GPT-2 Distillation based on Wasserstein Distance

Changhong Zhang
The George Washington University
Washington, DC, USA
edwinzhang@gwu.edu

Abstract

Knowledge distillation is a key technique for compressing large language models for resource-constrained environments. This study explores a novel hidden state alignment strategy based on Wasserstein Distance (WD) for distilling knowledge from GPT-2 (Teacher) to DistilGPT2 (Student). Leveraging the Kantorovich-Rubinstein duality, we train auxiliary Critic networks with a Gradient Penalty (GP) term to approximate the W_1 distance, aiming to match the hidden state distributions between corresponding Teacher and Student layers. We compare this method against a baseline using Mean Squared Error (MSE) for direct vector matching (or against the early training stages of the WD method). Experiments were conducted on the OpenWebText dataset using a composite training objective combining hidden state alignment loss, output-layer KL divergence, and standard cross-entropy loss.

Our results demonstrate the significant efficacy of the WD-based distillation approach. The Student model achieved a substantial reduction in validation perplexity (over 99%), indicating effective knowledge transfer. Analysis of layer-wise loss and t-SNE visualizations reveals that WD successfully aligns the intermediate hidden states of the Student with those of the Teacher, exhibiting superior structural similarity compared to the MSE baseline (or early-stage WD). However, we observed anomalous behavior in the final hidden layer alignment, potentially influenced by output-level prediction objectives. Overall, Wasserstein Distance shows considerable promise as a hidden layer distillation strategy for enhancing model performance while preserving internal representational consistency.

Keywords

GPT-2, Knowledge Distillation, Wasserstein Distance

ACM Reference Format:
Changhong Zhang. 2018. GPT-2 Distillation based on Wasserstein Distance. In . ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Knowledge distillation has become a key technique for compressing large language models while retaining most of their predictive capabilities. Originally introduced by Hinton et al. [4], the technique involves training a smaller "student" model to mimic the behavior of a larger "teacher" model. This process aims to create lightweight models that are more suitable for resource-constrained environments, such as mobile devices or real-time applications. As language models have scaled significantly in recent years, including architectures like BERT [3] and GPT-2 [7], distillation has gained increasing importance in balancing performance with computational efficiency.

Several notable efforts have advanced the field of model distillation. Sanh et al. [8] proposed DistilBERT, a compressed version of BERT that preserves approximately 95% of the original model's performance while reducing its size and inference time by nearly 40%. Similarly, DistilGPT-2 [5] adapts the principles of distillation to the generative GPT-2 architecture. More recent research has explored structured distillation techniques, such as layer-wise and task-aware strategies [6]. Other studies, such as Sun et al. [9], emphasized the importance of intermediate layer matching during the distillation process to improve student model performance. In addition, task-specific compression frameworks like those discussed in Xu et al. [10] demonstrated how domain-specific knowledge could be preserved during compression. Recent papers have also introduced multi-objective distillation, knowledge projection, and hybrid loss functions to further improve the training dynamics between teacher and student models [1]. Meanwhile, lightweight pretraining techniques [2] offer promising directions to accelerate model deployment across diverse tasks.

In this paper, we build upon this line of research by examining the distillation of GPT-2 models, discussing experimental setups, training strategies, and evaluation methods that enhance the effectiveness of model compression while preserving linguistic and generative abilities.

2 Methodology

This section outlines the core knowledge distillation framework and the specific loss functions compared in this study.

2.1 Knowledge Distillation Framework

We employ a knowledge distillation (KD) strategy where a Student model is trained to mimic a larger Teacher model. The training objective minimizes a composite loss:

$$\mathcal{L}_{\text{Total}} = \alpha \cdot \mathcal{L}_{\text{Hard}} + \beta \cdot \mathcal{L}_{\text{KL}} + \gamma \cdot \mathcal{L}_{\text{Hidden}}$$

This combines standard language modeling loss ($\mathcal{L}_{\text{Hard}}$), output distribution matching via KL divergence ($\mathcal{L}_{\text{KL}}$), and intermediate hidden state alignment ($\mathcal{L}_{\text{Hidden}}$). This work focuses on comparing two distinct approaches for calculating $\mathcal{L}_{\text{Hidden}}$.

Conference'17, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

2.2 KL Divergence Loss ($\mathcal{L}_{\text{KL}}$)

We align the output logits distributions using KL divergence with temperature scaling (T) :

$$\mathcal{L}_{\text{KL}} = T^2 \cdot \text{KL}\left(\text{softmax}\left(z_T/T\right) \| \log_{\text{softmax}\left(z_S/T\right)}\right)$$

2.3 Hidden State Alignment ($\mathcal{L}_{\text{Hidden}}$)

We investigate two methods to align Student hidden states $\left(h_{S_i}\right)$ with corresponding Teacher hidden states $\left(h_{T_j}\right)$ at matched layers (Student layer i vs Teacher layer $j = 2i$):

2.3.1 *Method A: Mean Squared Error (MSE) Alignment.*

Directly minimizes the L2 distance between hidden state vectors:

$$\mathcal{L}_{\text{MSE},i} = \text{MSE}\left(h_{S_i}, h_{T_{2i}}\right) = \mathbb{E}\left[\|h_{S_i} - h_{T_{2i}}\|_2^2\right]$$

with $\mathcal{L}_{\text{Hidden}} = \sum_i \mathcal{L}_{\text{MSE },i}$.

2.3.2 *Method B: Wasserstein Distance (WD) Alignment.*

Matches the distributions of hidden states using an approximation of the 1 -Wasserstein distance (W_1) based on the Kantorovich-Rubinstein duality. This involves training auxiliary Critic networks (c_i) for each layer pair.

- Critic Training: The Critic c_i maximizes the score difference between Teacher and Student states, constrained by a Gradient Penalty (GP) term:
  $\mathcal{L}_{\text{Critic },i} = -\left(\mathbb{E}\left[c_i\left(h_{T_{2i}}\right)\right] - \mathbb{E}\left[c_i\left(h_{S_i}\right)\right]\right) + \lambda_{\text{GP}} \cdot \text{GP}_i$
  $$\text{GP}_i = \mathbb{E}_{h_{\text{interp}}}\left[\left(\left\|\nabla_{h_{\text{interp}}} c_i\left(h_{\text{interp}}\right)\right\|_2 - 1\right)^2\right]$$
- Student WD Loss: The Student minimizes the negative score assigned by the Critic to its own states:
  $\mathcal{L}_{\text{WD},i} = -\mathbb{E}\left[c_i\left(h_{S_i}\right)\right]$
  with $\mathcal{L}_{\text{Hidden}} = \sum_i \mathcal{L}_{\text{WD},i}$.

3 Experiment

This section details the experimental setup used to evaluate the effectiveness of Wasserstein Distance (WD) based hidden state alignment against a Mean Squared Error (MSE) baseline for knowledge distillation from GPT-2 to DistilGPT2.

3.1 Models and Initialization

- **Teacher Model**: We utilized the pre-trained GPT-2 (base, 12 layers) model accessed via the Hugging Face transformers library. Its parameters remained frozen during student training.
- **Student Model**: The student architecture is DistilGPT2 (6 layers), also based on the Hugging Face implementation. The model was initialized with pre-trained DistilGPT2 weights, but the parameters of its first Transformer block (Layer 0) were re-initialized randomly before distillation commenced. The student model has approximately 81.9 million parameters.
- **Critic Network Architecture (for WD Method)**: For the WD-based distillation, separate Critic networks were employed for each of the 6 matched hidden layers. Each Critic is a Multi-Layer Perceptron (MLP) implemented using `torch.nn.Module`, consisting of four linear layers with ReLU activations between them, mapping the hidden state dimension (768) down to a scalar output (768 -> 512 -> 256 -> 128 -> 1), as defined in `critic.py`.

3.2 Dataset

Data Source: The experiments exclusively utilized the OpenWebTextCorpus, a large-scale linguistic corpus created by Aaron Gokaslan and Vanya Cohen (Skylion007) by scraping URLs shared on Reddit. It serves as an open-source replication of the dataset used to train GPT-2. Further details can be found **here**.

3.3 Training Configuration

The knowledge distillation process was conducted using the PyTorch framework, leveraging distributed training techniques for efficiency. Training was performed using PyTorch DistributedDataParallel (DDP) across 3 GPUs, employing the NCCL backend for inter-GPU communication. Parameter optimization was managed as follows:

- Student Model: Optimized using AdamW with a learning rate of 5e-6 and weight decay set to 1e-2.
- Critic Networks (for WD Method): Optimized using RMSprop with a learning rate of 1e-4.
- Learning Rate Scheduling: A CosineAnnealingLR scheduler was applied to all optimizers to adjust learning rates during training.

The composite training loss, which integrates hard label cross-entropy, KL divergence between output distributions ($T = 2$), and hidden state alignment losses, was carefully balanced using weighting coefficients (e.g., $\alpha = 0.3, \beta = 0.4, \gamma = 0.3$ for WD; $\alpha = 0.4, \beta = 0.3, \gamma = 0.3$ for MSE). Specific hyperparameters for the Wasserstein Distance alignment method included:

- Gradient Penalty Coefficient $\left(\lambda_{\text{GP}}\right)$: 10
- Critic Updates per Student Update: 5

Data was processed in batches during training, with the following parameters:

- Per-GPU Batch Size: 48
- Gradient Accumulation Steps: 1
- Maximum Sequence Length: 512 tokens (sequences were truncated or padded as necessary).

The training procedure was planned for 3 epochs over the training data. Logs and performance plots presented cover approximately the first 32,000 training steps within the initial epoch, during which model checkpoints were saved periodically (e.g., every 5000 steps). To mitigate GPU memory constraints associated with large model training, gradient checkpointing was enabled.

3.4 Evaluation Metrics

The evaluation metrics we used in this project are:

- Primary Metric: Perplexity (PPL) calculated periodically on the OpenWebText validation set.

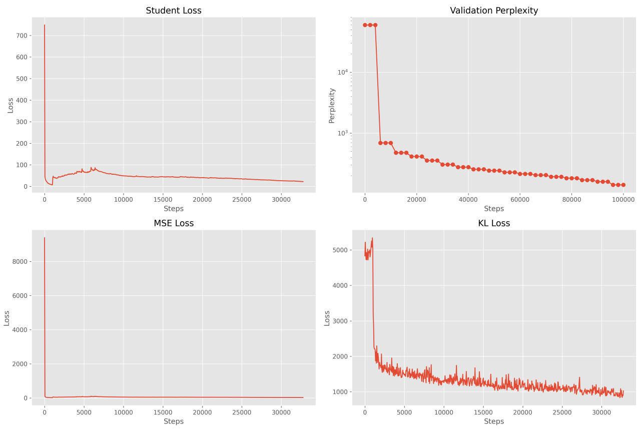

Figure 1: Convergence Summary

- Training Dynamics: Monitored via plots of:
  - Total Student Loss (reflecting WD loss)
  - KL Divergence Loss
  - MSE Loss (if run as baseline)
  - Layer-wise Student Loss (WD loss per layer)
- Qualitative Analysis: t-SNE visualization comparing hidden state alignment between Student and Teacher layers at different training points (early vs. later stage for WD method; WD vs. MSE if applicable).

4 Results

4.1 Training Dynamics and Performance (WD Method)

The training progression using the WD hidden layer alignment method is summarized in Figure 1.

- Validation Perplexity (Top Right): A significant and rapid decrease in perplexity on the validation set was observed. Starting from an extremely high initial value (59k), the perplexity dropped sharply within the first few thousand steps and continued to decrease steadily, reaching approximately 141.5 by the end of the observed training period (32k steps). This indicates very effective learning of language modeling capabilities.
- Student Loss (Total WD Loss) (Top Left): This loss, representing the primary objective of aligning hidden state distributions via WD, exhibited a similar pattern: a steep initial drop from 748 followed by gradual convergence towards 22.2. This demonstrates the effectiveness of the critic-based optimization in guiding the student's internal representations.
- KL Loss (Bottom Right): The KL divergence between teacher and student output logits also decreased substantially from its initial peak (5000), stabilizing around 1000-1500, indicating successful mimicry of the teacher's output distribution.

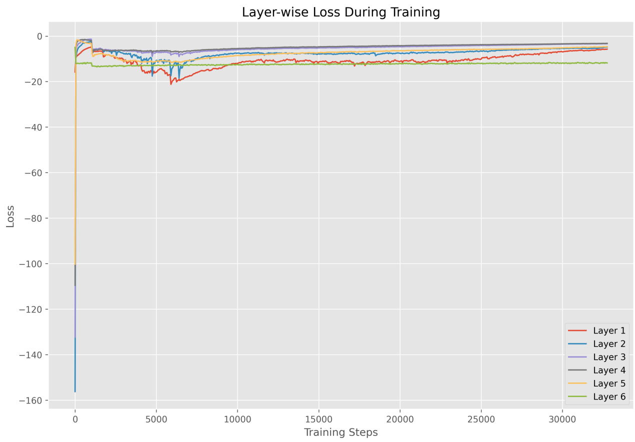

Figure 2: Layer-wise Loss During Training

4.2 Layer-wise Alignment Dynamics (WD Method)

Figure 2 illustrates the evolution of the layer-wise student loss (negative approximated W_1 distance) for each of the six matched layers during WD training.

- Intermediate Layers (L1-L5): These layers show a clear trend of improvement. The loss values increase (become less negative), indicating that the student's hidden state distributions for these layers are progressively becoming more similar to those of the corresponding teacher layers (L2, L4, L6, L8, L10). Layers 2, 3, and 4 show particularly strong convergence towards better alignment after the initial phase.
- Final Layer (L6): In contrast, the loss for the final student transformer layer (L6, matched with Teacher L12) initially improves but then tends to worsen slightly or plateau at a less aligned state compared to intermediate layers. This suggests that the objective of matching the final output distribution (driven by KL loss and potentially hard labels) might override the objective of strictly matching the teacher's penultimate hidden state distribution at this layer.

4.3 Qualitative Hidden State Alignment (t-SNE Visualizations)

We use t-SNE to visualize the structural similarity between hidden state representations of the student (DistilGPT2, red dots) and teacher (GPT-2, blue dots) for a specific input text at corresponding layers.

Figure 3: Wasserstein Distance (WD) Method Results: This visualization, corresponding to the well-trained WD model, shows strong alignment across most layers. Red and blue points representing the same token are closely clustered, and the overall geometric structure of the point clouds for both models is highly similar, especially in layers 1 through 5 (vs. teacher layers 2 through 10). This visually confirms that the WD method effectively captures and transfers the teacher's representational geometry.

Figure 4: Mean Squared Error (MSE) Method Results: Compared to the WD results, the alignment achieved using the MSE method (or

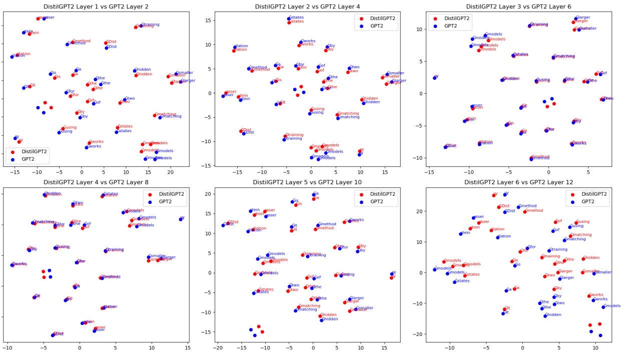

Figure 3: Wasserstein Distance-based Experiment Samples Visualization(t-SNE)

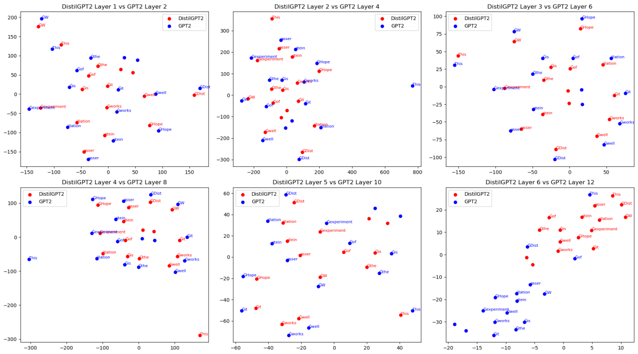

Figure 4: MSE-based Experiment Samples Visualization(t-SNE)

WD in its early stages) appears less refined. While some proximity exists, there is noticeably more separation between corresponding red and blue points, and the overall cluster structures are less congruent, particularly in the deeper layers. This suggests MSE, while enforcing direct numerical similarity, might be less effective at capturing the overall distributional and structural properties compared to WD after sufficient training.

In Summary, both the quantitative loss curves and the qualitative t-SNE visualizations indicate that the Wasserstein Distance based hidden state distillation method successfully trained the DistilGPT2 student model, leading to a dramatic reduction in perplexity. The WD loss effectively aligned intermediate layer representations, visually outperforming the alignment seen in the MSE/early WD comparison figure, although the final student layer showed some divergence, likely due to the influence of output-level objectives.

5 Conclusion

This study aimed to explore and compare different strategies for aligning intermediate hidden states during knowledge distillation, specifically transferring knowledge from a large GPT-2 model to a smaller DistilGPT2 model. We primarily compared the effectiveness of hidden state distribution alignment based on Wasserstein Distance (WD) against a baseline using Mean Squared Error (MSE) for direct matching (or against the early training stages of the WD method).

Our experimental results clearly demonstrate that employing WD for hidden layer alignment, combined with KL divergence loss at the output layer, leads to a significant improvement in the student model's performance. Validation perplexity was drastically reduced from a very high initial value to approximately 141.5 (a reduction exceeding 99%), validating the efficacy of the distillation framework. The convergence of the Student Loss (WD Loss) and KL Loss during training further supports this finding.

In-depth analysis of hidden state alignment, through layer-wise WD loss curves and t-SNE visualizations, revealed more nuanced insights:

(1) WD Achieves Strong Intermediate Layer Alignment: For most intermediate layers of the student model (L1-L5), the WD loss showed consistent improvement, and t-SNE visualizations confirmed that, after sufficient training, the student's hidden states became structurally highly similar to the corresponding teacher layers, visually outperforming the alignment observed with the MSE method (or early-stage WD). This suggests WD effectively captures and transfers the distributional characteristics of the teacher's internal representations.

(2) Anomalous Final Layer Behavior: In contrast to the intermediate layers, the WD loss for the student's final transformer layer (L6) did not consistently improve and even worsened relative to the teacher's corresponding layer (L12). This indicates a divergence in hidden state distributions at the topmost layer, possibly because the optimization objectives related to the final output logits (driven by KL and hard losses) take precedence over strictly mimicking the teacher's penultimate hidden state structure.

In conclusion, this research validates Wasserstein Distance as a potent and promising method for hidden state alignment in language model knowledge distillation. It not only yields substantial overall performance gains (measured by PPL) but also achieves superior structural alignment of internal representations, particularly in intermediate layers, compared to MSE (or its own early training stages). Future work could investigate the reasons behind the final layer's divergence, potentially by adjusting loss weights or layer-matching strategies, and evaluate the generalization capabilities of WD-distilled models on a broader range of downstream tasks.

References

[1] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D. Lawrence, and Zhenwen Dai. 2019. Variational Information Distillation for Knowledge Transfer. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 9163–9171. https://openaccess.thecvf.com/content_CVPR_2019/html/Ahn_Variational_Information_Distillation_for_Knowledge_Transfer_CVPR_2019_paper.html

[2] Viktoriia Chekalina, Georgii Novikov, Julia Gusak, Ivan Oseledets, and Alexander Panchenko. 2023. Efficient GPT Model Pre-training using Tensor Train Matrix Representation. (2023). arXiv:2306.02697 [cs.AI] https://arxiv.org/abs/2306.02697

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL* (2019). https://arxiv.org/abs/1810.04805

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* (2015). https://arxiv.org/abs/1503.02531

[5] Hugging Face. 2020. distilgpt2. https://huggingface.co/distilbert/distilgpt2.

[6] Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. Less is More: Task-aware Layer-wise Distillation for Language Model Compression. In *International Conference on Machine Learning (ICML)*. https://proceedings.mlr.press/v202/liang23j/liang23j.pdf

[7] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Technical Report* (2019). https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[8] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019). https://arxiv.org/abs/1910.01108

[9] Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient Knowledge Distillation for BERT Model Compression. *arXiv preprint arXiv:1908.09355* (2019). https://arxiv.org/abs/1908.09355

[10] Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2021. BERT-of-Theseus: Compressing BERT by Progressive Module Replacing. *arXiv preprint arXiv:2002.02925* (2021). https://arxiv.org/abs/2002.02925