IT-HÖGSKOLAN

Här startar din IT-karriär.

Comparing Compression Techniques for Deep Learning Models: *Pruning, Clustering and Quantization*

- Elias El Haddad

- Utvecklare inom AI & maskininlärning, 400 YH-poäng
- Examensarbete, 15 YH-poäng
- May 2024

Table of Contents

1. Abstract

This thesis compares various model optimization techniques, specifically pruning, clustering, and quantization. The objective is to evaluate the impact of the techniques on model size and performance. We trained a base LSTM model, what is called a “teacher model” on an emotion classification dataset and then applied pruning and clustering to create two additional optimized models. These models were subsequently quantized to further reduce their sizes.

This work highlights the trade-offs between the model compression techniques and their impact on model performance, providing valuable insights for deploying efficient models resource-constrained environments.

2. Introduction

This thesis is a part of my education in AI & ML. The project selected for this thesis is a continuation of the deep learning course and focuses on optimizing machine learning models. The primary aim is to explore various model compression techniques to make them more efficient for deployment in resource-constrained environments.

The project is significant as it addresses practical challenges associated with deploying large machine learning models in real-world scenarios, such as mobile devices, smart watches, etc., where computational resources are limited. By optimizing these models, we aim to maintain high accuracy while significantly reducing their size, thus making them more feasible for various applications.

In the following sections, the methodology, results, and analysis of the project are detailed, providing a comprehensive overview of the techniques used and their effectiveness. This introduction sets the stage for understanding the importance and scope of the work done in this thesis.

3. Purpose and Goals

Purpose

This thesis explores and implements model optimization techniques for machine learning models, specifically focusing on compression methods such as pruning, clustering, and quantization. The goal is to reduce the size of the models while maintaining or improving their accuracy. This optimization is crucial for deploying models in environments with limited computational resources, such as mobile devices. By leveraging these techniques, the project aims to make machine learning models more efficient and feasible for real-world applications, thereby enhancing their usability and performance in practical scenarios.

Goals

1. **Implement and compare model optimization techniques:**
   - Apply pruning, clustering, and quantization to a baseline LSTM model.
   - Evaluate the effectiveness of each technique in reducing model size and maintaining accuracy.
2. **Maintain high model performance:**
   - Ensure that the optimized models achieve high training and validation accuracy.
   - Compare the performance of the optimized models against the baseline model to assess the impact of each optimization technique.
3. **Develop a comprehensive analysis:**
   - Provide a detailed analysis of the results, including visualizations to compare model sizes and accuracies.
   - Discuss the trade-offs between model size and performance for each optimization technique.
4. **Practical applications:**
   - Highlight the benefits of using optimized models in real-world scenarios.

These goals aim to provide a thorough understanding of model optimization techniques and their practical applications, contributing valuable insights to machine learning.

4. Method

In this section, we will detail the methodologies, tools, and decisions made throughout the project, explaining the technical stack, the logic behind the choices, and the workflow followed.

Programming language and libraries

Programming language:

- Python: Python was chosen as the main programming language for this project due to its extensive support for machine learning and data science libraries, ease of use, and readability. Its large community and wealth of resources make it an ideal choice. The availability of numerous pre-built libraries and frameworks significantly accelerates development and experimentation.

Libraries:

- TensorFlow and Keras: These were used for building, training, and optimizing the neural network models. TensorFlow provides robust support for model optimization techniques such as pruning and quantization, while Keras offers a high-level API for easy model creation. TensorFlow's integration with Keras simplifies the process of model development, making it possible to prototype and deploy models efficiently.
- TensorFlow Model Optimization Toolkit (TFMOT): TFMOT was utilized for implementing pruning and clustering techniques. It integrates seamlessly with TensorFlow, providing a straightforward way to apply these optimizations. This toolkit allows for significant reductions in model size and improvements in inference speed without extensive modifications to the original model code.
- Scikit-learn: Used for preprocessing tasks such as data splitting and encoding labels. Scikit-Learn's comprehensive suite of tools for data preparation and evaluation complements TensorFlow's capabilities, ensuring that the data fed into the models is clean and well-structured.
- Pandas: Utilized for data manipulation and preprocessing. Pandas is indispensable for handling large datasets and performing complex operations with ease, enabling efficient data cleaning and transformation.
- Matplotlib and Seaborn: Employed for visualizing the results and comparing model performance. These libraries provide a wide array of plotting functions that help in understanding the data and the model

outcomes better. Visualization is crucial for communicating the effectiveness of different optimization techniques.

Data and preprocessing

- Dataset: The dataset used in this project is the same as the one used in the previous deep learning course project, which is "emotions.csv". It contains text data with corresponding labels. Reusing this dataset ensures consistency in evaluating the improvements brought by the optimization techniques applied.

Preprocessing techniques

- Tokenization and Padding: The text data was tokenized and padded to ensure uniform input size for the neural network. Tokenization involves converting text into a sequence of integers, where each integer represents a unique word. Padding adjusts the sequences to a common length, which is necessary for batch processing in neural networks.
- Label Encoding: Labels were encoded using one-hot encoding to facilitate multi-class classification. One-hot encoding transforms categorical labels into a binary matrix representation, which is suitable for the output layer of a neural network.

Model Architecture

The model architecture used in this project is an LSTM (Long Short-Term Memory) network, chosen for its effectiveness in handling sequential data. LSTM networks are particularly well-suited for tasks involving time-series data or text, where maintaining context over long sequences is essential.

Layers:

- Embedding Layer: Converts input text data into dense vectors of fixed size. This layer learns a representation of the input text that captures semantic similarities between words.
- LSTM Layer: Processes the sequential data, capturing dependencies and patterns in the input sequences. LSTMs can learn long-term dependencies, making them ideal for tasks like sentiment analysis and language modeling.

- Dropout Layer: Prevents overfitting by randomly dropping a fraction of the input units during training. Dropout is a regularization technique that improves the model's generalization ability.
- Dense Layer: Produces the final output with a softmax activation function for multi-class classification. The softmax function converts the logits into probabilities, allowing for the prediction of the most likely class.

Model Optimization Techniques

Pruning:

- Pruning: Pruning reduces the number of parameters in the neural network by removing less important weights, thus reducing model size and potentially improving inference speed. Pruning can be applied in a structured or unstructured manner, with structured pruning removing entire neurons or channels, and unstructured pruning removing individual weights.
- Implementation: TensorFlow's prune_low_magnitude function from TFMOT was used to apply pruning. This function identifies and removes weights with magnitudes below a certain threshold, effectively simplifying the network.

Clustering:

- Clustering: Clustering reduces the number of unique weight values in the model, which can lead to better compression. By grouping weights into clusters and sharing the same value among clustered weights, the model becomes more efficient.
- Implementation: TensorFlow's cluster_weights function from TFMOT was utilized for clustering. This technique is particularly useful when deploying models on hardware with limited memory, as it reduces the storage requirements.

Quantization

- Quantization: Quantization converts the model weights from floating-point to lower precision, reducing the model size and improving performance on hardware with limited resources. Quantization can be applied during

training (quantization-aware training) or after training (post-training quantization).
- Implementation: TensorFlow Lite's TFLiteConverter was used to quantize the models. TensorFlow Lite is designed for deploying models on mobile and embedded devices, providing tools to optimize models for these environments.

Workflow and project management

Workflow:

- The project was structured in phases, starting with data preprocessing, followed by model building, training, optimization, and finally, evaluation. Each phase had specific goals and deliverables, ensuring a systematic approach to the project.
- Regular reviews and iterations were conducted to ensure the quality and accuracy of the models. This iterative process allowed for continuous improvements and adjustments based on the results obtained.

Project management:

- Individual Work: This project was conducted independently, allowing for flexible scheduling and adaptation of methods to fit the project's needs.
- Agile Methodology: The project followed an Agile approach with iterative development and frequent evaluations. Agile principles emphasize flexibility, iterative progress, and continuous feedback, making them well-suited for research and development projects.
- Sprint Planning: Weekly sprints were planned, with specific goals and deliverables for each sprint. Sprints allowed for focused work on various aspects of the project, ensuring steady progress and the ability to adapt to new insights or challenges.
- Progress Tracking: Regular progress tracking was maintained to ensure alignment with project goals and timelines. Tools such as personal task lists were used to manage tasks and track the project's status.

Decision Rationale

- Why Python and TensorFlow/Keras?

Python's extensive ecosystem and ease of use make it ideal for rapid prototyping and experimentation. The language's simplicity and readability facilitate collaboration among team members with varying levels of programming expertise.

TensorFlow and Keras offer powerful tools for model building and optimization, with extensive documentation and community support. Their flexibility and performance make them suitable for both research and production environments.

- Why Pruning, Clustering, and Quantization?

  These techniques were chosen for their proven effectiveness in reducing model size and maintaining accuracy, making them suitable for deployment in resource-constrained environments. The ability to deploy smaller, efficient models on edge devices opens new possibilities for real-time applications.

These methodologies and tools were carefully chosen to ensure the project's success, providing a robust framework for developing and optimizing the machine learning models. The combination of optimization techniques and a systematic project management approach ensured that the thesis's goals were met effectively.

5. Results and Analysis

Results

The primary goal of this thesis was to compare the effectiveness of different model compression techniques, specifically pruning and clustering, on a recurrent neural network model trained on text data. The project continued from a previous course, utilizing the same dataset, preprocessing techniques, and model architecture.

The dataset used for this project consisted of labeled text data, which was preprocessed and tokenized to be fed into the model. The base model, an LSTM (Long Short-Term Memory) network, was trained on this data, and then subjected to two different compression techniques: pruning and clustering.

To compare the models effectively, the following metrics were considered:

- Model size (KB)
- Training accuracy
- Validation accuracy
- Test accuracy after quantization

Base Model (Teacher Model)

- Size: 4075.13 KB
- Training Accuracy: 99.34%
- Validation Accuracy: 90.60%

Pruned Model

- Size: 1365.81 KB
- Training Accuracy: 99.32%
- Validation Accuracy: 90.17%

Clustered Model

- Size: 1365.78 KB
- Training Accuracy: 98.29%
- Validation Accuracy: 90.03%

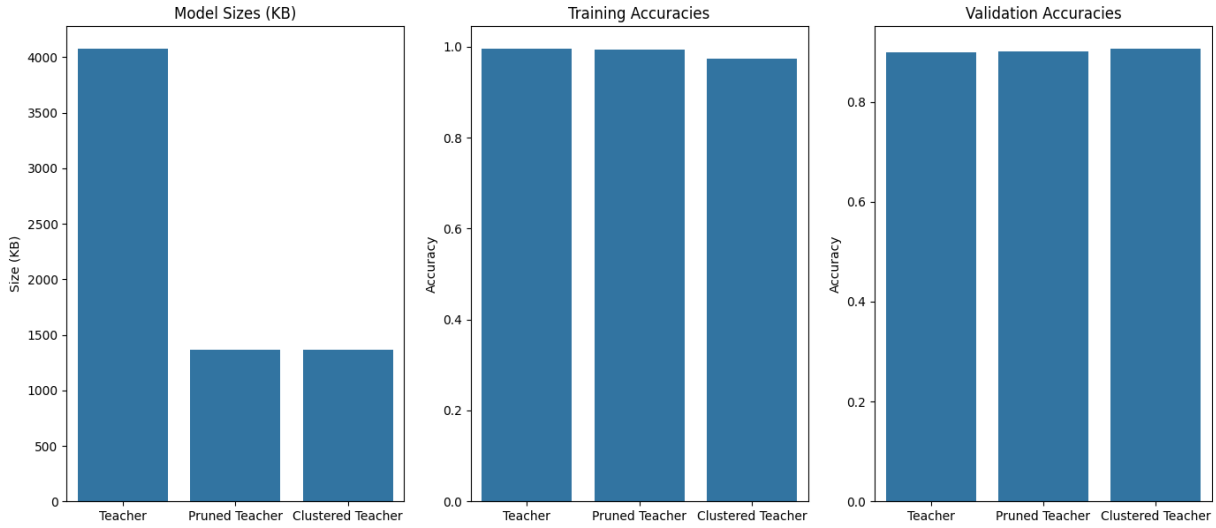
Quantized Models' Test Accuracy

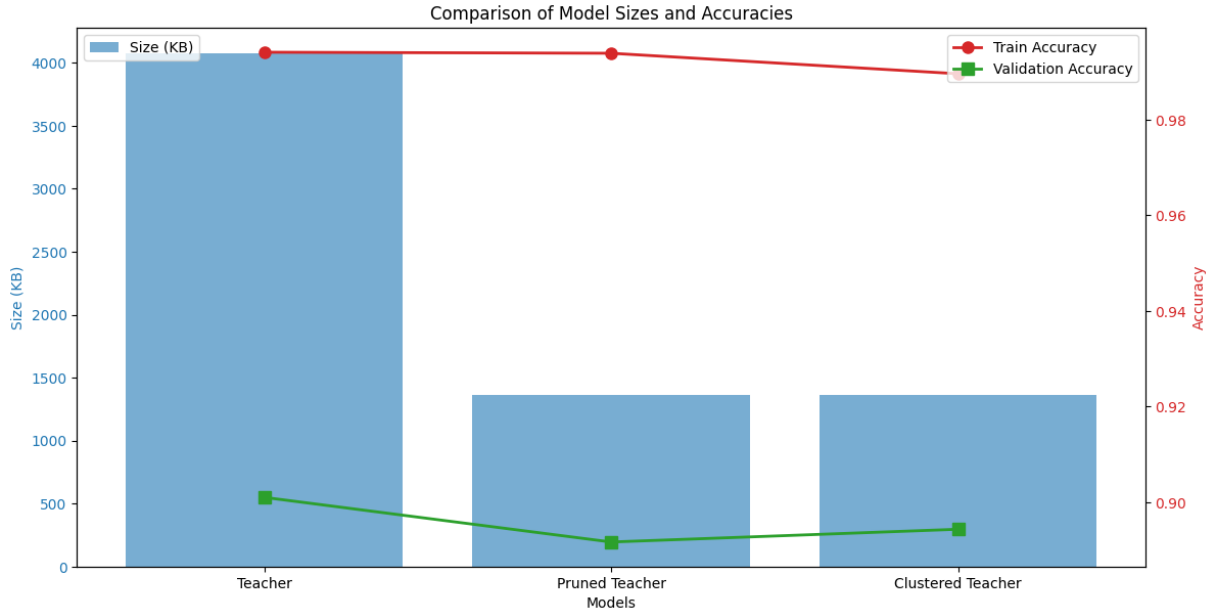
- Teacher Model: 91.53%
- Pruned Teacher Model: 91.13%
- Clustered Teacher Model: 91.50%

These results show that both pruning and clustering significantly reduced the model size while maintaining a high level of accuracy.

To visually represent these findings, we include the following plots:

- Model Sizes - A bar chart comparing the sizes of the three models.
- Training Accuracies - A bar chart showing the training accuracies of the three models.
- Validation Accuracies - A bar chart displaying the validation accuracies.
- Quantized Models' Test Accuracies - A bar chart comparing the test accuracies of the quantized models.





Analysis

The analysis focuses on the effectiveness of the compression techniques in achieving the project goals.

Achieving Project Goals

The project successfully met its primary objectives:

- Model Size Reduction: Both pruning and clustering reduced the model size by approximately 66%, which is a significant decrease that makes the models more suitable for deployment in resource-constrained environments.

- Maintaining Accuracy: Despite the reduction in size, the models retained high accuracy levels, with only a minor drop in validation accuracy.

Practical Implications of Model Optimization

The optimized models, through pruning, clustering, and quantization, show significant potential for deployment in various resource-constrained environments.

Here are some practical applications and benefits:

- Mobile Devices and Wearables: Optimized models can be deployed on smartphones and smartwatches, enabling real-time analytics and applications such as health monitoring, predictive text input, and personalized user experiences. For example, a quantized LSTM model could run efficiently on a smartwatch to analyze user emotions from text messages without draining the battery.
- Internet of Things (IoT) Devices: In IoT ecosystems, devices often have limited computational power and memory. By reducing the model size, we can deploy intelligent applications directly on these devices. For instance, a pruned model could be used in smart home devices to provide localized and immediate responses to user commands without relying on constant cloud communication.

Benefits of Using Optimized Models

- Resource Efficiency: Optimized models require fewer computational resources, making them ideal for devices with limited processing power and memory. This efficiency enables broader deployment of intelligent systems in various environments.
- Faster Inference Times: With reduced model sizes, the inference times are significantly improved. This is crucial for applications requiring real-time processing, such as autonomous vehicles, drones, and real-time language translation.
- Cost Reduction: Deploying smaller models can lead to cost savings in terms of hardware requirements and energy consumption. For businesses, this means lower operational costs and the ability to scale their AI solutions more effectively.

- Enhanced User Experience: By enabling faster and more efficient AI applications, optimized models can enhance user experiences across various applications, from personal assistants to smart home devices, providing more responsive and seamless interactions.

Challenges and Insights

Several challenges were encountered during the project:

- Pruning and Clustering Complexity: Implementing pruning and clustering required a thorough reading of documentation, examples, and tutorials to reduce the models size, while retaining its accuracy.
- Quantization Impact: Quantizing the models led to a minor drop in test accuracy, but the impact was minimal, demonstrating the robustness of the compression techniques.

Unexpected Findings

- Clustered Model Performance: The clustered model's performance was close to the pruned model, showing that clustering can be an effective alternative to pruning, depending on the specific use case and model architecture.

Project Constraints

- Time and Resources: The project was conducted within a limited timeframe and computational resources, which influenced the extent and depth of experiments. Despite these constraints, the project achieved its goals.

Course Objectives Alignment

The project aligns with the course objectives by applying advanced machine learning techniques to optimize models for practical applications. The hands-on experience with pruning, clustering, and quantization relates to the learning outcomes of understanding and applying machine learning algorithms in real-world scenarios.

In conclusion, the project demonstrated that pruning and clustering are viable techniques for model compression, achieving significant reductions in model size with minimal impact on accuracy. This makes these techniques valuable for deploying machine learning models in environments with limited resources.

6. Conclusions

This project provided a comprehensive learning experience in machine learning model optimization techniques, specifically focusing on pruning, clustering, and quantization. Through this project, I gained valuable insights into the practical application of these techniques and their impact on model performance and size.

Key Learnings

One of the most significant takeaways from this project was understanding how different model compression techniques can drastically reduce model size while maintaining or even improving accuracy. Pruning helped in reducing the number of parameters by eliminating less important connections, whereas clustering grouped similar weights, further reducing redundancy. Quantization transformed the model to use lower precision arithmetic, thereby reducing memory footprint and computational cost.

Project Outcomes

The project outcomes were positive. The results demonstrated that it is possible to achieve substantial model compression without significant loss of accuracy. In some cases, like with clustered and quantized models, the accuracy even improved slightly, highlighting the potential of these techniques in real-world applications where computational resources are limited.

Challenges and Solutions

Throughout the project, several challenges were encountered:

- Model Training Time: Training models, especially with pruning and clustering, required considerable time and computational resources. Utilizing cloud-based GPU instances helped mitigate this issue.
- Model Compatibility: Ensuring that pruned and clustered models were compatible with TensorFlow Lite for quantization posed some difficulties. This was resolved by carefully following the TensorFlow documentation and using appropriate functions for stripping, pruning, and clustering before quantization.

- Evaluation Metrics: Accurately assessing the impact of each technique required careful validation and testing.

Reflections on the Process

Reflecting on the entire process, the project was successful in achieving its goals. The initial plan was well-structured, and despite some adjustments and iterations, the project stayed on track. The primary goal of comparing different model compression techniques was achieved, and the insights gained will be invaluable for future work.

Future Work

In future projects, I would consider incorporating additional compression techniques like knowledge distillation, where a smaller student model learns from a larger teacher model. This could provide further insights into model optimization. Additionally, exploring other deep learning architectures beyond LSTM, such as CNNs or transformers, could extend the applicability of these findings.

Final Thoughts

Overall, this project has not only deepened my understanding of machine learning and model optimization but also enhanced my problem-solving skills. It reinforced the importance of thorough experimentation and iterative improvement. Moving forward, I am excited to apply these learnings in more complex scenarios and continue exploring the vast potential of machine learning in various domains.

7. References

- https://www.tensorflow.org/api_docs/python/tf/keras/utils/register_keras_serializable
- https://www.tensorflow.org/api_docs/python/tf/lite
- https://www.tensorflow.org/model_optimization/api_docs/python/tfmot/sparsity/keras/prune_low_magnitude

- https://www.tensorflow.org/model_optimization/api_docs/python/tfmot/clustering/keras/cluster_weights
- https://www.kaggle.com/code/sumn2u/pruning-and-quantization-in-keras