Article

An Accurate and Efficient Timing Prediction Framework for Wide Supply Voltage Design Based on Learning Method

Peng Cao *, Wei Bao and Jingjing Guo

National ASIC System Engineering Center, Southeast University, Nanjing 210096, China; 220184659@seu.edu.cn (W.B.); guojingjing@seu.edu.cn (J.G.)

* Correspondence: caopeng@seu.edu.cn; Tel.: +86-025-8379

Received: 11 January 2020; Accepted: 29 March 2020; Published: 30 March 2020

Abstract: The wide voltage design methodology has been widely employed in the state-of-the-art circuit design with the advantage of remarkable power reduction and energy efficiency enhancement. However, the timing verification issue for multiple PVT (process–voltage–temperature) corners rises due to unacceptable analysis effort increase for multiple supply voltage nodes. Moreover, the foundry-provided timing libraries in the traditional STA (static timing analysis) approach are only available for the nominal supply voltage with limited voltage scaling, which cannot support timing verification for low voltages down to near- or sub-threshold voltages. In this paper, a learning-based approach for wide voltage design is proposed where feature engineering is performed to enhance the correlation among PVT corners based on a dilated CNN (convolutional neural network) model, and an ensemble model is utilized with two-layer stacking to improve timing prediction accuracy. The proposed method was verified with a commercial RISC (reduced instruction set computer) core under the supply voltage nodes ranging from 0.5 V to 0.9 V. Experimental results demonstrate that the prediction error is limited by 4.9% and 7.9%, respectively, within and across process corners for various working temperatures, which achieves up to 4.4× and 3.9× precision enhancement compared with related learning-based methods.

Keywords: PVT corners; path delay prediction; ensemble model; CNN feature extractor

1. Introduction

IC (Integrated Circuit) designs must be verified in simulation for the expected range of operating conditions to ensure sufficient field [1]. With the rapid development of IC technology and the diversity of actual working situations, an increasing number of PVT (process–voltage–temperature) corners must be simulated at the same time to ensure the stability of chip operation. However, this undoubtedly creates a problem, in that a lot of design cycle time and simulation time will be consumed. Specifically, the above-mentioned diversity of working environments includes the expansion of supply voltage range, which would provide the benefit of both high energy efficiency and high performance at the cost of far more verification effort to ensure stability. However, the foundry-provided timing libraries only support the timing verification at nominal voltage, so designers must characterize libraries for low voltage with tremendous effort. Moreover, existing commercial timing analysis tools are mainly designed for nominal voltage to calculate cell delay with linear interpolation operation based on a two-dimensional library table, which may suffer from unacceptable accuracy loss at low voltages due to nonlinear characteristics.

In recent years, machine learning methods have emerged as promising solutions for timing analysis. In [2], a model for gate delay prediction under process variation was proposed by adopting a
neural network with a low-dimensional parameter space, but the proposed model was only verified with artificial paths instead of those from real chips. Kahng et al. [3] used a learning-based approach to fit analytical models of wire slew and delay to estimate timing with a signoff STA (static timing analysis) tool, which can improve the accuracy of delay and slew estimations. In [4], the authors further proposed a machine learning model based on the bigrams of path stages to predict expensive PBA (path-based analysis) results from relatively inexpensive GBA (graph-based analysis) results, which substantially reduced pessimism while retaining the lower turnaround time of GBA. Ganapathy et al. [5] present a multivariate regression-based technique that computes the propagation delay of circuits subject to manufacturing process variations with the reported error of less than 5%.

Machine learning is also very powerful in the time-delay prediction problem under multiple PVT corners. In [6], Michael et al. used Gaussian process regression modeling to gradually iterate and expand the training set so as to obtain a subset of PVT corners with worst corners at the minimum simulation cost. Their results showed that corner simulation was reduced by an average of 79%. In [7], a model using part of the corner delay value to predict the remaining "unobserved corner" delay value was proposed where the delay prediction across the voltage domain was not involved.

In prior related learning-based timing analysis approaches, none focused on the issue of wide voltage design and no feasible way was provided to tackle the tradeoff between prediction error and simulation cost across supply voltage range. In this paper, a learning-based framework is proposed to predict the path delay at specific voltage nodes with the path delay acquired at other nodes, in particular, to predict the path delay at low voltages with the delays acquired at higher voltages. Dual learning-based techniques are employed in the proposed timing prediction framework, which are summarized herein.

In this paper, a novel prediction framework for path delay considering local variation is introduced. The framework uses a learning-based method to obtain the relationship of the path delay with circuit features at some corners, and to predict the path delay at a specific corner. The contributions of the paper are as follows:

- During the process of feature engineering, a dilated CNN (convolutional neural network) model is utilized to expand the feature dimensions by extracting the correlations among path delays for multiple PVT corners, which is beneficial to reduce characterization effort for each voltage node.
- During the process of training and inference, an ensemble model is applied to predict path delay for cross-voltage domain by combining diverse sets of learners, e.g., individual models, to improve the stability and prediction accuracy.

The rest of paper is organized as follows. Following the introduction, two key observations are given as the motivation of this work in Section 2. The proposed learning-based timing prediction framework is described in detail in Section 3, which mainly consists of the dilated CNN-based feature engineering and the two-layer stacking ensemble model. Experimental results are demonstrated in Section 4 and compared with competitive models. Finally, Section 5 presents the conclusions.

2. Motivation

2.1. Correlation Among Path Delays Across Wide Voltage Range

As the supply voltage decreases from a nominal value down to the near- or sub-threshold domain, the cell and path delay both increase due to the exponential relationship with the drain-source voltage at low voltages. A key observation is that there exists a strong correlation among the path delays across various voltage domains, although they are differently impacted by the cell type, cell size, input transition time, and output load capacitance of each path stage. The correlograms for the path delays among the voltages ranging from 0.5 V to 0.9 V under various process corners and working temperatures are demonstrated in Figure 1, where the correlations are evaluated in terms of the Pearson correlation coefficients $\rho_{ij}$ as formulated in Equation (1). The symbols $T_i$ and $T_j$ in Equation (1)
and correlation, as shown in Figure 2a. An interesting comparison could be made between the process of computer vision and path delay prediction, as shown in Table 1 that although the path delays increase by more than 1.5x per 0.1 V on average between 0.5 V and 0.9 V, the high correlation is significant since the Pearson correlation coefficients are roughly larger than 0.9 for the voltage interval of 0.1 V and no smaller than 0.73 for all voltage combinations.

Table 1. Path delays and Pearson correlation coefficients under the TT corner and the temperature ~25 °C.

| Path Delay (ns) | Max. | Ave. | Min. | 0.5 V | 0.6 V | 0.7 V | 0.8 V | 0.9 V |
|----------------|------|------|------|-------|-------|-------|-------|-------|
| 0.5 V          | 9.61 | 4.93 | 1.33 | 1.00  | 0.91  | 0.89  | 0.81  | 0.76  |
| 0.6 V          | 3.79 | 1.85 | 0.96 | 0.91  | 1.00  | 0.92  | 0.90  | 0.83  |
| 0.7 V          | 1.95 | 0.97 | 0.57 | 0.89  | 0.92  | 1.00  | 0.94  | 0.89  |
| 0.8 V          | 1.27 | 0.68 | 0.44 | 0.81  | 0.90  | 0.94  | 1.00  | 0.91  |
| 0.9 V          | 0.95 | 0.54 | 0.35 | 0.76  | 0.83  | 0.89  | 0.91  | 1.00  |

2.2. Similarity Between CNN Structures for Traditional Applications and Timing Prediction

The CNN is a series of feedforward neural networks with deep structures that include convolutional calculations and is considered as a representative algorithm of deep learning [9,10].

Traditionally, the CNN has achieved a pervasive application in the field of computer vision, where the image data is structured firstly in different channels, and then in different rows and columns, before being processed by the CNN to extract the inherent correlation, as shown in Figure 2a. An interesting comparison could be made between the process of computer vision and path delay prediction, as shown...
in Figure 2b, where the correlation existing among different PVT corners should also be extracted by CNN-series algorithms.

Figure 2. Similarity between the processes of (a) computer vision and (b) timing prediction.

More specifically, it is obvious that the correlation among different PVT corners is not only restricted within the adjacent voltage nodes and temperature nodes, but also exists among those with relatively greater distances. To capture the correlation more widely without the issue of feature explosion, the dilated CNN [11,12] could be utilized as a new convolutional network module to systematically aggregate multi-scale contextual information by exponentially increasing the receptive field without losing resolution or analyzing rescaled features. Both one-dimensional and two-dimensional kernels could be used in the dilated CNN for applications such as computer vision [11] and natural language processing [12]. As illustrated in Figure 3, the path delays at a specific voltage and temperature could be predicted using the dilated CNN with those captured at other voltages and temperatures. By keeping the size of the convolution kernel constant, the coverage of the voltages and temperatures gradually expands with a linearly increased dilation rate from the first hidden layer to the output layer, so that the multi-scale contextual information could be used by the dilated CNN to improve the prediction accuracy without additional computational overhead.

Figure 3. Path delay prediction by utilizing the dilated convolutional neural network (CNN) with 1/2/4/8-dilated convolution.
3. Proposed Learning-based Timing Prediction Framework

3.1. Overview

An overview of the timing prediction framework proposed in this paper is demonstrated in Figure 4 based on the machine learning methods, where the dilated CNN is utilized for feature engineering while the ensemble model is used for supervised learning with high robustness. In order to predict the path delays at the voltage node $V_i$ for a specific temperature and process corner, with the path delays under $V_i$ for various temperatures and process corners, the process of feature engineering is performed firstly to extract the correlation between PVT corners based on a dilated CNN. Then, an ensemble model is utilized with the two-layer stacking method to improve the robustness and prediction accuracy. Details for the feature engineering and ensemble model are described in Sections 3.2 and 3.3 respectively.

![Figure 4. Overview of proposed timing prediction framework.](image)

3.2. Feature Engineering Based on Dilated CNN

Due to the consideration mentioned in Section 2.2, the dilated CNN [11] is utilized in the feature extraction process of this work to extract the correlation of path delays under various PVT corners. The structure of the dilated CNN is illustrated in Figure 5 and then described in detail, and consists of an input layer, convolutional layers, flatten layer, dense layer, and output layer. As shown in Figure 5, $Q_m$ features are extracted with the dilated CNN based on $N_f$ original features for each of $N_s$ samples.

![Figure 5. Structure of the dilated CNN.](image)

The input data is reshaped into a three-dimensional form as $N_s \times N_f \times 1$, where $N_s$ represents the number of samples and $N_f$ denotes the number of features, as formulated in Equation (2).

$$y = x_{N_s \times N_f \times 1}$$

With the input data, $n$ convolutional layers are cascaded to transform data into the shape of $N_s \times H_i \times F_i$, where $F_i$ is the number of convolutional kernels in layer $i$ ($1 \leq i \leq n$) and $H_i$ is determined...
by the related parameters of the corresponding convolutional kernels. The computation for each convolutional layer is defined as in Equation (3), where \( x \) and \( y \) denote the input and output data, respectively, and \( W \) is the weight coefficient of the convolutional kernel.

\[
y_{li+1}^{ki+1} = x_{li}^{ki}W_{li}^{ki}, \quad 1 \leq l_i \leq H_{i-1}, \quad 1 \leq k_i \leq F_{i-1}
\]  

(3)

It should be noted that for each convolutional layer of the dilated CNN, the coverage of the convolution kernel could be different even with an equal size, as demonstrated in Figure 3. In the flattened layer, the three-dimensional data is reshaped into a two-dimensional one as \( N_s \times H_n \times F_n \). As shown in Equation (4), the \( N_s \) matrixes with the shape of \( H_n \times F_n \) are flattened by concatenating \( H_n \) \( F_n \)-dimensional vectors, \( x_{trn} \), into \( y_j \) for each matrix.

\[
y_j = [x_{1i}, \ldots, x_{ni}, \ldots, x_{H_n}], \quad 1 \leq i \leq H_n, \quad 1 \leq j \leq N_s
\]  

(4)

In the following, there are \( m \) fully connected dense layers, where \( Q_j \) (\( 1 \leq j \leq m \)) is the number of neurons in layer \( j \) (\( 1 \leq j \leq m \)). The computation for the dense layer is formulated as in Equation (5), where the \( \tanh \) function is used as the activation function, and \( W \) and \( b \) are the weight coefficient and bias, respectively, to produce the output \( y \) with the input \( x \).

\[
y_{li+1} = \tanh(x_{li}W_{li} + b_{li}), \quad 1 \leq l_i \leq Q_{i-1}
\]  

(5)

Finally, the predicted results are output with the shape of \( N_s \times 1 \) with a linear transform formulated in Equation (6), where \( y_j \) denotes one of the \( N_s \) elements in the result vectors calculated from \( Q_m \)-dimensional \( x_j \) with the corresponding weight coefficient \( W \) and bias \( b \).

\[
y_j = x_jW + b_j, \quad 1 \leq j \leq N_s
\]  

(6)

It is worth noting that, in this work, the purpose of the utilization of the dilated CNN is to extract the correlation among the path delays from different PVT corners instead of prediction. To do this, the output of the \( m \)-th dense layer is collected as the input of the following process of training and inference.

Based on the dilated CNN, the process of feature extraction in this work is depicted as Figure 6, with the advantage of preventing data leakage and overfitting by using a cross-validation strategy. The flow of feature extraction in Figure 6 mainly consists of the training step, inference step, and feature concatenation step. The delays of \( N_p \) paths at specific voltage \( V_l \) and \( N_l \) various temperatures are partitioned into the training set and test set firstly, and include \( N_{trn} \) and \( N_{test} \) paths, respectively. In the training step, \( k \)-fold cross-validation is performed by selecting \( 1/k \) samples from the training set, which is iterated \( k \) times to train \( k \) different dilated CNNs for each \( k \)-fold cross-validation with the original path delays at \( V_l \). Then, in the inference step, the \( k \) groups of cross-validation data are predicted with the corresponding dilated CNN to extract new features based on the original path delays at \( V_l \) from the last dense layer with the shape of \( N_{trn}/k \times N_{new} \), where \( N_{new} \) is equal to the parameter of \( Q_m \) in the dilated CNN. The new features of the training set are then concatenated with the original features, e.g., the original path delays at \( V_l \) in the feature concatenation step. Similarly, the new features generated by the dilated CNN for the test set are also concatenated to the original ones, except that the generated new features for the test set from \( k \) different dilated CNNs should be averaged into one \( N_{test} \times N_{new} \) matrix before concatenation.
an ensemble model, as applied to build the ensemble model, as well as the accuracy. The formula for LR model is written as in Equation (7), where \( \theta \) represents the weight coefficients. The equation for the LightGBM model is given in Equation (8), where \( f_{\theta}(x) \) means the

### 3.3. Ensemble Model with Two-layer Stacking

In the process of training and inference, an ensemble approach is adopted to modeling, which is an art of combining diverse sets of learners, e.g., individual models, to improve the stability and predictive power of the model. Here we use a learner to combine the output from different learners, which leads to the decrease in either bias or variance error, depending on the combining learner we use. Compared with other commonly-used ensemble learning techniques, such as bagging and boosting, stacking can transfer the ensemble features to a simple model and does not require too many parameter tunings and feature selections [13,14]. In order to improve prediction precision while avoiding overfitting, a two-layer stacking method is applied to build the ensemble model, as illustrated in Figure 7, including a hidden layer and an output layer.

![Feature extraction flow.](image)

**Figure 6.** Feature extraction flow.

![Ensemble model flow.](image)

**Figure 7.** Ensemble model flow.

In the ensemble model flow shown in Figure 7, the linear regression (LR) [15] and light gradient boosting machine (LightGBM) [16] algorithms are utilized in the two layers due to their unique characteristics as explained in the following. LR is an efficient and simple machine learning algorithm, which does not require complicated calculations, even in the case of large amounts of data. However, LR only considers the linear relationship between variables so that it is very sensitive to outliers and the input features should be independent for the LR algorithm. In order to overcome its demerit, the LightGBM is applied in the ensemble model as a widely-used gradient boosting framework. Since it uses tree-based learning algorithms, the LightGBM is not sensitive to outliers and can achieve high accuracy. The formula for LR model is written as in Equation (7), where \( \theta \) represents the weight coefficients. The equation for the LightGBM model is given in Equation (8), where \( f_{\theta}(x) \) means the
initial solution, \( f_{i-1}(x) \) represents the \((i-1)\)-th solution, \( c_{ij} \) represents the weight coefficients, and \( T \) and \( J \) denote the number of iterations and weight coefficients, respectively.

\[
f_{LR}(x) = \sum_{i=0}^{q} \theta_i |x_i|
\]

\[
f_{LightGBM}(x) = f_0(x) + \sum_{i=1}^{J} \sum_{j=1}^{T} c_{ij} f_{i-1}(x)
\]

The parameters of \( \theta_i \) and \( c_{ij} \) used in the LR and LightGBM models are generated in the training process by the back-propagation method. In this work, the commonly-used gradient descent algorithm is applied to update them iteratively from a random initialization value, as formulated in Equation (9), where \( \theta^{(i+1)} \) and \( \theta^{(i)} \) represent the parameter \( \theta \) in the \((i+1)\)-th and \(i\)-th iterations, \( f(\theta) \) is the loss function, and \( \eta \) is the learning rate. The derivation process of the parameter \( c_{ij} \) is similar.

\[
\theta^{(i+1)} = \theta^{(i)} - \eta \frac{\partial f}{\partial \theta^{(i)}}(\theta^{(i)})
\]

The parameters of \( \theta \), are \((N_{new} + N_t + 1)\)-dimensional vectors with \( N_{new} + N_t \) weight coefficients and one bias for the \( N_{new} + N_t \) input features in the proposed framework for each voltage combination and each process corner. The parameters of \( c_{ij} \) consist of \( T \) vectors with lengths of no longer than \( J \) for each voltage combination and each process corner, where \( T \) indicates the number of trees and \( J \) means the upper bound of the number of leaves for each tree.

As shown in Figure 7, the hidden layer accepts the extracted features from feature engineering and the original path delays at the voltage of \( V_i \) with the shapes of \( N_{trn} \times (N_{new} + N_t) \) for the training set and \( N_{test} \times (N_{new} + N_t) \) for the test set, which are defined as \( X_{trn} \) and \( X_{test} \), respectively. The input features are trained by LR and LightGBM, respectively, with the predicted results denoted as \( X_{LR_{trn}}/X_{LR_{test}} \) and \( X_{LightGBM_{trn}}/X_{LightGBM_{test}} \), which are concatenated as the input features of the output layer with the shapes of \( N_{trn} \times 2 \) and \( N_{test} \times 2 \), respectively. In the output layer, the data is further trained by another LR model with the predicted results indicated as \( \hat{Y}_{trn} \) and \( \hat{Y}_{test} \) for the training set and test set respectively, where the path delays at \( V_i \) are predicted by the whole framework.

4. Experimental Results and Comparison

4.1. Experimental Setup

The proposed learning-based timing prediction framework was validated with a commercial RISC (reduced instruction set computer) core design, which was designed to operate under the supply voltage ranging from 0.5 V to 0.9 V for IoT (internet of things) application. The top thousand critical paths were extracted by the PrimeTime tool from the post-layout netlist and translated to SPICE (simulation program with integrated circuit emphasis) netlist using the write_spice_deck command, whose path delays were acquired by the HSPICE tool at the process corner of FF/TT/SS (slow-slow), RC (resistance capacitance) corners of CBEST/CWORST/RCBEST/RCWORST, temperatures of \( -25^\circ \)C/0\(^0\)C/25\(^0\)C/75\(^0\)C/125\(^0\)C, and voltages of 0.5 V/0.6 V/0.7 V/0.8 V/0.9 V as the learning samples for the proposed framework. The reason for using HSPICE instead of PrimeTime to obtain the path delay is due to the lack of a timing library at low voltages, as well as the accuracy of the SPICE simulation, whose results were also considered as the golden reference of the prediction results. The proposed framework was realized with the toolkits of keras [17], sklearn [18] and lightgbm [16] to build the models for dilated CNN, LR, and LightGBM, respectively. The relative RMSE (root mean squared error) is used as the evaluation criteria of the prediction accuracy in this framework and other competitive models. The definition of relative RMSE is given by Equation (10), where the RMSE and mean are...
defined in Equations (11) and (12), respectively. \( y_t \) and \( \hat{y}_t \) are the real value and its predicted value, respectively, and \( T \) is the number of paths.

\[
r\text{RMSE} = \frac{\text{RMSE}}{\bar{y}}
\]

\[
\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}
\]

\[
\bar{y} = \frac{\sum_{t=1}^{T} y_t}{T}
\]

The main parameters of the proposed framework are listed in Table 2.

| Parameter | Description                           | Value |
|-----------|---------------------------------------|-------|
| \( m \)  | # of convolutional layers in dilated CNN | 2     |
| \( n \)  | # of dense layers in dilated CNN       | 2     |
| \( k \)  | # of folds for cross validation        | 5     |
| \( H_1 \) | # of vectors calculated by convolution in convolutional layer 1 | 4     |
| \( F_1 \) | # of kernels in convolutional layer 1  | 16    |
| \( H_2 \) | # of vectors calculated by convolution in convolutional layer 2 | 2     |
| \( F_2 \) | # of kernels in convolutional layer 2  | 8     |
| \( Q_1 \) | # of neurons in dense layer 1          | 64    |
| \( Q_2 \) | # of neurons in dense layer 2          | 8     |
| \( T \)  | # of trees for LightGBM                | 100   |
| \( J \)  | Max. # of leaves in one tree for LightGBM | 100   |

4.2. Experimental Results

Considering that the process corner would have a non-negligible impact on the path delay, the accuracy of the proposed framework was evaluated for a wide voltage range in two cases, which are under the same process corners and across different ones. In case 1, as shown in Section 4.2.1, the path delays at different voltages were predicted by those path delays obtained under the same process corner. In case 2, in Section 4.2.2, the designer can achieve higher PVT corner verification effort reduction than in case 1 by predicting the path delays under all process corners with those obtained under only one single process corner, e.g., the TT corner. Since the evaluation accuracy remains similar for various RC corners according to the experimental results, only those from the CBEST corner are used for comparison in this paper.

4.2.1. Prediction under the Same Process Corner

The prediction errors in terms of the relative RSME for FF, SS, and TT corners are illustrated in Tables 3–5 for the path delays at the supply voltages from 0.5 V to 0.9 V, and are averaged for all working temperatures. Each row of the tables indicates the prediction error at a specific target voltage domain where the path delays are predicted, while each column of the tables means the error at a specific feature domain where the path delays are obtained as the input features. Although the designers are interested in the prediction accuracy for lower voltages in most cases, the prediction results for higher voltages are also included to give a comprehensive understanding of this work. It can be seen from these tables that although the prediction error increases as the difference between the feature voltage domain (indicated as the column index) and the target voltage domain (indicated as the row index) increases, it is restricted in a reasonable range to be no larger than 5%. Moreover, the proposed prediction framework demonstrated robustness among different process corners as shown in the three tables, with equivalent prediction precision for each pair of feature voltage and target voltage.
The heatmap of prediction error under the FF/SS/TT corner is illustrated in Figure 8, which shows that the prediction error rises as the voltage gap increases.

Table 3. Prediction error of the proposed framework under FF corner.

| Voltage (V) | 0.5 V | 0.6 V | 0.7 V | 0.8 V | 0.9 V |
|------------|-------|-------|-------|-------|-------|
| 0.5 V      | N/A   | 0.8%  | 2.4%  | 3.6%  | 4.7%  |
| 0.6 V      | 1.2%  | N/A   | 2.0%  | 2.8%  | 3.3%  |
| 0.7 V      | 2.6%  | 1.9%  | N/A   | 1.4%  | 2.2%  |
| 0.8 V      | 3.8%  | 3.7%  | 1.3%  | N/A   | 1.3%  |
| 0.9 V      | 4.6%  | 4.1%  | 1.8%  | 1.5%  | N/A   |

Table 4. Prediction error of the proposed framework under SS corner.

| Voltage (V) | 0.5 V | 0.6 V | 0.7 V | 0.8 V | 0.9 V |
|------------|-------|-------|-------|-------|-------|
| 0.5 V      | N/A   | 1.9%  | 3.4%  | 3.5%  | 4.1%  |
| 0.6 V      | 1.7%  | N/A   | 1.3%  | 3.1%  | 3.6%  |
| 0.7 V      | 3.3%  | 1.2%  | N/A   | 2.2%  | 2.8%  |
| 0.8 V      | 4.9%  | 1.6%  | 1.3%  | N/A   | 2.2%  |
| 0.9 V      | 5.9%  | 2.6%  | 1.7%  | 1.5%  | N/A   |

Table 5. Prediction error of the proposed framework under TT corner.

| Voltage (V) | 0.5 V | 0.6 V | 0.7 V | 0.8 V | 0.9 V |
|------------|-------|-------|-------|-------|-------|
| 0.5 V      | N/A   | 2.1%  | 3.1%  | 2.6%  | 4.4%  |
| 0.6 V      | 1.5%  | N/A   | 1.8%  | 2.5%  | 3.0%  |
| 0.7 V      | 3.1%  | 1.3%  | N/A   | 2.1%  | 3.2%  |
| 0.8 V      | 3.8%  | 1.7%  | 2.1%  | N/A   | 2.3%  |
| 0.9 V      | 3.9%  | 2.0%  | 2.7%  | 2.4%  | N/A   |

Figure 8. Heatmap of prediction error under the process corner of (a) FF; (b) SS; (c) TT.

Table 6 compares the maximum prediction errors of this work among all pairs of feature voltage and target voltage ranging from 0.5 V to 0.9 V for different process corners and different temperatures with other related models, including LR and LightGBM. It can be found that the commonly-used multivariate linear regression model suffers from nearly 20% precision loss for the timing prediction issue due to correlation between input features, which is more suitable for the utilization of tree-based models like LightGBM with the error of 5.6%. Owing to the ensemble model used in this work, the proposed framework outperforms the competitive algorithms in terms of prediction accuracy improvement by 4.4× compared to LR and by 1.3× compared to LightGBM.
4.2.2. Prediction under the Different Process Corner

The prediction errors for the path delays under FF and SS corners with the features obtained under the TT corner are illustrated in Tables 7 and 8, where the supply voltage range is consistent with the case in Section 4.2.1. It can be seen that, although the prediction error rises compared with the case in Section 4.2.1, the maximum error is still no larger than 8% for all pairs of feature voltage and target voltage under FF and SS corners, which happens when predicting the path delays at 0.5 V under FF/SS corners with the path delay at 0.9 V under the TT corner. The heatmap of the prediction errors under FF/SS corners is illustrated in Figure 9, which shows a similar trend to that under the same process corner.

**Table 7.** Prediction error of the proposed framework for FF corners with TT corner features.

| 0.5 V | 0.6 V | 0.7 V | 0.8 V | 0.9 V |
|------|------|------|------|------|
| 0.5 V | 1.9% | 2.2% | 5.2% | 5.7% | 7.8% |
| 0.6 V | 3.8% | 2.1% | 2.7% | 4.1% | 6.3% |
| 0.7 V | 5.1% | 3.3% | 1.5% | 1.9% | 5.4% |
| 0.8 V | 6.0% | 4.7% | 2.3% | 1.8% | 2.3% |
| 0.9 V | 6.3% | 6.1% | 3.8% | 2.2% | 1.9% |

**Table 8.** Prediction error of the proposed framework for SS corners with TT corner features.

| 0.5 V | 0.6 V | 0.7 V | 0.8 V | 0.9 V |
|------|------|------|------|------|
| 0.5 V | 2.0% | 2.8% | 5.0% | 5.6% | 7.4% |
| 0.6 V | 2.1% | 1.1% | 2.8% | 5.1% | 6.1% |
| 0.7 V | 2.5% | 3.8% | 1.7% | 1.9% | 5.3% |
| 0.8 V | 6.1% | 5.4% | 2.7% | 1.4% | 2.4% |
| 0.9 V | 6.5% | 5.2% | 3.2% | 2.0% | 1.8% |

**Figure 9.** Heatmap of prediction error with TT corner features for different process corner: (a) FF; (b) SS.

The prediction errors under the different process corners are illustrated in Table 9 for the related models and this work, where the proposed model still shows 3.9× and 1.3× better precision than the LR and LightGBM models, with less than 8% error to predict the path delays for the different process corners.

**Table 9.** Prediction error comparison with related models under the same process corner.

| Process Corner | LR | LightGBM | This Work |
|----------------|----|----------|-----------|
| FF             | 19.8% | 5.6% | 4.9% |
| TT             | 20.4% | 4.6% | 3.8% |
| SS             | 17.5% | 6.7% | 4.6% |
| Ave.           | 19.2% (4.4×) | 5.6% (1.3×) | 4.4% (1×) |

**Table 6.** Prediction error comparison with related models under the same process corner.
Table 9. Prediction error comparison with related models under different process corner.

| Process | LR     | LightGBM | This Work |
|---------|--------|----------|-----------|
| FF      | 31.9%  | 10.3%    | 7.9%      |
| SS      | 27.9%  | 9.7%     | 7.5%      |
| Ave.    | 29.9% (3.9×) | 10.0% (1.3×) | 7.7% (1×) |

5. Conclusions

In this work, a leaning-based timing prediction framework is proposed for wide voltage design with a CNN-based feature extraction technique and an ensemble method. Experimental results show that in the cases of both within and across process corners, the proposed method can obtain robust prediction results with low errors.

Author Contributions: P.C., W.B., and J.G. organized this work. P.C. and W.B. performed the modeling, simulation and experiment work. The manuscript was written by P.C. and W.B., and edited by J.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key Research and Development Program of China (Grant No. 2019YFB2205004), and National Natural Science Foundation of China (Grant No. 61834002).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. MShoniker, B.F.; Cockburn, J.H.; Pedrycz, W. Minimizing the Number of Process Corner Simulations during Design Verification. In Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2015; pp. 289–292.
2. Gao, M.; Ye, Z.; Wang, Y.; Yu, Z. On Modeling the Digital Gate Delay under Process Variation. *J. Semicond.* 2011, 32, 075010. [CrossRef]
3. Kahng, A.B.; Kang, S.; Lee, H.; Nath, S.; Wadhwani, J. Learning-Based Approximation of Interconnect Delay and Slew in Signoff Timing Tools. In Proceedings of the ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP), Austin, TX, USA, 2 June 2013; pp. 1–8.
4. Kahng, A.B.; Mallappa, U.; Saul, L. Using Machine Learning to Predict Path-Based Slack from Graph-Based Timing Analysis. In Proceedings of the IEEE 36th International Conference on Computer Design (ICCD), Orlando, FL, USA, 7–10 October 2018; pp. 603–612.
5. Ganapathy, S.; Canal, R.; Gonzalez, A.; Rubio, A. Circuit Propagation Delay Estimation through Multivariate Regression-Based Modeling under Spatio-Temporal Variability. In Proceedings of the Conference on Design, Automation & Test in Europe, Dresden, Germany, 8–12 March 2010; pp. 417–422.
6. Shoniker, M.; Oleynikov, O.; Cockburn, B.F.; Han, J.; Rana, M.; Pedrycz, W. Automatic Selection of Process Corner Simulations for Faster Design Verification. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 2018, 37, 1312–1316. [CrossRef]
7. Kahng, A.B.; Mallappa, U.; Saul, L.; Tong, S. “Unobserved Corner” Prediction: Reducing Timing Analysis Effort for Faster Design Convergence in Advanced-Node Design. In Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 168–173.
8. Rodgers, J.L.; Nicewander, W.A. Thirteen ways to look at the correlation coefficient. *Am. Stat.* 1998, 42, 59–66. [CrossRef]
9. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning (Vol. 1)*; MIT Press: Cambridge, MA, USA, 2016; pp. 326–366.
10. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, L.; Wang, G.; et al. Recent advances in convolutional neural networks. *arXiv Prepr.* 2015, arXiv:1512.07108. [CrossRef]
11. Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv Prepr.* 2015, arXiv:1511.07122.
12. Kim, Y. Convolutional neural networks for sentence classification. *arXiv Prepr.* 2014, arXiv:1408.5882.
13. Huang, F.; Xie, G.; Xiao, R. Research on Ensemble Learning. In Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence, Shanghai, China, 7–8 November 2009; pp. 249–252.
14. Yu-yan, J. Selective Ensemble Learning Algorithm. In Proceedings of the 2010 International Conference on Electrical and Control Engineering, Wuhan, China, 25–27 June 2010; pp. 1859–1862.
15. Seber, G.A.F. *Linear Regression Analysis*; Wiley: New York, NY, USA, 1997; pp. 48–51.
16. Ke, G.; Wang, T.; Chen, W.; Ma, W.; Liu, T.Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. *Adv. Neur. Inf. Process. Sys.* 2017, 30, 3147–3155.
17. Keras. Available online: https://keras.io/ (accessed on 10 November 2019).
18. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* 2011, 12, 2825–2830.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).