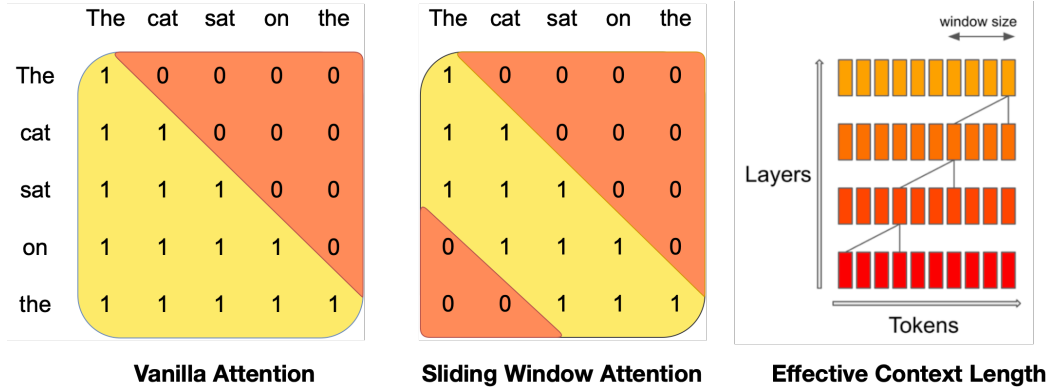


Mistral 7B is released under the Apache 2.0 license. This release is accompanied by a reference implementation<sup>1</sup> facilitating easy deployment either locally or on cloud platforms such as AWS, GCP, or Azure using the vLLM [17] inference server and SkyPilot<sup>2</sup>. Integration with Hugging Face<sup>3</sup> is also streamlined for easier integration. Moreover, Mistral 7B is crafted for ease of fine-tuning across a myriad of tasks. As a demonstration of its adaptability and superior performance, we present a chat model fine-tuned from Mistral 7B that significantly outperforms the Llama 2 13B – Chat model.

Mistral 7B takes a significant step in balancing the goals of getting high performance while keeping large language models efficient. Through our work, our aim is to help the community create more affordable, efficient, and high-performing language models that can be used in a wide range of real-world applications.

## 2 Architectural details



**Figure 1: Sliding Window Attention.** The number of operations in vanilla attention is quadratic in the sequence length, and the memory increases linearly with the number of tokens. At inference time, this incurs higher latency and smaller throughput due to reduced cache availability. To alleviate this issue, we use sliding window attention: each token can attend to at most  $W$  tokens from the previous layer (here,  $W = 3$ ). Note that tokens outside the sliding window still influence next word prediction. At each attention layer, information can move forward by  $W$  tokens. Hence, after  $k$  attention layers, information can move forward by up to  $k \times W$  tokens.

Mistral 7B is based on a transformer architecture [27]. The main parameters of the architecture are summarized in Table 1. Compared to Llama, it introduces a few changes that we summarize below.

**Sliding Window Attention.** SWA exploits the stacked layers of a transformer to attend information beyond the window size  $W$ . The hidden state in position  $i$  of the layer  $k$ ,  $h_i$ , attends to all hidden states from the previous layer with positions between  $i - W$  and  $i$ . Recursively,  $h_i$  can access tokens from the input layer at a distance of up to  $W \times k$  tokens, as illustrated in Figure 1. At the last layer, using a window size of  $W = 4096$ , we have a theoretical attention span of approximately 131K tokens. In practice, for a sequence length of 16K and  $W = 4096$ , changes made to FlashAttention [11] and xFormers [18] yield a 2x speed improvement over a vanilla attention baseline.

**Rolling Buffer Cache.** A fixed attention span means that we can limit our cache size using a rolling buffer cache. The cache has a fixed size of  $W$ , and the keys and values for the timestep  $i$  are stored in position  $i \bmod W$  of the cache. As a result, when the position  $i$  is larger than  $W$ , past values in the cache are overwritten, and the size of the cache stops increasing. We provide an illustration in Figure 2 for  $W = 3$ . On a sequence length of 32k tokens, this reduces the cache memory usage by 8x, without impacting the model quality.

Parameter	Value
dim	4096
n_layers	32
head_dim	128
hidden_dim	14336
n_heads	32
n_kv_heads	8
window_size	4096
context_len	8192
vocab_size	32000

**Table 1: Model architecture.**

<sup>1</sup><https://github.com/mistralai/mistral-src>

<sup>2</sup><https://github.com/skypilot-org/skypilot>

<sup>3</sup><https://huggingface.co/mistralai>