Model Compression for AI

23/10/2025

1 Introduction - Sending a Big Package Through a Small Mailbox

Imagine you're trying to send a huge, detailed model of a spaceship through the mail, but the mailbox slot is tiny. You can't just shove the whole model through—it won't fit! Instead, you carefully take apart the model, keeping only the most essential pieces, and pack them tightly to fit through the slot. When the recipient gets the package, they rebuild the model, maybe not with every tiny detail, but close enough to recognize it as the same spaceship. This process is a lot like model compression in machine learning, where we make large neural networks smaller and faster so they can run on devices like phones or smartwatches, while still performing well.

In this book, we'll explore two key techniques for model compression: *pruning* and *quantization*. These methods help shrink neural networks and make them more efficient, much like trimming unnecessary parts from the spaceship model or packing it more cleverly. We'll break down these concepts in a beginner-friendly way, with examples and visuals to guide you.

2 What is Model Compression?

Neural networks, especially deep ones used for tasks like image recognition or language processing, can be massive. They often have millions of parameters, requiring a lot of memory and computational power. This makes them slow and impractical for devices with limited resources, like mobile phones or embedded systems. Model compression is the process of reducing the size and complexity of these networks while trying to maintain their accuracy.

Think of a neural network as a huge recipe book for solving a problem. Compression is like rewriting the book to use fewer ingredients and simpler steps, but still produce a tasty dish. The two main techniques we'll cover are pruning (cutting out unnecessary parts of the network) and quantization (using simpler numbers to represent the network's parameters).

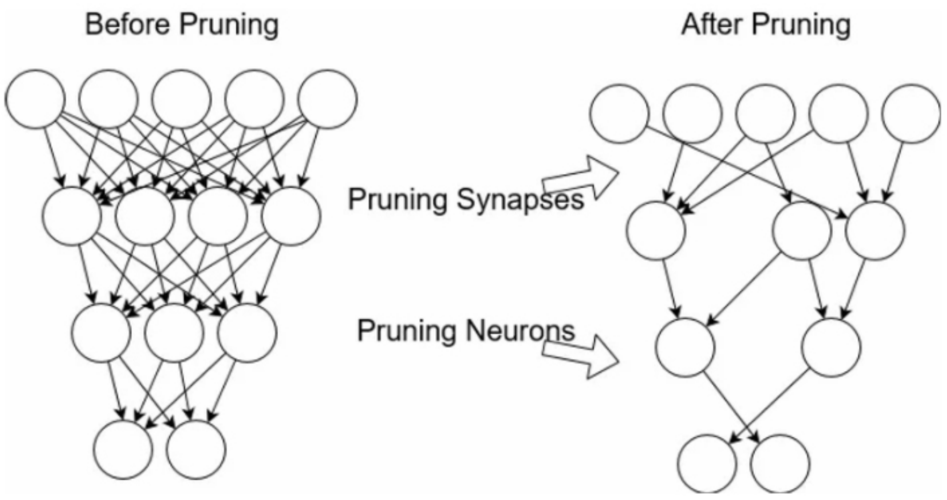


Figure 1: Overview of a neural network before and after compression

3 Pruning: Trimming the Excess

Pruning is like trimming a bush to keep it healthy and manageable. In neural networks, it involves removing parts—such as connections, neurons, or entire layers—that contribute little to the model's predictions. By cutting these less important parts, we make the network smaller and faster without losing too much accuracy.

3.1 How Pruning Works

Pruning typically involves these steps:

1. **Train the Network:** Start with a fully trained neural network.
2. **Identify Redundant Parts:** Analyze the network to find weights (connections between neurons) or neurons with low importance. For example, weights with values close to zero often have little impact on the output.
3. **Remove Parts:** Eliminate these low-impact weights or neurons.
4. **Fine-Tune:** Retrain the trimmed network to recover any lost accuracy.

Imagine you're organizing a group project. Some team members contribute a lot, while others barely participate. By politely asking the less active members to step back, the team becomes more efficient without losing much productivity. Pruning does the same for neural networks.

3.2 Types of Pruning

There are several ways to prune a network:

- **Weight Pruning:** Remove individual connections (weights) with small values. This creates a sparse network, where many connections are set to zero.
- **Neuron Pruning:** Remove entire neurons that don't contribute much.
- **Structured Pruning:** Remove larger structures, like entire filters in a convolutional neural network, to make the network more hardware-friendly.

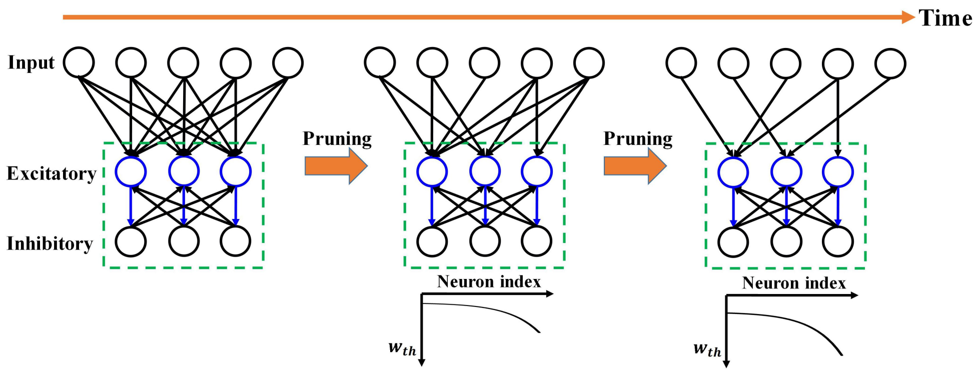


Figure 2: Example of weight pruning in a neural network

3.3 Benefits and Challenges

Pruning reduces the model's size and speeds up inference, making it ideal for devices with limited memory or processing power. For example, a pruned model can run faster on a smartphone, enabling real-time applications like voice assistants.

However, pruning isn't perfect. Removing too many parts can hurt accuracy, and finding the right balance requires careful experimentation. It's like trimming a bush—you want it to look good without cutting away too many healthy branches.

4 Quantization: Simplifying the Numbers

Quantization is about making the numbers in a neural network simpler. Neural networks often use high-precision numbers (like 32-bit floating-point values) for their weights and activations. Quantization reduces these to lower-precision formats, like 8-bit integers, to save memory and speed up computations.

4.1 How Quantization Works

Think of quantization like rounding numbers in a recipe. Instead of measuring 2.374 cups of flour, you round to 2 cups—it's close enough and easier to work with. In a neural network:

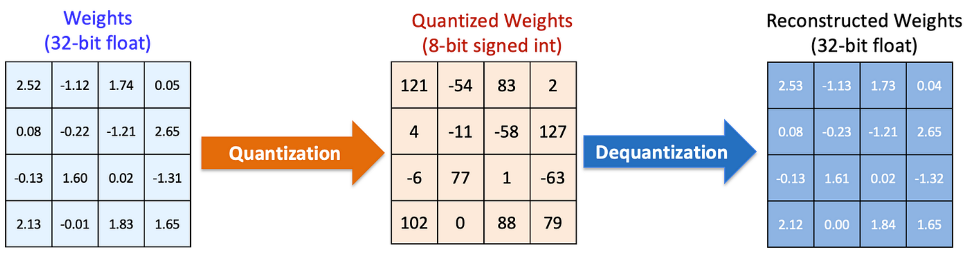
1. **Analyze the Range:** Look at the range of values for weights and activations.
2. **Map to Lower Precision:** Convert these values to a smaller set of numbers (e.g., from 32-bit floats to 8-bit integers).
3. **Adjust the Model:** Fine-tune the model to adapt to these simpler numbers.

For example, a weight of 0.123456 might be rounded to 0.12 in a quantized model. This reduces the memory needed to store the weight and makes calculations faster.

4.2 Types of Quantization

There are two main approaches:

- **Post-Training Quantization:** Apply quantization to a pre-trained model. This is simpler but may lead to some accuracy loss.
- **Quantization-Aware Training:** Train the model with quantization in mind, simulating low-precision numbers during training to minimize accuracy loss.



2.52	-1.12	1.74	0.05
0.08	-0.22	-1.21	2.65
-0.13	1.60	0.02	-1.31
2.13	-0.01	1.83	1.65

121	-54	83	2
4	-11	-58	127
-6	77	1	-63
102	0	88	79

2.53	-1.13	1.73	0.04
0.08	-0.23	-1.21	2.65
-0.13	1.61	0.02	-1.32
2.12	0.00	1.84	1.65

Figure 3: Illustration of quantization from 32-bit floats to 8-bit integers

4.3 Benefits and Challenges

Quantization significantly reduces model size and speeds up inference, making it ideal for edge devices like IoT sensors. For instance, a quantized model might use 4 times less memory than the original, enabling it to run on a tiny microcontroller.

The challenge is maintaining accuracy. Lower-precision numbers can introduce errors, like rounding 2.374 cups to 2 cups slightly changing the recipe's outcome. Careful tuning is needed to keep the model effective.

5 Combining Pruning and Quantization

Pruning and quantization are often used together for maximum efficiency. Pruning removes unnecessary parts, making the network smaller, while quantization simplifies the remaining parts, making computations faster. It's like trimming the spaceship model and then packing it into a smaller, lighter box.

For example, a neural network for image classification might first be pruned to remove 50% of its connections, reducing its size. Then, quantization can convert its weights from 32-bit floats to 8-bit integers, further shrinking the model and speeding up inference. The result is a compact model that runs efficiently on resource-constrained devices.

Table 1: Comparison of Pruning and Quantization

Feature	**Pruning**	**Quantization**
Goal	Remove unnecessary connections/neurons	Reduce precision of weights/activations
Impact	Smaller, sparser model	Smaller, faster computations
Complexity	Requires identifying low-impact parts	Requires mapping to lower-precision numbers
Accuracy Trade-off	May reduce accuracy if over-pruned	May introduce errors if precision is too low

6 Practical Applications

Model compression is critical for deploying AI in the real world. Here are some examples:

- **Mobile Devices:** Pruned and quantized models enable real-time image or speech recognition on smartphones with limited battery and memory.
- **IoT Devices:** Tiny sensors with compressed models can monitor environments or detect anomalies without needing powerful hardware.
- **Autonomous Vehicles:** Compressed models process sensor data quickly, ensuring fast decision-making in self-driving cars.

7 Conclusion

Model compression through pruning and quantization is like preparing a lightweight, efficient version of a complex machine. By trimming unnecessary parts and simplifying numbers, we make neural networks faster and smaller, enabling them to run on everyday devices. As you dive deeper into AI, experimenting with these techniques will help you build models that are not only powerful but also practical for real-world use.