Exploring Deep Neural Networks for Branch Prediction

Yonghua Mao[1, 3], Huiyang Zhou[2], Xiaolin Gui[1]

1.*Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China*
2. *Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, USA*
3. *Science, Xi'an Polytechnic University, Xi'an, China*

***Abstract*—**Recently, there have been significant advances in deep neural networks (DNNs) and they have shown superior performance in audio and image processing. In this paper, we explore DNNs to push the limit for branch prediction. We treat branch prediction as a classification problem and explore both deep convolutional neural networks (CNNs) and deep belief networks (DBNs) for branch prediction. We analyze the impact of the length of hashed program counter (PC), local history register (LHR), global history register (GHR) and branch global addresses (GA) of deep learning classifiers on the misprediction rate. We compare the effectiveness of DNNs with the state-of-the-art branch predictors, including the perceptron, the Multi-poTAGE+SC, and MTAGE+SC branch predictors. The last two are the most recent winners of championship branch prediction (CBP) contests in the category with unlimited resources. Several interesting observations emerged from our study. The first is that for branch prediction, the DBNs and CNNs outperform the perceptron predictor while only deeper CNN models could outperform Multi-poTAGE+SC and MTAGE+SC. Second, we analyze the impact of the depth of CNNs (i.e., the number of convolutional layers and pooling layers) on the misprediction rates. The results show that deeper CNN structures lead to lower misprediction rates**.**

Keywords*—branch predictor, DBN, CNN, branch misprediction rate; deep neural networks.*

I. INTRODUCTION

In a processor pipeline, control hazards may occur if the next fetched instruction differs from the outcome of a branch. Because of the high frequency of branch instructions in a program, branch prediction is widely used to eliminate the pipeline bubbles due to control hazards.

Given the importance, branch prediction has been studied extensively. Among the proposed branch predictors in the recent Championship Branch Prediction (CBP) competitions, most are variants of the TAGE and/or perceptron branch predictors. The success of perceptron-based predictors confirms that neural networks can be useful in branch prediction. However, only a few works explored more advanced machine learning methods on branch prediction [1].Given the recent remarkable advances in deep learning, deep neural networks in particular, it is worthwhile to examine whether the more advanced deep neural networks can discover new possibility for branch prediction.

This study was partly supported by the National Natural Science Foundation of China (61472316), the grant Basic Research Program of Shaanxi Province (2016 ZDJC-05), and the key Research and Development Program of Shaanxi Province (2017ZDXM-GY-011).

Corresponding Author: Xiaolin Gui, xlgui@mail.xjtu.edu.cn

This paper explores deep neural networks for branch prediction by treating it as a classification problem. We explore both deep belief networks (DBNs) and convolutional neural networks (CNNs) for branch prediction. Note that, in this work, we focus on pushing the limit of branch prediction and do not consider the complexity of the predictors. It is consistent with the CBP competition rules that "CBP will make no attempt to assess the cost/complexity of the predictor algorithms for predictors with the unlimited storage budget." We analyze the impact of the length of hashed program counter (PC), local history register (LHR), global history register (GHR) and branch global addresses (GA) of deep learning classifiers on the misprediction rate. We compare the effectiveness of DNNs with the state-of-the-art branch predictors, including the perceptron, the Multi-poTAGE+SC, and MTAGE+SC branch predictors. Several interesting observations emerged from our study. First, we confirm that deep learning algorithms outperform the perceptron predictor. Second, between DBNs and CNNs, we find CNNs are a better choice for branch prediction. Third, we analyze the impact of the depth of CNNs on misprediction rates. Our experimental results show that deeper CNN structures lead to lower misprediction rates. Fourth, we found deep CNNs could achieve lower misprediction rates than state-of-the-art branch predictors, Multi-poTAGE+SC and MTAGE+SC, with the unlimited storage budget.

The rest of the paper is organized as follows: Section 2 discusses the state-of-the-art branch predictors, TAGE and perceptron, in particular. Section 3 reviews DBNs and CNNs. Section 4 describes our experimental methodology. In Section 5, we present the comparison results and analysis of DNNs and other related predictors. Section 6 concludes and discusses the future work.

II. THE STATE OF THE ART BRANCH PREDICTORS

The TAGE (TAgged GEometric history length) branch predictor [2] is often considered as the most accurate branch predictor. TAGE is derived from Seznec's GEHLpredictor[3] and Michaud's tagged PPM-like predictors [4]. One key advantage of TAGE predictors over other predictors is to use a geometric series of history lengths for prediction. It enables the predictor to explore the correlation between branches in very long history lengths, while allocating most of the hardware resource to the short-length prediction components. Another key aspect of TAGE is that it uses tag-matches when accessing each prediction component to reduce aliases.

TAGE predictors have also been enhanced by adding prediction components targeting specific types of hard-to-predict branches. For example, TAGE-SC-L [5][6] improves the TAGE predictor by adding a statistical corrector predictor and a loop predictor. The idea of a statistical corrector predictor is to revert the prediction of the TAGE predictor when it statistically mispredicted in similar branch circumstances. The loop predictor targets at loop branches and uses loop counts to make accurate predictions.

Recently, Seznec and Michaud further improved the TAGE prediction accuracy via a Multi-poTAGE+SC predictor [7]. It combines multiple TAGE predictors and the final prediction is selected from these predictors via a combined output lookup table (COLT) predictor [8]. Each TAGE component takes a different combination of history, including both global and local history as inputs. This colossal multiple-TAGE predictor and its further fine-tuned version MTAGE+SC [9] are not meant for practical usage and are mainly for pushing the lower bound of misprediction rate of branch predictors, similar to the purpose of this work.

Another type of the state-of-the-art branch predictors is the perceptron predictor [10]. It uses a single-layer perceptron, one of the simplest neural networks to learn the correlation between the branch history and branch outcomes. The predictor builds a perceptron table, which is indexed by the branch program counter (PC). Each entry in the table consists of a set of weights. When making a prediction, the predictor first computes the output as the dot product of the input (i.e., history bits) and the indexed weights. Then the sign of the output provides the final prediction. After the branch is resolved, if it is mispredicted or the output is smaller than a pre-defined threshold, the selected weights will be trained. It trains each weight via adding the product of the corresponding input bit and the branch outcome. This training policy effectively strengthens the weights corresponding to the inputs with strong correlation to the outcome. Unfortunately, such a naïve single-layer perceptron is only capable of learning linear-separable branches. In order to overcome this shortcoming, different variations have been proposed. The piecewise perceptron [11] adds one more dimension to the perceptron table – global history address corresponding to the instruction address of each bit in the global history. However, both perceptron and piecewise perceptron predictors imply that a weight can only be assigned to a single history bit or history address. It means the complexity of the output computation grows linearly with the number of bits in the global history. Tarjan et al. [12] proposed that this side effect could be eliminated by a hashed perceptron [13], in which multiple history bits are hashed to a single weight.

III. DEEP LEARNING AND CONVOLUTION NEURAL NETWORKS (CNNS)

Perceptron is one of the simplest neural network models. It is a linear classifier algorithm for supervised classification. A perceptron model is presented in Fig. 1. Many branch predictors achieve low misprediction rate by such correlative

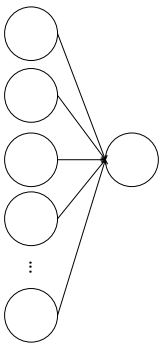

Fig. 1. The structure of a perceptron model.

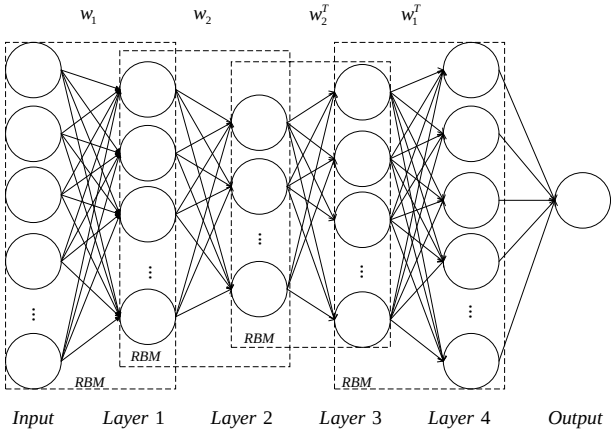

Fig. 2. A popular DBN structure (W_N is the weight between two layers).

perceptron classifiers [10, 12, 14]. With recent advances in deep learning showing highly impressive misclassification rate for image or audio based processing, we aim to further push the lower bound of misprediction rates by applying these algorithms to branch prediction.

Deep learning is a set of algorithms to train and utilize multi-layer neural networks. The deep hierarchical architecture tries to extract and represent the high-order features of the training data. However, traditional machine learning algorithms such as back-propagation were inadequate in training such a deep architecture because of the high probability of falling into poor local optima.

To deal with the complexity of training deep networks, Hinton et al. [15] proposed the Deep Belief Networks (DBN) and an efficient way to train the network [16]. A DBN is composed of several stacked restricted Boltzmann machines (RBMs). It first uses unlabeled data to pre-train the network layer by layer using contrastive divergence learning on every RBM . This step is a way of unsupervised feature learning. After pre-training, global training algorithms, such as back propagation, are used to fine-tune the weights in the network. A commonly used DBN structure with four RBM layers and an output layer is shown in Fig.2. The number of neurons in Layer 1 is around 1/3 ~ 2/3 of the inputs. Similarly, the number of neurons in Layer 2 is around 1/3 ~ 2/3 of Layer 1. Layer 3 has the same size as Layer 1, and Layer 4 has the same size as

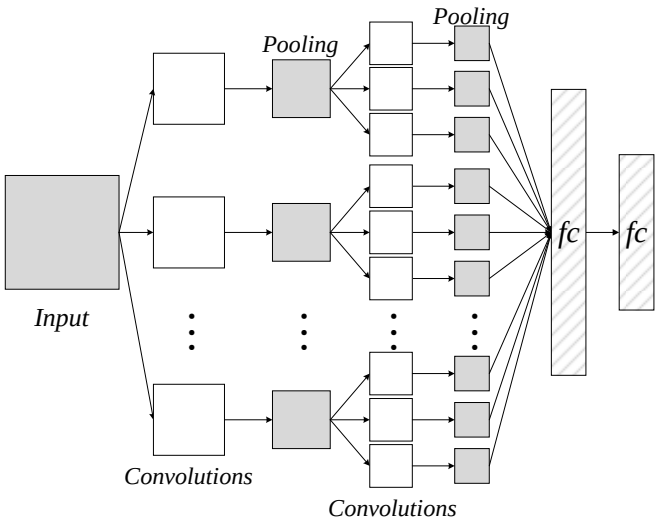

Fig. 3. The convolution neural network structure.

the input layer. The last layer is the output layer, which is constructed as a simple single-layer neural network, i.e., a perceptron. Although the network has 4 RBM layers, the training process is not exceedingly complex. Only the weights W_1 and W_2 will be trained. W_3 and W_4 are the transpose of W_2 and W_1, as in shown Fig. 2. The basic idea of DBN is that through layer-wise training the neural network, the input data is reconstructed in the last RBM layer (Layer 4 in Fig. 2) with the minimum reconstruction error. In an ideal case, if Layer 4 keeps all the information of the input data, which means the system does not introduce any information distortion, any inner layer between the input layer and output layer is another representation of the input. In other words, inner layers automatically extract some high-order features of the data since they have less neurons than the input vector. However, in reality, the information will distort layer-by-layer during the training process. From the input layer to Layer 2, the number of neurons reduces exponentially. This enforces Layer 2 to lose some information of the original data. From Layer 2 to Layer 4, the original data is reconstructed from low-dimensional layers while keeping the reconstruction error small. As a result, only the feature information, which is necessary to reconstruct the original data, will be preserved.

Locally connected networks with the longest history information, such as convolution neural networks (CNNs) [17], are likely to be a better choice for branch prediction as most branches show high correlation with nearby history. CNNs [18] are deep feedforward neural networks. The vital components of a CNN architecture are convolution layers and pooling layers, as shown in Fig. 3. They exploit the high correlation in local groups of data. The convolution layer [19] is used to detect the local conjunctions of features from the previous layer, and the pooling layer combines similar local features. A CNN typically has multiple stages of convolutional and pooling layers stacked one after the other, followed by a fully-connected (fc) layer. Backward propagation through a CNN is used to update the weights during the training phase, the same as training regular deep network layers.

TABLE I.

BENCHMARKS

Bench-marks	Dynamic Condi-tional Branches	Bench-marks	Dynamic Condi-tional Branches
F1	2213673	M1	2229289
F2	1792835	M2	3809780
F3	1546797	M3	3014850
F4	895842	M4	4874888
F5	2422049	M5	2563897
I1	4184792	S1	3660616
I2	2866495	S2	3537562
I3	3771697	S3	3811906
I4	2069894	S4	4266796
I5	3755315	S5	4291964

IV. EVALUATION METHODOLOGY

In order to evaluate various algorithms for branch prediction, we leverage the simulation framework provided in the 4th Championship Branch Prediction (CBP-4). The framework is based on trace-driven simulation and features 20 benchmarks, which are grouped into four categories: I (integer), F (floating point), M (multimedia) and S (server). In this work, we focus on the conditional branches from each trace as listed in Table I. The more recent CBP-5 has 233 traces, which would have resulted in impractical time spent on training the DNNs. Therefore, we focus on CBP-4 in this work.

In this paper, the problem of branch prediction is treated as a binary classification problem. We use an offline training method with a training set of first 90% branches, a validation set of next 5% branches and a testing set of last 5% branches. This way, the causality of data is maintained, meaning that future data are not used to train the network to produce a current prediction. We also use a random approach to select the training, validation, and test set. The training set is used to train the networks. The validation set is used to estimate how good a network has been trained in the training progress. If the network is good enough, meaning that the cost function on the validation set does not decrease between two continuous iteration epochs, the training process will stop. Then, the test set is used to evaluate the final classification error rate after the network has been trained. We specialize the network for each

TABLE II

CONFIGURATIONS OF DIFFERENT DBN STRUCTURES

Part	Architec-tures	Input info. bits	Source of bits				Configurations
			GA info.	PC (bits)	GHR (bits)	LHR (bits)	
A	DBN 1	140	none	32	100	8	140 – 96 – 40 – 96 – 140 – 1
	DBN 2	280	prior 32 bits PC		200	16	280 – 140 – 85 – 140 – 280 – 1
	DBN 3	592	prior 32 bits PC		512		592 – 260 – 130 – 260 – 592 – 1
B	DBN pc1	140	none	32	100	8	Same to DBN1
	DBN pc2	124		16			124 – 88 – 60 – 88 – 124 – 1
	DBN pc3	116		8			116 – 65 – 30 – 65 – 116 – 1
C	DBN L1	124	none	16	100	8	Same to DBN pc2
	DBN L2	128				12	128 – 90 – 40 – 90 – 128 – 1
	DBN L3	132				16	132 – 90 – 40 – 90 – 132 – 1
D	DBN G1	132	none	16	100	16	Same to DBN L3
	DBN G2	232			200		232 – 160 – 110 – 160 – 232 – 1
	DBN G3	544			512		544 – 325 – 190 – 325 – 544-1
E	DBN GA0	132	none	16	100	16	Same to DBN L3 & G1
	DBN GA1	260	16 8-bit				260 – 180 – 110 – 180 – 260 – 1
	DBN GA2	388	32 8-bit				388 – 225 – 140 – 225 – 388 – 1
	DBN GA3	516	48 8-bit				516 – 305 – 195 – 305 – 516 – 1
F	DBN L	944	48 8-bit	32	512	16	944-630-270-630-944-1

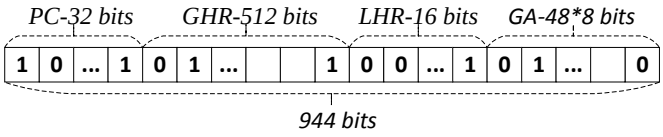

Fig. 4. The data organization from branch traces.

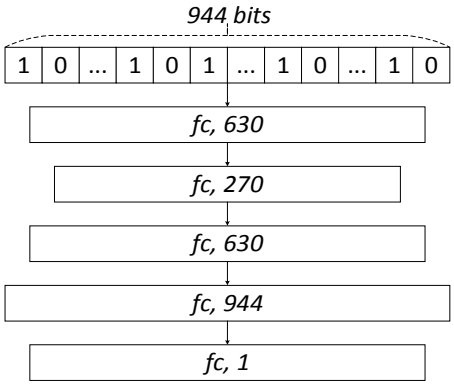

Fig. 5. *The DBN* architecture for branch prediction (fc: fully-connected)

trace separately. Our DNNs are constructed, trained, and tested using the *Caffe* framework [20].

Table II shows the input sizes, which include branch PCs, GHRs, LHRs, and GAs. In perceptron and our DNNs, Every bit is an input to a neuron of the input layer of a DNN. For example, a input data sample has a size of 944 bits, which includes a 32-bit PC, 16-bit LHR, 512-bit GHR, and 48 8-bit GAS (global addresses), as shown in Fig. 4.

We have tested DBNs and CNNs with various configurations, and have observed consistent results. To provide instances for discussion, we describe one DBN and two CNN models for branch prediction. The other DBN configurations are shown in Table II.

DBN. Based on a popular DBN structure, the numbers of neurons of the layer 1 and of the layer 2 are selected from a thorough search in the large design space of their structures, which is shown in last row of Table II. as shown in Fig. 5. Layer 3 has the same size as Layer 1 and Layer 4 is the same size as the input layer. The last layer is the output layer, which is constructed as a simple single-layer neural network. The notation 'fc' means fully-connected.

CNN1. The CNN1 configuration is inspired by the *LeNet* [17]. As shown in Fig. 6, it contains two convolutional layers

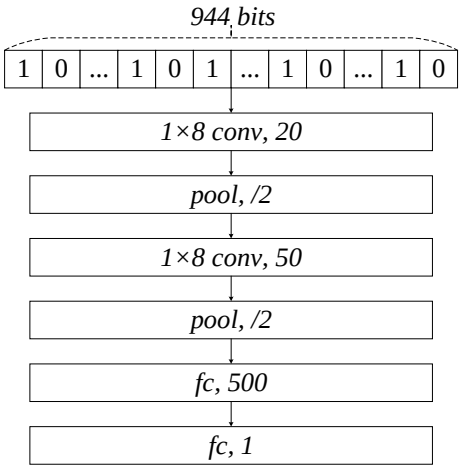

Fig. 6. CNN1 Architecture for branch prediction

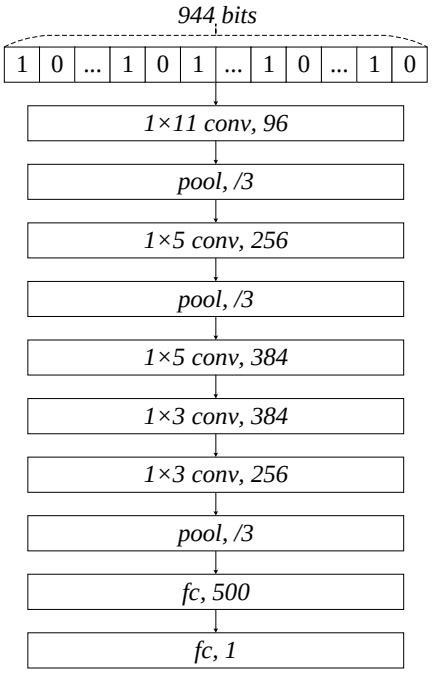

Fig. 7. CNN2 Architecture for branch prediction

and two fully-connected layers. The convolution layers have 1×8 filters. The network ends with a fully-connected layer with logistic regression. The configuration of the first fully-connected layer (i.e., 500) is also selected from a thorough search in the large design space of their structures.

CNN2. CNN2 configuration is based on the *AlexNet* [18], as shown in Fig. 7, We delete one fully-connected layer, and adapt the filters to branch data. The network contains five convolutional layers and two fully-connected layers. Its two fully-connected layers have the same configuration as in CNN1.

From comparison, we also report the prediction accuracy of multi-poTAGE+SC and MTAGE+SC predictors. While, the predictors are updated for all branch outcomes in a trace, we report the prediction accuracy of the branches in the same test set.

V. RESULTS AND DISCUSSION

The misprediction rate of the test set is our primary concern, so all results are on the testing sets. We report the misprediction rate as (#mispredictions / #predictions * 100) for all the prediction algorithms.

A. Comparing perceptron with popular DBN

Fig. 8 presents the misprediction rate of DBN and perceptron. The misprediction rates of the perceptron predictors with different history lengths are labeled 'pcptrn -140', 'pcptrn -280', and 'pcptrn -592', respectively; the misprediction rates of the corresponding DBNs are labeled 'DBN1'. 'DBN2', and 'DBN3'. From the Fig. 8, we can see that DBNs outperform perceptron consistently. On the benchmark 'I3', the DBN1 predictor reduces the mispredictioin rate by 9.81% compared to the perceptron with the same history length. Compared to perceptron, the average misprediction rate of the corresponding DBN predictor is 3.77%. This confirms that the DNNs could reduce the misprediction rates for the branches whose outcomes are separable in non-linear ways. If the branch outcomes of a benchmark are linearly separable, such

as those in the benchmarks ‘F3’, ‘F4’, ‘F5’, and ‘I5’, the perceptron and DBN have similar misprediction rates. On the contrary, the branch outcomes of a benchmark are not linearly separable, such as ‘I2-I4’, ‘M1-M2’, and ‘S1-S5’, the DBNs show their potential to greatly reduce the misprediction rate. The multiple layers of the DNNs are able to learn complex nonlinear functions more concisely, therefore they could work well on the nonlinearly separable branches.

From Fig. 8, we can also see the impact of the history length on the perceptron predictors. There are 19 benchmarks, for which the perceptron with 280-bit length has lower misprediction rates than the one with the 140-bit length. There are 17 benchmarks, for which the perceptron with the 592-bit length has lower misprediction rates than the one with the 280-bit length. As a result, we can see that although longer history lengths help to reduce misprediction rates, it is not always the case for any benchmark. This observation confirms the finding from the prior work [14], which shows that the longest match may not be the best for branch prediction. Fig. 8 also shows there are 19 benchmarks where the 592-bit input is better than the 140-bit input. This indicates that adding global addresses is profitable in perceptron predictors. We discuss the effect of PCs, LHRs, GHRs, and GAs in the DBN models separately next.

B. Comparing deep learning in the different lengths of hashed PC, LHR, GHR, and GA

Fig.9 shows the effect of the length of the hashed PC. DBN pc1, DBN pc2, and DBN pc3 use 32-bit raw PC, 16-bit hashed PC, and 8-bit hashed PC, respectively. The LHR and the GHR information are the same over the three structures, as shown in Table 2. Comparing the raw PC with the 16-bit hashed PC, most benchmarks show lower misprediction rates than the latter. Comparing DBN pc 2 and DBN pc3 (i.e., 16-bit PC vs. 8-bit PC), only 1 benchmark shows higher misprediction rate when 16-bit PC is used. Among the three configurations, the DBN with the raw 32-bit PC (i.e., DBN pc1) shows the lowest average misprediction rate.

Fig. 10 shows the effect of the length of the LHR. DBN L1, DBN L2, and DBN L3 use 8-bit, 12-bit, and 16-bit LHRs, respectively. The hashed PC and the length of GHR are 16 bits and 100 bits, respectively. There are 12 benchmarks, for which the DBN with the 12-bit LHR has lower misprediction

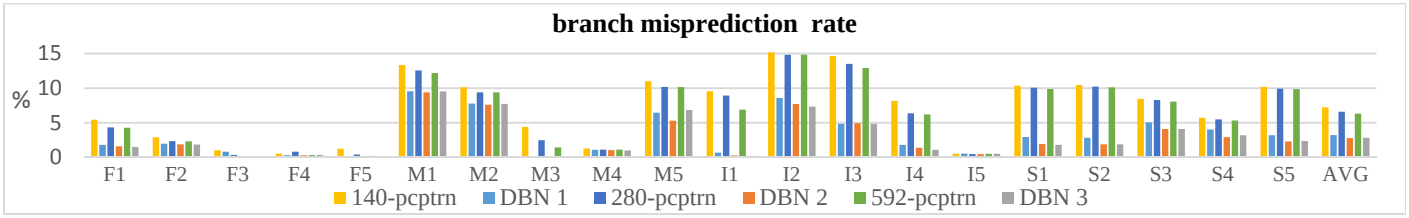

Fig. 8. The misprediction rate of perceptron and DBN on the testing set of the CBP-4 traces (the higher the better)

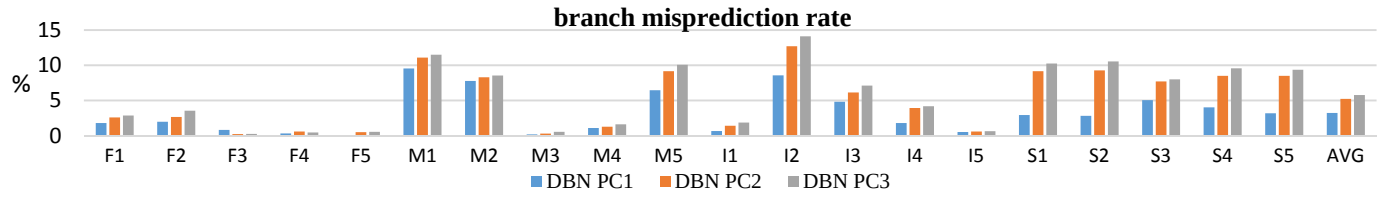

Fig.9. The effect of the hashed PC of DBN branch misprediction rate on CBP-4 traces

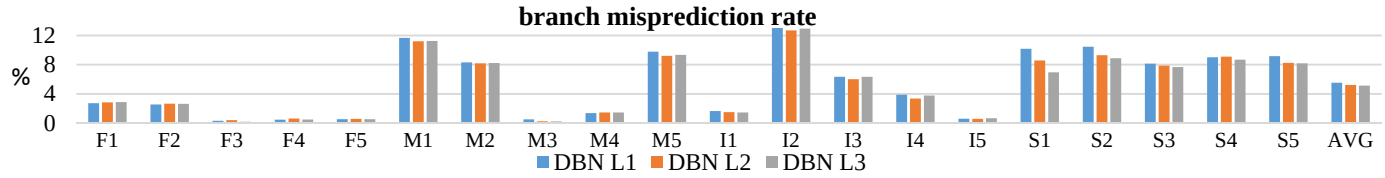

Fig.10. The effect of the length of the LHR of DBN branch misprediction rate on CBP-4 traces

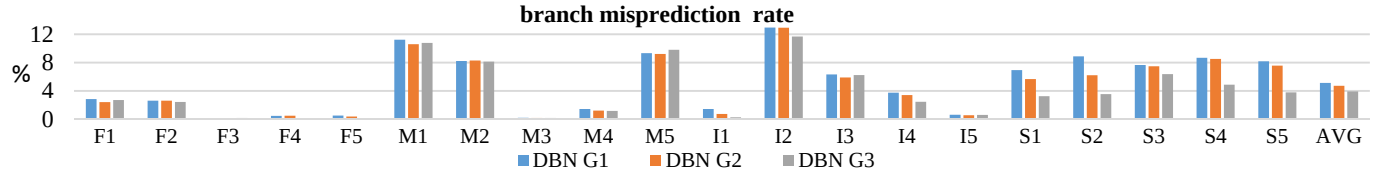

Fig.11. The effect of the length of the GHR of DBN branch misprediction rate on the testing set of CBP-4 traces

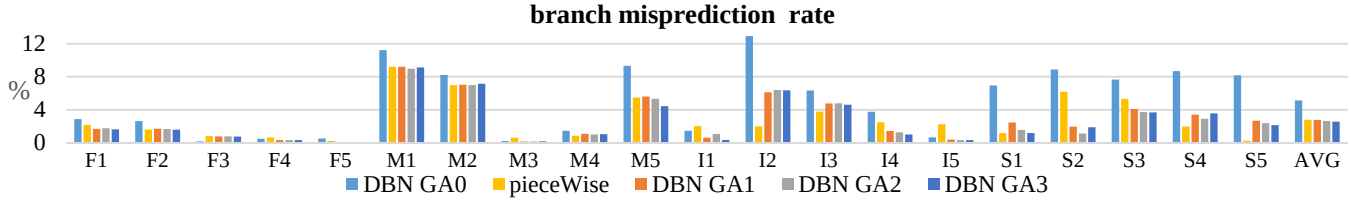

Fig.12. The effect of the length of the GA of DBN branch misprediction rate on the testing set of CBP-4 traces

rates than the DBN with the 8-bit LHR, and 10 benchmark, for which the DBN with the 16-bit LHR has lower mirepdiction rate than the one with the 12-bit LHR. There are only 5 benchmarks showing that the misprediction rate consistently reduces with an increase in the LHR length. There are also 2 benchmarks whose misprediction rates increase with longer LHRs. It indicates that the longest LHRs may not be the best for the DBN on every benchmarks.

Fig. 11 shows the influence of different GHR lengths in DBN G1, DBN G2, and DBN G3 structures as shown Table II. The lengths of the GHRs are 100 bits, 200 bits, and 512 bits respectively. The hashed PC address and LHR length are the same over three structures. There are only 12 benchmarks where the misprediction rate drops when the GHR length increases. Therefore, we could not conclude that, the longer GHR lengths, the lower the misprediction rate on all benchmarks. It implies that an adaptive GHR length is desired for different benchmarks.

To study the impact of GA, Fig. 12 shows the misprediction rate of the piecewise perceptron [11] and DBNs with 16, 32, and 48 8-bit GAs. Here 16/32/48 8-bit GAs mean 16/32/48 prior branch addresses with 8 bits for each address. In the DBN GA0, no GA is used. The Hashed PC, LHR, and GHR are consistent among the structures.

From Fig. 12, it can be seen that all DBN classification models outperform the piecewise perceptron for most of the benchmarks except the DBN G1, which does not contain any GA. Compared to non-GA, the impact from GA is significant in 'M5', 'I2', 'S1-S5'. Especially in 'S2', the reduction is as high as 7.01% between DBN GA0 (non-GA) and DBN GA 3(48 8-bit GA).

Another interesting observation is the impact of the GA length. There are 13 benchmarks for which the DBN with the 48-GA has lower misprediction rate than the one with the 32-GA, and 15 benchmarks for which the DBN with the 48-GA has lower misprediction rate than the one with the 16-GA. Therefore, we cannot reach a conclusion that a longer GA is always better than a short one.

As discussed above, the GHR length, the LHR length, and the GA length have significant impact on branch misprediction rates. Although not always longer the better, the misprediction rate correlates positively to the history length for over half of the benchmarks. Therefore, the best performing DBN (labeled as DBN-L) is the one the longest history information as shown in Table II.

In Fig. 13, we compare DBN L with a prior work, the AIP [14] perceptron predictors. We can see the misprediction rate of the DBN L is lower than AIP [14] for 15 benchmarks while the misprediction rate of the DBN1 is higher than AIP for 16 benchmarks.

C. *Comparing CNN and DBN with the state-of-the-art branch predictors*

In this part, all NNs have the same input with the size of 944 bits. Fig.14 shows the misprediction rates of the DBN L, CNN, and TAGE. Between DBNs and CNNs, our results show that CNN1 and DBN L have similar misprediction rates while CNN2 has much lower misprediction rates than DBN L or CNN1. CNN1 contains two convolution layers whereas CNN2 contains six convolution layers. This shows that for branch prediction, only deep CNNs are more effective than DBN, and a higher number of convolution layers and pooling layers could lead to lower misprediction rates.

The multi-poTAGE+SC predictor [8], which is labeled 'TAGE1' in Fig.14, is the winner of CBP-4 in the unlimited hardware resource category. The MTAGE+SC [10], which is labeled 'TAGE2' in Fig.14, is the winner in CBP-5. Between MTAGE+SC and multi-poTAGE+SC, the average reduction is 0.008%. They represent the state-of-the-art in achieving the upper bound of branch prediction. From Fig.14, we can see that there is only 1 trace on which the DBN and CNN1 have lower misprediction rates than multi-poTAGE+SC; but there are 16 benchmarks on which the CNN2 model has lower misprediction rates than multi-poTAGE+SC. The average of misprediction rate of the CNN2 model is 0.173% and 0.164% lower than multi-poTAGE+SC and MTAGE+SC, respectively. As a result, we can see that the CNN approaches have the potential to further reduce the lower bound of the branch misprediction rates. The deep convolution layers help to extract the local features which have same history information but different results in some branches, just like the separation of homonyms.

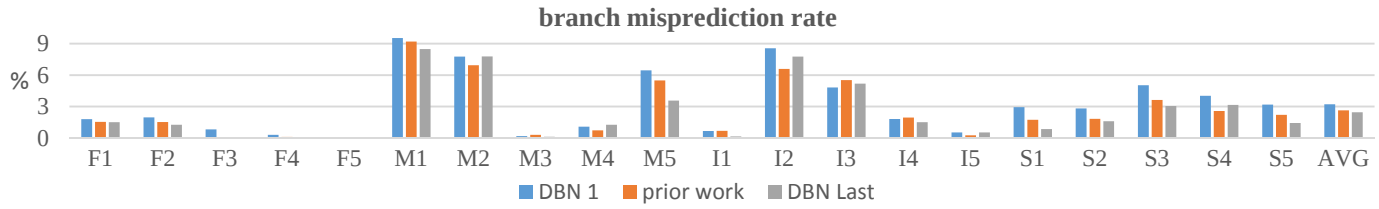

Fig.13. The branch misprediction rate of DBN1, the prior work[14], and DBN on the testing set of CBP-4 traces.

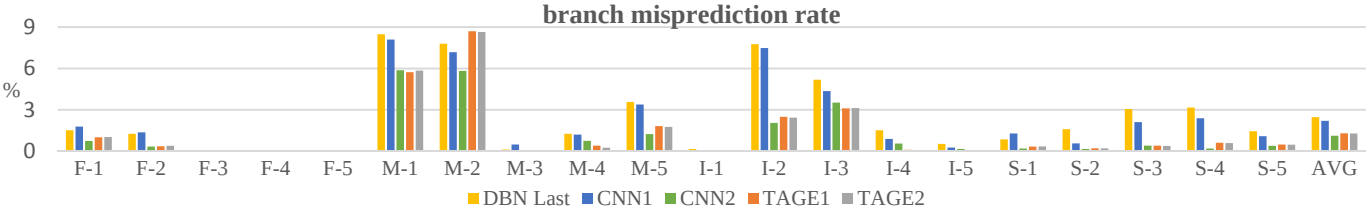

Fig.14. The branch misprediction rate of DBN Last, CNN1, CNN2, TAGE1, and TAGE2 on the testing set of CBP-4 traces

VI. CONCLUSIONS

This paper takes a binary classification perspective on the branch prediction problem. We utilize deep neural networks as a classifier and we explore both DBNs and CNNs to push the lower bound of branch misprediction rates. We made the following observations from our experiments: (1) deep neural networks significantly outperform simple perceptron classifiers; (2) deep CNNs outperform DBNs; and (3) only deep CNNs could outperform state-of-the-art TAGE-like branch predictors.

This paper treats branch prediction as a pure binary classification stochastic problem. To simplify the problem, we only implemented offline training. However, in order to apply deep learning for branch prediction, an online training algorithm needs to be employed. In addition, since most of the state-or-the-art branch predictors integrate several standalone predictors, it is also worthwhile to explore the influence of incorporating such complementary predictors into the deep CNNs.

REFERENCE

[1] D. Gope, M.H. Lipasti, "Bias-Free Neural Predictor", Proceedings of the 47th Annual IEEE/ACM International Symposium on microarchitecture, MICRO-47, IEEE, pages 521 – 532, Dec.2014.

[2] A. Seznec; P. Michaud, "A case for (partially) TAgged GEometric history length branch prediction", Journal of Instruction-Level Parallelism, vol. 8, pages 1-23, 2006.

[3] A. Seznec, Analysis of the O-GEometric History Length Branch Predictor, ISCA-32, pages 394-405, 2005.

[4] P. Michaud, "A PPM-like, tag-based branch predictor", Journal of Instruction-Level Parallelism, vol.7, pages 1-10, 2005.

[5] A. Seznec, "TAGE-SC-L Branch Predictors", In Proceedings of the 4th Championship on Branch Prediction, ISCA-41, June 2014.

[6] A. Seznec, "A new case for the TAGE branch predictor", In Proceedings of the MICRO 44, pages 117-127, Dec. 2011.

[7] A. Seznec,P. Michaud, "Pushing the branch predictability limits with the multi-poTAGE+SC predictor", In Proceedings of the 4th Championship on Branch Prediction, ISCA-41, June 2014.

[8] G. H. Loh, D. S. Henry, "Predicting conditional branches with fusion-based hybrid predictors". In Proc. of the Int. Conf. on Parallel Architectures and Compilation Techniques, pages 165-176, Sept. 2002.

[9] A. Seznec, "Exploring branch predictability limits with the MTAGE + SC predictor", 5th JILP Workshop on Computer Architecture Competitions (JWAC-5): Championship Branch Prediction, ISCA-43, June 2016.

[10] D. A. Jimenez, C. Lin, "Dynamic branch prediction with perceptrons", IEEE High-Performance Computer Architecture Symposium Proceedings, pages 197-206, Aug.2001.

[11] D. A. Jimenez. "Piecewise linear branch prediction". 32nd International Symposium on Computer Architecture (ISCA'05), pages 382-393, June 2005

[12] D. Tarjan, K. Skadron, "Merging path and g-share indexing in perceptron branch prediction", ACM Transactions on Architecture & Code Optimization, vol.2, no.3, pages 80-300, 2005.

[13] A. Seznec, Revisiting the perceptron predictor. Technical Report PI-1620, IRISA Report, May 2004.

[14] H. Gao, H. Zhou, "Adaptive Information Processing: An Effective Way to Improve Perceptron Branch Predictors", Journal of Instruction-Level Parallelism (JILP), vol.7, pages 1-10, 2005.

[15] G. E. Hinton, S. Osindero, et al., "A fast learning algorithm for deep belief nets", Neural Computation, vol.18, no.7, pages1527-1554, 2006.

[16] G. E. Hinton, R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", Science, vol.313, no.5786, pages 504–507, 2006.

[17] Y. LeCun, L. Bottou, et al., "Gradient-based learning applied to document recognition", Proceedings of the IEEE, vol.86, no.11, pages 2278–2324, 1998.

[18] K. He, X. Zhang, et al., "Deep Residual Learning for Image Recognition", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770 – 778, June 2016.

[19] A. Krizhevsky, I. Sutskever, et al., "ImageNet Classification with Deep Convolutional Neural Networks". Advances in Neural Information Processing Systems, vol. 25, no. 2, pages 1097 – 1105, 2012.

[20] Q. Jia, E. Shelhamer, et al., "Caffe: Convolutional Architecture for Fast Feature Embedding", ACM International Conference on Multimedia, pages 675-678, June 2014.

[21] M. Qureshi. CBP start. 4th JILP Workshop on Computer Architecture Competitions, 2014.