Academic Summary of Publication:

BranchNet: A Convolutional Neural Network to Predict Hard-To-Predict Branches (MICRO 2020)

Authors:
Siavash Zangeneh, Stephen Pruett, Sangkug Lym, and Yale N. Patt

Topic:
Branch Prediction / Machine Learning in Computer Architecture

1. Introduction

As microprocessor pipelines continue to deepen and widen to extract higher instruction-level parallelism (ILP), the cost of control-flow disruptions has grown correspondingly severe. Branch prediction, the mechanism responsible for speculating the outcome of conditional branches, remains a primary bottleneck in modern high-performance cores. Despite decades of research culminating in state-of-the-art predictors like TAGE-SC-L (Tagged Geometric History Length with Statistical Corrector), a significant "accuracy gap" remains. While TAGE-based predictors are exceptionally effective at capturing correlations in moderate-length histories, they struggle fundamentally with what are termed "Hard-to-Predict" (H2P) branches.

The authors of this paper identify a critical limitation in current runtime predictors: the inability to handle "noisy" global history. In complex integer workloads (e.g., SPEC2017), the execution path leading to a specific branch often contains hundreds or thousands of intervening branches that are irrelevant to the target branch's outcome. These irrelevant branches constitute "noise." Traditional predictors like TAGE rely on indexing tables with specific history bits; if the "noise" shifts the relevant correlating branch even slightly in the history buffer, the index changes, and the predictor fails to match the pattern. To capture correlations spanning thousands of instructions across varying noise levels, TAGE would require exponentially growing storage tables, which is physically infeasible.

This paper introduces BranchNet, a novel predictor based on Convolutional Neural Networks (CNNs). The central hypothesis is that H2P branches are not truly random; rather, their outcomes depend on semantic program features that are obscured by noise. Unlike TAGE, CNNs possess an inherent property called shift invariance (or translation invariance). This allows a CNN to identify a correlating pattern (a "feature") regardless of where it appears in the history buffer. By training a CNN offline to recognize these features, the authors propose a hybrid system where the bulk of branches are handled by a standard TAGE predictor, while a select few H2P branches are offloaded to the specialized, high-accuracy BranchNet.

2. Summary of the Proposed Work

The BranchNet framework is a hardware-software co-design that integrates offline deep learning with runtime execution. The proposal is divided into the algorithmic design of the CNN and the microarchitectural implementation (Mini-BranchNet) required to meet tight timing constraints.

2.1. The BranchNet Architecture

The core of the proposal is a custom CNN architecture tailored for branch prediction. Unlike generic image-processing CNNs, BranchNet is optimized for 1D binary data streams.

Input Representation: The network receives the Global Branch History (GBH). To reduce

the dimensionality and storage cost, the authors utilize Geometric History Lengths (GHL). This technique hashes the most recent history bits with higher resolution and older history bits with lower resolution, mimicking the "fading memory" aspect of human cognition.
Convolutional Layers: The first stage of the network applies 1D convolution filters to the history. Each filter is trained to recognize a specific sequence of taken/not-taken branches (a feature). Because convolution slides over the entire input window, it can detect this feature regardless of its position in the history.
Sum-Pooling (The Key Innovation): Standard CNNs often use Max-Pooling, but BranchNet employs Sum-Pooling. This layer aggregates the activations from the convolutional layer. By summing the occurrences of a feature, the network effectively counts how many times a specific pattern appeared in the history, completely discarding positional information. This transforms the history into a "bag-of-features," making the predictor robust against noise. If a correlating branch shifts position due to a slightly different execution path, the sum remains the same, and the prediction remains stable.
Classification: The pooled features are passed through fully connected layers to produce a final probability (confidence score) for the branch being Taken or Not-Taken.

2.2. Offline Training and Hybrid Deployment

BranchNet is not trained at runtime. Training a CNN requires significant computational power and backpropagation, which is too expensive for the processor core. Instead, the authors propose a Profile-Guided Optimization (PGO) approach:
Profiling: The application is run with a profiling input set. The system identifies the top H2P branches—those that contribute most to the misprediction rate of the baseline TAGE predictor.
Training: The traces for these specific branches are extracted, and a CNN is trained offline to learn their behaviors.
Inference: The trained weights are embedded into the program binary. At runtime, when the fetch unit encounters an H2P branch (marked by the compiler or a lookup table), it triggers the BranchNet hardware inference engine. All other branches default to the standard TAGE-SC-L predictor.

2.3. Hardware Implementation: Mini-BranchNet

A standard software CNN is too slow for the fetch pipeline. The authors introduce "Mini-BranchNet," a specialized hardware accelerator designed to deliver predictions within 4 clock cycles.
Quantization: The network weights and activations are quantized to 8-bit integers to reduce area and power.
Parallelization: The convolution and pooling operations are heavily parallelized.
Sparsity: The design exploits the fact that only H2P branches use the engine, allowing the hardware to be powered down or shared effectively.

3. Results and Evaluation

The authors evaluated BranchNet using the SPEC2017 Integer benchmarks, comparing it against a highly optimized 64KB TAGE-SC-L baseline.
MPKI Reduction: The constrained "Mini-BranchNet" reduced the Mispredictions Per Kilo Instructions (MPKI) by a geometric mean of 9.6% over the 64KB TAGE-SC-L baseline. In specific difficult benchmarks like 505.mcf_r (a pointer-chasing heavy workload), the reduction was as high as 17.7%.

Performance (IPC): This accuracy gain translated to a geometric mean IPC improvement of 1.3%, with a maximum gain of 7.9% on mcf. While 1.3% appears small, in the context of highly tuned mature processors, this is considered a significant architectural gain.
The "Capacity" Myth: The authors compared BranchNet against an "Unlimited TAGE" (a TAGE predictor with effectively infinite storage). Remarkably, Big-BranchNet outperformed Unlimited TAGE (7.6% MPKI reduction). This result is academically significant because it proves that TAGE's failure is not just due to limited size (capacity misses) but due to a fundamental inability to model complex, noise-tolerant correlations (capability misses). BranchNet captures semantic relationships that TAGE cannot see, regardless of size.
Storage Efficiency: BranchNet demonstrated higher accuracy per bit of storage than simply doubling the size of TAGE, validating the efficiency of the CNN representation for sparse features.

4. Proposed Enhancement

Proposal: Dynamic Confidence-Based Hybrid Selector

Rationale and Critique:
The primary weakness of the BranchNet design is its reliance on static offline profiling. The set of branches classified as "Hard-to-Predict" (H2P) is fixed at compile time based on profiling data. This assumes that the "hardness" of a branch is an intrinsic property of the code. However, branch behavior is often input-dependent. A branch might be H2P under the profiling input set but easy to predict (or behave differently) under the actual user runtime input.
If the runtime behavior deviates significantly from the training profile (a phenomenon known as "dataset shift"), BranchNet might actually perform worse than TAGE because its weights are "overfit" to the profiling data. The current paper blindly trusts the offline model for all marked branches.

Proposed Solution:
We propose replacing the static selection logic with a Dynamic Confidence-Based Hybrid Mechanism.
Mechanism: Both predictors (TAGE-SC-L and BranchNet) should generate a prediction for the designated H2P branches. TAGE-SC-L already includes confidence counters (the statistical corrector). We should extend the BranchNet hardware to output a "confidence score," derived from the magnitude of the final sigmoid activation function.
Runtime Selection: A lightweight hardware structure monitors the runtime accuracy of BranchNet. If BranchNet's confidence score for a specific prediction is below a tunable threshold, or if the dynamic monitor detects that the BranchNet model is currently mispredicting, the system should fall back to the TAGE-SC-L prediction.
Benefit: This creates a safety net that allows the processor to exploit the deep semantic learning of BranchNet when execution matches the profile, but seamlessly revert to the robust, online-learning TAGE predictor during unseen phases or data distributions.

Additional Suggestion (Secondary): Continuous Online Fine-Tuning

To further mitigate the offline training limitation, a background thread or micro-controller could perform slow backpropagation updates on the BranchNet weights at runtime. This would allow the CNN to gradually adapt to new phases absent in the profiling data, turning BranchNet into a semi-dynamic learner.

Sources
researchgate.net - microarch.org - utexas.edu - semanticscholar.org