IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 22, 2017, accepted October 23, 2017, date of publication November 10, 2017, date of current version March 15, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2772334

A Study on Deep Belief Net for Branch Prediction

YONGHUA MAO[1,2], JUNJIE SHEN[3], AND XIAOLIN GUI[1]

[1]School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[2]School of Science, Xi'an Polytechnic University, Xi'an 710048, China
[3]Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695 USA

Corresponding author: Xiaolin Gui (xlgui@mail.xjtu.edu.cn)

ABSTRACT Since 2006, there have been significant advances in deep learning algorithms, and they have shown superior performance in audio and image processing. In this paper, we explore the feasibility of applying deep learning algorithms to branch prediction. We treat branch prediction as a classification problem and compare the effectiveness of deep learning with existing branch predictors. We make several interesting observations from our study. The first is that for branch prediction, the deep learning algorithm based on deep belief networks outperforms the prior work, but only outperforms state-of-the-art branch predictors, such as the TAgged GEometric length (TAGE) predictors, for several benchmarks. Compared with the much simpler perceptron branch classifier, the deep learning classifier reduces the average misprediction rate by 3%–4% for the benchmarks in this paper. Second, we analyze the impact of the length of hashed program counter, local history register, global history register, and branch global addresses of deep learning classifiers on the misprediction rate. Our results show that an adaptive length of the history information is a better choice than the longest history. Third, compared with TAGE, the hardware budget of our model is less than 1% of the TAGE predictor.

INDEX TERMS Branch predictor, perceptron, misprediction rate, DBN, deep learning.

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 trend of deeper pipelines and larger instruction windows as well as the emphasis on energy efficiency, branch prediction is a vital component of a microprocessor.

Given the importance, branch prediction has been studied extensively. Among the proposed branch predictors in the last five Championship Branch Prediction (CBP) competitions, most are variants of the TAgged GEometric length (TAGE) and 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 machine learning, deep learning in particular, it is worthwhile to examine whether the more advanced deep neural networks can discover new possibility for branch prediction.

This paper explores the feasibility of applying deep learning in branch prediction. We treat branch prediction as a classification problem, and compare the effectiveness of deep learning algorithms with state-of-the-art branch predictors. We make several interesting observations from our study. First, we confirm that deep learning outperforms perceptron in terms of classification ability. By comparing different neural network structures, we find deep learning is a good choice for branch prediction. Second, we analyze the impact of the different lengths of hashed PC address. We also analyze the impact of the length of history information, including the local history register (LHR), the global history register (GHR), and global addresses (GA) [2], which means the most recently executed branch addresses, on the misprediction rate of deep learning classifier. Our simulation results show that the longest LHR, GHR, and GA do not always lead to the lowest misprediction rate for deep learning based predictors. We found DBN predictor show lower misprediction rate than state-of-the-art branch predictors only for several benchmarks. At last, compared to TAGE, the hardware budget of our model is less than 1% of the TAGE predictor.

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 the concepts of deep learning and perceptron. Section 4 describes our simulation methodologies and benchmarks. In Section 5, we present the comparison results and analysis of deep learning and

other related predictors. Section 6 concludes and discusses the future work.

II. STATE OF THE ART BRANCH PREDICTOR

The TAGE branch predictor is often considered as the lowest misprediction rate branch predictor. TAGE is derived from Seznec's GEHL predictor [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 and 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 a specific type of hard-to-predict branches. For example, TAGE-SC-L [5] improves the TAGE by adding a statistical corrector predictor and a loop predictor [6], [7]. The idea of a statistical corrector predictor is to revert the prediction of the TAGE when it statistically mispredicted in similar branch circumstances. The loop predictor targets at loop branches and uses the loop counts to make accurate predictions.

Recently, Seznec and Michaud [8] further improved the TAGE misprediction rate via a Multi-poTAGE predictor. It combines multiple TAGE predictors and the final prediction is selected from these predictors via a combined output lookup table (COLT) predictor [9]. Each TAGE component takes a different combination of history, including both global and local history as input. This colossal multiple-TAGE predictor is unrealistic and used to push the lower bound of misprediction rate of TAGE-like predictors.

Another type of state-of-the-art branch predictors [10] is the perceptron predictor. 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 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 branch 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 with 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 address of each bit in the global history. However, both perceptron and piecewise perceptron 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 and Skadron [12] proposed that this effect could be eliminated by a hashed perceptron, in which multiple history bits are hashed to a single weight.

III. PERCEPTRON AND DEEP LEARNING

Perceptron is one of the simplest neural network models. It is a linear classifier algorithm for supervised classification [13]. Many branch predictors achieve low misprediction rate by correlative perceptron classifiers [14]–[17]. With recent advances in deep learning showing highly impressive misclassification rate for image or audio based processing, in this work, we study the feasibility of applying deep learning 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, are 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. [18] proposed the Deep Belief Networks (DBN) [19] and an efficient way to train the network [20]. A DBN [21]–[23] 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 [24] learning on every RBM. This step [25], [26] is a way of unsupervised feature learning. After pre-training, global training algorithms such as back-propagation are used to fine-tune weights in the network.

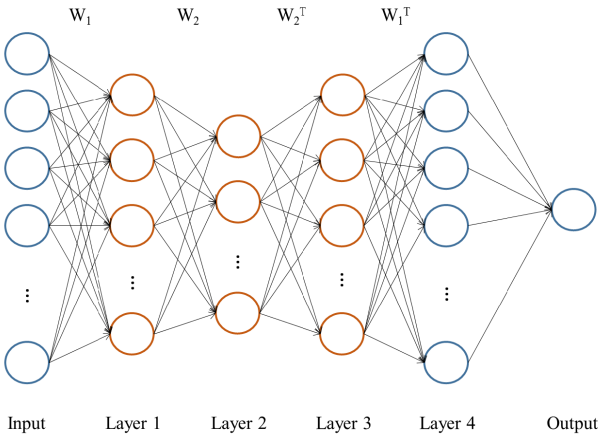


FIGURE 1. A popular DBN structure (WN is the weight between two layers).

A commonly used DBN [27] structure has four RBM layers and an output layer, as shown in Fig. 1. The number of neurons in Layer 1 is around $1/3 \sim 2/3$ of the inputs. Similarly, the number of neurons in Layer 2 is around $1/3 \sim 2/3$ of Layer 1. Layer 3 has the same size as Layer 1, and Layer 4 has the same size as the input layer. The last layer is the output layer, which is constructed as a simple

TABLE 1. Benchmarks.

Benchmarks	Dynamic Conditional Branches	Benchmarks	Dynamic Conditional Branches
F1	2213673	L0	25181955
F2	1792835	L1	25323638
F3	1546797	L2	22628704
F4	895842	L3	16754009
F5	2422049	L4	31520616
M1	2229289	L5	9409564
M2	3809780	L6	27139020
M3	3014850	L7	23532921
M4	4874888	L8	14565465
M5	2563897	L9	20449090
I1	4184792	L10	14312999
I2	2866495	L11	16145141
I3	3771697	L12	19679814
I4	2069894	L13	27946011
I5	3755315	L14	29462517
S1	3660616	L15	16836233
S2	3537562	L16	22064822
S3	3811906	L17	14796021
S4	4266796	L18	19691402
S5	4291964	L19	14435009

single-layer neural network, i.e., a perceptron. Although the network has 4 RBM layers, the training process is not exceedingly complex. Only W_1 and W_2 will be trained. W_3 and W_4 are the transposes of W_2 and W_1 as in shown Fig. 1. 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. 1) 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.

IV. EVALUATION METHODOLOGY

In order to evaluate various algorithms for branch prediction, we adopt the simulation framework provided in the 4th Championship Branch Prediction [28]. The framework is based on trace-driven simulation and features 20 short benchmarks, which are grouped into 4 categories: I (integer), F (floating point), M (multimedia) and S (server). Each benchmark contains approximately 30 million instructions, including both user and system activities. In this work, we focus on the conditional branches from each trace as

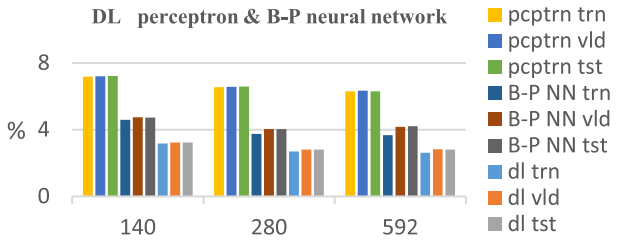


FIGURE 2. Average misprediction rate of perceptron (pcptrn), B-P NN, and DBN on the short traces.

listed in Table I. We also use the long traces, which are extracted from the SPEC CPU2006 (L) benchmarks, in the most experiments in the section 5. It seems that we can reach the same conclusion for both short and long traces.

Table II shows the input sizes that are used in our experiments. The inputs include branch PCs, GHRs, LHRs, and GAs. We explore the design space of the length of each input component to capture the correlation of component length and misprediction rate. Recently, the machine-learning accelerators achieve significant advances [29]–[36]. It gradually becomes a new research focus. The hardware implementation of deep learning branch predictors could be designed in similar fashion. In this paper, the problem of branch prediction is treated as a pure binary classification problem. Since this paper evaluates deep learning at the algorithm level, we do not focus on the hardware design of the predictor. We are not constrained by hardware resources. We use an offline training method with a training set of 90% branches, a validation set of 5% branches, and a testing set 5% branches in each trace. The dynamic conditional branches counts in each trace are shown in Table I. The training set is used to train the network. The validation set is used to estimate how good the network has been trained in the training progress. If the network is good enough, meaning that the misclassification rate on the validation set is not decreased in the latest several iterations, the training process will be stopped. Then, the test set is used to evaluate the final classification error rate after the network has been trained.

In our experiments, we evaluated different DBN structures as shown in Table II. They cover both shallow and deep neural networks. All these DBN and Backpropagation neural network (BP NN) [26] configurations are selected from a thorough search in the large design space of their structures. All DBN architectures are Popular DBNs. In order to compare with DBN conveniently, the offline models are used in our perceptron branch classifiers.

V. RESULTS AND DISCUSSION

In this work, we evaluate the misprediction rate of the training set, the validation set, and the testing set. The misprediction rate of the testing set is our primary concern, so the default is on the testing set.

A. COMPARING PERCEPTRON, BP NN, AND POPULAR DBN

Fig.2 presents the misprediction rate of the perceptron, the DBN, and the B-P NN classifier for different history

TABLE 2. Configurations of different DBN structures.

Part	Architectures	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
	BP NN1						140 – 40 – 1
	DBN 2	280	prior 32 bits PC		200	16	280 – 140 – 85 – 140 – 280 – 1
	BP NN2						280 – 85 – 1
	DBN 3	592	prior 32 bits PC		512		592 – 260 – 130 – 260 – 592 – 1
	BP NN3						592 – 130 – 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 Last	944	48 8-bit	32	512	16	944-630-270-630-944-1

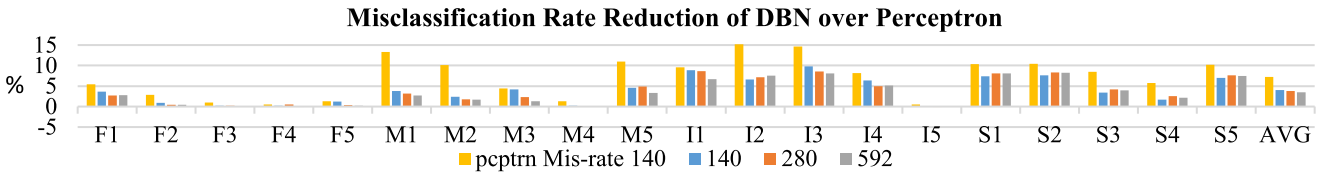


FIGURE 3. The misprediction rate reduction of DBN over perceptron on the testing set of the CBP-4 short trace.

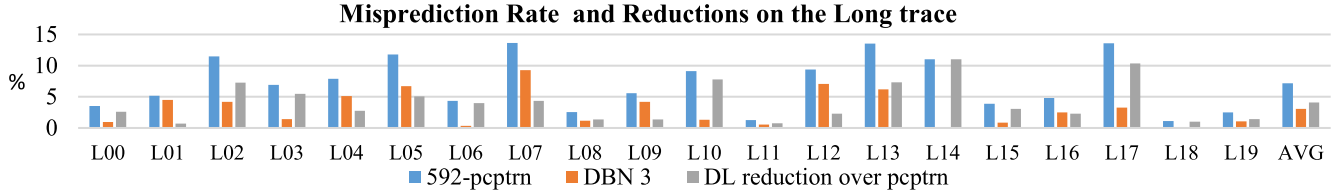


FIGURE 4. The misprediction rate reduction of DBN over BP NN on the testing set of the SPEC CPU2006 benchmarks.

information lengths. The misprediction rate is the geometric mean across the short 20 benchmarks. The results for training set are labeled 'trn', validation set labeled 'vld', and testing set labeled 'tst'. As we can see from Fig. 2, deep learning approaches achieve lowest misprediction rate. For the DBN and BP NN with the 280-inputs achieves the lowest average misprediction rate for validation sets and testing sets. For the perceptron model, the longer the history information length is, the lower the average misprediction is for all sets.

The reductions of misprediction rate DBN over perceptron for each benchmark are shown in the Fig. 3. On the benchmark 'I3', the DBN 1 reduces the mispredictioin rate by 9.811% compared to the corresponding perceptron. Compared to perceptron, the average misprediction rate of the corresponding DBN predictor is reduced by 3.774%. The lower the misprediction rate of the perceptron, the less the reduction, and vice versa. Different from linear classifiers, such as the perceptron, the DBN could reduce the misprediction rate on non-linear separable branches. If the benchmark is dominated by linear separable branches, such as 'F3' and 'I5', the reduction is little or negligible.

Fig.4 presents the misprediction rate of the DBN, perceptron, and the corresponding reductions on the long traces. Both DBN and perceptron use a history information length of 592 bits. The reductions are over 10% between the popular DBN and the perceptron for two long traces benchmarks, 'L14' and 'L17'. The average reduction is 4.112%.

Fig. 5 presents the reductions of misprediction rate of DBN over BP NN. On benchmark 'M2', 'I2', 'S1', 'S2', and 'S5', there are over 3-4% reduction between them. On the linear separable benchmarks, the reductions are also little.

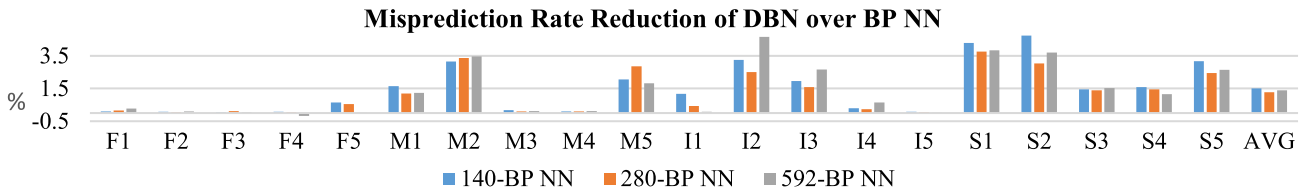


FIGURE 5. The misprediction rate reduction of DBN over BP NN on the testing set of the CBP-4 short traces.

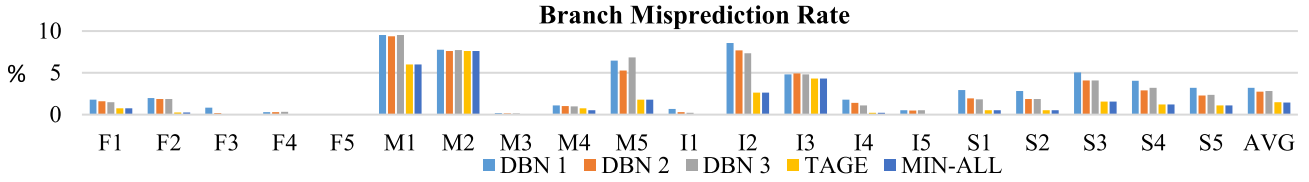


FIGURE 6. The branch misprediction rate of the DBNs, TAGE, and MIN-ALL on the testing set of CBP-4 short traces.

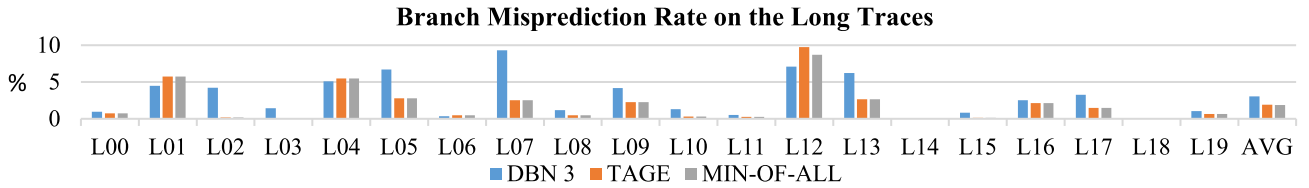


FIGURE 7. The branch misprediction rate of the DBN, TAGE, and the MIN-ALL on the testing set of the SPEC CPU2006 benchmarks.

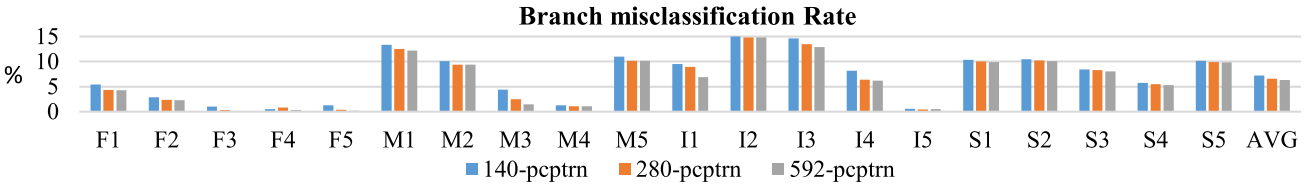


FIGURE 8. The perceptron branch misprediction rate of perceptrons on the testing set of CBP-4 short traces.

B. COMPARING DEEP LEARNING PREDICTOR OF THE DIFFERENT LENGTHS OF HISTORY BITS AND PC ADDRESSES WITH THE TAGE PREDICTOR

Fig. 6 shows the misprediction rate of different DBN models, the TAGE predictor, and the MIN-ALL, on the short traces. We include the results of popular DBN1, popular DBN2, and popular DBN3 on the testing set. Fig. 7 shows them on the long traces. The TAGE, represents for the multi-poTAGE-SC, which is the unlimited hardware resource winner of the CBP-4 champion; the MIN-ALL is the lowest of all predictors for each short benchmark from this champion, and represents the state-of-the-art in the unlimited hardware resources group.

From Fig. 6, it can be seen that all DBN classification models could not outperform the TAGE predictor on any short trace. From Fig. 7, there are only 4 long trace benchmarks, for which the DBN model has lower misprediction rates than the TAGE. This shows that for branch prediction, the pure DBN approaches are not as effective as state-of-the-art branch predictors, which are based on tag matching. Another interesting observation is the impact of the history information length from Fig. 6. There are 18 benchmarks, for which the DBN 2 has lower misprediction rates than the DBN1, and also 12 benchmarks, for which the DBN3 has lower misprediction rates than the DBN2. Only for half of the short benchmarks, the reduction of misprediction rate is positive correlated to the length of the history information.

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.

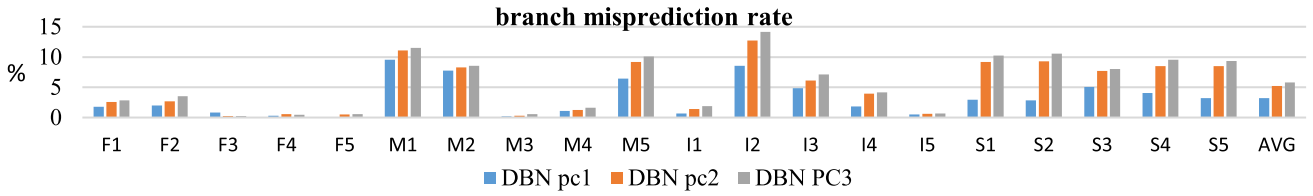


FIGURE 9. The effect of the hashed PC of DBN branch misprediction rate on CBP-4 short traces.

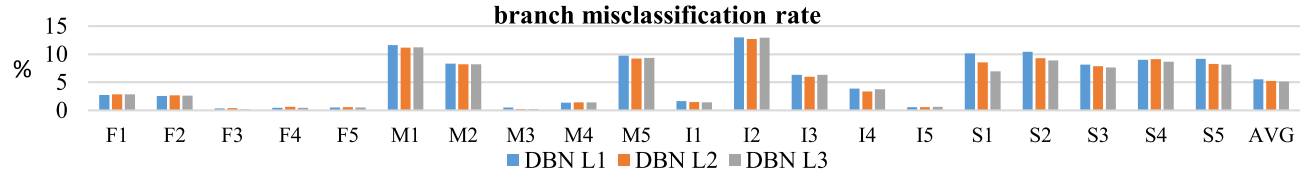


FIGURE 10. The effect of the length of the LHR of DBN branch misprediction rate on CBP-4 short traces.

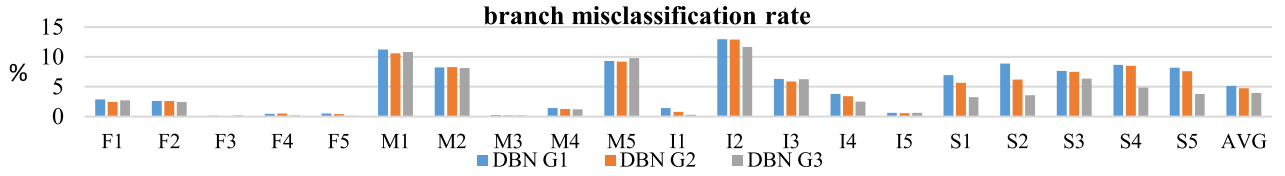


FIGURE 11. The effect of the length of the GHR of DBN branch misprediction rate on the testing set of CBP-4 short traces.

This indicates that adding global addresses is profitable in perceptron predictors. We discuss the effect of PCs, LHRs, GHRs, and Gas in the DBN model separately in the part C.

C. COMPARING DEEP LEARNING IN THE DIFFERENT LENGTH 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 pc2 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. It implies that the address memorization [37] is not obvious on our offline training DBN models.

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 12-bit LHR has lower misprediction rates than the DBN with 8-bit LHR, and 10 benchmark, for which the DBN with 16-bit LHR has lower misprediction rates than the one with 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 addressand 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] with 48 8-bit GA 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 could outperform the piecewise perceptron on 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).

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

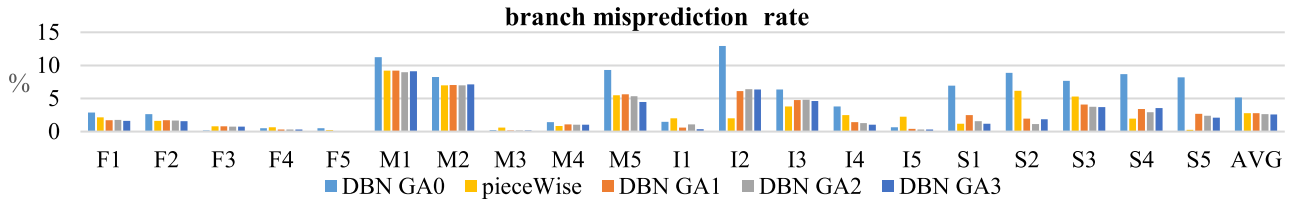


FIGURE 12. The effect of the length of the GA of DBN branch misprediction rate on the testing set of CBP-4 short traces.

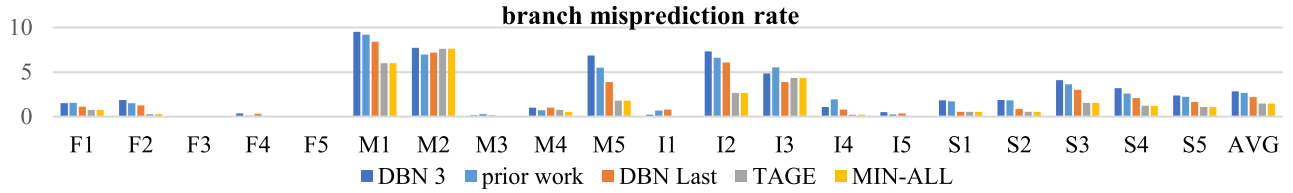


FIGURE 13. The branch misprediction rate of DBN3, the prior work, DBN Last, and MIN-ALL on the testing set of CBP-4 short trace.

DBN (labeled as DBN Last) 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 Last is lower than AIP [14] for 14 benchmarks while the misprediction rate of the DBN1 is higher than AIP for 16 benchmarks. Although there are 2 benchmarks where the DBN Last is lower than the TAGE and the MIN-ALL, but it reduces by 0.645% compare to DBN3 on average. It worth to explore the available length history information DBN model to compare with TAGE again in the future work. It also seem that, the misprediction rate decrease up to the increasing of the length of the history information for most of benchmarks. So local connected networks with the longest history information, such as convolution neural network (CNN) [38], is likely to be another better choice in the future work.

D. COMPARING THE HARDWARE BUDGETS OF THE PIECEWISE PERCEPTRON, THE PRIOR WORK, THE POPULAR DBN PREDICTOR, AND THE TAGE

The piecewise perceptron and the prior work are practical hardware implementation predictors. The TAGE is consist of five poTAGE and one statistical corrector predictor. The total hardware budget is about 5000 M bits. For the DBN, because we only use it as a static predictor, no hardware is need on the backpropagation logic. Hence, the majority of the hardware budget is spent on the weights of the network. For example, if the network configuration is 516-305-195-305-516-1, there are 434226 weights, which consume about $434,226 * 32$ bits $= 13,895,232$ bits. Adding to the input data consumption, the total budget is about 13.9M bits. It is less than 1% of the TAGE.

VI. CONCLUSIONS

This paper takes a binary classification perspective to explore the branch prediction problem. We apply deep learning as a classifier. The first observation is pure deep learning algorithm outperform our prior work, but does not outperform state-of-the-art predictors, or lower only on several benchmarks. Compare to the perceptron, which is commonly used together with other branch prediction approaches, the DBN reduces by 3.774% and 4.112% on average across the short and long benchmarks respectively. We discuss the impact of the length of hashed PC, LHR, GHR, and GA on the misprediction rate in the DBN. Our simulation shows that it is not always the case that longer LHR, GHR, and GA would always produce lower misprediction rate. For some cases, a shorter length may achieve better results. So an adaptive length of the history information might be a good choice in the DBN model. At last we comment on the hardware budget of DBN versus the piecewise perceptron, the prior work, and the TAGE. The budget of DBN predictor is less than 1% of the TAGE.

VII. FUTURE WORKS

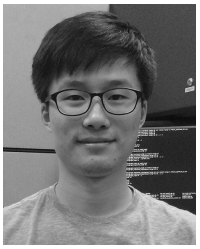
This paper takes the branch prediction as a pure binary classification stochastic problem. In order to simplify the problem, we only implemented offline training. However, a viable branch predictor must use only previous history information to train it. In order to apply deep learning in branch prediction, an online training algorithm may be employed. Since most of the state-of-the-art branch predictors are integrated of several standalone predictors, it is also worthwhile to explore the influence of incorporating other complementary predictors into the deep learning predictor. The CNN model with the longest history information is another experiment choice in the future. Several machine learning accelerators have been proposed in [29]–[36]. The hardware implementation of deep learning branch predictors would be designed in similar fashion in the future.

REFERENCES

[1] D. Gope and M. H. Lipasti, "Bias-free neural predictor," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2014, pp. 521–532.

[2] R. Nair, "Dynamic path-based branch correlation," in *Proc. 28th Annu. Int. Symp. Microarchitecture*, Dec. 1995, pp. 15–23.
[3] A. Seznec and P. Michaud, "A case for (partially) TAgged GEometric history length branch prediction," *J. Instruct.-Level Parallelism*, vol. 8, pp. 1–23, Jan. 2006.
[4] P. Michaud, "A PPM-like, tag-based branch predictor," *J. Instruct.-Level Parallelism*, vol. 7, pp. 1–10, Apr. 2005.
[5] A. Seznec, "TAGE-SC-L branch predictors," in *Proc. 4th Championship Branch Predict. (ISCA)*, Jun. 2014, pp. 1–8.
[6] A. Seznec, "A 64 Kbytes ISL-TAGE branch predictor," in *Proc. 3rd Championship Branch Prediction*, Jun. 2011, pp. 13–16.
[7] A. Seznec, "A new case for the TAGE branch predictor," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2011, pp. 117–127.
[8] A. Seznec and P. Michaud, "Pushing the branch predictability limits with the multi-poTAGE+SC predictor," in *Proc. 4th Championship Branch Predict. (ISCA)*, Jun. 2014, pp. 1–4.
[9] G. H. Loh and D. S. Henry, "Predicting conditional branches with fusion-based hybrid predictors," in *Proc. Int. Conf. Parallel Archit. Compilation Techn.*, Sep. 2002, pp. 165–176.
[10] D. A. Jimeénez and C. Lin, "Dynamic branch prediction with perceptrons," in *Proc. Int. Symp. High-Perform. Comput. Archit.*, Aug. 2001, pp. 197–206.
[11] D. A. Jimeénez, "Piecewise linear branch prediction," in *Proc. 32nd Int. Symp. Comput. Archit. (ISCA)*, 2005, pp. 382–393.
[12] D. Tarjan and K. Skadron, "Merging path and gshare indexing in perceptron branch prediction," *ACM Trans. Archit. Code Optim.*, vol. 2, no. 3, pp. 280–300, 2005.
[13] L. N. Kanalm "Perceptrons," *Int. Encyclopedia Social Behavioral Sci.*, vol. 39, no. 4, pp. 11218–11221, 2001.
[14] H. Gao and H. Zhou, "Adaptive information processing: An effective way to improve perceptron branch predictors," *J. Instruct.-Level Parallelism*, vol. 7, pp. 1–10, Apr. 2005.
[15] D. A. Jimeénez, "Generalizing neural branch prediction," *ACM Trans. Archit. Code Optim.*, vol. 5, no. 4, p. 17, 2009.
[16] D. A. Jiménez, "Fast path-based neural branch prediction," in *Proc. 36th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2003, pp. 243–252.
[17] A. Lifa, P. Eles, and Z. Peng, "Dynamic configuration prefetching based on piecewise linear prediction," in *Proc. Design. Autom. Test Europe Conf. Exhib. (DATE)*, Mar. 2013, pp. 815–820.
[18] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.
[19] G. E. Hinton, "Learning multiple layers of representation," *Trends Cognit. Sci.*, vol. 11, no. 10, pp. 428–434, Oct. 2007.
[20] G. Hinton, "A practical guide to training restricted Boltzmann machines," *Momentum*, vol. 9, no. 1, p. 926, 2012.
[21] G. E. Hinton, "Boltzmann machine," *Encyclopedia Mach. Learn.*, vol. 2, no. 5, p. 1668, 2007.
[22] A. Fischer and C. Igel, "An introduction to restricted Boltzmann machines," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Berlin, Germany: Springer, 2012, pp. 14–36.
[23] A. Fischer and C. Igel, "Empirical analysis of the divergence of Gibbs sampling based learning algorithms for restricted Boltzmann machines," in *Proc. Int. Conf. Artif. Neural Netw.*, vol. 6354. 2010, pp. 208–217.
[24] M. Á. Carreira-Perpigñán and G. E. Hinton, "On contrastive divergence learning," *Artif. Intell. Stat.*, vol. 10, pp. 33–40, Jan. 2008.
[25] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[26] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[27] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[28] (2014). *The 4th Championship on Branch Prediction*. [Online]. Available: http://www.jilp.org/cbp2014/
[29] T. Chen *et al.*, "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *Acm Sigplan Notices*, vol. 49, no. 4, pp. 269–284, 2014.
[30] Y. Chen *et al.*, "DaDianNao: A machine-learning supercomputer," in *Proc. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2014, pp. 609–622.
[31] D. Liu *et al.*, "Pudiannao: A polyvalent machine learning accelerator," *ACM SIGARCH Comput. Archit. News*, vol. 43, no. 1, pp. 369–381, Mar. 2015.
[32] X. Zhang, Q. Guo, Y. Chen, T. Chen, and W. Hu, "HERMES: A fast cross-ISA binary translator with post-optimization," in *Proc. IEEE/ACM Int. Symp. Code Generat. Optim.*, Feb. 2015, pp. 246–256.
[33] X. Yuan *et al.*, "ReCBuLC: Reproducing concurrency bugs using local clocks," in *Proc. 37th Int. Conf. Softw. Eng.*, May 2015, pp. 824–834.
[34] Z. Du *et al.*, "ShiDianNao: Shifting vision processing closer to the sensor," in *Proc. 42nd ACM/IEEE Int. Symp. Comput. Archit. News (ISCA)*, Jun. 2015, pp. 92–104.
[35] Z. Du *et al.*, "Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches," in *Proc. 48th ACM/IEEE Int. Symp. Microarchitecture*, Dec. 2017, pp. 494–507.
[36] S. Liu *et al.*, "Cambricon: An instruction set architecture for neural networks," in *Proc. 43rd ACM/IEEE Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 393–405.
[37] M. Qureshi, "CBPstart," in *Proc. 4th JILP Workshop Comput. Archit. Competitions, Championship Branch Prediction ISCA*, Minneapolis, MN, USA, Jun. 2014.
[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, no. 2, pp. 1097–1105, 2012.

YONGHUA MAO is currently pursuing the Ph.D. degree with the Department of Computer Science, Xi'an Jiaotong University. He was a Visiting Scholar for Dr. H. Zhou in computer engineering with North Carolina State University. He is a Teacher with Xi'an Polytechnic University. His research interests include branch prediction, deep learning, and machine learning.



JUNJIE SHEN received the M.Sc. degree in computer engineering from North Carolina State University. He is currently pursuing the Ph.D. degree in computer science with the University of California at Irvine. His research interests include compiler and architectural support for security and emerging applications.

XIAOLIN GUI received the B.Sc. degree from Xi'an Jiaotong University (XJTU), China, in 1988, and the M.Sc. and Ph.D. degrees in computer science from XJTU, in 1993 and 2001, respectively. Since 1988, he has been with XJTU as an Active Researcher involved in network computing, network security, and wireless networks. He has been the Director with the Key Lab of Computer Network, China, since 2008. He has served as a Deputy Dean of the School of Electronic and Information since January 2013. He is currently a Professor. His recent research covers secure computation of open network systems (include Grid, P2P, Cloud), dynamic trust management theory, and development on community network. He was a recipient of the New Century Excellent Talents in Universities of China.