Minor in AI

TinyML

Model Pruning

May 30, 2025

1. Introduction to Model Pruning

Definition

Model pruning refers to the process of reducing the complexity of a deep neural network by removing parameters (weights, neurons, filters, or even entire layers) that are considered redundant or have minimal impact on output prediction.

Why is pruning needed?

- Modern DNNs are often over-parameterized — many weights contribute little to the final output.
- Edge devices require compact and fast models with low memory and power consumption.
- Pruning helps to:
  - Reduce model size (memory footprint)
  - Speed up inference
  - Lower energy consumption
  - Preserve or even improve generalization (by acting as a regularizer)

Think!

How would you explain the benefit of pruning to someone building an app for real-time gesture recognition on a smartwatch?

2. Types of Pruning Techniques

2.1 Unstructured Pruning

- Removes individual weights in the weight matrix.
- Results in sparse matrices with many zeros.
- Requires specialized hardware/software to benefit from speedups.
- Often done based on a magnitude threshold: weights below a certain absolute value are set to zero.

Example Criteria

Prune weights w such that $|w| < \epsilon$ for some small ϵ.

2.2 Structured Pruning

- Removes entire structures like neurons, filters, channels, or even layers.
- Maintains dense tensors — better compatibility with existing hardware.
- More interpretable and practically efficient.

Structured Pruning Examples

- **Neuron Pruning:** Remove low-activation neurons in dense layers.
- **Filter Pruning:** Eliminate entire convolutional filters (3D kernel weights).
- **Channel Pruning:** Remove input/output channels in CNN layers.
- **Layer Pruning:** Drop entire layers from deep networks (e.g., residual blocks).

2.3 Pruning Time Strategies

- **Post-training Pruning:** Apply after full training; retrain (fine-tune) to recover accuracy.
- **During-training Pruning:** Prune while training; requires pruning schedules.
- **Progressive/Dynamic Pruning:** Gradually increase sparsity over epochs.

3. Pruning in Convolutional Neural Networks (CNNs)

CNN Characteristics

CNNs use convolutional filters (kernels) to extract spatial features. Many filters may be redundant or learn similar patterns.

How Pruning Works in CNNs

- **Filter Pruning:** Remove entire convolutional filters whose weights are small in magnitude or whose feature maps are not useful (e.g., low activation).
- **Channel Pruning:** Affects subsequent layers that consume the output.
- **Structured pruning** works better in CNNs due to spatial correlations and filter dependencies.

Effect of Pruning on CNN Architecture

- Reduction in number of feature maps.
- Reduction in parameters and FLOPs.
- Minor drop in accuracy can often be recovered by fine-tuning.

4. Transfer Learning + Pruning: A Powerful Combination

Transfer Learning

Transfer learning leverages a pre-trained model (e.g., MobileNet, ResNet) and adapts it to a new task by:

- Freezing earlier layers.
- Adding custom output layers.
- Fine-tuning with a small dataset.

Where to Apply Pruning in Transfer Learning?

- **Prune only the custom head:** Safest option. Keeps base stable.
- **Prune middle layers:** If retraining is possible and resources allow.
- **Prune entire blocks:** Try dropping residual units in ResNet for simplification.

Case Study: Fashion MNIST with MobileNetV2

- Use MobileNetV2 as base (without the top classifier).
- Add: GlobalAveragePooling → Dense(128) → Dropout → Dense(10, softmax)
- Apply pruning on the custom Dense layers.
- Train with a pruning schedule: start from 0% sparsity and increase to 50%.

5. TensorFlow Implementation Outline

Code Concept: Pruning With TF MOT

```
import tensorflow_model_optimization as tfmot

prune_low_magnitude = tfmot.sparsity.keras.prune_low_magnitude

pruning_schedule = tfmot.sparsity.keras.PolynomialDecay(
    initial_sparsity=0.0,
    final_sparsity=0.5,
    begin_step=0,
    end_step=end_step
)

pruned_model = prune_low_magnitude(model, pruning_schedule=pruning_schedule)
```

Evaluation Metrics

- Accuracy (before and after pruning)
- Model size (MB)
- Number of non-zero parameters
- Inference time

6. Practical Considerations and Trade-offs

- Too much pruning ⇒ underfitting or unstable training.
- Always evaluate performance after pruning.
- Fine-tuning is often necessary to recover lost accuracy.
- Pruned models may require hardware support for sparse ops.

7. Key Takeaways

1. Model pruning is a key compression technique in deep learning.
2. Unstructured pruning leads to sparse models; structured pruning leads to hardware-friendly models.
3. CNNs benefit from filter and channel pruning.
4. Transfer learning with pruning leads to lightweight, high-performing models.
5. Fine-tuning is crucial to regain performance after pruning.