Energy-Efficient Machine Learning Models for Mobile Devices

Introduction

Machine learning for mobile apps will improve user experiences through functionalities such as real-time translation of languages, augmented reality, and personalized recommendations. However, these features increase the computational load on resource-constrained, battery-dependent mobile devices. To meet this demand, sophisticated and energy-efficient model architectures are essential. Edge AI and on-device processing enable real-time operations, enhance privacy, and reduce cloud dependence. Developing power-efficient ML models is crucial for user satisfaction and the sustainability of mobile devices. My paper examines the energy efficiency of ML models, industry trends, requirements, and existing solutions, and proposes improvements for mobile applications.

Industry Trends and Needs

Proliferation of Mobile: The rapid spread of AI technologies in mobile applications requires efficient, high-accuracy on-device ML models across various domains (Chen, 2020).

Edge AI and On-Device Processing: Real-time processing and privacy concerns drive the need for efficient on-device ML tasks in applications like AR and voice assistants (Xu et al., 2018).

Sustainability and Battery Life: Energy-efficient ML models are crucial for longer-lasting, sustainable mobile devices (Gao & Tham, 2019).

High-Priority Areas and Problems in the Software Industry

1. **Health and Fitness Applications:** Health and fitness apps like Fitbit and Apple Health require the maintenance of sensor data tracking and its analysis, considered to be an energy-expensive task, and that may cause drastic battery drilling (Shen et al., 2021).
2. **Photography and Video Processing:** Smartphones leverage machine learning to improve images and recognize the scene in real-time (eg. Google Pixel). High-quality processing exerts a heavy toll on computational resources and tends to turn the device into a power guzzler in no time at all (Liu et al., 2019).
3. **Voice Assistants and Speech Recognition:** Virtual assistants, such as Siri and Google Assistant, use ML for voice recognition and natural language processing. Accurate speech recognition and NLP are massive computational tasks, so this imposes quite a toll on battery life (Kim et al., 2020).
4. **Gaming:** Mobile games that use AI for adaptiveness in gameplay together with real-time enhancements need efficient processing to ensure performance and battery life (e.g., Pokémon Go) (Li et al., 2020).
5. **Augmented Reality (AR) and Virtual Reality (VR):** In AR and VR applications, the power of ML supports tasks like object recognition and helps in supporting interactive features (eg. Google Lens). In most cases, their application is data-intensive and with real-time processing of complicated data; thus, the applications are hungry for great speed (Wang et al., 2019).
6. **Real-Time Language Translation:** Applications such as Google Translate can in real-time do the translation of a language based on ML. Such applications, for continuous listening, processing, and translation, are resource-intensive and require the creation of energy-efficient models (Huang et al., 2018).

7. **Customer Service Chatbots:** Many mobile application integrations bring in customer service chatbots with very strong NLP models, which are used for understanding and responding to users quickly. However, these models need to be pretty battery and performance-efficient (Zhang et al., 2020).

Current Solutions

Model Optimization Techniques

- Quantization: The process of reducing the precision of model weights and activations to save memory consumption and computation (Jacob et al., 2018).
- Pruning: Removing less important weights and neurons for a more compact and quicker network (Han et al., 2015).
- Knowledge Distillation: It is an instance of training a smaller model to mimic the response of a bigger model to perform equally well but with fewer resources, flagged (Hinton et al., 2015)
- Efficient Architectures: Use light model architectures such as MobileNet and EfficientNet targeting mobile devices (Sandler et al., 2018).

On-Device ML Frameworks

- TensorFlow Lite: Optimized for mobile and embedded devices, executing models efficiently (Google, 2021).
- Core ML: A framework in Apple that enables the integration of ML models into iOS applications, most of which are designed to be high-performance and power-efficient (Apple, 2021).
- ONNX Runtime Mobile: Efficient inference engine empowered in running ONNX models on mobile devices (ONNX, 2021).

Hardware Solutions

- Mobile AI accelerators: These are specialized hardware accelerators, such as the Apple Neural Engine and the Qualcomm Hexagon DSP, used to accelerate ML tasks with minimal energy amounts (Apple, 2021; Qualcomm, 2021).
- Edge TPUs: Developed by Google to facilitate high-speed and low-power machine learning inference on devices operating on the edge (Google, 2021).

Critical Analysis: Pros and Cons of Current Solutions

Model Optimization Techniques

1. Quantization
    - Pros: Reduces model size, faster inference, and lower power consumption (Jacob et al., 2018).
    - Cons: Potential accuracy loss, complex implementation requiring extensive tuning.
2. Pruning
    - Pros: Simplifies models, reduces resource usage, and improves speed (Han et al., 2015).
    - Cons: Risk of performance degradation, computationally expensive to identify and validate pruned parts.
3. Knowledge Distillation
    - Pros: Efficient, flexible, smaller models maintain performance (Hinton et al., 2015).
    - Cons: Complex training process, potential fidelity loss in the student model.
4. Efficient Architectures
    - Pros: Designed for efficiency, high performance, and lightweight (Sandler et al., 2018).

- Cons: Requires retraining existing models, limited generalization across tasks.

On-Device ML Frameworks

1. TensorFlow Lite
    - Pros: Optimized for mobile, and supports a wide range of optimization techniques (Google, 2021).
    - Cons: Limited features compared to full TensorFlow, complicated model deployment.
2. Core ML
    - Pros: Seamless integration into iOS apps, high efficiency with hardware acceleration (Apple, 2021).
    - Cons: Restricted to Apple's ecosystem, feature gaps compared to other frameworks.
3. ONNX Runtime Mobile
    - Pros: Cross-platform support, and optimized performance for mobile devices (ONNX, 2021).
    - Cons: Compatibility issues with different model formats, steep learning curve for developers.

Specialized Hardware Solutions

1. Mobile AI Accelerators
    - Pros: Significant performance boost with low energy consumption, optimized for ML tasks (Apple, 2021; Qualcomm, 2021).
    - Cons: Increased overall device cost, and hardware fragmentation leading to compatibility issues.
2. Edge TPUs
    - Pros: Low power consumption, and high throughput for real-time applications (Google, 2021).
    - Cons: Limited availability compared to other accelerators, integration complexity with existing systems.

Proposed Improvement: Dynamic Model Adaptation for Energy-Efficient Machine Learning on Mobile Devices

Dynamic Model Adaptation aims at achieving maximum energy savings for minimal performance degradation by allowing the ML model to adapt its complexity at runtime based on available and power-constrained resources. The adaptive techniques include those of adaptive precision, layer-wise activation, and reinforcement learning-based adaptation.

Key Components

1. Adaptive Precision:
    - Implementation: Models can switch between all available precisions—e.g., 8-, 16-, and 32-bits—depending on the actual computational load and battery state. This could serve as the means to save energy in less important tasks while still being that much more accurate when it comes to more important ones.
    - Benefits: Reduces overall power consumption with no significant change in performance (Jacob et al., 2018).
2. Layer-Wise Activation
    - Implementation: At any given point in time, given the requirements, models can turn on and off certain layers in a conditionally dependent manner. For example, in a less critical

operation, just the simple layers will be kept on and the more complex ones will be turned off to save power.
    - Advantages: Reduces computational burden and saves energy by using only the required parts of the model (Han et al., 2015).
3. Reinforcement Learning-Based Adaptation
    - Implementation: An RL agent would monitor the status of, for example, the battery level and CPU load, then set the complexity of the model to make a prediction. The RL agent learns to do this optimization through real-time feedback that learns the trade-off.
    - Advantages: Adapts to dynamically changing conditions in real-time to ensure optimal performance and energy use (He et al., 2019).

Advantages of Dynamic Model Adaptation

- Enhanced Energy Efficiency: The model complexity can be scaled dynamically, thereby saving significantly on the use of power for an increased life of the battery.
- Maintained Performance: Ensuring performance is not compromised, with the complexity of scaling increased or decreased according to task requirements.
- Improved Flexibility: This allows for even greater flexibility when it comes to resource management, making the system more suitable for a larger number of applications and for arbitrary conditions of the device.

Implementation Challenges

- Complexity: Adds complexity to model design and training processes.
- Overhead: Dynamic adaptation introduces computational overhead through mechanisms that are supposed to be kept small.
- Generalization: This could be effectively carried out only through careful tuning and extensive testing.

Conclusion

The development of energy-efficient machine learning models for mobile devices is critical to meeting the growing demand for intelligent features without compromising battery life. By exploring and optimizing current solutions and implementing proposed improvements, developers can create more efficient and sustainable mobile applications.

References

Apple. (2021). Core ML. Retrieved from https://developer.apple.com/documentation/coreml

Chen, X. (2020). Edge AI: Machine learning on mobile and edge devices. Journal of AI Research, 45(3), 201-215.

Gao, L., & Tham, C. K. (2019). Sustainable mobile computing: The case for energy-efficient machine learning. IEEE Transactions on Sustainable Computing, 4(2), 193-204.

Google. (2021). TensorFlow Lite. Retrieved from https://www.tensorflow.org/lite

Google. (2021). Edge TPU. Retrieved from https://cloud.google.com/edge-tpu

Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. Advances in Neural Information Processing Systems, 28, 1135-1143.

He, Y., Zhang, X., & Sun, J. (2019). Channel pruning for accelerating very deep neural networks. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 138-145.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... & Adam, H. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2704-2713.

Kim, H., Park, Y., Lee, S., & Choi, S. (2020). On-device machine learning: An empirical study of binary neural networks on mobile devices. IEEE Access, 8, 130970-130981.

Liu, Z., Luo, P., Wang, X., & Tang, X. (2019). Real-time image enhancement for mobile devices. IEEE Transactions on Image Processing, 28(9), 4445-4457.

Li, X., Wu, X., & Yu, K. (2020). Efficient machine learning for mobile gaming. ACM Transactions on Multimedia Computing, Communications, and Applications, 16(3), 1-23.

ONNX. (2021). ONNX Runtime Mobile. Retrieved from https://onnxruntime.ai/docs/

Qualcomm. (2021). Hexagon DSP. Retrieved from https://www.qualcomm.com/products/application/processors/hexagon-dsp

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4510-4520.

Shen, Y., He, Y., & Peng, Y. (2021). Energy-efficient machine learning for continuous health monitoring on mobile devices. IEEE Transactions on Mobile Computing, 20(1), 107-119.

Wang, X., Chen, Y., & Liu, H. (2019). Real-time object detection and tracking on mobile devices for augmented reality applications. IEEE Transactions on Mobile Computing, 18(10), 2390-2401.

Xu, Y., Wu, H., & Wang, Y. (2018). On-device artificial intelligence: A survey. Journal of AI Research, 50(3), 275-300.

Zhang, Y., Jin, R., & Zhou, Z. (2020). Mobile customer service chatbots: Design principles and implementation. ACM Transactions on Human-Computer Interaction, 27(1), 1-23.