1. Pruning type

Pruning type refers to the structural granularity at which parameters are removed from a neural network, and is generally categorized into three forms: **unstructured**, **semi-structured**, and **structured**. Unstructured pruning removes individual weights without regard for their location, resulting in fine-grained sparsity that typically requires specialized hardware or software for acceleration. Structured pruning eliminates entire computational units such as channels, filters, layers, or attention heads, producing a smaller, regular architecture that can be directly accelerated on general-purpose hardware. Semi-structured pruning takes a middle ground by enforcing fixed sparsity patterns (e.g., 2:4 sparsity) within weight tensors, balancing computational regularity with flexibility to preserve accuracy.

A. Group Magnitude Importance (p=1)

Group Magnitude Importance (p=1) in the context of torch-pruning is a metric used to evaluate the importance of structural groups (e.g., convolutional filters, channels, or neurons) in a neural network for the purpose of structured pruning. It relies on the L1 norm to quantify the significance of a group based on the magnitude of its weights. Below is a theoretical explanation:

Concept

Group Magnitude Importance (p=1) in the context of torch-pruning is a metric used to evaluate the importance of structural groups (e.g., convolutional filters, channels, or neurons) in a neural network for the purpose of structured pruning. It relies on the L1 norm to quantify the significance of a group based on the magnitude of its weights. Below is a theoretical explanation:

- Group: A group refers to a structural unit in the network, such as a filter in a convolutional layer, a channel, or a set of neurons. Structured pruning targets these groups to maintain hardware-friendly model architectures.

- L1 Norm (p=1): The L1 norm is used to calculate importance, where the importance score of a group is the sum of the absolute values of all weights in that group. Mathematically, for a group (G):

$$Importance(G) = \sum_{w \in G} |w|$$

Here, (w) represents individual weights within the group, and ($|w|$) is the absolute value of each weight.

Theoretical Basis

- Rationale: The L1 norm provides a straightforward measure of a group's contribution by focusing on the magnitude of its weights. Groups with lower L1 norm scores are considered less important because their weights are smaller, implying a weaker influence on the network's output.
- Pruning Process: During pruning, groups are ranked by their importance scores. Those with the lowest L1 norm values are removed, as they are assumed to have the least impact on the model's performance. After pruning, fine-tuning can help recover any potential performance loss.
- Structural Pruning: Unlike unstructured pruning (which removes individual weights), Group Magnitude Importance focuses on entire groups, ensuring that the pruned model retains a structured architecture suitable for efficient hardware execution.

Advantages

- Simplicity: The L1 norm is computationally efficient and easy to implement, making it a practical choice for importance evaluation.
- Structure Preservation: By pruning entire groups (e.g., filters or channels), the method aligns with hardware optimizations, as it reduces the model's computational footprint in a structured way.
- Intuitive: The approach assumes that larger weights are more critical to the network's function, providing a clear heuristic for pruning decisions.

Limitations

- Magnitude-Only Focus: The L1 norm only considers the absolute size of weights, ignoring their distribution or functional importance in the network. A group with small weights might still be critical to the model's output.

- Potential Information Loss: Pruning based solely on magnitude may remove groups that contribute to the model's robustness or generalization, especially if their weights are small but strategically important.
- Context Insensitivity: The method does not account for the interactions between groups or the broader network architecture, which may lead to suboptimal pruning decisions.

B. Group Magnitude Importance (p=2)

Group Magnitude Importance (p=2) in the context of torch-pruning is a metric used to evaluate the importance of structural groups (e.g., convolutional filters, channels, or neurons) in a neural network for structured pruning. It relies on the L2 norm (Euclidean norm) to quantify the significance of a group based on the magnitude of its weights. Below is a theoretical explanation:

Concept

Group Magnitude Importance (p=2) measures the importance of a group by computing the L2 norm of the weights within that group. The L2 norm represents the square root of the sum of the squared weights, effectively capturing the Euclidean magnitude of the group's weight vector. Groups with smaller L2 norm values are considered less important and are prioritized for pruning.

- Group: A group refers to a structural unit in the network, such as a filter in a convolutional layer, a channel, or a set of neurons. Structured pruning targets these groups to maintain a hardware-efficient model architecture.
- L2 Norm (p=2): The L2 norm is used to calculate importance, where the importance score of a group (G) is defined as:

$$Importance(G) = \sqrt{\sum_{w \in G} w^2}$$

Here, (w) represents individual weights within the group, and $w^2$is the square of each weight. The square root of the sum provides a measure of the group's overall magnitude.

Theoretical Basis

- Rationale: The L2 norm emphasizes the collective magnitude of weights in a group, giving more weight to larger individual weights due to the squaring operation. This makes it sensitive to outliers (i.e., weights with large magnitudes) within the group. Groups with lower L2 norm scores are assumed to have less impact on the network's output and are candidates for removal.
- Pruning Process: During structured pruning, groups are ranked based on their L2 norm scores. Those with the smallest L2 norms are pruned, as they are deemed to contribute less to the model's functionality. Post-pruning fine-tuning can help mitigate any performance degradation.
- Structural Pruning: By focusing on entire groups rather than individual weights, this method ensures that the pruned model maintains a structured architecture, which is advantageous for hardware acceleration and efficient inference.

Advantages

- Smooth Magnitude Measure: The L2 norm provides a smooth and continuous measure of importance, accounting for the collective effect of weights in a group. It is less sensitive to small, noisy weight variations compared to other norms.
- Emphasis on Larger Weights: The squaring in the L2 norm amplifies the contribution of larger weights, making it effective at identifying groups with significant influence.
- Hardware Compatibility: Pruning entire groups (e.g., filters or channels) aligns with hardware optimizations, reducing computational overhead in a structured manner.

Limitations

- Magnitude-Only Focus: Like other magnitude-based methods, the L2 norm focuses solely on weight magnitudes, potentially overlooking groups that are functionally important despite having smaller weights.
- Sensitivity to Outliers: The squaring operation in the L2 norm can overly emphasize large weights, which may lead to pruning decisions that undervalue groups with many small but collectively important weights.
- Context Insensitivity: The L2 norm does not consider the role of a group within the broader network architecture or its interactions with other groups, which could result in suboptimal pruning choices.

Comparison with p=1

Compared to Group Magnitude Importance (p=1), which uses the L1 norm (sum of absolute weights), the L2 norm (p=2) places greater emphasis on larger weights due to the squaring operation. This makes p=2 more sensitive to the presence of large weights within a group, while p=1 treats all weights equally in terms of their absolute values. As a result, p=2 may prioritize keeping groups with a few large weights over groups with many small weights, even if their L1 norms are similar.

C. BN Scale Importance

In the context of neural network pruning, particularly with libraries like torch-pruning, BN Scale Importance refers to an importance metric that evaluates the significance of structural groups (e.g., channels, filters, or neurons) in a neural network based on the scale parameters (γ, gamma) of Batch Normalization (BN) layers. Below is a theoretical explanation of BN Scale Importance, focusing on its role in structured pruning.

Concept

Batch Normalization (BN) is a widely used technique in deep neural networks to stabilize and accelerate training by normalizing the activations of each layer. A BN layer applies a linear transformation to the normalized activations using two learnable parameters: a scale factor (γ) and a shift factor (β). The scale parameter γ controls the magnitude of the output for each channel or feature, effectively determining how much that channel contributes to the network's output.BN Scale Importance uses the magnitude of the γ parameter (typically its absolute value, |γ|) to assess the importance of a specific channel or filter in a neural network. The intuition is that a channel with a small |γ| has a minimal impact on the network’s output, as it scales the normalized activations to near zero, making it a candidate for pruning.

$$Importance(G) = |\gamma_G|$$

where γ_G is the scale parameter corresponding to the group (G) (e.g., a specific channel or filter).

Theoretical Basis

- Rationale: The scale parameter γ in a BN layer directly influences the magnitude of a channel’s output. A small |γ| indicates that the channel’s contribution to the network’s output is negligible, as its activations are scaled down. Conversely, a large |γ| suggests that the channel plays a significant role in the network’s computations. By using |γ| as an importance metric, BN Scale Importance identifies channels that can be pruned with minimal impact on the model’s performance.

- Pruning Process: During structured pruning, channels or filters are ranked based on their BN scale importance scores (i.e., |γ|). Those with the lowest scores are removed, as they are deemed less critical. After pruning, the model’s architecture is modified, and fine-tuning is typically applied to recover any performance loss.
- Structured Pruning: BN Scale Importance is particularly suited for structured pruning, as it targets entire channels or filters, preserving the network’s structural integrity and ensuring compatibility with hardware optimizations.

Advantages

- Simplicity: The metric is straightforward to compute, as it relies solely on the absolute value of the γ parameter, which is readily available in BN layers.
- Direct Relevance to BN: Since γ directly controls the output scale of a channel, it provides a more precise measure of a channel’s contribution compared to weight-based metrics like Group Magnitude Importance.
- Hardware Efficiency: Pruning based on BN Scale Importance removes entire channels or filters, resulting in a compact model that is optimized for efficient inference on hardware.

Limitations

- Dependence on BN Layers: BN Scale Importance is only applicable to networks that use Batch Normalization. Models without BN layers cannot use this metric.
- Magnitude-Only Focus: Like other magnitude-based metrics, BN Scale Importance relies solely on the scale parameter’s magnitude, which may not fully capture the functional importance of a channel (e.g., its role in feature representation or interactions with other layers).
- Sensitivity to Training Dynamics: The γ parameters are learned during training and can be influenced by factors like initialization or optimization. A small |γ| might not always indicate a truly unimportant channel, especially if training is not fully converged.
- Limited Contextual Awareness: The metric evaluates channels in isolation based

on their γ values, potentially overlooking dependencies or interactions between channels across the network.

Comparison with Group Magnitude Importance

- Group Magnitude Importance (p=1 or p=2) evaluates importance based on the L1 or L2 norm of the weights in a group (e.g., filter or channel weights). In contrast, BN Scale Importance focuses on the γ parameter of the BN layer, which directly scales the output of a channel. This makes BN Scale Importance more specific to the output contribution of a channel, whereas Group Magnitude Importance considers the raw weight magnitudes, which may not directly reflect the channel's impact on the network's output.
- Use Case: BN Scale Importance is preferred when pruning networks with BN layers, as it leverages the learned scale parameters. Group Magnitude Importance is more general and can be applied to any network, regardless of BN usage.

2. Pruning method

Pruning method refers to the strategy for deciding how and when to remove parameters from a network, in contrast to pruning type, which defines what structural unit (e.g., individual weights, channels, filters) is removed. Methods are generally categorized by the importance criterion used (e.g., magnitude-based, sensitivity-based, similarity/clustering-based, learned via sparsity regularization or reinforcement learning), the timing of pruning (before, during, or after training), and the schedule (one-shot vs. iterative vs. dynamic). For example, BN Scale Pruner falls under a magnitude-based criterion method, using the absolute value of BatchNorm's scale parameter (γ) to measure channel importance; Group Norm Pruner is also magnitude-based, but scores each group (channel or filter) by its L1/L2 norm; and Growing Reg Pruner belongs to sparsity regularization-based methods, progressively applying a regularization term during training to suppress and remove less important structures.

A. BN Scale Pruner

The BN Scale Pruner, often referred to as "Network Slimming," is a neural network pruning technique introduced in the paper Learning Efficient Convolutional Networks through Network Slimming (arXiv:1708.06519), published at ICCV 2017. It leverages the scaling factors (γ) in Batch Normalization (BN) layers to identify and prune unimportant channels in convolutional neural networks (CNNs), achieving model compression and computational efficiency without significant accuracy loss.

Concept

BN Scale PrunerBN Scale Pruner, or Network Slimming, is designed to address three key challenges in deploying large CNNs:

- Model Size: Large networks have millions of parameters, requiring significant storage.
- Run-Time Memory: Intermediate activations consume substantial memory during inference.
- Computational Cost: Convolution operations are computationally intensive, especially on resource-constrained devices like mobile phones.

The method introduces a simple yet effective approach by imposing sparsity on the scaling factors (γ) of BN layers during training. These scaling factors are used to identify and prune less important channels, resulting in a compact, efficient model that maintains comparable accuracy.

How BN Scale Pruner Works

The BN Scale Pruner operates by integrating sparsity-inducing regularization into the training process, specifically targeting the γ parameters in BN layers. Here's a step-by-step explanation:

- Batch Normalization and Scaling Factors
  - In CNNs, BN layers normalize the output of convolutional layers to stabilize and accelerate training. Each BN layer includes a learnable scaling factor (γ) and a shift parameter (β) for each channel.
  - The γ factor scales the normalized output of a channel, directly influencing its contribution to the network's output. A small γ value reduces the channel's impact, making it a candidate for pruning.

Sparsity-Induced Regularization

- During training, the BN Scale Pruner adds an L1 regularization term to the loss function, specifically targeting the γ factors. The modified loss function is:
  - Loss = Training Loss (W, x, y) + λ * Σ|γ|,
  - where (x, y) are the input and target, W is the network's trainable parameters, λ is a balancing factor, and Σ|γ| is the L1 norm of the scaling factors.
- L1 regularization encourages many γ values to approach zero, effectively reducing the influence of less important channels.

Channel Pruning

- After training with L1 regularization, channels with small γ values (close to zero) are considered unimportant because their contribution to the network's output is minimal.

- A global threshold is set to determine which channels to prune. For example, to prune 70% of channels, the algorithm sorts all γ values across the network and selects the threshold at the 70th percentile of their absolute values.
- Channels with γ values below this threshold are pruned by removing the corresponding convolutional filters and their associated feature maps, reducing the model's size and computational complexity.

Fine-Tuning

- Pruning may temporarily degrade performance. To mitigate this, the pruned model undergoes fine-tuning (additional training) to recover accuracy, ensuring the compact model performs comparably to the original.

Multi-Pass Slimming (Optional)

- The process can be repeated iteratively: train with L1 regularization, prune channels, fine-tune, and repeat. This multi-pass approach yields even more compact models with minimal accuracy loss.

Key Mechanism: Why BN Scaling Factors?

- Channel Importance: The γ factor in a BN layer directly scales the output of a channel. A small γ indicates the channel contributes little to the network's output, as its feature map is scaled down and subsequent activations (after activation functions like ReLU) may approach zero.
- Sparsity via L1 Regularization: L1 regularization pushes γ values toward zero for

less important channels, making it easy to identify and remove them without complex computations (e.g., Hessian-based methods).

- Compatibility: Since BN layers are common in modern CNN architectures (e.g., ResNet, VGG, DenseNet), the method integrates seamlessly without requiring architectural changes.

Advantages

- Simplicity: The method is straightforward, adding only an L1 regularization term to the loss function, with no need for specialized hardware or software.
- Comprehensive Efficiency: It simultaneously reduces model size (up to 20x for VGGNet), run-time memory, and computational operations (up to 5x fewer FLOPs).
- Preserved Accuracy: Fine-tuning ensures the pruned model retains accuracy comparable to or sometimes better than the original due to the regularization's generalization benefits.
- Broad Applicability: Works with various CNN architectures (VGGNet, ResNet, DenseNet) and datasets (CIFAR-10, CIFAR-100, ImageNet).
- No Special Accelerators Needed: The resulting pruned model is dense and can run efficiently on standard hardware, unlike sparse models requiring specialized libraries.

Limitations

- Dependence on BN Layers: The method relies on BN layers, limiting its applicability to architectures without them (though BN is standard in most modern CNNs).

- Global Threshold Sensitivity: Choosing an appropriate pruning threshold requires careful tuning to balance compression and accuracy.
- Comparison Scope: The paper could benefit from comparisons with other iterative pruning methods to further validate its superiority.

B. Group Norm Pruner

Below is a theoretical explanation of the GroupNorm Pruner in the context of the torch-pruning framework, based on the paper DepGraph: Towards Any Structural Pruning. The explanation excludes code, as requested, and follows the same format used for the BN Scale Pruner and Growing Regularization Pruner (Concept, Theoretical Basis, Advantages, Limitations, Comparison with Other Pruners, Use in torch-pruning, Summary).

Concept

The GroupNorm Pruner is a high-level structural pruning method within the torch-pruning framework, designed to remove structurally grouped parameters (e.g., channels, filters, or neurons) from neural networks to reduce model complexity while maintaining performance. It leverages a group-level importance criterion based on the L2 norm of parameters within a dependency group, as facilitated by the Dependency Graph (DepGraph) introduced in the paper. Unlike pruners that evaluate individual layers independently, the GroupNorm Pruner considers the interdependencies between layers, ensuring that coupled parameters across multiple layers are pruned consistently to avoid structural issues. It is particularly suited for complex network architectures (e.g., CNNs, Transformers, RNNs, GNNs) where parameter dependencies are intricate.

- Group-Level Importance: The pruner evaluates the importance of a group of parameters (e.g., a channel across multiple layers) by computing the L2 norm of the weights within the group, as defined by the DepGraph.
- Target: The pruner focuses on structured pruning, removing entire structural units such as channels, filters, or neurons, ensuring compatibility with hardware acceleration.

Theoretical Basis

The GroupNorm Pruner builds on the Dependency Graph (DepGraph), a core contribution of the paper, which explicitly models inter-layer dependencies in neural networks. The pruner addresses the challenge of structural pruning, where removing a parameter in one layer (e.g., a channel in a convolutional layer) triggers the need to prune corresponding parameters in coupled layers (e.g., BatchNorm or subsequent convolutional layers).

- Dependency Graph (DepGraph): The DepGraph automatically identifies groups of parameters that must be pruned together due to structural coupling (e.g., residual connections in ResNet or attention mechanisms in Transformers). It represents the network as a graph where nodes are layers and edges denote dependencies, allowing the pruner to trace and group coupled parameters efficiently.
- Importance Criterion: The GroupNorm Pruner uses a group-level L2 norm to evaluate the importance of a pruning group:

$$Importance(G) = \sqrt{\sum_{w \in G} w^2}$$

where (G) is a group of parameters (e.g., weights corresponding to a channel across multiple layers), and (w) represents the weights within that group. Groups with lower L2 norms are considered less important and are prioritized for removal.

- Consistent Sparsity: To ensure that all parameters within a group are consistently unimportant, the pruner employs sparsity-inducing techniques (e.g., L2 regularization) during training. This aligns the importance of coupled parameters, preventing structural inconsistencies after pruning.
- Pruning Process:
    1. The DepGraph identifies all prunable groups in the network.
    2. The GroupNorm Pruner computes the L2 norm for each group.
    3. Groups with the lowest L2 norms are removed based on a specified pruning ratio.
    4. The model is fine-tuned post-pruning to recover performance.
- Structural Pruning: The pruner ensures that pruning respects the network's architecture, removing entire structural units (e.g., channels or filters) to reduce computational complexity and memory usage, making the model hardware-friendly.

Advantages

- Dependency-Aware Pruning: By leveraging the DepGraph, the GroupNorm Pruner handles complex inter-layer dependencies automatically, ensuring structural integrity across diverse architectures (e.g., CNNs, Transformers, RNNs).

- Group-Level Consistency: The group-level L2 norm ensures that all parameters within a pruning group are consistently evaluated, reducing the risk of performance degradation due to mismatched importance scores.
- Generalizability: The pruner is architecture-agnostic, applicable to a wide range of models (e.g., ResNet, Vision Transformer, LSTM, GAT), unlike architecture-specific pruners that rely on manual grouping schemes.
- Hardware Efficiency: By focusing on structured pruning, it produces models optimized for inference on standard hardware, reducing both memory and computational costs.
- Simplicity with Effectiveness: Despite using a simple L2 norm-based criterion, the pruner achieves competitive performance due to the DepGraph's comprehensive dependency modeling.

Limitations

- Dependence on DepGraph: The pruner's effectiveness relies heavily on the accuracy of the DepGraph in modeling dependencies. Errors in dependency identification could lead to structural issues or suboptimal pruning.
- L2 Norm Limitation: The group-level L2 norm may not capture functional importance (e.g., weights critical to specific tasks), as it focuses solely on magnitude, potentially removing critical parameters.
- Training Dependency: The pruner requires sparsity-inducing training to align

parameter importance within groups. Poor training configurations (e.g., insufficient regularization or epochs) may lead to inaccurate importance scores.

- Computational Overhead: While more efficient than Hessian-based methods, computing group-level L2 norms and managing dependencies can be more complex than single-layer pruners like BN Scale Pruner.

Use in torch-pruningIn the torch-pruning framework, the GroupNorm Pruner is implemented as a high-level pruner that integrates with the DepGraph to facilitate automated structural pruning. Key aspects of its use include:

- Dependency Graph Integration: The pruner relies on the DepGraph to automatically identify and group coupled parameters across layers, ensuring that pruning a channel in one layer (e.g., a convolutional layer) propagates correctly to dependent layers (e.g., BatchNorm or subsequent convolutions).
- Pruning Workflow:

  ① The DepGraph scans the network to identify all prunable groups.

  ② The GroupNorm Pruner computes the L2 norm for each group’s weights.

  ③ Groups with the lowest L2 norms are removed based on a user-specified pruning ratio (e.g., 50% of channels).

  ④ The pruned model is fine-tuned to recover accuracy.

C. Growing Reg Pruner

Concept

The Growing Regularization Pruner (GReg) is a neural network pruning method introduced in the paper Neural Pruning via Growing Regularization, designed to reduce model complexity by removing redundant parameters while maintaining performance. It employs a dynamic L2 regularization strategy where the penalty strength increases gradually during training, enabling effective pruning for both structured (e.g., channels, filters) and unstructured (e.g., individual weights) pruning. The GReg Pruner consists of two variants, GReg-1 and GReg-2, which address the pruning schedule and weight importance scoring, respectively, using a growing regularization approach.

- GReg-1: Focuses on optimizing the pruning schedule by gradually increasing regularization to drive unimportant weights toward zero, allowing the model to adapt before pruning.
- GReg-2: Enhances weight importance scoring by implicitly leveraging Hessian (second-order curvature) information through the growing regularization process, without directly computing the Hessian.
- Target: The pruner can target individual weights (unstructured pruning) or structural units like channels or filters (structured pruning), depending on the desired granularity.

Theoretical Basis

The GReg Pruner is built on the principle of applying a gradually increasing L2 regularization penalty to the loss function during training, which drives unimportant weights or groups closer to zero, facilitating their removal. This approach addresses two key challenges in pruning:

① Pruning Schedule (GReg-1): Traditional pruning methods often use one-shot pruning (removing parameters in a single step), which can degrade performance. GReg-1 introduces a dynamic schedule where the L2 regularization strength (λ) grows over time, allowing the model to redistribute its expressive power to remaining weights before pruning. After regularization, weights with the smallest magnitudes are removed.

② Weight Importance Scoring (GReg-2): GReg-2 improves the identification of unimportant weights by exploiting the local curvature of the loss function (Hessian) implicitly. The growing regularization causes weights in flatter regions (less critical) to approach zero faster than those in steeper regions (more critical), creating a magnitude gap that serves as an effective importance criterion.

- Pruning Process:
  - During training, an L2 regularization term is added to the loss function, with the penalty factor λ increasing according to a predefined schedule (e.g., linear or exponential growth).
  - For GReg-1, this process ensures that unimportant weights are gradually diminished, allowing the model to adapt before pruning.
  - For GReg-2, the regularization process separates weights based on their Hessian properties, enabling magnitude-based pruning that indirectly accounts for second-order information.
  - After training, parameters (weights or groups) with the smallest magnitudes are pruned, and the model is fine-tuned to recover performance.

- Structured Pruning: For structured pruning, the pruner groups weights (e.g., into channels or filters) and applies regularization to the group's aggregated weights, ensuring hardware-friendly model compression.

Advantages

- Optimized Pruning Schedule: GReg-1's gradual regularization allows the model to adapt during training, reducing performance loss compared to one-shot pruning, especially at high pruning ratios.
- Implicit Hessian Exploitation: GReg-2 incorporates second-order (Hessian) information without the computational cost of explicit Hessian calculations, improving the accuracy of importance scoring.
- Versatility: The pruner supports both unstructured (weight-level) and structured (channel/filter-level) pruning, making it adaptable to various network architectures and use cases.
- Scalability: The method is computationally efficient, avoiding expensive Hessian computations, and scales well to large networks and datasets like ImageNet.
- Improved Accuracy: By combining dynamic regularization with magnitude-based pruning, GReg achieves better accuracy retention than traditional magnitude-based methods.

Limitations

- Regularization Schedule Sensitivity: The effectiveness of GReg depends on the design of the growing regularization schedule (e.g., how λ increases). An poorly tuned schedule may lead to suboptimal pruning or performance degradation.

- Magnitude-Based Final Pruning: Despite leveraging Hessian information implicitly, GReg ultimately relies on weight magnitudes for pruning, which may overlook functionally important parameters with small magnitudes.
- Training Dependency: The pruner's performance relies on proper training dynamics (e.g., initialization, optimization). Inadequately trained models may produce unreliable importance scores.
- Complexity in Implementation: Designing and tuning the growing regularization schedule adds complexity compared to simpler pruners like BN Scale Pruner, which use fixed metrics.

Use in torch-pruning While the paper does not explicitly state that GReg is implemented in torch-pruning, the library's framework is flexible enough to support custom pruners like GReg. In torch-pruning, the GReg Pruner would integrate with the library's dependency-aware pruning pipeline:

- Implementation: The pruner would use a custom importance criterion based on growing L2 regularization, applied during training. The DepGraph module would ensure that pruning (e.g., channel or filter removal) is propagated correctly across layers to maintain structural integrity.
- Process: The model is trained with a growing L2 penalty, followed by pruning based on the resulting weight magnitudes. For structured pruning, the pruner groups weights (e.g., by channel or filter) and applies regularization to the group level.
- Customization: Users can specify the regularization schedule (e.g., linear or

exponential growth of λ) and the pruning ratio (e.g., percentage of channels to remove).