Efficient world models with tree-structured sparsity

Reza Sedghi [*], Marcel Nieveler [‡*], Anand Subramoney [†], David Kappel [*], Robin Schiewer [*]

[*]CITEC, Bielefeld University, Bielefeld, Germany
{reza.sedghi, marcel.nieveler, david.kappel, robin.schiewer}@uni-bielefeld.de

[†]Department of Computer Science, Royal Holloway, University of London, Egham, United Kingdom
anand.subramoney@rhul.ac.uk

[‡]Chair of Highly-Parallel VLSI Systems and Neuro-Microelectronics, Dresden University of Technology, Germany

***Abstract*—Reliable robot autonomy requires internal models that capture semantic structure and temporal consequences of actions, yet such world models are often too computationally expensive for on-robot deployment. We study tree-structured Fast Feedforward (FFF) layers to sparsify transformer world models without substantially degrading predictive quality. FFF replaces dense feed-forward blocks by hard-routed hierarchical computation, thereby reducing the active computation per token. Building on prior evidence that FFF preserves performance in large transformer models under high sparsity, we evaluate a dense world-model baseline against FFF-based variants and analyse the trade-off between prediction quality and computational efficiency. Our results suggest that structured sparsity is not only a model compression technique, but also an enabling mechanism for executing richer semantic prediction and control directly on resource-constrained robots. This positions sparse world models as a promising ingredient for reliable robot autonomy.**

***Index Terms*—world models, sparsity, efficient computation**

I. Introduction

To interact with the world, autonomous robots require versatile and accurate decision-making and prediction capabilities. They must interpret scenes semantically, reason about object properties and relations, and anticipate how the world may evolve under candidate actions [1], [2]. These capabilities are especially important when downstream decision making depends on internal prediction, for example when a controller, planner, or policy improvement routine queries a world model before acting [3].

A practical limitation is that such predictive models must often run under tight latency, energy, and memory budgets [4], [5]. This is particularly acute for edge robotics, where repeated model rollouts may be needed inside a control loop, but onboard compute is far more constrained than in datacenter settings [6]. As a result, there is a growing need for world models that are not only expressive, but also computationally efficient enough to support frequent inference directly on the robot [3]. Transformers are a natural candidate architecture for semantic and temporal prediction because they can model long-range dependencies and can be trained on diverse sequential data. However, their computational footprint complicates deployment in resource-constrained settings. A major contributor to this cost is the feed-forward sublayer in each transformer layer, which accounts for a large share of both parameter count and computation.

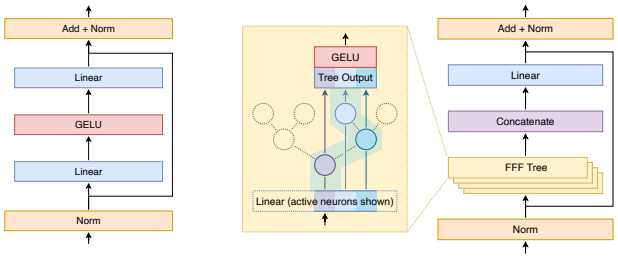
Recent work on Fast Feedforward (FFF) layers addresses this bottleneck by replacing the dense feed-forward computation with tree-structured conditional layers that induce exponentially increasing sparsity with tree depth [7]–[9]. FFF has been demonstrated to remain competitive in larger transformer settings, maintaining performance close to dense baselines even when fewer than 1% of feed-forward units are active per token [8], [9].

We argue that FFF is not only a promising language model component, but also a capable mechanism for sparsifying world models in robotics. Efficient semantic world models can expand what is feasible on resource-constrained robotic platforms. If each model evaluation becomes cheaper, a robot can either reduce inference latency or evaluate more candidate futures within a fixed control budget. This is directly relevant to reliable autonomy, because safety-critical behaviour often depends not only on the quality of a predictive model, but also on whether the model can be queried often enough, and quickly enough, to support planning and replanning in changing environments.

We instantiate our idea by comparing a standard 125M-parameter GPT-style transformer against a variant in which the dense feed-forward sublayers are replaced by FFF sublayers. Our setup follows a recent line of work that investigates the world modelling capabilities of large language and vision-language models [10]–[12]. We train and evaluate both models on a symbolic dataset generated from various gridworld scenarios spanning navigation, memory and simple puzzle tasks. This proof-of-concept setup isolates the effect of FFF while avoiding possible confounding factors introduced with more complex architectures. Our evaluations show that FFF can reach an order-of-magnitude speedup and 97% computational sparsity, while maintaining predictive quality.

II. Related Work

World models have become a central ingredient in model-based decision making because they enable agents to predict the consequences of candidate actions before execution. In reinforcement learning, latent world models have been shown to support efficient planning and policy improvement across diverse domains, as exemplified by Dreamer-style methods [13] and more control-centric approaches such as TD-MPC and TD-MPC2 [14], [15]. For robotics, this perspective is



(a) MLP block (b) Fast FeedForward (FFF) layer

Fig. 1: MLP layers and sparse replacements. **a)** The standard MLP feed-forward block. **b)** An FFF block that contains $P = 4$ parallel trees of depth $D = 2$, shown in yellow. Tree output is integrated via a concatenation of the individual sparse tree outputs and a final linear projection. Note that the concatenation and linear projection together implement the summation presented in Equations 3 and 4. A single path through the tree (green) is activated through hard routing decisions to produce the layer output.

particularly attractive because repeated internal rollouts can improve action selection under uncertainty, but it also makes inference efficiency a high priority concern when planning must run under tight compute budgets.

A separate line of work has therefore studied how to reduce the computational cost of transformer architectures. Sparse and conditional-computation methods are especially relevant here because, in large transformers, the feed-forward sublayers account for a substantial fraction of parameters and floating-point operations. Mixture-of-Experts (MoE) models activate only a subset of experts per token and have demonstrated that sparse activation can scale model capacity without proportional compute growth [16]. However, MoE layers typically rely on explicit routing networks and often require specialized systems support to translate algorithmic sparsity into practical speedups. This systems aspect has been studied systematically in work such as MegaBlocks [17].

III. Methods

The proposed approach builds on recent work showing that FFF-style layers can be scaled in transformer architectures while remaining competitive with dense baselines at high sparsity [8], [9]. FFF replaces the dense feed-forward sublayer with tree-structured conditional computation and evaluates only a small input-dependent subset of nodes during each forward pass. A schematic comparison of FFF with regular dense sublayers is shown in Fig. 1.

An FFF layer consists of P trainable, perfect binary trees of depth D, which are evaluated in parallel (see Fig. 1b). The computation proceeds sequentially through each tree root-to-leaf. Nodes are indexed by a triple (p, ℓ, s), where $p \in \{1, \dots, P\}$ denotes the tree, $\ell \in \{0, \dots, D\}$ the level, and $s \in \{0, \dots, 2^\ell - 1\}$ the position within the level. Each node (p, ℓ, s) is associated with two sets of trainable parameters:

- **Routing (linear_in):** a weight vector $w^{\text{in}}_{p,\ell,s} \in \mathbb{R}^{d_{\text{in}}}$ and bias $b^{\text{in}}_{p,\ell,s} \in \mathbb{R}$, producing a single scalar logit;
- **Output (linear_out):** a weight vector $w^{\text{out}}_{p,\ell,s} \in \mathbb{R}^{d_{\text{out}}}$.

For each input sample $x \in \mathbb{R}^{d_{\text{in}}}$ and each tree, traversal starts at the root $s_0 = 0$. For $\ell = 0, \dots, D$, exactly one node is evaluated:

$$z_{p,\ell}(x) = \left\langle x,\ w^{\text{in}}_{p,\ell,s_\ell} \right\rangle + b^{\text{in}}_{p,\ell,s_\ell} \in \mathbb{R}. \tag{1}$$

For $\ell = 0, \dots, D-1$, the next node index is updated via a hard routing decision:

$$s_{\ell+1} = 2s_\ell + \mathbf{1}[\text{sg}(z_{p,\ell}(x)) \geq 0]\,, \tag{2}$$

where $\mathbf{1}(\cdot)$ is the indicator function and $\text{sg}(\cdot)$ denotes the stop-gradient operator. Consequently, each tree requires a fixed number of D routing steps and the layer has $P \times D$ active nodes in total. All active nodes contribute directly to the output, i.e. nodes on levels $0, \dots, D-1$ have a double role of routing and output generation.

The output $y = \text{FFF}(x) \in \mathbb{R}^{d_{\text{out}}}$ of the layer is obtained by summing the contributions of the visited nodes:

$$y = b^{\text{out}} + \sum_{p=1}^{P} \sum_{\ell=0}^{D} \text{GELU}(z_{p,\ell}(x))\ w^{\text{out}}_{p,\ell,s_\ell}\,. \tag{3}$$

Due to the interplay between its routing mechanism and the GELU activation function, FFF produces a static, asymmetric utilization pattern in its trees over the course of training as shown in Fig. 2, left. Interestingly, we found that this pattern emerges independently of the input data distribution. Following the assumption that balanced network utilization is preferable, perfectly uniform utilization can be achieved (see Fig. 2, right) through a simple architectural modification where the GELU activation function is applied *after* integrating individual tree outputs:

$$\text{FFF}^{\text{post}}(x) = \text{GELU}\left(b^{\text{out}} + \sum_{p=1}^{P} \sum_{\ell=0}^{D} z_{p,\ell}(x)\ w^{\text{out}}_{p,\ell,s_\ell} \right). \tag{4}$$

As the emerging sparsity in FFF is static, an additional advantage is that it permits pruning inactive and rarely active paths after model convergence with minimal performance impact.

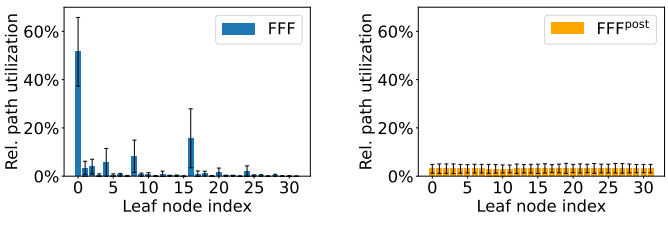


Fig. 2: **Utilization in a d=5 tree.** Path utilization as the normalized fraction of samples visiting each path for FFF (left) and FFF$^{\text{post}}$ (right). The x-axis denotes the leaf node index.

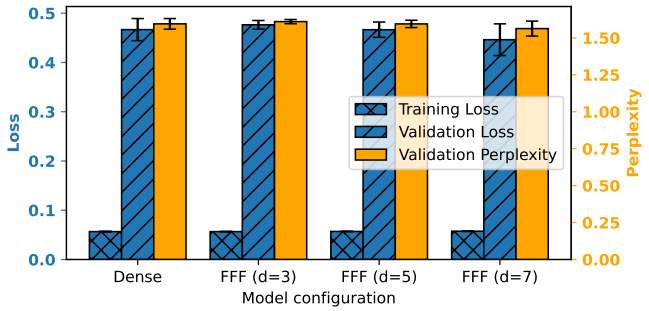


Fig. 3: Final loss and perplexity of three FFF variants and the dense baseline for the symbolic Minigrid dataset.

Thus, FFF effectively combines two separate principles of sparsity: dynamic sparsity through selective tree routing and static sparsity through pruning after training. This makes it a natural candidate for the implementation of transformer world models in constrained resource settings.

IV. Experiments and Results

We present three experiments in total. The first two experiments study the world-modelling performance of FFF and its sparsity characteristics. In these two experiments, we use an FFF implementation based on standard building blocks provided by the PyTorch [18] deep learning library. A third experiment evaluates practical speedups using a custom CUDA implementation, allowing us to assess whether the structured sparsity of FFF can be translated into actual wall-clock benefits. In all transformer-based experiments and for all FFF variants, the dense feed-forward sublayer in the transformer layers is replaced by our sparse version, whereas dense baselines remain unmodified vanilla transformers. Throughout all experiments, we train the models from scratch and do not make use of pretrained weights.

In the first experiment, we trained a dense 125M-parameter OPT [19] model and several sparse counterparts on a symbolic gridworld dataset. Although gridworlds are simplified domains, they provide a controlled testbed for action-conditioned sequential prediction, making them a useful case study for the world-modelling capabilities of FFF. The dataset was generated by sweeping over all standard configurations of the Minigrid environment [20] and collecting 1000 random trajectories in each of them. The trajectories were divided into a train and a validation set, whereas all basic elements (doors, keys, etc.) are ensured to be in the train set, which means the world models do not need to generalize to entirely new concepts. The depth and therefore relative parameter sparsity levels of the used FFF models are 75% for $d = 3$, 90% for $d = 5$ and 97% for $d = 7$. Note that we adjusted the number of parallel trees P accordingly for all FFF variants to match the overall amount of parameters between them and the dense baseline model. We trained each model configuration in this experiment using three distinct random seeds for parameter initialization and present averaged results with standard deviations. The training objective is next token prediction and we report raw loss values as well as validation set perplexities. Since the symbolic trajectories encode sequential observations and actions, next-token prediction closely aligns with learning the transition structure of the environment, i.e. world modelling in symbolic form.

We find the three FFF variants to perform at least on par with the dense baseline (see Fig. 3), positioning FFF as a viable alternative to dense layers in this world modelling setup. One possible explanation for FFF even slightly surpassing the dense baseline in some settings is that its input-dependent sparsity can act as an implicit regulariser by restricting the subset of parameters active for each token. Similar regularisation effects have been discussed more broadly for conditional computation and sparse subnetworks, although confirming this explanation would require more targeted ablations [21]–[23].

The second experiment explores the static pruning capabilities of FFF. It uses a simple 2-dimensional toy dataset in which samples from two different classes are arranged in a 16 by 16 chess grid pattern. Adjacent grid cells always contain samples from different classes. We applied a small model consisting of a 512-dimensional linear input projector, followed by an FFF layer with $d = 4$ and $P = 256$, followed by a 2-dimensional classification head. To illustrate the stability of FFF under various static sparsity regimes, we evaluated a pruned model on unseen data points at fixed intervals during training. Note that pruning was only applied temporarily for evaluation; no modification was made to the original model during training. As in the previous experiment, we did three independent runs with randomly initialized model parameters on the same data and report averaged results with standard deviations. Our analysis reveals that FFF maintains the same final performance even under high static sparsity levels of up

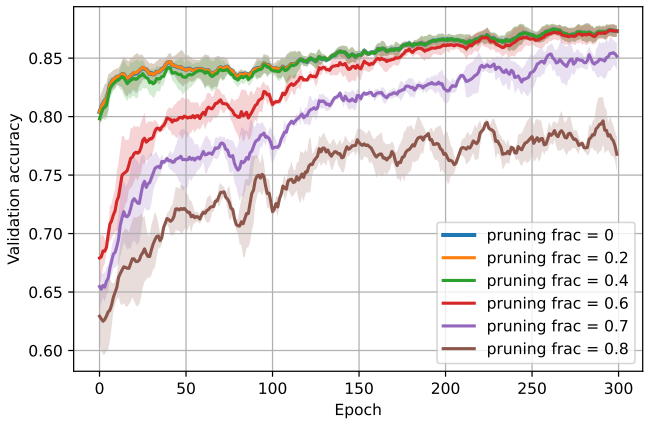


Fig. 4: Validation accuracy of FFF on the toy dataset using various levels of static pruning. Note that the graphs were smoothed via a moving window approach with window size 10 to improve visual clarity. Lines indicate average values over three independent runs, shaded areas denote standard deviations.

to 60%, as shown in Fig. 4. This demonstrates the ability of FFF to drastically reduce the overall amount of parameters and therefore memory footprint of a model, which is of central importance in resource-constrained settings. Another interesting observation is that initial performance differences between models with a higher degree of pruning and the baseline vanish over the course of training, e.g. in case of the 60% pruning variant. This illustrates the relevance of all tree paths during early training stages, where computation is unstructured and distributed. However, its tree structure allows FFF to perform conditional and targeted parameter updates, which leads to strongly localised, pruning-compatible computation during later training phases.

The third experiment measures the execution speed of a custom CUDA-based FFF implementation relative to dense PyTorch [18] feed-forward layers of comparable size. Inspired by recent developments in the area of tree ensemble execution on parallel hardware [24], our CUDA implementation encapsulates the sequential tree traversal of FFF efficiently inside a single kernel invocation. It furthermore optimizes memory access patterns by rearranging computation based on tree layout and activated paths. In robotic settings, faster model execution can be used either to reduce decision latency or to evaluate more candidate futures within a fixed control budget, which makes runtime an important practical metric. We use randomly generated inputs and feed them to sparse and dense single-layer test networks. We report one dense and three FFF configurations with varying tree depth d and match the amount of parallel trees P to demonstrate different dynamic sparsity regimes within defined parameter envelopes. The results in Table I show that FFF reduces execution time for all tested layer sizes, but provides increasing advantages for larger layers. Within a single size class, the depth and therefore relative sparsity level of the used FFF model (75% for $d = 3$, 90% for $d = 5$ and 97% for $d = 7$) does not strongly influence execution speeds. Within a given layer size, runtime varies only weakly across the tested tree depths, despite the differences in induced dynamic sparsity. This suggests that our current CUDA implementation is relatively insensitive to tree depth and may not yet fully realize the potential runtime differences between sparsity regimes. At the same time, all tested FFF variants already provide clear speedups over dense feed-forward layers. In practice, the choice of tree depth is likely to depend more on task requirements and accuracy–efficiency trade-offs than on raw runtime considerations alone.

TABLE I: Inference speedup of custom CUDA implementation in comparison to regular dense feed-forward layers for single-layer test networks executed with random inputs. Layer size refers to the total number of parameters of the tested layers, with small, medium, and large corresponding to approximately 500k, 8M, and 33M parameters, respectively.

Layer size	**dense**	**FFF (d = 3)**	**FFF (d = 5)**	**FFF (d = 7)**
small	1.0	1.237	1.192	1.195
medium	1.0	1.351	1.353	1.371
large	1.0	12.231	12.083	12.018

V. Conclusion

We investigated Fast Feedforward (FFF) layers as a sparse replacement for dense transformer feed-forward blocks in a world modelling setting. Across our experiments, FFF preserved predictive performance despite substantial dynamic sparsity and additionally tolerated high levels of static pruning. These results position FFF as a practical alternative to dense feed-forward sublayers in transformer-based world models that must operate under limited compute and memory budgets.

Our results support the view that structured sparsity can help bring predictive models closer to onboard deployment on resource-constrained robots. If world model evaluations become cheaper, more frequent inference or a larger number of candidate rollouts can be accommodated within a fixed compute budget, which is valuable for model-based decision making on resource-constrained platforms. Moreover, reducing the overall parameter count through static pruning may allow larger models to be deployed on a given hardware platform. While our study is limited to controlled symbolic environments, it suggests that sparse feed-forward computation is a promising design direction for efficient local execution of transformer world models.

A practical advantage of FFF is that it can be integrated into existing transformer architectures as a drop-in replacement for dense feed-forward blocks, without requiring changes to the overall training objective or additional routing or balancing losses. This makes FFF a simple and attractive mechanism for exploring efficient world models in future robotic systems.

VI. Acknowledgments

This work was funded by the German Federal Ministry for Economic Affairs and Climate Action (BMFTR) project ESCADE (01MN23004D). DK is funded by project SAIL (grant no. NW21-059A). The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer JUWELS at Jülich Supercomputing Centre (JSC).

References

[1] P. Ard'on, É. Pairet, K. S. Lohan, S. Ramamoorthy, and R. P. A. Petrick, "Affordances in robotic tasks - a survey," *ArXiv*, vol. abs/2004.07400, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:215785873

[2] W. Liu, A. Daruna, M. Patel, K. Ramachandruni, and S. Chernova, "A survey of semantic reasoning frameworks for robotic systems," *Robotics and Autonomous Systems*, vol. 159, p. 104294, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092188902200183X

[3] P. Zhang, Y. Cheng, X. Sun, S. Wang, F. Li, L. Zhu, and H. T. Shen, "A step toward world models: A survey on robotic manipulation," *ArXiv*, vol. abs/2511.02097, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:282749247

[4] U. Grover, R. Ranjan, M. Mao, T. T. Dong, S. Praveen, Z. Wu, J. M. Chang, T. Mohsenin, Y. Sheng, A. Polyzou, E. Kanjo, and X. Lin, "Embodied foundation models at the edge: A survey of deployment constraints and mitigation strategies," 2026. [Online]. Available: https://api.semanticscholar.org/CorpusID:286638661

[5] S. Pohland, X. Foukas, G. Ananthanarayanan, A. Kolobov, S. Mehrotra, B. Radunovic, and A. Verma, "Offload or overload: A platform measurement study of mobile robotic manipulation workloads," 2026. [Online]. Available: https://api.semanticscholar.org/CorpusID:286669952

[6] N. Tahir and R. Parasuraman, "Edge computing and its application in robotics: A survey," *ArXiv*, vol. abs/2507.00523, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:279671906

[7] P. Belcak and R. Wattenhofer, "Fast feedforward networks," 2023. [Online]. Available: https://arxiv.org/abs/2308.14711

[8] P. Belcák and R. Wattenhofer, "Exponentially faster language modelling," *ArXiv*, vol. abs/2311.10770, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265294694

[9] R. Sedghi, R. Schiewer, A. Subramoney, and D. Kappel, "Dynamic sparsity in tree-structured feed-forward layers at scale," *ArXiv*, vol. abs/2604.08565, 2026. [Online]. Available: https://api.semanticscholar.org/CorpusID:287351395

[10] Z. Zhao, W. S. Lee, and D. Hsu, "Large language models as commonsense knowledge for large-scale task planning," *ArXiv*, vol. abs/2305.14078, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258841057

[11] S. Yin, C. Fu, S. Zhao, K. Li, X. Sun, T. Xu, and E. Chen, "A survey on multimodal large language models," *National Science Review*, vol. 11, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259243718

[12] J. Bruce, M. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps, Y. Aytar, S. Bechtle, F. M. P. Behbahani, S. Chan, N. M. O. Heess, L. Gonzalez, S. Osindero, S. Ozair, S. Reed, J. Zhang, K. Zolna, J. Clune, N. de Freitas, S. Singh, and T. Rocktaschel, "Genie: Generative interactive environments," *ArXiv*, vol. abs/2402.15391, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267897982

[13] D. Hafner, W. Yan, and T. P. Lillicrap, "Training agents inside of scalable world models," *ArXiv*, vol. abs/2509.24527, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:281674095

[14] N. Hansen, X. Wang, and H. Su, "Temporal difference learning for model predictive control," in *International Conference on Machine Learning*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247318709

[15] N. Hansen, H. Su, and X. Wang, "Td-mpc2: Scalable, robust world models for continuous control," *ArXiv*, vol. abs/2310.16828, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:264451720

[16] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *ArXiv*, vol. abs/2101.03961, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:231573431

[17] T. Gale, D. Narayanan, C. Young, and M. A. Zaharia, "Megablocks: Efficient sparse training with mixture-of-experts," *ArXiv*, vol. abs/2211.15841, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:254069783

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[19] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. T. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, "Opt: Open pre-trained transformer language models," *ArXiv*, vol. abs/2205.01068, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248496292

[20] M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry, "Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks," *CoRR*, vol. abs/2306.13831, 2023.

[21] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l0 regularization," *ArXiv*, vol. abs/1712.01312, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:30535508

[22] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv: Learning*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:53388625

[23] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup, "Conditional computation in neural networks for faster models," *ArXiv*, vol. abs/1511.06297, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:16049527

[24] Z. Xie, W. Dong, J. Liu, H. Liu, and D. Li, "Tahoe: tree structure-aware high performance inference engine for decision tree ensemble on gpu," in *Proceedings of the Sixteenth European Conference on Computer Systems*, ser. EuroSys '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 426–440. [Online]. Available: https://doi.org/10.1145/3447786.3456251