university of groningen / faculty of science and engineering / artificial intelligence

Exploring the Behavior of Outlier Channels in Hybrid-LLM Architectures

Bachelor's Project Artificial Intelligence

Davide Zani, s5054702, d.zani@student.rug.nl,
Supervisors: Steven Abreu & Prof. Herbert Jaeger

Abstract: Transformer-based Large Language Models excel through self-attention mechanisms that capture complex dependencies across sequences, but their quadratic computational scaling is inefficient. State Space Models (SSMs) offer an alternative approach with linear scaling, yet they are limited by their fixed-size latent state, which limits their memory abilities. Hybrid architectures like RecurrentGemma emerge as a promising alternative, combining the efficiency of SSMs with the performance of attention. All architectures are faced with a common challenge for quantization: outlier channels. While these have been studied in pure transformer and SSM architectures, their distribution and impact in hybrid models remains underexplored. This study investigates outlier channels in RecurrentGemma-2B using statistical analysis and controlled clipping experiments across six benchmarks. We identified that most outlier channels are concentrated in MLP blocks and deeper layers, with late layers containing 44.4% more outliers than early layers. Clipping experiments revealed that aggressive outlier suppression caused severe performance degradation, while conservative thresholds maintained near-baseline performance. Mathematical reasoning tasks showed greatest sensitivity to outlier removal, whereas commonsense reasoning tasks were more robust. Results demonstrate that outlier channels are essential for model functionality.

1 Introduction

1.1 Background

Transformer-based architectures have been leading the language-modeling revolution, dominating the space due to the effectiveness of the self-attention mechanism (Vaswani et al., 2023). However, this mechanism comes at a cost, specifically in terms of computation. The computational cost of the self-attention mechanism scales quadratically, $O(n^2)$, with the sequence length n, which severely limits their scalability for applications with extended input sequences (H. Ren et al., 2021).

State Space Models (SSMs), such as Mamba (Gu & Dao, 2023), RWKV (Peng et al., 2023), and HGRN (Qin et al., 2024), have recently gained traction, presenting an alternative architecture whose computational cost scales linearly with sequence length. This makes them a very appealing choice for large-scale sequence data, given that they could theoretically support very long context windows (Gu & Dao, 2023). However, the potential of SSMs has not yet achieved the competitive performance seen in attention-based models (Merrill et al., 2024).

For this reason, to make up for the shortcomings of SSMs, whilst taking advantage of their strengths, Hybrid-LLM architectures like RecurrentGemma (Botev et al., 2024), Jamba (Lieber et al., 2024), and Samba (L. Ren et al., 2024a) have emerged. These combine the recurrent components from SSMs with the attention mechanism of Transformers, allowing for improved computational efficiency while minimizing representational power trade-offs.

This has allowed Hybrid-LLM architectures to achieve competitive performance with superior efficiency compared to either model type alone, performing particularly well in throughput and memory efficiency for long sequences (Lieber et al., 2024), making them a promising solution for complex sequence-modeling tasks.

RecurrentGemma, based on the Griffin architecture, (De et al., 2024a) demonstrates this approach by integrating recurrent blocks with traditional multi-head attention layers (Figure 1.1).

The architecture alternates between Griffin blocks, which provide efficient linear-scaling sequence processing through their recurrent state space formulation, and attention layers, which maintain the model's ability to capture complex long-range dependencies and contextual relationships.

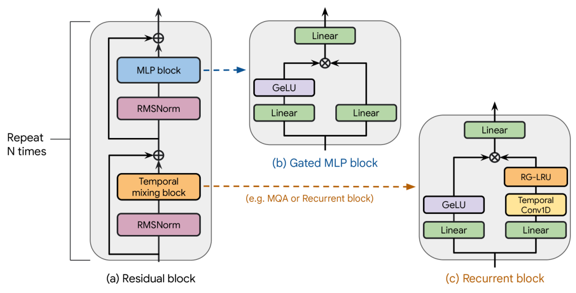


Figure 1.1: RecurrentGemma (Griffin) architecture showing the alternating pattern of recurrent blocks and local attention layers

1.2 Outlier Channels

Common to both transformer and SSM foundation models are outlier channels—neurons whose activation magnitudes significantly exceed the average over a neural network's (NN) width. Research shows that these outlier channels emerge early during training and appear most frequently in layers that process the residual stream (Nrusimha et al., 2024).

The underlying cause involves signal propagation dynamics: when input correlations grow large during training and approach rank collapse, the mathematical relationship between input and feature correlations forces activation variance to concentrate in fewer channels, increasing feature-wise kurtosis. This problem worsens with network depth, as activation distributions become increasingly heavy-tailed in deeper layers, making extreme outlier values more probable. The optimization process further accelerates outlier formation through adaptive learning rates in optimizers like Adam, where subleading order terms in gradient updates that are quadratic in the learning rate consistently drive increases in activation kurtosis, which explains why larger learning rates and higher optimizer adaptivity favor outlier development (He et al., 2024).

Outlier channels have also been proven to be essential for the correct functioning of transformer-based architectures (Liu et al., 2023) and SSM architectures (Pierro & Abreu, 2024). Formally, for a layer's activations $\mathbf{a} \in \mathbb{R}^d$, a channel i is considered an outlier if:

$$|a_i| > \mu + \sigma \cdot \tau$$

where a_i is the activation value in channel i, $\mu = \frac{1}{d}\sum_{i=1}^{d} a_i$ is the mean activation across all channels, $\sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(a_i - \mu)^2}$ is the standard deviation, and τ is a scaling factor that determines the threshold for identifying outliers, typically set to $\tau = 6$ (Pierro & Abreu, 2024), (Liu et al., 2023). We used the standard deviation approach described above as it has been used in previous studies in both transformer (Bondarenko et al., 2023) and SSM architectures (Pierro & Abreu, 2024), thereby facilitating direct comparisons between model architectures. Although alternative methods for identifying outlier channels exist in the literature, such as using activation kurtosis measures (Nrusimha et al., 2024), percentile-based thresholding (Dettmers et al., 2022), or frequency-based correlation analyses (Puccetti et al., 2022), we chose the standard deviation method for consistency with prior work.

The importance of outliers is relevant for both model understanding and model performance optimization, particularly for quantization strategies where extreme activations can significantly impact precision and efficiency.

From an interpretability perspective, because of their larger dynamic range, which seemingly allows them to capture complex patterns and variations in data compared to the other channels, these outlier channels could reveal important insights into how neural networks encode and process information.

1.3 Quantization Challenges

Understanding outlier channels is relevant for the quantization of models. Quantization has become fundamental for optimizing machine learning models, especially when using resource-limited hardware (Li et al., 2024). By reducing the precision of model parameters from 32-bit floating-point values to lower-precision representations (such as 8-bit integers), quantization helps decrease memory usage, as well as speed up computation.

Let us consider a tensor $\mathbf{X} \in \mathbb{R}^{B\times T\times C}$ containing scalar values $x_{i,j,k} \in \mathbb{R}$. . For uniform symmetric quantization to b-bit integers, the quantization function $Q_b : \mathbb{R} \to \mathbb{Z}_b$ maps each real value x to an integer value in the range $[-2^{b-1}, 2^{b-1}-1]$ as follows:

$$Q_b(x) = \text{clip}\left(\lfloor s_{\mathbf{X}} \cdot x + 0.5 \rfloor, -2^{b-1}, 2^{b-1}-1\right)$$

where $\lfloor \cdot \rfloor$ denotes the floor operation (truncation), $\text{clip}(z, a, b) = \max(a, \min(z, b))$ constrains the value to the representable range, and $s_{\mathbf{X}}$ is the scaling factor.

The dequantization function $Q_b^{-1} : \mathbb{Z}_b \to \mathbb{R}$ reconstructs an approximation $\hat{x}$ of the original value:

$$Q_b^{-1}(z) = \hat{x} = \frac{z}{s_{\mathbf{X}}}$$

One of the main challenges when quantizing models is the presence of outlier channels. For channel-wise quantization, where each channel $c \in \{1, \ldots, C\}$ is quantized independently, the scaling factor becomes:

$$s_c = \frac{2^{b-1}-1}{\max_{i,j} |x_{i,j,c}|}$$

When an outlier channel contains values with magnitudes significantly larger than typical channels, it leads to two distinct problems. For tensor-wise quantization, the large dynamic range forces a small scaling factor:

$$s_{\mathbf{X}} = \frac{2^{b-1}-1}{\max_{i,j,k} |x_{i,j,k}|}$$

which causes most non-outlier values to be quantized to a narrow range of integers, thereby reducing their precision. For channel-wise quantization, the disparate scaling factors between channels can lead to computational inefficiencies, as different multipliers must be applied per channel.

The quantization error $\epsilon_x = \hat{x} - x$ is directly affected by the scaling factor. Smaller scaling factors (resulting from the outliers) produce larger errors across the distribution of values.

1.4 Research Gap and Questions

Efforts in handling outlier channels have mainly been focused on transformer-based Large Language Models (LLMs), where various techniques have been developed to try and circumvent and/or mitigate the drawbacks induced by outliers on quantization. An example of this is found in the paper by Wei et al. (2023), where a method to shift and scale outlier channels makes them more amenable to low-bit quantization without sacrificing performance. Another example is the paper by Heo et al. (2023), where the reconfiguration of the channel dimensions to isolate outliers and quantize them separately improves quantization efficiency and reduces information loss.

In State Space Models, the paper by Pierro & Abreu (2024) looks into the presence of outlier channels in recurrent architectures, identifying patterns in outliers similar to those found in transformer models. Just like in transformer-based architectures, the paper indicates that the difficulty in quantizing SSMs stems from the activation outliers, which disrupt the precision of remaining channels and affect post-training quantization.

However, the behavior of outlier channels in hybrid architectures that combine both attention and recurrent mechanisms remains unexplored. RecurrentGemma (Botev et al., 2024) represents one such hybrid model, integrating Griffin's recurrent blocks (De et al., 2024b) with traditional attention layers to take advantage of the linear scaling properties of SSMs while maintaining the representational power of transformers. This new architectural combination raises questions about how outlier patterns show across different component types, and whether existing outlier management strategies apply to hybrid systems.

This gap in the literature motivates our research questions:

1. Do outlier channels exist in the Hybrid-LLM model RecurrentGemma2B?

2. Does the removal of these outlier channels affect overall model performance?

In this Bachelor's Project, we successfully identify outlier channels in RecurrentGemma2B, and analyze their impact on model performance.

2 Methodology

Our methodology consists of two main phases: (1) collecting activation statistics to identify the location and extent of outlier channels, and (2) evaluating the impact of outlier clipping on model performance across a variety of benchmarks.

2.1 Model and Dataset Selection

For our experiments, we used the RecurrentGemma-2B model, accessed via the Hugging Face Transformers library (Wolf et al., 2020). This model was chosen because of its small size and relatively good general performance (Botev et al., 2024). We ran inference on the model with the WikiText-2 dataset (Merity et al., 2016) to collect activation statistics. We used 80% of the WikiText-2 dataset, in order to use sufficient data for reliable statistical collection while lowering computational costs. For activation monitoring, we deliberately chose PyTorch hooks over recorders. This was because recorders typically store complete copies of activation tensors for later analysis. Hooks, on the other hand, can compute statistics on-the-fly and discard raw values after processing, which reduces memory requirements. This was particularly important when monitoring numerous activation points across many layers.

2.2 Activation Statistics Collection

We positioned PyTorch Hooks at 26 distinct locations within each layer of the model. These recording points included:

- Layer inputs and outputs
- Normalization layers (temporal and channel pre-normalization, both inputs and outputs)
- Attention block components (query/key/value projection outputs, o-projection input, and final output)
- Recurrent block components (linear_x/linear_y outputs, conv1d output, RG-LRU output, linear_out input/output)
- MLP block components (gate/up projection outputs, down projection input, and final output)
- Final normalization layer (input and output)

This allowed us to analyze activation patterns throughout the entire network architecture, and to observe how values propagate and transform through each computational step.

2.3 Online Statistics Calculation

We implemented Welford's online algorithm (Welford, 1962) for computing running statistics with memory-efficient streaming updates. For each activation tensor at recording point p, we maintained both per-channel and layer-level aggregate statistics.

For each channel c, we tracked running statistics using:

$$\mu_{n,c} = \mu_{n-1,c} + \frac{x_{n,c} - \mu_{n-1,c}}{n} \tag{2.1}$$

$$M_{2,n,c} = M_{2,n-1,c} + (x_{n,c} - \mu_{n-1,c})(x_{n,c} - \mu_{n,c}) \tag{2.2}$$

where $\mu_{n,c}$ is the running mean and $M_{2,n,c}$ is used to compute variance. The standard deviation σ_c is then:

$$\sigma_c = \sqrt{\frac{M_{2,n,c}}{n-1}} \tag{2.3}$$

Additionally, we tracked minimum and maximum values per channel:

- $\min_c = \min\{x_{1,c}, x_{2,c}, \ldots, x_{n,c}\}$
- $\max_c = \max\{x_{1,c}, x_{2,c}, \ldots, x_{n,c}\}$

We also computed layer-level aggregate statistics across all channels to understand the overall activation distribution at each recording point.

2.4 Outlier Channel Identification

Our outlier detection employed a statistical approach based on the distribution of channel means. For each recording point, we first calculated the grand mean of all channel means as:

$$\mu_{\text{grand}} = \frac{1}{C}\sum_{c=1}^{C} \mu_c$$

which provides a central reference point for the overall activation distribution. We then computed the standard deviation of channel means using:

$$\sigma_{\text{grand}} = \sqrt{\frac{1}{C-1}\sum_{c=1}^{C}(\mu_c - \mu_{\text{grand}})^2}$$

to quantify the variability across channels.

Using these statistical measures, we identified outlier channels by determining where individual channel means deviated significantly from the grand mean. Specifically, a channel c is classified as an outlier when its mean satisfies the condition:

$$\text{Channel } c \text{ is an outlier} \iff$$

$$|\mu_c - \mu_{\text{grand}}| > k_{\text{outlier}} \sigma_{\text{grand}}$$

where k_{outlier} is a configurable threshold factor that controls the sensitivity to outlier detection. This approach ensures that only channels exhibiting statistically significant deviations from the typical activation pattern are targeted for clipping intervention.

3 Clipping System

Our clipping method targets outlier channels while preserving normal channels intact. We use k_{clip} to set the upper and lower bounds that constrain the activation values.

For an activation tensor $\mathbf{X} \in \mathbb{R}^{B\times T\times C}$ where B denotes batch size, T sequence length, and C channels, we first define channel-specific clipping bounds $\mathbf{L}, \mathbf{U} \in \mathbb{R}^C$ where:

$$L_c = \mu_c - k_{\text{clip}}\sigma_c \quad \text{and} \quad U_c = \mu_c + k_{\text{clip}}\sigma_c \tag{3.1}$$

The clipping operation $\Phi : \mathbb{R}^{B\times T\times C} \to \mathbb{R}^{B\times T\times C}$ is then defined as:

$$\Phi(\mathbf{X})_{b,t,c} = \begin{cases} \text{clip}(x_{b,t,c}, L_c, U_c), & \text{if } c \in \mathcal{O} \\ x_{b,t,c}, & \text{if } c \notin \mathcal{O} \end{cases} \tag{3.2}$$

where $\mathcal{O} = \{c : |\mu_c - \mu_{\text{grand}}| > k_{\text{outlier}} \cdot \sigma_{\text{grand}}\}$ represents the set of outlier channels identified during statistics collection, and the clipping function is defined as:

$$\text{clip}(x, l, u) = \max(l, \min(x, u)) \tag{3.3}$$

This selective approach ensures that the transformation Φ is identity-preserving for non-outlier channels, maintaining the majority of the model's computational pathways unchanged.

3.1 Hook Implementation Strategy

The clipping system uses PyTorch hooks that can be attached either before or after a module's computation, depending on whether we want to clip the input or output activations. For a module $\mathcal{M}$ with forward pass $\mathcal{M} : \mathbf{X} \mapsto \mathbf{Y}$, we can apply clipping in two ways:

$$\mathbf{Y} = \begin{cases} \mathcal{M}(\Phi(\mathbf{X})), & \text{for input clipping} \\ \Phi(\mathcal{M}(\mathbf{X})), & \text{for output clipping} \end{cases} \tag{3.4}$$

The clipping bounds remain constant throughout inference, eliminating redundant calculations during the forward pass.

The system also enables tensor shape detection and broadcasting mechanisms to handle heterogeneous activation formats. For a tensor $\mathbf{T}$ with shape $\mathcal{S}(\mathbf{T})$, we define a shape-adaptive broadcasting function $\mathcal{B} : \mathbb{R}^C \times \mathcal{S} \to \mathbb{R}^{\mathcal{S}}$ that reshapes channel-wise statistics to match activation dimensions.

For standard three-dimensional tensors with shape (B, T, C), the broadcasting operation expands channel statistics $\mathbf{v} \in \mathbb{R}^C$ to $\mathcal{B}(\mathbf{v}, (B, T, C)) \in \mathbb{R}^{1\times 1\times C}$, enabling element-wise operations through automatic broadcasting.

For convolutional outputs with shape (B, C, T), the system transposes the tensor to (B, T, C) before applying statistics, thereby providing consistent channel alignment: $\mathbf{X}' = \text{permute}(\mathbf{X}, (0, 2, 1))$.

The shape adaptation function maintains a cache $\mathcal{C} : \mathcal{S} \to (\mathbb{R}^{\mathcal{S}}, \mathbb{R}^{\mathcal{S}}, \mathbb{B}^{\mathcal{S}})$ that maps tensor shapes to pre-computed broadcast forms of lower bounds, upper bounds, and outlier masks, which contributes to reducing computational requirements during inference.

The outlier mask $\mathbf{M} \in \{0, 1\}^C$ identifies which channels require clipping based on their deviation from the grand mean of channel means:

$$M_c = \begin{cases} 1, & \text{if } |\mu_c - \mu_{\text{grand}}| > k_{\text{outlier}} \cdot \sigma_{\text{grand}} \\ 0, & \text{otherwise} \end{cases}$$

(3.5)

where:

$$\mu_{\text{grand}} = \frac{1}{C}\sum_{c=1}^{C}\mu_c$$

and

$$\sigma_{\text{grand}} = \sqrt{\frac{1}{C-1}\sum_{c=1}^{C}(\mu_c - \mu_{\text{grand}})^2}$$

.

The actual clipping transformation leverages conditional selection to minimize computational impact. For an activation tensor $\mathbf{X}$ and outlier mask $\mathbf{M} \in \{0,1\}^C$, the operation proceeds as:

$$\hat{\mathbf{X}} = \mathbf{M} \odot \text{clamp}(\mathbf{X}, \mathbf{L}, \mathbf{U}) + (\mathbf{1} - \mathbf{M}) \odot \mathbf{X} \quad (3.6)$$

where $\odot$ denotes element-wise multiplication with broadcasting, $\mathbf{1}$ is a tensor of ones, and clamp performs element-wise bounding. This formulation allows for vectorized execution without explicit conditional branching.

3.2 Clipping Thresholds

The system supports iterative evaluation across various threshold combinations $(k_{\text{outlier}}, k_{\text{clip}})$. For each parameter pair $(k_{\text{outlier},i}, k_{\text{clip},i})$, the evaluation proceeds through the following pipeline:

1. The outlier identification threshold and clipping range factors are set independently: the channel outlier criterion uses $|\mu_c - \mu_{\text{grand}}| > k_{\text{outlier},i} \cdot \sigma_{\text{grand}}$ while the clipping bounds use $\mu_c \pm k_{\text{clip},i}\sigma_c$. This allows independent optimization of outlier detection sensitivity and clipping aggressiveness.

2. Statistics are reprocessed to generate a new outlier mask $\mathbf{M}^{(k_{\text{outlier},i})} \in \{0,1\}^C$ for each recording position, where the superscript denotes threshold-specific identification.

3. Hooks are instantiated with updated parameters:

$$\Theta_{k_{\text{outlier},i}, k_{\text{clip},i}} = \{\mathbf{L}^{(k_{\text{clip},i})}, \mathbf{U}^{(k_{\text{clip},i})}, \mathbf{M}^{(k_{\text{outlier},i})}\}$$

creating a threshold-specific clipping configuration.

4. Model performance $P(k_{\text{outlier},i}, k_{\text{clip},i})$ is evaluated, yielding a performance surface that characterizes the clipping-accuracy trade-off across both parameter dimensions.

When monitoring is enabled, the system tracks clipping statistics through accumulator functions. For each hook h at position p, we maintain:

$$N_{\text{total}}^{(h)} = \sum_{k=1}^{K} |\mathbf{X}_k| \quad (3.7)$$

$$N_{\text{clipped}}^{(h)} = \sum_{k=1}^{K}\sum_{b,t,c} \mathbb{1}[\Phi(\mathbf{X}_k)_{b,t,c} \neq (\mathbf{X}_k)_{b,t,c}] \quad (3.8)$$

where $\mathbb{1}[\cdot]$ is the indicator function and $|\mathbf{X}_k|$ denotes the number of elements in tensor $\mathbf{X}_k$. The clipping rate

$$\rho^{(h)} = N_{\text{clipped}}^{(h)} / N_{\text{total}}^{(h)}$$

provides insight into the intervention frequency at each network position.

4 Results

4.1 Benchmark Selection and Descriptions

To evaluate the impact of outlier clipping across diverse language/reasoning abilities, we selected six benchmarks that cover key aspects of language model performance:

This benchmark composition was chosen to attempt to represent a spectrum of capabilities typically evaluated in language models: mathematical reasoning (GSM8K), factual knowledge (TriviaQA, ARC-Easy), commonsense reasoning (HellaSwag, Winogrande), and broad multidomain understanding (MMLU). The allowed us to assess whether outlier sensitivity varies across different cognitive functions.

- **ARC-Easy** (Clark et al., 2018): Contains grade-school level science questions requiring basic reasoning and factual knowledge.
- **GSM8K** (Cobbe et al., 2021): Presents grade school math word problems that test multi-step mathematical reasoning and problem decomposition.

- **HellaSwag** (Zellers et al., 2019): Evaluates commonsense reasoning through sentence completion tasks requiring understanding of everyday situations.
- **MMLU** (Hendrycks et al., 2020): (Massive Multitask Language Understanding) spans 57 academic subjects from elementary to professional levels, testing broad knowledge and reasoning across domains.
- **TriviaQA** (Joshi et al., 2017): Combines trivia questions with reading comprehension, requiring both factual knowledge retrieval and text understanding.
- **WinoGrande** (Sakaguchi et al., 2019): Tests commonsense reasoning through pronoun resolution tasks that require understanding implicit relationships and world knowledge.

All evaluations were conducted using one A100 GPU with 40GB VRAM, requiring approximately 45 compute hours total across all benchmark evaluations.

4.2 Outlier Channel Distribution Analysis

To understand the impact of outlier clipping on model performance, we first analyzed the distribution of outlier channels across the RecurrentGemma-2B architecture. RecurrentGemma-2B consists of 26 layers alternating between Griffin recurrent blocks (De et al., 2024b) and multi-head attention layers (Vaswani et al., 2023), with a hidden dimension of 2,560 and MLP intermediate dimension of 7,680 (Botev et al., 2024) . Using activation statistics collected from a calibration dataset, we identified channels whose mean activations deviate by more than 6 standard deviations from the grand mean of all channel means within each tensor position.

4.2.1 Component-Level Distribution

Our analysis reveals a highly non-uniform distribution of outlier channels across architectural components. Table 4.1 presents the total outlier channels identified in each component type:

The MLP blocks contain over half (52.3%) of all outlier channels, demonstrating their dominant role in producing extreme activations. This concentration is particularly striking when compared to attention mechanisms, which account for only 4.3% of outliers despite being central to the model's architecture. The recurrent blocks, unique to this hybrid architecture, contribute 22.6% of outliers, suggesting they also rely on extreme activation patterns for their computational function.

4.2.2 Layer-wise Distribution Patterns

The distribution of outlier channels across layers reveals a clear pattern of increasing concentration in deeper layers. Figure 4.1 and Table 4.2 detail this:

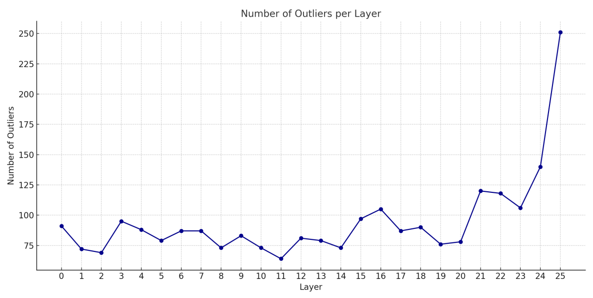


Figure 4.1: Distribution of outlier channels across layers

Table 4.2: Layer-wise outlier channel statistics

Layer Group	Average Outliers per Layer	Total Outliers
Early (0-8)	81.1	730
Middle (9-17)	82.7	744
Late (18-25)	117.1	937

The late layers (18-25) contain 44.4% more outlier channels per layer compared to early and middle layers, with a particularly dramatic spike in the final layer. This pattern suggests that the model increasingly relies on extreme activations for its final processing stages, potentially for making decisive predictions or aggregating information across the sequence.

4.2.3 Visualization of Outlier Patterns

We provide three-dimensional visualizations that reveal the spatial distribution of outlier channels

Table 4.1: Distribution of Outlier Channels by Component Type

Component Type	Total Channels	Outlier Channels	% of Component	% of Total Outliers
MLP	732,160	1,284	0.18%	52.11%
Recurrent	322,560	555	0.17%	22.52%
Normalization	271,360	349	0.13%	14.16%
Layer Boundary	133,120	170	0.13%	6.90%
Attention	86,016	106	0.12%	4.30%
Total	**1,545,216**	**2,464**		**100.00%**

across key architectural components. These visualizations provide an intuition into how the outlier channels cluster within the specific layers and channel ranges. The position shown are the ones that across all layers, per position type (attention, recurrent, MLP) show the highest number of outliers. We selected these for visualization given that they are the most essential components of the RecurrentGemma2B Architecture (further graphs can be found in Appendix A).

Figure 4.2a shows the outlier distribution in attention components, specifically at the query projection (q_proj) outputs position. The query projection transforms input embeddings into query vectors that determine what information each token should attend to. The lower concentration confirms that attention mechanism relies less heavily on extreme activation patterns compared to the other components.

Figure 4.2b shows the MLP up-projection outputs, which represent a subset of the MLP blocks. The up-projection layer expands the hidden dimension to create higher-capacity representations for complex feature learning. The visualization shows higher outlier density with in deeper layers, empirically noticeable in the dramatic concentration toward layers 20-25. This distribution directly supports our quantitative finding that MLP blocks contain 52.3% of all outliers.

Figure 4.2c presents the outlier pattern at the input to the final linear output layer. The final linear layer projects the model's internal representations into the output vocabulary space for token prediction. With the outlier density being the highest among our visualized components. This position shows how recurrent blocks aggregate and amplify certain activation patterns before final output generation. The visualization shows outlier channels scattered across the channel dimensions, with higher density in deeper layers. This demonstrates how recurrent blocks contribute 22.6% of outliers despite being architecturally distinct from traditional transformer blocks.

4.3 Global Clipping Performance Analysis

We evaluated the effect of global outlier clipping using standard deviation thresholds ranging from 2.0 to 6.0 across the six benchmark tasks described above. In order to isolate the impact of outlier removal from quantization effects, all experiments were conducted using full-precision (32-bit) numerics with outlier channels clipped to the specified thresholds, rather than performing actual quantization to lower bit-widths. Table 4.3 presents the comprehensive results, revealing consistent patterns in how different clipping thresholds affect model performance.

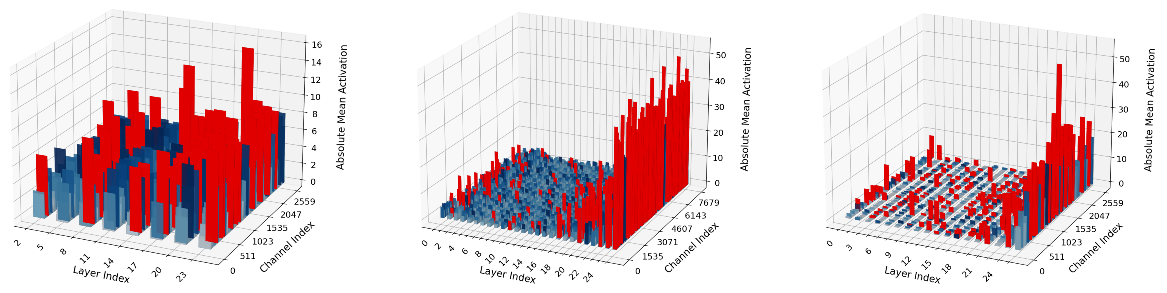
4.3.1 Threshold Sensitivity Analysis

The most striking observation is the severe performance degradation at the strictest threshold (std=2.0), where accuracies drop dramatically across all benchmarks. For instance, ARC-Easy accuracy plummets from the baseline of 0.7189 to 0.4827 (a 32.8% relative decrease), while TriviaQA experiences an even more catastrophic reduction from 0.3086 to 0.0152 (a 95.1% relative decrease).

As the threshold increases to 3.0, we observe substantial recovery across most benchmarks. ARC-Easy improves to 0.6730 (93.6% of baseline), and HellaSwag recovers from 0.4323 to 0.6807 (95.9% of baseline). This sharp improvement suggests that the channels identified as outliers at the 2.0 threshold include many that are essential for model functionality.

The performance continues to improve with higher thresholds, approaching baseline levels at

Figure 4.2: Three-dimensional visualization of outlier channel distribution across key architectural components. Height represents absolute mean activation magnitude, with red bars indicating channels exceeding the 6σ outlier threshold.



(a) Attention Query Projection outputs. Outlier Channels: 45/20480 (0.220%) **(b) MLP up-projection outputs. Outlier Channels: 293/199680 (0.147%)** **(c) Recurrent Final Linear Output inputs. Outlier Channels: 227/46080 (0.493%)**

Table 4.3: Model performance across different global clipping configurations with varying standard deviation thresholds. All values represent accuracy scores (proportion of correct answers)

Clipping Config.	ARC-Easy	GSM8K	HellaSwag	MMLU	TriviaQA	Winogrande
baseline	0.7189	0.0561	0.7094	0.3137	0.3086	0.6756
std=6.0	0.7104	0.0516	0.6957	0.3044	0.3005	0.6456
std=5.0	0.7130	0.0455	0.6932	0.3064	0.2920	0.6511
std=4.0	0.7193	0.0394	0.6974	0.3017	0.2730	0.6582
std=3.0	0.6730	0.0212	0.6807	0.2849	0.1915	0.6535
std=2.0	0.4827	0.0167	0.4323	0.2520	0.0152	0.5359

std=4.0 and beyond. Notably, at std=4.0, ARC-Easy achieves 0.7193, marginally exceeding the baseline performance, while other benchmarks reach 95-98% of their baseline accuracies.

4.3.2 Task-Specific Sensitivity Patterns

Different benchmark tasks exhibit varying sensitivity to outlier clipping:

Mathematical Reasoning (GSM8K): This task shows the most gradual recovery pattern, with performance improving from 0.0167 at std=2.0 to 0.0516 at std=6.0, still below the baseline of 0.0561. This suggests that mathematical reasoning capabilities are particularly dependent on activation patterns that deviate significantly from the mean, requiring higher thresholds to preserve functionality.

Factual Retrieval (TriviaQA): TriviaQA demonstrates extreme sensitivity to aggressive clipping, with accuracy dropping to near-zero (0.0152) at std=2.0. However, it shows steady recovery with increasing thresholds, reaching 0.3005 at std=6.0 (97.4% of baseline). This pattern indicates that factual knowledge retrieval relies heavily on specific outlier activation patterns that are disrupted by strict clipping.

Commonsense Reasoning (HellaSwag, Winogrande): These benchmarks show relatively robust recovery, achieving over 95% of baseline performance at std=3.0 and maintaining stable performance at higher thresholds. This suggests that commonsense reasoning capabilities are more distributed across the network and less dependent on extreme outlier activations.

Multi-task Understanding (MMLU): MMLU follows a similar pattern to commonsense reasoning tasks, with performance recovering to 90.8% of baseline at std=3.0 and reaching 97.0% at std=6.0, indicating moderate sensitivity to outlier clipping.

4.3.3 Optimal Threshold Analysis

The results suggest an optimal threshold range between 4.0 and 5.0 standard deviations, where we define optimality as achieving at least 95% of baseline performance across the majority of benchmarks while providing meaningful outlier control. At std=4.0, five out of six benchmarks reach at least 95% of baseline accuracy, with only GSM8K lagging at 70.2% of baseline.

4.4 Implications for Model Quantization

These findings have significant implications for quantization strategies in RecurrentGemma and similar architectures. The severe performance degradation at low thresholds (std=2.0) indicates that many channels classified as statistical outliers are, in fact, carrying critical information for model functionality.

The gradual performance recovery with increasing thresholds suggests a continuum of outlier importance, where channels with more extreme deviations (beyond 4-5 standard deviations) may represent less critical activation patterns that can be constrained without significant performance loss. This could help adaptive quantization, which applies different precision levels based on the statistical properties of activation channels.

The task-specific sensitivity patterns observed across benchmarks show that different capabilities within the model rely on outlier activations to varying degrees. Mathematical and factual reasoning tasks seem particularly dependent on preserving extreme activation patterns, whereas the commonsense reasoning tasks show greater robustness to outlier constraints.

4.5 Architectural Component Sensitivity

The component-specific analysis reveals interesting parallels:

Transformer Components: Bondarenko et al. (2023) identified that outliers in transformer models are particularly concentrated in residual connections and affect attention pattern formation. This aligns with our observation that attention components in RecurrentGemma show moderate sensitivity to outlier clipping, though they are more robust than MLP blocks.

SSM Components: The analysis of Mamba models by Pierro & Abreu (2024) showed that W4 quantization leads to near-complete model failure (for Mamba-1.4B), while W8 quantization maintains reasonable performance. This mirrors our findings where aggressive clipping ($\sigma = 2.0$) causes dramatic failures, while more conservative thresholds preserve functionality.

MLP Blocks: Both architectural paradigms show that MLP components are particularly sensitive to quantization.Pierro & Abreu (2024) report significant degradation when quantizing MLP blocks in Mamba (Mamba-2.8B drops from 66.16% to 48.74% on Arc-Challenge with W8A8 quantization of MLPs). Our results similarly indicate that MLP blocks in RecurrentGemma require careful handling of outliers.

4.6 Implications for Hybrid Model Quantization

The comparison reveals that hybrid models like RecurrentGemma inherit quantization challenges from both their constituent architectures. The fact that RecurrentGemma achieves 95–100% of baseline performance at $\sigma = 4.0$–5.0 thresholds indicates that the outlier patterns, while present in both attention and recurrent components, can be managed with a consistent approach.

This convergence of findings across architectural paradigms strengthens the case for outlier-aware quantization as a fundamental requirement for deploying large language models, whether they employ attention, recurrence, or hybrid mechanisms. Future work should explore whether architecture-specific optimizations (such as Bondarenko et al.'s per-embedding-group approach for transformers) could be adapted to further improve quantization outcomes in hybrid models.

This Bachelor's Project investigated the presence and impact of outlier channels in hybrid foundation models, specifically focusing on RecurrentGemma-2B. Our research has made several important contributions to understanding how these statistical anomalies function within hybrid architectures that combine attention mechanisms and recurrent components.

4.7 Summary of findings

Our analysis shows outlier channels exist in hybrid foundation models, similarly to their presence in transformer and SSM architectures. We confirmed that, just like in the aforementioned architectures, outlier channels are not mere statistical artifacts but rather fundamental components that significantly contribute to model performance across various reasoning tasks.

The analysis of outlier distribution yielded important insights. First, MLP blocks contained the highest concentration of outlier channels, accounting for 52.11% of all 2,464 identified outliers despite representing only 0.18% of channels within MLP components. This concentration was particularly noticeable when compared to attention mechanisms, underlining the different activation dynamics across architectural components.

Secondly, our layer-wise analysis showed the existence of a clear pattern of increasing outlier density in deeper layers. Late layers (18–25) exhibited 44.4% more outlier channels per layer compared to early and middle layers, with layer 25 showing particularly dramatic concentrations. This pattern indicates that the model increasingly relies on extreme activation patterns for its final processing stages, potentially for decisive predictions or information aggregation.

Thirdly, global clipping experiments with standard deviation thresholds ranging from 2.0 to 6.0 revealed severe performance degradation at aggressive thresholds. At $\sigma = 2.0$, ARC-Easy accuracy dropped by 32.8% while TriviaQA experienced a catastrophic 95.1% reduction. Performance recovered substantially at $\sigma = 3.0$, with most benchmarks achieving over 90% of baseline, and reached near-optimal levels at thresholds between 4.0 and 5.0 standard deviations.

Finally, different reasoning tasks exhibited distinct sensitivity patterns to outlier restriction. Mathematical reasoning (GSM8K) showed the most gradual recovery, achieving only 70.2% of baseline performance even at $\sigma = 4.0$. Factual retrieval (TriviaQA) demonstrated extreme initial sensitivity but robust recovery at higher thresholds. On the other hand, commonsense reasoning tasks (HellaSwag, WinoGrande) showed greater resilience, achieving over 95% of baseline performance at $\sigma = 3.0$.

4.8 Implications

For quantization strategies, the identification of optimal clipping thresholds between 4.0 and 5.0 standard deviations can provide guidance for preserving model functionality while, at the same time, enabling outlier control. The degradation at lower thresholds shows that many statistically-identified outliers carry critical information that must be preserved in quantization schemes.

From an architectural perspective, the concentration of outliers in MLP blocks (52.11%) versus attention mechanisms (4.30%) shows fundamental differences in how these components process information. This seems to indicate that quantization approaches should apply component-specific strategies, with MLP blocks requiring more careful outlier preservation than attention layers.

4.9 Limitations and Future Work

While the project focused on RecurrentGemma-2B, examining other hybrid architectures (such as Jamba (Belinkov et al., 2024) or Samba (L. Ren et al., 2024b)) would provide more insights into whether our findings generalize across different hybrid implementations. Furthermore, testing RecurrentGemma-9B would help us see how the presence of outlier channels in the same model architectures scales with more parameters.

Our use of standard deviation-based thresholds ($\mu + 6\sigma$) for outlier detection follows established practices but may not capture all patterns of outlier behavior, particularly in distributions with multiple modes or heavy tails. Additionally, expanding to generation, translation, or other capabilities benchmarks would provide a more complete picture of outlier impacts.

Future work should explore using absolute deviation around the median rather than standard deviation around the mean for outlier detection (Leys et al., 2013). This is believed to be more robust to extreme values and potentially provide a more accurate identification of functionally significant outlier channels.

A direct comparative analysis of outlier patterns in pure transformer, pure SSM, and hybrid architectures of comparable size would yield valuable insights into architectural differences and potentially inform better hybrid design principles. The development of quantization techniques that preserve the functionality of identified outlier channels while optimizing resource usage elsewhere in the network could lead to more efficient deployment of hybrid models.

Finally, looking into how outlier channels emerge during training and whether their occurrence can be predicted or controlled would help our understanding of these phenomena.

4.10 Final Remarks

Our research has demonstrated that outlier channels, far from being statistical anomalies, are fundamental components of hybrid foundation models with significant impacts on model performance. By identifying where these outliers occur and quantifying their effects across different architectural components and reasoning tasks, we have contributed to both the theoretical understanding of hybrid architectures and practical knowledge for their optimization.

As model architectures continue to evolve, understanding the fundamental aspects of their internal dynamics will be important for advancing the field. The insights from our study provide a foundation for future work in hybrid-foundation model interpretation and optimization.

References

Belinkov, Y., et al. (2024). Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*. Retrieved from https://arxiv.org/abs/2403.19887

Bondarenko, Y., Nagel, M., & Blankevoort, T. (2023). *Quantizable transformers: Removing outliers by helping attention heads do nothing.* Retrieved from https://arxiv.org/abs/2306.12929

Botev, A., De, S., Smith, S. L., Fernando, A., Muraru, G. C., Haroun, R., & de Frietas, N. (2024). Recurrentgemma: Moving past transformers for efficient open language models. *arXiv preprint arXiv:2404.07839*.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think you have solved question answering? try arc, the ai2 reasoning challenge. In *Proceedings of the 2018 conference on empirical methods in natural language processing (emnlp)* (pp. 4144–4153).

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., . . . Schulman, J. (2021). Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

De, S., Smith, S. L., Fernando, A., Botev, A., Cristian-Muraru, G., Gu, A., . . . Gulcehre, C. (2024a). *Griffin: Mixing gated linear recurrences with local attention for efficient language models.* Retrieved from https://arxiv.org/abs/2402.19427

De, S., Smith, S. L., Fernando, A., Botev, A., Cristian-Muraru, G., Gu, A., . . . Gulcehre, C. (2024b). Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*. Retrieved from https://arxiv.org/abs/2402.19427

Dettmers, T., Lewis, M., Belkada, Y., & Zettlemoyer, L. (2022). *Llm.int8(): 8-bit matrix multiplication for transformers at scale.* Retrieved from https://arxiv.org/abs/2208.07339

Gu, A., & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

He, B., Noci, L., Paliotta, D., Schlag, I., & Hofmann, T. (2024). *Understanding and minimising outlier features in neural network training.* Retrieved from https://arxiv.org/abs/2405.19279

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Zou, E., . . . Song, D. (2020). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Heo, J. H., Kim, J., Kwon, B., Kim, B., Kwon, S. J., & Lee, D. (2023). Rethinking channel dimensions to isolate outliers for low-bit weight quantization of large language models. *arXiv preprint arXiv:2309.15531*.

Joshi, M., Choi, E., Weld, D. S., & Zettlemoyer, L. (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th annual meeting of the association for computational linguistics (acl)* (pp. 1601–1611).

Leys, C., Ley, C., Klein, O., Bernard, P., & Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of experimental social psychology*, *49*(4), 764–766.

Li, S., Ning, X., Wang, L., Liu, T., Shi, X., Yan, S., & Wang, Y. (2024). Evaluating quantized large language models. *arXiv preprint arXiv:2402.18158*.

Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., & Shoham, Y. (2024). Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.

Liu, J., Gong, R., Wei, X., Dong, Z., Cai, J., & Zhuang, B. (2023). Qllm: Accurate and efficient low-bitwidth quantization for large language models. *arXiv preprint arXiv:2310.08041*.

Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2016). Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Merrill, W., Petty, J., & Sabharwal, A. (2024). The illusion of state in state-space models. *arXiv preprint arXiv:2404.08819*.

Nrusimha, A., Mishra, M., Wang, N., Alistarh, D., Panda, R., & Kim, Y. (2024). *Mitigating the impact of outlier channels for language model quantization with activation regularization.* Retrieved from https://arxiv.org/abs/2404.03605

Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Biderman, S., & Zhu, R. J. (2023). Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*.

Pierro, A., & Abreu, S. (2024). Mamba-ptq: Outlier channels in recurrent large language models. *arXiv preprint arXiv:2407.12397*.

Puccetti, G., Rogers, A., Drozd, A., & Dell'Orletta, F. (2022). Outlier dimensions that disrupt transformers are driven by frequency. In *Findings of the association for computational linguistics: Emnlp 2022* (p. 1286–1304). Association for Computational Linguistics. Retrieved from http://dx.doi.org/10.18653/v1/2022.findings-emnlp.93 doi: 10.18653/v1/2022.findings-emnlp.93

Qin, Z., Yang, S., & Zhong, Y. (2024). Hierarchically gated recurrent neural network for sequence modeling. In *Advances in neural information processing systems* (Vol. 36).

Ren, H., Dai, H., Dai, Z., Yang, M., Leskovec, J., Schuurmans, D., & Dai, B. (2021). Combiner: Full attention transformer with sparse computation cost. In *Advances in neural information processing systems* (Vol. 34, pp. 22470–22482).

Ren, L., Liu, Y., Lu, Y., Shen, Y., Liang, C., & Chen, W. (2024a). Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv:2406.07522*.

Ren, L., Liu, Y., Lu, Y., Shen, Y., Liang, C., & Chen, W. (2024b). Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv:2406.07522*. Retrieved from https://arxiv.org/abs/2406.07522

Sakaguchi, K., Bras, R. L., Bhagavatula, C., & Choi, Y. (2019). *Winogrande: An adversarial winograd schema challenge at scale.* Retrieved from https://arxiv.org/abs/1907.10641

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2023). Attention is all you need. *arXiv preprint arXiv:1706.03762*. (http://arxiv.org/abs/1706.03762)

Wei, X., Zhang, Y., Li, Y., Zhang, X., Gong, R., Guo, J., & Liu, X. (2023). Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*.

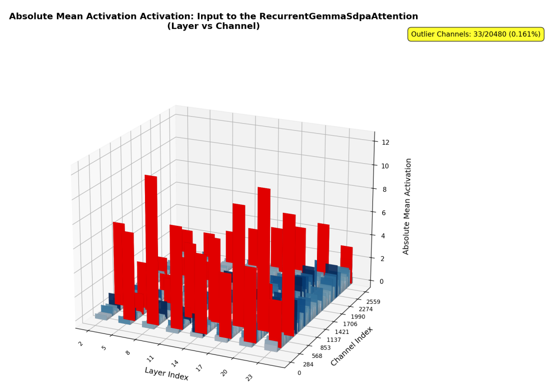
Welford, B. P. (1962). Note on a method for calculating corrected sums of squares and products. *Technometrics*, *4*(3), 419–420.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . Rush, A. M. (2020, October). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations* (pp. 38–45). Online: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/2020.emnlp-demos.6

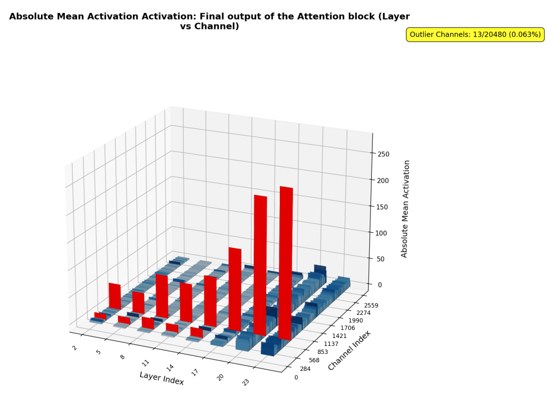
Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., & Choi, Y. (2019, July). HellaSwag: Can a machine really finish your sentence? In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 4791–4800). Florence, Italy: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P19-1472 doi: 10.18653/v1/P19-1472

A Appendix: Activation Analysis Barplots

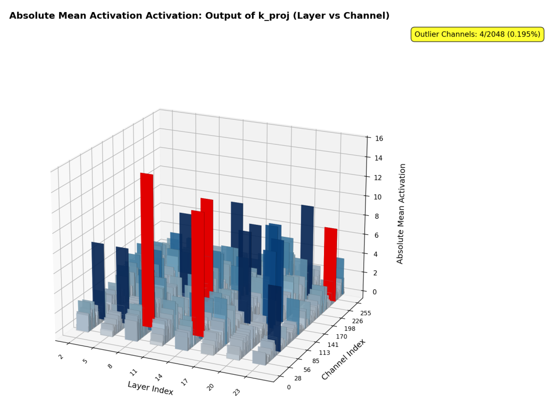
Attention Block Analysis



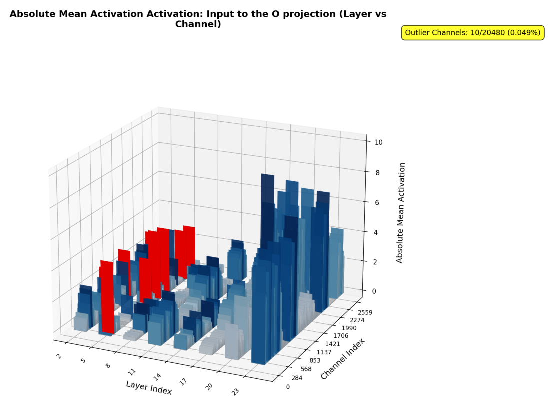
(a) Attention Block Input



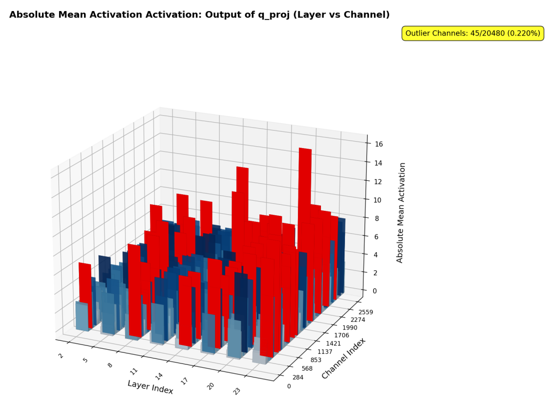
(b) Attention Final Output



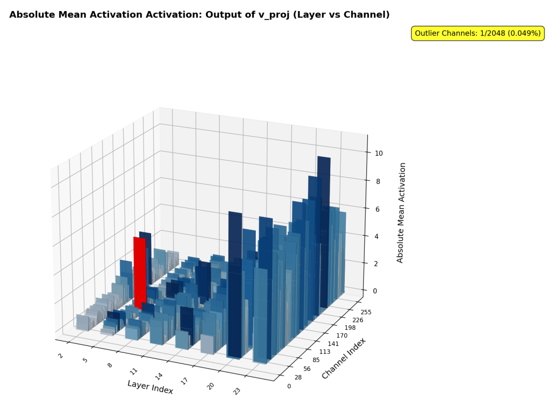
(c) Attention K Projection Output



(d) Attention O Projection Input

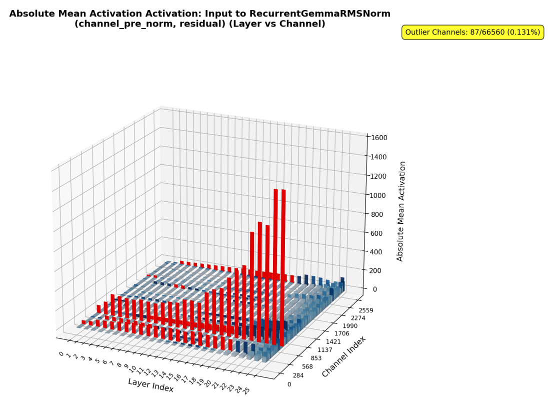


(e) Attention Q Projection Output

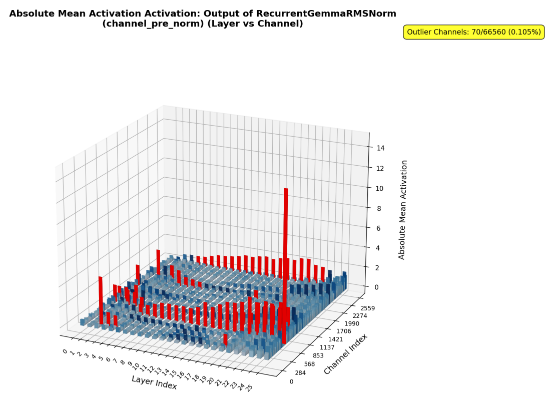


(f) Attention V Projection Output

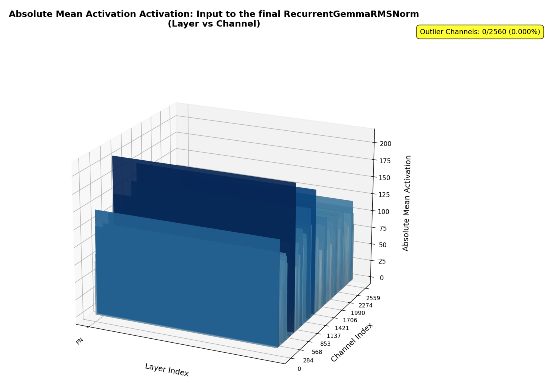
Channel and Layer Normalization Analysis



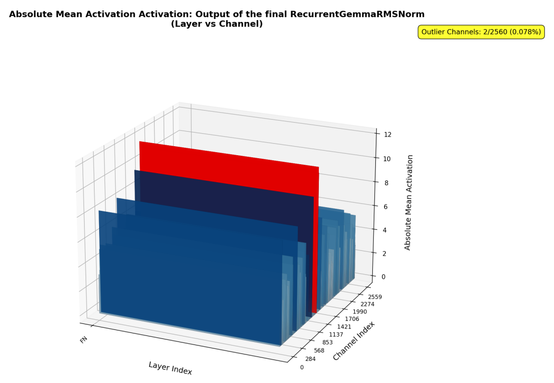
(a) Channel Pre-Norm Input



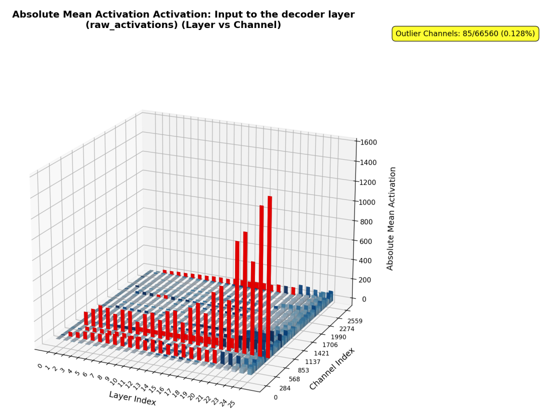
(b) Channel Pre-Norm Output



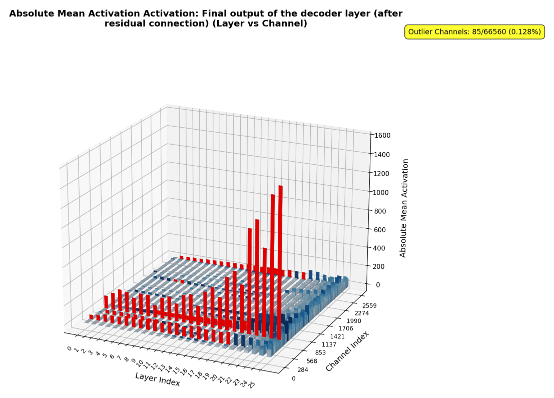
(c) Final Norm Input



(d) Final Norm Output

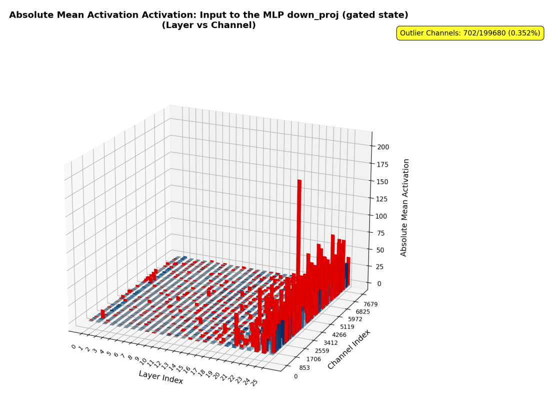


(e) Layer Input

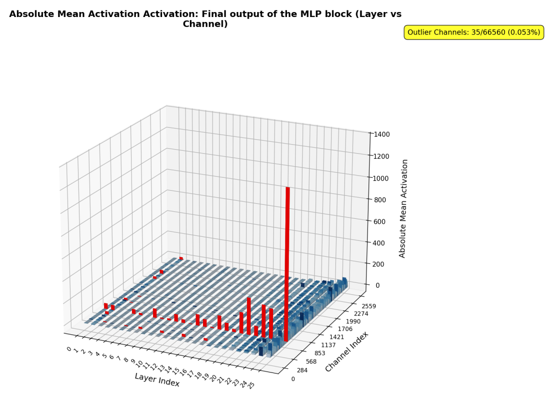


(f) Layer Output

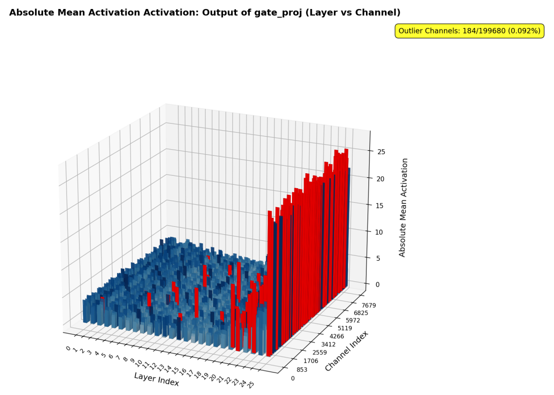
MLP Analysis



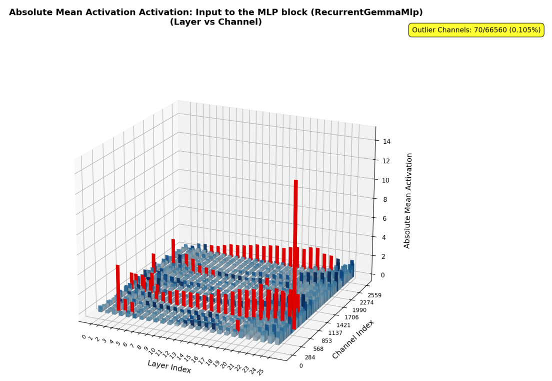
(a) MLP Down Projection Input



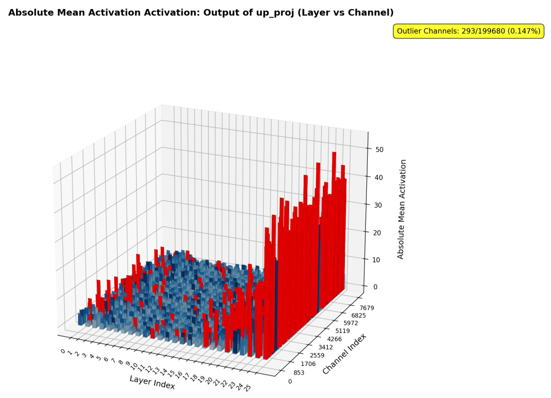
(b) MLP Final Output



(c) MLP Gate Projection Output

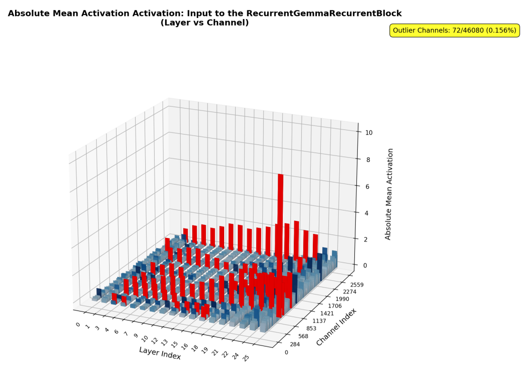


(d) MLP Input

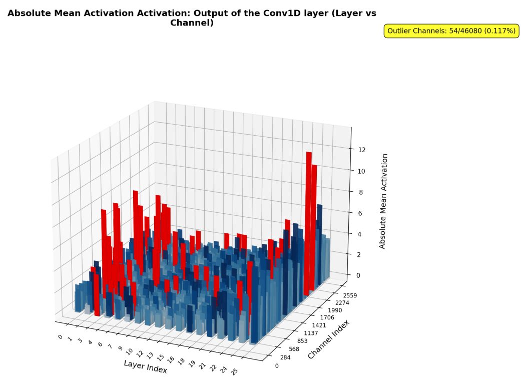


(e) MLP Up Projection Output

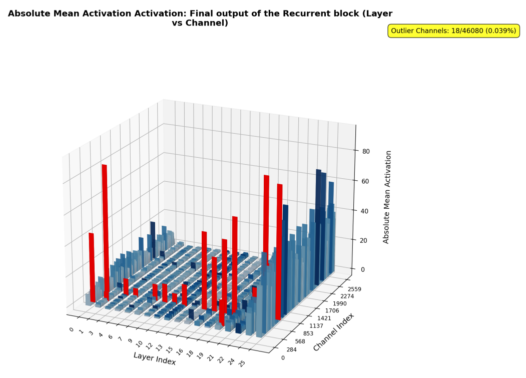
Recurrent Block Analysis



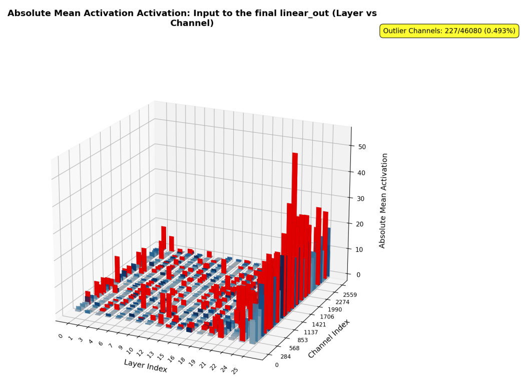
(a) **Recurrent Block Input**



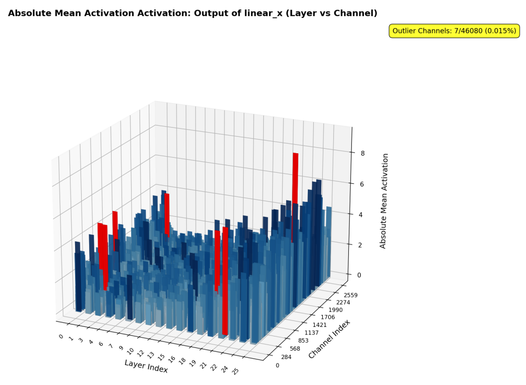
(b) **Recurrent Conv1D Output**



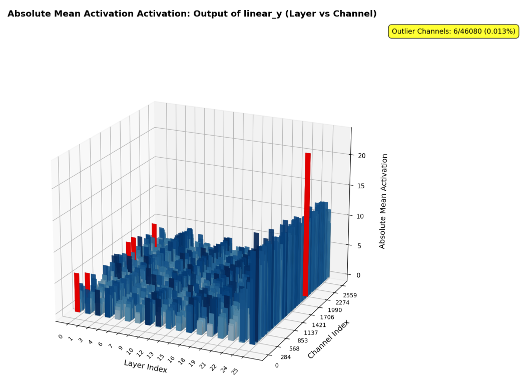
(c) **Recurrent Final Output**



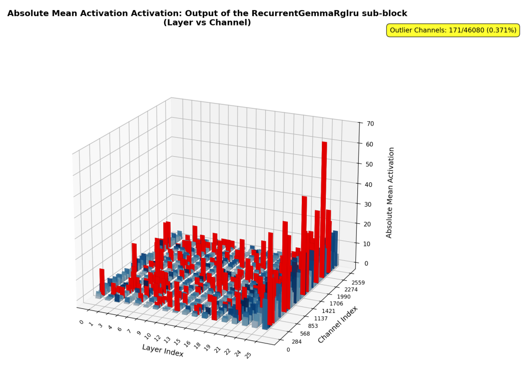
(d) **Recurrent Linear Out Input**



(e) **Recurrent Linear X Output**

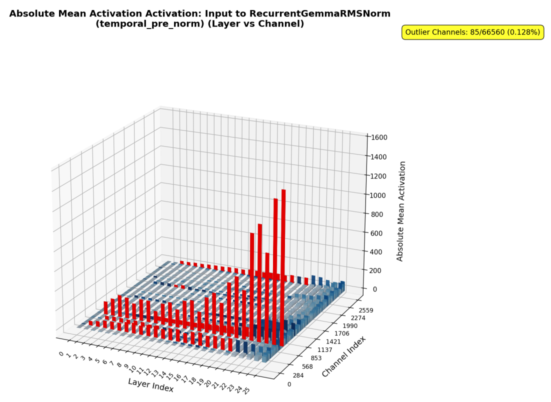


(f) **Recurrent Linear Y Output**

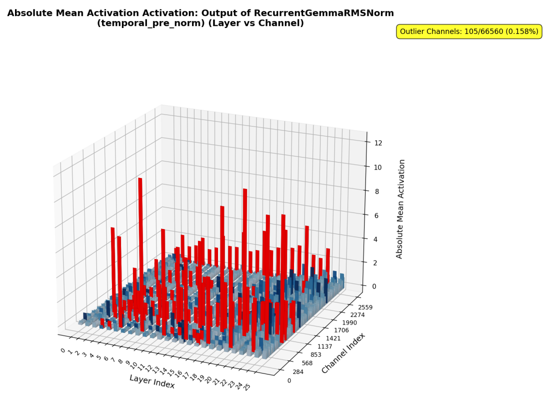


(g) **Recurrent RGLRU Output**

Temporal Pre-Normalization Analysis



(a) Temporal Pre-Norm Input



(b) Temporal Pre-Norm Output