Fast and Accurate Training of an AI Radiologist on Intel Xeon-based Dell EMC Supercomputers

Lucas A. Wilson, Vineet Gundecha, Srinivas Varadharajan, Alex Filby and Quy Ta
HPC & AI Engineering, Dell EMC
Email:{luke_wilson, vineet_gundecha, s_varadharajan, alex_filby, quy_ta}@dell.com

***Abstract*—The health care industry is expected to be an early adopter of AI and deep learning to improve patient outcomes, reduce costs, and speed up diagnosis. We have developed models for using AI to diagnose pneumonia, emphysema, and other thoracic pathologies from chest x-rays. Using the Stanford University CheXNet model as inspiration, we explore ways of developing accurate models for this problem with fast parallel training on Zenith, the Intel Xeon-based supercomputer at Dell EMC's HPC and AI Innovation Lab. We explore various network topologies to gain insight into what types of neural networks scale well in parallel and improve training time from days to hours. We then explore transferring this learned knowledge to other radiology subdomains, such as mammography, and whether this leads to better models than developing subdomain models independently.**

***Index Terms*—Deep Learning, Medical Imaging, Radiology, Distributed Training, Benchmarking, Best Practices**

I. Introduction

The field of artificial intelligence (AI) has seen a resurgence in interest in recent years thanks first to the application of statistical machine learning (ML) techniques to Big Data, and more recently through the application of neural networks trained to extract useful information from very unstructured data. This unstructured data - in the form of images, audio snippets and spoken language, video, social media posts, and many other forms - has more inherent value than was previously appreciated, and the application of neural networks to extracting information from these types of data is the subject of a great deal of both academic study and industry/commercial interest.

Deep Learning (DL) - the practice of training and deploying artificial neural networks models trained on this unstructured data - is a critical component of academic and industry research, corporate strategy, and business models. Outside of large HPC centers and cloud service providers, the practice of training DL models is still ad hoc and mostly driven on the data scientist's laptop. As DL becomes an increasingly viable solution for multiple applications for enterprise customers and startup companies, the ability to rapidly train new models, retrain existing models with new data, and quickly apply these models to gain insight and create competitive advantage means that DL will move off of the laptop and into the data center.

II. Background and Motivation

Artificial neural networks have been an area of research since the 1950s [1], but modern interest and the explosion of investment and research in deep learning can be thought to begin with the development of AlexNet [2], which not only achieved state of the art classification performance (for the time) on the ImageNet [3] data set, but introduced into the mainstream both the use of the computationally-simplified Rectified Linear Unit (ReLU) [4] as well as the use of graphics processing units (GPUs) for accelerated linear algebra transformations for training the network by backpropagation [5].

While incredible progress has been made in image classification (see [2], [6], [7], among others) and object detection [8]–[10], deep learning has been making headway in many other intractable problems including voice translation and synthesis [11], [12], natural language processing [13]–[15], sentiment analysis [16]–[18], game play [19]–[21], and autonomous driving/control [22]–[25].

The first model we look at is the CheXNet model [26] from Stanford University. CheXNet provides state-of-the-art machine detection of pneumonia, outperforming a panel of radiologists [26]. We use the same National Institutes of Health (NIH) ChestX-ray data set [27], which consists of 112,120 images labeled with 14 different thoracic diseases (plus "No Findings"), including pneumonia (see Table I and Figure 2). All images are labelled with either single or multiple pathologies, making this a multi-label classification problem. Images in the ChestX-ray data set are 3 channel (RGB) with dimensions 1024 x 1024.

We then take the CheXNet model and train it on the CBIS-DDSM dataset. Curated Breast Imaging Subset of DDSM is an updated and standardized version of the Digital Database for Screening Mammography (DDSM). The DDSM is a database of 2,620 scanned film mammography studies. It contains normal, benign, and malignant cases with verified pathology information. We train the model to detect if a given scan contains a tumour (malignant or benign). The idea here is to see if features learned by the model on a set of Chest X-ray scans are helpful in training a model to detect breast cancer tumours. This could be helpful for cases were available training data is scarce.

Improving the speed and accuracy of disease detection is one of the primary motivators for the adoption of AI in health care, as it can lead to dramatically improved patient outcomes with reduced wait time and reduced cost. This motivation translates to many other industries, as well. Improving time-to-solution, with either no or minimal change in model accuracy, is critically important if DL is to become part of any company's business strategy.

TABLE I: Disease Frequencies in ChestX-ray Data Set

Disease	Images	Percentage
Atelectasis	11535	10.28
Consolidation	4667	4.16
Infiltration	19871	17.72
Pneumothorax	5298	4.72
Edema	2303	2.05
Emphysema	2516	2.24
Fibrosis	1686	1.50
Effusion	2516	2.24
Pneumonia	1353	1.20
Pleural Thickening	3385	3.01
Cardiomegaly	2772	2.47
Nodule	6323	5.64
Mass	5746	0.05
Hernia	227	0.20
No Findings	60412	53.88
Totals	**112,120**	**100**

The CheXNet model is an example of transfer learning [29]–[31], which is the process of using a model which has already been trained for a potentially unrelated application, and using that model to jump start the learning process for a new application. In the case of CheXNet, a 121-layer DenseNet [32] architecture has been previously trained on the ImageNet data set [3]. This trained model provides a better-than-random starting point for training CheXNet to identify features in chest xray images, and helps to reduce the number of epochs of training necessary to converge to a functional model.

III. Identifying Pathologies in Chest Xrays

Our first task was to develop a topology which was able to accurately identify thoracic pathologies in chest xrays, and then optimize the runtime environment and the topology in order to improve both single node and distributed, parallel training performance.

A. Single-node Optimization

In this section, we look at how to not only the performance of the training phase from a time-to-solution perspective, but also whther improving the performance with these runtime tweaks has any effect, positive or negative, on the quality of the final model.

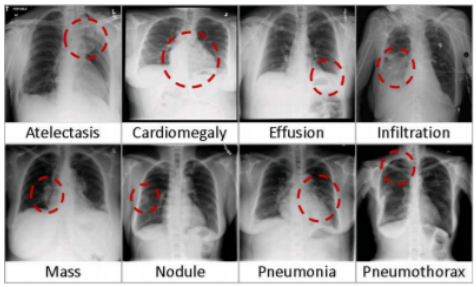


Fig. 1: Sample of Thoracic Diseases Labeled in the NIH ChestX-ray Data set [27]

1) Adjusting Batch Size: Artificial neural networks are trained by minimizing a loss function which measures the deviation of the model output from the ground truth. Mathematically, this can be expressed as:

$$\min_x f(x) := \frac{1}{N}\sum_{i=1}^{N} f_i(x)$$

where $f_i(x)$ is the loss for training example i $\in \{1, 2...N\}$, and x is the vector of parameters. Stochastic Gradient Descent (SGD) and its variants are usually employed for optimizing the loss function. These methods work by iteratively taking small steps in the direction of negative gradient as follows:

$$x_{k+1} = x_k - \alpha(\frac{1}{|B_k|}\sum_{i\in B_k} \nabla f_i(x_k))$$

where B_k is the batch sampled from the training set at iteration k and α is the learning rate. The batch gradients are an approximation to the true gradient and are therefore inherently noisy. The batch size is a critical parameter for model accuracy as well as for distributed performance. Larger batch sizes offer more parallelism. However, increasing the batch size negatively affects the generalization performance of the model [33]. We investigate the model performance with increasing batch sizes. To account for the fewer parameter updates per data pass with increasing batch sizes, we double the learning rate every time the batch size is doubled. We use the Adam [34] optimizer with a base learning rate of 0.0005 and a batch size of 8. Figure 3 shows the effect of batch size on the model performance. Batch size 8 and 16 give the best AUC values for all the 14 classes. The model accuracy decreases as the batch size is increased. However, we see that we get better throughput with larger batch sizes. Figure 5 details the time taken per epoch for various batch sizes.

2) Adjusting Number of Threads: Training a neural network model on a single-node comes with a challenge of training time. Thread based parallelism could help to improve the time taken to train a model. Hence, selecting the proper number of threads and the way to bind threads to physical cores is a significant factor while training on a single-node.

In this section we show how the training time and AUC values are affected when different number of threads are chosen. These experiments were performed by setting the following environment variables:

Fig. 2: Sample of a scan with a marked tumour in the CBIS-DDSM data set [28]

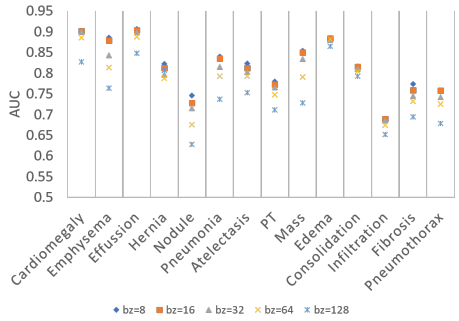


Fig. 3: AUC values for each category for different batch sizes

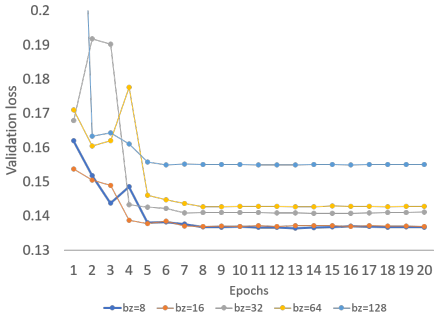


Fig. 4: Validation loss for different batch sizes. We can see that the validation loss increases with batch size

```
export KMP_AFFINITY=granularity=fine,compact,1,0
export KMP_BLOCKTIME=1
export OMP_NUM_THREADS=<No. of Threads>
```

Figure 5 shows the average training time per epoch for various batch sizes and the number of threads (single process). Based on the observation, choosing the thread count of 32 seems to be efficient for most of the batch sizes. Anomalies were detected for batch size 32. Choosing any thread count above 16 for this particular batch size seemed to increase the corresponding training time. Further investigation is required to understand the cause. Figure 6 shows much clearer trends for thread counts 16 and 32 across various batch sizes.

Another important aspect to report is the effect of the number of threads on the AUC value. Figure 7 shows the AUC values for different pathologies and the number of threads. The resulting AUC values for different thread counts are

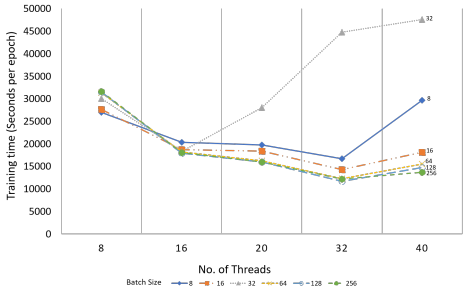


Fig. 5: Training time per epoch for different thread counts and batch sizes

overlapping. Hence, it is evident from the plot that adjusting the number of threads doesn't affect the AUC value.

All results reported in this subsection are based on 5 runs. Figure 8 shows the box plot with detailed distribution. It can be seen that the smaller thread runs have less variation when compared to using all 40 threads, potentially due to OS jitter. It can be observed that there are very less variation for most of the runs, this may be be attributed to appropriate thread binding and number of threads for the particular batch size. Further investigation will allow us to determine the exact cause of this behavior.

B. Multi-node Optimization

The long-term goal is efficient scale-out parallel training in both data-parallel (i.e., where the training data set is distributed across multiple nodes) and model-parallel (i.e., where the neural network is distributed across multiple nodes) forms. Our work focuses exclusively on data-parallel training, using Uber's Horovod [35] package. Horovod takes advantage of MPI [36] collectives which have been heavily tuned for performance on large-scale HPC systems.

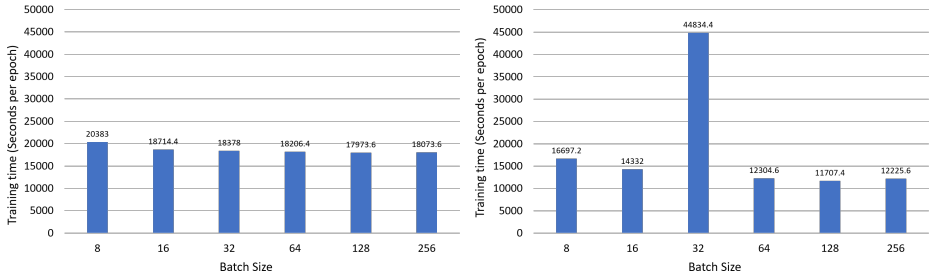
In this section we report results for parallelizing CheXNet training using Horovod. We are interested in comparing not only scale-out training performance as measured by throughput (seconds per epoch), but also in the convergence and accuracy of the model. AUC numbers reported for parallel training jobs are after a fixed number of epochs (10), and throughput numbers are based on the Keras-reported training time after each epoch.

For our parallel performance tests, we are evaluating the strong scaling of the training phase using two different local batch sizes (LBZ): 4 and 16. The reason for this is to compare the throughput and accuracy of models based on their global batch size (GBZ), which is the product of the LBZ and the number of processes. This way we can compare the GBZ to the batch size (BZ) used in the single-process tests (see Section III-A).

Figure 9 shows the seconds per epoch for training different GBZ in three different configurations (single-node, LBZ=4, LBZ=16) for GBZ ranging from 32 to 512. The number of MPI processes is the GBZ divided by the LBZ. For example, the tests run with GBZ=512 used 128 processes for the LBZ=4 case, and 32 processes for the LBZ=16 case. All experiments were run with 2 MPI processes per node, with each MPI process allowed to spawn up to 20 OpenMP threads.

For the most part, AUC values for all parallel training experiments closely tracked one another, with many of the categorical accuracy values showing little variation between configurations (see Figure 10 for individual configuration results and summary statistics, respectively). For nearly every category, larger GBZ experiments converged to poorer accuracy than smaller GBZ experiments. This trend becomes more apparent as one scales out (thereby increasing GBZ). For the GBZ=512 experiments the LBZ=4 experiment did not generalize at all, with each category's accuracy no better than a coin flip.

To get a sense of why the LBZ=4,GBZ=512 experiment did not generalize, we looked at the learning progression of



(a) 16 Threads (b) 32 Threads

Fig. 6: Runtime for 16 & 32 threads across various batch sizes

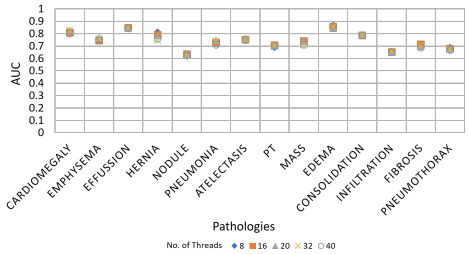


Fig. 7: Comparing the AUC values of pathologies for different thread counts with batch size 64

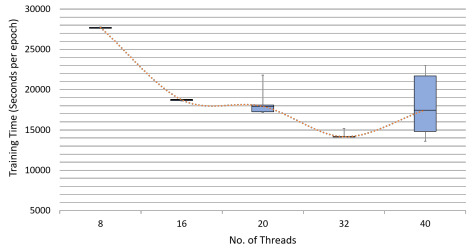


Fig. 8: Performance test for different thread counts with batch size 16

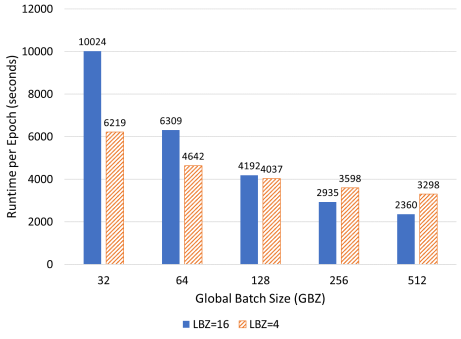


Fig. 9: Runtime Performance for Varying GBZ

the LBZ=4,GBZ=512 experiment in comparison (Figure 11) with both the best performing single-process experiment (BZ=8), and the worst performing single-process experiment (BZ=256). Both of the single-process experiments converged to a solution with validation loss < 0.16 (see Figure 3) in fewer than 7 epochs. And while the LBZ=4,GBZ=512 experiment did converge (little learning occurred after epoch 6), the model was unable to reliably predict in any category.

C. *Improving Multi-node Accuracy*

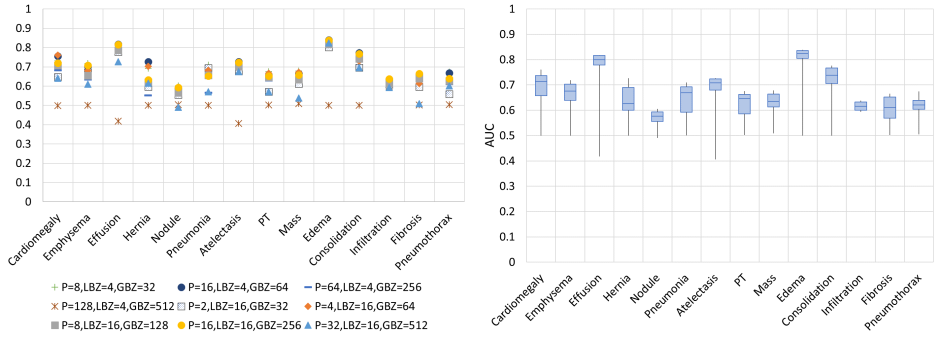
Our initial suspicion was that perhaps the learning optimizer used for our experiments (Adam) was too fast/aggressive for large-batch training. To determine if this was the case, we tested additional optimizer algorithms (AdaDelta, SGD with momentum). Loss values at each epoch are shown in Figure 12.

AdaDelta significantly outperforms Adam or SGD with momentum, with the best attempt achieving a loss of 0.159. This is a significant improvement in validation loss over parallel trained with Adam (0.159 vs. 0.59), and at greater scale than with Adam (GBZ=4096 vs. GBZ=512). However, the validation loss is still too high for this model, given that this is a multi-classification problem with 14 different labels (see Figure 13).

Our next attempt to improve the accuracy of the distributed large-batch trained model was to look closely at the topology of the network. DenseNet consists of many repeating blocks of convolutional and batch normalization layers. Because we are attempting to increase the batch size - both per process and globally through parallelism, we determined that the prevalence of batch normalization layers may be the limiting factor in scaling this topology.

To test this hypothesis, we decided to start with a different topology without any batch normalization: VGG-16 [37]. We used Keras' VGG-16 topology pretrained with ImageNet, and built a classification layer with 14 categories (replicating the method used to build the DenseNet variant).

We tested VGG-16 at various scales using the AdaDelta optimizer, along with a 5 epoch warmup to a learning rate of 0.5 and a learning rate decay factor of 0.1 per epoch starting at epoch 50. We observed increased model accuracy (see Figures 14a and 14b) over the DenseNet topology trained in parallel, and saw improve accuracy over our benchmark



(a) AUC Values

(b) Summary Statistics

Fig. 10: Accuracy of Parallel Trained Models

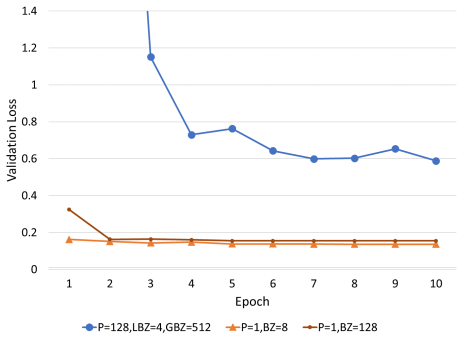


Fig. 11: Validation Loss After Each Training Epoch for GBZ=512 and Best/Worst Single-process Models

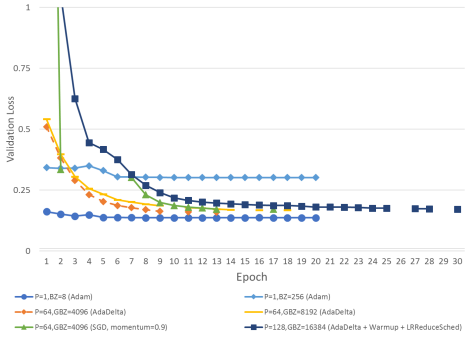


Fig. 12: Validation Loss with Various Optimizers

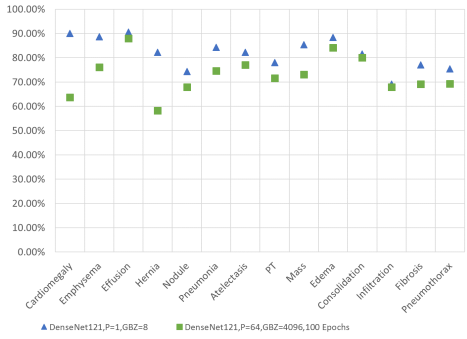


Fig. 13: Accuracy of Single-process DenseNet vs. Parallel DenseNet (GBZ=4096)

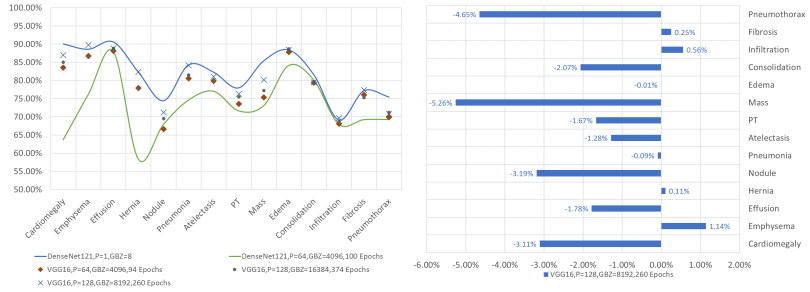
single-process, BZ=8 DenseNet model in 4 categories (Figure 14b). With the VGG topology, which doesn't include batch normalization, we were able to train at larger batch sizes (GBZ=8192) while improving accuracy in several categories over the benchmark DenseNet BZ=8. In categories which did not beat the benchmark, their accuracy was significantly improved over distributed large-batch trained DenseNet variants. In large-batch VGG variants, the worst categorical accuracy (Mass) was less than 6% off from the benchmark.

In addition to being more accurate than distributed large-batch DenseNet, the VGG-16 topology variants are also faster to train. We were able to achieve a speedup of 6.29x over the AdaDelta-optimized DenseNet with GBZ=4096 (P=64, 32 nodes), and a speedup of 293x over the Adam-optimized single-process BZ=8 DenseNet (see Figure 15).

IV. Identifying Tumors in Mammograms

In this section, we investigate if pre-training on chest x-ray images helps in training a model to detect breast cancer tumours. We first train the DensetNet-121 topology on the CBIS-DDSM dataset without pre-training it on the chest X-ray dataset to get a baseline on the model accuracy. The positive (CBIS-DDSM) images had their regions of interest (ROIs) extracted using the masks with a small amount of padding to provide context. Each ROI was then randomly cropped three times into 598x598 images, with random flips and rotations, and then the images were resized down to 299x299. The negative images are random crops of size 598x598, which were resized down to 299x299. The dataset is available at https://www.kaggle.com/skooch/ddsm-mammography/home

The model is trained with a batch size of 16 using the cross-entropy loss function, learning rate of 0.001 and the Adam optimizer for 10 epochs. Learning rate is reduced whenever the validation loss plateaus. All the hyper-parameters remain the same when using pre-training. From Figure 16, we can see that pre-training on the chest x-ray data set helps in faster convergence of the model, and also results in a more stable learning curve. This shows empirically that the features learned by the model from training on a similar but unrelated dataset are transferable to different tasks. Figure 18 shows that even if both models eventually converge to the same accuracy, the pre-trained model gets there faster than the model without



(a) AUC Values (b) Relative Accuracy

Fig. 14: Categorical accuracy of VGG-16 topology vs. DenseNet topology

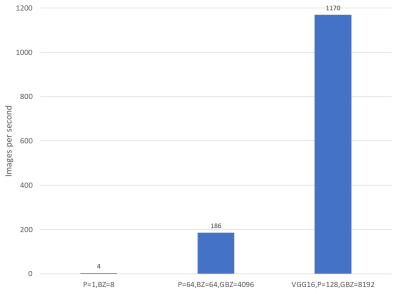


Fig. 15: Training Throughput of DenseNet vs. VGG-16

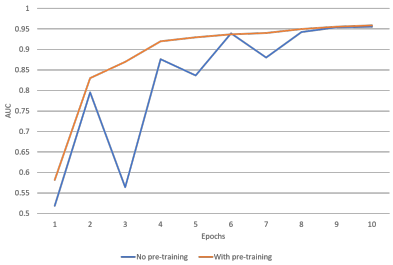


Fig. 17: Comparison of AUC with and without pre-training

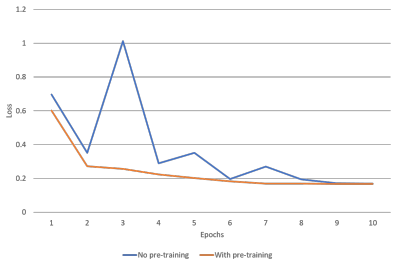


Fig. 16: Comparison of validation loss with and without pre-training

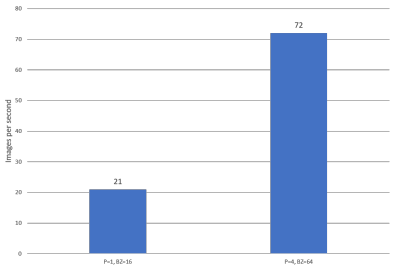


Fig. 18: Training throughput for the mammography model

pre-training. This could potentially speed up the process of training a model.

We then train the model in a distributed setting with 4 MPI processes using Horovod. The global batch size is 64 and the learning rate is 0.004 which is linearly scaled from the single node run. We got an AUC of 0.94 for the pre-trained model which is very close to AUC for the single node run (0.95). However, the AUC for the model without pre-training degraded to 0.9. This suggests that pre-training also helps in model accuracy when training with a large batch size.

V. Future Work and Conclusion

Our work has shown that when developing deep learning models to be trained using parallel distributed training with large batches, several things need to be considered. It is not only important to use a slower optimizer function than would normally be used in a small batch training situation, but it is important to dynamically change the learning rate during the training process. Also, batch normalization can be problematic with performing large batch training, so it should be used sparingly or removed entirely in order to improve convergence and improve training performance.

Other factors had limited or no effect on training accuracy, but did affect the training performance. In particular, being sure to take advantage of multiple threads on a single process had no effect on the eventual model accuracy, yet contributed significantly to improving training performance.

Our work also shows that, while transfer learning (in our case from chest xrays to mammograms) does not necessarily improve the accuracy of the final model, it does smooth the loss landscape considerably during training, and reduces the number of epochs necessary to approach convergence. This is important, as being able to train models in less time by using pre-learned features from other data sets allows models to be refined, retooled, and deployed more quickly.

Going forward, we intend to explore other strategies for performing normalization on training images besides batch normalization. Other approaches, such as group normalization and instance normalization, appear to be promising approaches

for large batch training. We will also look at other learning rate schedule options, such as LARS [38] and collapsed ensembles [39]. We will also investigate if other neural network topologies, such as ResNet [7], might yield more accurate models for these use cases.

REFERENCES

[1] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[4] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[10] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 922–928.

[11] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams *et al.*, "Recent advances in deep learning for speech research at microsoft," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8604–8608.

[12] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[13] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 160–167.

[14] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[15] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[16] C. dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2014, pp. 69–78.

[17] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 959–962.

[18] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification." in *AAAI*, vol. 333, 2015, pp. 2267–2273.

[19] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang, "Deep learning for real-time atari game play using offline monte-carlo tree search planning," in *Advances in neural information processing systems*, 2014, pp. 3338–3346.

[20] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[22] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Computer Vision (ICCV), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2722–2730.

[23] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue *et al.*, "An empirical evaluation of deep learning on highway driving," *arXiv preprint arXiv:1504.01716*, 2015.

[24] R. Hadsell, A. Erkan, P. Sermanet, M. Scoffier, U. Muller, and Y. LeCun, "Deep belief net learning in a long-range vision system for autonomous off-road driving," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 628–633.

[25] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2174–2182.

[26] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya *et al.*, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *arXiv preprint arXiv:1711.05225*, 2017.

[27] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 3462–3471.

[28] A. H. D. R. . Rebecca Sawyer Lee, Francisco Gimenez. Curated breast imaging subset of ddsm. the cancer imaging archive. [Online]. Available: http://dx.doi.org/10.7937/K9/TCIA.2016.7O02S9CY

[29] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 17–36.

[30] L. Y. Pratt, "Discriminability-based transfer between neural networks," in *Advances in neural information processing systems*, 1993, pp. 204–211.

[31] L. Pratt, *Reuse of neural networks through transfer*. Carfax Publishing Company, 1996.

[32] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, no. 2, 2017, p. 3.

[33] J. N. M. S. N. S. Keskar, D. Mudigere and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836, 2016*, 2016.

[34] J. B. DP Kingma, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, 2014.

[35] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.

[36] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the mpi message passing interface standard," *Parallel computing*, vol. 22, no. 6, pp. 789–828, 1996.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[38] Y. You, I. Gitman, and B. Ginsburg, "Scaling SGD batch size to 32k for imagenet training," *CoRR*, vol. abs/1708.03888, 2017. [Online]. Available: http://arxiv.org/abs/1708.03888

[39] V. Codreanu, D. Podareanu, and V. Saletore, "Scale out for large minibatch sgd: Residual network training on imagenet-1k with improved accuracy and reduced time to train," *arXiv preprint arXiv:1711.04291*, 2017.