




# Datasets Arrow

## What is Arrow?

[Arrow](#) enables large amounts of data to be processed and moved quickly. It is a specific data format that stores data in a columnar memory layout. This provides several significant advantages:

- Arrow's standard format allows [zero-copy reads](#) which removes virtually all serialization overhead.
- Arrow is language-agnostic so it supports different programming languages.
- Arrow is column-oriented so it is faster at querying and processing slices or columns of data.
- Arrow allows for copy-free hand-offs to standard machine learning tools such as NumPy, Pandas, PyTorch, and TensorFlow.
- Arrow supports many, possibly nested, column types.

## Memory-mapping

 Datasets uses Arrow for its local caching system. It allows datasets to be backed by an on-disk cache, which is memory-mapped for fast lookup.

This architecture allows for large datasets to be used on machines with relatively small device memory.

For example, loading the full English Wikipedia dataset only takes a few MB of RAM:

```

>>> import os; import psutil; import timeit
>>> from datasets import load_dataset

# Process.memory_info is expressed in bytes, so convert to megabytes
>>> mem_before = psutil.Process(os.getpid()).memory_info().rss / (1024 * 1024)
>>> wiki = load_dataset("wikimedia/wikipedia", "20220301.en", split="train")
>>> mem_after = psutil.Process(os.getpid()).memory_info().rss / (1024 * 1024)

>>> print(f"RAM memory used: {(mem_after - mem_before)} MB")
RAM memory used: 50 MB

```

This is possible because the Arrow data is actually memory-mapped from disk, and not loaded in memory.

Memory-mapping allows access to data on disk, and leverages virtual memory capabilities for fast lookups.

## Performance

Iterating over a memory-mapped dataset using Arrow is fast. Iterating over Wikipedia on a laptop gives you speeds of 1-3 Gbit/s:

```

>>> s = """batch_size = 1000
... for batch in wiki.iter(batch_size):
...     ...
... """

>>> elapsed_time = timeit.timeit(stmt=s, number=1, globals=globals())
>>> print(f"Time to iterate over the {wiki.dataset_size >> 30} GB dataset: {elapsed_time:.1f} sec,
...       f"ie. {float(wiki.dataset_size >> 27)/elapsed_time:.1f} Gb/s")
Time to iterate over the 18 GB dataset: 31.8 sec, ie. 4.8 Gb/s

```