Branch Prediction with Neural Networks

Vikraman Senthil Kumar
Department of Computer Science
viksenthil@ucdavis.edu

Madhumitha Santhanakrishnan
Department of Computer Science
msan@ucdavis.edu

Gabriel Eduardo Castellanos
Department of ECE
gecsatellanos@ucdavis.edu

Abstract—**Branch prediction plays a pivotal role in enhancing processor performance by mitigating the latency incurred during the resolution of conditional branches. Despite the integration of prediction algorithms, inaccuracies persist, impacting computational efficiency. This paper addresses this issue by introducing two variants of perceptron predictors capable of accommodating longer historical data for more accurate predictions, thereby substantially diminishing misprediction rates. Notably, a streamlined iteration of this approach demonstrated a remarkable 53% reduction in misprediction rates compared to the g-share predictor [1]. Additionally, the study delineates a circuit-level blueprint for the perceptron predictor, presenting strategies for optimizing operational efficiency. The central hypothesis posited in this research asserts that the development of a sophisticated perceptron model can yield notable improvements in instructions per cycle performance compared to the prevailing simplistic two-bit counters utilized in contemporary systems.**

1. Introduction

The efficient execution of instructions within processors heavily relies on effective branch prediction, a critical aspect in minimizing pipeline stalls and performance bottlenecks caused by mispredicted branches. These mispredictions substantially impede overall execution efficiency, emphasizing the significance of accurate prediction mechanisms.
Traditional branch prediction methods have historically relied on pattern history tables, branch history tables, and perceptrons to anticipate branch outcomes based on historical patterns or specific algorithms. However, the evolution of processor architectures necessitates more sophisticated approaches capable of handling longer historical data to capture intricate dependencies in branch behavior.
Using longer branch histories has emerged as a promising avenue for improving prediction accuracy, as it enables the learning of intricate relationships and dependencies present in extended historical data [1]. This capability potentially allows for exploiting longer-term dependencies in branch behavior, a factor critical for enhancing prediction accuracy and reducing misprediction rates.
The advent of neural networks has brought about a paradigm shift in branch prediction methodologies. Neural networks offer distinct advantages over traditional predictors due to their capacity to learn complex patterns and relationships from data. This adaptability grants neural networks the potential to capture and comprehend intricate patterns in branch behavior that may pose challenges for conventional prediction mechanisms. Thus, introducing neural networks, particularly the perceptron, has yielded remarkable advancements in branch prediction. The potential of neural networks, specifically the perceptron, in significantly improving branch prediction accuracy, thereby contributing to substantial enhancements in overall processor performance.
In this paper, we dive deep into the realm of neural network architectures to revolutionize branch prediction. By evaluating diverse neural network models, we aim to uncover approaches that surpass traditional predictors in accuracy and efficiency. Our focus is on tailored, domain-specific applications of these advanced models, proposing that their strategic implementation can lead to considerable improvements in processor performance.

2. Problem Definition

The deployment of neural networks for branch prediction in modern processor architectures presents a multifaceted set of challenges, among which latency is a critical concern [2].
Acquiring and preparing comprehensive datasets tailored for training neural networks in branch prediction encounters complexity due to the diverse nature of program behaviors. This diversity necessitates extensive execution traces, complicating the acquisition of large-scale datasets essential for effective neural network training. This issue is further compounded by the need to minimize latency during data acquisition and preparation to ensure timely training of neural network models.
Hardware constraints pose additional challenges, encompassing resource allocation, power efficiency, and real-time prediction requirements. The intricate and resource-intensive nature of neural network architectures exacerbates these challenges, contributing to increased latency in processing and prediction and impeding the timely execution of instructions.
Moreover, the significant memory demands of neural networks, including storage requirements for weights, activations, and parameters, present hurdles within the confined

memory resources of processors. This constraint further contributes to latency concerns, as efficient data retrieval and utilization become essential for maintaining real-time performance in neural network-based predictors for branch prediction.
Consequently, the challenges encompassing dataset complexity, hardware limitations, memory constraints, and the associated latency issues collectively hinder the seamless integration and effective utilization of neural network-based predictors within modern processor architectures. Addressing these challenges is imperative to minimize latency and enable the efficient operation of neural network-based branch prediction mechanisms in processors.

3. Related Work

Because of the importance of the Branch Predictors, substantial literature exists exploring various optimization techniques. We divide the section into four sub-sections based on the general ideas used for the optimization of Branch Predictors:

- Fast Path-based Neural Branch Prediction
- Combining Local and Global History Hashing
- TAGE
- BranchNet

3.1. Fast Path-based Neural Branch Prediction

The "Fast Path-Based Neural Branch Prediction" [3] model, set a new benchmark in the field when it was first published. This innovative approach leverages path-based neural networks to analyze and predict branch behavior, a substantial departure from traditional branch prediction methods.
The model functions by considering sequences of branches as paths, allowing it to capture patterns over longer sequences than conventional predictors. This path-based approach is particularly effective in addressing the limitations of previous predictors, which often struggle with complex branch patterns in modern applications. The neural network core of this model adapts to dynamic program behaviors, offering a higher degree of prediction accuracy.
Additionally, the implementation of this model demonstrates a significant reduction in misprediction penalties, a critical factor in enhancing processor performance. This was due to the neural network considering the path of the branch. The adaptive nature of the neural network also ensures that the predictor remains effective across a wide range of applications, including those with irregular branching behaviors.
Figure 1 delineates a clear trend: as the hardware budget increases from 1KB to 64KB, the Path-Based Neural Predictor consistently outperforms traditional models. This model's curve is noteworthy for its lower misprediction rates, evidencing enhanced prediction accuracy without proportional increases in hardware resources.

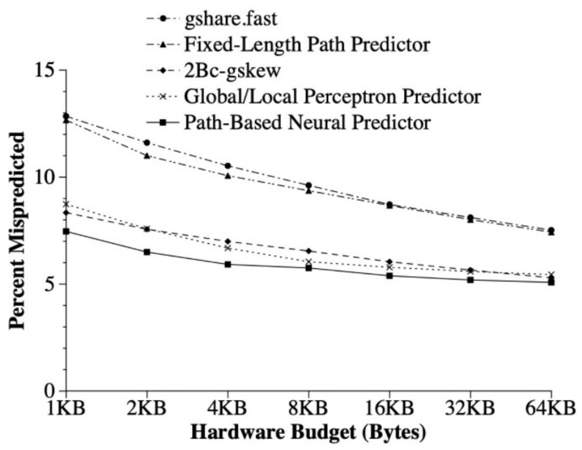

Figure 1. Compares Hardware Budget and Percent Mispredicted [3]

This research presents a pivotal step in branch prediction technology, suggesting that neural network-based predictors could be the key to unlocking higher levels of processor efficiency and performance. It sets a foundation for future research in the domain, opening avenues for more sophisticated and adaptable branch prediction mechanisms.

3.2. Combining Local and Global History Hashing

The methodology of "Combining Local and Global History Hashing" [4] in branch prediction stands as a significant milestone in the evolution of branch predictors. This technique synergizes local and global branch history information, encapsulating a broader context for more accurate predictions. By hashing together local and global histories, this approach mitigates the aliasing problem that plagues many predictors, leading to a reduction in misprediction rates.
Local history hashing provides detailed insight into the behavior of individual branches, while global history offers a macroscopic view of branching patterns across the program. The combined hashing technique effectively balances these perspectives, enabling the predictor to discern nuanced patterns and correlations that might be invisible to predictors considering only one history type.
Empirical evaluations of this combined approach have demonstrated its superiority over predictors using isolated local or global histories. It offers a robust solution to the challenges of dynamic and complex branch behaviors seen in modern applications. This innovative technique underlines the critical role of comprehensive history analysis in enhancing the accuracy of branch prediction.
The combined local and global history hashing method represents a leap forward, paving the way for more sophisticated prediction models that can harness the intricate interplay of branch histories to optimize processing pipelines.

3.3. TAGE

TAGE (Tagged Geometric Length) [5] predictor is a sophisticated branch prediction mechanism known for its multi-layered approach to predicting program execution paths. At its core, TAGE employs a series of tables, each filled with tagged saturating counters, that collectively contribute to the overall prediction decision. These tables are differentiated by the length of the branch history they utilize, with each one tailored to a specific history length, allowing TAGE to dynamically select the most appropriate table based on the current context.
This multi-table design is a strategic response to the varying demands of different branch patterns. When predictions based on short-term history are insufficient, TAGE seamlessly shifts to tables that analyze a longer history, ensuring that it taps into the most relevant historical data to make a prediction. This is particularly vital when the predictor encounters branches that are correlated with distant past events, buried within a noisy backdrop of global history [6].

3.4. BranchNet

BranchNet [7] is an advanced branch prediction model that builds upon the CNN predictor proposed by Tarsa et al [8]. The main advantage of BranchNet is its ability to discern patterns in noisy global branch histories, focusing on correlated branches and efficiently filtering out noise. By recognizing correlated branch patterns throughout the entire history, BranchNet provides a robust and expressive prediction function that maintains efficiency, even in the face of long and complex histories.
In contrast to conventional branch predictors that function in real-time, BranchNet employs a CNN that is trained offline, during compile-time. This training enables BranchNet to establish input-independent branch correlations, providing it with the capacity to predict branches accurately, even in scenarios involving previously unseen inputs. The compile-time training approach ensures that BranchNet's predictions are not just reactive, but proactive, effectively handling the dynamic nature of program execution.

4. Limitations of Related Work

Similar to the previous section, we divide this section into three broad categories to discuss the limitations of the mentioned related work

4.1. Performance Gains Do Not Scale

In branch prediction, performance gains do not scale linearly with the enlargement of branch history and hardware budgets, a limitation that is crucial to address in the study of predictive models. Despite the sophistication of combining local and global history hashing, this approach encounters the law of diminishing returns when excessively scaled. The augmentation of branch history length and the corresponding increase in hardware budget tend to yield substantial improvements initially. However, as these resources expand, the incremental benefits diminish. The predictors reach a saturation point beyond which additional history or hardware resources contribute minimally to prediction accuracy. This plateau suggests that there is an optimal threshold for resource allocation beyond which efficiency gains taper off.
This limitation is evident in Figure 1, which demonstrates that while increasing the hardware budget does lead to a decrease in misprediction rates, the benefit plateaus beyond a certain point. This indicates a diminishing return on investment, where additional hardware resources fail to correspond to proportionate improvements in prediction accuracy.
Moreover, the burgeoning complexity of managing extensive histories can lead to increased computational overhead and latency. It becomes imperative to devise intelligent methods to utilize history and hardware resources effectively, ensuring that performance improvements are proportional to the investments made.

4.2. Latency

In exploring neural network-based solutions for branch prediction, it is imperative to address a critical trade-off highlighted by Jiménez et al [2]: "Future branch predictors must consider not only area and accuracy but also delay." The incorporation of neural networks into branch prediction mechanisms, while significantly improving accuracy, often results in heightened latency, a factor that can offset the performance benefits.

Neural network-based predictors, due to their complex and computational-intensive nature, introduce a higher latency compared to their traditional counterparts. As these predictors analyze extensive patterns and histories, the time taken to generate a prediction increases. This latency can be particularly problematic in a pipeline where branch prediction speed is crucial to maintaining a swift instruction flow.

This underscores the necessity of a holistic approach in the development of branch predictors. The quest for accuracy must be balanced with the practical considerations of hardware constraints and the critical impact of latency. While neural networks offer a sophisticated means of prediction, their deployment in branch prediction is not without challenges. They require a carefully engineered balance to ensure that the delay introduced does not undermine the overall performance gains.

Consequently, future research must navigate the delicate equilibrium between enhancing accuracy and minimizing latency. Innovations in neural network design, training methods, and integration into the processor's pipeline are essential to devise solutions that uphold this balance. This would involve not only optimizing the neural networks themselves but also the surrounding architecture to accommodate the additional processing time without detriment to the system's throughput.

4.3. Inability to address Non-Linear Branches

A significant limitation identified in existing branch prediction models, particularly in the perceptron-based predictors, is their inherent inability to effectively predict linearly inseparable behaviors. This issue stems from the perceptron's reliance on a single-layer neural network architecture. As outlined by the authors in their comparative analysis, the perceptron's structure restricts its capability to accurately forecast non-linear branch patterns. This shortcoming is not readily rectifiable by simply adding multiple layers to the neural network. The reason is, the enhanced complexity introduced by additional layers renders the training process at runtime excessively computationally expensive to be practical.

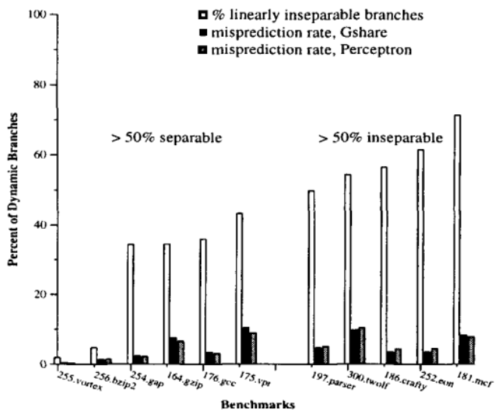

Figure 2. Comparing the misprediction rates of two branch prediction models—Gshare and Perceptron—across various benchmarks [9]

The impact of this limitation is evident when comparing the misprediction rates across different models. For instance, the gshare predictor often outperforms the perceptron in scenarios involving non-linear branches. This is attributed to the perceptron's inherent limitation in handling linearly inseparable behaviors, as visually represented in Figure 2. Consequently, this limitation emphasizes the need for more sophisticated, multi-layered neural network approaches or alternative predictive models that can efficiently and accurately address the complexity of non-linear branch behaviors in modern computational systems [9].

5. Plans for Improvement

In the realm of branch prediction, challenges such as latency and performance scaling necessitate innovative approaches. Traditional methods show limitations under evolving computational demands, and even advanced neural network-based predictors suffer from issues of complexity and latency. Our research focuses on novel strategies, including domain-specific branch prediction and the integration of Sliced Recurrent Neural Networks (SRNN) with TAGE, aiming to enhance accuracy and efficiency while addressing these key challenges.

5.1. Domain Specific Branch Prediction

Addressing the challenges presented by the limitations of traditional and neural network-based branch predictors, our research proposes a domain-specific branch prediction approach. This strategy recognizes that the advanced capabilities of neural network predictors, while beneficial, come with the tradeoff of increased complexity and latency. Thus, employing these sophisticated models only when necessary can optimize the balance between accuracy, area, and delay.

To evaluate the effectiveness of our domain-specific branch prediction methodology, we began by creating a synthetic branch trace generator. This generator was designed to produce traces with distinct patterns reflective of different application domains: machine learning applications, I/O heavy applications, and general-purpose applications. Using Python, we scripted functions to generate datasets for each of these domains, with varying branch outcomes based on different probabilities and conditions. The datasets simulate real-world branch behavior in a controlled environment, allowing us to test our predictors with a high degree of fidelity.

Once we generated these traces, we developed a simulator to assess the performance of various branch prediction strategies, including an Always Taken predictor, a Never Taken predictor, a Bimodal predictor, a Gshare predictor, and a Perceptron predictor. Each predictor was implemented with its unique logic to forecast branch outcomes, and their accuracies were compared using the generated datasets.
Our simulator, written in Python, used the CSV module for data handling and implemented classes and functions to represent different predictors. The core idea behind the domain-specific approach is to leverage the strengths of neural network predictors in scenarios where their accuracy outweighs the cost of their latency. For example, in machine learning applications where branch patterns are highly repetitive and predictable, a neural network predictor's slight delay might be justifiable due to its superior accuracy in such contexts. In contrast, for general applications where branch behavior is less predictable and performance is more sensitive to latency, a simpler and faster predictor might be preferable.
We meticulously recorded the performance of each predictor against our synthetic traces, using a combination of traditional metrics and neural network-based models. This comprehensive simulation allowed us to measure not only the accuracy of predictions but also the time efficiency of each predictor in operation.
In Figure 3, The neural network-based predictors demonstrated high accuracy but at the cost of latency, which was in line with the established literature. However, when these predictors were employed selectively, for instance,

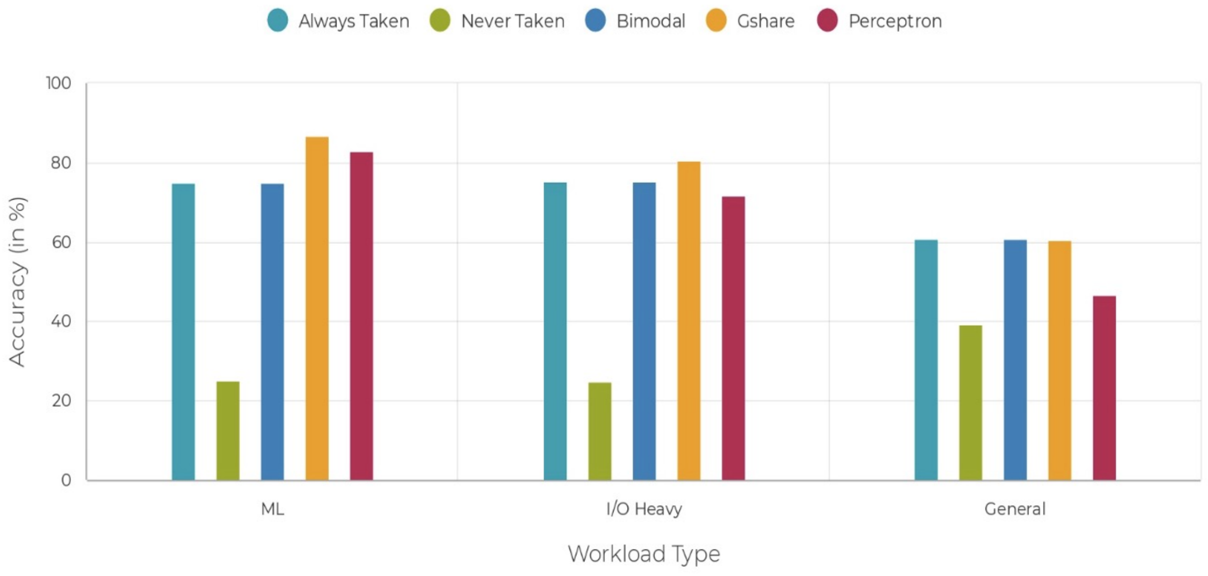

Figure 3. Results of various types of Branch Predictors on different work loads

only in machine learning applications as per our domain-specific strategy, the trade-off between accuracy and latency became favorable. The domain-specific approach allowed us to exploit the strengths of neural network predictors without succumbing to their weaknesses.
This nuanced approach aligns with the prescient observation by Jiménez et al [2] that future branch predictors must account for "not only area and accuracy but also delay." By tailoring the prediction strategy to the domain, we ensure that the performance gains from neural network predictors are harnessed efficiently, without incurring the excessive costs associated with their use in every context.
In conclusion, our domain-specific branch prediction methodology introduces a tailored solution that optimizes the tradeoff between complexity, latency, and accuracy. Through careful evaluation using a synthetic trace generator and a detailed simulation, we've demonstrated that this approach can lead to significant improvements in branch prediction performance in targeted scenarios. This research lays the groundwork for future developments in branch prediction, suggesting that a one-size-fits-all model may not be as effective as one that is finely tuned to the demands of the application domain.

5.2. Sliced Recurrent Neural Networks with TAGE

The Sliced Recurrent Neural Network (SRNN) represents a significant advancement over the traditional Recurrent Neural Network (RNN). By operating on sliced segments of input data, SRNN overcomes the sequential processing limitations of RNNs, allowing for parallel processing and thereby reducing training times and computational delays [10]. This architectural modification is crucial in handling complex data sequences more efficiently (see Figure 4).

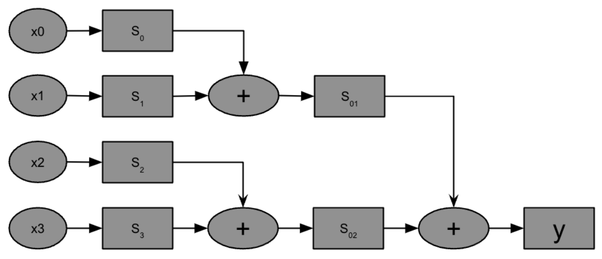

Figure 4. Structure of a 4-input Sliced Recurrent Neural Network (SRNN)

In our model, the integration of SRNN with TAGE, a state-of-the-art branch predictor known for its high accuracy in complex branch prediction scenarios, yields several notable benefits.
Firstly, SRNN's rapid adaptability and learning capabilities enhance TAGE's dynamic, history-based prediction. This combination is poised to achieve high accuracy swiftly, even in the initial learning phase, offering a robust solution for complex, evolving branch behaviors. Secondly, the parallel nature of SRNN, being up to 136 times faster than traditional RNNs, is particularly advantageous in managing long data sequences and non-linear branch patterns [11].
The essence of our approach lies in achieving an optimal balance. We aim to utilize the historical precision of TAGE, which excels in capturing extended patterns and trends in branch prediction. Concurrently, we leverage the agility of

SRNN, renowned for its quick adaptation to changing data sequences. This integration is carefully designed to avoid over-complicating the prediction system and to minimize the introduction of additional latency.
However, it is important to note the trade-offs involved. While SRNN enhances speed and adaptability, its combination with TAGE introduces additional system complexity and potential latency. To mitigate these challenges, we are exploring optimization techniques like model pruning and quantization for SRNN, alongside a streamlined version of TAGE. These measures are aimed at reducing the latency introduced by the combined system and ensuring that the benefits of both SRNN and TAGE are harnessed effectively without burdening the computational resources.
Looking ahead, implementing our branch prediction models within the gem5 simulation environment presents a great opportunity for further research. Given additional time and resources, the insights gathered from using such a robust and detailed architectural simulator could significantly surpass those obtained from our current Python-based simulator, which operates on synthetic data. The gem5 simulator's comprehensive framework would allow for a more nuanced and reliable evaluation of our predictive techniques, paving the way for potentially transformative improvements in branch prediction accuracy and efficiency.

6. Conclusion

In our study, we delve into the intricate landscape of neural network-based branch prediction, examining various optimization techniques and their inherent limitations. The essence of progress in branch prediction technology lies in navigating the trade-offs between complexity, accuracy, and processing speed. Our research advocates for a domain-specific approach to branch prediction, applying neural network models selectively where they are most effective, rather than adopting a uniform solution. Moreover, we propose that integrating Sliced Recurrent Neural Networks (SRNN) with the TAGE predictor could significantly improve prediction accuracy and speed, signifying a potential advancement in the realm of branch prediction technologies.

References

[1] D. A. Jiménez and C. Lin, *Neural methods for Dynamic Branch Prediction*. ACM Transactions on Computer Systems, vol. 20, no. 4, pp. 369–397, November 2002, doi: 10.1145/571637.571639.

[2] D. Jiménez, S. W. Keckler, and C. Lin, *The impact of delay on the design of branch predictors*. IEEE/ACM International Symposium on Microarchitecture (MICRO), vol. 33, November 2002, doi: 10.1109/micro.2000.898059.

[3] D. Jiménez, *Fast path-based neural branch prediction*. IEEE/ACM International Symposium on Microarchitecture (MICRO), vol. 36, May 2004, doi: 10.1109/micro.2003.1253199.

[4] C.-C. Ho and A. S. Fong, *Combining Local and Global History Hashing in Perceptron Branch Prediction*. International Conference on Computer and Information Science (ACIS), vol. 6, January 2007, doi: 10.1109/icis.2007.81.

[5] A. Seznec and P. Michaud, *A case for (partially) tagged geometric history length branch prediction*. J. Instruction-Level Parallelism, vol. 8, 2006.

[6] André Seznec, *A New Case for the TAGE Branch Predictor*. MICRO 2011: The 44th Annual IEEE/ACM International Symposium on Microarchitecture, ACM-IEEE, December 2011, Porto Allegre, Brazil. Ffhal-00639193f

[7] S. Zangeneh, S. Pruett, S. Lym, and Y. N. Patt, *BranchNet: A Convolutional Neural Network to Predict Hard-To-Predict Branches*. IEEE/ACM International Symposium on Microarchitecture (MICRO), vol. 53, pp. 118–130, October 2020, doi: 10.1109/MICRO50266.2020.00022.

[8] S. J. Tarsa, C.-K. Lin, G. Keskin, G. Chinya, and H. Wang, *Improving branch prediction by modeling global history with convolutional neural networks*. The 2nd International Workshop on AI-assisted Design for Architecture, 2019.

[9] D. A. Jimenez and C. Lin, *Dynamic Branch Prediction with Perceptrons*. Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture, vol. 7, January 2001, doi: 10.1109/hpca.2001.903263.

[10] Z. Yu and G. Liu, *Sliced Recurrent Neural Networks*. arXiv, July 2018. [Online]. Available: https://arxiv.org/abs/1807.02291

[11] L. Zhang, N. Wu, F. Ge, F. Zhou, and M. R. Yahya, *A Dynamic Branch Predictor Based on Parallel Structure of SRNN*. IEEE Access, vol. 8, pp. 86230–86237, May 2020, doi: 10.1109/access.2020.2992643.

[12] A. N. Eden and T. Mudge, *The YAGS branch prediction scheme*. In Proceedings of the 31st annual ACM/IEEE International Symposium on Microarchitecture (MICRO 31), IEEE Computer Society Press, Washington, DC, USA, 1998, pp. 69–77.

[13] Jared Stark, Marius Evers, and Yale N. Patt, *Variable length path branch prediction*. SIGOPS Oper. Syst. Rev. 32, 5, December 1998, 170–179. doi: 10.1145/384265.291042

[14] A. S. Fong and C. Y. Ho, *Global/Local Hashed Perceptron Branch Prediction*. IEEE Xplore, April 01, 2008. [Online]. Available: https://ieeexplore.ieee.org/document/

[15] Jiménez, Daniel A., *Multiperspective Perceptron Predictor*. 5th JILP Workshop on Computer Architecture Competitions (JWAC-5): Championship Branch Prediction (CBP-5). 2016.

[16] C. Lin and S. Tarsa, *Branch Prediction Is Not A Solved Problem: Measurements, Opportunities, and Future Directions* IEEE International Symposium on Workload Characterization (IISWC), Orlando, FL, USA, 2019 pp. 228-238. doi: 10.1109/IISWC47752.2019.9042108