# Diffusion Denoised Pruning: Iterative Full Mask Reconstruction for Sparse, Layer-Balanced Subnetworks

Paul Mello
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, Nevada 89557
Email: pmello@unr.edu

***Abstract*—As neural network architectures have exploded in parameterization, compression techniques have become a central topic in deep learning. Most pruning approaches either rely on expensive pruning techniques with heuristically designed operations or require training networks to convergence before pruning, effectively wasting compute. SNIP, a model agnostic framework, addresses this by scoring each weight by the magnitude of its gradient effect on the loss in a single shot approach. We build on SNIP and propose Diffusion Denoised Pruning (DDP), which retains SNIP's connection sensitivity scoring while extending it into an iterative mask denoising framework inspired by denoising diffusion probabilistic models. At each step, DDP computes importance scores on the current masked subnetwork, then injects noise into the importance scores and rebuilds the mask from scratch. This allows for weight revival over pruning iterations and improves the layer-wise sparsity distribution. We find that this evolving mask approach often matches SOTA techniques like SNIP and SNIP-it while significantly reducing FLOPs in heterogeneous model architectures. On AlexNet with CIFAR-10 at $90\%$ weight sparsity, DDP delivers a $77.98\%$ FLOPs reduction against SNIP's $64.89\%$ while retraining to within $1.4$ percentage points in test accuracy. At $99\%$ sparsity, where single shot SNIP catastrophically collapses to chance level accuracy, DDP retains $75$-$88\%$ test accuracy across tested model architectures. We illustrate the strengths of DDP and highlight a previously under utilized component of pruning that may offer potential points of improvements for future work in structured pruning.**

## I. Introduction

At its core, pruning neural networks is a combinatorial optimization problem. Given a dense model with $d$ parameters and a target sparsity $s$, the goal is to identify a binary mask $m \in 0, 1^d$ where the masked subnetwork retains as much accuracy of the original model as possible. Methods which can prune to extreme sparsity find significant reductions in model size, memory footprint, inference FLOPs, and enables strong models on edge devices. The primary question becomes, of all the potential candidate subnetworks, how do we find a strong sparse subnetwork efficiently? In this work, we show that an iterative mask rebuilding procedure, done at initialization, yields structurally more balanced sparse subnetworks than single shot pruning. Fundamentally reducing total training FLOPs at equivalent weight sparsity while preserving accuracy.

Early methods in model compression treated pruning as a post training procedure where pruning was applied after convergence to low magnitude weights. This approach had strong results and could be repeated indefinitely until the target sparsity was reached. However, this wastes significant computational resources to train weights in a dense model that will ultimately be pruned. Moreover, given the heterogeneous architectures and complexity of these large models, this also requires heuristic pruning strategies. As a result pruning post training is bottle-necked by the upstream training process.

Recently, methods have emerged which leverage novel observations to estimate each weights importance with regard to loss at initialization. This leverages a new concept, where the gradient is immediately informative before any training has occurred. This insight was formalized in SNIP [1] where it was used to immediately predict connection sensitivity by decoupling the weight importance from its sensitivity to loss. This saliency criterion depends only on the data to inform its pruning. By utilizing minibatch data, SNIP builds a loss landscape which can inform the pruning in a single shot and can be done efficiently by using the first derivative. Despite this, SNIP has limitations in that it forces all pruning decisions to be made simultaneously on the dense network, potentially missing better subnetworks. SNIP-it [2], an extension to SNIP, addresses this by iterating the process over smaller steps, allowing stronger pruning and enabling SNIP to succeed in extremely sparse parameter regimes. Despite this, SNIP-it has its own limitations as it builds its mask monotonically so that every pruned weight is permanently removed regardless of the evolving subnetwork topology.

Inspired by the denoising process of denoising diffusion probabilistic models [3], we propose DDP. DDP tackles mask monotonicity by rebuilding the mask from scratch every $T$ timesteps rather than accumulating a mask over time. At each step, the original weight values are restored and importance scores are fully recomputed on the current masked subnetwork. A weight which was previously pruned at timestep $t-1$ can receive a stronger gradient signal at step $t$ if the network has become more dependent on it as a result of the changed network topology. Noise is injected into the importance scores

according to a decaying scheduler and enables stochastic revival of any weight. Together, this iterative masking redistributes sparsity more evenly across the depth of the network than single shot or monotonic iterative baselines, which yields a substantially lower FLOP model at matched weight sparsity and preserves competitive accuracies. This is especially useful in sparsity regimes like $99\%$ as single shot methods like SNIP are known to collapse through layer elimination.

### A. Key Contributions

- We propose DDP, an iterative pruning method that discovers sparse subnetworks at initialization by recomputing a binary mask from scratch every timestep. At each step, importance scores are perturbed with decaying Gaussian noise, allowing the method to stochastically explore the mask space before deterministic convergence.
- We show that the iterative mask rebuild procedure at the core of DDP produces a structurally more balanced layer-wise sparsity than prior pruning at initialization methods, yielding up to a $12.3\%$ improvement in FLOPs reduction at matched weight sparsity. In the $99\%$ sparsity regime, DDP matches the accuracy of iterative SNIP-it on convolutional networks where single shot SNIP collapses to chance.
- We evaluate DDP across seven models and dataset combinations spanning convolutional, fully connected, and recurrent architectures at sparsity levels ranging from $0\%$ to $99\%$.

## II. Background

### A. The Lottery Ticket Hypothesis

The lottery ticket hypothesis [4] has become a central paper in the AI domain. Here, the authors demonstrated that dense networks contain sparse subnetworks, called "winning tickets", that can be identified at initialization and, that when trained separately from their original dense networks, can match or exceed the accuracy of the full network. This fundamentally reframed pruning as a structural discovery problem to be done at initialization rather than a after converging. The authors identify winning tickets at around 10-20$\%$ sparsity through an iterative magnitude pruning method.

While the procedure proposed in the lottery ticket hypothesis is exceptionally expensive, often requiring many full training runs, it established a critical insight that the mask structure, paired with the initial weights, matters more than the final trained weight values. This fundamentally shifted the focus of literature from post-training pruning methods to pre-training pruning methods to avoid wasting training compute on weights that will be pruned.

### B. SNIP: Single shot Network Pruning based on Connection Sensitivity

SNIP [1] utilizes the insights revealed in the lottery ticket hypothesis to achieve its strong results. SNIP leverages the gradient at initialization to identify important connections using a minibatch. The authors decouple the existence of a weight from its perceived value through the Hadamard product, defining the importance of each weight $\theta_j$ as the absolute value of the product of the weight and its gradient with respect to the loss:

$$c_j = \left| \frac{\partial \mathcal{L}}{\partial \theta_j} \cdot \theta_j \right| \tag{1}$$

This first order criterion relaxes the binary connection importance from its sensitivity to enable efficient calculations of a weights importance with respect to loss. Rather than checking each weight individually, which would be expensive, the first order derivative reduces this significantly. Weights with high sensitivity to loss are retained and those with low sensitivity to loss, and below a sparsity threshold, are pruned. The approach is solely reliant on data, it is architecture agnostic, and requires no pretraining, no additional hyperparameters, and no augmented objective. This makes it a particularly attractive approach to utilize since it is inherently flexible. The core benefits of SNIP based approaches fundamentally rely on its effectiveness with minimal cost overhead and extensive flexibility.

Despite these impressive improvements, some issues exist including the single shot nature of SNIP which can miss complex topologically dependent subnetworks. At high sparsity levels this can cause irreversible damage to the network. The single shot nature also reveals an additional issue, that once a weight is pruned it can not be brought back. This can lead to layer collapse in highly sparse environments where global top-$k$ selection removes all weights from an individual layer. In the following section III, we detail related works which address these issues and demonstrate the differences between those solutions and DDP.

## III. Related Works

### A. Pruning via Iterative Ranking of Sensitivity Statistics

SNIP-it [2] is a direct response to SNIP's single shot limitation. The core mechanisms of SNIP remain, but SNIP-it applies the sensitivity criterion iteratively to gradually augment the mask for the subnetwork. At each iteration, the sensitivity scores are recomputed on the current partially pruned network. This allows later importance estimates to understand the structural differences in the network topology that were introduced by earlier pruning iterations and often produce more reliable decisions as the network reaches the desired sparsity level. The authors state that this process also provides improved robustness to overfitting and adversarial perturbations compared to single shot methods.

### B. Progressive Skeletonization: Trimming more fat from a network at initialization

In a subsequent work, titled "Progressive Skeletonization: Trimming more fat from a network at initialization" [5] the authors propose a method they call "FORCE". They extend the concepts from SNIP-it to address the iterative monotonicity by recomputing the full mask to enable weight revival over $T$ timesteps. FORCE follows a very similar approach to

TABLE I
COMPARISON OF OUR PROPOSED METHOD AGAINST OTHER PRUNING AT INITIALIZATION METHODS BY KEY PROPERTIES. ALL METHODS OPERATE BEFORE TRAINING, REQUIRE NO PRETRAINING, IMPOSE NO ARCHITECTURE CONSTRAINTS, AND LEAVE THE TRAINING OBJECTIVE UNMODIFIED. HERE, $\sigma$ REPRESENTS THE NOISE SCHEDULER.

| Feature | DDP | SNIP | SNIP-it | FORCE |
|---|---|---|---|---|
| Iterative Refinement | ✓ | ✗ | ✓ | ✓ |
| Weight Revival | ✓ | ✗ | ✗ | ✓ |
| Full Mask Reconstruction | ✓ | ✗ | ✗ | ✓ |
| Stochastic Mask Exploration | ✓ | ✗ | ✗ | ✗ |
| Additional Hyperparameters | $\sigma$, $T$, $s$ | $s$ | $T$, $s$ | $T$, $s$ |

DDP with a few distinct differences. The key differences between DDP and FORCE is that DDP injects Gaussian noise into the importance scores and utilizes a noise scheduler while FORCE performs a deterministic recomputation at each step. Fundamentally, while FORCE adds mask rebuilding and enables weight revival, we add stochasticity to our Importance scores. FORCE is notably susceptible to converging to the same local optimum across iterations due to its determinism. This means FORCE tends to find the same sparse subnetwork topologies.

## IV. DIFFUSION DENOISED PRUNING

DDP reframes the combinatorial optimization problem of sparse network discovery as a progressive denoising process over $T$ timesteps. The dense randomly initialized network is treated as a noisy initial state whose redundant connections obscure the quality sparse subnetwork within. Pruning is akin to progressively denoising these noisy initial structures towards a cleaner sparse subnetwork. Unlike DDPM, where a reverse process learns to denoise, DDP replaces the learned denoiser with a modification to SNIP's importance criterion. Here we inject noise to the importance scores to enable exploration of the mask space at each time step. DDP maintains a binary mask $m^{(t)} \in 0, 1^d$ that evolves as the network topology and loss landscape changes, similar to SNIP-it and FORCE, toward the target sparsity $s$ according to a monotonically increasing sparsity schedule defined by $\sigma \colon \{0, \ldots, T\} \rightarrow [0, s]$ where $\sigma(0) = 0$ and $\sigma(T) = s$ over $T$ timesteps.

The original weights $\theta_{orig}$ are frozen at their random initialization values for the entire DDP process. This decouples the mask search problem from weight adaptation. If weights changed alongside the mask, gradient signals would confuse changes in importance due to topology with changes due to weight values, making scores incomparable as we iterate across timesteps. By freezing the initial weights $\theta_{orig}$, the only quantity that is evolving is the mask topology while importance scores remain linked to weight magnitudes through DDP. This evolving mask topology follows the intuition presented in the lottery ticket hypothesis that the quality of the mask is more critical than the trained weight values.

Table I compares DDP to prior pruning at initialization methods. Since all methods are derived from SNIP, each method requires no pretraining, no augmented training objectives, and no architecture dependent constraints.

TABLE II
SPARSITY SCHEDULE FAMILIES

| Schedule | Formula | Shape |
|---|---|---|
| Linear | $\sigma(t) = s \cdot (t/T)$ | Constant rate |
| Cosine | $\sigma(t) = s \cdot (1 - \cos(\pi t/T))/2$ | S-shaped |
| Exponential | $\sigma(t) = 1 - (1 - s)^{t/T}$ | Front-loaded |

### A. Gradient Accumulation

At each timestep $t$, the original weights $\theta_{\text{orig}}$ are first restored so that all weights, including those previously masked, are considered for pruning. Gradient accumulation then occurs on the current iteration of the masked subnetwork $\theta_{orig} \odot m^{(t-1)}$ over $B$ minibatches given by the following equation where $j$ represents the weight:

$$G_j^{(t)} \leftarrow \sum_{b=1}^{B} \left| \frac{\partial \mathcal{L}\big(\theta_{orig} \odot m^{(t-1)};\, B_b\big)}{\partial \theta_j} \right| \tag{2}$$

This builds on the SNIP connection sensitivity criterion by computing gradients through the current sparse topology while scoring all weights using their original magnitudes. The gradient reflects the loss landscape of the subnetwork at timestep $t-1$ rather than the dense model. By accumulating these gradients over $B$ minibatches, we are effectively reducing variance in the sensitivity estimation.

### B. Importance Scores

Individual weight importance scores $I_j^{(t)}$ combine the gradient sensitivity with the original weight magnitude.

$$I_j^{(t)} \leftarrow \left|\theta_j^{orig}\right| \cdot G_j^{(t)} \tag{3}$$

Here, we utilize SNIP's first order approximation to loss sensitivity caused by removing a weight $j$ from the current sparse subnetwork. We use the complete dense weight list rather than the current masked subnetwork. In other words, weights which were previously removed by the mask at timestep $t-1$ keeps its original weight magnitude $|\theta_j^{\text{orig}}|$ and can be revived by the gradient sensitivity $G_j^{(t)}$ if the network topology has increased its reliance on any given connection.

### C. Sparsity Schedules

Scheduling additive noise is a crucial component to the models success. Here, the shape of $\sigma(t)$ controls the strength of pruning at each timestep. This effectively governs the decay rate of $\eta_t$ and balances the exploration and exploitation tradeoffs of DDP. As network topology gradually shrinks through the sparsity constraints, the space of quality subnetworks also shrinks. As a result, schedulers which explore earlier and exploit later tend to perform better. Simple schedules include those outlined in table II.

### D. Noise Level

The noise level at any given step scales inversely with the fraction of target sparsity already reached:

$$\eta_t \leftarrow \max\left(0,\ 1 - \frac{\sigma(t)}{s}\right) \tag{4}$$

When $\sigma(t)$ is small, relative to the target sparsity $s$, noise dominates and encourages exploration over early steps. As $\sigma(t) \to s$, the noise vanishes and scoring becomes deterministic.

### E. Score Perturbation

Importance scores are injected with noise before ranking weights and introduces stochasticity. This mask rebuilding and reranking is determined by this importance score equation:

$$\tilde{I}_j^{(t)} \leftarrow \begin{cases} \log I_j^{(t)} + \eta_t\, z_j, \quad z_j \sim \mathcal{N}(0,1) & \eta_t > 0 \\ I_j^{(t)} & \eta_t = 0 \end{cases} \tag{5}$$

In the early iterations, when $\sigma(t) \ll s$, the noise level $\eta_t$ is close to 1 and substantial log space perturbation is applied via $z_j \sim \mathcal{N}(0,1)$. Importance scores are transformed into log space before perturbation because raw scores can span several orders of magnitude. Additive noise in the original space would cause the same weights to return consistently, whereas using the log perturbation acts proportionally across the full distribution. This allows weights between low and high to survive pruning stochastically and prevents the network topology from converging prematurely.

In later timesteps, as $\sigma(t) \to s$ and $\eta_t \to 0$, the ranking converges to a deterministic importance ordering. In the final step, when $\sigma(T) = s$ and $\eta_T = 0$, scores return to being unperturbed to calculate a deterministic importance ranking $I_j^{(t)}$ defined by equation 3. This ensures there is no stochasticity or divergence in ranking in the final steps.

### F. Mask Update

The mask is recomputed at each timestep from scratch by recalculating the top-$k$ weights determined by the perturbed scores of equation 5. This mask update is given by the following equation:

$$m^{(t)} \leftarrow \text{top-}k\Big(\tilde{I}^{(t)},\ \max(1,\ \lfloor(1-\sigma(t))\, d\rfloor)\Big) \tag{6}$$

During every mask update, to enable revival, we recompute every weight from the global dense network, based on the loss landscape from the sparse subnetwork. We determine the mask sparsity according to the sparsity level imposed by the noise scheduler. The global sparsity imposed by the top-$k$ selection distributes sparsity according to the perturbed importance scores alone, with no explicit layer awareness. As the network topology changes due to sparsity between each step in DDP, the loss landscape changes, which helps to inform the selection of robust weights.

### G. Diffusion Denoised Pruning Algorithm

The full algorithm of DDP is included for completeness here, Algorithm 1.

**Algorithm 1** Diffusion Denoised Pruning

**Require:** model $\theta \in \mathbb{R}^d$; sparsity $s \in (0,1)$; steps $T \in \mathbb{N}$; schedule $\sigma\colon \{0,\dots,T\} \to [0,s]$; mini-batches $B \in \mathbb{N}$
**Ensure:** mask $m \in \{0,1\}^d$

$\theta_{\text{orig}} \leftarrow \theta$; $m^{(0)} \leftarrow 1_d$
**for** $t = 1,\dots,T$ **do**
  $\theta \leftarrow \theta_{\text{orig}}$ ▷ Restore original weights
  $G_j^{(t)} \leftarrow \sum_{b=1}^{B} \left| \frac{\partial\, \mathcal{L}\big(\theta_{orig} \odot m^{(t-1)};\, B_b\big)}{\partial\, \theta_j} \right|$
  $I_j^{(t)} \leftarrow \left|\theta_j^{\text{orig}}\right| \cdot G_j^{(t)}$
  $\eta_t \leftarrow \max\left(0,\ 1 - \frac{\sigma(t)}{s}\right)$
  $\tilde{I}_j^{(t)} \leftarrow \begin{cases} \log I_j^{(t)} + \eta_t\, z_j, \quad z_j \sim \mathcal{N}(0,1) & \eta_t > 0 \\ I_j^{(t)} & \eta_t = 0 \end{cases}$
  $m^{(t)} \leftarrow \text{top-}k\Big(\tilde{I}^{(t)},\ \max(1,\ \lfloor(1-\sigma(t))\, d\rfloor)\Big)$
**end for**
**return** $m^{(T)}$

## V. Experiment Settings

All results are averaged over three fixed random seeds. All pruning methods receive the same random initialization as their starting point, and methods proceed from that initialization. Experiments run on two NVIDIA A30 GPUs for approximately 120 hours of combined wall clock time. We evaluate on three image classification benchmarks: MNIST [6], CIFAR-10 [7], and Tiny ImageNet [8].

Our chosen models follow SNIP's selection [1]. On MNIST we train LeNet-300-100 [9], LeNet-5-Caffe [10], and single-layer GRU-B [11] and LSTM-B [12] recurrent models with hidden size 256. On CIFAR-10 we train AlexNet-B [13], VGG-D [14]. AlexNet-B is also trained on Tiny ImageNet. All models train in bfloat16 mixed precision.

Hyperparameters follow the SNIP paper [1]. MNIST models use SGD [15] with lr $= 0.1$, momentum $= 0.9$, weight decay $\lambda = 5{\times}10^{-4}$, batch size 100, $50,000$ iterations, and a step learning rate schedule that decays by $10\times$ at iteration $25,000$. CIFAR-10 uses the same SGD configuration with batch size 128 and $150,000$ iterations, decaying at iteration $30,000$. Tiny ImageNet uses the same SGD configuration with cosine annealing for $200,000$ iterations.

For a strong comparison against SNIP, SNIP-it, and DDP we set $B = 1$ minibatch for all three methods, and $T = 10$ timesteps for SNIP-it and DDP. We report results at a default weight sparsity of $90\%$ and $99\%$ and sweep $\{50, 70, 80, 85, 90, 93, 95, 97, 99\}\%$. Due to resource constraints, Wide ResNet-22-8 [16] on CIFAR-10 and VGG-D on Tiny ImageNet are evaluated only in terms of FLOPs reduction, which requires only mask computation without full retraining.

TABLE III

MEAN TEST ACCURACY (%) AVERAGED OVER THREE SEEDS. BEST PRUNING METHOD PER COMBINATION ARE DISPLAYED IN BOLD. $s$ DENOTES THE TARGET WEIGHT SPARSITY. ALL STANDARD DEVIATIONS ARE BELOW 2.2% WITH THE VAST MAJORITY BELOW 0.3%.

| Dataset/Architecture | $s$ | Dense | SNIP | SNIP-it | DDP |
|---|---|---|---|---|---|
| MNIST/LeNet-300-100 | 90% | 98.67 | **98.42** | 98.35 | 98.35 |
| MNIST/LeNet-5-Caffe | 90% | 99.40 | 99.26 | 99.30 | **99.35** |
| MNIST/GRU-B | 90% | 99.34 | 98.72 | 98.73 | **98.84** |
| MNIST/LSTM-B | 90% | 99.25 | 98.74 | **98.83** | 98.67 |
| CIFAR-10/AlexNet-B | 90% | 85.76 | **84.76** | 83.83 | 83.41 |
| CIFAR-10/VGG-D | 90% | 92.76 | **92.69** | 92.27 | 92.12 |
| Tiny ImageNet/AlexNet-B | 90% | 44.52 | 25.54 | **44.16** | 23.15 |

TABLE IV

MEAN TEST ACCURACY (%) AVERAGED OVER THREE SEEDS. BEST PRUNING METHOD PER COMBINATION ARE DISPLAYED IN BOLD. $s$ DENOTES THE TARGET WEIGHT SPARSITY. ALL STANDARD DEVIATIONS ARE BELOW 1.7% WITH THE VAST MAJORITY BELOW 0.2%.

| Dataset/Architecture | $s$ | Dense | SNIP | SNIP-it | DDP |
|---|---|---|---|---|---|
| MNIST/LeNet-300-100 | 99% | 98.67 | 94.68 | 95.90 | **95.93** |
| MNIST/LeNet-5-Caffe | 99% | 99.40 | **98.80** | 98.76 | 98.57 |
| MNIST/GRU-B | 99% | 99.34 | 81.09 | **84.55** | 82.13 |
| MNIST/LSTM-B | 99% | 99.25 | 86.65 | **92.14** | 88.35 |
| CIFAR-10/AlexNet-B | 99% | 85.76 | 10.00 | 74.48 | **75.70** |
| CIFAR-10/VGG-D | 99% | 92.76 | 10.00 | **89.01** | 88.97 |
| Tiny ImageNet/AlexNet-B | 99% | 44.52 | 0.50 | **18.05** | 4.46 |

## VI. RESULTS

### A. Test Accuracy Experiments

For our test accuracy experiments, we test DDP against SNIP and SNIP-it at 90% and 99% sparsity. These tests are conducted by initializing a dense network with random weights, applying the each pruning technique, then train the resulting sparse subnetwork and report the results. We find that, on average, DDP competes with SNIP-it and SNIP in test accuracy with degradation in more complex tasks.

*1) Test Accuracies in Moderately Sparse Regimes:* Table III reports the test accuracy results after training each masked subnetwork to a target sparsity of 90%. In four out of seven instances, the average gap between methods is at most 0.16%, a marginal difference. On Tiny ImageNet trained with AlexNet-B, we find that SNIP-it overwhelming outperforms DDP maintaining baseline accuracy performance. This is likely a product of the weight revival mechanism that will be covered in section VI-D. Despite this, DDP is able to maintain strong accuracy in moderate to extremely sparse regimes.

*2) Test Accuracies in Extremely Sparse Regimes:* In extremely sparse regimes, we find that iterative approaches like SNIP-it and DDP improve significantly over SNIP. In some instance, SNIP result in layer collapse, where all neurons from a single layer are zeroed out preventing backpropogation and learning. In the future section VI-C, we cover this layer collapse and illustrate that DDP tends to have better layer distribution. Table IV demonstrates these claims by SNIP's failure to reach above random accuracy on complex tasks at extreme sparsity. Moreover, this table illustrates that when DDP improves over SNIP-it, it is by a marginal improvement, while when SNIP-it wins it is a much larger gap. This particularly occurs in our recurrent networks and on Tiny ImageNet, both of which are considered complex models and tasks respectively.

*3) Test Accuracy Sparsity Sweep:* Here, we include a sparsity sweep for CIFAR-10 on VGG-D to illustrate how each method performs in a complex environment at varying levels of sparsity. At every sparsity level, DDP is able to match SNIP and SNIP-it, except in extremely sparse regimes where SNIP suffers from layer collapse and DDP briefly dips below SNIP-it before recovering at 99% sparsity.

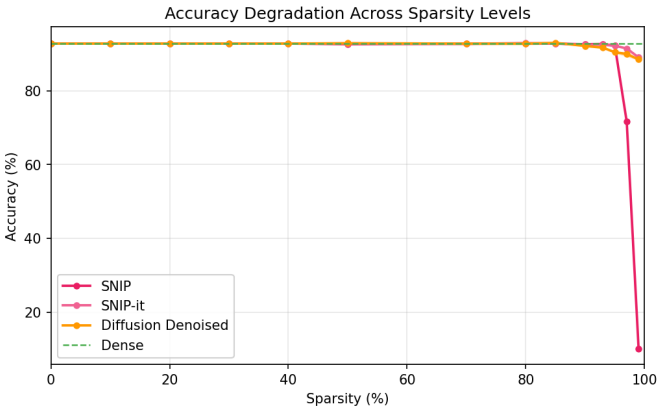


Fig. 1. Test accuracy across the full sparsity sweep on CIFAR-10 using VGG-D. SNIP collapses to 10% at 99% weight sparsity while SNIP-it and DDP track to within 0.04% of each other.

### B. FLOP Reduction Across Methods

One unexpected, but interesting finding from our experiments is that DDP achieves meaningful inference FLOPs reduction than both SNIP and SNIP-it at matched weight sparsity. Tables V and VI illustrate these findings. The advantages tend to be architecturally dependent, but on more complex datasets and large models, DDP consistently outperforms SNIP by an average of 8.9% and SNIP-it by an average of 3.2% at 90% sparsity. At 99% sparsity these numbers reduce to 5.5 and 1.7 respectively. Additional experiments omitted for brevity illustrate that these FLOP advantages extend to moderately sparse regimes between $30\% - 90\%$ as well.

It is very likely that this is a product of better layer-wise sparsity distribution. Convolutional layers, for example, contribute FLOP's proportionally to the kernel size and spatial dimensions. SNIP is more likely to prune fully connected layers because more parameters are fully connected rather than convolutional. This becomes obvious in reference to Figure 2 where the second fully connected layer, FC2, is pruned to 99.8% sparsity, while leaving the convolutional layers at $63\% - 75\%$. Since convolutional layers are more expensive, SNIP and SNIP-it's lack of layer distribution fundamentally requires more FLOPs. DDP's iterative mask reconstruction with weight revival allows it to explore alternative layer distributions and aggressively prunes convolutions layers than SNIP or SNIP-it, with sparsity reaching $80\% - 86\%$ on Conv2

TABLE V

MEAN FLOPS REDUCTION (%) AVERAGED OVER THREE SEEDS. BEST METHOD PER COMBINATION IN BOLD. $s$ DENOTES THE TARGET WEIGHT SPARSITY AND MFLOPS THE UNPRUNED INFERENCE BUDGET IN MILLIONS OF FLOPS. ALL STANDARD DEVIATIONS ON THE REDUCTION COLUMNS ARE BELOW 0.4 PERCENTAGE POINTS.

| Dataset/Architecture | $s$ | MFLOPs | SNIP | SNIP-it | DDP |
|---|---|---|---|---|---|
| MNIST/LeNet-300-100 | 90% | 0.5 | 90.00 | 90.00 | 90.00 |
| MNIST/LeNet-5-Caffe | 90% | 4.6 | 61.28 | **68.01** | 65.32 |
| MNIST/GRU-B | 90% | 22.0 | 91.47 | 91.38 | **91.48** |
| MNIST/LSTM-B | 90% | 29.4 | **91.21** | 91.10 | 91.20 |
| CIFAR-10/AlexNet-B | 90% | 146.7 | 65.02 | 70.28 | **77.27** |
| CIFAR-10/VGG-D | 90% | 627.5 | 62.35 | 67.35 | **71.07** |
| CIFAR-10/WRN-22-8 | 90% | 4908.2 | 73.53 | 78.96 | **80.00** |
| Tiny ImageNet/AlexNet-B | 90% | 147.4 | 64.29 | 71.45 | **76.47** |
| Tiny ImageNet/VGG-D | 90% | 627.6 | 61.95 | 66.87 | **71.74** |

TABLE VI

MEAN FLOPS REDUCTION (%) AVERAGED OVER THREE SEEDS. BEST METHOD PER COMBINATION IN BOLD. $s$ DENOTES THE TARGET WEIGHT SPARSITY AND MFLOPS THE UNPRUNED INFERENCE BUDGET IN MILLIONS OF FLOPS. ALL STANDARD DEVIATIONS ON THE REDUCTION COLUMNS ARE BELOW 0.3 PERCENTAGE POINTS.

| Dataset/Architecture | $s$ | MFLOPs | SNIP | SNIP-it | DDP |
|---|---|---|---|---|---|
| MNIST/LeNet-300-100 | 99% | 0.5 | 99.00 | 99.00 | 99.00 |
| MNIST/LeNet-5-Caffe | 99% | 4.6 | 87.80 | 92.54 | **94.73** |
| MNIST/GRU-B | 99% | 22.0 | **99.58** | 99.36 | 99.36 |
| MNIST/LSTM-B | 99% | 29.4 | **99.53** | 99.36 | 99.31 |
| CIFAR-10/AlexNet-B | 99% | 146.7 | 90.48 | 94.00 | **94.76** |
| CIFAR-10/VGG-D | 99% | 627.5 | 86.28 | 91.64 | **94.14** |
| CIFAR-10/WRN-22-8 | 99% | 4908.2 | 94.64 | 96.53 | **97.19** |
| Tiny ImageNet/AlexNet-B | 99% | 147.4 | 90.25 | 93.62 | **95.10** |
| Tiny ImageNet/VGG-D | 99% | 627.6 | 86.29 | 91.53 | **94.11** |

and Conv3 layers.

Reviewing Tables V and VI, and cross referencing them with Tables III and IV, an interesting pattern emerges where the FLOP improvements directly translate to sacrificing accuracy. At $90\%$ sparsity on CIFAR-10 with AlexNet-B, DDP achieves the highest FLOPs reduction ($77.27\%$ vs $65.02\%$ for SNIP), but the lowest test accuracy ($83.41\%$ vs $84.76\%$). On CIFAR-10 with VGG-D the same is true, DDP leads in FLOPs reduction ($71.07\%$ vs $62.35\%$ for SNIP), but trails in accuracy ($91.12\%$ vs $92.69\%$). In this way, DDP is more structurally efficient, but removes some important representations. This trend reverses in extremely sparse regimes as DDP achieves the highest FLOPs reduction and the highest test accuracy Table IV. This makes it clear that layer distribution matters as sparsity reaches extreme levels and implies that not all parameters are weighted equal.

Overall, on architectures where all layers have similar FLOPs per-parameter ratios, this parameter redistribution provides no advantage as all our tested methods converge to the same FLOPs as demonstrated in our MNIST on LeNet-300-100 results on Table V and VI. On architectures with heterogeneous layer types, it translates to meaningful inference time speedups, reaching up to $12.3\%$ over SNIP at equivalent parameter counts.

### C. Per-Layer Sparsity

Figure 2 illustrates a common trend we find in layer distribution caused by our weight revival and iterative process.

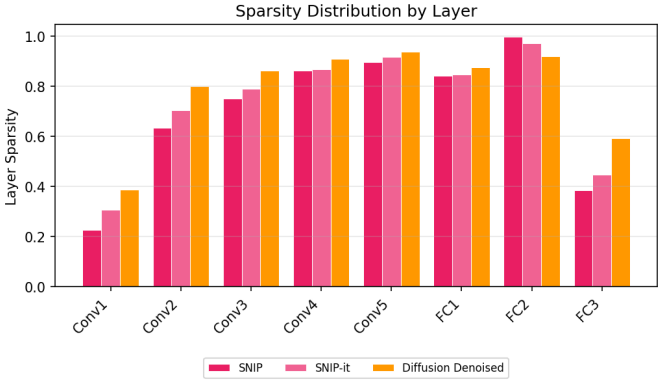


Fig. 2. Per-layer sparsity at $90\%$ global weight sparsity on CIFAR-10 with AlexNet-B. SNIP eliminates $99.8\%$ of FC2 while DDP preserves $8.1\%$ of FC2. DDP focuses its pruning on earlier layers and particularly convolutional layers more aggressively.

Here each method is compared with CIFAR-10 on AlexNet-B at $90\%$ sparsity. SNIP drives the $2048 \times 2048$ fully connected layer to $99.8\%$ sparsity. This preserves roughly $8,400$ of its $4,196,352$ weights and consequently the fully connected layers are pruned less aggressively with layers Conv2 and Conv3 reaching $63.4\%$ and $75.2\%$ respectively.

Our approach produces a meaningful FLOP reduction as referenced in section VI-B, by cutting more from convolutions rather than fully connected layers. This process helps to prevent the elimination of any single layer and explains the extreme sparsity survivability of section VI-A3. Interestingly, we also notice, throughout most architecture and model experiments except for recurrent networks, that DDP tends to prune layers more aggressively earlier in the model and at the final layer over both SNIP and SNIP-it. One potential for such a pattern is that earlier method parameters, and the final layer, contribute less to model performance in heterogeneous architectures.

### D. Ablations

In this section we cover ablation experiments to DDP to explore various components. Here we test the effects of both noise injection to the importance scores and weight revival on how they each effect the final test accuracy. This allows us to determine the importance and value of each component for DDP. We also test different types of image augmentations across different strengths and their effects on test accuracy under each pruning technique. We find that noise severity levels do not change accuracy trends. This effectively measures the robustness of each system to input perturbations across types and severity levels.

*1) Noise Injection and Weight Revival:* In this section we discuss one of the core consequences of DDP, weight revival as a product of noise injection. We ablate at $90\%$ sparsity on two components of DDP's step based scoring mechanism. The Gaussian noise injected into the log space importance scores and the revival of pruned weights before scoring.

TABLE VII

ABLATION OF DDP'S NOISE INJECTION AND WEIGHT REVIVAL COMPONENTS. $s$ DENOTES THE TARGET WEIGHT SPARSITY. MEAN TEST ACCURACY (%) AVERAGED OVER THREE SEEDS. ALL STANDARD DEVIATIONS ARE BELOW 1.3 PERCENTAGE POINTS.

| | | Noise | | Revival | |
|---|---|---|---|---|---|
| Dataset/Architecture | $s$ | On | Off | On | Off |
| MNIST/LeNet-300-100 | 90% | 98.38 | 98.41 | 98.45 | 98.43 |
| MNIST/LeNet-5-Caffe | 90% | 99.36 | 99.34 | 99.33 | 99.31 |
| MNIST/GRU-B | 90% | 98.57 | 98.71 | 98.68 | 98.64 |
| MNIST/LSTM-B | 90% | 98.75 | 98.68 | 98.74 | 98.83 |
| CIFAR-10/AlexNet-B | 90% | 83.68 | 83.57 | 83.48 | 84.03 |
| CIFAR-10/VGG-D | 90% | 90.98 | 91.14 | 90.89 | 92.04 |
| Tiny ImageNet/AlexNet-B | 90% | 22.59 | 22.12 | 22.30 | 40.71 |

Turning our attention to the noise column in Table VII, we find that when setting our noise to zero at every step, $\eta_t = 0$, the final accuracy changes by at most $0.50\%$, with an average change of $0.05\%$ on any combination. Here it is noticeable that the noise injection tends to improve robustness marginally with slightly better accuracies across the board. This points to noise being a marginal improvement technique and that our method does help, especially considering most of our test accuracies exist in a highly accurate $98.5\%$ environment.

Turning our attention now to the revival column in Table VII, we find that on average, utilizing revival in MNIST datasets does not effect the overall test accuracy that much, with a marginal increase of $0.05\%$ on average. However, in more complex domains like CIFAR-10, we find that weight revival can actually have a damaging effect on test accuracy. This is made even more evident in Tiny ImageNet, where weight revival reduces test accuracy by $18.41\%$. This is essentially the gap to SNIP-it's test accuracy that we found in Table III, which directly implies the weight revival has a negative effect on test accuracy. The results on CIFAR-10 and Tiny ImageNet express that revival is beneficial on easy combinations like MNIST and small architectures, but harmful on harder combinations like CIFAR-10 and Tiny ImageNet on larger architectures. Future work may find that longer time horizons, adaptive variance of restoration, or more structured sparsity are candidate directions.

*2) Robustness to Input Perturbations:* Figure 3 illustrates the prevailing trend of SNIP based pruning techniques. Regardless of noise intensity levels, if a pruning method like DDP performs the best at a level 1 noise severity, it also performs the best at level 5 noise severity. Severity differs between augmentation types, but generally severity levels follow this pattern: $1 = 0.02$, $2 = 0.05$, $3 = 0.10$, $4 = 0.15$, and $5 = 0.20$.

Overall, across architectures and methods, we find that SNIP wins 3 (LeNet-300, AlexNet-B, VGG-D), DDP wins 2 (LeNet-5, LSTM-B), and SNIP-it wins 1 (GRU-B). These results can be seen in Table VIII. On saturated MNIST MLPs the drops in test accuracy amount to a $2.15\% - 2.55\%$ and all methods are interchangeable. On MNIST sequence models drop by $15.72\% - 17.78\%$, with DDP winning LSTM but SNIP-it winning GRU. On CIFAR-10 the drops are $24.83\% - 47.09\%$ with SNIP consistently having the smallest drop. Finally, on Tiny ImageNet the results are confounded by DDP's revival induced collapse in accuracy.

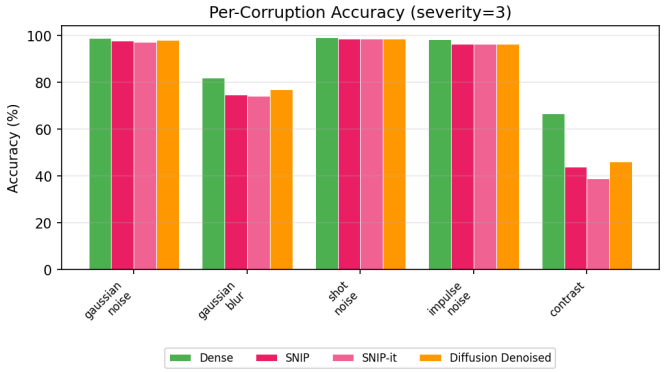


Fig. 3. Corruption severity test accuracy at severity 3 on MNIST/LSTM-B at $90\%$ weight sparsity. DDP retains higher accuracy than both baselines under Gaussian blur and contrast shift two structural perturbations while all three methods track each other on the noise-type corruptions (Gaussian, shot, and impulse noise).

TABLE VIII

DISTRIBUTION SHIFT ROBUSTNESS AT $90\%$ WEIGHT SPARSITY. CLEAN ACCURACY IS REPORTED FROM TABLE III, MEAN CORRUPTED DESCRIBES ACCURACY AVERAGED OVER FIVE CORRUPTION TYPES $\times$ FIVE SEVERITY LEVELS, AND ACCURACY DROP IS DEFINED BY CLEAN $-$ MEAN CORRUPT. BEST DROP PER COMBINATION (LOWEST) IN **BOLD**.

| Dataset/Architecture | Method | Clean | Mean Corrupt | Drop |
|---|---|---|---|---|
| MNIST/LeNet-300-100 | SNIP | 98.42 | 96.06 | **2.36** |
| | SNIP-it | 98.35 | 95.83 | 2.52 |
| | DDP | 98.35 | 95.84 | 2.51 |
| MNIST/LeNet-5-Caffe | SNIP | 99.26 | 97.03 | 2.23 |
| | SNIP-it | 99.30 | 96.75 | 2.55 |
| | DDP | 99.35 | 97.23 | **2.12** |
| MNIST/GRU-B | SNIP | 98.72 | 82.45 | 16.27 |
| | SNIP-it | 98.73 | 83.01 | **15.72** |
| | DDP | 98.84 | 81.81 | 17.03 |
| MNIST/LSTM-B | SNIP | 98.74 | 81.93 | 16.81 |
| | SNIP-it | 98.83 | 81.05 | 17.78 |
| | DDP | 98.67 | 82.82 | **15.85** |
| CIFAR-10/AlexNet-B | SNIP | 84.76 | 59.93 | **24.83** |
| | SNIP-it | 83.83 | 58.69 | 25.14 |
| | DDP | 83.41 | 58.48 | 24.93 |
| CIFAR-10/VGG-D | SNIP | 92.69 | 48.50 | **44.19** |
| | SNIP-it | 92.27 | 45.98 | 46.29 |
| | DDP | 92.12 | 45.03 | 47.09 |
| Tiny ImageNet/AlexNet-B | SNIP | 25.54 | 10.43 | 15.11 |
| | SNIP-it | 44.16 | 19.77 | 24.39 |
| | DDP | 23.15 | 10.97 | **12.18** |

With this in mind our MNIST on LSTM-B produces the best results for DDP, where at $90\%$ DDP is significantly better at avoiding drops in test accuracy based on input perturbations. This can be seen in Table VIII. sparsity the trends remain effectively the same. One important consideration is that a smaller drop does not always mean a better accuracy as is the case on Tiny Imagenet with AlexNet-B where SNIP-it drops twice as much in accuracy from perturbations, but has twice the final accuracy of DDP due to their initial clean starting point.

## VII. FUTURE WORK

Our results suggest several promising directions for future work. Firstly, ablations to the minibatch size, the timestep

horizon, and noise schedulers may reveal stochastic mechanisms become more impactful through more data, time, and better schedules. Secondly, our layer-wise sparsity distribution findings indicate that parameter type plays a role in importance that is not tracked in SNIP as each type of architecture: convolutional, recurrent, and fully connected weights differ on their FLOPs cost and representational role. Exploration here may find sparse masks with better accuracies. Thirdly, our FLOPs advantage is an emergent consequence of our iterative mask reconstruction with weight revival rather than an explicit objective to optimize. A FLOPs aware adaptive importance score my be a strong optimization target for research. Finally, DDP utilizes SNIPs first order importance scoring, it is possible that second order derivatives may offer more reliable importance rankings, particularly in complex tasks. Each of these approaches offer an interesting research direction for future work to explore.

## VIII. Conclusion

In this work, we introduced Diffusion Denoised Pruning (DDP), an iterative pruning method that works at initialization to reconstruct the full weight mask at each time step. Across our experiments, DDP matches SNIP and SNIP-it in test accuracy while producing substantially more balanced layer-wise sparsity on heterogeneous architectures. This structural balance translates directly to better throughput on inference FLOPs, fundamentally reducing necessary computations and naturally avoids layer collapse unlike single shot approaches. Our ablations isolate the iterative mask reconstruction itself as the source of these gains as noise injection changes final accuracy minimally and weight revival is essentially neutral on simple tasks, but harmful on more complex tasks. Taken together, these results establish layer-wise balancing as a productive direction for reducing inference costs with minimal accuracy losses. Future work may find extending these notions to be a fruitful endeavour for model compression techniques.

## References


[1] N. Lee, T. Ajanthan, and P. H. S. Torr, "Snip: Single-shot network pruning based on connection sensitivity," 2019. [Online]. Available: https://arxiv.org/abs/1810.02340

[2] S. Verdenius, M. Stol, and P. Forré, "Pruning via iterative ranking of sensitivity statistics," 2020. [Online]. Available: https://arxiv.org/abs/2006.00896

[3] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[4] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1803.03635

[5] A. Peste, E. Iofinova, A. Vladu, and D. Alistarh, "Ac/dc: Alternating compressed/decompressed training of deep neural networks," 2021. [Online]. Available: https://arxiv.org/abs/2106.12379

[6] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[7] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," 2009. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[8] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[10] ——, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. [Online]. Available: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

[11] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014. [Online]. Available: https://arxiv.org/abs/1406.1078

[12] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2015. [Online]. Available: https://arxiv.org/abs/1409.2329

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[15] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[16] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.