Hierarchical Resolution Training (HRT)

Authors: Dhaval Khatri

I present *Hierarchical Resolution Training (HRT)*, a progressive learning approach where neural networks are trained from simplified data representations toward increasingly detailed inputs. Unlike conventional progressive resizing, which scales image resolution, HRT focuses on *data abstraction*—for example, starting with edge maps before introducing full-resolution images. Experiments on image-based convolutional networks show that this approach stabilizes early feature learning and improves generalization, even without additional regularization or tuning. While current results are limited to simple image datasets, the findings support the hypothesis that layered data exposure can enhance learning efficiency and model stability.

1. Introduction

Experimenting with how neural networks are trained has become almost ritualistic. We borrowed inspiration from nature, mimicking neurons — and they worked. You feed in data, and it gives you results. Because it worked, we stopped asking why or what else might work.

Once neural networks became effective, the field largely shifted toward depth and complexity — smoothing activations, stacking layers, and refining architectures to capture ever more intricate patterns. In that pursuit, we often overlooked simpler, more intuitive ways of learning.

This paper revisits that simplicity through Hierarchical Resolution Training (HRT) — a method that mirrors how humans learn. We don't begin by replicating the fine details of reality; we start with structure. An artist begins with shapes before color, a child learns letters before words. HRT applies this principle to neural networks: teach broad structure first, then add detail.

While this idea could extend across modalities such as vision, text, and audio, this work focuses on the visual domain. The goal is to show how starting from simplified, structural data (edges) and gradually increasing input complexity can help models learn more generalizable and meaningful representations.

2. Related Work

Several existing approaches have explored elements of hierarchical or progressive learning:

1. **Progressive / curriculum learning (data-centric):**
    1. **From Coarse to Fine (Xu & Hall, 1994)** — introduced the concept of training models progressively, from simple to complex representations.
    2. **Curriculum Learning (Bengio et al, 2009)** — emphasized the importance of learning easy examples first and gradually introducing harder ones.

    3. **Progressive Image Resizing (Howard & Gugger, 2018)** — used in fast.ai and EfficientNet, where models are trained with low-resolution images initially, then refined with higher resolutions for improved convergence.
2. **Hierarchical / layer-wise network training (architecture-centric) :**
    1. **Deep Belief Networks (Hinton, 2006)** — introduced a deep generative architecture built from stacked Restricted Boltzmann Machines (RBMs), trained via greedy layer-wise unsupervised pretraining followed by supervised fine-tuning.
    2. **Stacked Autoencoders (Bengio, 2007)** — proposed layer-wise training of encoder–decoder blocks, where each autoencoder learns a latent representation that is passed up the hierarchy, then fine-tuned end-to-end.

3. How This Work Differs

Most existing approaches reduce *resolution*, not *complexity*. They blur the image, but don't simplify the world.

My initial thought was to go a step further — to teach models through simplified forms: polygons, curves, and lines. Like how artists sketch before painting, or how kids (and cavemen) represent people with circles and lines. It's the oldest learning trick there is: abstraction before detail.

Of course, converting images into clean geometric primitives is a challenge on its own. So, for this first experiment, I decided to use something simpler: **edges** — plain Sobel edge maps (Canny edges also work without harming convergence). Not "edge-enhanced" images, just edges themselves.

Now, I understand the hesitation. Edge data is sparse. It looks like a terrible idea on paper. But surprisingly, when paired with **layer freezing**, it can help a lot.

Here's the key intuition:
Feed each layer what it naturally wants to learn.
Early layers love edges and boundaries; later ones thrive on texture and fine detail. So why not help them along? Let the early layers stabilize on edge data, and then give them the full images to learn deeper features.

Why People Usually Avoid This

There are familiar reasons why this line of thought rarely gets explored:

- Sparse data hurts convergence.
- "Let the model learn everything by itself."
- Neural networks are black boxes — better not unscrew anything that seems to work.

The second argument, “let it learn by itself,” is probably the most sacred one in the field. And to be fair, it comes from a good place — neural networks are astonishing at learning their own features. Convolutional layers can discover edges, textures, shapes, and even abstract concepts without ever being told what they are. Given enough data and time, a deep network can, in theory, learn everything it needs to represent the world.

So the reasoning goes: why interfere? Why feed it edges or boundaries when it can find them on its own? Let it adapt, let it evolve — after all, that’s the beauty of end-to-end learning.

But this faith in self-organization sometimes becomes overconfidence. Models do learn, yes, but often inefficiently. They re-learn the same low-level features over and over, wasting compute cycles and data that could have been used for higher-level understanding. We let them reinvent edges, gradients, and shapes — things we already know they’ll need.

So the real question isn’t can the network learn by itself — it’s should it have to? If we can guide its early understanding — just a little — we might help it learn faster, generalize better, and spend less energy rediscovering the obvious.

How HRT Differs From Prior Layer-Wise Methods

Classical layer-wise approaches such as **Deep Belief Networks** (Hinton, 2006) and **Stacked Autoencoders** (Bengio, 2007) train each layer using the representation produced by the layer below. Although these methods change the form of the input at each depth, every layer still learns from the **same full-resolution data distribution**, only encoded differently. Their objective is to obtain better latent representations or more stable initializations—not to control the complexity of information delivered to each layer.

HRT diverges both in **intent** and **mechanism**. Instead of stacking progressively encoded versions of the same data, HRT explicitly **modulates the information scale** given to each depth. Early layers see simplified, edge-focused inputs; mid-layers receive blurred or partially detailed images; deeper layers observe the full-resolution data. The training signal is aligned with the spatial scale each layer is architecturally suited to learn, making HRT a *resolution-aware*, supervised reinterpretation of layer-wise learning, distinct from earlier unsupervised encoding-based methods.

A Note on Originality: *The idea behind HRT arose independently while exploring how neural networks—especially large or AGI-scale systems—could be made* ***denser, more compact, and able to use their neurons more efficiently****. The goal was to reduce redundancy in learned representations and find a way to train models so that each layer contributes meaningfully without wasting capacity. This line of thinking led to the insight that different layers prefer different levels of detail, and that training them with information matched to their representational role could yield tighter, more efficient networks. Only later did parallels with classical layer-wise pretraining become apparent, even though the underlying principle—****structurally aligning the information scale with the layer’s function****—is distinct from prior work.*

4. Methodology

To test the hypothesis, two architectures — SimpleCNN and ResNet-18 — were trained under two regimes:

- **Baseline:** Standard end-to-end training on full images.
- **Edge + Freeze (HRT):** Initial training on edge data, with deeper layers frozen, followed by standard training on full images.

Datasets: CIFAR-10, CIFAR-100, and Caltech-256.

SimpleCNN and ResNet-18 were evaluated on CIFAR-10 and CIFAR-100.
ResNet-18 was additionally tested on Caltech-256.

Training Details:

- Edge Training Epochs: 4
- Seeds: 42, 123, 3407, 7777
- Batch size: 64 for CIFAR-10 and CIFAR-100 and 32 for Caltech-256.
- Optimizer: Adam was used in all experiments.
- CIFAR-10 and CIFAR-100: Learning rate set to 0.001, with no scheduler or decay.
- Caltech-256: Pretrained model with layer-wise learning rates (values available in the accompanying code repository).
- Identical hyperparameters settings were used for both training modes.
- No special tuning, regularization, or aggressive augmentation was applied — ensuring the observed differences stemmed from HRT itself.
- For ResNet-18, **layer3** and **layer4** were frozen during the edge phase, enabling adaptation only in early convolutional and fully connected layers. **SimpleCNN** required no freezing due to its shallow architecture.

Scaling Perspective

While this study focuses on compact CNNs, the same principle could extend naturally to larger architectures. Conceptually, Hierarchical Resolution Training can be viewed as a staged unfreezing process: beginning with only the earliest layers active while the model is exposed to edge-based or simplified data, and progressively unfreezing deeper layers as input detail increases. As more complex textures and semantics are introduced, the model incrementally adapts its higher-level representations — mirroring how hierarchical abstraction forms in human learning. Code: https://github.com/dhaval-khatri1996/HRT

5. Results

Results are averaged over 4 seeds.

			Baseline			Edge + freeze		
Model	**Dataset**	**Epochs**	**val acc**	**train acc**	**val loss**	**val acc**	**train acc**	**val loss**
SimpleCNN	CIFAR-10	10	70.405	77.1675	0.877375	72.4625	78.4975	0.8042
	CIFAR-100	15	37.975	47.6725	2.503025	39.8425	49.24	2.40425
Resnet-18(Modified for CIFAR)	CIFAR-10	10	81.73	88.045	0.565225	83.3725	89.62	0.514625
	CIFAR-100	10	53.895	70.46	1.754825	55.84	68.015	1.644025

DATASET: Caltech-256		Baseline			Edge + freeze		
Model	**Epochs**	**Val acc**	**Top5 acc**	**val loss**	**val acc**	**Top5 acc**	**val loss**
Resnet-18 (pretrained, reset FC layer)	15	80.9425	92.605	0.84715	81.7075	93.57	0.79085
Resnet-18 (pretrained, reset conv1 and FC layer)	15	75.17	89.4025	1.111125	78.8875	91.8375	0.9203

Note: *Sparse data is generally expected to hinder neural networks, especially pretrained models. The results below show how edge-only inputs* ***can improve performance even during fine-tuning****. This supports the HRT hypothesis: stabilizing early layers on edges enables deeper layers to learn richer representations. It also motivates exploring HRT when data is limited or sparse.*

6. Discussion

Across all experiments, validation accuracy improved and validation loss decreased, indicating that HRT enhances both learning stability and generalization. After edge training, validation metrics quickly recovered to baseline and surpassed them within a few epochs of normal training.

Interestingly, baseline models exhibited higher mean activations than edge-trained ones. Although this might appear counterintuitive, lower activation magnitudes in HRT models indicate more selective and efficient feature learning — the network activates primarily for meaningful structures while ignoring redundant patterns. This corresponds with improved generalization, suggesting that lower activations reflect denser, more informative feature representations.

7. Limitations

This study tests the hypothesis in its simplest form. Ideally, HRT should be validated on deeper architectures and higher-resolution datasets. Due to limited computational resources, the experiments were kept lightweight and concept-focused.

While these results do not fully establish the hypothesis, they provide strong evidence of its effectiveness. The simplicity of the design was intentional — isolating the core principle without interference from large-scale tuning. The aim is not to demonstrate scale, but to validate a training philosophy: make learning easier by giving models what they need, when they need it.

Although similar staged-learning concepts exist in specific subfields, HRT deserves recognition as a general training paradigm and should be studied further across modalities.

8. Conclusion and Future Work

Despite relying on sparse edge data and lacking specialized tuning, models trained with HRT achieved superior results — challenging the assumption that sparse inputs hinder learning. These findings validate Hierarchical Resolution Training as an effective approach: models benefit from learning progressively, beginning with simplified structural inputs and gradually increasing data complexity. This approach facilitates optimization and strengthens generalization.

Intuitively, I see HRT as a training approach that facilitates the flow of features deeper into the network. By stabilizing the early layers, the deeper layers are free to learn richer and more abstract representations. Through progressive freezing and unfreezing, the model gradually builds the capacity to capture increasingly complex features.

I understand the idea may appear simple — almost too simple to matter. But sometimes simplicity reveals what we've overlooked. This small experiment leaves much unsaid, yet it opens the door to many directions — in how we train, how we structure models, and how we think about learning itself. I hope this work sparks those conversations, even if it only serves as a quiet beginning.

Future Work

The results here represent only the first step. HRT naturally points toward several exciting research directions:

1. Unsupervised HRT Modules (Resolution-Aware Stacked Learning)

A promising extension is to pretrain layer groups using unsupervised objectives, where each group receives data at the complexity level it is built to learn.
After training, these modules can be stacked and fine-tuned, potentially yielding:

- denser internal representations,
- reduced redundancy across layers,

- and smaller models with stronger expressive power.

This would be a resolution-aware reinterpretation of classic DBNs and stacked autoencoders.

Note: This is conceptually similar to multi-modal networks, where each modality is trained separately and later combined. Here, each sub-network learns a specific aspect of the input (e.g., edges, shapes, textures) before integration. This idea could also be applied recursively in multi-modal systems, where each modality is itself composed of stacked, progressively detailed modules.

2. Extending HRT to Multi-Modal Systems

In multi-modal models (vision, text, audio), staged-learning could reveal shared structural priors:
low-detail early knowledge forming "instinct-like" foundations for deeper cross-modal understanding.
This could open doors to entirely new architectures where modalities grow together in progressive stages of complexity.

3. Sparse Neural Networks via Unsupervised HRT Modules (Resolution-Aware Stacked Learning)

A natural extension of HRT is the design of **sparse neural networks built from unsupervised, resolution-aware modules** that learn in progressively richer stages. Sparse representations may encourage models to form denser and more meaningful internal features, rely less on texture memorization, generalize better by emphasizing stable structures, and drastically reduce computational cost.
Combining sparsity with resolution-aware stacked learning could produce compact, efficient networks that maintain strong generalization — an appealing direction for building smaller, practical AGI-oriented systems.

These directions point toward a long-term goal: **neural networks that learn the way humans draw, think, and reason — coarse first, fine later.**

References

1. Xu, L.Q., & Hall, T. (1994). From Coarse to Fine: A Novel Way to Train Neural Networks. ICANN ’94, Springer.
2. Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum Learning. ICML.
3. Howard, J., & Gugger, S. (2018). fast.ai: Progressive Image Resizing and Transfer Learning.
4. Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. Neural Computation, 18(7), 1527–1554.
5. Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy Layer-Wise Training of Deep Networks. NIPS 19, 153–160.