Notes on Distillation

Anjan Goswami

February 15, 2025

Abstract

Knowledge distillation has become a crucial framework for transferring knowledge from large, complex models to smaller, more efficient ones, facilitating practical deployment across various domains. This work provides a structured exploration of distillation, covering its theoretical underpinnings, algorithmic frameworks, and domain-specific applications. We begin by establishing a rigorous mathematical foundation, examining loss formulations, knowledge transfer mechanisms, and optimization strategies. The discussion extends to practical considerations such as feature transformation, attention-based alignment, and efficient student model initialization, ensuring both faster convergence and robust generalization.

Beyond core principles, we explore large language models (LLMs) and vision architectures, detailing transformer-specific distillation techniques, attention and hidden state transfer, and multi-task vision distillation. Additionally, we address emerging challenges in federated learning, cross-modal knowledge transfer, and adaptive distillation frameworks, which require novel solutions for scalability and efficiency.

The paper concludes with an analysis of open problems and future directions, emphasizing the interplay between theoretical guarantees, model compression strategies, and deployment constraints. By bridging mathematical insights with practical implementations, these notes serve as a comprehensive resource for researchers and practitioners aiming to refine distillation techniques in modern machine learning.

1 Introduction and Motivation

The rapid advancement of deep learning has led to increasingly powerful yet computationally demanding models. While these models achieve remarkable performance across various tasks, their practical deployment often faces significant challenges due to computational and memory constraints. Knowledge distillation, first formally introduced by Hinton et al. [5], emerges as a principled framework for transferring knowledge from complex models to simpler ones, effectively addressing the crucial challenge of model efficiency without severely compromising performance.

The fundamental premise of knowledge distillation lies in the observation that neural networks produce rich, informative probability distributions over their outputs, containing valuable information beyond mere class predictions. These "soft targets" encode subtle relationships learned by the model about the task domain. Formally, for a classification task with K classes, a teacher model T produces a probability distribution $\mathbf{p}_T = [p_1, ..., p_K]$ where:

$$p_i = \frac{\exp(z_i/\tau)}{\sum_{j=1}^{K} \exp(z_j/\tau)} \tag{1}$$

Here, z_i represents the logit for class i, and τ is a temperature parameter controlling the softness of the distribution. The student model S is then trained to match these distributions while maintaining accuracy on the original task.

Recent developments have significantly expanded this paradigm. Self-distillation, where knowledge is transferred between models with identical architectures [3], has demonstrated surprising performance gains, challenging our understanding of the optimization dynamics in deep networks. This phenomenon has been attributed to the implicit regularization effect of the distillation process, where each generation of the model learns a more refined representation of the task.

The scope of knowledge distillation extends beyond the traditional single teacher-student setup. Multiple teacher scenarios allow the integration of knowledge from diverse expert models, each potentially specialized in different aspects of the task. Formally, given teachers $\{T_1, ..., T_M\}$, we can formulate the objective as:

$$\min_{\theta_S} \sum_{m=1}^{M} w_m \mathcal{L}(S(x;\theta_S), T_m(x)) + \lambda \mathcal{R}(\theta_S) \tag{2}$$

where w_m represents the importance weight for each teacher, and $\mathcal{R}(\theta_S)$ is a regularization term. Similarly, multiple student scenarios enable parallel knowledge transfer, potentially capturing different aspects of the teacher's knowledge in specialized student models.

The architectural flexibility in knowledge distillation deserves special attention. While early work focused on transferring knowledge between models of similar architecture but different sizes, recent research has shown effective knowledge transfer between fundamentally different architectures. For instance, knowledge from transformer-based models can be distilled into CNNs or RNNs, opening new possibilities for deployment scenarios.

This rich landscape of possibilities has led to numerous theoretical and practical advances in the field, as comprehensively surveyed by Wang et al. [13]. The following sections will delve deeper into the mathematical foundations of these approaches, examining both the theoretical frameworks that underpin effective knowledge transfer and the practical algorithms that realize these insights in real-world applications.

2 Theoretical Foundations of Model Distillation

Model distillation aims to transfer knowledge from a large, complex **teacher model** $T(\theta_T)$ to a smaller, efficient **student model** $S(\theta_S)$. This section establishes the mathematical foundations of distillation, discussing **loss functions, transfer mechanisms, initialization strategies, stability-plasticity tradeoffs, convergence, sample complexity, generalization limits, and post-training.**

2.1 Loss Functions and Regularization

The student model is trained to minimize a weighted combination of multiple loss terms:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{task}} + \beta \mathcal{L}_{\text{logit}} + \gamma \mathcal{L}_{\text{feat}} + \delta \mathcal{L}_{\text{att}} + \lambda \mathcal{R}(\theta_S). \tag{3}$$

where: - **$\mathcal{L}_{\text{task}}$**: Task-specific loss (e.g., cross-entropy for classification). - **$\mathcal{L}_{\text{logit}}$**: KL divergence between teacher and student output probabilities. - **$\mathcal{L}_{\text{feat}}$**: Feature-based loss matching hidden layer representations. - **$\mathcal{L}_{\text{att}}$**: Attention-based loss for aligning attention distributions. - **$\mathcal{R}(\theta_S)$**: Regularization term to ensure stability and generalization.

Regularization Techniques. To stabilize training and improve generalization, we introduce:

- **Weight Decay**: $\mathcal{R}_{\text{L2}}(\theta_S) = \|\theta_S\|^2$ prevents overfitting.

- **Gradient Norm Regularization**: $\mathcal{R}_{\text{grad}} = \sum_i \|\nabla_{\theta_S} \mathcal{L}_{\text{logit}}(x_i)\|^2$ controls smoothness.
- **Feature Covariance Regularization**: $\mathcal{R}_{\text{cov}} = \sum_m \|\Sigma_S^m - \Sigma_T^{M(m)}\|^2$ ensures feature similarity.
- **Mutual Information Maximization**: $\mathcal{R}_{\text{MI}} = -I(\phi_S(X); Y)$ retains task-relevant information.

2.2 Knowledge Transfer Mechanisms

Logit-Based Transfer. The student is trained to match the softened teacher logits:

$$\mathcal{L}_{\text{logit}} = D_{\text{KL}}(p_T \parallel p_S) = \sum_i p_T(i) \log \frac{p_T(i)}{p_S(i)}. \tag{4}$$

where $p_T(i)$ and $p_S(i)$ are teacher and student probability distributions.

Feature-Based Transfer. Hidden layer representations are aligned via:

$$\mathcal{L}_{\text{feat}} = \sum_m \|P_m \phi_T^{M(m)}(x) - \phi_S^m(x)\|^2. \tag{5}$$

where P_m is a projection matrix mapping teacher features to student features.

Attention-Based Transfer. The student aligns its attention patterns with the teacher:

$$\mathcal{L}_{\text{att}} = \sum_m \|W_m A_T^{M(m)} - A_S^m\|^2. \tag{6}$$

where W_m is a projection matrix aligning teacher attention heads to student heads.

2.3 Student Model Initialization via Projection

To accelerate learning, we initialize the student weights using the teacher:

$$W_S^m = P^* W_T^{M(m)}, \tag{7}$$

where P^* is the optimal projection matrix minimizing $\sum_m \|W_S^m - P W_T^{M(m)}\|^2$.

For embedding layers and attention heads, we apply dimensionality reduction:

$$E_S = P E_T, \quad W_S^m = P W_T^{M(m)}. \tag{8}$$

using learned P matrices for structured adaptation.

2.4 Theoretical Analysis of Distillation

Stability vs. Plasticity Tradeoff. The student must retain knowledge from the teacher (stability) while adapting to new tasks (plasticity). This tradeoff is characterized as:

$$\rho_t(S) \cdot \|\nabla_{\theta_S} \mathcal{L}\| \leq C\sqrt{I(\phi_T(X); \phi_S(X))}. \tag{9}$$

where $\rho_t(S)$ is the stability margin.

Convergence Conditions. Distillation converges when:

$$\frac{d}{dt} I(\phi_T(X); \phi_S(X)) \leq -\frac{1}{\tau^2} \|\nabla_{\theta_S} \mathcal{L}\|^2. \tag{10}$$

where τ is the temperature parameter controlling gradient magnitude.

Sample Complexity. The number of samples needed for the student to approximate the teacher within error ϵ is bounded by:

$$N \geq \frac{C}{\epsilon^2} \left(\frac{\dim(\mathcal{H}_T) - \dim(\mathcal{H}_S)}{\text{Capacity}(S)} \right). \tag{11}$$

Larger models require more data, and knowledge loss increases with capacity mismatch.

2.5 Generalization Limits and Post-Training

Generalization Bound. The student's performance is bounded by:

$$\text{Error}_S \geq \text{Error}_T + \mathcal{D}(\phi_T, \phi_S). \tag{12}$$

where $\mathcal{D}(\phi_T, \phi_S)$ represents the representation gap.

Post-Training and Fine-Tuning. To refine knowledge, we post-train the student with:

$$\mathcal{L}_{\text{post-train}} = \mathcal{L}_{\text{new-task}} + \lambda \mathcal{D}_{\text{KL}}(p_T \parallel p_S). \tag{13}$$

where λ balances task adaptation and teacher retention.

2.6 Summary of Theoretical Insights

Distillation effectiveness is influenced by:

- **Initialization**: Proper projection of teacher weights improves convergence.
- **Loss Function Composition**: Balancing task loss, logit loss, and feature losses.
- **Sample Complexity**: A smaller student requires more data.
- **Generalization**: The student is fundamentally constrained by the teacher's error and architecture mismatch.
- **Post-Training**: Retains useful knowledge while adapting to new tasks.

3 Algorithmic Frameworks and Implementation

Building upon the theoretical foundations, we now focus on the **algorithmic aspects** of knowledge distillation. The key goal is to develop efficient methods for transferring knowledge while ensuring faster convergence and minimal computational overhead. This section explores both **training-based** and **non-training-based** techniques for constructing student models.

3.1 Designing an Efficient Distillation Process

Knowledge distillation can be approached in multiple ways, depending on the constraints of **computational efficiency, convergence speed, memory limitations, and architectural differences**. The algorithmic process follows a structured pipeline:

- **Model Selection**: Choosing a teacher model that possesses strong generalization while balancing computational constraints.
- **Knowledge Extraction**: Defining the type of knowledge transfer, whether through logits, feature maps, or attention distributions.
- **Student Model Construction**: Creating an efficient student either through non-training-based techniques (weight copying, quantization, pruning) or through training-based distillation methods.
- **Optimization Strategy**: Selecting training optimizations like adaptive temperature scheduling, progressive layer freezing, or curriculum learning.

3.2 Non-Training-Based Approaches for Student Construction

While standard distillation requires training, several methods allow constructing students directly from teacher models without expensive backpropagation.

Algorithm 1 Weight Copying and Quantization

```
Require: Trained teacher model $T(\theta_T)$, quantization level $Q$
Ensure: Compressed student model $S(\theta_S)$
1: $\theta_S \leftarrow \theta_T$                ▷ Copy teacher weights
2: $\theta_S \leftarrow Q(\theta_S)$             ▷ Apply quantization (e.g., INT8)
3: return $S(\theta_S)$
```

Weight copying and quantization [4] involve directly transferring teacher weights to the student while reducing precision (e.g., FP32 to INT8). This provides significant memory and inference efficiency gains without retraining.

Another alternative is **layer dropping and selective pruning** [16], which involves removing redundant layers while retaining model performance.

Algorithm 2 Layer Dropping Strategy

```
Require: Trained teacher model $T$, layer sensitivity function $S(l)$
Ensure: Reduced-depth student model $S$
1: for each layer $l$ in $T$ do
2:     if $S(l) < \tau$ then                 ▷ Threshold-based removal
3:         Drop layer $l$ from student model
4:     end if
5: end for
6: return $S$
```

Layer dropping systematically removes layers that contribute little to final predictions, allowing faster inference while maintaining performance [10].

3.3 Training-Based Approaches for Distillation

If training is permitted, several techniques can improve efficiency.

1. Adaptive Temperature Scheduling. Instead of using a fixed temperature τ, we dynamically adjust it:

- Begin with **high temperatures** (smoother knowledge transfer).
- Gradually decrease τ to refine student predictions.
- Helps stabilize early training and fine-tune later stages [17].

2. Progressive Layer-Freezing. Instead of backpropagating through the entire student model, we can **freeze lower layers early** [15]:

Algorithm 3 Progressive Layer-Freezing

```
Require: Initialized student model $S$, epochs $E$
Ensure: Trained student model $S^*$
1: for $t = 1$ to $E$ do
2:     Train upper layers $S_t$ while freezing lower layers
3:     Freeze progressively more layers as $t$ increases
4: end for
5: return $S^*$
```

This strategy reduces training time while ensuring stable feature representations.

3.4 Convergence Guarantees and Final Considerations

To ensure a student model retains sufficient knowledge from the teacher, we analyze key guarantees:

1. Generalization Bounds. The student model's error is bounded by [9]:

$$\mathbb{E}[\mathcal{L}(S_{\theta_S^*})] \leq \mathcal{L}(T) + \mathcal{O}\left(\sqrt{\frac{\log(1/\delta)}{n}}\right), \tag{14}$$

where n is the number of training samples and δ is the confidence parameter.

2. Convergence Rate. The knowledge transfer process converges when [12]:

$$\frac{d}{dt} I(\phi_T(X); \phi_S(X)) \leq -\frac{1}{\tau^2} \|\nabla_{\theta_S} \mathcal{L}\|^2. \tag{15}$$

This suggests adjusting temperature τ for optimal learning speed.

3.5 Final Considerations on Efficient Distillation

Choosing the right approach depends on constraints:

- **If no training is allowed** → Use weight copying, quantization, or pruning [4].
- **If training is feasible** → Use adaptive temperature, progressive freezing, or curriculum distillation [17].
- **For large models** → Matrix factorization or autoencoder-based compression can be effective [2].

4 Applications in Modern Architectures

The theoretical frameworks of knowledge distillation find distinct implementations in various domains. Two of the most prominent areas where distillation has demonstrated remarkable success are **large language models (LLMs)** and **vision models**. Each of these architectures presents unique challenges that require specialized distillation techniques.

4.1 Large Language Models

With the emergence of billion-parameter-scale transformer models, efficient distillation has become crucial for making these models deployable in real-world settings, particularly for edge devices and latency-sensitive applications. The primary challenge in LLM distillation lies in preserving the reasoning and generation capabilities of large models while significantly reducing computational complexity.

4.1.1 Transformer-Specific Distillation

For transformer-based architectures, the multi-head attention mechanism and deep feed-forward layers pose key challenges for distillation. Unlike CNNs, which rely on spatial feature hierarchies, transformers depend on contextual relationships across long sequences, making the transfer of knowledge more complex.

The distillation objective in transformers can be decomposed into three key components:

$$\mathcal{L}_{\text{transformer}} = \alpha\mathcal{L}_{\text{attn}} + \beta\mathcal{L}_{\text{hidden}} + \gamma\mathcal{L}_{\text{pred}} \tag{16}$$

where:

- $\mathcal{L}_{\text{attn}}$ ensures attention alignment between the teacher and student.
- $\mathcal{L}_{\text{hidden}}$ focuses on preserving intermediate layer representations.
- $\mathcal{L}_{\text{pred}}$ aligns the final outputs to retain the overall task performance [6].

4.1.2 Hidden State and Attention Transfer

Transformer-based distillation can be further refined through hidden state alignment and attention map matching, ensuring that the student model captures hierarchical knowledge at multiple layers.

Attention Alignment: For each transformer layer l, the attention transfer loss is formulated as:

$$\mathcal{L}_{\text{attn}}^{l} = \sum_{h=1}^{H} \left\| \frac{A_T^{l,h}}{\|A_T^{l,h}\|_F} - \frac{A_S^{l,h}}{\|A_S^{l,h}\|_F} \right\|^2 \tag{17}$$

where:

- $A_T^{l,h}$ and $A_S^{l,h}$ are attention weights of the teacher and student for head h.
- H represents the number of attention heads.
- Frobenius normalization ensures scale invariance.

Hidden State Transfer: The hidden state transfer aligns intermediate layer activations between teacher and student models:

$$\mathcal{L}_{\text{hidden}}^{l} = \left\| W_l H_T^l - H_S^l \right\|^2 \tag{18}$$

where W_l is a learnable projection matrix mapping the teacher's hidden states into the student's space [11].

4.1.3 Prompt-Based Knowledge Distillation

A more recent direction in LLM distillation involves using prompt-based alignment. Instead of transferring knowledge from raw logits, prompt-based distillation ensures that task-specific responses remain consistent across teacher and student models:

$$\mathcal{L}_{\text{prompt}} = \mathbb{E}_{p\sim\mathcal{P}}[\text{KL}(T(x|p)\|S(x|p))] \tag{19}$$

where:

- $\mathcal{P}$ is a distribution over prompts,
- The expectation is taken over different prompt templates, ensuring diverse generalization [1].

This approach is particularly effective for few-shot and zero-shot learning settings, where the teacher model generalizes better due to its exposure to large-scale training data.

4.2 Vision Models

Knowledge distillation in vision models presents distinct challenges compared to LLMs. Unlike language models, which focus on sequential dependencies, vision models emphasize spatial feature extraction and hierarchical representation learning.

4.2.1 Feature Pyramid Transfer for Object Detection

Object detection models, such as Faster R-CNN and YOLO, rely on feature pyramids for multi-scale object detection. Standard distillation methods often fail for these models because of their deep hierarchical structure. Feature-based distillation can be improved through feature pyramid transfer, formulated as:

$$\mathcal{L}_{\text{FPN}} = \sum_{l=1}^{L} w_l \|\phi_T^l(x) - \phi_S^l(x)\|^2 + \lambda \mathcal{L}_{\text{loc}} \tag{20}$$

where:

- $\phi_T^l(x)$ and $\phi_S^l(x)$ are feature maps at pyramid level l.
- $\mathcal{L}_{\text{loc}}$ ensures consistency in localization loss [8].

4.2.2 Cross-Modal Alignment for Vision-Language Models

In models like CLIP and ViLT, vision-language knowledge transfer is challenging due to the different nature of embeddings. Knowledge distillation must ensure that cross-modal relationships remain intact:

$$\mathcal{L}_{\text{cross}} = \mathcal{L}_{\text{vision}} + \mathcal{L}_{\text{text}} + \lambda \mathcal{L}_{\text{align}} \tag{21}$$

where:

$$\mathcal{L}_{\text{align}} = \left\| \frac{V_T^\top T_T}{\|V_T\|\|T_T\|} - \frac{V_S^\top T_S}{\|V_S\|\|T_S\|} \right\|^2 \tag{22}$$

Here, V and T denote visual and textual embeddings, ensuring consistent knowledge transfer across modalities [7].

4.2.3 Edge Deployment Optimization

For real-time applications, model compression techniques such as quantization-aware distillation are used:

$$\mathcal{L}_{\text{edge}} = \mathcal{L}_{\text{KD}}(Q(S), T) + \lambda \mathcal{R}(Q) \tag{23}$$

where:

- $Q(S)$ is the quantized student model.
- $\mathcal{R}(Q)$ imposes constraints to optimize memory efficiency [14].

5 Conclusion

Knowledge distillation has proven to be a versatile and effective strategy for compressing and improving deep learning models across different architectures. Through a combination of logit-based, feature-based, and attention-based distillation, it enables models to retain high performance with reduced computational costs.

As models continue to scale, distillation will remain essential for making AI accessible across diverse platforms, from cloud-based inference to real-time edge deployments. Future directions include:

- Enhancing theoretical understanding of multi-modal and federated distillation.
- Developing architecture-agnostic distillation techniques.
- Adapting distillation for dynamic and continuously learning systems.

Knowledge distillation will continue to be a **critical enabler** of practical AI, ensuring that powerful models remain efficient, accessible, and scalable.

References

[1] Yusheng Chen, Xiaobo Zhong, Tao Gui, Qi Zhang, and Yongfeng Yang. Promptbert: Improving bert with prompt-based learning and distillation. *arXiv preprint arXiv:2109.13166*, 2021.

[2] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014.

[3] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *International Conference on Machine Learning*, pages 1607–1616, 2018.

[4] Song Han, Jeff Pool, John Tran, and William Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[6] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. In *ICLR*, 2020.

[7] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *ICML*, 2021.

[8] Quanquan Li, Shengying Jin, and Junjie Yan. Feature pyramid distillation for object detection. In *ECCV*, 2020.

[9] Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5142–5151. PMLR, 2019.

[10] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[11] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. Mobilebert: a compact task-agnostic bert for resource-limited devices. In *ACL*, 2020.

[12] Jiaxi Tang, Rakshith Shivanna, Zhe Zhao, Dong Lin, Anima Singh, Ed H Chi, and Sagar Jain. Understanding the role of temperature in knowledge distillation. *arXiv preprint arXiv:2006.12987*, 2020.

[13] Haotao Wang, Zhenyu Xie, and Zhangyang Wu. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[14] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Compressed learning for neural networks: A study on quantization and knowledge distillation. In *AAAI*, 2020.

[15] Hengshu Xu, Jie Luo, Li Zou, Zheng Wen, and Jian Yin. Knowledge transfer in multi-task learning: A survey. *arXiv preprint arXiv:2007.08116*, 2020.

[16] Jiahui Yu, Linjie Yang, Ningning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *International Conference on Learning Representations*, 2018.

[17] Lianwei Yuan, Francis EH Wang, Gaofeng Meng, Lin Li, and Jianbing Shen. Revisiting knowledge distillation via label smoothing regularization. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3903–3911, 2020.

[18] Amir R Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.