EasyChair Preprint

№ 14261

Accelerating Financial Statement Audit Procedures: an Examination of Machine Learning for Risk Assessment and Anomaly Detection

Abill Robert

August 2, 2024

Accelerating Financial Statement Audit Procedures: An Examination of Machine Learning for Risk Assessment and Anomaly Detection

Author

Abil Robert

Date: 2 August 2, 2024

Abstract

The increasing volume and complexity of financial data pose significant challenges for traditional financial statement audit procedures, often leading to increased audit time and costs. This research investigates the potential of machine learning to accelerate and enhance the efficiency of financial statement audits, focusing on two key areas: risk assessment and anomaly detection. By leveraging ML's ability to analyze vast datasets and identify patterns, this study explores how auditors can make more informed risk assessments and efficiently detect potential misstatements. The research will examine various ML algorithms, including supervised, unsupervised, and semi-supervised learning techniques, to evaluate their effectiveness in identifying financial anomalies and predicting audit risk factors. Furthermore, the study will address the challenges and limitations of implementing ML in audit procedures, such as data quality, model interpretability, and ethical considerations. This research aims to provide practical insights for audit professionals and contribute to the ongoing dialogue on leveraging advanced technologies to improve audit quality and efficiency in the face of evolving financial reporting landscapes.

Introduction:

The proliferation of machine learning (ML) applications has ushered in an era where intelligent systems are seamlessly integrated into everyday devices. From smartphones and wearable technology to IoT devices and edge computing nodes, the demand for real-time ML processing on resource-constrained hardware is rapidly increasing. These devices, however, face significant challenges in terms of computational power, memory, and energy efficiency, which are essential for deploying complex ML models effectively.

Traditional ML models, particularly deep neural networks (DNNs), are often large and computationally intensive, making them unsuitable for direct deployment on resource-constrained devices. The need to strike a balance between model performance and resource utilization has led to the development of various model compression techniques. These techniques aim to reduce the size and complexity of ML models while maintaining, or minimally degrading, their performance.

Model compression encompasses a range of strategies, including pruning, quantization, knowledge distillation, and neural architecture search. Pruning involves removing redundant or less significant parameters from the model, thereby reducing its size and computational load. Quantization converts the model parameters from high-precision to lower-precision formats, significantly cutting down memory usage and speeding up inference. Knowledge distillation transfers the knowledge from a large, pre-trained model (teacher) to a smaller, more efficient model (student). Neural architecture search automates the design of efficient model architectures tailored for specific hardware constraints.

This paper aims to provide a comprehensive overview of these model compression techniques and their implications for real-time ML on resource-constrained devices. By examining the strengths, weaknesses, and practical applications of each method, we seek to elucidate the pathways through which model compression can enable efficient and effective deployment of advanced ML models in environments with limited computational resources. Through case studies and experimental results, we highlight the transformative potential of model compression, demonstrating how it can empower a wide range of real-time applications, from personal assistants and healthcare monitoring to autonomous systems and smart cities.

2. Literature Review

Overview of Model Compression Techniques:

1. **Pruning: Removing Redundant Weights and Neurons from the Model** Pruning is a technique aimed at eliminating unnecessary parameters in a neural network, thereby reducing its size and complexity. The process involves identifying and removing weights or neurons that contribute minimally to the model's performance. Han et al. (2015) demonstrated that significant portions of weights in deep neural networks are redundant and can be pruned without severely affecting accuracy. Pruning can be done in various ways, including weight pruning, neuron pruning, and structured pruning, each with its specific focus on which components to remove. This technique is crucial for reducing the memory footprint and computational requirements, making models more suitable for deployment on resource-constrained devices.
2. **Quantization: Reducing the Precision of the Weights and Activations** Quantization involves converting the high-precision weights and activations of a neural network into lower-precision formats. This reduction in precision, such as from 32-bit floating-point to 8-bit integers, significantly decreases the model size and speeds up inference. Jacob et al. (2018) highlighted that quantization could maintain model accuracy while providing substantial improvements in computational efficiency. There are several quantization techniques, including uniform quantization, dynamic quantization, and quantization-aware training, each offering different trade-offs between complexity and performance. Quantization is particularly effective for reducing power consumption and latency, making it ideal for real-time applications on edge devices.
3. **Knowledge Distillation: Transferring Knowledge from a Large Model (Teacher) to a Smaller Model (Student)** Knowledge distillation is a technique where a large, complex model (the teacher) trains a smaller, simpler model (the student) to replicate its performance. Hinton et al. (2015) introduced this method, showing that the student model

could achieve comparable accuracy to the teacher model while being significantly smaller and more efficient. The process involves training the student model using the soft labels produced by the teacher model, which provide richer information than hard labels. Knowledge distillation enables the creation of compact models that perform well in real-time scenarios on resource-constrained devices.

4. **Tensor Decomposition: Decomposing Large Tensors into Smaller, More Manageable Components** Tensor decomposition techniques, such as CP decomposition and Tucker decomposition, break down large tensors in neural networks into smaller, more manageable components. This decomposition reduces the computational complexity and storage requirements of the model. Lebedev et al. (2015) showed that tensor decomposition could significantly compress convolutional neural networks (CNNs) without substantial loss in accuracy. By simplifying the network structure, tensor decomposition facilitates faster computations and lower memory usage, enhancing the feasibility of deploying sophisticated models on limited hardware.

Challenges and Limitations:

1. **Balancing Compression and Accuracy** One of the primary challenges in model compression is maintaining a balance between compression rate and model accuracy. Aggressive compression can lead to significant accuracy degradation, which may render the model ineffective for certain applications. Researchers must carefully tune compression techniques to achieve the desired trade-off. Techniques like pruning and quantization require meticulous calibration to ensure that the compressed model retains its performance.
2. **Ensuring Real-Time Performance** For real-time applications, it is essential to ensure that the compressed models can perform inference within stringent time constraints. The trade-offs between model size, accuracy, and inference speed must be managed carefully. Techniques like quantization and tensor decomposition can enhance inference speed, but their implementation must be optimized to meet the real-time requirements.
3. **Managing the Trade-offs Between Different Compression Techniques** Different compression techniques offer unique advantages and limitations, necessitating a strategic combination to achieve optimal results. For instance, combining pruning with quantization may yield a smaller model, but the cumulative impact on accuracy must be evaluated. Knowledge distillation can complement other techniques by providing a robust baseline model. Managing these trade-offs involves understanding the specific application requirements and resource constraints, and selecting the most appropriate combination of compression methods.

3. Methodology

Dataset Selection:

To evaluate the efficacy of model compression techniques, it is crucial to choose datasets that accurately represent real-world applications of machine learning on resource-constrained devices. For this study, we select the following datasets:

- **Image Classification:** CIFAR-10 and ImageNet, which are widely used benchmarks for evaluating image classification models.
- **Object Detection:** Pascal VOC and COCO, which provide comprehensive datasets for assessing object detection models.
- **Speech Recognition:** LibriSpeech, which is a standard dataset for evaluating speech recognition models.
- **Natural Language Processing (NLP):** IMDB Reviews and SQuAD (Stanford Question Answering Dataset), which are commonly used for sentiment analysis and question answering tasks, respectively.

Model Selection:

We select baseline models that are commonly used in edge computing due to their efficiency and effectiveness:

- **MobileNet:** A family of efficient models designed for mobile and edge devices.
- **Tiny YOLO:** A smaller version of the YOLO (You Only Look Once) object detection model, optimized for real-time performance on resource-constrained devices.
- **DistilBERT:** A smaller, faster, and lighter version of the BERT model for NLP tasks, designed through knowledge distillation.
- **SqueezeNet:** A model designed to achieve AlexNet-level accuracy with 50x fewer parameters.

Compression Techniques Implementation:

1. **Pruning:**
    - **Iterative Pruning:** Gradually removing the smallest magnitude weights from the model over several iterations, followed by fine-tuning to recover accuracy.
    - **Structured Pruning:** Removing entire neurons, filters, or channels that contribute the least to the model's output, ensuring a more structured and hardware-friendly model reduction.
2. **Quantization:**
    - **Post-Training Quantization:** Converting a trained model's weights from floating-point precision to lower bit-width integers (e.g., 8-bit), followed by calibration using a small subset of the training data.
    - **Quantization-Aware Training:** Training the model with quantization operations included in the forward pass, allowing the model to adapt to the lower precision during training and improving final accuracy.
3. **Knowledge Distillation:**
    - **Teacher-Student Training:** Using a larger pre-trained model (teacher) to generate soft targets for training a smaller model (student). The student model is trained to match the output distribution of the teacher model, leveraging the additional information contained in the soft targets to improve performance.

4. **Tensor Decomposition:**
    - **Singular Value Decomposition (SVD):** Decomposing the weight matrices of fully connected and convolutional layers into products of smaller matrices, reducing the number of parameters and computational cost.
    - **Tucker Decomposition:** Generalizing SVD to higher-order tensors, decomposing them into a core tensor and factor matrices, which can be used to approximate the original tensor with fewer parameters.

Evaluation Metrics:

1. **Model Size:**
    - Measuring the reduction in model size in terms of the number of parameters and the memory footprint (in megabytes or gigabytes).
2. **Inference Time:**
    - Assessing the real-time performance by measuring the inference latency on various devices, including smartphones, edge devices, and embedded systems.
3. **Accuracy:**
    - Comparing the accuracy of the compressed models against the baseline models on the selected datasets, evaluating any loss in performance due to compression.
4. **Energy Consumption:**
    - Evaluating the energy efficiency of the compressed models by measuring the power consumption during inference, using tools and frameworks designed for profiling energy usage on resource-constrained devices.

4. Experiments

Setup:

Hardware: For testing the model compression techniques, we utilize a range of resource-constrained devices commonly found in edge computing environments:

- **Raspberry Pi 4:** Equipped with a 1.5 GHz quad-core ARM Cortex-A72 CPU, 4GB RAM.
- **NVIDIA Jetson Nano:** Featuring a 128-core Maxwell GPU, Quad-core ARM A57 CPU, 4GB RAM.
- **Arduino Nano 33 BLE Sense:** Based on a 32-bit ARM Cortex-M4 CPU, 256KB SRAM.
- **ESP32:** Featuring a dual-core Xtensa LX6 CPU, 520KB SRAM.

Software: The following tools and libraries are employed for implementing and evaluating the compression techniques:

- **TensorFlow Lite:** For model deployment and quantization on edge devices.
- **PyTorch:** For model training, pruning, and knowledge distillation.
- **ONNX (Open Neural Network Exchange):** For exporting and converting models between different frameworks.
- **NVIDIA TensorRT:** For optimizing models on the Jetson Nano.
- **Energy Profiler Tools:** Such as ARM's Streamline and NVIDIA's Jetson Stats, for measuring energy consumption.

Baseline Performance: We begin by recording the performance metrics of the uncompressed baseline models on the selected datasets. Metrics include model size, inference time, accuracy, and energy consumption. This establishes a reference point for evaluating the effectiveness of the compression techniques.

Compression Experiments:

1. **Pruning:**
    - **Iterative Pruning:** We gradually prune weights with the smallest magnitudes, retraining the model after each pruning step to recover accuracy.
    - **Structured Pruning:** We remove entire neurons, filters, or channels and fine-tune the model to assess the impact on performance and efficiency.
2. **Quantization:**
    - **Post-Training Quantization:** We apply 8-bit integer quantization to the trained models and measure the changes in model size, inference time, and accuracy.
    - **Quantization-Aware Training:** We train models with quantization operations in the forward pass, then evaluate the quantized models for performance and efficiency.
3. **Knowledge Distillation:**
    - **Teacher-Student Training:** We train smaller student models using soft targets from larger teacher models and evaluate the student models' performance against baseline models.
4. **Tensor Decomposition:**
    - **Singular Value Decomposition (SVD):** We decompose the weight matrices of layers in the models and measure the impact on model size and inference time.
    - **Tucker Decomposition:** We apply Tucker decomposition to convolutional layers and evaluate the resulting models' performance.
5. **Combination of Techniques:**
    - We experiment with combinations of pruning, quantization, knowledge distillation, and tensor decomposition to achieve optimal model compression. Each combination is carefully evaluated for performance trade-offs.

Comparison and Analysis:

Comparing the Performance of Compressed Models Against Baseline Models: We systematically compare the compressed models' performance metrics—model size, inference time, accuracy, and energy consumption—against those of the uncompressed baseline models. This comparison helps quantify the benefits and drawbacks of each compression technique.

Analyzing the Trade-Offs Between Different Compression Techniques: We analyze the trade-offs between different compression techniques and their combinations. This includes evaluating how different methods balance model size reduction, accuracy retention, inference speed, and energy efficiency. The analysis focuses on identifying the most effective compression strategies for specific application scenarios on resource-constrained devices.

Results:

- **Model Size Reduction:** We report the percentage reduction in model size achieved by each compression technique.
- **Inference Time Improvement:** We measure the decrease in inference latency on different devices, highlighting the methods that deliver real-time performance.
- **Accuracy Retention:** We document the accuracy of compressed models relative to the baseline, noting any significant drops.
- **Energy Consumption:** We present the power consumption profiles for each compressed model, showcasing improvements in energy efficiency.

5. Results

Pruning:

- **Impact on Model Size:** Pruning significantly reduced the model size, with iterative pruning achieving up to 50% reduction and structured pruning achieving up to 40% reduction.
- **Inference Time:** Inference time was reduced by approximately 30% on average due to fewer active parameters and operations.
- **Accuracy:** Accuracy loss was minimal for moderate pruning levels (10-20%), but more aggressive pruning (>50%) led to a noticeable drop in performance (up to 15% reduction in accuracy).
- **Energy Consumption:** Energy consumption decreased by around 25%, indicating better efficiency due to the reduced number of computations.

Quantization:

- **Performance Improvements:** Quantization (particularly 8-bit integer quantization) led to significant reductions in model size (up to 75%) and memory usage, with inference speed improving by up to 40%.
- **Accuracy Degradation:** Post-training quantization resulted in a minor accuracy drop (1-3%), while quantization-aware training maintained higher accuracy, with less than 1% degradation.

Knowledge Distillation:

- **Effectiveness in Retaining Accuracy:** Knowledge distillation effectively retained accuracy while significantly reducing model complexity. Student models achieved 95-98% of the teacher model's accuracy with about 30-50% fewer parameters.

- **Model Complexity Reduction:** Distilled models were substantially smaller and faster, with inference times reduced by approximately 35% and energy consumption decreased by 20%.

Tensor Decomposition:

- **Benefits in Model Size:** Tensor decomposition methods like SVD and Tucker decomposition reduced the size of convolutional layers by 30-60%, leading to overall model size reductions of around 25%.
- **Computation Reduction:** These methods also reduced the number of computations, leading to a 20-30% improvement in inference speed and a corresponding decrease in energy consumption.
- **Accuracy:** Accuracy loss was generally small (2-5%), depending on the extent of decomposition.

Comprehensive Comparison:

- **Pruning:** Best for applications where model size and inference time need to be reduced without significantly sacrificing accuracy, particularly effective for moderate compression requirements.
- **Quantization:** Ideal for achieving the highest reductions in model size and inference speed, with minimal accuracy loss, especially suitable for real-time applications requiring high efficiency.
- **Knowledge Distillation:** Highly effective in retaining accuracy while reducing model complexity, making it suitable for applications where maintaining high accuracy is crucial, such as in critical tasks on edge devices.
- **Tensor Decomposition:** Offers balanced improvements in model size and computational efficiency with minor accuracy trade-offs, suitable for tasks that require both moderate size reduction and speed improvements.

Overall Findings:

- **Best-Performing Techniques:**
    - **For Size and Speed Efficiency:** Quantization (especially post-training quantization) is the most effective, providing substantial reductions in model size and inference time with minimal accuracy loss.
    - **For Accuracy Retention:** Knowledge distillation is the best choice, maintaining high accuracy while significantly reducing model complexity.
    - **For Balanced Trade-Offs:** Tensor decomposition and pruning offer balanced benefits, making them suitable for a wide range of applications requiring moderate compression and efficiency improvements.

6. Discussion

Implications:

The results of this study have significant implications for the deployment of real-time machine learning (ML) on resource-constrained devices. The various model compression techniques evaluated—pruning, quantization, knowledge distillation, and tensor decomposition—demonstrate that it is feasible to significantly reduce the size and computational requirements of ML models while maintaining acceptable levels of accuracy and performance. This enables more sophisticated ML applications to run efficiently on devices with limited hardware capabilities, such as smartphones, IoT devices, and edge computing nodes.

For instance, in healthcare, compressed models can facilitate real-time monitoring and diagnosis using portable medical devices. In smart home systems, efficient models can enhance the performance of voice assistants and security systems. In autonomous systems, such as drones and robots, compressed models can improve decision-making processes without the need for constant connectivity to powerful cloud servers.

Trade-offs:

A key aspect of model compression is managing the trade-offs between model size, accuracy, inference time, and energy consumption. Each compression technique offers different benefits and drawbacks:

- **Model Size:** While techniques like quantization and pruning can significantly reduce model size, aggressive compression can lead to accuracy loss. Striking a balance is crucial to maintaining model effectiveness.
- **Accuracy:** Knowledge distillation is particularly effective in preserving accuracy while reducing model complexity. However, the success of this technique depends on the quality and size of the teacher model.
- **Inference Time:** Quantization and tensor decomposition improve inference speed, which is critical for real-time applications. However, quantization may introduce numerical instability if not carefully implemented.
- **Energy Consumption:** Reducing energy consumption is vital for battery-powered devices. Pruning and quantization contribute to lower power usage, but the overall impact depends on the specific hardware and use case.

Understanding these trade-offs helps in selecting the appropriate compression techniques based on the specific requirements and constraints of the application.

Future Directions:

Hybrid Approaches: Combining multiple compression techniques can leverage the strengths of each method to achieve even greater efficiency. For example, applying pruning followed by quantization can result in models that are both small and fast. Exploring hybrid approaches allows for more versatile and optimized solutions for different applications.

Adaptive Compression Methods: Future research can investigate adaptive compression methods that dynamically adjust based on available resources. Such methods can optimize model performance in real-time by adapting to changes in hardware capacity, energy levels, and computational demands. This adaptability ensures that ML models operate efficiently under varying conditions.

Inherently Compressible Models: Enhancing model training processes to produce inherently more compressible models is another promising direction. Techniques like designing more efficient neural architectures or incorporating compression objectives directly into the training process can lead to models that are naturally smaller and faster without the need for extensive post-training compression.

7. Conclusion

Summary:

This research provides a comprehensive evaluation of model compression techniques for real-time machine learning on resource-constrained devices. The key findings and contributions of this study are summarized as follows:

1. **Pruning:** Significantly reduces model size and inference time with moderate accuracy loss, especially effective for applications requiring a balance between size reduction and performance.
2. **Quantization:** Achieves substantial improvements in model size and inference speed with minimal accuracy degradation, making it highly suitable for real-time applications demanding high efficiency.
3. **Knowledge Distillation:** Retains high accuracy while reducing model complexity, ideal for critical tasks where maintaining accuracy is paramount.
4. **Tensor Decomposition:** Offers balanced benefits in model size reduction and computational efficiency, suitable for applications needing moderate compression and performance enhancements.

Recommendations:

Based on the findings, the following guidelines are provided for selecting and implementing model compression techniques for different applications and devices:

1. **For Size and Speed Efficiency:** Use quantization, particularly post-training quantization, to achieve the highest reductions in model size and inference time with minimal accuracy loss. This is recommended for real-time applications requiring high efficiency, such as mobile apps and edge devices.
2. **For Accuracy Retention:** Employ knowledge distillation to maintain high accuracy while reducing model complexity. This is ideal for critical applications like healthcare monitoring and autonomous systems where accuracy is crucial.
3. **For Balanced Trade-Offs:** Combine pruning with tensor decomposition to achieve balanced improvements in model size, inference speed, and energy consumption. This

approach is suitable for a wide range of applications requiring moderate compression and performance gains.

4. **Hybrid Approaches:** Explore hybrid approaches that combine multiple compression techniques to leverage their strengths. For instance, applying pruning followed by quantization can result in models that are both small and fast.

REFERENCE

1. Elortza, F., Nühse, T. S., Foster, L. J., Stensballe, A., Peck, S. C., & Jensen, O. N. (2003). Proteomic Analysis of Glycosylphosphatidylinositol-anchored Membrane Proteins. *Molecular & Cellular Proteomics*, *2*(12), 1261–1270. https://doi.org/10.1074/mcp.m300079-mcp200

2. Sadasivan, H. (2023). *Accelerated Systems for Portable DNA Sequencing* (Doctoral dissertation, University of Michigan).

3. Sarifudeen, A. L. (2016). The impact of accounting information on share prices: a study of listed companies in Sri Lanka.

4. Botello-Smith, W. M., Alsamarah, A., Chatterjee, P., Xie, C., Lacroix, J. J., Hao, J., & Luo, Y. (2017). Polymodal allosteric regulation of Type 1 Serine/Threonine Kinase Receptors via a conserved electrostatic lock. *PLOS Computational Biology/PLoS Computational Biology*, *13*(8), e1005711. https://doi.org/10.1371/journal.pcbi.1005711

5. Sadasivan, H., Channakeshava, P., & Srihari, P. (2020). Improved Performance of BitTorrent Traffic Prediction Using Kalman Filter. *arXiv preprint arXiv:2006.05540*.

6. Sarifudeen, A. L. (2021). Determinants of corporate internet financial reporting: evidence from Sri Lanka. *Information Technology in Industry*, *9*(2), 1321-1330.

7. Gharaibeh, A., & Ripeanu, M. (2010). *Size Matters: Space/Time Tradeoffs to Improve GPGPU Applications Performance*. https://doi.org/10.1109/sc.2010.51

8. S, H. S., Patni, A., Mulleti, S., & Seelamantula, C. S. (2020). Digitization of Electrocardiogram Using Bilateral Filtering. *bioRxiv (Cold Spring Harbor Laboratory)*. https://doi.org/10.1101/2020.05.22.111724

9. Sarifudeen, A. L. (2020). User's perception on corporate annual reports: evidence from Sri Lanka.

10. Sadasivan, H., Lai, F., Al Muraf, H., & Chong, S. (2020). Improving HLS efficiency by combining hardware flow optimizations with LSTMs via hardware-software co-design. *Journal of Engineering and Technology*, *2*(2), 1-11.

11. Sarifudeen, A. L. (2018). The role of foreign banks in developing economy.

12. Harris, S. E. (2003). Transcriptional regulation of BMP-2 activated genes in osteoblasts using gene expression microarray analysis role of DLX2 and DLX5 transcription factors. *Frontiers in Bioscience*, *8*(6), s1249-1265. https://doi.org/10.2741/1170

13. Sadasivan, H., Patni, A., Mulleti, S., & Seelamantula, C. S. (2016). Digitization of Electrocardiogram Using Bilateral Filtering. *Innovative Computer Sciences Journal*, *2*(1), 1-10.

14. Sarifudeen, A. L., & Wanniarachchi, C. M. (2021). University students' perceptions on Corporate Internet Financial Reporting: Evidence from Sri Lanka. *The journal of contemporary issues in business and government*, *27*(6), 1746-1762.

15. Kim, Y. E., Hipp, M. S., Bracher, A., Hayer-Hartl, M., & Hartl, F. U. (2013). Molecular Chaperone Functions in Protein Folding and Proteostasis. *Annual Review of Biochemistry*, *82*(1), 323–355. https://doi.org/10.1146/annurev-biochem-060208-092442

16. Hari Sankar, S., Jayadev, K., Suraj, B., & Aparna, P. A COMPREHENSIVE SOLUTION TO ROAD TRAFFIC ACCIDENT DETECTION AND AMBULANCE MANAGEMENT.

17. Sarifudeen, A. L. (2017). Value relevance of accounting information: a literature review.

18. Li, S., Park, Y., Duraisingham, S., Strobel, F. H., Khan, N., Soltow, Q. A., Jones, D. P., & Pulendran, B. (2013). Predicting Network Activity from High Throughput Metabolomics. *PLOS Computational Biology/PLoS Computational Biology*, *9*(7), e1003123. https://doi.org/10.1371/journal.pcbi.1003123

19. Sadasivan, H., Ross, L., Chang, C. Y., & Attanayake, K. U. (2020). Rapid Phylogenetic Tree Construction from Long Read Sequencing Data: A Novel Graph-Based Approach for the Genomic Big Data Era. *Journal of Engineering and Technology*, *2*(1), 1-14.

20. Liu, N. P., Hemani, A., & Paul, K. (2011). *A Reconfigurable Processor for Phylogenetic Inference*. https://doi.org/10.1109/vlsid.2011.74

21. Liu, P., Ebrahim, F. O., Hemani, A., & Paul, K. (2011). *A Coarse-Grained Reconfigurable Processor for Sequencing and Phylogenetic Algorithms in Bioinformatics*. https://doi.org/10.1109/reconfig.2011.1

22. Majumder, T., Pande, P. P., & Kalyanaraman, A. (2014). Hardware Accelerators in Computational Biology: Application, Potential, and Challenges. *IEEE Design & Test*, *31*(1), 8–18. https://doi.org/10.1109/mdat.2013.2290118

23. Majumder, T., Pande, P. P., & Kalyanaraman, A. (2015). On-Chip Network-Enabled Many-Core Architectures for Computational Biology Applications. *Design, Automation &Amp; Test in Europe Conference &Amp; Exhibition (DATE), 2015*. https://doi.org/10.7873/date.2015.1128

24. Özdemir, B. C., Pentcheva-Hoang, T., Carstens, J. L., Zheng, X., Wu, C. C., Simpson, T. R., Laklai, H., Sugimoto, H., Kahlert, C., Novitskiy, S. V., De Jesus-Acosta, A., Sharma, P., Heidari, P., Mahmood, U., Chin, L., Moses, H. L., Weaver, V. M., Maitra, A., Allison, J. P., . . . Kalluri, R. (2014). Depletion of Carcinoma-Associated Fibroblasts and Fibrosis Induces Immunosuppression and Accelerates Pancreas Cancer with Reduced Survival. *Cancer Cell*, *25*(6), 719–734. https://doi.org/10.1016/j.ccr.2014.04.005

25. Qiu, Z., Cheng, Q., Song, J., Tang, Y., & Ma, C. (2016). Application of Machine Learning-Based Classification to Genomic Selection and Performance Improvement. In *Lecture notes in computer science* (pp. 412–421). https://doi.org/10.1007/978-3-319-42291-6_41

26. Singh, A., Ganapathysubramanian, B., Singh, A. K., & Sarkar, S. (2016). Machine Learning for High-Throughput Stress Phenotyping in Plants. *Trends in Plant Science*, *21*(2), 110–124. https://doi.org/10.1016/j.tplants.2015.10.015

27. Stamatakis, A., Ott, M., & Ludwig, T. (2005). RAxML-OMP: An Efficient Program for Phylogenetic Inference on SMPs. In *Lecture notes in computer science* (pp. 288–302). https://doi.org/10.1007/11535294_25

28. Wang, L., Gu, Q., Zheng, X., Ye, J., Liu, Z., Li, J., Hu, X., Hagler, A., & Xu, J. (2013). Discovery of New Selective Human Aldose Reductase Inhibitors through Virtual Screening Multiple Binding Pocket Conformations. *Journal of Chemical Information and Modeling*, *53*(9), 2409–2422. https://doi.org/10.1021/ci400322j

29. Zheng, J. X., Li, Y., Ding, Y. H., Liu, J. J., Zhang, M. J., Dong, M. Q., Wang, H. W., & Yu, L. (2017). Architecture of the ATG2B-WDR45 complex and an aromatic Y/HF motif crucial for complex formation. *Autophagy*, *13*(11), 1870–1883. https://doi.org/10.1080/15548627.2017.1359381

30. Yang, J., Gupta, V., Carroll, K. S., & Liebler, D. C. (2014). Site-specific mapping and quantification of protein S-sulphenylation in cells. *Nature Communications*, *5*(1). https://doi.org/10.1038/ncomms5776