrete-time state-space model defined by the following recurrence:

$$h_t = \bar{A}h_{t-1} + \bar{B}x_t, \quad y_t = Ch_t$$

Here, x_t is the input at time step t , h_t is the hidden state, and y_t is the output. The matrices $\bar{A}$ and $\bar{B}$ are discretized versions of the continuous-time parameters A and B , obtained via a learned step size Δt .

The key contribution of Mamba over prior SSMs is selectivity: the parameters B , C , and Δt are not fixed but are computed as functions of the input x_t (Figure 1.3).

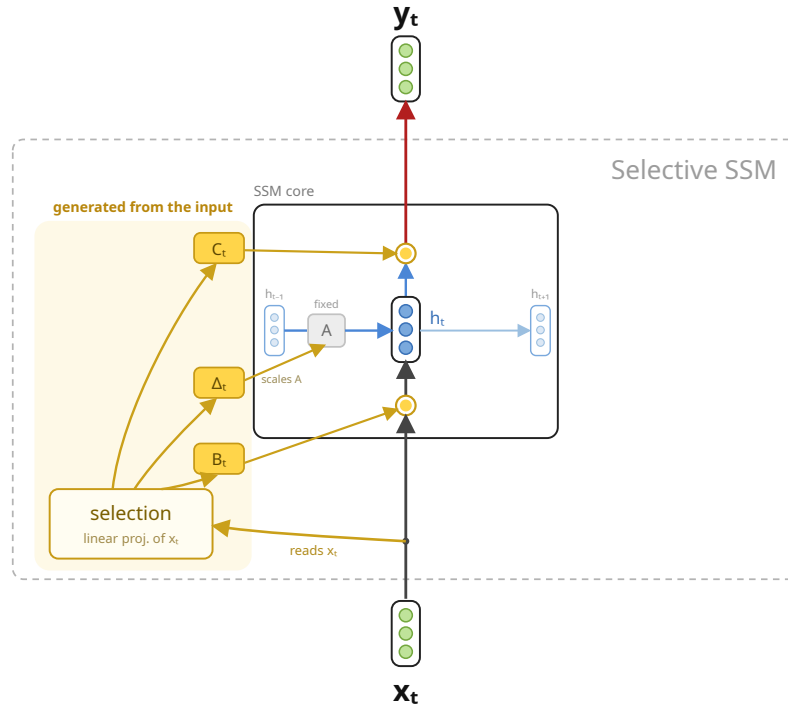


Figure 1.3: Selective state-space model block diagram.

This input-dependent parameterization allows the model to selectively retain or discard

information at each step, which gives it a context-awareness comparable to attention but with linear complexity in the sequence length.

Mamba models have been evaluated at sizes up to 2.8B parameters, where they show consistent improvements over Transformer-based models of comparable size on benchmarks such as MMLU, HellaSwag, and WinoGrande (see Section 1.4).

xLSTM

xLSTM (Extended Long Short-Term Memory) [43] modernizes the original LSTM [44] cell so that it can scale to larger parameter counts while maintaining linear computational complexity. The architecture introduces two main modifications:

1. Exponential gating: the sigmoid gates are replaced with exponential functions, combined with normalization and stabilization techniques, which improves selective information flow.
2. Matrix memory: the scalar cell state is replaced by a matrix, which increases storage capacity and enables richer internal representations.

Evaluations of xLSTM models up to 7B parameters report performance competitive with Transformer models on standard benchmarks and superior results on state-tracking tasks such as parity checking.

1.4. Benchmarks

Evaluating small language models requires benchmarks that probe the abilities most affected by reduced scale. This section covers benchmarks focused on knowledge, commonsense reasoning, and language understanding, since these are the capabilities most sensitive to model size and most relevant to the objectives of this thesis.

These benchmarks are designed for general-purpose language models and assume capabilities, such as broad world knowledge and multi-domain reasoning, that lie outside the scope of very small, task-specific models. For this reason, none of them are used directly in this work. Their design principles, however, informed the construction of the custom evaluation benchmarks tailored to the specific domains and tasks targeted by this thesis, as outlined in Section 2.2.

1.4.1. Commonsense and Knowledge

The benchmarks in this group share a common structure: the model is presented with a question and a set of candidate answers, but the prompt itself does not contain the information needed to select the correct one. Success depends entirely on knowledge the model has internalized during training. This makes them useful for measuring how much world knowledge is retained as model size decreases.

- **MMLU** (Massive Multitask Language Understanding): A multiple-choice benchmark covering 57 subjects across STEM, the humanities, and the social sciences, designed to measure the breadth of world knowledge and multi-domain problem-solving ability that a model acquires during training [45].
- **HellaSwag**: A multiple-choice benchmark for physically situated commonsense, in which the model must select the most plausible continuation of a short activity description from four options. The correct answer cannot be identified from the prompt alone and requires grounded knowledge of how everyday activities unfold [46].
- **PIQA** (Physical Interaction QA): A binary-choice benchmark for physical commonsense, in which the model must select the more plausible of two solutions to an everyday physical task. As with HellaSwag, the prompt does not contain the answer, so the model must draw on internalized knowledge of the physical world [47].

1.4.2. Reasoning and Language Understanding

The benchmarks in this group are structured so that all the information required to answer correctly is present in the prompt itself. The model does not need prior world knowledge; instead, it must read, comprehend, and reason over the provided context. This isolates linguistic and reasoning ability from factual recall, which makes these benchmarks particularly informative for evaluating very small models whose knowledge retention is inherently limited.

A clarification is needed on the term *reasoning*. In recent years it has increasingly come to denote a model’s ability to reason explicitly, producing a chain of intermediate steps through special reasoning tokens before committing to a final answer. This is not the sense intended here. In this thesis, reasoning refers to the model’s internal ability to apply the knowledge acquired during training to a question presented in a different format, without any explicit intermediate steps.

- **BoolQ**: A reading comprehension benchmark consisting of binary (yes/no) questions, each paired with a short passage that contains all the information needed to answer. The model must identify and reason over the relevant part of the passage to arrive at the correct answer [48].
- **WinoGrande**: A fill-in-the-blank benchmark for commonsense reasoning based on pronoun resolution, scaled up from the Winograd Schema Challenge. The candidate answers are adversarially filtered to remove statistical shortcuts, which requires the model to resolve references through contextual understanding rather than word associations [49].
- **BLiMP** (Benchmark of Linguistic Minimal Pairs): A benchmark for grammatical sensitivity consisting of sentence pairs that differ by a single linguistic element, one grammatically valid and one invalid. The model must assign a higher probability to the valid sentence, which reveals whether it has internalized the underlying syntactic rules or is relying on surface-level patterns [50].

1.5. Language Model Lifecycle

Building a language model is not a single operation but a sequence of stages, each of which adds a specific capability. A model first learns the general structure of language during pretraining or distillation, acquiring fluency and broad knowledge from large amounts of unlabeled text. Supervised fine-tuning then specializes it for a concrete task.

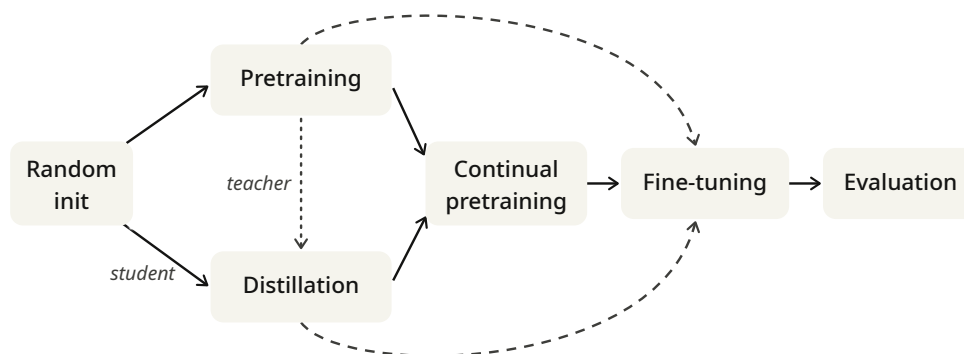


Figure 1.4: The language model lifecycle, from initialization to evaluation.

1.5.1. Pretraining

Pretraining is the initial phase in the development of a machine learning model, during which the model is trained on very large datasets to learn the general patterns, structures, and relationships present in the data. In NLP, the goal of pretraining is to give the model

a broad understanding of language, including its grammar, syntax, semantics, and the contextual relationships between words and sentences. This phase is typically carried out through self-supervised learning, in which the training signal is generated automatically from the data itself, so that the model can learn without manually labeled datasets. The knowledge acquired during pretraining serves as a foundation that can later be adapted to specific downstream tasks through fine-tuning.

The standard objective of this phase is causal language modeling, in which the model learns to predict the next token given the previous ones by minimizing the cross-entropy loss:

$$L_{CE} = - \sum_i^V y_i \log p(i)$$

Here, y_i is the one-hot encoding of the true next token, $p(i)$ is the probability distribution over the vocabulary produced by the model, and V is the size of the vocabulary. By minimizing this loss across a large corpus of text, the model learns to capture the statistical properties of language, which can then be used for various NLP tasks.

Continual Pretraining

Continual pretraining is a continuation of the pretraining phase, but the process does not start from a randomly initialized model. It starts instead from a model that has already been pretrained on a large general-purpose corpus and therefore already possesses fluent language abilities, text comprehension, and broad world knowledge. This existing model undergoes a second round of self-supervised training on a new corpus, typically focused on a specific domain, language, or body of information that was underrepresented in the original data. The objective remains the causal language modeling loss used during the initial pretraining, so no manually labeled data is required; the only change is the composition of the training corpus.

The main advantage of this approach is efficiency: the model reuses the general linguistic competence acquired during the first pretraining and only needs to absorb the new specialized information, which considerably reduces the amount of data and computation required compared to training an equivalent model from scratch. The principal difficulty is catastrophic forgetting [51], in which adapting the parameters to the new corpus degrades the general capabilities learned previously. This effect is usually mitigated by carefully tuning the learning rate and by mixing a portion of general-purpose data into the new corpus, so that the model retains its original knowledge while acquiring the new one [52].

1.5.2. Distillation

Distillation [53] is a training technique used to transfer knowledge from a large and complex model, called the teacher, to a smaller and more efficient model, called the student. Its main goal is to reduce the computational and memory requirements of the model while preserving most of its performance. During distillation, the teacher produces soft predictions, namely probability distributions over the possible outputs, which carry richer information than simple hard labels. The student is then trained to reproduce these soft predictions, so that it learns not only the correct outputs but also the relationships and uncertainties captured by the teacher. As a result, the student can reach competitive performance despite its significantly smaller architecture, which makes distillation particularly useful for deploying language models in resource-constrained environments such as mobile devices or real-time systems.

Distillation requires a loss that can measure the distance between the output distribution of the teacher and that of the student. The most common choice is the Kullback-Leibler divergence with a scale factor, defined as:

$$L_{KL} = T^2 \sum_i^V p(i) \log \frac{p(i)}{q(i)}$$

Here, $p(i)$ and $q(i)$ are the probability distributions produced by the teacher and the student, respectively, both computed with a softmax scaled by the temperature parameter T . The temperature plays two roles. Inside the distributions, it controls their smoothness: a higher temperature produces softer distributions that expose more of the teacher’s knowledge and help the student learn more effectively. The leading T^2 factor then compensates for the effect of this scaling on the gradients, which keeps the contribution of this term balanced with the cross-entropy loss.

$$L = \alpha L_{CE} + (1 - \alpha) L_{KL}$$

The total loss used for distillation combines the cross-entropy loss, which measures the distance between the student’s output and the true labels, and the KL divergence, which measures the distance between the student’s output and the teacher’s output. The hyperparameter α controls the balance between these two terms.

1.5.3. Supervised Fine-tuning

After pretraining, a base model can be fine-tuned on a task-specific dataset to adapt its knowledge and abilities to a particular task. Instead of training a model from scratch, fine-tuning reuses the representations and patterns already learned during the initial large-scale training, which significantly reduces both the computational cost and the training time. In NLP in particular, large language models are adapted to specialized tasks such as text classification, question answering, summarization, and domain-specific conversational systems. During fine-tuning, the model parameters are updated on the new data so that the model better captures the characteristics, terminology, and objectives of the target task. These updates only displace the weights slightly within the loss landscape: rather than relearning the task from scratch, fine-tuning starts from an already good solution and moves it toward a nearby region better suited to the target task. This is why the general knowledge acquired during pretraining is largely retained.

The term "supervised" refers to the nature of the training signal. Unlike pretraining, whose objective is self-supervised and derives its target automatically from raw text by predicting the next token, fine-tuning relies on a labeled dataset of explicit input-output pairs, in which the desired output is provided externally, typically through human annotation or curated task-specific data. The model is trained to reproduce these target outputs given the corresponding inputs. It reuses the same causal language modeling objective and cross-entropy loss introduced earlier, but the loss is now computed against the known correct answers rather than against the input text itself.

1.5.4. Evaluation

Evaluating a language model is less direct than evaluating a standard classifier. Its output is free text, generated one token at a time from a probability distribution over the vocabulary, so it does not necessarily match a predefined label or the exact wording of the expected answer. The model may express the correct answer in an unexpected form, or produce text that falls outside the set of valid responses entirely. Any evaluation procedure therefore needs a criterion that maps this free-form output onto the quantity of interest. For the closed-form benchmarks used in this thesis, in which each question has a small set of candidate answers, a natural criterion is to compare the probabilities that the model assigns to each candidate, rather than to inspect the text it generates.

The main tool used for this purpose is the EleutherAI `lm-evaluation-harness` library [54], which provides a simple and consistent interface for evaluating language models on a wide range of benchmarks. It also makes it easy to add new benchmarks and models.

The metric used throughout this work is accuracy. It is simple and intuitive, but applying it to a language model is not straightforward, again because the output is free text that may not exactly match the expected answer. The most direct option is to read the answer the model generates and count how often it matches the expected one, a quantity that can be called "raw accuracy":

$$\text{Raw Accuracy} = \frac{\# \text{ correct answers}}{\# \text{ total questions}}$$

`lm-evaluation-harness` instead follows the probability-based criterion described above: for each question it computes the probability that the model assigns to every valid answer (for example "yes" and "no" for the yes/no benchmark) and selects the most probable one as its prediction. In this way, a model that would otherwise generate a token outside the set of valid answers is still scored correctly, as long as the highest-probability token within that set is the correct one:

$$\text{lm-eval Accuracy} = \frac{\# \text{ correct predictions among valid options}}{\# \text{ total questions}}$$

These two metrics almost never coincide, since, especially at the start of fine-tuning, the models are not accustomed to predicting the correct format and may respond with free text, just as they were trained to do during pretraining (Figure 3.2 shows a practical example).

2 | Data Collection

Small language models are not meant to compete with large ones on every topic. With few parameters, a small model works best on a single, well-defined task, where all of its limited capacity goes into one area instead of being spread across many [55–57]. Yet small models are usually tested in the opposite way, on general-purpose datasets that measure broad world knowledge and commonsense reasoning, such as the benchmarks described in Section 1.4.

This is a mismatch. These benchmarks expect wide knowledge and reasoning across many domains, which a very small model simply cannot have. Testing a specialized model on them reveals little about the quality that matters here: how well it performs inside its own domain. A useful evaluation has to measure depth in a single domain instead.

For this reason, the thesis takes a different path. Two domains were chosen, finance and medicine, and a dedicated question-answering benchmark was built for each. The rest of the chapter explains how, in two steps: Section 2.1 presents the source datasets, and Section 2.2 shows how they are turned into structured, reproducible evaluation tasks.

2.1. Datasets

Three datasets were chosen for this work, each with a clear role. Two of them are domain-specific and provide the material for the new benchmarks, one for medicine and one for finance. The third is a general-purpose corpus used to pretrain new models.

The two domain-specific datasets serve two purposes. They give the models a rich source of text to learn from, and they keep the vocabulary and concepts narrow, which is what a small model needs in order to specialize. Each domain also has its own terms, conventions, and patterns of reasoning, so training on one of them is a realistic test of how well a small model can master a focused field.

To keep the problem focused, all datasets are restricted to English. Conneau et al. [58] show that a model with a fixed number of parameters can suffer from the "curse of multilinguality": adding more languages lowers its performance in each one, because its

limited capacity must be shared among them. Using a single language avoids this loss and lets the model spend all of its capacity on the target domains.

2.1.1. WikiDoc

The medical domain is covered by WikiDoc [59], a collection of question-answer pairs taken from the WikiDoc website [60], a collaborative platform where medical professionals share up-to-date clinical knowledge. The dataset matches the target domain closely, but two caveats should be kept in mind:

- Medicine is a broad field, and although it is narrower than general knowledge, it may still be too wide for a small model to gain the full benefit of specialization.
- The question-answer pairs were not written by people but generated from the source articles by a large language model, a process that can introduce bias or lower the quality of the data.

An example pair is shown in Example 2.1.

Example 2.1 Question-answer pair in the WikiDoc dataset.

Question: "Can you provide an overview of the lung's squamous cell carcinoma?"

Answer: "Squamous cell carcinoma of the lung may be classified according to the WHO histological classification system into 4 main types: papillary, clear cell, small cell, and basaloid."

2.1.2. Investopedia

The financial domain is covered by Investopedia [61], a collection of question-answer pairs taken from the Investopedia website [62], a leading online resource for financial education. Each pair also comes with a short context passage. The dataset shares the same two limitations as WikiDoc (a broad topic and LLM-generated pairs), but it also brings two advantages:

- Its scope is narrower than finance as a whole. Investopedia is written for readers with no financial background, so it stays close to introductory concepts and plain vocabulary.
- To reduce hallucination, the authors added a self-verification step that checks each generated answer against its source passage, giving more confidence that the pairs are factually grounded.

An example item is shown in Example 2.2.

Example 2.2 Question-answer pair and context in the Investopedia dataset.

Context: "Because home equity loans are secured by your home, they generally offer low interest rates when compared to unsecured options like personal loans. These low rates can help you save money on your home remodel costs. Unlike home equity lines of credit (HELOCs), home equity loans have fixed interest rates. This means you aren't vulnerable to rate changes, and your monthly payment will remain stable for your entire repayment term. Home equity loans can be used for a wide variety of purposes, including home improvements. While your lender may ask what the money will be used for, you generally won't have to provide any sort of documentation"

Question: "What are some advantages of using a home equity loan for a home remodel compared to unsecured options like personal loans?"

Answer: "home equity loans generally offer low interest rates when compared to unsecured options like personal loans, which can help save money on home remodel costs. Additionally, they have fixed interest rates, providing stability in monthly payments and protection from rate changes during the entire repayment term."

2.1.3. Wikipedia

The third dataset plays a different role from the other two. The domain datasets are about specialization, but a model cannot specialize before it has a grasp of the language itself. This is the purpose of pretraining: training on a large corpus to learn the basic structure of English and acquire a broad base of world knowledge.

To this end, up to 900k English-only articles were taken from a curated subset of Wikipedia [63]. Unlike the two domain datasets, this corpus has not been processed by any language model. It comes as plain text, cleaned of HTML and other non-textual elements, which makes it a reliable source of language structure and factual knowledge for the BabyMamba pretraining phase.

2.2. Task Generation

The WikiDoc and Investopedia datasets each contain thousands of question-answer pairs in free text, where the answer is a full written sentence. This is a problem for evaluation: the same fact can be phrased in many ways, so deciding automatically whether the model is right becomes unreliable. A controlled and reproducible benchmark instead needs answers in a fixed form that is easy to check.

To make this possible, the original pairs were used as raw material to generate two kinds of structured questions: binary yes/no questions and multiple-choice questions. The conversion was carried out with the large language model GPT-5 [22], using a one-shot prompting strategy. For each item, the model received a system prompt describing the

task and the required output format, one example question-answer pair, and the source passage from which the new question had to be built. The result is two benchmarks per dataset, one binary and one multiple-choice.

2.2.1. Yes/No Benchmark

The first benchmark is inspired by BoolQ and is made of yes/no questions. There is one key difference: BoolQ gives the model a context passage to read, whereas here no passage is provided, so the model must answer from what it has learned during training (Example 2.3).

Example 2.3 Example of yes/no prompt generated from the WikiDoc dataset.

Question: Is papillary a main type of lung squamous cell carcinoma according to the WHO histological classification system?
 Answer:yes

The prompt follows a fixed structure. The question comes after "Question:" and a space, while the answer comes immediately after "Answer:", with no space in between. This missing space is deliberate. The evaluation tool `lm-evaluation-harness` reads the answer by matching the exact token "yes" or "no", and a leading space would turn it into a different token and break the match. During training the model sees the full prompt, including the answer token; during evaluation it sees only the question and the "Answer:" prefix, and it must predict the right token itself.

There is also a risk of bias. Because the prompts are generated by an LLM, the model might pick up on patterns in how a question is written and guess the answer from its form alone. To prevent this, GPT-5 is asked to produce two questions for each sample, one whose answer is "yes" and one whose answer is "no", identical in structure and differing only in content. The dataset is then balanced between the two answers, and the structure alone gives nothing away: the model must understand what the question actually asks in order to answer it.

2.2.2. Multiple-choice Benchmark

The second benchmark is inspired by MMLU and is made of multiple-choice questions, each with four options (A, B, C, D) and a single correct answer. As in the yes/no benchmark, no context is provided, so the model must rely on what it learned during training (Example 2.4).

As before, the prompt follows a fixed format, which the evaluation tool relies on to read

Example 2.4 Example of multiple-choice prompt generated from the Investopedia dataset.

Question: What type of rate do home equity loans have?

A: Variable

B: Fixed

C: Unsecured

D: HELOC

Answer:B

the answers correctly.

This benchmark faces a similar risk, but of a different kind. GPT-5 is free to choose the wording of the question and its four options, and if the correct answer kept landing in the same position, the model could learn to favor that position instead of reading the options. To rule this out, a deterministic algorithm shuffles the options of every sample so that, across the whole dataset, each of the four positions is correct the same number of times. The correct answer is then equally likely to be A, B, C, or D, and position alone gives nothing away.

One practical constraint shapes the format. Very small models such as BabyLlama have a context window of only 128 tokens, so each option is limited to a single word to keep the prompts short. A few prompts still exceed this limit. Rather than shortening them in advance, truncation is applied dynamically and only when the model's context window is too small to hold the full prompt. In that case the prompt is cut from the left rather than the right, since trimming words from the start of the question is safer than dropping answer options, and the most important information usually sits near the end of the prompt. A model with a larger context window always receives the complete prompt.

3 | Baseline Training

This chapter describes the procedure used to create the baseline models for this thesis, together with the results obtained.

The models available on Hugging Face are already pretrained, and they reproduce the results reported in their original papers on standard benchmarks such as BoolQ and MMLU; on the new benchmarks introduced in this thesis, however, they perform poorly. For this reason, although the main goal of this thesis is to create a new model, the existing models must first be fine-tuned on these benchmarks in order to provide a fair comparison with the new BabyMamba model.

3.1. Fine-tuning

The first approach considered is to fine-tune the models directly on the new datasets, after splitting the data into training, validation, and test sets (60%, 20%, 20%). Although the models are small, the available hardware and the time constraints do not allow cross-validation to be performed for every experiment, so a single split of the data is used.

This section first describes the methodology adopted for the fine-tuning phase, then presents a complete example of the pipeline applied to the fine-tuning of MobileLLM on the wikidoc-multi dataset, and finally summarizes the results for all the models and benchmarks.

3.1.1. Methodology

For each of the three models (BabyLlama, MobileLLM, and Mamba), a small search is performed to identify the best learning rate for the fine-tuning phase. The fine-tuning is run for 20 epochs using a set of learning rates ranging from 5×10^{-6} to 1×10^{-4} with a constant scheduler, and the learning rate and epoch that yield the best performance on the validation set are selected.

In addition, the loss is computed only on the answer token and not on the question, since in this phase the model should learn to answer the question and to become familiar

with the required format. This choice follows works such as "LIMA: Less Is More for Alignment" [64], which show that the model learns the basic structure of the language and acquires general knowledge during pretraining, whereas during fine-tuning it learns to perform specific tasks and to respond in a specific format.

The first problem observed when evaluating the models manually is that they do not follow the required format: instead of responding with "yes/no" or "A/B/C/D", they produce free text that is neither easily evaluable nor within the scope of the task. Computing the loss only on the answer token forces the model to adopt the correct format without degrading the understanding of the language and the knowledge already acquired during pretraining.

3.1.2. Example pipeline

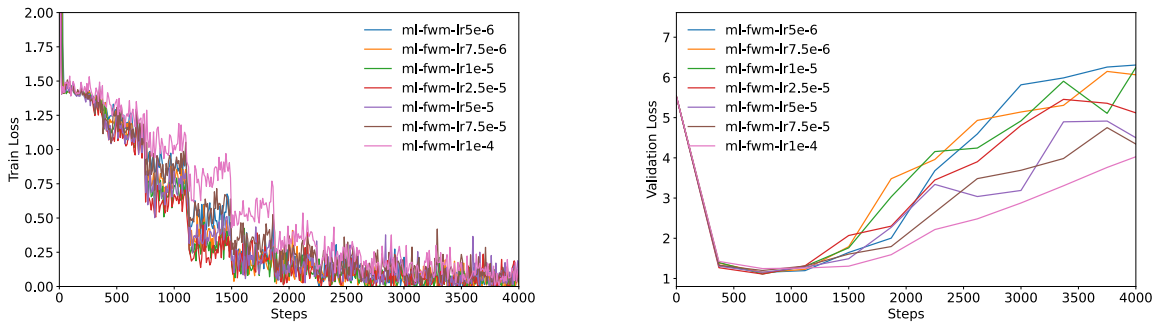


Figure 3.1: Train and validation loss for different learning rates during the fine-tuning phase of MobileLLM on the wikidoc-multi dataset.

The analysis begins with the training and validation loss during the fine-tuning of MobileLLM on the wikidoc-multi dataset for the different learning rates. As shown in Figure 3.1, after two or three epochs the training loss begins to decrease in a staircase pattern while the validation loss begins to increase, which are typical signs of overfitting.

As noted in Zhou et al. [64], only a few examples and a few epochs are needed to align a model to a specific task.

Since the metric to be maximized on the test benchmark is accuracy, Figure 3.2 shows the two accuracy metrics and the difference between them. The `lm-eval` accuracy starts at around 0.25, which for a multiple-choice dataset with four options corresponds to random guessing, while the raw accuracy starts at around 0, indicating that the model does not respond with the correct format at all.

This difference can be explained by examining the total probability that the model assigns

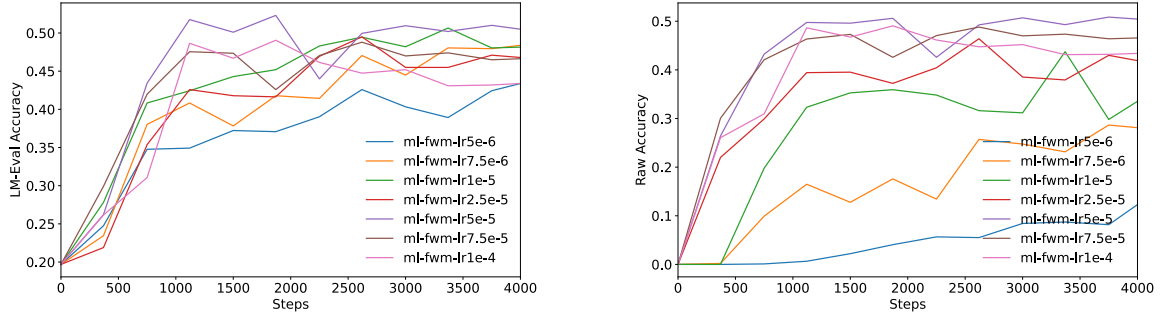


Figure 3.2: `lm-eval` and raw accuracy for different learning rates during the fine-tuning phase of MobileLLM on the wikidoc-multi dataset.

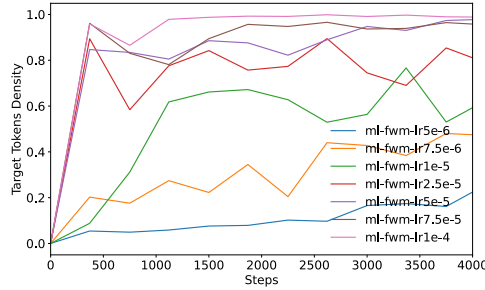


Figure 3.3: Sum of the probability assigned to each of the possible answers during the fine-tuning phase for different learning rates of MobileLLM on the wikidoc-multi dataset.

to the possible answers. As shown in Figure 3.3, at the start of fine-tuning the model assigns a very low probability to all of them, indicating that it tends to respond with free text rather than with the correct format. In this regime the `lm-eval` accuracy essentially measures noise, since it is meaningless to compare the probabilities assigned to the answers when they are all around 1×10^{-5} . After two or three epochs, for the most aggressive learning rates, the model begins to assign almost all the probability to one of the possible answers. At this point the `lm-eval` accuracy reaches 0.5, clearly above the random classifier, and the raw accuracy, which is also around 0.5, confirms that the model now responds with the correct token rather than with free text.

Figure 3.2 also shows that the best learning rate is 5×10^{-5} , since it reaches the highest `lm-eval` accuracy, 0.5230, at epoch 4.

3.1.3. Experiments

All the models follow the same pattern described in Section 3.1.2; Figure 3.4 shows the evolution of the `lm-eval` accuracy for the best model of each kind on all the datasets.

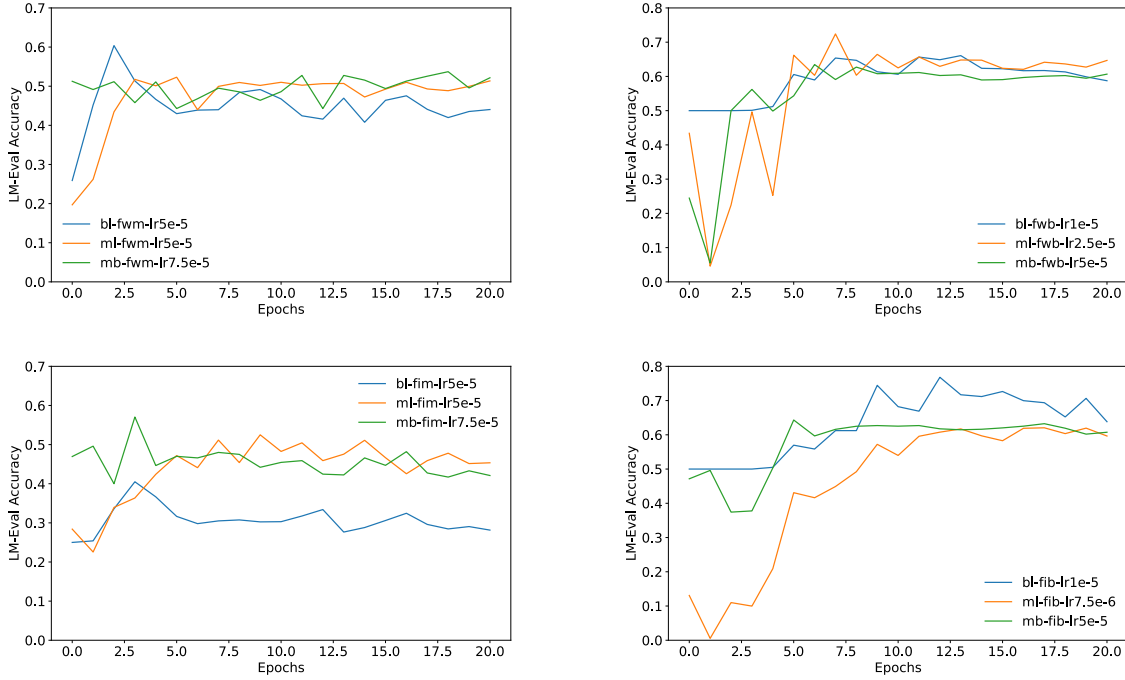


Figure 3.4: `lm-eval` accuracy of the best models of each kind for all the datasets.

These runs, however, also reveal some differences.

The first observation is that Mamba models are unstable at high learning rates; Figure 3.5 shows that the training of the Mamba model with learning rate 1×10^{-4} stops at the second epoch, because the loss becomes NaN and the model can no longer learn.

The same plot shows that lower learning rates fail to make the model assign a high probability to the possible answers, so for very low learning rates the `lm-eval` accuracy is not meaningful, while even for higher ones the raw accuracy remains much lower than the `lm-eval` accuracy, which would be a problem in a real scenario.

The same problem also appears for the best MobileLLM model on the investopedia-binary dataset (the last plot of Figure 3.4), where the selected learning rate is only 5.5×10^{-6} ; the same behavior can be seen in Figure 3.3, even though that figure refers to a different dataset.

A second observation is that BabyLlama, the smallest model, performs comparably to or worse than the other models, as would be expected, except on the investopedia-binary dataset, where it is the best model. This is a surprising result, since the binary benchmarks are those on which the models tend to struggle the most.

To investigate this result, the training was repeated using different splits of the data as the

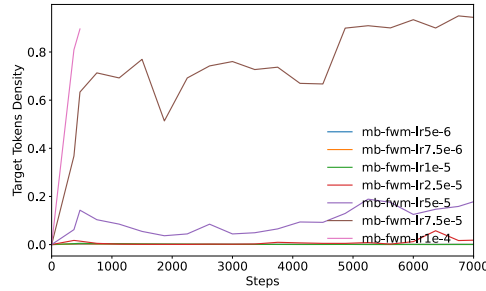


Figure 3.5: Sum of the probability assigned to each of the possible answers during the fine-tuning phase for different learning rates of Mamba on the wikidoc-multi dataset.

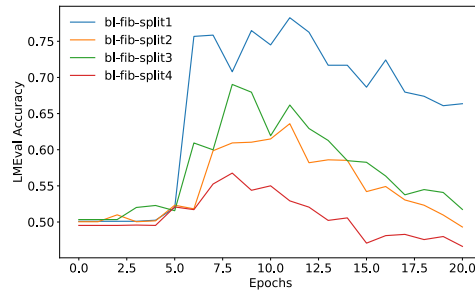


Figure 3.6: BabyLlama performance on the investopedia-binary dataset with different splits of the data as validation set during the fine-tuning phase.

validation set. Figure 3.6 shows that the choice of split has a large impact on performance: the accuracy ranges from almost 0.8 down to 0.5, equivalent to random guessing.

To treat all models equally, the original result, which uses the same validation set for every model, is retained, even though the performance on the binary benchmarks remains highly unstable.

3.2. Continual pretraining

The simple fine-tuning approach appears solid, but it focuses only on teaching the model to answer in the required format, and it may not be enough for the model to learn the specific knowledge of the domain. This is especially true because the model is trained only on the information contained in the training split. The training, validation, and test splits share the same format; however, given the large variance of the data, the validation and test sets are likely to contain questions about facts never seen during training, so the model is effectively asked about an entirely unfamiliar topic.

3.2.1. Methodology

For this reason, a different approach is also tested: the model first undergoes a continual pretraining phase on the whole dataset (training, validation, and test) and is then fine-tuned on the training split. In this way the model sees every fact at least once during continual pretraining and then learns to answer in the correct format during fine-tuning.

The original datasets are used to create new full-text samples, which are then shuffled and split into training and validation sets (80%, 20%).

This may raise concerns about data leakage, since the model could learn to answer the questions during continual pretraining; however, this is not the case, because during this phase the model is fed the original text of the datasets rather than the question-answer pairs. The model can therefore acquire the knowledge contained in the dataset, but it still has to reason about and understand each question in order to answer it correctly during fine-tuning.

It is important to note that continual pretraining is not the same as a standard pretraining phase: the model weights and the knowledge acquired during the original pretraining must not be degraded, especially since that pretraining was carried out on a much larger dataset and with far greater computational resources than those available for this work. For this reason, smaller learning rates than in standard pretraining are used.

3.2.2. Continual pretraining experiments

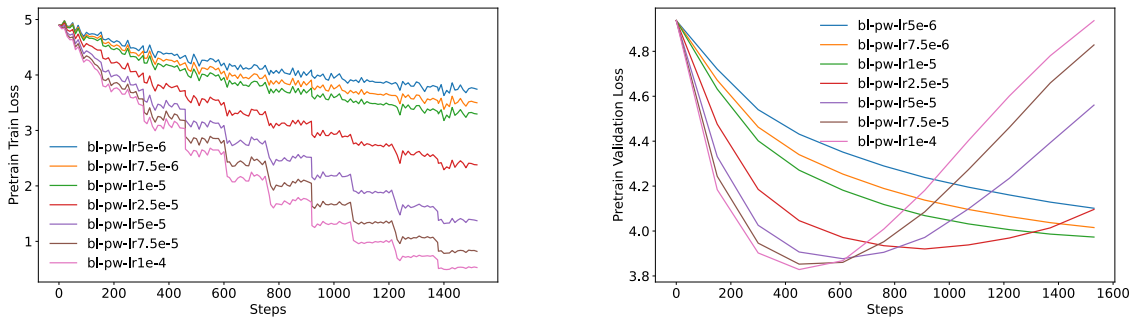


Figure 3.7: Train and validation loss for different learning rates during the continual pretraining phase of BabyLlama on the wikidoc-text dataset.

Figure 3.7 shows an example of the hyperparameter search over learning rates during the continual pretraining of BabyLlama on the wikidoc-text dataset. The validation loss drops rapidly in the first epochs, but the higher the learning rate, the sooner it begins to

increase again, entering an overfitting regime in which the staircase pattern in the training loss reappears.

This behavior is the same for all models and datasets, as shown in Figure 3.8, and it is a clear sign that the models need only a few epochs to learn the knowledge contained in the dataset before they begin to overfit, so the learning rate and the number of epochs for continual pretraining must be chosen carefully.

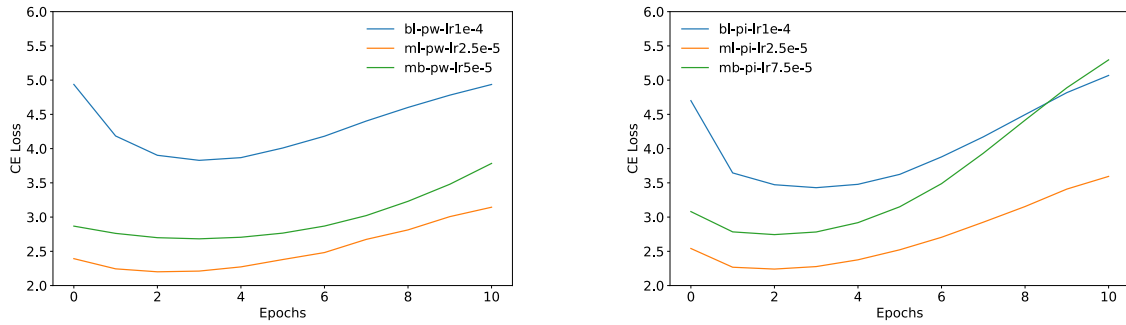


Figure 3.8: Validation loss of the best models of all kinds and datasets during the continual pretraining phase.

An interesting observation is that the best learning rate is almost the same for all the models across the two datasets, as are the minimum validation loss reached and the epoch at which it occurs.

As expected, BabyLlama, the smallest model, has the worst performance, while Mamba and MobileLLM appear to model the data better.

3.2.3. Fine-tuning after continual pretraining

After the best models for the continual pretraining phase had been identified, they were fine-tuned again on the downstream tasks, following the same procedure described in Section 3.1.

Figure 3.6 already showed that the performance of the models depends strongly on the specific split of the data, and Figure 3.9 now shows that BabyLlama has significantly lower performance than the other two models on the multiple-choice datasets.

On the multiple-choice datasets, the models again reach performance well above the random classifier, whereas on the binary datasets the gap is smaller, as was already the case in the simple fine-tuning phase (Figure 3.4).

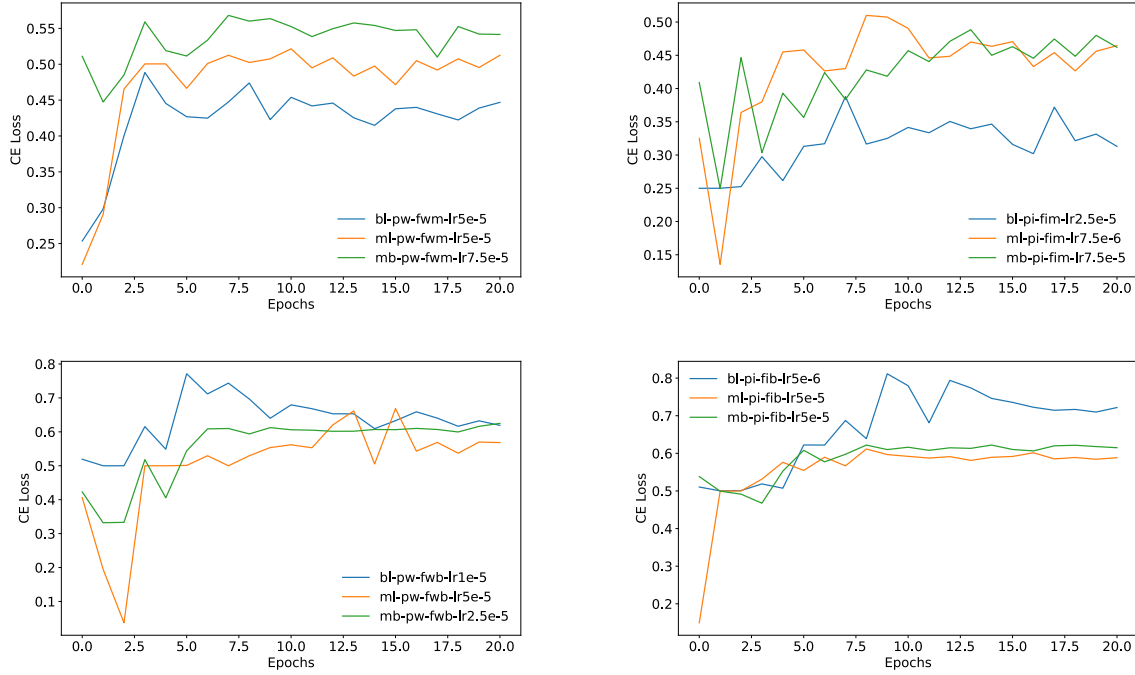


Figure 3.9: `lm-eval` accuracy of the best models of each kind for all the datasets in the fine-tuning phase after continual pretraining.

3.3. Results

Finally, the best model of each kind is collected and evaluated on the test set of the benchmarks using the `lm-eval` tool. Figure 3.10 reports these results for all the benchmarks.

The models reach good performance on the multiple-choice benchmarks, with accuracies well above the random classifier.

On both datasets, the base models from Hugging Face perform almost exactly at the level of the random classifier, because they are not accustomed to answering questions in that specific format and, as shown in Figure 3.11, they always predict the same answer. It should be remembered that these models assign a very low probability to all the possible answers, so the `lm-eval` accuracy is not meaningful in this case.

As expected, even with the direct fine-tuning approach the models learn the format and begin to understand the questions, responding correctly far more often. The more interesting pattern, however, is that the models fine-tuned after continual pretraining almost always outperform those fine-tuned directly; the difference is small, but it suggests that continual pretraining allows the model to acquire some of the knowledge contained in the dataset.

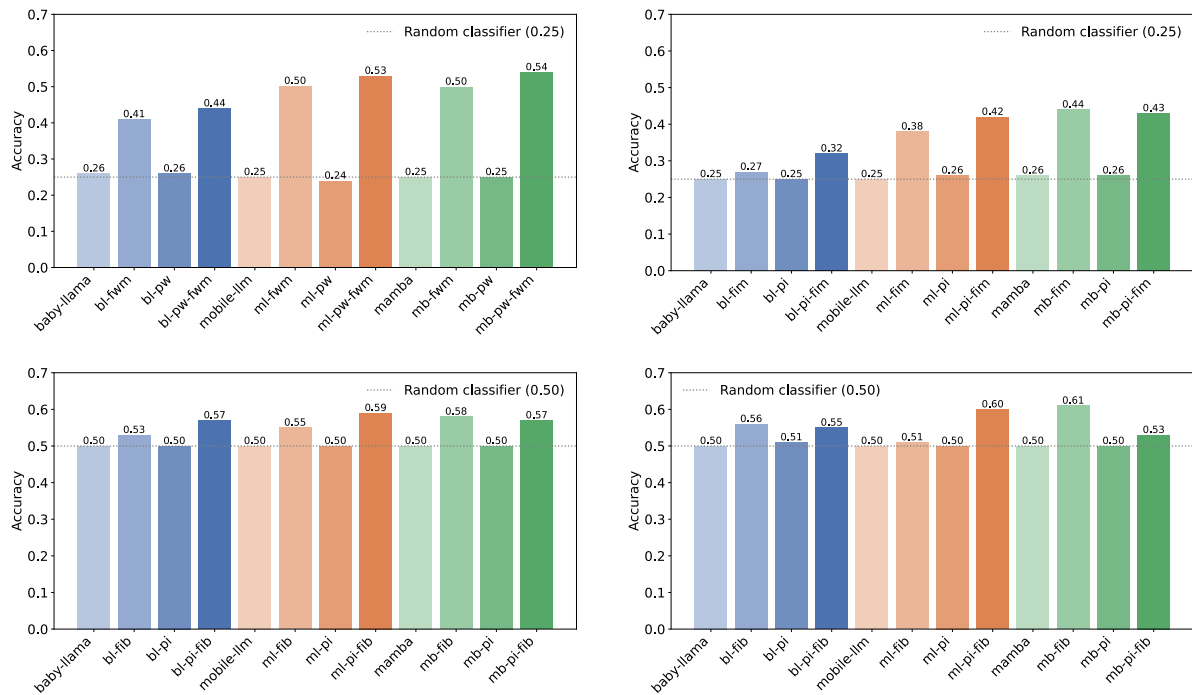


Figure 3.10: Results of the best models of each kind for all the benchmarks. Each family appears in four configurations: the base model (e.g. `baby-llama`), fine-tuned (`-fwm`), continually pretrained (`-pw`), and fine-tuned after continual pretraining (`-pw-fwm`).

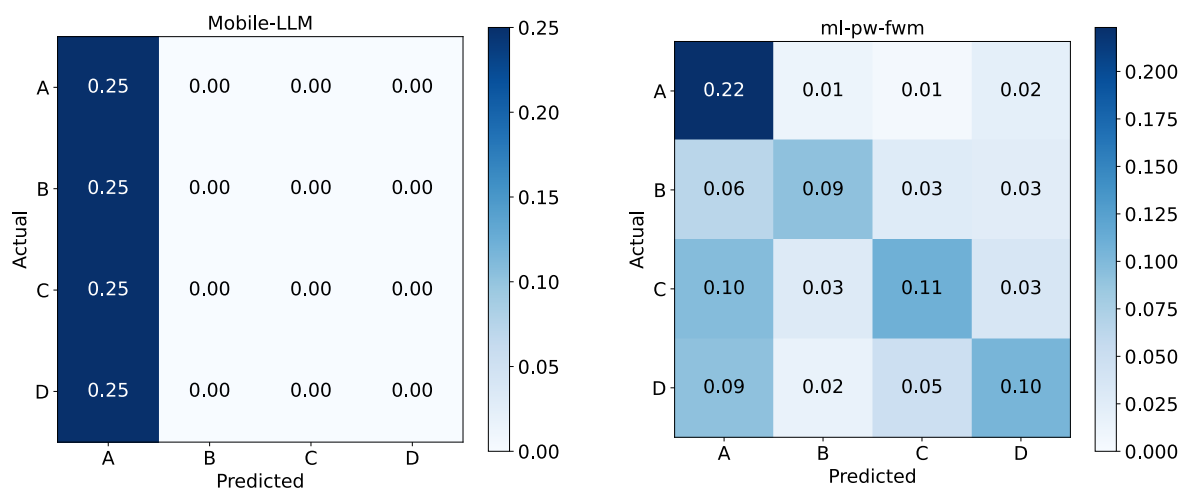


Figure 3.11: Comparison between the confusion matrices of base MobileLLM model and the best fine-tuned MobileLLM model on the wikidoc-multi benchmark.

On the binary benchmarks, by contrast, the performance is much lower and in some cases barely different from random guessing. Except where the validation set happened to be a favorable split, the validation metrics had already suggested that the models would not reach good performance. A further test was carried out with Llama 3 3B Instruct [65], a much larger model that has already been fine-tuned on many tasks, and it reaches the same performance; this suggests that the task is genuinely difficult for these models and may require either more data or, as in BoolQ, a supporting context passage.

4 | BabyMamba

This chapter describes the procedure used to create and train the new BabyMamba model. The aim is to push the Mamba architecture below the smallest size released in the original work and to measure how well such a compact model can be specialized on the benchmarks of this thesis. The chapter first presents the architecture of the model and then the training techniques used to optimize its performance.

4.1. Architecture

BabyMamba is a very small version of the Mamba architecture. In the original paper the authors trained models ranging from 130M to 2.8B parameters, and the 130M model is one of the baselines used in this thesis. The goal here is to go even smaller, in order to test the limits of the architecture and of the training techniques. For this reason BabyMamba has only 50M parameters, roughly the same size as BabyLlama, the other small language model used as a baseline.

This choice enables two complementary comparisons: between two models of the same size but with different architectures (BabyMamba and BabyLlama), and between two models of the same architecture but different sizes (BabyMamba and the 130M Mamba baseline).

The design of such small models must account for the fact that the embedding layers make up a significant fraction of the parameter count. Because this thesis tests these models on specific tasks with a limited vocabulary, the most natural choice would be a custom tokenizer with a small vocabulary, such as the one used in the TinyStories paper [31], which keeps only English-like tokens. This option is ruled out by a competing constraint: the training relies on distillation, and distilling a student from a teacher that uses a different vocabulary is far more complex and may not work well, since the two models would not share the same output space. For this reason BabyMamba adopts the same tokenizer as all the other Mamba models, the GPT-NeoX tokenizer (EleutherAI/gpt-neox-20b on Hugging Face) introduced in the EleutherAI paper [32]. Its vocabulary contains roughly

50k tokens, considerably smaller than the tokenizers used for standard language models such as LLaMA, whose vocabulary reaches about 150k tokens.

BabyMamba is built with the API provided by the Hugging Face Transformers library, starting from the configuration of the 130M Mamba model. The only change is the `hidden_size`, reduced from 768 to 384, which is what brings the parameter count down from 130M to 50M; all other settings, including the number of layers, are left unchanged. Reducing the width rather than the depth is a deliberate choice: it keeps the model deep and thin, in line with the finding of MobileLLM [34] that, at this scale, thin and deep architectures outperform wide and shallow ones.

4.2. Training methods

Training a new language model from scratch is an expensive and time-consuming process. Two main techniques are tested here: pretraining and distillation. Both rely on the Wikipedia dataset described in Section 2.1, a large source of high-quality text and language structure well suited to training the new BabyMamba model.

4.2.1. Pretraining

The pretraining uses the first 900k articles of the Wikipedia dataset, which provide a suitable amount of data for a small model such as BabyMamba. This is still a large dataset for the available computational resources, so the learning rate is first selected through a hyperparameter search on a smaller subset of 200k articles (Figure 4.1), and the pretraining is then run on the full dataset with the selected value.

Selecting the hyperparameters on only 200k articles is admittedly not optimal: a high learning rate, for example, may work well for a short pretraining run yet converge worse than a smaller one over the full dataset. Even so, this is a reasonable compromise between the quality of the results and the available computational resources.

4.2.2. Offline distillation

A classic distillation setup, in which the teacher and the student run in parallel at runtime, was not feasible with the available computational resources. The approach adopted is therefore offline distillation: the soft labels for the whole dataset are generated once with the teacher model, and the student is then trained on these stored labels. This removes the need to keep both models in memory at the same time, at the cost of additional storage to save the soft labels.

The teacher used is Mamba 2.8B, the largest model in the same family as BabyMamba. Choosing a teacher from the same family ensures that teacher and student share the same architecture and, above all, the same tokenizer, which as noted above is what makes the distillation viable in the first place.

Two distillation strategies are considered: top-k knowledge distillation and random sampling knowledge distillation. A student model is trained with each strategy, and both are then evaluated on the downstream benchmarks.

Top-K Knowledge Distillation

In knowledge distillation the teacher supplies, for every input position, a full probability distribution over the vocabulary rather than a single hard label. Storing this distribution in its entirety is impractical: with a vocabulary of about 50k tokens, the soft labels for a corpus of this size would far exceed the available storage. The distribution is, however, very sparse, since for each position only a few tokens carry a non-negligible probability while the long tail is close to zero. Top-k knowledge distillation exploits this property by keeping, for each position, only the k tokens with the highest probability and discarding the rest. These k probabilities are then renormalized to sum to one, and the resulting truncated distribution becomes the target that the student is trained to reproduce.

Here k is set to 40. This value is dictated by the storage budget: the size of the soft labels grows roughly linearly with k, and at $k = 40$ the resulting dataset already occupies around 200GB, close to the limit of the available resources. A larger k would give a more faithful copy of the teacher's distribution but would multiply the storage cost beyond what was feasible.

Random Sampling KD

An alternative is suggested by the paper "Sparse Logit Sampling: Accelerating Knowledge Distillation in LLMs" [66], which studies exactly the offline, cached-logits setting used here and asks how few stored tokens are enough to match full distillation.

The authors show that the top-k strategy, despite being the intuitive choice, is a biased estimator of the teacher distribution, for two reasons. First, keeping only the k most probable tokens and renormalizing them to sum to one inevitably inflates the head of the distribution, so the student learns an up-scaled, over-confident copy of the teacher and ends up poorly calibrated; this is the same renormalization step described above for top-k distillation, and the effect worsens as k shrinks. Second, truncation discards the tail of the distribution, which still carries useful signal: when the ground-truth token happens to

fall in that tail, top-k throws it away entirely and leaves the student with a poor training target.

To remove this bias, the authors replace truncation with random sampling: rather than keeping the k largest probabilities, they draw k tokens at random from the teacher’s output distribution. This gives an unbiased estimate of the teacher distribution while keeping the stored target just as sparse as top-k.

Empirically, these properties translate into a large gap in efficiency. Top-k distillation needs hundreds of stored tokens before it approaches full-distillation performance, and with only a handful of tokens it can perform worse than plain cross-entropy. Random sampling, by contrast, matches full distillation with only about a dozen tokens and outperforms top-k at every storage budget tested. Since the whole motivation for sparse distillation is storage, as discussed above for the 200GB top-k dataset, reaching the same fidelity with fewer tokens translates directly into a smaller footprint.

To allow a fair comparison with the top-k strategy, k is kept at 40 for the random sampling approach as well, rather than at the value the paper reports as optimal.

4.2.3. Continual Pretraining and Fine-tuning

The pretraining and distillation phases produce three different base models: one pre-trained from scratch, one distilled with the top-k strategy, and one distilled with random sampling. At this stage they are in the same position as the off-the-shelf baseline models: having been trained only to predict the next token, they cannot be expected to know the topics of the benchmark datasets, nor to answer questions in the required format.

For this reason they undergo the same two-stage adaptation applied to the baselines in Section 3.2: a continual pretraining phase on the domain text of each benchmark, followed by fine-tuning on the specific task.

4.3. Experiments

This section analyzes the experiments carried out on the BabyMamba model. It first compares the pretraining and distillation phases on their own, and then examines the fine-tuning phase on the downstream tasks.

4.3.1. Pretraining and Distillation

As explained above, the first step is to pretrain BabyMamba in order to identify the best learning rate. Section 4.1 shows the training and validation loss during this phase. Because the model is learning from scratch, a higher learning rate can be used than in continual pretraining, and indeed the run with learning rate 5×10^{-4} reaches the lowest validation loss at the end of the phase. This learning rate is therefore selected for the pretraining on the whole dataset.

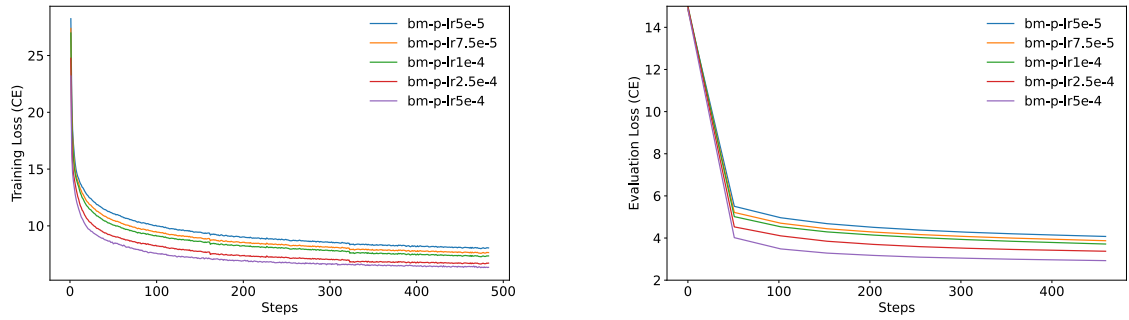


Figure 4.1: Training and validation loss during the pretraining phase of BabyMamba for different learning rates (first 200k articles only).

The distillation phase behaves in the same way. Figure 4.2 shows the training and validation loss for the top-k technique; here the loss is the weighted sum of the distillation loss and the cross-entropy loss, but the pattern is unchanged: the run with learning rate 5×10^{-4} again reaches the lowest validation loss, with no signs of overfitting.

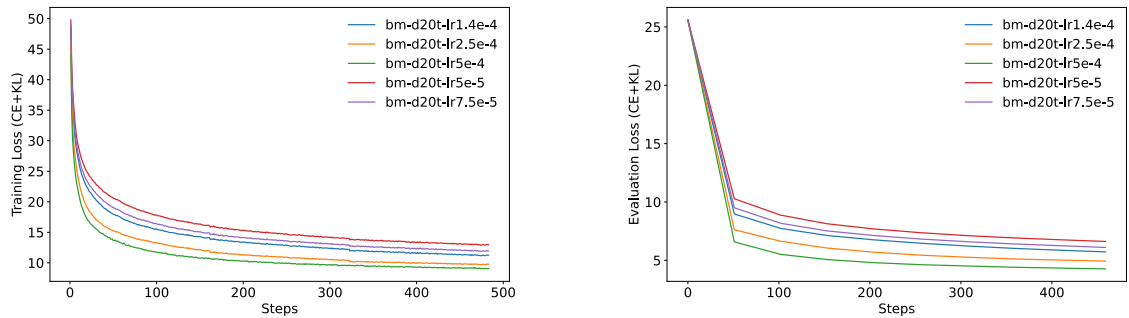


Figure 4.2: Training and validation loss during the distillation phase (topk technique) of BabyMamba for different learning rates (first 200k articles only).

The pretraining and both distillation runs are then repeated with the selected learning rate on the whole dataset. Figure 4.3 compares the resulting cross-entropy loss, and the outcome is surprising in two respects.

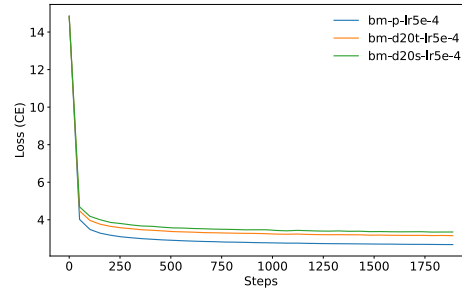


Figure 4.3: Validation cross-entropy loss of the best models of each training technique on the whole dataset (900k articles).

First, the lowest loss is reached by the pretrained model rather than by either distilled model. This is unexpected, since distillation should help the student learn more from the same number of tokens by exposing it to the teacher’s full output distribution instead of a single correct answer. The likely cause is the severity of the truncation: storing only 40 tokens is an extreme reduction of the teacher signal. Indeed, Anshumann et al. [66] place this regime just above the threshold below which top-k and random sampling are no better than plain cross-entropy training, so at this budget the distillation signal is only marginally informative; this is consistent with the pretrained model, trained with plain cross-entropy, coming out ahead here.

Second, the top-k model reaches a lower cross-entropy loss than the random sampling model, the opposite of what Anshumann et al. report. This comparison should not be over-interpreted: cross-entropy loss is only a proxy for language-modeling quality and is not the metric of interest. The quantity that matters is downstream accuracy on the benchmarks, so any conclusion about the two distillation strategies is deferred to those results.

4.3.2. Continual pretraining and Fine-tuning

The fine-tuning phase follows the same procedure described in Section 3.1: a continual pretraining phase on the domain dataset, followed by fine-tuning on the specific task.

Figure 4.4 shows the validation loss during continual pretraining for the best model of each type (pretrained, distilled with top-k, and distilled with random sampling) on the two text datasets. As expected from the previous phase, the pretrained model starts from the lowest loss, but already after the first epoch both distilled models drop below it. All three reach their minimum loss between the fourth and fifth epoch, notably later than what was seen in Section 3.2 for the baselines, where the loss began to rise after the second

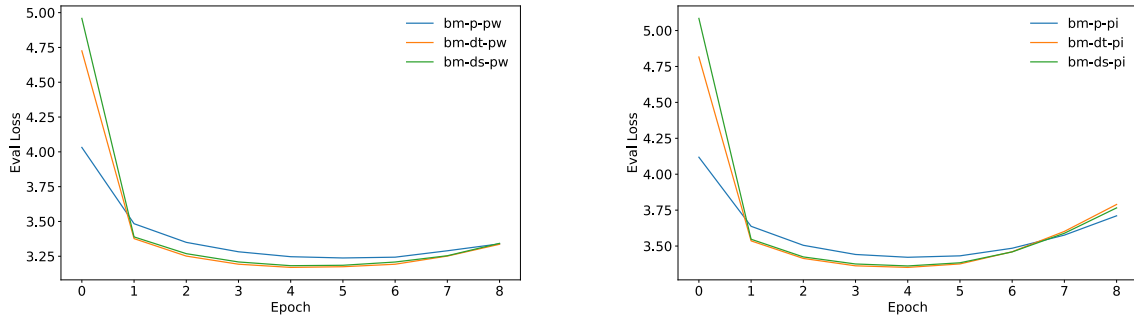


Figure 4.4: Validation loss of the best model of the continual pretraining phase for each distillation and pretraining techniques on the two text datasets.

epoch.

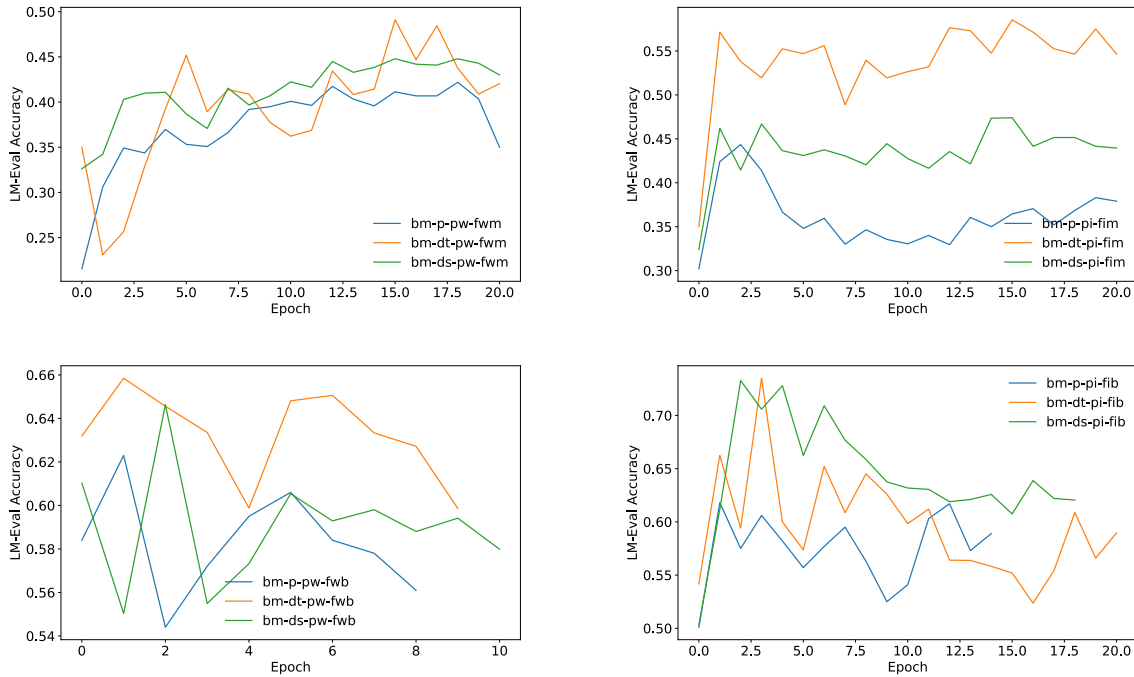


Figure 4.5: `lm-eval` accuracy of the best model of the fine-tuning phase for each distillation and pretraining techniques on the two datasets.

Figure 4.5 collects the `lm-eval` accuracy of the best fine-tuned model for each training technique, on the two benchmark types and the two datasets.

As before, the binary datasets are harder than the multiple-choice ones. Performance on them is low and unstable: the gap over the random classifier is very small, and the curves are scattered, without a clear trend.

On the multiple-choice types the picture differs by dataset. For WikiDoc there is no clear winner, as all models perform similarly; for Investopedia the top-k distillation technique clearly outperforms the others, although every model still beats the random classifier.

These benchmarks are known to have high variance, but the Investopedia multiple-choice result is still notable, since the reference paper [66] reports the opposite ordering on downstream tasks, with random sampling outperforming top-k.

4.4. Results

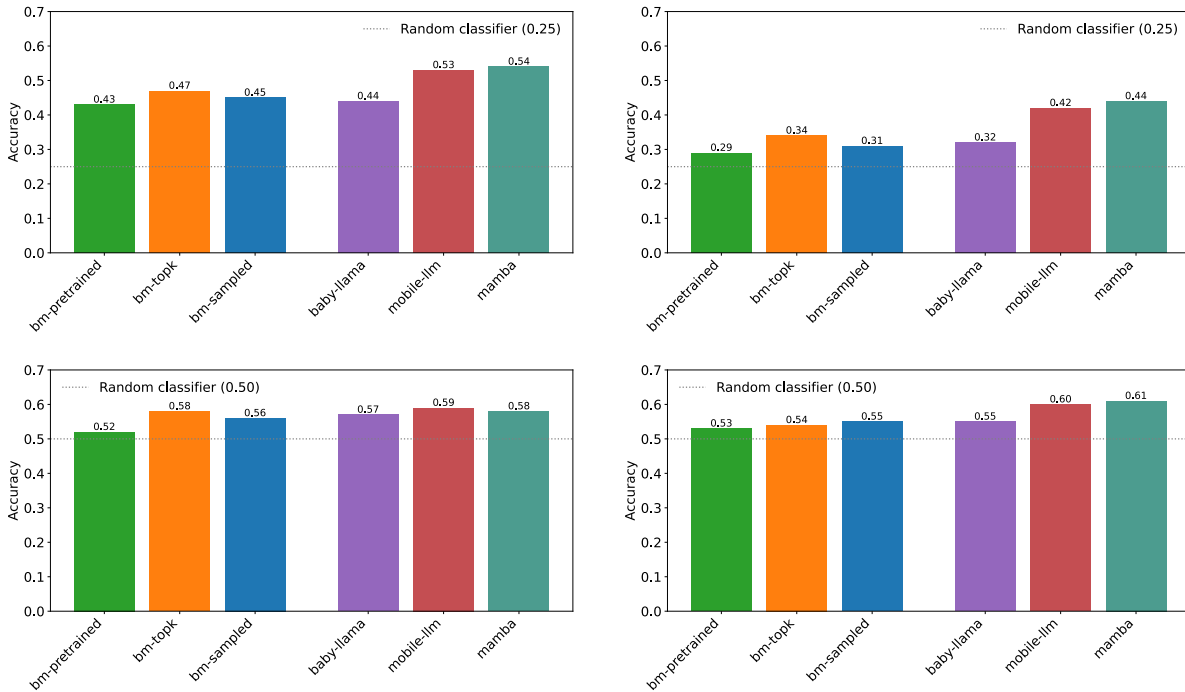


Figure 4.6: Comparison of the best BabyMamba models with the baselines on all the benchmarks.

Figure 4.6 compares the best BabyMamba models with the baselines on all the benchmarks. On the binary benchmarks performance is very low, as expected, which makes any conclusion difficult; on the wikidoc-binary benchmark, for instance, the best BabyMamba model matches the best Mamba model, which is twice its size.

On the multiple-choice benchmarks the picture is clearer: BabyMamba sits in the same class as the best BabyLlama models, which share its size but use a different architecture, while the larger Mamba and MobileLLM models retain a clear advantage.

5 | Conclusion

This thesis set out to follow a frontier opposite to the one that has shaped most of language modeling. The dominant story is one of growth, from GPT-2 to contemporary frontier systems built from ever more parameters, data, and compute and served from large data centers. This work asked the reverse question: how little has to be given up as a model is scaled down, until it is small enough to run not in a data center but on an edge device such as a wearable, a sensor, or a battery-powered board.

Pursuing this direction meant first looking at the hardware these models must run on. Below a certain size, memory and energy matter more than raw capability, and the attention mechanism that made the Transformer successful is structurally ill-suited to the small accelerators found at the edge. This motivated the turn toward alternative architectures with sub-quadratic complexity, among which Mamba was chosen for its competitive performance and its mature, widely supported implementations.

A second idea guided the experimental part of the thesis. A model with few parameters cannot hold broad world knowledge, so it has to specialize on a narrow domain instead. Yet small models are usually evaluated on general-purpose benchmarks that measure exactly the wide knowledge they cannot have (Section 1.4). To correct this mismatch, two narrow domains were chosen, medicine and finance, and a dedicated question-answering benchmark was built for each from the WikiDoc and Investopedia datasets. The free-text answers of the original data were unsuitable for automatic and reproducible scoring, so they were converted with GPT-5 into two controlled formats: a balanced yes/no benchmark inspired by BoolQ and a multiple-choice benchmark inspired by MMLU (Section 2.2).

With the benchmarks in place, the existing small models, BabyLlama, MobileLLM, and the 130M Mamba, were adapted to them. Two strategies were compared. The first is direct fine-tuning, in which the loss is computed only on the answer token, so that the model learns the required format without losing the knowledge acquired during pretraining. The second is a two-stage procedure in which a continual pretraining phase on the domain text precedes the fine-tuning (Section 3.2). The continual pretraining variant

proved consistently, if modestly, better, which suggests that the model does absorb part of the domain knowledge from the unlabeled text. Across both strategies the multiple-choice benchmarks were learned well, with accuracies clearly above the random classifier, whereas the binary benchmarks remained hard and unstable. The same difficulty appeared even for a much larger model, Llama 3 3B Instruct, which indicates that the task itself is genuinely demanding when no supporting passage is given.

The central contribution of the thesis is BabyMamba, a Mamba model of 50M parameters trained from scratch. Its size was chosen deliberately to sit below the smallest model of the original Mamba work and to match BabyLlama, which makes it possible to compare two models of the same size but different architecture and two models of the same architecture but different size. BabyMamba was produced with three training techniques, namely pretraining from scratch and offline distillation from a Mamba 2.8B teacher with two strategies, top-k and random sampling. It was then adapted to the benchmarks with the same two-stage procedure used for the baselines.

Several of the results proved informative precisely because they ran against expectation. Distillation, which should let a student learn more from the same data, did not improve on plain pretraining. The most likely reason is the severity of the truncation imposed by the storage budget, which kept only 40 tokens of the teacher distribution and left the distillation signal only marginally informative. For the same reason the comparison between top-k and random sampling did not reproduce the ordering reported in the literature. On the downstream tasks the picture was more encouraging. On the multiple-choice benchmarks BabyMamba reached the same class as BabyLlama, a model of equal size but different architecture, while the larger Mamba and MobileLLM models kept a clear advantage, and on the wikidoc-binary benchmark the best BabyMamba model matched the 130M Mamba, a model twice its size. The Mamba architecture can therefore be pushed well below the smallest size of the original work and still be specialized on a narrow domain, which is precisely the regime that matters for deployment on constrained hardware.

5.1. Limitations and Future Developments

Each limitation of this work points to a concrete way to extend it.

The most important of these concerns the data. The performance a small model can reach is bounded above all by the quantity and quality of the reliable, domain-specific text available to it, and sources such as WikiDoc and Investopedia, however valuable, remain limited in size. Data scarcity is not a new obstacle. In the early days of machine

learning it was one of the main factors that slowed the development of new models, and although general-purpose data is now abundant, public domain-specific datasets remain rare. This makes data the clearest lever for improvement: assembling larger and cleaner domain corpora, or narrowing the domains so that a fixed amount of text covers them more densely, would give specialization considerably more room to act. A related opportunity lies in the way the tasks are framed. The binary benchmarks stayed difficult for every model, including a much larger one, which points to a promising experiment rather than a dead end: supplying a short context passage in the manner of BoolQ would make these tasks tractable and, at the same time, separate what a model can read and reason about from what it has had to memorize.

A second family of directions concerns the training of BabyMamba. The distillation experiments stored 40 tokens of the teacher distribution, which already sits near the lower edge of the regime in which distillation is informative. Keeping a larger share of that distribution is the natural next step, since it is precisely the regime in which random sampling is expected to overtake top-k and in which distillation should begin to improve on plain pretraining, so revisiting the two strategies with a richer teacher signal is likely to be the single most informative experiment to run.

A complementary direction is to relax the constraint that ties the student to the teacher’s vocabulary. Freeing the student to use a smaller, domain-specific tokenizer, would let a larger share of the limited parameter budget go to the body of the model rather than to its embeddings, which is especially valuable at this scale.

The natural endpoint of the whole line of work, however, is deployment. Porting a specialized model such as BabyMamba onto the embedded GPUs and microcontrollers described at the start of the thesis, and measuring its accuracy, latency, and energy use on real hardware, would close the loop between the motivation that opened this work and the model that resulted from it.

Bibliography

- [1] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL <https://arxiv.org/abs/2001.08361>.
- [2] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22*, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.
- [3] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9, 2023. doi: 10.1109/HPEC58863.2023.10363447.
- [4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [5] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. URL <https://arxiv.org/abs/2312.00752>.
- [6] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August

2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL <https://aclanthology.org/P16-1162/>.
- [7] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd edition, 2026. URL <https://web.stanford.edu/~jurafsky/slp3/>. Online manuscript released January 6, 2026.
- [8] Robin M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview, 2019. URL <https://arxiv.org/abs/1912.05911>.
- [9] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. doi: https://doi.org/10.1207/s15516709cog1402_1. URL https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1.
- [10] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.
- [11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016. URL <https://arxiv.org/abs/1409.0473>.
- [12] Elena Voita. NLP Course For You, Sep 2020. URL https://lena-voita.github.io/nlp_course.html.
- [13] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.
- [14] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf.
- [15] Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and

- Pablo Villalobos. Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022. doi: 10.1109/IJCNN55064.2022.9891914.
- [16] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S. Vetter. Nvidia tensor core programmability, performance & precision. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 522–531, 2018. doi: 10.1109/IPDPSW.2018.00091.
- [17] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 03 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.1.79. URL <https://doi.org/10.1162/neco.1991.3.1.79>.
- [18] DeepSeek-AI. Deepseek-v3 technical report, 2024. URL <https://arxiv.org/abs/2412.19437>.
- [19] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL <http://jmlr.org/papers/v23/21-0998.html>.
- [20] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022. URL <https://arxiv.org/abs/2205.01068>.
- [21] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL <https://arxiv.org/abs/2302.13971>.
- [22] OpenAI. Gpt-5 system card. System card, OpenAI, 8 2026. URL <https://cdn.openai.com/gpt-5-system-card.pdf>.
- [23] NVIDIA Corporation. Jetson Orin Nano series modules data sheet (ds-11105-001). Technical datasheet, 2023. URL <https://developer.nvidia.com/downloads/jetson-orin-nx-orin-nano-series-thermal-design-guide>.
- [24] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently

- scaling transformer inference. In D. Song, M. Carbin, and T. Chen, editors, *Proceedings of Machine Learning and Systems*, volume 5, pages 606–624. Curan, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/file/c4be71ab8d24cdfb45e3d06dbfca2780-Paper-mlsys2023.pdf.
- [25] STMicroelectronics. STM32N6 series: The first STM32 MCU with ST Neural-ART accelerator NPU. Product page, 2025. URL <https://www.st.com/en/microcontrollers-microprocessors/stm32n6-series.html>.
- [26] GreenWaves Technologies. GAP9: Ultra-low-power parallel RISC-V processor for the edge. Product brief, 2023. URL https://greenwaves-technologies.com/gap9_processor/.
- [27] GreenWaves and ETH Zurich (PULP Platform). GAP9Shield: A 150GOPS AI-capable ultra-low power module for vision and ranging applications on nano-drones. arXiv preprint arXiv:2407.13706, 2024. URL <https://arxiv.org/abs/2407.13706>.
- [28] Daliang Xu, Hao Zhang, Liming Yang, Ruiqi Liu, Gang Huang, Mengwei Xu, and Xuanzhe Liu. Fast on-device llm inference with npus. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, ASPLOS ’25, pages 445–462, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400706981. doi: 10.1145/3669940.3707239. URL <https://doi.org/10.1145/3669940.3707239>.
- [29] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. volume 105, pages 2295–2329, 2017. doi: 10.1109/JPROC.2017.2761740.
- [30] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.
- [31] Ronen Eldan and Yuanzhi Li. Tinstories: How small can language models be and still speak coherent english?, 2023. URL <https://arxiv.org/abs/2305.07759>.
- [32] Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-NeoX-20B: An open-source autoregressive

- language model. In Angela Fan, Suzana Ilic, Thomas Wolf, and Matthias Gallé, editors, *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bigscience-1.9. URL <https://aclanthology.org/2022.bigscience-1.9/>.
- [33] Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. URL <https://doi.org/10.5281/zenodo.5297715>.
- [34] Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. MobileLLM: Optimizing sub-billion parameter language models for on-device use cases. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32431–32454. PMLR, 21–27 Jul 2024. URL <https://proceedings.mlr.press/v235/liu24ce.html>.
- [35] Noam Shazeer. Glue variants improve transformer, 2020. URL <https://arxiv.org/abs/2002.05202>.
- [36] Inar Timiryasov and Jean-Loup Tastet. Baby llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty. In Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell, editors, *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 279–289, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.conll-babylm.24. URL <https://aclanthology.org/2023.conll-babylm.24/>.
- [37] Alex Warstadt, Leshem Choshen, Aaron Mueller, Adina Williams, Ethan Wilcox, and Chengxu Zhuang. Call for papers – the babylm challenge: Sample-efficient pretraining on a developmentally plausible corpus, 2023. URL <https://arxiv.org/abs/2301.11796>.
- [38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learn-

- ing with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL <http://jmlr.org/papers/v21/20-074.html>.
- [39] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR, 13–18 Jul 2020. URL <https://proceedings.mlr.press/v119/katharopoulos20a.html>.
- [40] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9355–9366. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/schlag21a.html>.
- [41] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 56501–56523. PMLR, 21–27 Jul 2024. URL <https://proceedings.mlr.press/v235/yang24ab.html>.
- [42] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations (ICLR)*, 2022.
- [43] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 107547–107603. Curran Associates, Inc., 2024. doi: 10.52202/079017-3417. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c2ce2f2701c10a2b2f2ea0bfa43cfaa3-Paper-Conference.pdf.
- [44] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.
- [45] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn

- Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021. URL <https://arxiv.org/abs/2009.03300>.
- [46] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hel-laSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL <https://aclanthology.org/P19-1472/>.
- [47] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, pages 7432–7439, 2020. URL <https://arxiv.org/abs/1911.11641>.
- [48] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL <https://aclanthology.org/N19-1300/>.
- [49] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, pages 8732–8740, 2020. URL <https://arxiv.org/abs/1907.10641>.
- [50] Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics (TACL)*, 8:377–392, 2020. doi: 10.1162/tacl_a_00321. URL <https://arxiv.org/abs/1912.00582>.
- [51] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786, 2025. doi: 10.1109/TASLPRO.2025.3606231.

- [52] Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and scalable strategies to continually pre-train large language models, 2024. URL <https://arxiv.org/abs/2403.08763>.
- [53] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL <https://arxiv.org/abs/1503.02531>.
- [54] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL <https://zenodo.org/records/12608602>.
- [55] What are small language models (slms)? URL <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-are-small-language-models>.
- [56] Rina Diane Caballar and Cole Stryker. What are small language models? URL <https://www.ibm.com/think/topics/small-language-models>.
- [57] John Johnson. Small language models (slm): A comprehensive overview. URL <https://huggingface.co/blog/jjokah/small-language-model>.
- [58] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL <https://aclanthology.org/2020.acl-main.747/>.
- [59] medalpaca. Medical meadow wikidoc. https://huggingface.co/datasets/medalpaca/medical_meadow_wikidoc, 2023. Accessed: 2026-05-18.
- [60] WikiDoc Contributors. Wikidoc: The open-source medical textbook. URL https://www.wikidoc.org/index.php/Main_Page.
- [61] FinLang. Investopedia instruction tuning dataset. <https://huggingface.co/datasets/FinLang/investopedia-instruction-tuning-dataset>, 2024. Accessed: 2026-05-18.

- [62] Investopedia Contributors. Investopedia: Sharper insight. smarter investing. URL <https://www.investopedia.com/>.
- [63] Wikimedia Foundation. Wikipedia dataset: 20231101.en. <https://huggingface.co/datasets/wikimedia/wikipedia/viewer/20231101.en>, 2023. English Wikipedia dump processed for Hugging Face datasets. Accessed: 2026-05-18.
- [64] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 55006–55021. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ac662d74829e4407ce1d126477f4a03a-Paper-Conference.pdf.
- [65] Llama Team, AI @ Meta. The llama 3 herd of models. 2024. URL <https://arxiv.org/abs/2407.21783>.
- [66] Anshumann, Mohd Abbas Zaidi, Akhil Kedia, Jinwoo Ahn, Taehwak Kwon, Kangwook Lee, Haejun Lee, and Joohyung Lee. Sparse logit sampling: Accelerating knowledge distillation in LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18085–18108, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.885. URL <https://aclanthology.org/2025.acl-long.885/>.

List of Figures

1.1	Self-attention mechanism diagram (inspired by the work of Elena Voita [12]).	4
1.2	Example of a Mixture-of-Experts architecture, in which only a subset of the experts is activated to process a given token.	6
1.3	Selective state-space model block diagram.	13
1.4	The language model lifecycle, from initialization to evaluation.	16
3.1	Train and validation loss for different learning rates during the fine-tuning phase of MobileLLM on the wikidoc-multi dataset.	28
3.2	<code>lm-eval</code> and raw accuracy for different learning rates during the fine-tuning phase of MobileLLM on the wikidoc-multi dataset.	29
3.3	Sum of the probability assigned to each of the possible answers during the fine-tuning phase for different learning rates of MobileLLM on the wikidoc-multi dataset.	29
3.4	<code>lm-eval</code> accuracy of the best models of each kind for all the datasets. . . .	30
3.5	Sum of the probability assigned to each of the possible answers during the fine-tuning phase for different learning rates of Mamba on the wikidoc-multi dataset.	31
3.6	BabyLlama performance on the investopedia-binary dataset with different splits of the data as validation set during the fine-tuning phase.	31
3.7	Train and validation loss for different learning rates during the continual pretraining phase of BabyLlama on the wikidoc-text dataset.	32
3.8	Validation loss of the best models of all kinds and datasets during the continual pretraining phase.	33
3.9	<code>lm-eval</code> accuracy of the best models of each kind for all the datasets in the fine-tuning phase after continual pretraining.	34
3.10	Results of the best models of each kind for all the benchmarks. Each family appears in four configurations: the base model (e.g. <code>baby-llama</code>), fine-tuned (<code>-fwm</code>), continually pretrained (<code>-pw</code>), and fine-tuned after continual pretraining (<code>-pw-fwm</code>).	35

3.11	Comparison between the confusion matrices of base MobileLLM model and the best fine-tuned MobileLLM model on the wikidoc-multi benchmark. . .	35
4.1	Training and validation loss during the pretraining phase of BabyMamba for different learning rates (first 200k articles only).	41
4.2	Training and validation loss during the distillation phase (topk technique) of BabyMamba for different learning rates (first 200k articles only). . . .	41
4.3	Validation cross-entropy loss of the best models of each training technique on the whole dataset (900k articles).	42
4.4	Validation loss of the best model of the continual pretraining phase for each distillation and pretraining techniques on the two text datasets.	43
4.5	lm-eval accuracy of the best model of the fine-tuning phase for each distillation and pretraining techniques on the two datasets.	43
4.6	Comparison of the best BabyMamba models with the baselines on all the benchmarks.	44