Deep Learning has advanced dramatically over the last ten years [1-7] and shown remarkable achievements in areas including the recognition and classification of images, speech and handwriting. Some emerging critical challenges include the development of a general easy to use library ton which an environment can be built to explore and develop new learning networks. The networks can address new problem areas such as analysis of exascale visualizations and new algorithms for different(all) parts of the system. At the highest level, one needs to evaluate Stochastic Gradient Descent and other Steepest Descent approaches to see if they can be used with larger sample sizes so that efficient parallelism over input data sets such as images is possible. Currently the pioneering work on 64 GPU’s [] is harder to run on larger systems as it exploits parallelism over pixels and communication overheads grow as number of pixels assigned to each node declines; a well-known parallel computing issue. At a lower level we need to enable rapid experimentation with different network structures – number of layers and linkage between layers. The first step is development of an attractive “problem solving environment” for this problem that makes it as convenient as possible to modify algorithms and develop new applications while executing each experiment quickly. We have started to address this in collaboration with Stanford/Baidu and the University of Tennessee at Knoxsville. This initial work is focused on enhancing the popular Caffe system from UC Berkeley. It will provide optimized kernels that run in parallel on systems with multiple CPU’s and GPU’s. We will package these kernels in a python front-end that can support the specification of neural and manage its execution on an HPC system as well as its visualization/analysis. This problem is an excellent place to test and compare the different approaches to exascale runtime and one extension is to also support kernels using the ParalleX runtime developed in CREST center at Indiana University. Further different execution frameworks other than Python could be considered as it is not possible for example to use a Python scripting approach without also parallelizing all the data analysis steps one needs to evaluate each execution. At the most ambitious this could involve a parallel Python invoking the optimized runtime described above. In any case one needs modules for the final analysis stage that run in parallel and handle large networks such as recent run with 11 billion parameters. Our current work aims at producing a single benchmark based on ImageNet. This is not sufficient to even provide a proper set of requirements. We need to survey deep learning applications of interest to DoE and define both system requirements and benchmarks.

My group has pioneered the development of HPC-ABDS which is an integration of HPC and Apache (and other) open source big data technologies ABDS; our current catalog has identified 200 software subsystems divided into 17 layers summarized in figure. The key idea is that new HPC ideas should be developed so they integrate well with ABDS rather than competing with this rapidly developing software stack which has a clear vitality and innovation with a sustainable software model. In particular I have shown that previous standalone enhanced versions of MapReduce can be replaced by a Hadoop plug-in that gives one data abstractions useful for science, high performance iteration and communication using best available (MPI) approaches. We suggest that deep learning be set up in this architecture exploiting resource managers like Yarn, as well as storage models such as HDFS and MongDB/Hbase. This would enable one to look at approaches such as Apache Pig ()data parallel language) and Crunch (workflow) to drive the deep learning explorations.

