

PP10: Preparing Big Data for Analysis

Thomas Ellebæk

Statistical Operations, Global Biometrics, Ferring Pharmaceuticals A/S

PHUSE EU Connect 2019
Amsterdam, The Netherlands

Introduction

Training a deep neural network on terabytes of e.g. imaging data requires both computing power and proper parallelization techniques. When the amount of data exceeds what can be stored in memory, batches of data needs to be read repeatedly from more persistent storage.

At Ferring Global Biometrics we are planning to train a deep neural network on a large amount of image data (+500 GB). The data will be available to Ferring from start of 2020, so we haven't started the actual modelling yet. However, in order to train a model on a large amount of data we need an efficient way of preparing and pipelining an image repository in support of deep learning. We have explored three different approaches; The TensorFlow™ data API (tf.data), the Keras Sequence class and HDF5 files.

Conclusion

Building an efficient data pipeline for deep learning model training remains a challenge where the optimal solution depends on various factors like experiment design, dataset size and hardware configuration. A good approach is to design an experiment, identify the pipeline bottlenecks and improve from there.

The TensorFlow™ data API is most efficient for sequential batch reading without data shuffling. However, it is recommended to shuffle data between learning epochs, and with big datasets, the tf.data API do not provide an efficient way to shuffle data properly. **The most efficient batch shuffling data generator was obtained as a custom Keras sequence generator reading images from blob storage and repeatedly preprocessing them in each epoch.**

Constraints

The Ferring Global Biometrics image repositories reside in Azure storage accounts. We have some freedom as to whether we place the data in file shares or blob containers.



Blobs

REST-based object storage for unstructured data

[Learn more](#)



Files

File shares that use the standard SMB 3.0 protocol

[Learn more](#)

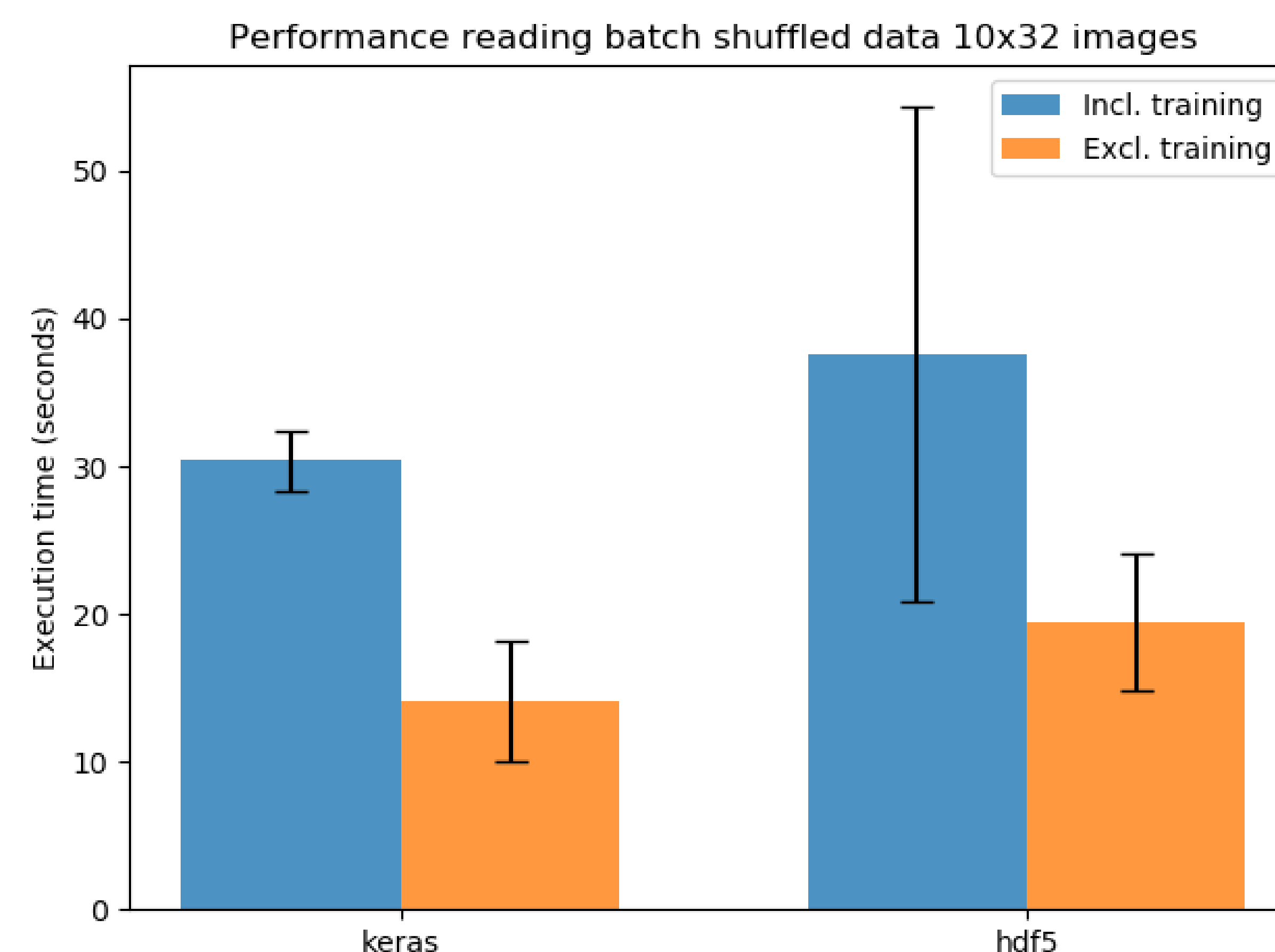
The Main Experiment

Three different approaches to pipelining batches of persistent data to Keras model training with TensorFlow™ backend has been explored:

tf.data: TensorFlow™ data API. Iterating over dataset of file paths and repeatedly loading and preprocessing images via tf.io.read_file method. Not compatible with Azure Blob Storage. Batch shuffling not available.

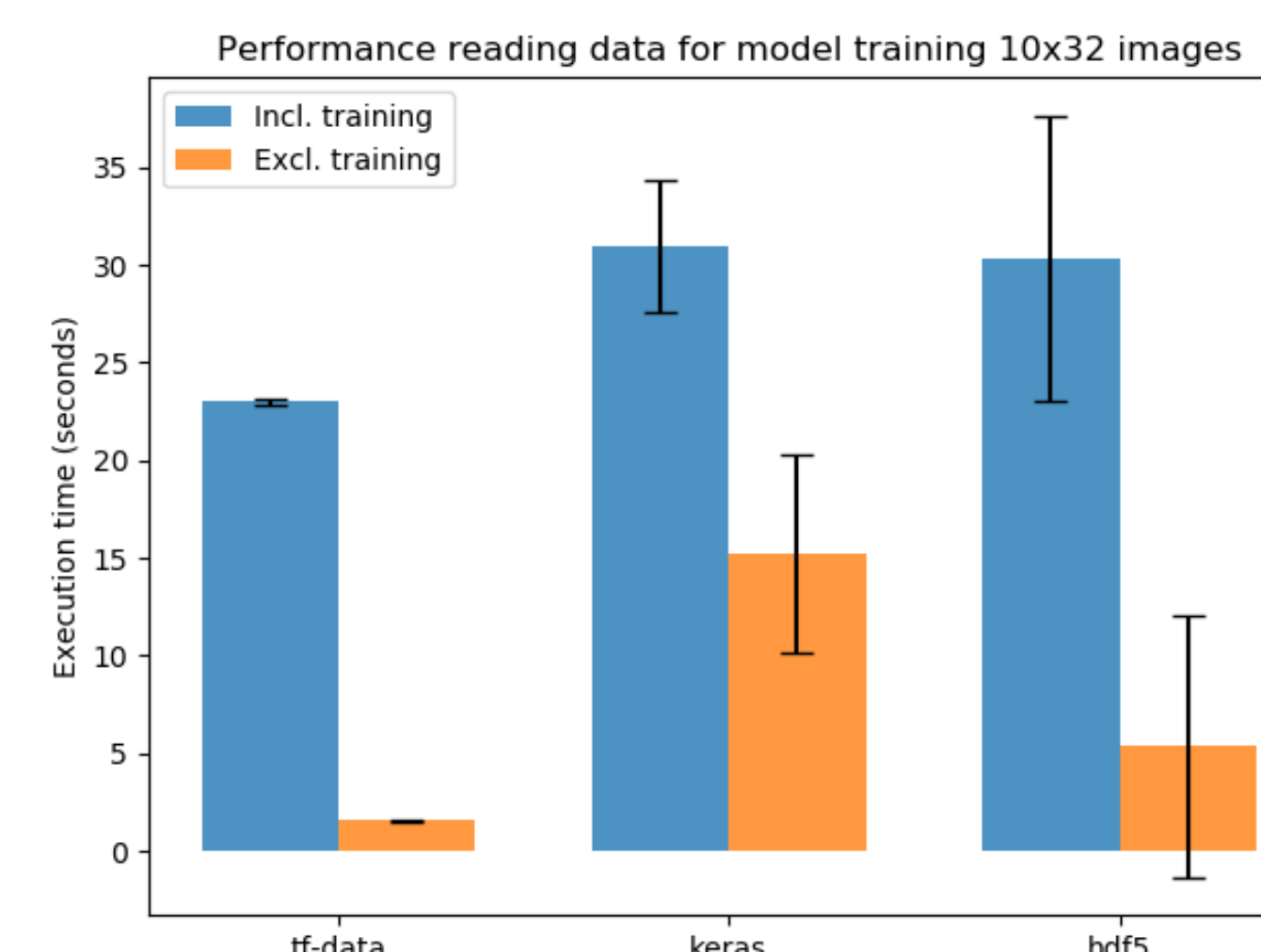
Keras: Custom Keras sequence generator to load and preprocess images batch by batch from Azure Blob Storage via imageio.imread method.

HDF5: Custom Keras sequence generator to load preprocessed arrays of pixel intensities batch by batch from HDF5 file (Hierarchical Data Format). Not compatible with Azure Blob Storage.



Above: On average blob loading and preprocessing with Keras sequence generator is faster than reading preprocessed data stored in HDF5 file on file share.

Right: If sequential load is an option tf.data API is very efficient.



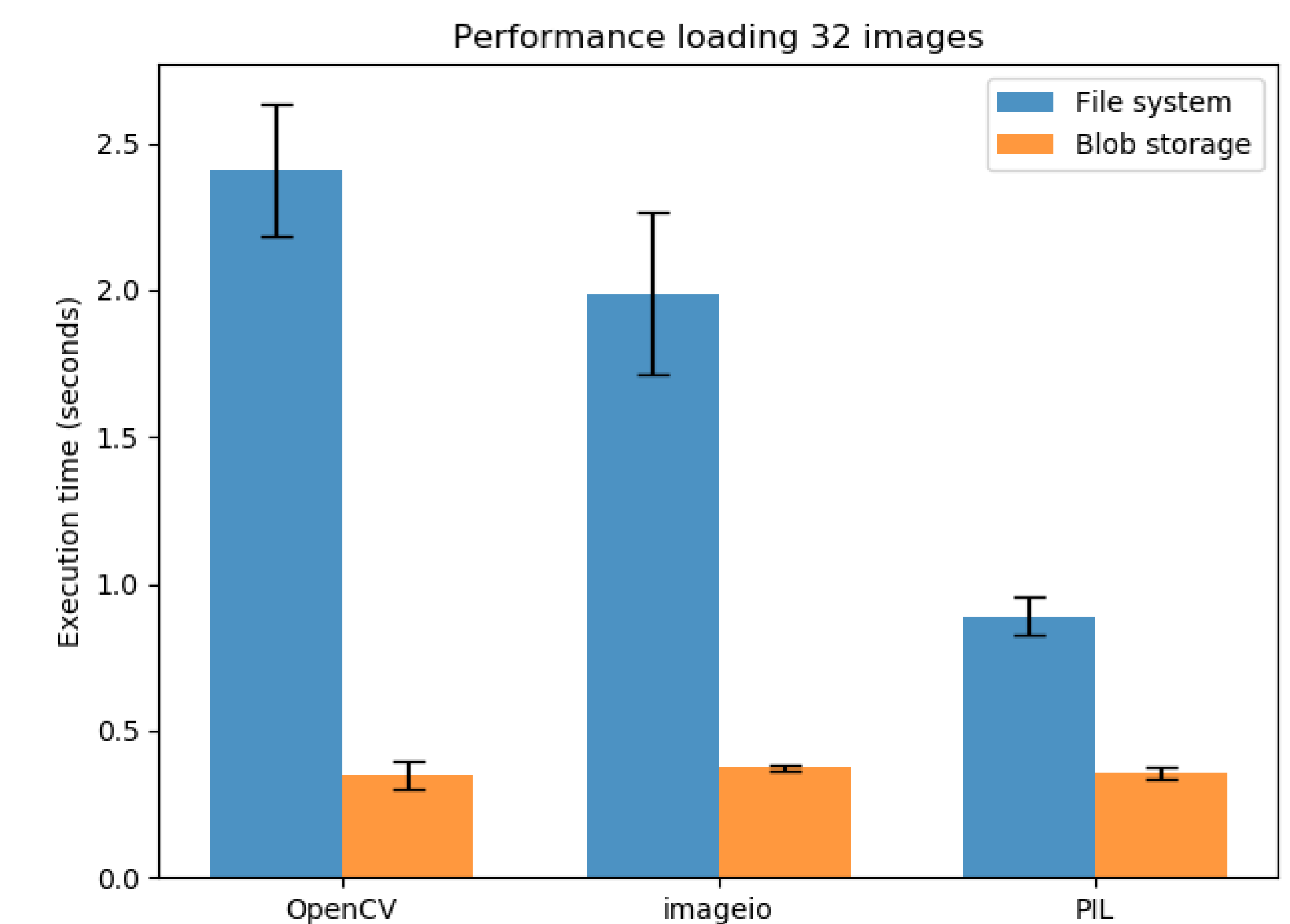
Loading Images in Python

To create an HDF5 file of preprocessed images, and in order to load and preprocess data on the fly with the custom Keras sequence generator, we have explored different Python libraries to load images:

OpenCV: Open Source Computer Vision Library. Cross platform.

PIL: The Python Image Library. Through pillow package.

imageio: Python library providing an easy interface to read and write a wide range of image data.



Above: The results shows that loading blobs is much faster than loading jpegs from file shares.

Experiment details

In the experiments we are measuring performance as the time it takes to perform a task. We are reporting mean +/- standard deviation of 10 repetitions.

The work is performed on an Azure Data Science Virtual Machine of type NC6s-v2, with 6 virtual CPUs, 1 NVIDIA Tesla P100 GPU and 112 GiB Ram.