TensorBoard Reducer: A Python package for aggregating TensorBoard logs

2022-08-21

Summary

TensorBoard Reducer is a `pip`-installable package to compute statistics (`mean`, `std`, `min`, `max`, `median` or any other `numpy` operation) [1] of multiple TensorBoard runs and export the results back to disk as either new TensorBoard logs or CSV / JSON / Excel files. It is aimed at ML researchers dealing with large numbers of training runs. These commonly arise when e.g.

1. training regular or deep [2] model ensembles,
2. measuring epistemic uncertainty, or
3. when developing new model architectures / techniques and the goal is to reduce noise in loss/accuracy/error curves and establish stronger statistical significance of potential performance improvements.

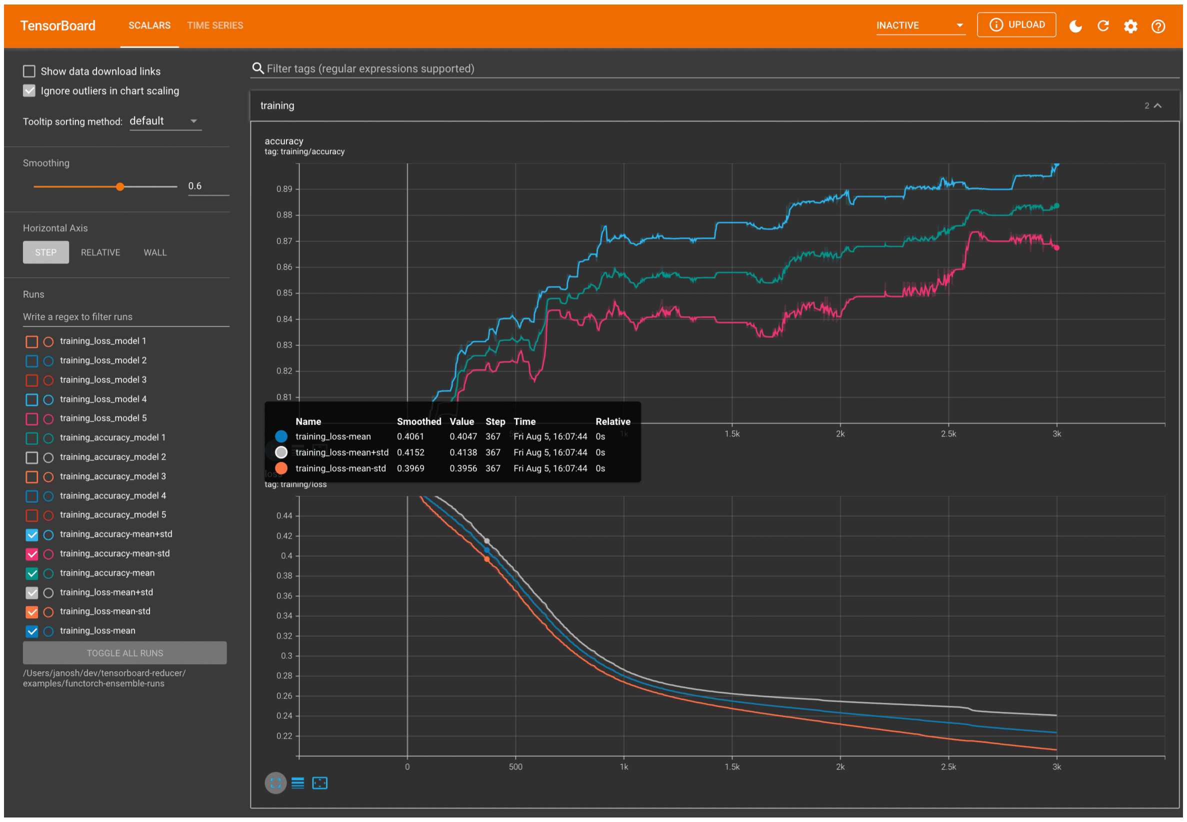
Statement of need

Over the past decade, artificial neural networks have proven a very versatile and scalable optimization technique. An entire industry has sprung up around them in record time, not least thanks to the high-quality open-source tooling and infrastructure built partly by academics [3] and even more so by big tech (e.g. PyTorch by Meta [4], TensorFlow [5] and more recently JAX [6] by Google. While the upfront investment of developer time and computational resources required to build such tools is immense, they make it possible for researchers from all fields of science to train and deploy large models without the need to become or hire full-time engineers themselves, leading to high profile breakthroughs in several domain problems such as protein folding [7] and solving the fractional electron problem [8] in DFT.

However, some aspects surrounding the application of these models have yet to catch up to their meteoric rise. Two we want to highlight are aggregate analysis and uncertainty estimation.

While tools like TensorBoard [5] and Weights and Biases [9] and others go a long way towards addressing the monitoring and debugging of NNs training as well as inspecting their output, they don't allow aggregation of multiple runs. Similarly, uncertainty estimation remains a largely unsolved problem [10] in neural networks. Yet to use them effectively and deploy into real-world applications, knowing when their predictions can be trusted and when to be wary is essential. While research into Bayesian deep learning is ongoing and may one day become the go-to solution, end-to-end learning of high-dimensional probability distributions over model weights remains a fragile and costly technique to this day. The equally expensive yet conceptually much simpler baseline technique of ensemble models remains hard to beat with approximate Bayesian methods [2]. In this setting one trains multiple independent copies of a model from different random initializations to garner a few glimpses into different regions of the loss landscape. The mean prediction of such ensembles tends to outperform each individual model's accuracy and also yields an epistemic uncertainty estimate from the variance across single-model predictions.

We believe the extra work of training model ensembles should be more prevalent across most regions of ML (with the exception of large language models due to the immense training cost) and offer TensorBoard Reducer as a tool to make the analysis of such ensembles easier.



☐ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis: STEP | RELATIVE | WALL

Runs
Write a regex to filter runs
☐ training_loss_model 1
☐ training_loss_model 2
☐ training_loss_model 3
☐ training_loss_model 4
☐ training_loss_model 5
☐ training_accuracy_model 1
☐ training_accuracy_model 2
☐ training_accuracy_model 3
☐ training_accuracy_model 4
☐ training_accuracy_model 5
☑ training_accuracy-mean+std
☑ training_accuracy-mean-std
☑ training_accuracy-mean
☑ training_loss-mean+std
☑ training_loss-mean-std
☑ training_loss-mean

/Users/janosh/dev/tensorboard-reducer/examples/functorch-ensemble-runs

	Name	Smoothed	Value	Step	Time	Relative
●	training_loss-mean	0.4061	0.4047	367	Fri Aug 5, 16:07:44	0s
●	training_loss-mean+std	0.4152	0.4138	367	Fri Aug 5, 16:07:44	0s
●	training_loss-mean-std	0.3969	0.3956	367	Fri Aug 5, 16:07:44	0s

Figure 1: Mean and standard deviation computed using `tensorboard-reducer` and exported back to TensorBoard event files for the loss and accuracy curves of an ensemble model consisting of 5 `functorch` MLPs trained in parallel (see `functorch_mlp_ensemble`).

Features and Application

Specifically, it designed to make the process of aggregating the results of related training runs fast and flexible. Built on top of Numpy and Pandas, it is well-integrated into the NumFOCUS stack, supporting many aggregation operations such as `mean`, `std`, `min`, `max`, `median` (see `numpy.statistics`) and data export options such new TensorBoard event files as well as CSV / JSON / Excel data files (see `pandas.io`) which can easily be extended should the need arise. It has comprehensive test coverage (98% at time of writing), doc string for all of the public API as well as 3 example notebooks demonstrating various use cases that can be launched in Binder with a single click.

1. **Basic Python API Demo**

   launch binder   View on GitHub

   Demonstrates how to work with local TensorBoard event files.

2. **Functorch MLP Ensemble**

launch binder View on GitHub

Shows how to aggregate run metrics with TensorBoard Reducer when training model ensembles using `functorch`.

3. **Weights & Biases Integration**

   launch binder View on GitHub

   Trains PyTorch CNN ensemble on MNIST, logs results to WandB, downloads metrics from multiple WandB runs, aggregates using `tb-reducer`, then re-uploads to WandB as new runs.

Acknowledgements

JR acknowledges support by the German Academic Scholarship Foundation (Studienstiftung). JR would also like to thank all contributors that reported bugs and suggested features on GitHub.

References

[1] C. R. Harris et al., *Array Programming with NumPy*, Nature **585**, 357 (2020).

[2] B. Lakshminarayanan, A. Pritzel, and C. Blundell, *Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles*, (2016).

[3] M. Fey and J. E. Lenssen, *Fast Graph Representation Learning with PyTorch Geometric*, arXiv:1903.02428 (2019).

[4] A. Paszke et al., *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, arXiv:1912.01703 (2019).

[5] M. Abadi et al., *TensorFlow: Large-scale Machine Learning on Heterogeneous Systems*, (2015).

[6] J. Bradbury et al., *JAX: Composable Transformations of Python+NumPy Programs* (2018).

[7] J. Jumper et al., *Highly Accurate Protein Structure Prediction with AlphaFold*, Nature **596**, 583 (2021).

[8] J. Kirkpatrick et al., *Pushing the Frontiers of Density Functionals by Solving the Fractional Electron Problem*, Science **374**, 1385 (2021).

[9] L. Biewald, *Weights and Biases*, (2020).

[10] A. Kendall and Y. Gal, *What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?*, (2017).