Knowledge Distillation

Prof. Marco Cococcioni,
Gabriele Giudici,
Gabriele Billi Ciani

A.Y. 2024/2025

GitHub Repository

The following repository contains the current slides and the notebooks which accompany the presentation.

`https://github.com/gabrielebilliciani/knowledge-distillation`

Outline of Contents (1 / 2)

Outline of Contents (2 / 2)

Outline

The Challenge of Large Models

- State-of-the-art models are often very large and computationally expensive.
  - Examples: Large ensembles, deep neural networks with many parameters.
- These “cumbersome models” (Hinton et al., 2015) excel at learning from data.
- However, they pose challenges for deployment:
  - Latency constraints (real-time applications).
  - Computational resource limits (mobile devices, edge computing).
  - Memory footprint.
- **Question:** How can we leverage their power in resource-constrained environments?

What is Knowledge Distillation?

- A technique to transfer “knowledge” from a large, cumbersome *teacher* model (or an ensemble) to a smaller, more efficient *student* model.
- Goal: Student mimics teacher’s performance with significantly reduced complexity.
- **Early Idea (Buciluǎ et al. (2006) “Model Compression”):**
  - Demonstrated that a smaller, simpler model could be trained to *mimic* the output behaviour of a large, complex model ensemble.
  - This effectively transferred the “learned intelligence” of the ensemble to the compact model.
  - This laid the groundwork for a new approach to model compression.
- **Refinement (Hinton et al. (2015)):** Focused on how neural networks can learn more effectively from richer information provided by a teacher.

Motivation

Why use Knowledge Distillation (KD)?

- Reduce model size and computational cost for deployment on edge devices, mobile phones, etc.
- Improve inference speed (reduce latency).
- Achieve better student performance: The rich knowledge from the teacher acts as a powerful regulariser, often **helping the student generalise better than if it were trained from scratch on the same data.**
- Potentially accelerate training: By providing a more informative and less noisy learning signal than sparse labels, the teacher can guide the student to converge more quickly.
- Maintain (or approach) the teacher's high accuracy with a compact architecture.

Outline

Teacher-Student Framework

The core idea of KD involves two key components:

Teacher Network

- A high-capacity model (e.g. deep NN, ensemble) that has already been trained and performs well on the task.
- Its “knowledge” about the task (learned function, internal representations, or output behaviour) is to be transferred.

Student Network

- A more compact model (e.g. fewer parameters, shallower architecture).
- Learns from the teacher by trying to mimic certain aspects of its behaviour or internal state, in addition to (or instead of) learning directly from the original data labels.

The specific mechanisms of knowledge transfer differ between tasks like classification and regression (more about this later).

Outline

History of Knowledge Distillation (1 / 2)

Based on an overview from Wikipedia (Wikipedia contributors, 2025a)

The core idea of transferring knowledge from larger to smaller models has several historical precedents:

- **Early Model Simplification:** Initial efforts focused on *pruning* or reducing the complexity of already trained neural networks to make them more efficient.
- **Teacher-Student Concepts:** The notion of one network (a "student") learning from another (a "teacher") began to appear in the 1990s, exploring how networks could guide each other's learning.
- **Mimicking Complex Model Outputs:** A significant step involved training smaller models to directly replicate the outputs (specifically, the pre-softmax logits) of larger, high-performing model ensembles. This demonstrated that the "behaviour" of a complex system could be captured by a simpler one.

History of Knowledge Distillation (2 / 2)

Based on an overview from Wikipedia (Wikipedia contributors, 2025a)

- **Formalisation of "Knowledge Distillation":** The term and modern concept were prominently articulated by Hinton et al. (2015), particularly for **classification**.

Due to its clear formulation of core principles for classification, we'll begin the detailed exploration with this domain, primarily drawing from Hinton et al. (2015).

Outline

Teacher-Student in Classification

For the following discussion on classification, we primarily consider a **multi-class, mutually exclusive classification scenario**.

In this classification scenario, the framework is typically realised as:

Teacher Network

- Its final layer often uses a **softmax function** to output a *full probability distribution* over the classes.
- This ***entire distribution*** (not just the teacher's single highest-probability class – hard prediction), **serves as the "soft targets"** for the student.

Student Network

- Also typically uses a softmax output layer.
- Learns by trying to match the teacher's **soft targets** (the full probability distribution).
- Optionally, it can also learn from the true hard labels (ground truth classes).

Dark Knowledge

From Hinton et al. (2015):

> *"[...] the trained model assigns probabilities to all of the incorrect answers and even when these probabilities are very small, some of them are much larger than others.* ***The relative probabilities of incorrect answers tell us a lot about how the cumbersome model tends to generalize.*** *An image of a BMW, for example, may only have a very small chance of being mistaken for a garbage truck, but that mistake is still many times more probable than mistaking it for a carrot."*
>
> *"[...]* ***a small model trained to generalize in the same way [as a large one] will typically do much better on test data than a small model that is trained in the normal way*** *on the same training set as was used to train [the large one]."*

Soft Outputs and Temperature Scaling

- Soft outputs reveal *dark knowledge*: subtle relations between outputs. Exploiting this is crucial for helping the student learn a nuanced similarity structure between classes.
    - E.g. in MNIST classification "one version of a 2 may be given a probability of 10^{-6} of being a 3 and 10^{-9} of being a 7, whereas for another version it may be the other way around. This is valuable information that defines a rich similarity structure over the data (i.e. it says which 2's look like 3's and which look like 7s)" (Hinton et al., 2015).
- Problem: in the previous example, even though we have the described valuable information, "it has very little influence on the cross-entropy cost function during the transfer stage because **the probabilities are so close to zero**."
- Solution: Hinton et al. (2015) proposed to use a **temperature parameter** T in the softmax function to control the sharpness of the output distribution. With this proposal, "distillation" was born.

Temperature Scaling

- In Hinton et al. (2015) teacher logits z_i are converted into probabilities with temperature $T > 1$:

$$p_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}.$$

"[...] raise the temperature of the final softmax until the cumbersome model produces a suitably soft set of targets. ***We then use the same high temperature when training the small model to match these soft targets****."*

- $T > 1$ smooths the probability distribution.
- A higher T "softens" the distribution, increasing the magnitude of non-target class probabilities.
- This allows the student to learn more effectively from the "dark knowledge" encoded in the relative probabilities of incorrect classes.

Logits vs. Softmax Outputs

- In their earlier approach to the problem, Buciluǎ et al. (2006) had the student model match the **logits** (the inputs to the teacher's final softmax layer) directly.
- This effectively bypasses the issue of tiny probabilities, as logits are on a different scale. However, logits are raw, uncalibrated internal scores, which can be on an unstable scale and are less directly informative about class relationships than probability distributions.
- The approach from Hinton et al. (2015) still leverages the information in the full output distribution but allows finer control over how "soft" the targets are.
- Hinton et al. (2015) also showed that matching logits can be seen as a **special case** of their distillation method **under high temperature** and zero-meaned logits.

Outline

Distillation Signals

- So far, our primary focus in classification has been on matching the teacher's (softened) output probability distribution.
- However, the "knowledge" a teacher model possesses can be captured and transferred in various forms, often called "distillation signals."
- These signals guide the student's learning process, aiming to make it learn not just **what** to predict, but also **how** the teacher arrives at its predictions or represents information.
- These methods are often applicable beyond classification, including in regression tasks and LLMs (more on these later).

Signal 1: Soft Target Matching (Hinton's Distillation)

Primarily for Classification

- As discussed extensively for classification (Hinton et al., 2015), the student learns to mimic the teacher's softened class probabilities obtained using temperature scaling.
- The loss function is typically the Kullback-Leibler (KL) divergence (Wikipedia contributors, 2025b) between the teacher's ($p^{(t)}$) and student's ($p^{(s)}$) probability distributions, both computed with temperature T:

$$L_{\text{soft}} = D_{\text{KL}}(p^{(t)} || p^{(s)}) = \sum_i p_i^{(t)} \log \left(\frac{p_i^{(t)}}{p_i^{(s)}} \right)$$

  (p_i denotes the probability for class i; the sum is over all classes i).
- This is a cornerstone of modern KD, especially in classification.
- In their earlier approach using logits, Buciluă et al. (2006) performed soft matching using Mean Squared Error (MSE) instead of KL divergence.

Signal 2: Direct Output Matching

Primarily for Predictions (Regression)

- The student can be trained to directly match the teacher's outputs, especially meaningful in numerical predictions (regression tasks).
- Loss: Typically Mean Squared Error (MSE).

$$L_{\text{output\_match}} = \|\mathbf{y}^{(t)} - \mathbf{y}^{(s)}\|_2^2$$

  where $\mathbf{y}$ are final predictions.
- This term can be referred to as L_{HP} (High-Precision). It encourages the student to match teacher's high-precision outputs.
- **Why this is powerful for Regression:**
  - The term "High-Precision" reflects that the teacher's output is a **refined, "denoised" estimate** of the gound truth.
  - It provides a more robust and regularised learning target for the student than a single, potentially noisy, ground-truth label.
- Why this is weak for Classification:
  - Compared to the true labels, the teacher's hard prediction is **redundant if correct, and harmful if wrong**.

Signal 3: Intermediate Feature Matching (1 / 2)

Learning Intermediate Representations

- Instead of (or in addition to) matching outputs (whether hard or soft ones), the student can learn to mimic the teacher's activations at intermediate layers (feature maps).
- Let $f^{(t)}(x)$ and $f^{(s)}(x)$ be the feature representations from chosen layers in the teacher and student, respectively.
- Loss: Often MSE between these feature maps.

  $$L_{\text{feature}} = \|f^{(t)}(x) - f_w(f^{(s)}(x))\|_2^2$$

  An optional transformation f_w (e.g. a linear regressor) might be needed if the dimensions or shapes of the teacher's hint layer and student's guided layer differ.
- This encourages the student to learn similar internal data representations as the teacher, effectively providing "hints" during training.

Signal 3: Intermediate Feature Matching (2 / 2)

Learning Intermediate Representations

- For example, this was explored in FitNets by Romero et al. (2015), where “hints” from a teacher’s middle layers were used to guide the training of a student.
    - In their work, the student was often deeper and thinner than the teacher. The rationale was that guiding these deeper layers helps in their challenging optimisation.
    - However, the principle of using intermediate feature maps as supervisory signals is general and can be valuable even when distilling to shallower students, helping them learn more structured representations from specific teacher layers.
- More recently, Ji et al. (2021) advanced this by using an attention mechanism to dynamically learn which teacher and student feature maps are most suitable to match, creating a more flexible and powerful form of feature distillation.
- On feature distillation, see also Liu et al. (2023)

Signal 4: Input Gradient Matching (1 / 4)

Matching Sensitivity to Inputs (Both for Classification and Regression)

The student can be guided to learn how the teacher's loss changes with respect to the inputs. This captures the teacher's input sensitivity or the local geometry of the learned function.

The most straightforward way to do this, is to use

1. Direct Gradient Matching:

- The student directly minimises a distance (e.g. L2 norm) between its input gradients and the teacher's.

$$L_{\mathsf{GR}} = \left\| \nabla_x \mathcal{L}^{(t)} - \nabla_x \mathcal{L}^{(s)} \right\|_2^2$$

- For a fair comparison, both $\mathcal{L}^{(t)}$ and $\mathcal{L}^{(s)}$ refer to the **same standard task loss** (e.g. Cross-Entropy in classification), calculated using the ground-truth labels. The resulting L_{GR} term is then added to the student's overall loss function.

Signal 4: Input Gradient Matching (2 / 4)

Matching Sensitivity to Inputs (Both for Classification and Regression)

- This encourages the student's output to react to input changes similarly to the teacher, guiding it along the optimal learning directions.
- **This requires an additional backward pass through both models for each training batch**, solely to obtain these input gradients without updating weights.
- This approach is often referred to as Sobolev Training (Czarnecki et al., 2017).
- This might be linked to vanishing gradients, as the input gradient is the most affected by the vanishing gradient problem, especially in deep networks.

2. Adversarial Gradient Matching

Signal 4: Input Gradient Matching (3 / 4)

Matching Sensitivity to Inputs (Both for Classification and Regression)

- An alternative, often more robust approach, aims to match the *distributions* of teacher and student gradients, as proposed by Sinha et al. (2018) in their “Gradient Adversarial Training” (GREAT) framework.
- An auxiliary discriminator network D is trained to distinguish between teacher gradients $\nabla_x \mathcal{L}^{(t)}$ and student gradients $\nabla_x \mathcal{L}^{(s)}$.
- The student network is then trained to produce gradients that “fool” this discriminator, effectively making its gradient distribution statistically indistinguishable from the teacher’s.
- The student’s gradient matching loss component (to be minimised by the student) becomes something like:

$$L_{\text{gradient\_adv}} = -\log D(\nabla_x \mathcal{L}^{(s)}(\mathbf{x}))$$

Signal 4: Input Gradient Matching (4 / 4)

Matching Sensitivity to Inputs (Both for Classification and Regression)

- The student's goal is to maximise the discriminator's output D (0 or 1). Minimising $-\log(D)$ achieves this: the loss is low when D is high (the student succeeds) and high when D is low (the student fails).
- Sinha et al. (2018) argue this can be more effective than direct regression-based (e.g. L2) matching, especially for complex, high-dimensional gradient tensors.

In the following, when L_{gradient} is mentioned, it can refer to either of these approaches.

Expanding on Gradient Matching: Other Signals (1 / 3)

Beyond Input Gradients

The principle of gradient matching can be generalised to capture other forms of knowledge from the teacher.

1. **Gradients w.r.t. Intermediate Layers**

- Instead of the input, we match the gradient of the loss with respect to an **intermediate feature map** ($\nabla_f \mathcal{L}$).
- **The Intuition:** This distils the teacher's knowledge about *how its internal representations should be corrected* to reduce the final loss, providing a deeper form of guidance.
- **Example: CNNs**
  - Think of a CNN as two parts: a *feature extractor* (convolutional base) and a *classifier* (dense layers).
  - The intermediate feature map f is the output of the feature extractor.
  - By matching $\nabla_f \mathcal{L}$, we teach the student's feature extractor how the teacher's would adjust its features to improve the final prediction.

Expanding on Gradient Matching: Other Signals (2 / 3)

Beyond Input Gradients

- **In Practice:**
  - Requires choosing a *semantically corresponding* layer in the student (e.g. the penultimate layer).
  - If dimensions differ, a simple **transformation layer** (e.g. a linear regressor) can be used to align the gradient tensors.

2. Gradients w.r.t. Model Weights (An Ill-Posed Goal)

- **The Ideal Goal:**
  Ideally, we would distil the teacher's learning dynamic by matching the gradient of the loss with respect to its weights ($\nabla_{\text{weights}}\mathcal{L}$).
  - **Challenge:** This is ill-defined, as the student and teacher have different architectures and parameters, making a direct gradient comparison impossible.

Expanding on Gradient Matching: Other Signals (3 / 3)

Beyond Input Gradients

- **Practical Proxies for the Gradient's Path:** Since matching weight gradients directly is not feasible, research has found clever proxies to transfer the knowledge encoded in the teacher's optimisation path.
    - An interesting idea (Jin et al., 2019): **Mimicking the Training Route.** The student learns from a *curriculum* of teacher checkpoints saved at different stages of its training. This forces the student to follow the teacher's optimisation "route" from an early, general model to a final, specialised one.
    - An alternative idea by (Tung and Mori, 2019): **Preserving Feature Space Structure.** Instead of the optimisation path, this method distills the final *geometry* of the teacher's learned feature space. It uses a similarity-preserving loss that ensures if two data samples are represented similarly by the teacher (i.e. are "close" in its feature space), they must also be represented similarly by the student. This effectively transfers the teacher's understanding of semantic relationships between inputs.

Further Developments

Beyond the methods discussed, the literature explores various other approaches to knowledge transfer. These include alternative distillation schemes. To mention a few, there are online learning, where teacher and student are trained concurrently (like in the previously mentioned "curriculum learning"), and self-distillation, where a network effectively becomes its own teacher.

Another aspect is the fact that, as one can see from some of the previously presented signals, studies can be done to determine the best architecture for the student given that of the teacher. For instance, one could think of a student where each layer semantically corresponds to n teacher layers, thus creating a correspondence between the two models. For example, Sanh et al. (2020) (DistilBERT, more on it later) have the number of student's layers "reduced by a factor of 2" compared to the teacher.

A recent literature survey on the topic of KD was made by Mansourian et al. (2025).

Outline

The Combined Distillation Loss (1 / 2)

The student's learning is typically guided by a composite loss function. This function strategically combines the standard task-specific loss (learning from ground truth data) with one or more distillation terms (learning from the teacher's "wisdom"):

$$L_{\text{total}} = \alpha L_{\text{task}} + \sum_k \lambda_k L_{\text{distill},k}$$

- L_{task}: The student's primary objective related to the ground truth labels (e.g. Cross-Entropy for classification, MSE for regression). This can also be referred to as L_{hard}.
- $L_{\text{distill},k}$: One or more distillation signals, chosen to transfer specific aspects of the teacher's knowledge.
- α, λ_k: Hyperparameters that balance the influence of each term. Their careful tuning is crucial for successful distillation and is often determined empirically based on validation performance.

The Combined Distillation Loss (2 / 2)

The art of knowledge distillation often lies in selecting the right combination and weighting of these terms for the specific task, model architectures, and the desired characteristics to be transferred from the teacher.

Using a variety of complementary signals provides several key advantages:

- **Richer Guidance & Better Features:** Different signals provide complementary information, guiding the student to develop more robust and informative feature representations than hard labels alone could.
- **Improved Generalisation & Robustness:** The teacher's signals act as a powerful form of regularisation, helping the student to generalise better, reduce overfitting, and become more robust.
- **Smoother Optimisation:** The "soft" nature of the teacher's guidance can create a smoother optimisation landscape for the student, often leading to faster and more stable convergence.

Distillation Setup for Regression

For regression tasks, a strategy is to combine the standard task loss with the teacher's refined predictions:

$$L_{\text{total}} = \alpha L_{\text{task}} + \beta L_{\text{HP}}$$

- $L_{\textbf{task}}$: The standard MSE loss between the student's predictions and the ground-truth "hard" targets.
- $L_{\textbf{HP}}$ **(High-Precision Loss)**: An MSE loss pushing the student's predictions ($\mathbf{y}^{(s)}$) to match the teacher's refined predictions ($\mathbf{y}^{(t)}$).
- α, β: Hyperparameters to balance the two learning objectives.

Notebook link:

`https://colab.research.google.com/github/gabrielebilliciani/knowledge-distillation/blob/main/Notebook1_KD_Regression.ipynb`

Classic Distillation for Classification

The foundational approach popularised by Hinton et al. (2015) combines the standard task loss with the soft target signal:

$$L_{\text{total}} = (1 - \lambda) L_{\text{task}} + \lambda L_{\text{soft}}$$

- $L_{\textbf{task}}$: The standard Cross-Entropy loss (ground-truth labels).
- $L_{\textbf{soft}}$: The distillation loss (teacher's temperature-softened "soft" targets).
- λ: A hyperparameter to balance learning from the two objectives.

Notebooks links:

`https://colab.research.google.com/github/gabrielebilliciani/knowledge-distillation/blob/main/Notebook2_KD_Classification_MLPs.ipynb`
`https://colab.research.google.com/github/gabrielebilliciani/knowledge-distillation/blob/main/Notebook3_KD_Classification_CNNs.ipynb`

Distillation in Large Language Models (LLMs)

- Distilling knowledge into smaller LLMs often leverages techniques from classification, as next-token prediction is essentially a very large classification task over the vocabulary.
- **Example – DistilBERT (Sanh et al. (2020)):**
  - This work successfully distilled BERT into a significantly smaller model (DistilBERT) by training it to mimic BERT's output distribution (using temperature-scaled softmax) and internal hidden states during the pre-training phase.
  - This resulted in a model that retains a high percentage of BERT's performance while being substantially faster and lighter, demonstrating the power of distillation for complex generative and understanding tasks.
  - They directly used the idea from Hinton et al. (2015).
- Distillation in LLMs has gone much further, but this is beyond the scope of the current presentation.

Outline

References (1 / 3)

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 535–541, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933395. doi: 10.1145/1150402.1150464. URL `https://doi.org/10.1145/1150402.1150464`.

Wojciech Marian Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Świrszcz, and Razvan Pascanu. Sobolev training for neural networks, 2017. URL `https://arxiv.org/abs/1706.04859`.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL `https://arxiv.org/abs/1503.02531`.

Mingi Ji, Byeongho Heo, and Sungrae Park. Show, attend and distill:knowledge distillation via attention-based feature matching, 2021. URL `https://arxiv.org/abs/2102.02973`.

References (2 / 3)

Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization, 2019. URL `https://arxiv.org/abs/1904.09149`.

Dongyang Liu, Meina Kan, Shiguang Shan, and Xilin Chen. Function-consistent feature distillation, 2023. URL `https://arxiv.org/abs/2304.11832`.

Amir M. Mansourian, Rozhan Ahmadi, Masoud Ghafouri, Amir Mohammad Babaei, Elaheh Badali Golezani, Zeynab Yasamani Ghamchi, Vida Ramezanian, Alireza Taherian, Kimia Dinashi, Amirali Miri, and Shohreh Kasaei. A comprehensive survey on knowledge distillation, 2025. URL `https://arxiv.org/abs/2503.12067`.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets, 2015. URL `https://arxiv.org/abs/1412.6550`.

References (3 / 3)

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

Ayan Sinha, Zhao Chen, Vijay Badrinarayanan, and Andrew Rabinovich. Gradient adversarial training of neural networks, 2018. URL https://arxiv.org/abs/1806.08028.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation, 2019. URL https://arxiv.org/abs/1907.09682.

Wikipedia contributors. Knowledge distillation – Wikipedia, the free encyclopedia, 2025a. URL https://en.wikipedia.org/wiki/Knowledge_distillation. [Online; accessed 2025-05-23].

Wikipedia contributors. Kullback–leibler divergence – Wikipedia, the free encyclopedia, 2025b. URL https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence. [Online; accessed 2025-07-10].