IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 22, 2020, accepted August 11, 2020, date of publication August 17, 2020, date of current version August 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3017196

Exploring Convolution Neural Network for Branch Prediction

YONGHUA MAO[1,2,3], HUIYANG ZHOU[2], (Senior Member, IEEE), XIAOLIN GUI[3], AND JUNJIE SHEN[2,4]

[1]School of Science, Xi'an Polytechnic University, Xi'an 710048, China
[2]Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695, USA
[3]School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China
[4]Department of Computer Science, University of California at Irvine, Irvine, CA 92697, USA

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

This work was supported in part by the International Cooperation on Science and Technology of the Key Research and Development Program of Shaanxi Province under Grant 2018KW-021, in part by the National Key Research and Development Program of China under Grant 2018YFB1800304, and in part by the Key Research and Development Program of Shaanxi Province under Grant 2019GY-005.

ABSTRACT Recently, there have been significant advances in deep neural networks (DNNs) and they have shown distinctive performance in speech recognition, natural language processing, and image recognition. In this paper, we explore DNNs to push the limit for branch prediction. We treat branch prediction as a classification problem and employ both deep convolutional neural networks (CNNs), ranging from LeNet to ResNet-50, and deep belief network (DBN) for branch prediction. We compare the effectiveness of DNNs with the state-of-the-art branch predictors, including the perceptron, our prior work, Multi-poTAGE+SC, and MTAGE+SC branch predictors. The last two are the most recent winners of championship branch prediction (CBP) contests. Several interesting observations emerged from our study. First, for branch prediction, the DNNs outperform the perceptron model as high as 60–80%. Second, we analyze the impact of the depth of CNNs (i.e., number of convolutional layers and pooling layers) on the misprediction rates. The results confirm that deeper CNN structures can lead to lower misprediction rates. Third, the DBN could outperform our prior work, but not outperform the state-of-the-art TAGE-like branch predictor; the ResNet-50 could not only outperform our prior work, but also the Multi-poTAGE+SC and MTAGE+SC.

INDEX TERMS Branch prediction, CNN, deep learning, VGG, ResNet.

I. INTRODUCTION

The ever-increasing computational power is a key factor for recent advances in deep learning. In microprocessor design, pipelining is a critical technique for the high performance as it facilitates high clock rates and instruction-level parallelism (ILP). As pipelines become deeper, the control hazard due to conditional branches (e.g., as a result of an if-then-else structure) incurs higher performance overhead. The reason is that the outcome of the conditional branch, i.e., 'Taken' or 'Not Taken', is computed late in the pipeline, i.e., the execution stage after it is fetched, decoded, and dispatched. In this case, the pipeline would have to wait for the branch outcome before the next instruction can be fetched. To avoid this situation, branch prediction, which guesses the branch outcome before it is executed, is commonly used to continue fetching and executing instructions along the predicted paths. And the accuracy of predicting the execution results of branches directly affects the performance of deep pipelined processors as a result of the severe performance penalty of mispredictions.

Due to its importance, branch prediction has been a focus in computer architecture research [1]. The accuracy of the traditional predictors, such as 2-bit Counter, Yeh/Patt [2], Agree [3], Filter [4], and YAGS [5] branch predictor, is up to 95%. The bottleneck of this accuracy has arisen in the traditional hybrid predictor. The improvement of both performance and energy efficiency of the contemporary super-deep pipelined processor could be a lot, even if improved branch prediction accuracy is very little [6]. In fact, even for modern architectures, an average of 30% total instructions executed in an integer program are wasted due to branch misprediction [1].

There have been a great deal of research and practice on branch prediction in academia and the industry and new branch prediction algorithms [7]–[12] are emerging. The global Championship Branch Prediction (CBP)

The associate editor coordinating the review of this manuscript and approving it for publication was Chunbo Xiu.

is organized by the International Symposium on Computer Architecture (ISCA) or International Symposium on Microarchitecture (MICRO) every few years. In the past five championships, the high performing branch predictors could be divided into two primary categories: one is based on the TAGE (TAgged GEometric history length) branch predictor; the other is based on neural networks. The TAGE, which is commonly considered as the lowest misprediction rate predictor, is derived from Seznec's GEHL predictor [13] and Michaud's tagged PPM-like predictors [14]. TAGE-SC-L [15] adds a statistical corrector predictor and a loop predictor. Multi-poTAGE+SC predictor [16], which is the champion of the unlimited resource group of CPB-4 in 2014, combines multiple TAGE predictors and the final prediction is selected from these predictors via a combined output lookup table [17] (COLT) predictor. This colossal multiple-TAGE predictor and its further fine-tuned version MTAGE+SC [18], which is the champion of the unlimited resource group of CPB-5 in 2016, 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 [19]–[21]. It uses a single-layer perceptron, one of the simplest neural networks to learn the correlation between the branch history and branch outcomes. It confirms that neural networks can be useful in branch prediction. However, only a few works [22] explored more advanced machine learning methods on the branch prediction problem, much less deep neural networks. Recently, there have been significant advances in deep learning [23] and they have shown superior performance in image recognition [24], [25], speech recognition [26] and natural language processing [27]. It is worthwhile to examine whether the more advanced deep neural networks can discover new possibilities for branch prediction.

This paper explores deep neural networks for branch prediction by treating it as a classification problem. We explore both deep belief network (DBN) 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 actual predictor designs. It is consistent with the CBP competition rules that "CBP will make no attempt to assess algorithm-level exploration for predictors the unlimited storage budget". We make several interesting observations from our study. First, we confirm that deep learning outperforms the perceptron predictor. Second, between DBN and CNNs, we find CNNs is 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 have lower misprediction rates. Fourth, we found deep CNN algorithms could achieve lower misprediction rates than state-of-the-art branch predictors, Multi-poTAGE+SC and MTAGE+SC, with unlimited storage limits.

The rest of the paper is organized as follows: Section 2 reviews perceptron branch predictors, and describes the DBN and CNN predictor model. Section 3 describes our experimental methodologies and benchmarks. In Section 4, we present the comparison results and analysis of DNNs and other related predictors. Section 5 concludes our work, and section 6 discusses the future work.

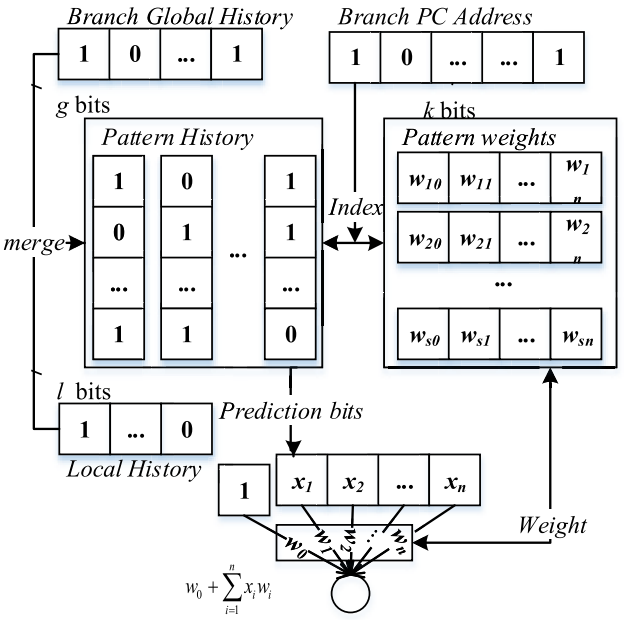


FIGURE 1. A perceptron-based predictor model.

II. PERCEPTRON PREDICTOR AND DEEP LEARNING

Perceptron is widely used to build branch predictors, which achieves low misprediction rate. Perceptron uses a layer of hand-coded features and tries to recognize objects by learning how to weight these features. 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 [28] 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 and Skadron [20] proposed that this side effect could be mitigated by a hashed perceptron, in which multiple history bits are hashed to a single weight.

A. PERCEPTRON-BASED PREDICTOR

A perceptron-based predictor 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 address (PC) [21], as shown in Fig.1. 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 which is merged from local and global histories (LHR/GHR [21])) 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 between 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.

A perceptron-based predictor is essentially a perceptron binary classification model for each branch PC address. If the PC address is merged into branch history bits, as shown in Fig.2, the predictor becomes a classic input-data-to-output-binary-classification model. The middle black box may use any machine learning classification method.

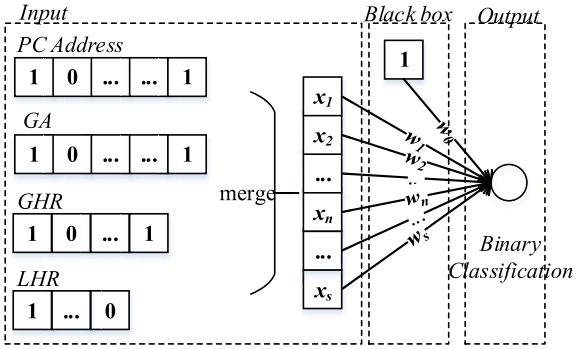


FIGURE 2. Branch p2p black box classification model.

B. DNN AND DNN-BASED PREDICTOR MODELS

With recent advances in deep learning showing highly impressive misclassification rate for image or audio-based processing [29], we aim to further push the lower bound of misprediction rates by applying deep learning algorithms to branch prediction. Deep learning [23] 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. Deep belief network (DBN) and convolution neural network (CNN) are the focus of this work.

1) DBN AND DBN-BASED PREDICTOR MODEL

DBN [30] is a class of deep neural networks, composed of multiple layers of hidden units, with connections between the layers but not between units within each layer. A DBN is built from 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 [31] 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.

For branch prediction, as shown in Fig.3, the input and output are same as those to the perceptron-based predictor and the classification black box is replaced with 4 fully-connected layers of hidden units.

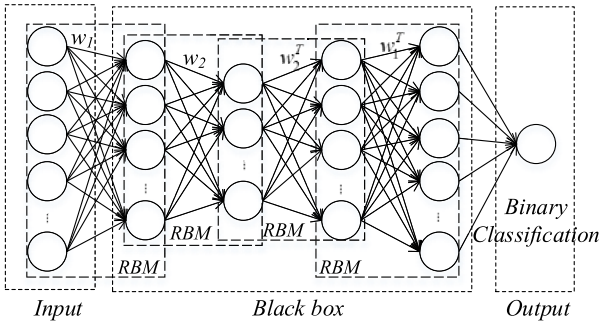


FIGURE 3. The structure of DBN (W_N is the weight between two layers).

2) CNN AND CNN-BASED PREDICTOR MODEL

Locally connected networks with the longest history bits [22], such as convolution neural networks (CNNs) [32], [33], are likely to be a better choice for branch prediction as most branches show high correlation with nearby history. CNNs [29], [34]–[38] are deep feedforward neural networks. The vital components of a CNN architecture are convolution layers and pooling layers, as shown in Fig.4. They exploit the high correlation in local groups of data. The convolution layer [39] 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. As shown in Fig.4, for branch prediction, the input and output are also same as those to the perceptron-based predictor while the classification black box is replaced with convolution layers, pooling layers, and several fully-connected layers of hidden units.

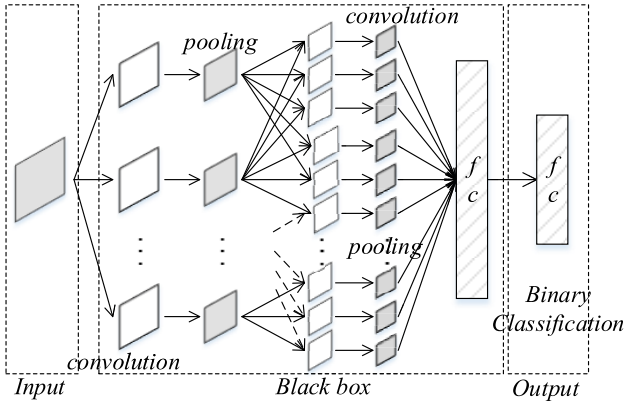


FIGURE 4. The structure of convolution neural network.

III. EVALUATION METHODOLOGY

A. THE DATASET

In order to evaluate various algorithms for branch prediction, we leverage the simulation framework provided in the

4th Championship Branch Prediction (CBP-4) which was held with 41th ACM/IEEE International Symposium of Computer Architecture(ISCA-41). 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 1. The number of branches shown in table 1 is the total number of sequential executions of all branches. For example, a same branch repeat to execute N times, N branches record is counted in the number of the branches on a trace. As shown in figure 5, the times of branches execution is only once, accounted for nearly 20%; and the times of branches execution is less than 10, accounted for nearly 50%. Compared with early high repeat branch data set [40], it was more difficult to predict. 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.

TABLE 1. Benchmarks.

Benchmark	Dynamic Conditional Branches	Benchmark	Dynamic Conditional 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

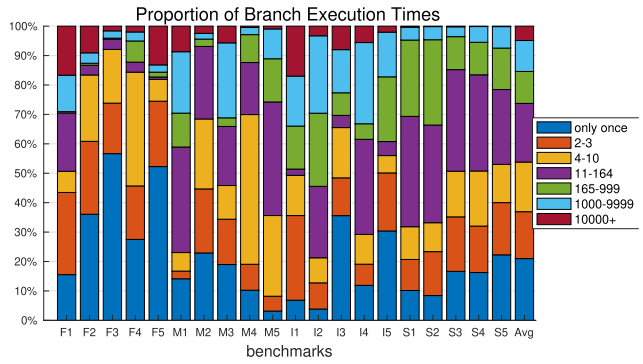


FIGURE 5. Proportion of branch execution times of CBP-4 traces.

In this paper, the problem of branch prediction is treated as a binary classification problem. We use an off-line training method with a training set of 90% random sampling branches, a validation set of 5% random sampling branches and a testing set of 5% random sampling branches. The training set is used to train the networks. The validation set is used to estimate how good a network has been trained during the training progress. If the network is good enough, meaning that the classification error rate on the validation set does not decrease in the latest several iterations, the training process will stop. Then, the test set is used to evaluate the final classification error rate after the network has been trained. Our DNNs are constructed, trained, and tested using the Caffe [41] framework.

B. THE ARCHITECTURE

We have already discussed the influence of the length of history bits for the misprediction rate of DBN models [22], and DBN with a 944 history bits get a lower misprediction rate. So, the 944 history bits are used as input date in our DNNs. Each 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 [19]. Every bit is an input to a neuron of the input layer of a DNN as shown in Fig 6. So all input to our models is a fixed-size 944 dimensional vector.

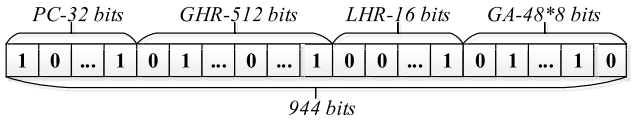


FIGURE 6. The input data structures of branch traces.

We have tested DBN and CNNs with various configurations, and have observed consistent results. To provide instances for discussion, we describe one DBN and four CNN models for branch prediction as follows.

Perceptron: Based on the perceptron branch model shown in Fig. 2, the input data is contain 944 history bits, and every bit of the input data is one neuron of the input layer of the perceptron model, and the output layer is a neuron of binary classification.

DBN: Based on a popular DBN structure [22], 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, as shown in Fig 7. Layer 3 has the same size as Layer 1 and Layer 4 is the same size as the input layer. The last layer is

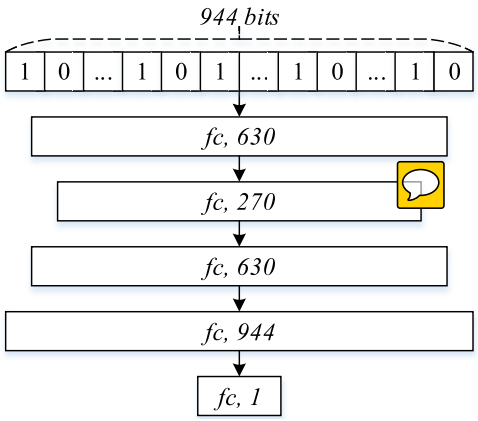


FIGURE 7. The architecture of DBN for branch prediction (fc means fully-connected).

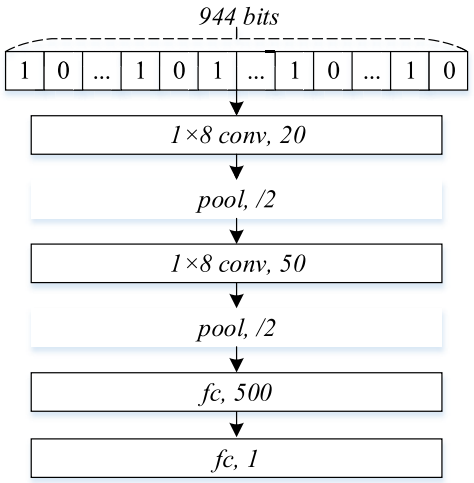


FIGURE 8. The architecture of LeNet for branch prediction (fc means fully-connected).

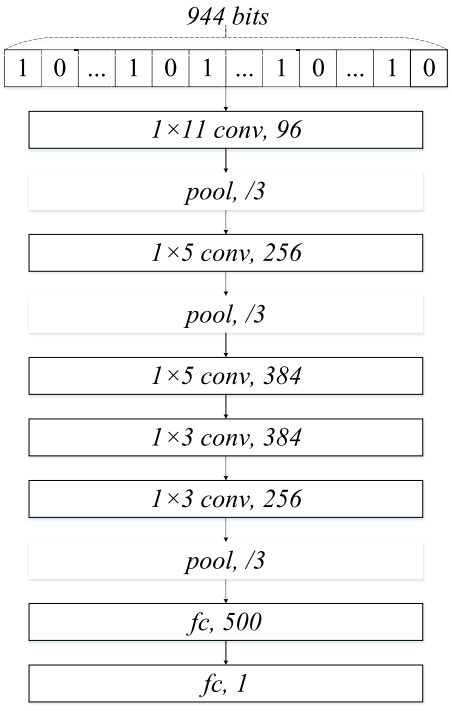


FIGURE 9. The architecture of AlexNet for branch prediction (fc means fully-connected).

the output layer, which is constructed as a simple single-layer neural network. The notation 'fc' means fully-connected.

LeNet: The first CNN branch prediction model is inspired by the LeNet [32], which contains two convolutional layers 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.

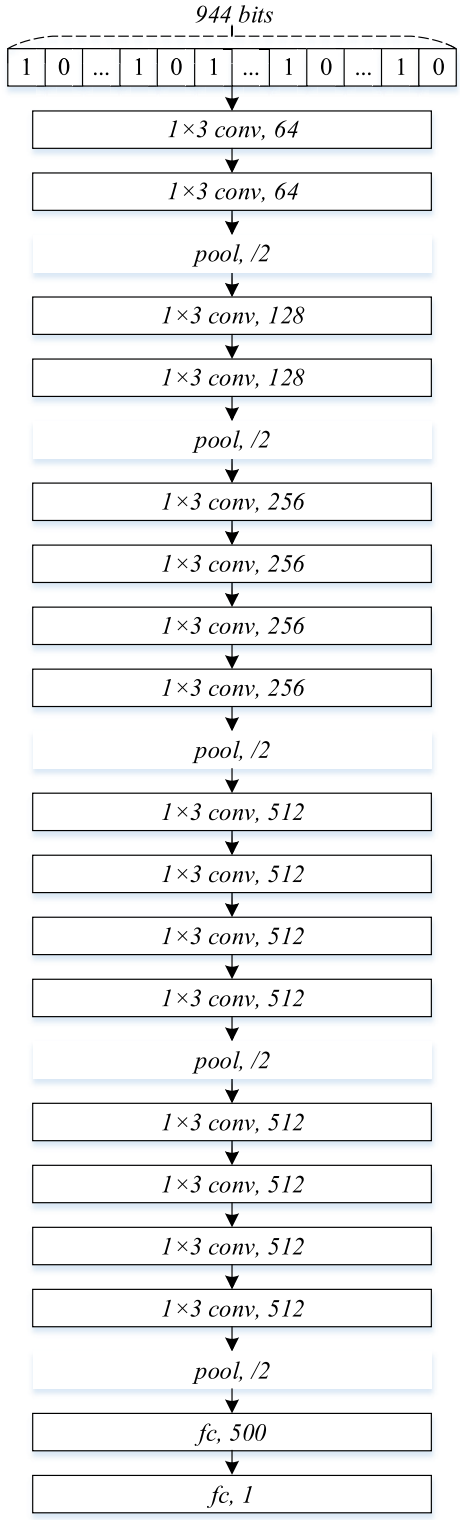


FIGURE 10. The architecture of VGG for branch prediction.

AlexNet: The second is based on the AlexNet [34]. We delete one fully-connected layer, and adapt the filters to branch data. The network contains five convolutional layers and two fully-connected layers. The first convolutional layer

filters the 944 bits input history information with 96 kernels of size 1 × 11 with a stride of 4 bits. The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size 1 × 5. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size 1 × 3 connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size 1 × 3, and the fifth convolutional layer has 256 kernels of size 1 × 3. The fully-connected layer are the same as the LeNet branch prediction model, which is different to AlexNet [34].

VGG: The third is based on the VGG [35], as shown in Fig. 10. The network contains sixteen convolutional layers and two fully-connected layers. All convolutional layers mostly have 1 × 3 filters. The number of kernels starts from 64 in the first convolution layer and then increases by a factor of 2 after each max-pooling layer, until it reaches 512. The convolution layers are not all followed by max-pooling. Every two convolution layers are followed one pooling layer for the 64-kernel layers or the 128-kernel layer in front. Every four convolution layers are followed one pooling layer for the 256-kernel, 512-kernel layers. The fully-connected layers are also same to the CNN1.

ResNet-50: The fourth is based on ResNet [37], as shown in Fig. 11. It contains 50 parameter layers which are 49 convolution layers and one fully-connected layers. The first convolutional layer has 64 kernels of size 1 × 7, followed by a 1 × 3 max-pooling. The other convolution layers are several 3-layer bottleneck blocks. The three convolution layers are 1 × 1, 1 × 3, and 1 × 1. The first kind bottleneck, which are stacked 3 times, contains 64, 64, and 256 kernels. The second kind bottleneck, which are stacked 4 times, contains 128, 128, and 512 kernels. The third kind bottleneck, which are stacked 6 times, contains 256, 256, and 1024 kernels. The last one contains 512, 512, and 2048 kernels, is stacked 3 times, followed by a 1 × 3 max-pooling. Only one fully-connected layer is followed in the last of whole network.

IV. RESULT AND DISCUSSION

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

A. COMPARING PERCEPTRON AND DNNs

Fig.12 shows the misprediction rate of perceptron, DBN and CNNs branch prediction models which are described in the last section. The misprediction rates of the perceptron predictors with is labeled 'Pcptrn'. The misprediction rates of the DBN is labeled 'DBN'. The misprediction rates of CNNs models are directly labeled 'LeNet', 'AlexNet', 'VGG', and 'ResNet-50' respectively.

From the Fig.12, we can see that DNNs outperform perceptron consistently. Compared to perceptron, the average

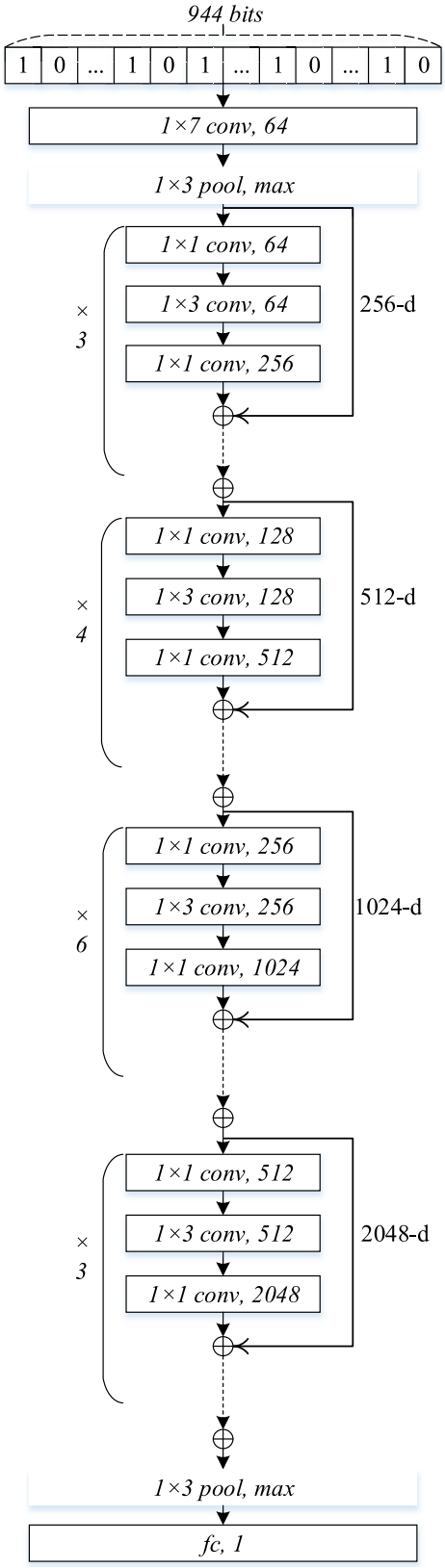


FIGURE 11. The architecture of ResNet-50 for branch prediction.

reductions on the misclassification rate of the DBN and ResNet-50 are 60.7% (= (6.266-2.461)/6.266) and 84.0% respectively. The misprediction rate of perceptron is much

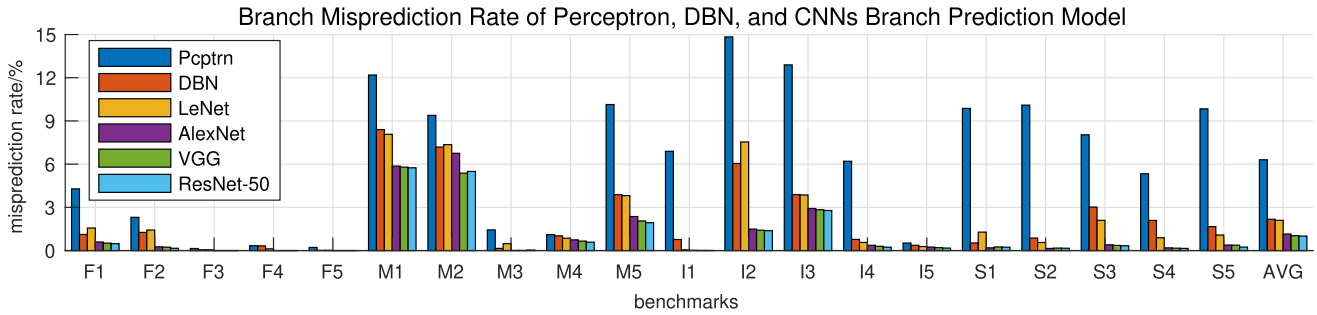


FIGURE 12. The branch misprediction rate of the perceptron, DBN, and CNNs branch prediction model on the testing set of CBP-4 traces.

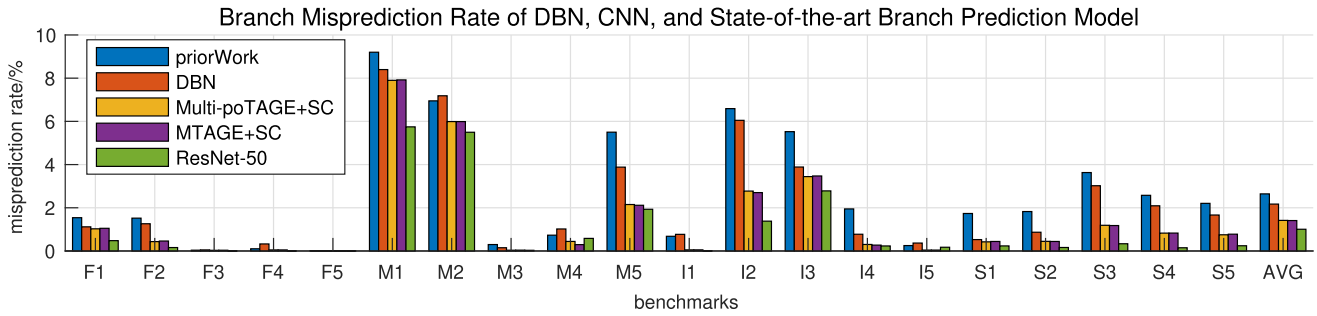


FIGURE 13. The branch misprediction rate of DBN, the prior work [21], Tages, and ResNet-50 on the testing set of CBP-4 traces.

lower on the benchmark 'F3', 'F4', 'F5', and 'I5', So we could simplify say that these benchmarks are linearly separable. All DNNs branch model have similar lower misprediction rates on them on which the improvement is very little. On the contrary, the branch outcomes of a benchmark are not linearly separable well, such as 'I2-I4', 'M1-M2', and 'Ss', the DNNs show their potential to greatly reduce the misprediction rate. This confirms that the DNNs could reduce the misprediction rates of the non-linearly separable branches. The multiple layers of the DNNs help to implement complex nonlinear functions more concisely, and it could work well on the nonlinear separable branches.

Form the Fig.12, we can also see that LeNet and DBN have similar misprediction rates while other CNNs has much lower misprediction rates than DBN or LeNet. LeNet contains only two convolution layers whereas AlexNet contains 5, VGG contains 16, and ResNet-50 contains 49 convolution layers. This shows that for branch prediction, deep CNNs are more effective than DBN, and a higher number of convolution layers and pooling layers could lead to lower misprediction rates.

Fig.12 shows that, compare to the LeNet branch prediction model which contains only two convolution layers, the misprediction rate of AlexNet, VGG, and ResNet-50 are all lower on every benchmark. So, compared with the traditional convolution neural network 'LeNet', the deeper convolution neural network could lead to lower misprediction rate. There are 14 benchmarks show that the misprediction rate consistently reduces with an increase in the convolution layer depth which is from 2 convolution layers of the LeNet to 49 convolution layers of the ResNet-50, Except linear separable benchmark 'F3-F5' on which the misprediction rate of the deeper CNN model(AlexNet, VGG, and ResNet-50) is too lower to equal to almost 0. There are none benchmark whose misprediction rates increase from the LeNet to the ResNet-50 adversely. It clearly demonstrates the importance of depth of convolution in CNN branch prediction model.

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 bits but different results in some branches, just like the separation of homonyms. Since the misprediction rate of ResNet-50 is the lowest among the four CNN branch prediction models, we focus on only one CNN branch prediction model, which is the ResNet-50, in our evaluation and discussion in the next sections.

B. COMPARING DNNs WITH THE STATE-OF-THE-ART BRANCH PREDICTORS

Fig.13 presents the misprediction rate of DBN, ResNet-50, state-of-the-art models. Our prior work, AIP [21] (Adaptive Information Processing, AIP), which is the winner of CBP1, is labeled 'priorWork'. It is a state-of-the-art perceptron-based predictor. The Multi-poTAGE+SC predictor [16], which is labeled 'Multi-poTAGE+SC', is the winner of CBP-4. The MTAGE+SC [18], which is labeled 'MTAGE+SC' in Fig.13, is the winner in CBP-5. These two TAGEs represent the state-of-the-art in achieving the upper bound of branch prediction.

From the Fig.13, we can see the misprediction rate of the DBN is lower than AIP for 14 benchmarks. The average misprediction rate of the prior work is higher than that of the DBN. In Fig 13, We can hardly find one benchmark on which the misprediction rate of DBN is obviously lower than the TAGEs. Conversely the average of the misprediction rate of DBN is obviously higher than the TAGEs. In other words, DBN could outperform our prior work, but not outperform the state-of-the-art TAGE-like branch predictor.

From Fig.13, we can also see that the misprediction rate of the ResNet-50 is lower than the prior work, and DBN on any benchmarks. And there are 18 benchmarks on which the ResNet-50 model has lower misprediction rates than Multi-poTAGE+SC and MTAGE+SC. The average of misprediction rate of the ResNet-50 model is 29% and 28.6% lower than Multi-poTAGE+SC and MTAGE+SC, respectively. Thus, the ResNet-50 could not only outperform our prior work, but also the Multi-poTAGE+SC and MTAGE+SC.

C. COMPARING THE RANDOM-SELECTED BRANCHES WITH THE BRANCHES IN THE DYNAMIC ORDER

In this experiment, we also select branches in their dynamic order: the training set is the first 90% branches, a validation set is the next 5% branches and a testing set is last 5% branches. In this way, the causality of data is maintained, meaning that future data are not used to train the network to produce a current prediction. Fig.14 shows the average misprediction rate on the 20 CBP-4 traces. The difference between the two approaches, i.e., the random selection and the selection based on the dynamic order, is very small. It confirms that the discussion on static random branch traces is also valid for the ones based on the dynamic order.

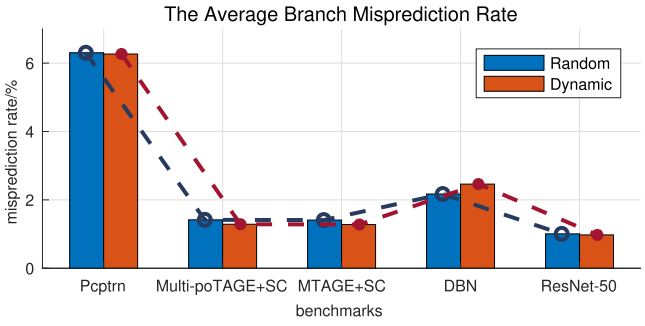


FIGURE 14. The average misprediction rate on the testing set of CBP-4 traces.

V. CONCLUSION

This paper takes a binary classification perspective on the branch prediction problem. We utilize deep neural networks as a classifier and explore both DBN 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 DBN; (3) The depth of convolution layer is also important for branch prediction. (4) DBN could outperform the prior work, but not outperform the state-of-the-art TAGE-like branch predictor; (5) ResNet-50 could outperform state-of-the-art TAGE-like branch predictors; and (6) the discussion on static random branch traces is also valid for the ones based on the dynamic order.

FUTURE WORK

This paper takes branch prediction as a pure binary classification stochastic problem. In order to simplify the problem, we only implemented off-line training. In order to apply deep learning for branch prediction, an online training algorithm needs to be employed. In addition, since most of the state-of-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.

REFERENCES

[1] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, 6th ed. San Mateo, CA, USA: Morgan Kaufmann, 2017.

[2] T.-Y. Yeh and Y. N. Patt, "Alternative implementations of two-level adaptive branch prediction," in *Proc. 19th Annu. Int. Symp. Comput. Archit.*, 1992, pp. 124–134.

[3] E. Sprangle, R. S. Chappell, M. Alsup, and Y. N. Patt, "The agree predictor: A mechanism for reducing negative branch history interference," in *Proc. 24th Annu. Int. Symp. Comput. Archit. ISCA*, 1997, pp. 284–291.

[4] P.-Y. Chang, M. Evers, and Y. N. Patt, "Improving branch prediction accuracy by reducing pattern history table interference," in *Proc. Conf. Parallel Archit. Compilation Technique*, 1996, pp. 48–57.

[5] A. N. Eden and T. Mudge, "The YAGS branch prediction scheme," in *Proc. 31st Annu. ACM/IEEE Int. Symp. Microarchitecture*, Dec. 1998, pp. 69–77.

[6] M. Mohammadi, S. Han, E. Atoofian, A. Baniasadi, T. M. Aamodt, and W. J. Dally, "Energy efficient on-demand dynamic branch prediction models," *IEEE Trans. Comput.*, vol. 69, no. 3, pp. 453–465, Mar. 2020.

[7] H. Gao and H. Zhou, "PMPM: Prediction by combining multiple partial matches," *J. Instruct.-Level Parallelism*, vol. 9, pp. 1–18, May 2007.

[8] H. Gao, Y. Ma, M. Dimitrov, and H. Zhou, "Address-branch correlation: A novel locality for long-latency hard-to-predict branches," in *Proc. IEEE 14th Int. Symp. High Perform. Comput. Archit.*, Feb. 2008, pp. 74–85.

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

[10] A. Seznec, "A new case for the TAGE branch predictor," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture - MICRO*, Dec. 2011, pp. 117–127.

[11] Y. Ma, H. Gao, and H. Zhou, "Using indexing functions to reduce conflict aliasing in branch prediction tables," *IEEE Trans. Comput.*, vol. 55, no. 8, pp. 1057–1061, Aug. 2006.

[12] K. B. Rao, N. Satyanaryana, and K. Vikram, "Instruction level parallelism using PPM branch prediction," *Int. J. Comput. Eng. Technol.*, vol. 3, no. 3, pp. 137–146, 2012.

[13] A. Seznec, "Analysis of the O-GEometric history length branch predictor," in *Proc. 32nd Int. Symp. Comput. Archit. (ISCA)*, Jun. 2005, pp. 394–405.

[14] P. Michaud, "A PPM-like, tag-based branch predictor," in *Proc. 1st Championship Branch Predict. (CBP-1) Held With 37th Int. Symp. Microarchitecture (MICRO)*, 2004, pp. 1–4.

[15] A. Seznec, "TAGE-SC-L branch predictors," in *Proc. JWAC Championship Branch Predict. (ISCA)*, Minneapolis, MN, USA, 2014, pp. 1–9.

[16] P. Michaud and A. Seznec, "Pushing the branch predictability limits with the multi-po TAGE+SC predictor," in *JWAC Championship Branch Prediction(ISCA)*. Minneapolis, MN, USA: ACM, 2014.

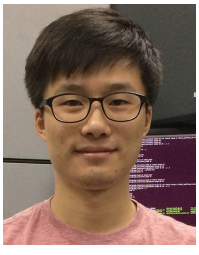
[17] G. H. Loh and D. S. Henry, "Predicting conditional branches with fusion-based hybrid predictors," in *Proc. Int. Conf. Parallel Archit. Compilation Techn.*, Sep. 2002, pp. 165–176.

[18] A. Seznec, "Exploring branch predictability limits with the MTAGE+SC predictor," in *Proc. 5th JILP Workshop Comput. Archit. Competitions (JWAC), Championship Branch Predict. (CBP)*, Seoul, South Korea, 2016, p. 4.
[19] D. A. Jiménez, "Idealized piecewise linear branch prediction," in *Proc. 1st Championship Branch Predict. (CBP-1) Held with 37th Int. Symp. Microarchitecture (MICRO)*, 2004, pp. 1–11.
[20] 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, Sep. 2005.
[21] H. Gao and H. Zhou, "Adaptive information processing: An effective way to improve perceptron branch predictors," in *Proc. 1st Championship Branch Predict. (CBP-1) Held with 37th Int. Symp. Microarchitecture (MICRO)*, 2004, pp. 1–10.
[22] Y. Mao, J. Shen, and X. Gui, "A study on deep belief net for branch prediction," *IEEE Access*, vol. 6, pp. 10779–10786, 2018.
[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[24] J. Kiros, W. Chan, and G. Hinton, "Illustrative language understanding: Large-scale visual grounding with image search," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2018, pp. 922–933.
[25] Y. Qin, N. Frosst, S. Sabour, C. Raffel, G. Cottrell, and G. Hinton, "Detecting and diagnosing adversarial images with class-conditional capsule reconstructions," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–20.
[26] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, "Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 4, pp. 694–707, Apr. 2016.
[27] J. Gonzalez-Dominguez, I. Lopez-Moreno, P. J. Moreno, and J. Gonzalez-Rodriguez, "Frame-by-frame language identification in short utterances using deep neural networks," *Neural Netw.*, vol. 64, pp. 49–58, Apr. 2015.
[28] D. A. Jimenez, "Piecewise linear branch prediction," in *Proc. 32nd Int. Symp. Comput. Archit. (ISCA)*, Jun. 2005, pp. 382–393.
[29] V. Passricha and R. K. Aggarwal, "A comparative analysis of pooling strategies for convolutional neural network based hindi ASR," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 2, pp. 675–691, Feb. 2020.
[30] G. E. Hinton, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[31] G. Hinton, "A practical guide to training restricted Boltzmann machines," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. UTML TR 2010–003, 2010.
[32] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Proc. Adv. Neural Inf. Process. Syst.*, D. S. Touretzky, Ed. San Mateo, CA, USA: Morgan Kaufmann, 1990, pp. 396–404.
[33] G. Xu, X. Su, W. Liu, and C. Xiu, "Target detection method based on improved particle search and convolution neural network," *IEEE Access*, vol. 7, pp. 25972–25979, 2019.
[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[35] A. Z. Karen Simonyan, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.
[36] C. Xiu and X. Su, "Composite convolutional neural network for noise deduction," *IEEE Access*, vol. 7, pp. 117814–117828, 2019.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[38] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "BlockDrop: Dynamic inference paths in residual networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8817–8826.
[39] H. H. Aghdam and E. J. Heravi, *Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification*. New York, NY, USA: Springer, 2017.
[40] C. Chen, "Research on key techniques of conditional branch processing," Ph.D. dissertation, Dept. Comput. Sci. Eng., Zhejiang Univ., Hangzhou, China, 2013.
[41] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia MM*, 2014, pp. 675–678.

YONGHUA MAO received the Ph.D. degree in computer science from the School of Computer Science and Technology, Xi'an Jiaotong University, in 2019. He was a Visiting Scholar in computer engineering with North Carolina State University under the supervision of Dr. H. Zhou. He is currently a Teacher with Xi'an Polytechnic University. His research interests include branch prediction, deep learning, and machine learning.

HUIYANG ZHOU (Senior Member, IEEE) received the bachelor's degree in electrical engineering from Xi'an Jiaotong University, China, in 1992, and the Ph.D. degree in computer engineering from North Carolina State University, in 2003. From 2003 to 2009, he was an Assistant Professor with the School of Electrical Engineering and Computer Science, University of Central Florida. He is currently a Professor with the Department of Electrical and Computer Engineering, North Carolina State University. His research interests include high-performance microarchitecture, low-power design, GPU computing (General Purpose computing on Graphics Processing Units or GPGPU), OpenCL for FPGA, architecture support for system dependability, and backend compiler optimization. He is a Senior Member of ACM. He was a recipient of the NSF CAREER Award.

XIAOLIN GUI received the Ph.D. degree in computer science from Xi'an Jiaotong University, China, in 2001. Since 2008, he has been the Director of the Key Laboratory of Computer Network, Shaanxi, China. From 2009 to 2012, he was the Vice Head of the Department of Computer Science and Technology. He is currently a Professor and the Deputy Dean of the School of Electronic and Information, Xi'an Jiaotong University. He leads the Center for Grid and Trusted Computing (CGTC). His current research interests include high-performance computing, secure computation of open network systems, dynamic trust management theory, and development on community networks. He was a recipient of the New Century Excellent Talents in University of China, in 2005.



JUNJIE SHEN received the master's degree in computer engineering from North Carolina State University, in 2015. 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.