HCL-FF: Hierarchical and Contrastive Learning for Forward-Forward Algorithm

Supplementary Material

1. Implementation Details

For all experiments, we adopt a residual Forward-Forward (FF) architecture [4] consisting of four residual blocks. Table 1 summarizes the full architectural configurations and training hyperparameters. CIFAR-10 and CIFAR-100 use channel widths of $[100, 200, 400, 800]$ across the four residual blocks; MNIST and F-MNIST use lighter widths of $[40, 80, 160, 320]$; and Tiny-ImageNet employs a larger configuration of $[200, 400, 800, 1600]$. The channel width of the very first layer of the network is set equal to the width of the first residual block. Note that the channel dimension must be greater than or equal to the number of classes. Therefore, for Tiny-ImageNet, which contains 200 classes, the model begins with 200 channels. For contrastive learning, we employ a projection head implemented by a single linear layer that maps features into a 128-dimensional embedding space.

All models are trained using the Adam optimizer with a weight decay of 1×10^{-4}. For datasets with larger label spaces, such as CIFAR-100 and Tiny-ImageNet, we use a batch size of 512, while a batch size of 128 is used for CIFAR-10, MNIST, and F-MNIST. Models are trained for 1000 epochs on CIFAR-10/100 and Tiny-ImageNet, and for 150 epochs on MNIST and F-MNIST due to their simpler visual complexity. We employ cosine annealing from an initial learning rate of 8×10^{-2} down to 2×10^{-4}.

To stabilize the early stages of contrastive learning and progressively tighten semantic alignment, we schedule the contrastive temperature τ throughout training. Specifically, τ is linearly warmed up from 0.8 to 0.2 over the first 100 epochs, followed by cosine decay to 0.08 for the remainder of training. This schedule mitigates unstable gradients at initialization and encourages increasingly fine-grained feature separation as training progresses.

For natural-image datasets (CIFAR-10, CIFAR-100, and Tiny-ImageNet), we apply strong augmentations to increase intra-class variability. The augmentation pipeline consists of `RandomResizedCrop` with a scale range of $(0.6, 1.0)$, `RandomHorizontalFlip` with probability 0.5, `ColorJitter` (brightness/contrast/saturation = 0.2 and hue = 0.1) with probability 0.8, and `RandomGrayscale` with probability 0.2. For MNIST and F-MNIST, we adopt lightweight geometric augmentations to avoid distorting digit or clothing structures. The augmentation pipeline includes `RandomRotation` within $\pm 10^{\circ}$ and `RandomAffine` with up to 10% translation in both spatial directions and a scaling factor in the range $[0.9, 1.1]$.

All HCL-FF experiments follow a two-stage training procedure. In the first stage, we pretrain a model without hierarchical learning. The resulting pretrained model is then used to construct the data-driven hierarchy. In the second stage, we train the full HCL-FF model from scratch with both hierarchical and contrastive learning.

For comparison, the BP-trained ResNet-20 baseline is reproduced using the exact configuration described in the original ResNet paper [1]. We also train a matching-parameter variant by increasing all channel widths of the original configuration by a factor of ten to closely match the parameter count of our HCL-FF models.

2. Residual Shortcuts

Residual structures are commonly used in backpropagation-based (BP) networks to ease optimization by providing gradient shortcut pathways [1]. In the FF setting, where gradients are not propagated across layers, residual shortcuts instead serve to fuse information across depths. Following DeeperForward [4], we adopt parameter-free shortcuts that adapt spatial and channel dimensions without introducing any learnable projection layers. For completeness, we describe the residual mechanism in detail in this section.

Spatial alignment. Let $\mathbf{z}_r^{(\ell)} \in \mathbb{R}^{C \times H_r \times W_r}$ denote the shortcut feature for layer ℓ, and let the goodness-decoupled feature of the current layer be $\mathbf{z}^{(\ell)} \in \mathbb{R}^{C \times H \times W}$. When transitioning between residual blocks, the spatial resolution is downsampled by a factor of two, causing $(H_r, W_r) \neq (H, W)$. We therefore apply a 2×2 average-pooling operation with stride 2 when a resolution mismatch occurs:

$$\tilde{\mathbf{z}}_r^{(\ell)} = \begin{cases} \mathrm{AvgPool}_{2\times 2}\left(\mathbf{z}_r^{(\ell)}\right), & (H_r, W_r) \neq (H, W), \\ \mathbf{z}_r^{(\ell)}, & (H_r, W_r) = (H, W). \end{cases} \tag{1}$$

Channel alignment across residual blocks. At the transition between residual blocks, the main branch doubles its channel width. Let $\mathbf{z}^{(\ell)} \in \mathbb{R}^{C \times H \times W}$ be the goodness-decoupled feature at the end of one residual block, and let the shortcut feature be $\tilde{\mathbf{z}}_r^{(\ell)} \in \mathbb{R}^{C \times H \times W}$. To construct the input to the next residual block, which expects $2C$ channels, the two tensors are concatenated along the channel dimension:

$$\mathbf{z}_{\text{merge}}^{(\ell)} = \mathrm{Concat}_{\text{channel}}\left(\mathbf{z}^{(\ell)},\, \tilde{\mathbf{z}}_r^{(\ell)}\right) \in \mathbb{R}^{(2C) \times H \times W}. \tag{2}$$

Table 1. Architectural configurations and training hyperparameters for all datasets.

	CIFAR-10	CIFAR-100	MNIST	F-MNIST	Tiny-ImageNet
Channels per residual block	[100, 200, 400, 800]	[100, 200, 400, 800]	[40, 80, 160, 320]	[40, 80, 160, 320]	[200, 400, 800, 1600]
Batch size	128	512	128	128	512
Optimizer	Adam	Adam	Adam	Adam	Adam
Weight decay	1e-4	1e-4	1e-4	1e-4	1e-4
Initial learning rate	8e-2	8e-2	8e-2	8e-2	8e-2
Minimum learning rate	2e-4	2e-4	2e-4	2e-4	2e-4
Epochs	1000	1000	150	150	1000

Table 2. Effect of residual shortcuts on accuracy.

Dataset	w/o residual	w/ residual
CIFAR-10 Accuracy	90.85	91.83
CIFAR-100 Accuracy	65.43	70.76

This yields a channel-aligned shortcut without any learned projection.

Residual fusion within a residual block. Within the same residual block, the main branch and shortcut share the same spatial and channel dimensions. Thus, we perform simple additive fusion:

$$\mathbf{z}^{(\ell)}_{\text{merge}} = \mathbf{z}^{(\ell)} + \tilde{\mathbf{z}}^{(\ell)}_{r} \in \mathbb{R}^{C\times H\times W}. \tag{3}$$

Discussion. The shortcut path participates in neither gradient flow nor weight updates, preserving the locality of FF training. However, these lightweight residuals substantially improve representational quality by propagating useful features across depth. As shown in Table 2, removing residual shortcuts causes significant accuracy degradation on CIFAR-10 and CIFAR-100, highlighting their importance for stable and effective layer-wise FF training.

3. Signal Integrating and Pruning Module

In this section, we elaborate on the Signal Integrating and Pruning (SIP) module introduced in DeeperForward [4]. Unlike BP-based models, which rely solely on the final layer's logits, FF networks produce a goodness score at every layer. Each layer-wise goodness vector $\mathbf{g}^{(\ell)}$ already encodes partial class evidence at depth ℓ, providing an opportunity to aggregate predictions across the entire network hierarchy.

Formulation. Given a model with L layers, each layer produces a class-wise goodness vector $\mathbf{g}^{(\ell)} \in \mathbb{R}^K$, where K is the number of classes. Instead of relying exclusively on the deepest layer, SIP forms an aggregated prediction by averaging goodness scores over a selected contiguous interval of layers:

$$\tilde{\mathbf{g}} = \frac{1}{e-s+1}\sum_{\ell=s}^{e}\mathbf{g}^{(\ell)}, \qquad 0 \le s \le e \le L-1. \tag{4}$$

The final class label is obtained by taking the $\arg\max$ over $\tilde{\mathbf{g}}$. The interval $[s, e]$ is chosen on a validation set and kept fixed during inference.

Table 3. Accuracy under different prediction strategies. "All layers" averages goodness scores from all layers, "Last layer" uses only the deepest layer, and "SIP" reports the accuracy and the selected interval $[s, e]$ found on the validation set.

Method	All layers	Last layer	SIP [s, e]
MNIST			
DeeperForward	99.61	99.50	99.67 [4, 8]
Ours	99.51	99.62	99.70 [13, 16]
F-MNIST			
DeeperForward	93.01	91.73	92.97 [1, 13]
Ours	89.15	93.75	93.97 [13, 16]
CIFAR-10			
DeeperForward	86.16	79.77	86.24 [1, 16]
Ours	80.39	92.00	91.83 [10, 16]
CIFAR-100			
DeeperForward	52.63	26.56	52.43 [0, 12]
Ours	49.23	67.42	70.76 [7, 16]
Tiny-ImageNet			
DeeperForward	33.96	3.05	35.95 [1, 12]
Ours	33.39	48.46	48.46 [16, 16]

Analysis. Table 3 reveals distinct SIP behaviors for DeeperForward and HCL-FF. For HCL-FF, SIP consistently selects deeper intervals, e.g., $[13, 16]$ on MNIST and F-MNIST, $[10, 16]$ on CIFAR-10, $[7, 16]$ on CIFAR-100, and $[16, 16]$ on Tiny-ImageNet. This pattern aligns with our hierarchical design: early layers focus on coarse distinctions, while deeper layers progressively refine semantics, making them more informative for final prediction. In contrast,

DeeperForward frequently selects shallow or mid-level layers, indicating that deeper layers contribute weaker or noisier predictions. This supports our analysis that goodness decoupling, without hierarchical or contrastive grounding, leads to semantic drift in deeper representations, preventing meaningful development across depth.

We additionally consider two baselines: averaging goodness across all layers and using only the final layer. For HCL-FF, all-layer averaging reduces performance because early layers produce coarse super-class signals that hinder fine-grained classification. However, our last-layer accuracy substantially exceeds that of DeeperForward, demonstrating that HCL-FF maintains discriminative structure even at the deepest layers and benefits from a meaningful depth-wise learning trajectory.

4. Building Hierarchies

In this section, we describe how we construct the class hierarchy used for hierarchical supervision in detail and clarify the rationale behind this design. Following the Neural-Backed Decision Tree (NBDT) framework [5], we employ a data-driven procedure that derives a semantic hierarchy directly from a trained classifier. This approach eliminates the need for manual specification of coarse and fine semantic groups, offering a systematic method to construct a hierarchy for any dataset.

Classifier weights as class prototypes. As outlined in Sec. 3.2 of the main paper, we begin by training a linear classifier on top of the frozen backbone. Let $W \in \mathbb{R}^{K \times D}$ denote the weight matrix of a trained linear classifier over K classes, where each row $W_k \in \mathbb{R}^D$ corresponds to class k. For an input feature $\mathbf{z}$, the classifier logit for class k is

$$\text{logit}_k(\mathbf{z}) = W_k^\top \mathbf{z} + b_k. \tag{5}$$

Because the dot product $W_k^\top \mathbf{z}$ is maximized when $\mathbf{z}$ aligns with W_k, each row vector W_k implicitly acts as the prototype or anchor direction defining the region of feature space associated with class k. This interpretation is well supported by prior work [2, 5–7].

From prototypes to semantic hierarchy. Before computing inter-class similarity, each prototype is L2-normalized, $\hat{W}_k = \frac{W_k}{\|W_k\|_2}$. We then compute pairwise cosine distances,

$$d(k, k') = 1 - \hat{W}_k^\top \hat{W}_{k'}, \tag{6}$$

and apply hierarchical agglomerative clustering with Ward linkage to recursively merge the most similar classes. This produces a full binary hierarchy that reflects the semantic relationships encoded by the classifier. Intuitively, classes with similar prototype directions are more confusable and therefore merge earlier in the hierarchy, yielding a semantically meaningful tree structure.

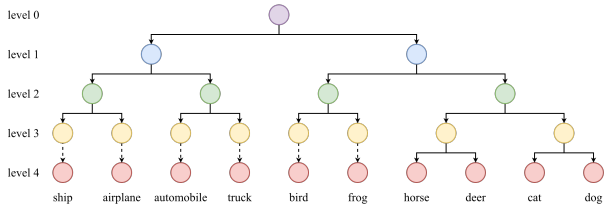

Figure 1. Hierarchy of CIFAR-10 classes constructed via clustering on class prototypes derived from a pre-trained classifier. Each level represents a different granularity of super-class grouping.

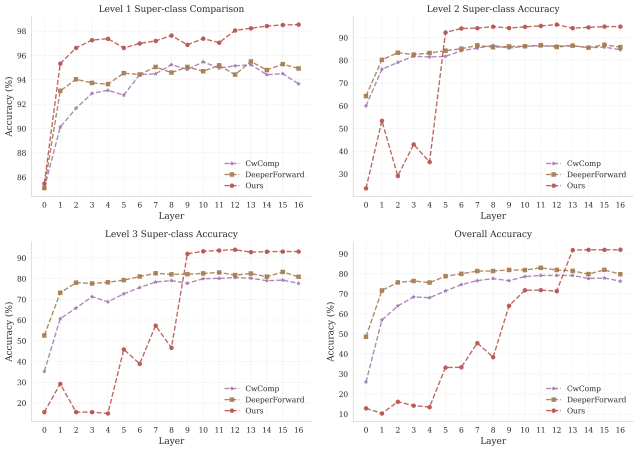

Figure 2. Layer-wise super-class accuracy at different hierarchy levels and overall fine-grained accuracy.

5. Super-class Accuracy

To complement the per-layer analysis presented in Fig. 3 of the main paper, we further examine layer-wise accuracy at each hierarchy level, comparing against CwComp [3] and DeeperForward [4]. Figure 2 reports Level 1, Level 2, and Level 3 super-class accuracy, as well as overall fine-grained accuracy. The hierarchy levels correspond to those defined in Fig. 1.

Our model exhibits a clear and structured progression across depth. In the shallow network from layer 0 to layer 4, HCL-FF achieves strong performance on Level 1 super-classes while showing limited ability to discriminate fine-grained categories, which is precisely the behavior encouraged by hierarchical supervision. As depth increases, the model receives finer-grained targets and correspondingly improves its performance on deeper hierarchy levels and on fine-grained classification. This demonstrates that hierarchical supervision successfully guides the network to develop increasingly specialized semantic representations.

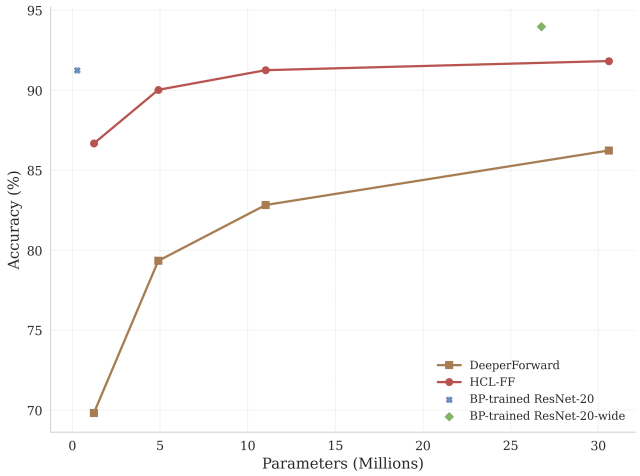

Figure 3. CIFAR-10 accuracy versus parameter count for DeeperForward, HCL-FF, and BP-trained ResNet baselines.

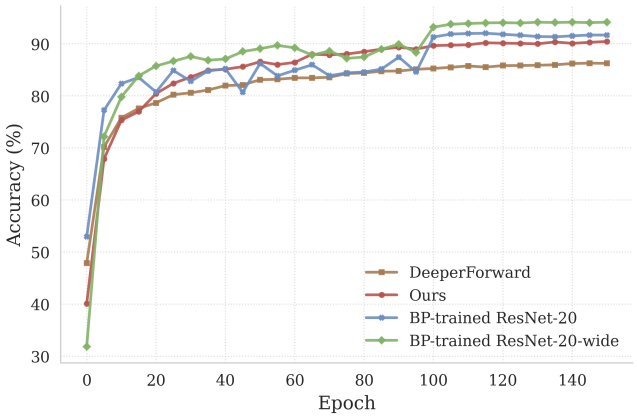

Figure 4. CIFAR-10 training curves under a matched 150-epoch schedule. HCL-FF converges faster and reaches higher accuracy than DeeperForward.

In contrast, both CwComp and DeeperForward require the shallow layers to immediately solve the full fine-grained classification problem. Although this can yield higher fine-grained accuracy in the early layers, it comes at the expense of poorer performance on coarse super-class distinctions. This pattern suggests that these methods prematurely force early layers toward high-level abstraction, reducing the stability and generality of their representations.

6. Model Efficiency

To evaluate parameter efficiency, we compare HCL-FF with DeeperForward, along with BP-trained ResNet-20 and the parameter-matched ResNet-20-wide baseline. Figure 3 reports CIFAR-10 accuracy as a function of total parameter count. We evaluate four model capacities for both HCL-FF and DeeperForward, where the channel widths of the four residual blocks are set as follows:

- **Tiny**: $[20, 40, 80, 160]$ channels,
- **Small**: $[40, 80, 160, 320]$ channels,
- **Medium**: $[60, 120, 240, 480]$ channels,
- **Large**: $[100, 200, 400, 800]$ channels (as in Sec. 1).

Across all model sizes, HCL-FF achieves substantially higher accuracy than DeeperForward. The HCL-FF curve consistently lies above that of DeeperForward, demonstrating a markedly more favorable accuracy-capacity trade-off. Moreover, the tiny configuration (1.225M parameters) of our HCL-FF model attains 86.68% accuracy, exceeding DeeperForward's large model (30.603M parameters, 86.24%) while using less than 5% of its parameters.

Nevertheless, we acknowledge that a performance gap remains between layer-wise training and full end-to-end BP optimization. Our medium configuration (11.018M parameters) achieves a comparable accuracy of 91.26% to BP-trained ResNet-20 (91.25%). However, it requires more parameters to achieve this level of accuracy. Moreover, our large HCL-FF model still falls short of the parameter-matched ResNet-20-wide by a few percentage points. This suggests that while hierarchical and contrastive guidance greatly enhance FF-based training, further advances in local learning objectives are needed to close the remaining gap to BP.

Overall, the results in Figure 3 show that HCL-FF delivers significantly improved parameter efficiency over prior FF-based methods, bringing FF training meaningfully closer to BP-trained networks while maintaining its fully local training paradigm.

7. Learning Dynamics

To further compare the convergence behavior of HCL-FF, DeeperForward, and standard backpropagation, we examine their CIFAR-10 learning curves under a matched training schedule. All models are trained for 150 epochs for a fair comparison. Figure 4 reports test accuracy across training epochs.

HCL-FF converges noticeably faster and to a higher accuracy than DeeperForward. By epoch 20, HCL-FF already surpasses DeeperForward and maintains this advantage throughout training, indicating that hierarchical supervision and contrastive grounding provide stronger and more stable learning signals.

BP-trained ResNet-20 exhibits larger fluctuations during the early and mid stages of training, reflecting the dynamics of joint gradient updates across all layers. Despite this instability, it eventually achieves higher final accuracy, highlighting the remaining challenges of purely layer-wise optimization compared to end-to-end backpropagation.

Overall, the learning curves confirm that HCL-FF delivers both faster and smoother convergence than prior FF-based approaches, while narrowing the gap to BP-trained

Table 4. Effect of alternative hierarchy mapping strategies.

Dataset	Incremental	Decremental	Balanced
CIFAR-100 Accuracy	70.30	68.40	70.76

networks under identical training conditions.

8. Hierarchy Mapping

In addition to the hierarchy mapping method described in Eq. 10 of the main paper, we evaluate two alternative strategies for assigning hierarchy levels across layers.

(1) Incremental mapping. This strategy assigns coarse supervision to the earliest layers and gradually increases granularity with depth. Specifically, for a network with L layers and a hierarchy of depth D, we define:

$$\text{level}(i) = \min(1 + i, D), \quad i = 0, \ldots, L - 1, \tag{7}$$

Thus, layer 0 receives Level 1 supervision, layer 1 receives Level 2, and so forth, until the maximum depth D is reached, after which all deeper layers are supervised at Level D. This mapping encourages early layers to transition quickly toward finer semantic resolutions.

(2) Decremental mapping. Conversely, this strategy assigns fine-grained supervision to the deepest layers and progressively coarser supervision toward the input:

$$\text{level}(i) = \max(D - (L - 1 - i), 1), \quad i = 0, \ldots, L - 1, \tag{8}$$

Here, shallow and mid-level layers are trained on very coarse super-classes, and only the final few layers receive fine-grained targets.

Results. Table 4 compares the performance of these mapping strategies. For clarity, we refer to the hierarchy assignment defined in Eq. 10 of the main paper as the *balanced* mapping, since it distributes hierarchy levels more evenly across depth. The incremental mapping performs similarly to the balanced strategy, while the decremental mapping results in a clear performance drop. This outcome reflects the importance of assigning sufficiently fine supervision to the middle layers. Although these layers have substantial representational capacity, the decremental mapping constrains them to overly coarse tasks, preventing the model from developing nuanced intermediate features. As a result, fine-grained supervision is concentrated only at the very end of the network, and the deeper hierarchy levels are not effectively leveraged across depth.

Table 5. Effect of batch size.

Batch size	32	64	128	256	512
CIFAR-10 Accuracy	92.10	92.22	91.83	91.73	91.96

9. Effect of Batch Size

Table 5 shows the effect of batch size on CIFAR-10 performance. HCL-FF demonstrates stable performance across a broad range, with accuracy varying by less than 0.5% between batch sizes of 32 and 512. This indicates that the contrastive component of HCL-FF does not rely on extremely large batches, in contrast to many contrastive learning frameworks that require thousands of samples per batch for stable gradients. Two factors may contribute to this robustness. First, layer-wise training restricts each update to a smaller subset of parameters, which stabilizes optimization and reduces the need for large batch sizes. Second, hierarchical supervision provides a strong coarse-to-fine semantic structure that helps maintain stable training dynamics even at smaller batch scales.

Finally, we observe slightly higher accuracy at smaller batch sizes. Because the number of epochs is fixed, smaller batches yield more update steps. Since layer-wise learning is less sample-efficient than end-to-end backpropagation, these additional updates give deeper layers more opportunities to refine their representations using information propagated from earlier layers.

10. Computational Cost

To demonstrate the computational aspect of HCL-FF, we compare HCL-FF against a BP-trained ResNet variant and DeeperForward under matched training conditions. All models have comparable numbers of parameters and are trained for 200 epochs with a batch size of 512, ensuring fair model size and equal numbers of weight updates.

For HCL-FF, we report the computational cost of its two training stages separately. The first stage (*Ours-pretraining*) trains the model without hierarchical learning and is used to construct the data-driven hierarchy. The second stage (*Ours*) trains the full model from scratch with both hierarchical and contrastive objectives. To further contextualize the computational overhead, we additionally include an extended DeeperForward baseline trained for 750 epochs (*DeeperForward-extend*), which approximately matches the total GPU hours consumed by our two-stage pipeline.

Table 6 summarizes classification accuracy, total GPU hours, and GPU memory usage. Several observations can be made. First, FF-based methods are substantially more memory-efficient than BP-based training. ResNet20-Wide requires 9916 MiB of GPU memory, whereas all FF-based

Method	ResNet20-Wide	DeeperForward	Ours-pretraining	Ours	DeeperForward-extend
CIFAR-100 Acc.	74.91	53.57	63.63	69.03	55.01
Training GPU Hours (NVIDIA RTX A6000)	2.21	0.79	1.32	1.36	3.02
Training GPU Memory Usage (MiB)	9916	3552	3848	3848	3552

Table 6. Comparison of computational cost and performance on CIFAR-100.

variants require between 3552 and 3848 MiB. This reduction stems from the layer-wise local learning paradigm, which avoids storing full-network activations for backpropagation.

Second, incorporating contrastive learning introduces only a modest increase in memory usage compared to DeeperForward (3848 MiB vs. 3552 MiB), while providing a significant improvement in accuracy (63.63% vs. 53.57%). Although the contrastive objective increases total GPU hours, the computational overhead remains moderate relative to the accuracy gains.

Third, the additional cost of hierarchical learning is minimal. The final HCL-FF model (*Ours*) exhibits nearly identical GPU memory usage and training time to the pretraining stage (*Ours-pretraining*), while achieving a further improvement in accuracy (69.03% vs. 63.63%). This indicates that hierarchical learning introduces negligible computational overhead while achieving substantial accuracy gains.

Finally, increasing computation alone does not bridge the performance gap. The extended DeeperForward baseline consumes more GPU hours (3.02 hours) than our complete two-stage pipeline (1.32 + 1.36 hours), yet achieves substantially lower accuracy (55.01% vs. 69.03%). This demonstrates that the performance gains of HCL-FF arise from the proposed hierarchical and contrastive learning mechanisms rather than from increased computational budget.

Overall, HCL-FF achieves significant accuracy improvements over prior FF-based methods with only modest additional training time and minimal memory overhead, while retaining the memory efficiency advantages of fully local learning.

11. Additional Qualitative Results

To further illustrate the hierarchical refinement learned by HCL-FF, we visualize the full 17-layer goodness responses for the same example shown in Fig. 5 of the main paper. The results in Fig. 5 reveal a clear coarse-to-fine progression: shallow layers respond mainly to low-level cues such as edges and textures, producing broad activations across many classes or super-classes, while deeper layers yield increasingly selective responses. Classes within the same super-class also tend to activate together, reflecting the structure imposed by hierarchical supervision.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[2] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017. 3

[3] Andreas Papachristodoulou, Christos Kyrkou, Stelios Timotheou, and Theocharis Theocharides. Convolutional channel-wise competitive learning for the forward-forward algorithm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 14536–14544, 2024. 3

[4] Liang Sun, Yang Zhang, Jiajun Wen, Linlin Shen, Weicheng Xie, and Weizhao He. Deeperforward: Enhanced forward-forward training for deeper and better performance. In *International Conference on Learning Representations*, 2025. 1, 2, 3

[5] Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Scott Lee, Henry Jin, Suzanne Petryk, Sarah Adel Bargal, and Joseph E Gonzalez. Nbdt: Neural-backed decision trees. *arXiv preprint arXiv:2004.00221*, 2020. 3

[6] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1041–1049, 2017.

[7] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5265–5274, 2018. 3

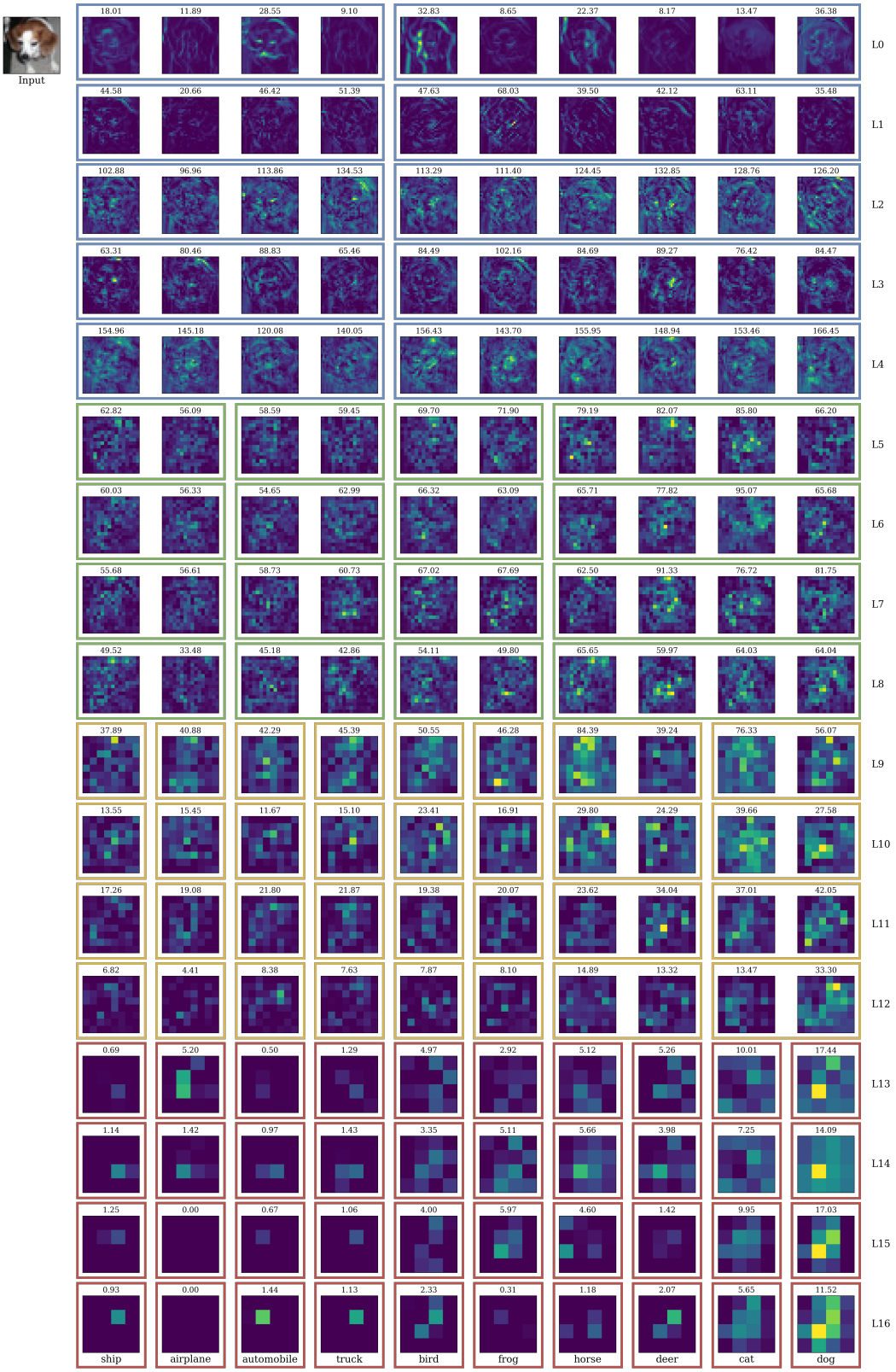

Figure 5. Visualization of class-wise goodness responses from every layer. Color frames correspond to the super-class grouping.