A Unified Framework for Knowledge Transfer in Bidirectional Model Scaling

Supplementary Material

A. The Discrete Wavelet Transform

This appendix provides a detailed mathematical background for the Discrete Wavelet Transform (DWT) and its inverse (IDWT), building from the 1D case to the 3D case used in our work.

A.1. The 1D Discrete Wavelet Transform

The 1D-DWT is the fundamental building block. It decomposes a discrete signal into two components: a low-frequency approximation and a high-frequency detail. This is achieved through filtering and down-sampling.

A.1.1. Decomposition (Analysis)

Given a discrete signal $x[n]$ of length N, the decomposition involves two main steps:

1. **Filtering:** The signal is passed through two complementary filters: a low-pass filter $g[n]$ (associated with the scaling function) and a high-pass filter $h[n]$ (associated with the wavelet function). This is performed via convolution:

$$y_{\text{low}}[n] = (x * g)[n] = \sum_k x[k] \cdot g[n-k]$$
$$y_{\text{high}}[n] = (x * h)[n] = \sum_k x[k] \cdot h[n-k]$$

2. **Down-sampling:** The output of each filter contains redundant information. To create a compact representation, both filtered signals are down-sampled by a factor of f (denoted by $\downarrow f$), meaning only every other sample is kept.

This process yields the **approximation coefficients** (cA) from the low-pass filter and the **detail coefficients** (cD) from the high-pass filter. Each set of coefficients has a length of approximately $N/2$. The complete formulas are:

$$\begin{aligned} cA[k] &= (x * g)_{\downarrow f}[k] = \sum_n x[n] \cdot g[2k-n] \\ cD[k] &= (x * h)_{\downarrow f}[k] = \sum_n x[n] \cdot h[2k-n] \end{aligned} \tag{5}$$

A.1.2. Reconstruction (Synthesis)

The Inverse DWT (IDWT) perfectly reconstructs the original signal from its approximation and detail coefficients. This process reverses the decomposition steps.

1. **Up-sampling:** First, both the cA and cD coefficient vectors are up-sampled by a factor of f (denoted by $\uparrow f$). This is achieved by inserting a zero between every sample, restoring them to the original signal's length N.
2. **Filtering:** The up-sampled signals are then passed through corresponding synthesis filters, $g'[n]$ (low-pass) and $h'[n]$ (high-pass).

The outputs from these two filters are summed to perfectly reconstruct the original signal, $x[n]$. The complete reconstruction formula for the signal $x'[n]$ is:

$$\begin{aligned} x'[n] &= (cA_{\uparrow f} * g')[n] + (cD_{\uparrow f} * h')[n] \\ &= \sum_k cA[k] \cdot g'[n-2k] + \sum_k cD[k] \cdot h'[n-2k] \end{aligned} \tag{6}$$

A.2. Extension to Multi-Dimensional DWT

The multi-dimensional DWT is implemented by applying the 1D-DWT **separably** along each dimension of the input.

2D-DWT To illustrate the separable approach, consider a 2D input (image) T.

1. First, the 1D-DWT is applied to each **row** of T. This results in two intermediate output: one containing the approximation coefficients of the rows (cA_{rows}) and one containing the detail coefficients (cD_{rows}).
2. Next, the 1D-DWT is applied to each **column** of these two intermediate weights.

Applying the 1D-DWT to the columns of cA_{rows} yields the final approximation coefficients ($cA = T_{LL}$) and the vertical detail coefficients ($cD_{\text{vertical}} = T_{LH}$). Applying it to the columns of cD_{rows} yields the horizontal details ($cD_{\text{horizontal}} = T_{HL}$) and the diagonal details ($cD_{\text{diagonal}} = T_{HH}$). Thus, a 2D input is decomposed into one approximation sub-band (cA) and three detail sub-bands (cD).

3D-DWT The 3D-DWT extends this principle to one more dimension. For a 3D input T:

1. Apply the 1D-DWT along the first dimension (*e.g.*, rows, axis i).
2. Apply the 1D-DWT along the second dimension (*e.g.*, columns, axis j) to the results of the first step.
3. Apply the 1D-DWT along the third dimension (*e.g.*, depth, axis k) to the results of the second step.

This three-stage separable process results in one low-frequency approximation sub-band (T_{LLL}) and seven high-frequency detail sub-bands ($T_{LLH}, T_{LHL}, T_{LHH}, \ldots, T_{HHH}$).

A.3. Connection to Main Paper's Notation

The operator notation used in Section 3.1 of the main paper provides a compact representation of this separable filtering process.

- The operator Φ_d represents the application of the 1D-DWT's low-pass analysis stage (filtering with g and down-sampling by f) along dimension d.
- The operator Ψ_d represents the application of the 1D-DWT's high-pass analysis stage (filtering with h and down-sampling by f) along dimension d.

Therefore, Equation (1) from the main paper,

$$cA = \Phi_k(\Phi_j(\Phi_i(T)))$$

is the formal mathematical notation for sequentially applying the low-pass filter transform along the i, then j, and finally k axes of the input T. Similarly, the various detail coefficients (cD_m) are derived from applying different combinations of Φ_d and Ψ_d operators. The IDWT operation reverses this process stage by stage, as described conceptually in Section A.2.

B. Overview of Wavelet Families

In our ablation study (Section 4.5.1), we evaluated several common wavelet families provided by the PyWavelets library [20], as shown in Table 5. The choice of wavelet family is critical as it determines the properties of the low-pass and high-pass filters used in the DWT and IDWT, such as their length, support, symmetry, and smoothness. These properties, in turn, influence how information is decomposed and reconstructed. Below, we provide a brief overview of the families used in our experiments.

Haar Wavelet. The Haar wavelet is the simplest and first known wavelet. It is a discontinuous, piecewise-constant function that resembles a step function.

- **Properties:** It has the most compact support possible (length 2 filter) and is the only real, orthogonal, and symmetric wavelet (excluding trivial examples). Its primary drawback is its lack of continuity, which can introduce blocky artifacts in signal processing applications.
- **Relevance to Our Work:** Its simplicity and compact support make it extremely computationally efficient. Our results suggest its piecewise-constant nature is surprisingly effective for compressing the knowledge in encoder-based models like BERT (L2S).

Daubechies Family (db). Named after their creator, Ingrid Daubechies, this family is a cornerstone of modern wavelet theory [4]. The "dbN" wavelets (*e.g.*, "db2", "db4") are orthogonal wavelets where "N" refers to the number of vanishing moments, which is half the filter length.

- **Properties:** They are characterized by compact support and increasing smoothness as "N" increases. However, they are asymmetric (except for Haar, which is "db1"), which can sometimes be a disadvantage.

Table 5. Overview of Wavelet Families Used in Ablation Study. The "Visualization" column shows the wavelet function (ψ), which determines how high-frequency details are captured.

Wavelet	Length	Key Characteristics	Visualization
haar	2	Discontinuous, piecewise-constant. Has the most compact support. Highly efficient. Optimal for: BERT (L2S)	
db2	4	Daubechies family. Orthogonal and compact, but asymmetric. Optimal for: GPT (S2L)	
db4	8	Daubechies family. Smoother than db2 due to longer filter and more vanishing moments.	
sym8	16	Symlet family. "Least asymmetric" modification of Daubechies wavelets. Orthogonal with compact support.	
coif3	18	Coiflet family. Nearly symmetric with vanishing moments for both scaling and wavelet functions. Optimal for: GPT (L2S)	
bior3.3	8 (Decomp.)	Biorthogonal family. Symmetric (linear phase) by relaxing orthogonality. Uses separate analysis/synthesis filters.	
bior4.4	10 (Decomp.)	Biorthogonal family. Smoother and longer filter than bior3.3.	
bior6.8	18 (Decomp.)	Biorthogonal family. Highly smooth with a long filter, ideal for interpolation. Optimal for: BERT (S2L)	
rbio3.3	8 (Decomp.)	Reverse Biorthogonal. Swaps the decomposition and reconstruction filters of bior3.3.	
dmey	(Approximated)	Discrete Meyer. FIR approximation of an infinitely smooth, orthogonal wavelet defined in the frequency domain.	

- **Relevance to Our Work:** The 'db' family offers a direct way to trade off between filter length and smoothness. Our experiments show that the compact "db2" is particularly effective for expanding decoder-based models like GPT (S2L).

Symlets (sym) and Coiflets (coif). The Symlet and Coiflet families were designed to be modifications of the Daubechies wavelets to improve symmetry.

- **Properties:** "symN" wavelets are "least asymmetric" for their support length, making them a popular choice when

near-symmetry is desired. "coifN" wavelets are constructed to have both the scaling function and wavelet function possess vanishing moments, which can be useful for compression.

- **Relevance to Our Work:** "sym8" and "coif3" represent wavelets with longer filters and greater smoothness compared to "db2"/"db4". Our results show them to be strong performers in several settings, particularly "coif3" for GPT (L2S).

Biorthogonal Family (bior) and Reverse Biorthogonal (rbio). This family relaxes the orthogonality constraint, instead using two different sets of wavelets for decomposition and reconstruction (analysis and synthesis). This trade-off allows for the construction of wavelets that are both symmetric and have compact support (unlike the "db" family).

- **Properties:** The key advantage is linear phase, which means they do not introduce phase distortion. The notation "biorNr.Nd" indicates the filter lengths for reconstruction ("Nr") and decomposition ("Nd"). The "rbio" family is simply the reverse pairing.
- **Relevance to Our Work:** The flexibility of this family is evident in our results. The highly smooth "bior6.8" proves to be the best for expanding encoder-based models (BERT S2L), suggesting its longer, smoother filters are ideal for interpolating complex, bidirectional knowledge.

Discrete Meyer Wavelet (dmey). The Meyer wavelet is an infinitely smooth, orthogonal wavelet defined in the frequency domain. "dmey" is its FIR (Finite Impulse Response) approximation.

- **Properties:** It is known for its excellent smoothness properties.
- **Relevance to Our Work:** As a representative of infinitely regular wavelets, "dmey" serves as a point of comparison against wavelets with finite support.

This overview illustrates the rich diversity within wavelet families. Our empirical results suggest a strong correlation between these properties (*e.g.*, smoothness, support length) and their suitability for transferring knowledge across different neural network architectures and scaling directions.

Table 6. The structures of DeiT models used in our experiments.

Config	DeiT-Ti	DeiT-S	DeiT-B	DeiT-L
# layers	3	6	12	24
# hidden	192	384	768	1024
# heads	3	6	12	16
input size	224	224	224	224
patch size	16	16	16	16

Table 7. Hyper-parameters for all baseline on ImageNet-1K.

Training Settings	Configuration
optimizer	AdamW
base learning rate	S: 2.5e-4 | B: 1.25e-4
warmup learning rate	1e-6
weight decay	0.05
optimizer momentum	0.9
batch size	S: 256 | B: 128
training epochs	150
learning rate schedule	cosine decay
warmup epochs	0
color jitter	0.4
auto augment	rand-m9-mstd0.5-inc1
mixup	0.8
cutmix	1.0
label smoothing	0.1
drop path	0.1

Table 8. Characteristics of downstream datasets.

Dataset	Classes	Total	Training	Testing
Oxford Flowers [24]	102	8,189	2,040	6,149
CUB-200-2011 [43]	200	11,788	5,994	5,794
Stanford Cars [10]	196	16,185	8,144	8,041
CIFAR10 [18]	10	60,000	50,000	10,000
CIFAR100 [18]	100	60,000	50,000	10,000
Food101 [1]	101	101,000	75,750	25,250
iNat-2019 [38]	1010	268,243	265,213	3,030

C. Training Details

C.1. Vision Models Training

C.1.1. Experiment Structures.

We show DeiT structures in Table 6.

C.1.2. Details of Scaling Experiments on DeiT

To evaluate the effectiveness of our bidirectional transfer method, BoT, we conducted scaling experiments on the ImageNet-1K dataset using DeiT architectures. The experiments cover both L2S and S2L scenarios.

For the L2S transfer, we initialized a DeiT-S model using the weights from a pre-trained DeiT-B. For the S2L transfer, we initialized a DeiT-B model using a pre-trained DeiT-S. These initialized models, along with all baselines, were then pre-trained for 150 epochs. The key hyper-parameters for this pre-training phase are detailed in Table 7. Note that distinct learning rates and batch sizes were used for the DeiT-S and DeiT-B to ensure stable and optimal training for each architecture. All other settings, including data augmentation strategies like Mixup and Cutmix, remained consistent across experiments to allow for a fair comparison of the initialization methods.

Table 9. Hyper-parameters for neural networks trained on downstream datasets.

Dataset	Batch Size	Epoch	Learning Rate	Drop Last	Warmup Epochs	Droppath Rate	Color Jitter	Auto Augment	Random Rrase	Mixup	Cutmix	Scheduler	Optimizer
Oxford Flowers	512	300	3e-4	False	0	0	0.4	rand-m9-mstd0.5-inc1	0.25	0	0	cosine	AdamW
CUB-200-2011	512	300	3e-4	False	0	0.1	0		0.25	0	0	cosine	AdamW
Stanford Cars	512	300	3e-4	False	0	0.1	0		0.25	0	0	cosine	AdamW
CIFAR10	512	300	5e-4	True	0	0.1	0.4		0.25	0	0	cosine	AdamW
CIFAR100	512	300	5e-4	True	0	0.1	0.4		0.25	0	0	cosine	AdamW
Food101	512	300	5e-4	True	0	0.1	0.4		0.25	0	0	cosine	AdamW
iNat-2019	512	100	5e-4	True	0	0.1	0.4		0.25	0	0	cosine	AdamW

Table 10. The structures of BERT models.

Config	BERT-Ti	BERT-S	BERT-B	BERT-L
# layers	3	6	12	24
# hidden	192	384	768	1024
# heads	3	6	12	16
# vocab	30522	30522	30522	30522
seq. length	128	128	128	128

Table 11. Hyper-parameters for BERT and RoBERTa pre-training.

Training Setting	Configuration
Optimizer	AdamW
Adam ϵ	1e-6
Learning Rate	2e-4
Weight Decay	0.1
Learning Rate Schedule	Linear Decay
Warmup Steps	0
Total Training Steps	400,000
Batch Size	256
Sequence Length	128
Dataset	English Wikipedia

Table 12. The structures of RoBERTa and GPT-2 models used in our experiments. RoBERTa shares a similar architecture to BERT but uses a different vocabulary.

Config	RoBERTa-S	RoBERTa-B	GPT-S	GPT-B
# layers	6	12	6	12
# hidden	384	768	384	768
# heads	6	12	6	12
# vocab	50265	50265	50257	50257
seq. length	128	128	1024	1024

C.1.3. Details of Downstream Datasets

Additional datasets include Oxford Flowers [24], CUB-200-2011 [43], Stanford Cars [10], CIFAR-10, CIFAR-100 [18], Food-101 [1], and iNaturalist-2019 [38]. Table 8 presents the details of seven downstream datasets, which are sorted by the size of datasets. Table 9 presents the basic settings, including batch size, warmup epochs, training epochs and other settings for training the models initialized with all baselines on various datasets.

Table 13. Hyper-parameters for GPT-2 pre-training.

Training Setting	Configuration
Optimizer	AdamW
Adam ϵ	1e-8
Adam β_1, β_2	0.9, 0.95
Learning Rate	1e-3
Weight Decay	0.1
Learning Rate Schedule	Cosine Decay
Warmup Steps	0
Total Training Steps	S: 150000 | B: 50000
Batch Size	512
Sequence Length	1024
Dropout	0.1
Dataset	Wikipedia + Book Corpus

C.2. Bert Training

C.2.1. Experiment Structures.

We show BERT structures in Table 10.

C.2.2. Details of Scaling Experiments on BERT

To validate BoT's training acceleration on BERT [5], we conducted bidirectional pre-training experiments on the English Wikipedia corpus. The experiments covered both L2S and S2L transfers. For the L2S scenario, we initialized a BERT-S model using a pre-trained BERT-B. Conversely, for the S2L scenario, a BERT-B model was initialized from a pre-trained BERT-S. All initialized models, along with their respective baselines, were then pre-trained for 400K steps. The key hyper-parameters for this pre-training phase are detailed in Table 11. A consistent batch size of 256 was used for all runs.

C.2.3. Details of Downstream Task Fine-tuning

To assess the quality of the transferred knowledge, all initialized models were subsequently evaluated on the GLUE [45] and SQuAD [28, 29] benchmarks without any intermediate pre-training.

For the GLUE benchmark, we performed a hyper-parameter search for each task. We set the batch size to 32 and the learning rate to 1e-4 and the optimal number of training epochs from $\{3, 6, 10\}$. We used the Adam

optimizer [17] for all runs and report accuracy for most tasks, with the exception of CoLA, for which we report the Matthews Correlation Coefficient (MCC), and STS-B, for which we report Pearson correlation.

For the SQuAD (v1.1 and v2.0) fine-tuning, we used a fixed setting with a batch size of 12, a learning rate of 3e-5, and training epochs from $\{2, 4, 8\}$. Performance is measured by the standard Exact Match (EM) and F1 scores.

For all tasks, we report the final metric on the development set, averaged over 5 independent runs with different random seeds to ensure robust and reproducible results. An average score is also computed for each benchmark suite.

C.3. RoBerta Training

C.3.1. Experiment Structures.

We show RoBERTa structures in Table 12.

C.3.2. Details of Scaling Experiments on RoBERTa

To validate BoT's training acceleration on RoBERTa [22], we conducted bidirectional pre-training experiments on the English Wikipedia corpus. The experiments covered both L2S and S2L transfers. The key hyper-parameters for this pre-training phase are detailed in Table 11.

C.4. GPT2 Training

C.4.1. Experiment Structures.

We show GPT-2 structures in Table 12.

C.4.2. Details of Scaling Experiments on GPT

To evaluate the performance of our method, BoT, on autoregressive models, we performed both L2S and S2L transfers. For the L2S scenario, we initialized a GPT-Small model using a pre-trained GPT-Base. Conversely, for the S2L scenario, a GPT-Base model was initialized from a pre-trained GPT-Small. The initialized models, along with baselines, were then pre-trained on a dataset combining the English Wikipedia corpus and the Toronto Book Corpus [53]. The total training duration was 200K steps. The specific hyper-parameters used for this pre-training phase are detailed in Table 13.

C.5. LiGO and Mango

In experiments of DeiTs, we train LiGO and Mango operators with AdamW optimizer for 100 steps on ImageNet-1K. The learning rate is 5e-4. Batch size is 256. In experiments of BERT and RoBERTa models, we train LiGO and Mango operators with Adamw optimizer for 100 steps on the concatenation of English Wikipedia. The learning rate is 2e-4. Minibatch is 256. In experiments of the GPT models, we train LiGO and Mango operators with Adamw optimizer for 100 steps on the concatenation of English Wikipedia and Toronto Book Corpus. The learning rate is 1e-5. Minibatch is 512.

D. Additional Results

D.1. Results on RoBERTa

To further assess the generality of our framework, we conducted parallel experiments on RoBERTa [22]. Following the same protocol, we performed bidirectional pre-training on the English Wikipedia corpus, initializing a RoBERTa-S from a RoBERTa-B and vice versa. The initialized models were subsequently evaluated on the GLUE and SQuAD benchmarks to measure the quality of transferred knowledge.

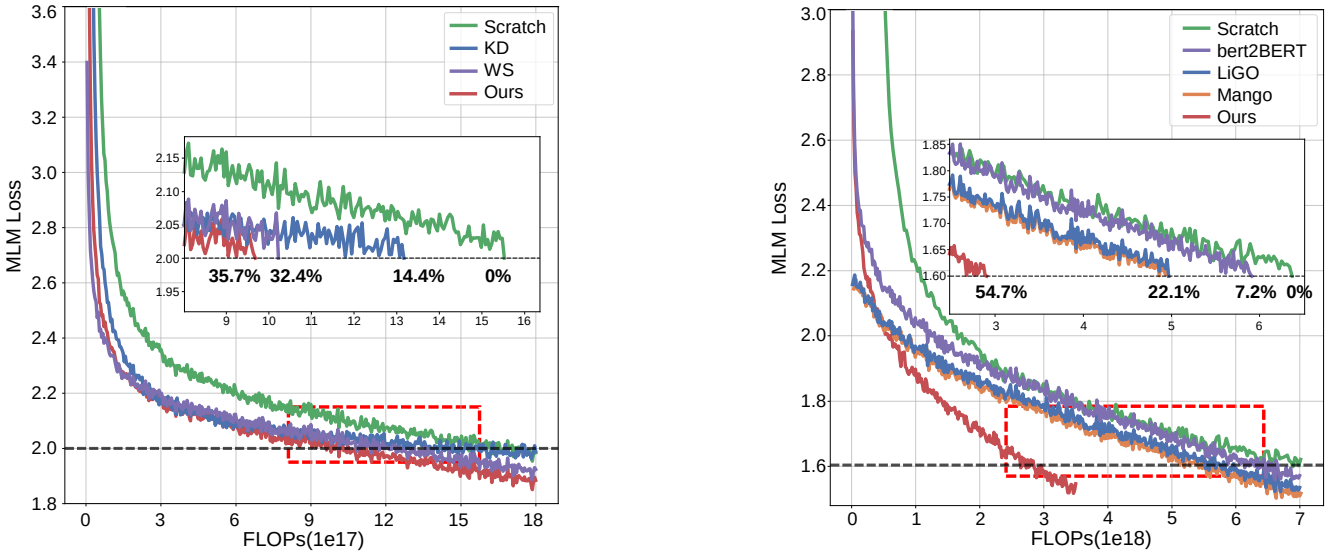
The pre-training convergence curves for RoBERTa, illustrated in Figure 8, affirm BoT's robust, state-of-the-art performance. In the L2S scenario, BoT achieves the fastest convergence, yielding a 35.7% FLOPs saving that significantly outperforms both KD and WS. This strong performance is mirrored in the S2L direction, where our approach again leads all baselines, delivering a 54.7% FLOPs saving, a massive +32.6% improvement over the expansion methods, Mango and LiGO. These results demonstrate that our framework's efficiency gains are not specific to one architecture but are broadly applicable across different Transformer variants.

The downstream task results, presented in Table 14, further validate the superiority of our initialization. In the L2S setting for RoBERTa-S, BoT achieves the highest average scores on both GLUE (74.74) and SQuAD (35.98), dramatically outperforming the WS baseline, especially on SQuAD (+14.15 points). In the S2L setting for RoBERTa-B, BoT again secures the top performance on both benchmarks, surpassing all contemporary expansion methods including LiGO and Mango. This consistent, state-of-the-art performance across different model and scaling directions solidifies our framework as a powerful and universally effective solution for cross-architecture knowledge transfer.

To further assess the generality of our framework, we conducted parallel experiments on RoBERTa [22]. We evaluated bidirectional transfers (RoBERTa-B $\rightarrow$ RoBERTa-S and vice versa) by first pre-training the initialized models and then fine-tuning them on the GLUE and SQuAD benchmarks. The results are presented in Table 15. In both L2S and S2L scenarios, BoT once again demonstrates unparalleled computational efficiency. On RoBERTa-S, BoT leads in efficiency with a 35.7% FLOPs saving, surpassing the standard WS baseline. Similarly, on RoBERTa-B, BoT achieves a state-of-the-art 54.7% FLOPs saving and 51.5% Walltime saving, which is over 30 percentage points higher than the methods, LiGO and Mango. While delivering this massive efficiency gain, its final downstream performance on GLUE (81.19) and SQuAD (60.5) remains highly competitive and on par with these more resource-intensive baselines. In contrast, BoT provides strong downstream performance while strictly reducing training costs. These results

Table 14. Performance comparison on the GLUE and SQuAD benchmarks for RoBERTa. Models are initialized and then directly fine-tuned without additional pre-training. The "Avg" columns show the average scores for each benchmark.

Model	Methods	GLUE Benchmark								SQuAD Benchmark				
		SST-2	MNLI	MRPC	CoLA	QNLI	QQP	STS-B	**Avg**	v1.1 (EM)	v1.1 (F1)	v2.0 (EM)	v2.0 (F1)	**Avg**
RoBERTa-S	Scratch	88.19	75.30	76.47	21.27	85.19	88.48	82.87	73.97	7.99	18.02	3.39	8.16	9.39
	WS	88.42	75.26	78.43	25.04	83.53	88.12	80.84	74.23	23.75	32.72	12.79	18.04	21.83
	DWT	**88.42**	**76.08**	**78.43**	**23.27**	**85.58**	**88.62**	**82.81**	**74.74**	**42.58**	**54.22**	**20.26**	**26.87**	**35.98**
RoBERTa-B	Scratch	81.65	31.82	69.61	2.16	61.58	77.50	15.69	48.57	8.80	18.76	3.92	8.86	10.09
	bert2BERT	81.19	61.91	70.59	10.51	59.86	79.94	20.86	54.98	13.64	24.08	5.39	10.62	13.43
	LiGO	87.61	76.87	71.08	21.72	83.40	87.83	70.86	71.34	68.34	78.41	33.58	38.84	54.79
	Mango	87.50	77.04	70.59	21.72	83.34	87.79	69.46	71.06	68.75	78.63	33.53	38.90	54.95
	DWT	**88.92**	**78.82**	**72.38**	**22.45**	**84.54**	**87.91**	**71.12**	**72.31**	**69.87**	**79.18**	**34.22**	**39.72**	**55.75**



(a) RoBERTa-B → RoBERTa-S

(b) RoBERTa-S → RoBERTa-B

Figure 8. Results of pretraining RoBERTa-S and RoBERTa-B. BoT can achieve the highest savings in FLOPs with 35.7% for RoBERTa-S, 54.7% for RoBERTa-B from the Scratch models.

on RoBERTa reinforce our primary finding: BoT achieves both high computational efficiency and strong final model performance, making it a highly practical solution for cross-architecture knowledge transfer.

D.2. Impact of Scale Disparity in Bidirectional Transfer

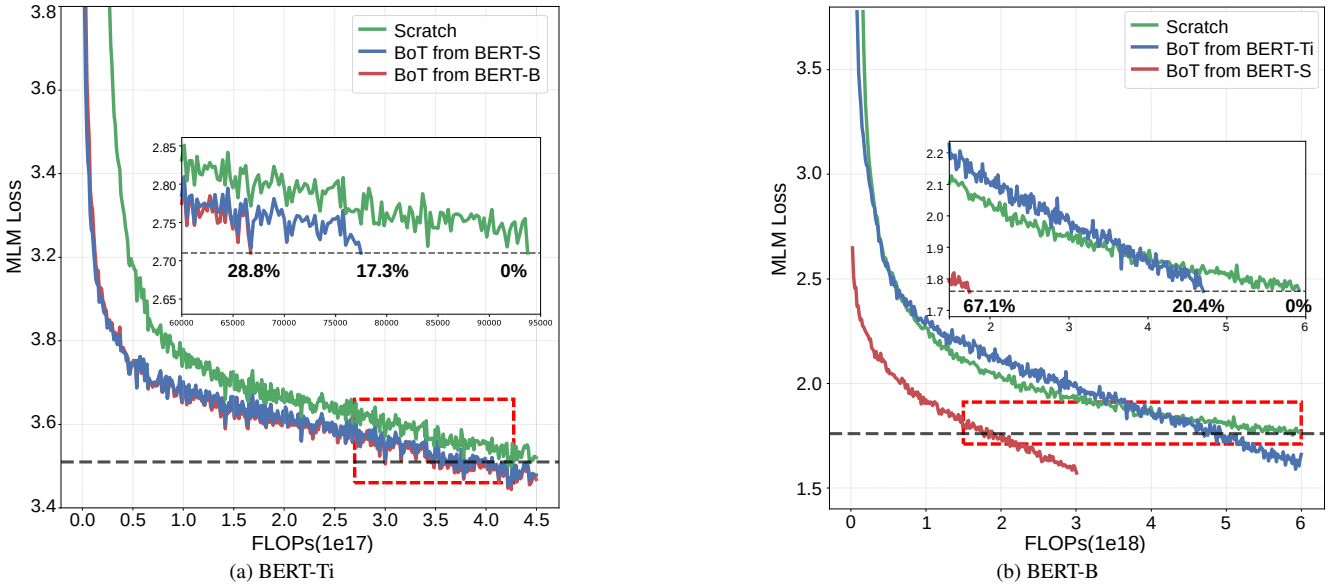
To analyze how the disparity in scale between source and target models affects BoT's performance, we conducted two sets of bidirectional experiments. First, in a L2S setting, we initialized a BERT-Ti model from both a BERT-S and a BERT-B. Second, in a S2L setting, we initialized a BERT-B model from a BERT-S and an even smaller BERT-Ti. We compared the convergence of all initialized models against a standard from-scratch baseline. The results, illustrated in Figure 9, reveal two key insights into BoT's ability to transfer core knowledge.

In the L2S scenario for initializing BERT-Ti, a larger source model provides a marginal but consistent advantage. As seen in Figure 9a, initializing from BERT-B achieves the fastest convergence with a **28.8% FLOPs saving**. While initializing from BERT-S is also highly effective (17.3% FLOPs saving), its performance does not fully match that of the larger source. This result implies that while the most critical knowledge required by the Tiny model is largely present in the BERT-S architecture, the additional capacity of BERT-B does contain further, albeit diminishing, transferable benefits. This suggests a trade-off between the source model's size and the efficiency gains for very small target models.

In the S2L scenario for initializing BERT-B, a larger source model provides a more potent initialization. As shown in Figure 9b, initializing from BERT-S is highly effective, achieving a **67.1% FLOPs saving**. While initializing from BERT-Ti still yields a substantial **20.4% FLOPs saving**, its acceleration is less pronounced. This suggests

Table 15. Performance comparison on GLUE and SQuAD benchmarks for RoBERTa after pre-training from initialization. This table shows the final performance of models that were first pre-trained and then fine-tuned. "Avg" columns show the average scores for each benchmark.

Model	Methods	Savings		GLUE Benchmark								SQuAD Benchmark		
		FLOPs	Walltime	SST-2	MNLI	MRPC	CoLA	QNLI	QQP	STS-B	**Avg**	v1.1 (f1/em)	v2.0 (f1/em)	**Avg**
RoBERTa-S	Scratch	0.0%	0.0%	89.56	76.64	79.41	20.39	84.90	88.32	82.59	74.54	76.2 / 65.8	37.5 / 32.0	52.9
	WS	32.4%	32.3%	88.50	76.85	78.19	21.62	83.54	88.15	81.43	74.04	75.3 / 64.7	37.6 / 32.1	52.4
	BoT	**35.7%**	**35.8%**	**89.33**	**76.78**	**76.47**	**23.18**	**84.61**	**88.71**	**82.93**	**74.57**	**76.7 / 66.3**	**37.9 / 32.4**	**53.3**
RoBERTa-B	Scratch	0.0%	0.0%	90.71	82.59	83.33	45.87	89.75	90.38	85.44	81.15	84.4 / 74.7	43.6 / 38.8	60.4
	bert2BERT	7.2%	6.6%	91.63	82.00	86.03	41.04	89.00	90.21	85.20	80.73	83.8 / 74.2	41.5 / 36.6	59.0
	LiGO	22.1%	20.3%	90.37	82.38	83.82	44.96	90.19	90.18	85.16	81.01	84.7 / 75.5	42.3 / 37.3	59.9
	Mango	22.2%	20.4%	90.83	81.99	84.31	44.36	90.43	90.23	85.87	81.15	84.6 / 76.3	42.2 / 38.4	60.4
	BoT	**54.7%**	**51.5%**	**90.60**	**82.91**	**84.82**	**44.00**	**90.13**	**90.19**	**85.70**	**81.19**	**84.7 / 74.2**	**43.8 / 39.2**	**60.5**



(a) BERT-Ti (b) BERT-B

Figure 9. Analysis of the impact of scale disparity on BoT's performance. (a) Pre-training convergence of a BERT-Ti model initialized from BERT-S and BERT-B sources. (b) Pre-training convergence of a BERT-B model initialized from BERT-Ti and BERT-S sources.

that a larger source model contains a richer, more comprehensive low-frequency knowledge, which is more beneficial when expanding to a much larger architecture.

Collectively, these bidirectional experiments demonstrate that BoT effectively isolates and transfers the most critical, low-frequency knowledge. For adaptation to a small target, a moderately-sized source can provide most of the benefits of a massive one, highlighting BoT's efficiency. Conversely, for model expansion (S2L), a larger and more capable source model is clearly more advantageous. This underscores the robustness of our framework across different scaling directions and disparities.

D.3. Ablation on High-Frequency Coefficient Padding Strategies in S2L Transfer

To further investigate the role of high-frequency detail coefficients in **Small-to-Large (S2L)** transfers, we conduct an ablation study comparing different padding strategies when reconstructing the target base model from a smaller source model using IDWT. In our framework (BoT), the high-frequency sub-bands $\{cD_m\}_{m=1}^{7}$ are typically set to zero tensors (zero padding). We hypothesize that the initialization distribution of these coefficients may influence convergence and downstream transferability.

We perform S2L transfer from a DeiT-S (6-layer, hidden size 384) to a DeiT-B (12-layer, hidden size 768). We compare three strategies for initializing the high-frequency detail coefficients before applying 3D-IDWT:

Method	CIFAR-10	CIFAR-100	CUB200	Flowers102	Food-101	iNat-2019	Cars	Avg.
Random Uniform	95.13	74.73	62.53	69.01	73.94	51.92	52.11	68.85
Random Gaussian	96.87	80.14	73.91	90.63	83.19	65.02	79.92	81.67
Zero Padding (BoT)	**97.24**	**82.38**	**74.08**	**94.71**	**84.44**	**67.14**	**88.64**	**84.37**

Table 16. Ablation on padding strategies for high-frequency coefficients in IDWT-based S2L transfer. Zero padding consistently yields the best average accuracy. We report top-1 accuracy (%) on seven downstream datasets after direct fine-tuning.

1. **Random Uniform Padding**: coefficients are sampled from $\mathcal{U}(-\sigma, \sigma)$ with σ defined as above. (Similar to Gaussian padding; omitted in main table.)
2. **Random Gaussian Padding**: all cD_m sub-bands are filled with values drawn from $\mathcal{N}(0, \sigma^2)$, where σ matches the empirical standard deviation of the corresponding low-frequency coefficients cA in the source model.
3. **Zero Padding (default BoT)**: all cD_m sub-bands are set to zero tensors.

From Table 16, zero padding consistently outperforms other padding across all datasets—yielding a $+2.70\%$ average accuracy improvement. This confirms that setting high-frequency coefficients to zero provides a cleaner and more stable initialization for IDWT-based reconstruction, avoiding the noise introduced by randomly sampled detail bands. Although Gaussian padding still significantly outperforms training from scratch, our results suggest that such random detail initialization may misalign with the structural correlations embedded in the learngene, thereby hindering fine-grained recognition tasks (e.g., Cars, CUB200). In practice, **zero padding remains the most effective and robust strategy**, reinforcing our choice in the main BoT pipeline.