The Journal of Open Source Software

Elephas: Distributed Deep Learning with Keras & Spark

Max Pumperla[1,2] **and Daniel Cahall**[3]

1 IU Internationale Hochschule, Erfurt, Germany **2** Anyscale Inc, San Francisco, USA **3** Independent researcher, Philadelphia, USA

DOI: 10.21105/joss.04073

Software

- Review
- Repository
- Archive

Editor: Patrick Diehl

Reviewers:

- @sepandhaghighi
- @nmoran

Submitted: 18 November 2021
Published: 15 December 2022

Summary

Elephas is an extension of Keras, which allows you to run distributed deep learning models at scale with Apache Spark. It was built to allow researchers and developers to distribute their deep learning experiments as easily as possible on a Spark computer cluster. With elephas, researchers can currently run data-parallel training of deep learning models with distribution modes as suggested in (Dean et al., 2012), (Recht et al., 2011) and (Noel & Osindero, 2014). Additionally, elephas supports distributed training of ensemble models. Until version 2.1., elephas also supported distributed hyper-parameter optimization of Keras models.

Elephas keeps the simplicity and high usability of Keras, thereby allowing for fast prototyping of distributed models. When ready, researchers can then scale out their experiments on massive data sets. Elephas comes with full API documentation and examples to get you started. Initiated in late 2015, elephas has been actively maintained since then and has reached maturity for distributed deep learning on Spark.

Statement of need

Modern deep learning solutions require ever more data and computation power. A study by OpenAI suggests that the number of operations needed for AI systems, by now measured in *petaflops*, shows exponential growth and has in fact been doubling every 3.4 months since 2012. This vastly outweighs the computational gains to expect from single machines, even according to the most optimistic version of Moore's law. There's a clear need for solutions that can scale to compute clusters. While some large companies have such large deep learning models that they have to resort to distributing the models themselves (model-parallelism), for most researchers and the majority of companies, compute time and data volume are the predominant bottlenecks.

Apache Spark has established itself as one of the most popular platforms for distributed computing. However, its native machine learning (ML) capabilities are limited by design. Spark excels when transforming massive datasets and applying built-in ML algorithms with Spark MLlib, but does not support implementation of custom algorithms with deep learning frameworks such as Google's TensorFlow and specifically its convenient Keras API.

Elephas was the first open-source framework to support distributed training with Keras on Spark. It was followed by libraries such as Yahoo's TensorFlowOnSpark, which does not follow Keras API design principles, later on. In recent years, other popular distributed deep learning frameworks have emerged, such as the powerful Horovod (Sergeev & Balso, 2018), which initially did not have Spark support. BigDL (Dai et al., 2019) is another such framework worth mentioning, especially in conjunction with Intel's Analytics Zoo. Elephas is still in active use and has been leveraged by millions of users in the Python deep learning community.

Design and API

Elephas has a tight integration with many of Spark's core abstraction layers. Apart from the basic integration for Spark's resilient distributed datasets (RDDs), elephas works with MLlib models, with Spark ML estimators, can train ensemble models and run distributed inference.

Basic Spark training

Elephas' core abstraction to bring Keras models to Spark is called `SparkModel`. To use it, you define a Keras model, load your data, define your `SparkModel` with the training mode and update frequency of your choice, and then fit your model on distributed data:

```
from elephas.spark_model import SparkModel
from elephas.utils.rdd_utils import to_simple_rdd
from pyspark import SparkContext, SparkConf
from tensorflow.keras.models import Sequential

# Define Spark context
conf = SparkConf().setAppName('Elephas_App')\
                  .setMaster('local[8]')
sc = SparkContext(conf=conf)

# Define Keras model
model = Sequential()
model.add(...)
model.compile(...)

# Load training data
x_train, y_train = ...

# Convert to Spark RDD
rdd = to_simple_rdd(sc, x_train, y_train)

# Define Elephas model
spark_model = SparkModel(model, frequency='epoch', mode='asynchronous')

# Run distributed training
spark_model.fit(rdd, ...)
```

Afterwards your training job can be submitted to any Spark cluster like this:

```
spark-submit --driver-memory 1G ./your_script.py
```

Distributed Inference

When your model `spark_model` has finished training, you can easily run distributed inference on your test data or compute evaluation metrics on it as follows.

```
# Load test data
x_test, y_test = ...

# Perform  distributed inference
predictions = spark_model.predict(x_test)

# Run distributed evaluation/scoring
evaluation = spark_model.evaluate(x_test, y_test)
```

Spark MLlib and ML integrations

To leverage Spark's `LabeledPoint` RDD to encode features and labels for a prediction task, you can use helper functions provided by elephas and then train a so-called `SparkMLlibModel` on your data for your Keras `model`.

```
from elephas.utils.rdd_utils import to_labeled_point
from elephas.spark_model import SparkMLlibModel

# Create a LabeledPoint RDD
lp_rdd = to_labeled_point(sc, x_train, y_train, categorical=True)

# Define and train a SparkMLlib model
spark_model = SparkMLlibModel(model, frequency='batch', mode='hogwild')
spark_model.train(lp_rdd, ...)
```

Likewise, to create a Spark ML `Estimator` to fit it on a Spark `DataFrame`, you can use elephas' s `SparkEstimator`.

```
from elephas.ml.adapter import to_data_frame
from elephas.ml_model import ElephasEstimator

# Create a Spark DataFrame
df = to_data_frame(sc, x_train, y_train, categorical=True)

# Define and fit an Elephas Estimator
estimator = ElephasEstimator(model, ...)
fitted_model = estimator.fit(df)
```

To summarize, elephas provides you with training and evaluation support for custom Keras models for many practical scenarios and data structures supported on a Spark cluster.

Acknowledgements

We would like to thank all the open-source contributors that helped making `elephas` what it is today.

References

Dai, J. (Jinquan)., Wang, Y., Qiu, X., Ding, D., Zhang, Y., Wang, Y., Jia, X., Zhang, L. (Cherry)., Wan, Y., Li, Z., Wang, J., Huang, S., Wu, Z., Wang, Y., Yang, Y., She, B., Shi, D., Lu, Q., Huang, K., & Song, G. (2019). BigDL: A distributed deep learning framework for big data. *Proceedings of the ACM Symposium on Cloud Computing*, 50–60. https://doi.org/10.1145/3357223.3362707

Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., Le, Q., & Ng, A. (2012). Large scale distributed deep networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/6aca97005c68f1206823815f66102863-Paper.pdf

Noel, C., & Osindero, S. (2014). *Dogwild! – distributed hogwild for CPU & GPU*.

Recht, B., Re, C., Wright, S., & Niu, F. (2011). Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems*

(Vol. 24). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2011/file/218a0aefd1d1a4be65601cc6ddc1520e-Paper.pdf

Sergeev, A., & Balso, M. D. (2018). Horovod: Fast and easy distributed deep learning in TensorFlow. *arXiv Preprint arXiv:1802.05799*.