Learning Interpretable Hierarchies: Recursive Distillation into Neural Decision Tree Routed Mixture of Experts

Nicholas Cantrell*
CyberGolem AI Research †

Abstract

Large transformer models have achieved remarkable success but largely operate as black boxes, limiting our understanding of their internal reasoning and hierarchical feature learning. This work proposes a novel architecture, the Neural Decision Tree routed Mixture of Experts (NDT-MoE), designed to enhance interpretability and explicitly model hierarchical processing. Our approach utilizes differentiable decision trees (inspired by NDF/NODE) as interpretable routing mechanisms within an MoE framework, guiding inputs to specialized expert subnetworks. To overcome training complexities, we employ knowledge distillation, transferring knowledge from a pre-trained large transformer (teacher) to the NDT-MoE student. We further explore stacking these NDT-MoE layers, potentially enabling recursive analysis of routing decisions. We design experiments on tabular benchmarks known to favor tree-based methods, standard interpretability benchmarks, and complex synthetic datasets generated from differential equations. Evaluation focuses on task performance compared to baseline transformers and MoE models, alongside qualitative and quantitative analysis of routing interpretability, expert specialization, and hierarchical feature representation. [Placeholder: Our results indicate that NDT-MoE achieves competitive performance on tabular and synthetic tasks while providing tangible insights into the model's decision-making process through routing analysis.] We believe this architecture offers a promising direction towards building more transparent and structured large-scale models.

1 Introduction

The phenomenal capabilities of large language models (LLMs) and other transformer-based architectures are undeniable (Vaswani et al., 2017; Brown et al., 2020). However, their monolithic, densely connected nature renders them opaque "black boxes." Understanding *how* these models arrive at their predictions and *whether* they learn meaningful hierarchical representations of data remains a significant challenge. This lack of interpretability hinders trust, debugging, and the ability to verify alignment with desired principles.

Current approaches to understanding these models often rely on *post-hoc* analysis techniques like attention visualization or feature attribution (Lundberg and Lee, 2017; Ribeiro et al., 2016), which provide limited glimpses into complex internal dynamics. Model compression via knowledge distillation (KD) (Hinton et al., 2015) primarily focuses on efficiency, typically creating smaller black-box students. While Mixture of Experts (MoE) architectures (Shazeer et al., 2017; Fedus et al., 2022) improve efficiency through sparse computation, their standard gating networks offer little inherent interpretability.

We propose a novel approach that integrates interpretability directly into the architecture: the Neural Decision Tree routed Mixture of Experts (NDT-MoE). Inspired by the success of

*For partnership inquiries contact lab@cybergolem.ai

†Nicholas Cantrell generated this text in part with Gemini 2.5 Pro Preview 03-25, Google's large-scale language-generation model. Upon generating draft language, the author reviewed, edited, and revised the language to their own liking and takes ultimate responsibility for the content of this publication.

differentiable decision trees/forests (Kontschieder et al., 2015) and Neural Oblivious Decision Ensembles (NODE) (**?**) in providing interpretable yet powerful models (especially for tabular data), we employ them as the *routing mechanism* within an MoE layer. Instead of a dense gate, an NDT routes inputs to different expert subnetworks based on learned, potentially sparse and hierarchical, criteria.

Training such an architecture from scratch is challenging. We leverage knowledge distillation from a pre-trained large transformer model to guide the NDT-MoE student, transferring rich semantic knowledge and stabilizing training. Furthermore, by stacking NDT-MoE layers, possibly with DenseNet-style connections (Huang et al., 2017) inspired by NODE, we aim to facilitate the learning of hierarchical features, where routing decisions at deeper layers operate on representations refined by earlier expert processing. The potential for *recursive interpretation* arises – analyzing routing decisions at layer l based on the outputs and routing from layers $< l$.

Our main contributions are:

- A novel NDT-MoE architecture using differentiable decision trees for interpretable routing in MoE models.
- A training methodology using knowledge distillation from large transformers to train the NDT-MoE student.
- A design for analyzing the interpretability of routing decisions and the potential emergence of hierarchical expert specialization.
- An experimental framework comparing NDT-MoE against relevant baselines on diverse datasets, including synthetic data with known generative processes. [Placeholder: Highlight key experimental finding here.]

This paper proceeds as follows: Section 2 reviews related work. Section 3 details the NDT-MoE architecture and training. Section 4 outlines the experimental design. Section 5 presents [Placeholder: results] and analysis. Section 6 discusses limitations and future work, and Section 7 concludes.

2 Related Work

2.1 Mixture of Experts (MoE)

MoE models, originating from (Jacobs et al., 1991), aim to increase model capacity without proportional computational cost by activating only a subset of "expert" subnetworks for each input. Modern implementations (Shazeer et al., 2017; Fedus et al., 2022) employ sparse gating networks, often simple linear layers followed by softmax and a Top-K selection, to route tokens to experts (typically Feed-Forward Networks, FFNs). While effective for scaling, these dense gates lack transparency. Load balancing, ensuring experts receive comparable computational load, is a key challenge addressed by auxiliary losses (Shazeer et al., 2017). Alternative routing mechanisms like Expert Choice (Zhou et al., 2022) have also been proposed. Our work replaces the standard dense gate with an interpretable NDT structure.

2.2 Differentiable Decision Trees and Forests

Classical decision trees are highly interpretable but traditionally not trained end-to-end within deep learning frameworks. Recent work has focused on creating differentiable versions. Deep Neural Decision Forests (NDF) (Kontschieder et al., 2015) introduced differentiable routing functions and leaf predictions, enabling end-to-end training. Neural Oblivious Decision Ensembles (NODE) (**?**) generalized ensembles of oblivious decision trees (decision tables) with differentiable feature selection and splitting, showing strong performance on tabular data. Techniques

like the Gumbel-Softmax/Concrete distribution (Jang et al., 2017; Maddison et al., 2017) or entmax (Peters et al., 2019) provide crucial mechanisms for allowing gradient flow through discrete choices or learning sparse selections, which are essential for differentiable tree splits and feature selection. We leverage these concepts to build our routing mechanism.

2.3 Knowledge Distillation (KD)

KD (Hinton et al., 2015) is a widely used technique for model compression, where a smaller "student" model is trained to mimic the outputs (logits) of a larger, pre-trained "teacher" model. Variations involve matching intermediate hidden states (Romero et al., 2015) or attention patterns (Zagoruyko and Komodakis, 2017). While typically used to create efficient black-box students, we employ KD primarily to guide the training of our structurally different and potentially more interpretable NDT-MoE student, transferring knowledge from a powerful transformer teacher.

2.4 Interpretability and Explainable AI (XAI)

The field of XAI seeks to make complex models understandable. Many techniques are *post-hoc*, such as analyzing attention weights (Jain and Wiegreffe, 2019) (often debated for interpretability), using proxy models (LIME, (Ribeiro et al., 2016)), or computing feature attributions (SHAP, (Lundberg and Lee, 2017)). While useful, these methods provide external explanations rather than revealing the model's inherent logic. Our work pursues *intrinsic interpretability* by designing an architecture whose components (NDT routers) are intended to be directly interpretable.

2.5 Hierarchical Models

Learning hierarchical representations is considered crucial for complex tasks. Architectures like Convolutional Neural Networks (CNNs) implicitly learn feature hierarchies. More explicit attempts include Capsule Networks (Sabour et al., 2017) and various structured models. Our use of stacked NDT-MoE layers with potential DenseNet-style connections aims to explicitly encourage hierarchical processing through layered, specialized expert computations guided by interpretable routing.

3 Methodology: NDT-MoE Architecture and Training

3.1 Overall Framework

We utilize a teacher-student knowledge distillation (KD) setup. The teacher model T is a pre-trained large transformer [Placeholder: e.g., Qwen2-0.5B-Instruct]. The student model S is our NDT-MoE architecture, trained to mimic T while incorporating interpretable structures.

3.2 NDT Routing Mechanism

We adapt the NODE (**?**) architecture based on Oblivious Decision Trees (ODTs) for routing. An ODT uses the same feature and threshold for all nodes at a given depth level. Given an input representation $x \in \mathbb{R}^{d_{in}}$ (e.g., token embedding or previous layer output):

1. **Feature Selection:** For each tree level $i = 1...depth$, learnable feature selection weights $F_i \in \mathbb{R}^{d_{in}}$ are used with α-entmax (Peters et al., 2019) to compute a sparse combination of input features: $f_i(x) = \sum_{j=1}^{d_{in}} x_j \cdot \text{entmax}_\alpha(F_i)_j$.

2. **Splitting Decision:** The selected feature combination $f_i(x)$ is compared to a learnable threshold b_i using a smoothed, differentiable step function, also based on α-entmax. Let $c_i(x) = \text{entmax}_\alpha([(f_i(x) - b_i)/\tau_i, 0])_0$, where τ_i is a learnable temperature/scale parameter. $c_i(x)$ approximates $\mathbb{I}(f_i(x) > b_i)$.

3. **Path/Leaf Probability:** The probability of reaching a specific leaf node (representing a path through the tree) is calculated by multiplying the probabilities of taking the corresponding left/right splits at each level: $P(\text{leaf}_k|x) \propto \prod_{i=1}^{depth} [c_i(x) \text{ or } (1 - c_i(x))]$. This can be efficiently computed via outer products, forming a "choice tensor" $C(x) \in \mathbb{R}^{2^{depth}}$ (**?**).

4. **Expert Assignment:** The leaf probabilities $C(x)$ are mapped to probabilities over the N experts using a linear layer followed by softmax: $p(x) = \text{Softmax}(\text{Linear}(C(x))) \in \mathbb{R}^N$. Alternatively, Top-K selection can be applied based on $p(x)$.

The NDT router function $p = \text{NDT}(x)$ outputs the expert probability distribution.

3.3 MoE Layer Structure

The NDT-MoE layer operates as follows:

1. Compute expert probabilities $p = \text{NDT}(x)$.
2. Select the Top-K experts based on p, forming the active set S.
3. Compute outputs for active experts: $o_i = \text{Expert}_i(x)$ for $i \in S$. Experts (Expert_i) are typically FFNs or small transformer blocks.
4. Combine expert outputs, weighted by router probabilities: $y = \sum_{i \in S} p_i \cdot o_i$. (Normalization of p_i over S might be needed).
5. Add a load balancing loss $L_{\text{load}} = w_{\text{load}} \cdot \text{CV}(\text{Importance})^2$ to the total loss, where $\text{Importance}_i = \sum_{\text{batch}} p(x)_i$ and CV is the coefficient of variation (Shazeer et al., 2017). w_{load} is a hyperparameter.

3.4 Multi-Layer / Recursive Structure

We stack L NDT-MoE layers. Following NODE (**?**) and DenseNet (Huang et al., 2017), we use dense connections: the input to the router and experts at layer l is derived from the concatenation of the initial embedding h_0 and outputs of all preceding layers $h_1, ..., h_{l-1}$. The final model prediction is typically an aggregation (e.g., average or linear combination) of the outputs h_l from all layers. This structure allows deeper layers to build upon features computed by shallower layers and potentially enables recursive interpretation.

3.5 Knowledge Distillation Training

We employ a distillation setup similar to that in the provided '$\text{run}_g lue_f net_d istill.py$'.$Acustom$'$Trainer$'$subclass$(be the ground truth label. The total loss is:

$$L = \alpha L_{\text{hard}}(S(x), y_{\text{true}}) + (1 - \alpha) L_{\text{distill}}(S(x), T(x)) + \delta L_{\text{load}} + \gamma L_{\text{inter}} \tag{1}$$

where:

- L_{hard} is the standard task loss (e.g., CrossEntropy or MSE).
- L_{distill} is the KL divergence loss between softened predictions:

$$L_{\text{distill}} = T^2 \cdot \text{KLDiv}(\text{LogSoftmax}(S(x)/T), \text{Softmax}(T(x)/T)) \tag{2}$$

  with temperature T.

- L_{load} is the load balancing loss.
- L_{inter} (optional) is a loss (e.g., MSE) matching intermediate hidden states between teacher and student.
- α, δ, γ are loss weighting hyperparameters.

The teacher model T is kept frozen and in evaluation mode during training.

4 Experimental Design

4.1 Baselines

We compare our NDT-MoE student against:

- **Teacher Model:** The pre-trained transformer used for distillation ([Placeholder: e.g., Qwen2-0.5B-Instruct]).
- **Distilled Dense Transformer:** A standard transformer student with comparable parameters/FLOPs, trained with the same KD setup (excluding L_{load}).
- **Distilled Dense-Gate MoE:** An MoE student with a standard linear gate, trained with the same KD setup (including L_{load}) and comparable parameters/FLOPs.
- **Distilled FNet:** An FNet student (**?**) trained with the same KD setup, serving as a non-attention baseline.
- **[Optional] Original NODE/NDF:** Performance reported in original papers (**?**Kontschieder et al., 2015) or re-implemented on relevant tabular tasks.

4.2 Datasets

- **Tabular Benchmarks (from (?)):** Epsilon, YearPredictionMSE, Higgs, Microsoft LETOR, Yahoo LETOR, Click. (Metrics: Classification Error, MSE).
- **Interpretability Benchmarks:**
  - UCI Adult/Census Income: Binary classification (Accuracy, F1). Analyze routing based on demographic vs. financial features.
  - SST-2 (GLUE): Binary sentiment classification (Accuracy). Analyze routing based on syntactic vs. semantic cues.
- **Synthetic Complex Data:**
  - Lorenz Attractor: Regression (predict next state, MSE). Analyze expert specialization w.r.t. attractor lobes.
  - Lotka-Volterra (Predator-Prey): Regression (predict next state, MSE). Analyze expert specialization w.r.t. population cycles.

4.3 Evaluation Metrics

- **Performance:** Task-specific metrics (Accuracy, F1, MSE, etc.).
- **Interpretability:**
  - *Routing Feature Importance:* Permutation importance or integrated gradients on NDT router inputs.

- *Routing Path Visualization:* Trace decision paths for sample inputs.
- *Expert Utilization Analysis:* Frequency, variance, load balancing factor (max/avg load). Cluster inputs and analyze expert assignment purity per cluster. Correlate expert usage with known states in synthetic data.
- *Hierarchical Analysis:* Compare feature importance and expert patterns across different layers.

- **Computational:** Parameter Count, Training/Inference FLOPs (sparse), Inference Latency, Load Balancing Factor.

4.4 Implementation Details

- **Framework:** PyTorch, 'transformers', 'datasets', 'evaluate'.
- **Teacher Model:** [Placeholder: Specify Model, e.g., Qwen/Qwen2-0.5B-Instruct].
- **Student Architecture:** NDT type (NODE-style Oblivious), depth ([Placeholder: e.g., 6]), layers ([Placeholder: e.g., 4]), experts/layer ([Placeholder: e.g., 8]), expert type ([Placeholder: e.g., 2-layer FFN]), DenseNet connections (Yes).

- **Training:** Use 'DistillationTrainer'. Optimizer (AdamW), LR schedule ([Placeholder: Linear warmup + decay]), batch size ([Placeholder: e.g., 32]), KD temp T ([Placeholder: e.g., 2.0]), loss weights α ([Placeholder: e.g., 0.1]), δ ([Placeholder: e.g., 0.01]), γ (0 if not used). NDT α for entmax ([Placeholder: e.g., 1.5]). Use '$\text{load}_best_model_at_end = True$'$with appropriate$'$metric_for_best_model$'.**Hardwa

5 Results and Analysis

[TODO: This section will be populated with experimental results.]

5.1 Task Performance

[Placeholder: Present Table 1 comparing performance metrics across datasets and models. Discuss key performance differences and trade-offs.]

Table 1: Performance comparison on benchmark datasets. Values are [Placeholder: Accuracy/MSE/Error Rate]. Best student performance in bold.

Dataset	Teacher	Distilled Dense	Distilled FNet	Distilled Dense-MoE	NDT-M
Epsilon	[Placeholder: N/A]	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	[Placeh
YearPred	[Placeholder: N/A]	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	[Placeh
Higgs	[Placeholder: N/A]	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	[Placeh
Adult	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	[Placeh
SST-2	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	[Placeh
Lorenz	[Placeholder: N/A]	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	[Placeh
Lotka-Volterra	[Placeholder: N/A]	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	[Placeh

[Placeholder: Add notes about metrics used for each dataset.]

5.2 Interpretability Analysis

[Placeholder: Present qualitative and quantitative interpretability results.]

- **Routing Decisions:** [Placeholder: Show Figure 1 with routing path examples. Discuss feature importance results (Table 2). Did the routers use meaningful features?]

- **Expert Specialization:** [Placeholder: Show analysis of expert utilization (e.g., Figure 2). Did experts specialize? How did this relate to input clusters or states in synthetic data?]
- **Hierarchy:** [Placeholder: Discuss findings from analyzing multi-layer models. Was there evidence of hierarchical feature processing?]

[Placeholder: placeholder_routing_viz.png]

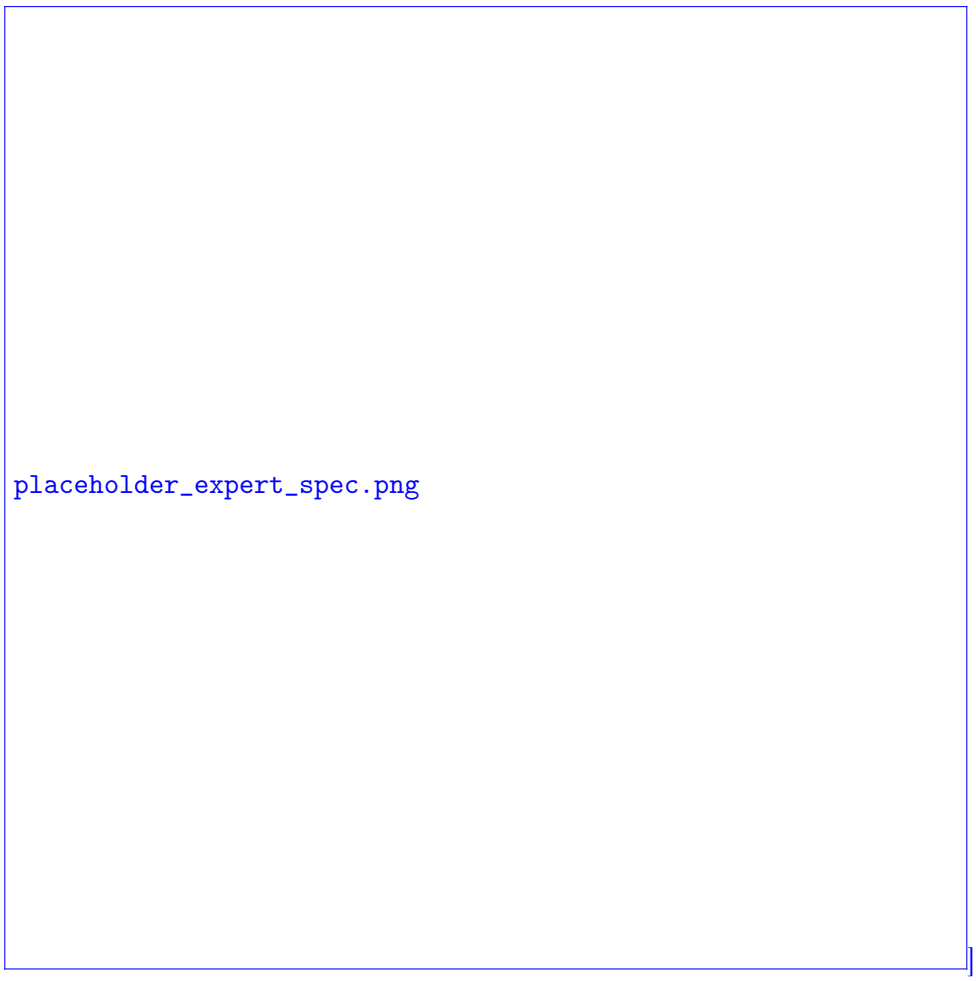
Figure 1: Examples of NDT routing paths for selected inputs from [Placeholder: Dataset Name]. Highlighted nodes indicate decisions based on specific features.

Table 2: Aggregated feature importance scores for NDT routers across datasets.

Feature/Feature Type	Aggregated Importance Score
[Placeholder: Feature 1]	[Placeholder: Score]
[Placeholder: Feature Type A]	[Placeholder: Score]

5.3 Computational Efficiency and Load Balancing

[Placeholder: Present Table 3 comparing computational metrics. Discuss the efficiency gains/losses and the effectiveness of load balancing.]



[Placeholder:]

Figure 2: Analysis of expert specialization. Left: Expert utilization frequencies. Right: Visualization of input clusters (colors) and dominant expert assignments within clusters.

Table 3: Computational comparison: Parameters, Inference FLOPs (Sparse), Latency, Load Balancing Factor.

Model	Params	FLOPs	Latency (ms)	Load Factor
Distilled Dense	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	N/A
Distilled FNet	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	N/A
Distilled Dense-MoE	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]
NDT-MoE (Ours)	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]	[Placeholder: Val]

5.4 Ablation Studies

[Placeholder: Present results from ablation studies (e.g., Table 4) investigating the impact of NDT depth, number of experts, loss components, etc.]

6 Discussion and Future Work

[Placeholder: Discuss the significance of the findings. Did NDT-MoE offer meaningful interpretability advantages compared to baselines? Was there a clear performance trade-off? Did the hierarchical structure show promise? Discuss the effectiveness of KD for this novel architecture.]

Table 4: Ablation study results on [Placeholder: Dataset Name] ([Placeholder: Metric]).

Configuration	Performance
Full Model	[Placeholder: Val]
- No Distillation ($\alpha = 1$)	[Placeholder: Val]
- No Load Balancing ($\delta = 0$)	[Placeholder: Val]
- NDT Depth = [Placeholder: D-1]	[Placeholder: Val]
- Experts = [Placeholder: N/2]	[Placeholder: Val]

Limitations: [Placeholder: Acknowledge challenges like training stability, complexity of interpreting deep NDTs, scalability concerns, the effectiveness of load balancing compared to dense gates, reliance on a good teacher model.]

Future Work: [Placeholder: Suggest extensions: exploring different NDT variants (non-oblivious), developing adaptive temperature/sparsity for routers, creating better visualization tools for hierarchical routing, applying to other domains (vision, reinforcement learning), investigating self-supervised pre-training for NDT-MoE.]

7 Conclusion

The opacity of large transformer models remains a significant barrier. We introduced the NDT-MoE architecture, leveraging differentiable decision trees for interpretable routing within a Mixture of Experts framework, trained via knowledge distillation. Our experiments on tabular, interpretability, and synthetic datasets [Placeholder: demonstrated [key finding, e.g., the model's potential to provide insights into decision-making paths and expert specialization while maintaining competitive performance].] The NDT-MoE offers a promising, albeit complex, direction towards building large-scale models that are not only powerful but also more transparent and structurally aligned with hierarchical reasoning.

Acknowledgements

[Placeholder: Acknowledge funding sources, helpful discussions, computational resources, etc.]

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 1877–1901, 2020.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research (JMLR)*, 23(120):1–39, 2022.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. arXiv preprint, also presented at NIPS 2014 Deep Learning Workshop.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017. doi: 10.1109/CVPR.2017.243.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Sarthak Jain and Sarah Wiegreffe. Attention is not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, 2019. doi: 10.18653/v1/D19-1002.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR 2017)*, 2017.

Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1467–1475, 2015. doi: 10.1109/ICCV.2015.172.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 4765–4774, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR 2017)*, 2017.

Ben Peters, Vlad Niculae, and André F. T. Martins. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1504–1519, 2019. doi: 10.18653/v1/P19-1146.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, 2016. doi: 10.1145/2939672.2939778.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations (ICLR 2015)*, 2015. ICLR 2015 Conference Track Paper.

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 3856–3866, 2017. URL https://papers.nips.cc/paper/2017/hash/2cad8fa43a1379a65491956458937474-Abstract.html.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR 2017)*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008, 2017. URL https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations (ICLR 2017)*, 2017. ICLR 2017 Conference Track Paper.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pages 7071–7084, 2022.

A Appendix

A.1 Hyperparameter Details

[Placeholder: Provide detailed tables of hyperparameters used for NDT-MoE and baseline training for each dataset.]

A.2 Dataset Details

[Placeholder: Provide further details on dataset preprocessing, generation specifics for synthetic data, and splits used.]

A.3 Additional Visualizations and Results

[Placeholder: Include more routing path examples, expert specialization plots across different datasets/layers, detailed feature importance breakdowns, learning curves, etc.]