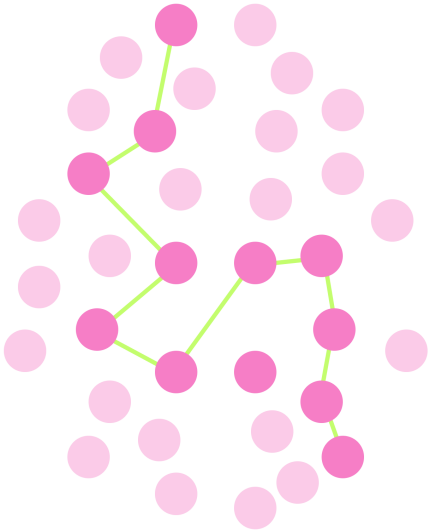
CATS Blog

Human Brains Versus LLMs



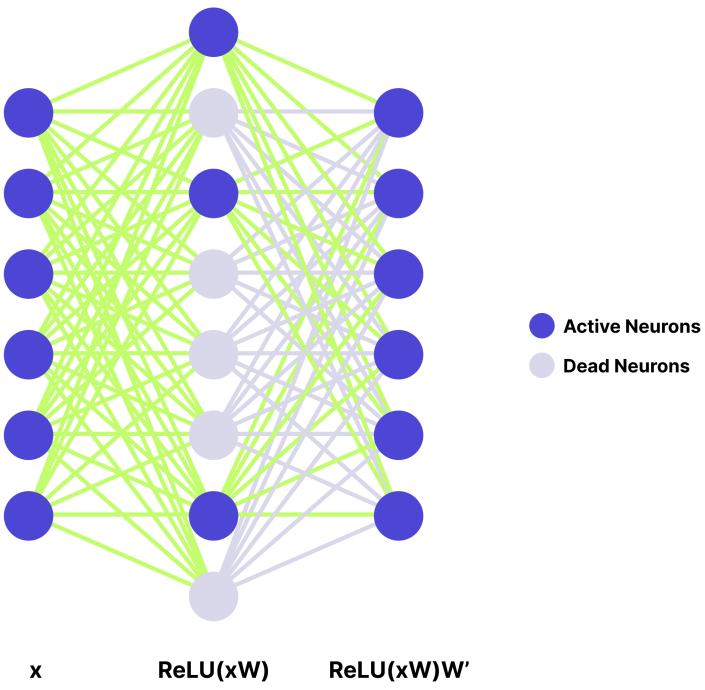
The human brain operates with remarkable energy efficiency, despite its complexity. Usually, a human brain activates only a sparse array of neurons at any given moment. Do state-of-the-art large language models (LLMs) behave

similarly? Until recently, the answer was no: they typically exhibit over 99% non-zero activations during inference. However, our recent research unveils a surprising observation: LLMs activations are intrinsically sparse.

In our work, we introduce CATS, a simple post-training technique that achieves 50% activation sparsity for MLP layers with a negligible drop in downstream evaluations. CATS requires little to no finetuning of existing LLMs. We leverage this newfound sparsity to achieve 15% improvement in end-to-end generation latency for both Mistral-7B and Llama-7B models using custom GPU kernels.

Early LLMs Were Sparse

The choice of activation function is crucial in neural networks and determines when a neuron should activate in response to incoming signals. Activation functions inject indispensable non-linearities to the model; without them, a neural network is equivalent to linear regression.

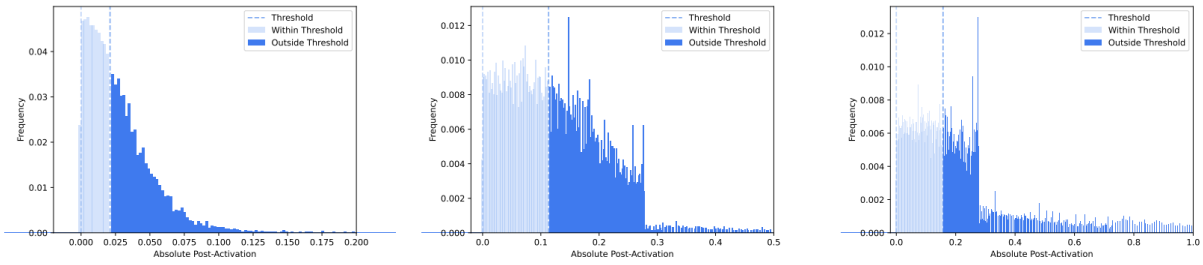


The ReLU activation function is popular for its simplicity and effectiveness. It operates on a clear principle: positive inputs are unaltered, while negatives are zeroed out. This creates a sparse network in which a majority of neurons lie dormant and drastically reduces computational costs. Models like GPT-2 and OPT-3 have cleverly utilized ReLU to achieve up to 85% sparsity to combine speed and efficiency in a way that mirrors the activity of the human brain.

Are New LLMs Sparse Too?

More complex, recently-developed activation functions, however, elicit different behavior. The SwiGLU (Sigmoid-Weighted Linear Unit) activation, used by newer LLMs like Llama and Mistral, adapts its response dynamically and ensures almost continuous activation across the network; nearly 99.9% of neurons are activated. This power comes at a cost: the loss of the elegant sparsity seen with ReLU.

The quest to reinstate this sparsity led to initiatives like "ReLU Strikes Back", which aimed to retrofit SwiGLU-equipped models with ReLU. This approach, however, required excessive fine-tuning and consumed 30 billion tokens to match previous performance benchmarks. Such work highlights an open research question: **can we restore sparsity in state-of-the-art language models that use more complex activation functions?**



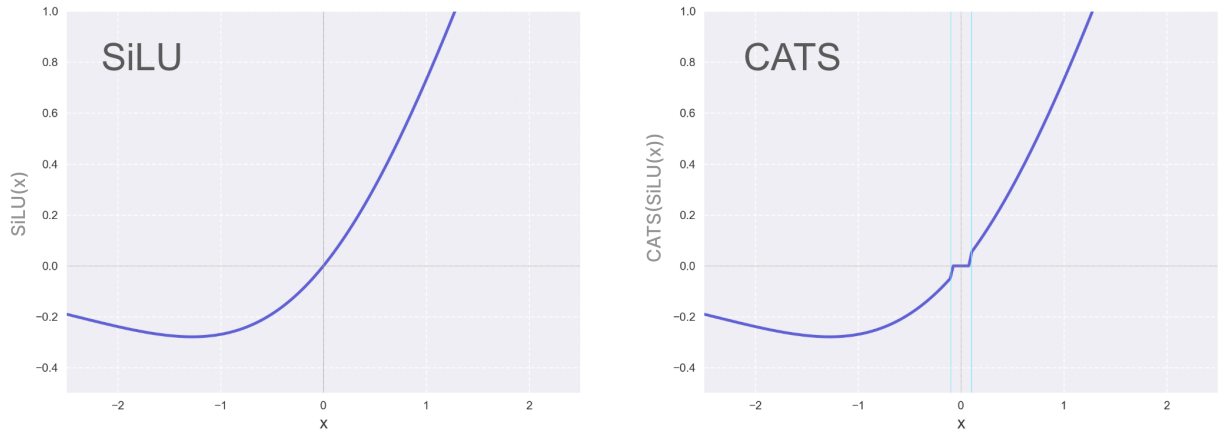
(a) Llama2 Layer 0. (b) Llama2 Layer 15. (c) Llama2 Layer 31.

But what if the key to unlocking this efficiency has been lurking within the models all along? Our observations revealed a curious pattern: activations in Gated-MLP layers consistently cluster "near" zero. This nuanced -- yet significant -- behavior indicates that many weights in these layers have minimal impact on the results and suggests potential approaches.

CATS: Reclaiming Activation Sparsity

The SwiGLU activation applies the SiLU activation function to one set of transformed inputs, then multiplies this result with another transformed set of inputs. The use of SiLU is where the typical sparsity of activations is lost (in contrast with ReLU).

In our approach, we introduce a novel activation function to replace SiLU designed to reclaim activation sparsity in MLP layers of large language models efficiently. The design of our novel activation function proceeds in two steps: determining values below which activations can be zeroed out and then dynamically zeroing these activations during inference.

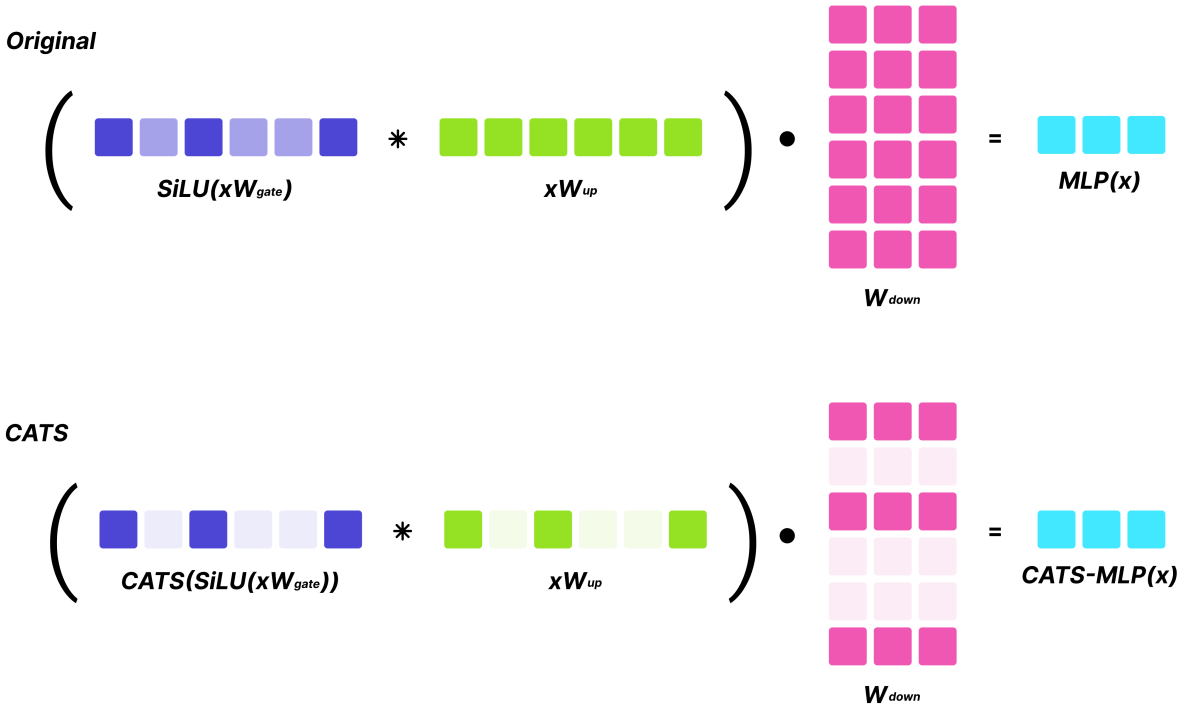


1. Establishing a Percentile Threshold: The first step of our approach involves determining a specific percentile threshold for each MLP layer. This threshold is critical as it dictates the point below which activations will be considered insignificant and thus set to zero. We establish this threshold by analyzing activation patterns within a small subset of the RefinedWeb Dataset. This analysis is performed offline to ensure that our threshold settings are robust and effective without the need for real-time computation.
2. **Zeroing Out Small Activations**: Once the thresholds are established, we apply them across the MLP layers during the model's operational phase. Any activation whose absolute value falls below the determined threshold is zeroed out.

In contrast with prior work, we minimally alter the existing activation function and only zero out negligible values. This adjustment leads to a marginal decrease in zero-shot performance without any fine-tuning. In our work, however, we also show that we can reclaim downstream task performance via minimal, parameter-efficient fine-tuning.

Leveraging Sparsity for Performance

Previous work has shown that optimizing for FLoPS does not necessarily lead to wall-clock inference time improvements. To harness the sparsity achieved by our CATS method for inference gains, we utilize a custom GPU kernel tailored for Gated-MLP blocks in large language models. In a Gated-MLP block, two weight tensors W_{up} and W_{down} follow W_{gate}.



With the CATS activation function, we can think of W_{gate} as a "router" and the columns/rows of W_{up} and W_{down} as "experts". The router W_{gate} chooses the

most relevant experts based on the input characteristic and skips many of the weights in W_{up} and W_{down}, thereby bypassing unnecessary computations.

This streamlined process is supported by our custom GPU kernel, which efficiently handles the loading and computing of only the essential weights from W_{up} and W_{down}.

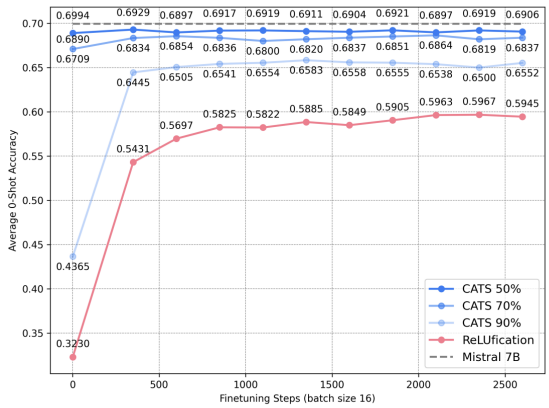
Custom GPU Kernel 1 MLP using CATS

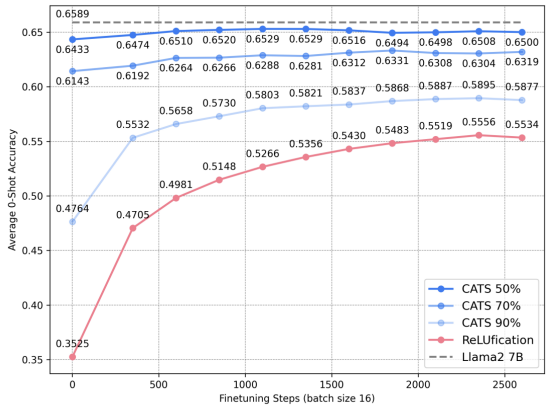
Input: threshold $t > 0$, hidden layer x, weights W_{gate}, W_{down}, and W_{up}
$v \leftarrow \text{SiLU}(xW_{gate})$
$\texttt{Mask} \leftarrow 1 \text{ if } |v| \geq t \text{ else } 0$
$x_1 \leftarrow (xW_{\text{up}}[\texttt{Mask}] * v[\texttt{Mask}])$
$y \leftarrow x_1 W_{\text{down}}[\texttt{Mask}]$

The kernel is designed to minimize memory access, which in turn reduces the inference time for the MLP. Since MLP computations are primarily limited by communication costs rather than computation costs, this optimization is crucial. Furthermore, the kernel makes memory use more efficient and accelerates processing by fusing operation. Additionally, we employ a binary mask to manage the activation of experts, streamlining the process by replacing complex indexing mechanisms with a simpler, faster alternative.

Efficiency and Performance Analysis

CATS significantly preserves the zero-shot capabilities of the base model across various benchmarks. Remarkably, it achieves this without any fine-tuning, showing a modest performance reduction of only about 1.5% in 8 evaluations even at 50% sparsity. With minimal, parameter-efficient fine-tuning, CATS delivers performance comparable to the original model at the same sparsity level.

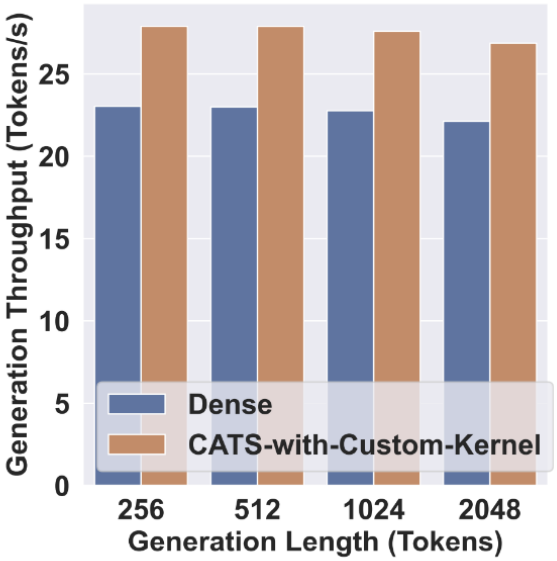




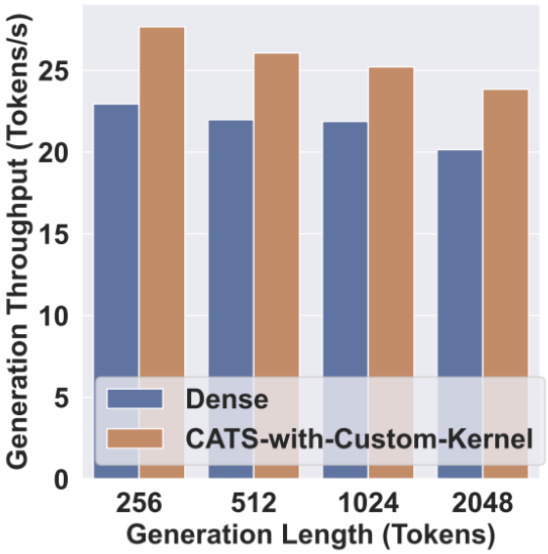
Moreover, directly fine-tuning CATS on a downstream task shows that it nearly retains the performance of the original model.

Dataset/Sparsity	Base Model	0.5	0.7	0.9	ReLUfication
Cola	**0.8667**	0.8658 (-0.10%)	0.8552 (-1.32%)	0.8303 (-4.21%)	0.6922 (-20.13%)
SST2	0.9644	0.9656 (+0.12%)	**0.9702** (+0.60%)	0.9427 (-2.25%)	0.7856 (-18.55%)
BoolQ	**0.8905**	0.8862 (-0.48%)	0.8807 (-1.10%)	0.7920 (-11.06%)	0.6624 (-25.61%)
Average	**0.9072**	0.9059 (-0.13%)	0.9020 (-0.52%)	0.8550 (-5.22%)	0.7134 (-19.38%)

Finally, CATS improves the computational efficiency of an MLP layer by at least 40%, which significantly enhances the overall throughput by 15% during token generation.



(a) Mistral-7B



(b) Llama2-7B

Conclusions

Our research reveals that LLMs are naturally "nearly" sparse. With this insight, we propose CATS, which capitalizes on this near-sparsity to significantly boost efficiency. CATS effectively maintains high performance with minimal adjustments and notably increases throughput, showcasing sparsity as a key lever for optimizing computational resources in AI systems. This advancement will hopefully pave the way for more sustainable and efficient LLM operations.

For a deeper dive into our methodology and findings, please see our paper. Keep an eye out for the upcoming release of our code and reach out with any questions!