Deep Learning for Blast Furnaces: Skip-Dense Layers Deep Learning Model to Predict the Remaining Time to Close Tap-holes for Blast Furnaces

Keeyoung Kim
Department of Computer Science
The State University of New York (SUNY), Korea
Republic of Korea
kykim@sunykorea.ac.kr

Byeongrak Seo
IngenioAI
Republic of Korea
brseo@ingenio.ai

Sang-Hoon Rhee
Linkgenesis
Republic of Korea
rhee@linkgenesis.co.kr

Seungmoon Lee
Pohang Iron and Steel Company (POSCO)
Republic of Korea
smoonlee@posco.com

Simon S. Woo
Department of Computer Science & Engineering
Sungkyunkwan University
Republic of Korea
swoo@g.skku.edu

ABSTRACT

Manufacturing steel requires extremely challenging industrial processes. In particular, predicting the exact time instance of opening and closing tap-holes in a blast furnace has a great influence on steel production efficiency and operating cost, in addition to human safety. However, currently predicting the time to open and close tap-holes of the blast furnace still highly relies on manual human expertise and labor. Also, most of the prior research is limited to indirectly model the level of liquids in the hearth, using complex mathematical models or classical machine learning approaches.

In this paper, we use a data-driven deep learning method to more accurately predict the remaining time to close each tap-hole in a blast furnace and develop an AI-enabled automated advisory system to reduce manual human efforts as well as operation cost. We develop a multivariate time series forecasting algorithm using Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) models to more accurately predict the opening and closing time for the Pohang Iron and Steel Company (POSCO) blast furnace. In particular, we use and validate data from one of the largest operating furnaces in the world to develop our system. Our proposed Skip-dense CNN (S-CNN) model achieves more than 90% accuracy within ±30 minutes tolerance, compared to other LSTM baseline models. Our S-CNN model has been successfully deployed at a large-scale blast furnace of POSCO since January 2018 and has achieved similar accuracy. And we even exceeded the reported human performance in a real operational environment.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by regression**; • **Applied computing** → **Industry and manufacturing**.

KEYWORDS

Blast Furnace, Deep Learning, Multivariate Time Series, Convolutional Neural Network, Skip-Dense Layer

ACM Reference Format:
Keeyoung Kim, Byeongrak Seo, Sang-Hoon Rhee, Seungmoon Lee, and Simon S. Woo. 2019. Deep Learning for Blast Furnaces: Skip-Dense Layers Deep Learning Model to Predict the Remaining Time to Close Tap-holes for Blast Furnaces. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3357384.3357803

1 INTRODUCTION

Steel is fundamental to our everyday lives [3], being a major component used in a countless number of products and infrastructures such as automobiles, ships, tools, buildings, etc. In 2017, the world produced more than 1,600 million tonnes [3]. As demands for steel production continues to grow, it becomes increasingly important to produce steel more efficiently and safely. Gwang-yang Steel Works under the Pohang Iron and Steel Company (POSCO)[1] is one of the largest integrated steelworks in the world [1]. In Gwangyang Steel Works, there are three large-scale blast furnaces that are larger than 5,500 cubic meters. Operating and maintaining a large-scale furnace that is greater than 5,000 cubic meters requires advanced technologies and long-term expertise to operate.

Inside a blast furnace, molten iron is produced from iron ore and coke by a complex chemical reaction at high temperatures up to 2,300℃. The molten iron and slag are pooled in the hearth, which is the lower part of the blast furnace as shown in Fig. 1. There are 4 tap-holes at the hearth, where tapping is a process to

[1]http://www.posco.com

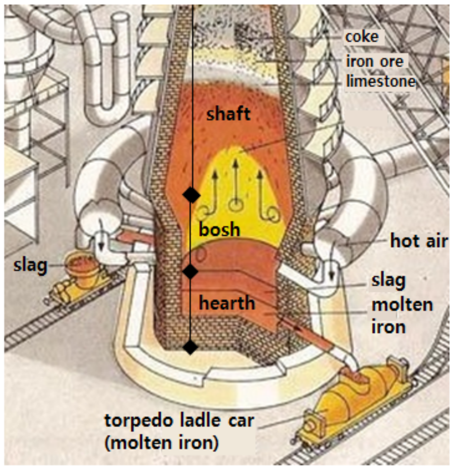

Figure 1: Description of a blast furnace and steel making process [2]: the bottom of the blast furnace is a hearth, where molten iron and slag are flowed down. Tapping is a process to drainage those molten iron and slag, and molten iron is flowed out from the blast furnace to TLC.

make a hole on the hearth to drain the molten iron from the hearth. The hole can be opened with a drill and closed with a mud gun. Once a tap-hole is opened, molten iron drainage continues for 5~6 hours. The level of liquid inside the hearth is hardly predictable because of its high temperature and complex behavior in the blast furnace. Also, no sensors can be installed in the hearth as it cannot endure the high temperature. That is the primary reason that blast furnace operation still highly depends on operators' experience and expertise.

During the operation of the blast furnace, it is crucial to properly maintain the liquid level in the hearth. If the liquid level in the hearth is too low, toxic gas and dust are spouted out from the tap-holes. On the other hand, if the liquid level is spilled over the level of the tuyere, then the whole blast furnace is stopped. Moreover, it is dangerous for humans to perform this process in proximity to a blast furnace due to incurred toxic gas and high temperature, and it can possibly cause significant injuries to operators.

To address some of these challenges, much research has aimed to predict the level of liquids in the hearth to decide when to open and close tap-holes. Autoregressive integrated moving average (ARIMA) models [7, 8], or machine learning-based models [5, 7] have been applied to predict the tap-holes opening and closing time, accuracy has not been high enough (around 81%) to apply these approaches in a real environment.

In this work, we propose a novel CNN model (S-CNN) with Skip-dense layers, which has a dense layer with skip connections and four CNN branches as shown in Fig. 9, to improve the aforementioned blast furnace process. We model the opening and closing time for each tap-hole as a multivariate time series. Then, we propose to forecast the remaining time to close each tap-hole at every minute. Our baseline model is composed of three different LSTM blocks. We chose the LSTM based approach as a baseline, since LSTMs are widely used to model and predict time series problems.

We train and experiment our deep learning model with real sensor datasets collected from the POSCO blast furnace, which has about 5,000 data fields and 700,000 records. They are collected over every minute for 16 months, totaling 3.5 billion items. These datasets include 1) sensor data such as temperature, and pressure, 2) periodical measurement data such as molten iron temperature or chemical ingredient of coke and ore, 3) operational control parameters such as air blows, coke to ore ratio, and 4) calculated values from the other data. We explain our dataset in more details in Section 3.

With these datasets, we achieved over 90% accuracy in prediction performance. In addition, we have deployed our algorithms and models since Jan. 2018 at POSCO, and have successfully achieved similar 90% accuracy. This clearly demonstrates that deep-learning based models can provide high production efficiency, while lowering manual human efforts for one of most challenging steel production processes. Our contributions are summarized as follows:

- Our research aims to predict the remaining time to open and close tap-holes directly for a blast furnace, unlike much prior research which explored hearth level.
- We developed the novel Skip-dense layer (S-CNN) model and achieve 90% accuracy within a 30 min tolerance time interval with real dataset from Gwang-yang Steel Works.
- We successfully deployed our models at Gwangyang Steel Works and have achieved 90% accuracy in a real environment, expecting POSCO to save \$1.1 million operation cost per year.

2 BACKGROUND AND RELATED WORK

Blast Furnace in Steelmaking: Steelmaking is a complex process for producing steel from iron ore and scrap. The blast furnace is an essential facility in the steelmaking process for producing hot molten iron from raw materials such as iron ore and coke. Figure 1 describes the inside of a blast furnace, where iron ore and coke are piled up layer-by-layer. Air with high temperature and high pressure is blown from tuyere at the bottom of the blast furnace, then iron is melted from iron ores and flowed down to the hearth, in the form of hot melted iron and slag. During this process, other gases such as N_2, CO, and CO_2 are emitted upward. Unreacted coke is accumulated at the bottom of the hearth. In particular, POSCO's blast furnace, there are many sensors installed and form sensor array (or mesh) on the outside of the wall of the blast furnace, such as temperature and pressure sensor array.

After drilling the hearth to open a tap-hole, the hot molten iron and slag, and toxic gas are flowed out. Then, molten iron and slag run into torpedo ladle car (TLC) along its path as shown in Fig. 1. Slag is divided with molten iron, according to its gravity difference. TLC takes molten iron to the revolving furnace to remove impurities in the molten iron and make iron into steel. Then, steel can be processed into wire or plate as a product of a steel plant.

To operate a blast furnace efficiently, it is necessary to precisely analyze and predict the state of the blast furnace and adjust the operating factors, accordingly. Also, there are hundreds of important factors that can influence the smooth operation of the blast furnace. Especially, the hearth liquid level indicates how much of the molten iron is stored in the hearth. It is one of the major indicators to determine the opening and closing time of the tap-hole. However, the temperature of the blast furnace is up to 2,300℃, and no sensors can be sustained at this temperature. Therefore, it is extremely challenging to measure the exact hearth liquid level in the hearth. As the liquid drainage continues, the level of the hearth liquid decreases, and eventually gas is ejected from the tap-hole. This is called "gas leakage", which includes dust and toxic carbon monoxide. Hence, blast furnace operators need to prepare to close a tap-hole and open another tap-hole, when there is gas leakage. Unfortunately, operators cannot be sure about when to close and open solely based on a gas leakage sign, because even if there is a gas leakage, it is still possible that there is enough liquid in the hearth. That is why blast furnace operations are highly dependent on operators' experience and expertise.

Prior research: Traditionally, the hearth liquid level is predicted and approximated by the complex mathematical equations. Agrawal et al. [4] constructed the hearth liquid level monitoring system using a large number of equations based on domain knowledge about oxygen rate, slag, hot metal volume, mass, etc. The mathematical equations-based models have advantages in reflecting domain-specific knowledge. However, it also has some disadvantages with several assumptions and simplifications, which may not always hold in a real environment. First, the equations cannot reflect all the complex factors in the blast furnace. Second, observations for these factors may include noise, where mathematical equations are limited to model simple types of noise models.

On the other hand, Gomes et al. [7, 8] used a totally different approach. They applied electromagnetic force (EMF) sensors on the outside of the wall of a blast furnace to measure the hearth liquid level indirectly. Although they deployed dedicated sensors to measure EMF signals, the real ground truth data - the hearth liquid level - is not measurable. Therefore, they had to depend on the assumption that EMF signals are proportional to the hearth liquid level. Based on this assumption, they regressed EMF signals and predicted the future values of EMF signals to estimate the level of liquid in the hearth.

Multivariate time series prediction: Deep learning is suitable to model multivariate time series, because it does not require hand-crafted features. For example, a multivariate time series collected sensor data can be used for predictive maintenance. Liao and Ahn [12] combined LSTM and MLP layers for asset health management. Malhotra et al. [13] used LSTM to find abnormal state from the power demand time series. Also, Cui et al. [6] used CNN model for time series. Their approaches apply smoothing methods using various filter sizes and downsampling methods with multi-scales. All these branches have 1D CNN, and then they concatenated together. Yao et al. [15] composed a network, which has CNN time frame feature extraction and GRU time series prediction part. Also, Mehdiyev et al. [14] researched the prediction problem to determine whether post-processing is required to remove the steel surface defects in making a steel slab product. They used a deep learning model with stacked LSTM auto-encoder and data from sensors. Recently, Hsu et al. [10] proposed Multi-Domain Deep Neural Network (MDCNN) using time and frequency domain data. In this work, our goal is to predict the remaining time of opening and closing events for the blast furnace using both LSTM and CNN models. With CNN and RNN models, we focus on predicting tapping behavior directly rather than modeling the liquid level in the hearth, which much prior research has tried to model.

3 POSCO GWANGYANG STEEL WORKS DATASET DESCRIPTION

We train and evaluate our model using data from a blast furnace in POSCO Gwangyang Steel Works, where various sensor data have been accumulated and recorded at every minute from September 2016 to December 2017. Dataset has 5,000 data fields from already deployed sensors and we obtained 5,000 types of data per every 2 seconds. We average 30 of each data field for a minute.[2] Consequently, we produced more than 700,000 records every minute for 16 months. Therefore, it is a very large dataset with more than 5,000 $\times$ 700,000 = 3.5 billion items in total.

Data fields analysis and selection. Although there are 5,000 data fields, it is not necessary to use all of the data fields for prediction. Initially, domain experts in POSCO recommend about 800 data fields, which they believe in having a close relationship with opening or closing tap-hole events. Unfortunately, some data fields have missing or garbage values due to sensor failures for an extended period. Therefore, we have to discard a few data fields. In addition, we analyze data with a correlation matrix and PCA. However, it did not help much to select essential data fields, because none of the data field has significant correlations.

Next, we extensively tried many different combinations of data fields to find a set of best parameters. For example, the temperature sensor data field was helpful to improve prediction accuracy, but the heat load data field and pressure sensors data field did not. We performed many empirical studies, and further built and trained about 200 different models with more than 30 different structures. Although we cannot prove our feature combination is optimal, they empirically achieved the best performance in our study.

After performing feature analysis and empirical studies extensively, we chose the final 278 data field for our LSTM-3 baseline model, where we focus on capturing the average temperate values, pressure, and operational data more. For our Skip-dense-CNN (S-CNN) model, 411 data fields are used as shown in Table 1, in order to capture the temperature sensor values more effectively using CNNs. We describe more details about our models in the next section.

Detailed data fields. Table 1. presents all the data fields used by our 'Skip-dense-CNN (S-CNN)' and 'LSTM-3', where 'Type' means the type of data fields, and 'Data field' represents the name of the data field group, and 'Dims' refers to the dimension of the sensor array of data field group. In total, we have 5 different types of data fields as shown in Table 1. In the temperature type, SB represents a temperature sensor values at shaft and Stave is a temperature sensor at bosh, and HW12, HW36, and HW78 are temperature sensor values located at the hearth wall level 1~2, 3~6, and 7~8, respectively. For the pressure type, SP 6~8 are used, which represent pressure sensor values at the shaft. For molten iron type, we use the molten iron casting speed and accumulated amount of molten

[2] In this paper, the definition of the data field and data records follows the definition of conventional relational database. Data fields are types of data from each sensor, and data records are values for all the data fields at every moment.

iron received for all four tap-holes. Also, we use the wind blow amount from 44 tuyere. In addition, we use operational data fields, which include average temperature at the top of the blast furnace, total wind amount, wind pressure, amount of enriched oxygen, amount of ore and coke loaded in a certain time, quality of ore and coke, mix rate of coke with various granularity, ingredient of coke and ore, silicon and sulfur in slag and hot metal, as shown in Table 1. For our baseline model (LSTM-3), we use temperature only at the height of tap-holes. We also use pressure data at each height (SP 6-8), AB probe data, and operational data. Although we used fewer data fields for the LSTM-3 model to train, it incorporates a total of 83 operational data fields, which are more than the number of operational data used by S-CNN. The major difference in the number of data fields used by LSTM-3 and S-CNN is caused by the number difference in the temperature type in Table 1. While our S-CNN focuses more on capturing general temperature data fields, the LSTM-3 baseline tends to characterize more on the temperatures at the certain geometrical locations or regions (e.g., HW36 is around the center of tap-holes) in a blast furnace.

Table 1: Summary of data fields used by the S-CNN and LSTM model, where some data such as temperature and pressure, heat load are acquired from sensor array on the outside of the wall of blast furnace (Acc. means accumulated, Amt. means amount, and Prop means property).

Type	Data field	Dims.	S-CNN	LSTM-3
Temperature	SB (shaft)	7×12	84	0
	Stave (bosh)	4×8	32	0
	HW78 (hearth 7∼8)	2×26	52	0
	HW36 (hearth 3∼6)	4×22	88	88
	HW12 (hearth 1∼2)	2×18	36	0
	AB probe	4×6+1	0	25
	Gas at top	0	0	6
Pressure	SP 6∼8 (pressure)	3×8	0	24
Molten Iron	Casting speed	4	4	4
	Acc. amount	4	4	4
Tuyere	Tuyere	44	44	44
Operational data	Avg. temperature	0	9	0
	Wind blowing	0	3	8
	Permeability	0	7	3
	Amt. of material	0	15	27
	Prop. of material	0	26	34
	Produced material	0	7	5
	Produced gas	0	0	6
Total			411	278

4 OUR SKIP-DENSE CNN APPROACH

Our model predicts the remaining time $(Y - t)$, where Y is the future tap closing time, and t is the current time. And our model's inputs are various sensor and operational data fields from Table 1, and the output of the network is the remaining time left to close for four tap-holes. More formally, at any current time t, our model predicts the remaining time until the tap closing time $Y - t$ of each tap-hole. We used 360 minutes time window, because raw materials load at the top of the blast furnace, which needs 6 hours to melt and flow down to the hearth. Therefore, in summary, our Skip-dense-CNN (S-CNN) model takes 411×360 = 147,960 inputs, while LSTM-3 baseline takes 278×360 = 100,080 inputs per minute.

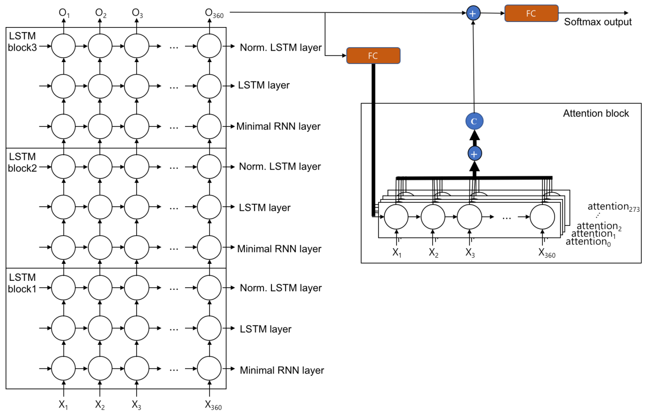

Figure 2: Description of the LSTM-3 model, which has 3 layers (left-side) with minimal RNN, LSTM, and normalized LSTM. It also has an attention block (right-side) and we apply the attention block before the final dense layer.

In addition, we tried various RNN model structures with different types of layer such as vanilla RNN, LSTM, minimal RNN, and various regularization methods. Also, we found that the following LSTM-3 model performed the best and chose the best performing LSTM-3 as a baseline. We also tried the various structures with different hyperparameters for CNN models as follows: 1 to 6 branches, 7 to 107 layers, skip-connections, different initialization methods, and fully-connected or Skip-dense-CNN classifier, global average pooling, input or output dropout, and regularization. Due to space limitations, we only describe the detail of our final model structure of our Skip-dense-CNN and the best performing baseline LSTM-3 model.

4.1 Preprocessing

All data fields are numerical and their update intervals vary from every 2 seconds to several hours and sensor data are acquired every 2 seconds. On the other hand, manual measurements such as the 'slag temperature' data field are updated every 6 hours. Hence, we calculate one minute average of all data fields, which is updated every 2 seconds. Then, we normalize those 1 minute average data as follows: $\mathbb{D}_n = \frac{(\mathbb{D}-\mu)}{\sigma^2}$, where $\mathbb{D}$ is each minute data, μ is average of $\mathbb{D}$ for the entire train period, and σ^2 is variance.

4.2 LSTM-3 baseline model

Our baseline LSTM-3 model is described in Fig. 2, where its input sequence length is 360 minutes. In the LSTM-3 model, we hypothesize that the recent state of the blast furnace is more important than old state. Hence, we summarize observations in far past, while keeping the details for recent observations using LSTM-3.

To summarize the older observations, we use average pooling. We have 278 input values at every minute during 360 minutes, so that input sequence length is 360. It is divided into 4 sub-sequences

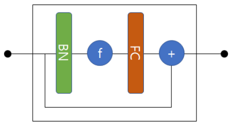

Figure 3: Skip-dense module, where BN is batch normalization, f is activation function, FC is a Fully-connected layer, and '+' indicates the addition of feature map.

of lengths 181, 112, 45, and 22. Then, we apply the average pooling of different sizes and strides, and finally, combine them to make the reduced sequence of length 86. We call this 'Sequence pooling'.

In Table 2, we describe the detail of Sequence pooling. As shown in Table 2, for the past 360 minutes to 180 minutes of input data time window (L1), we apply average pooling with its filter size 15 and stride 8. Also, 179~68 minute before (L2), the filter size 9 and stride 5 are applied. In addition, for L3 (67~23 minute prior), filter size 3 and stride 2 are used. There is no average pooling for the input values between 22~1 minute (L4) prior. In addition, Sequence pooling can make LSTM-3 network lighter. Therefore, the reduced sequence is fed into stacked LSTM blocks. The last block outputs predicted values for the remaining time left to close $(Y - t)$ of four tap-holes, where each represents the 480-valued multinomial outputs. Therefore, total outputs of model is composed of total $480 \times 4 = 1,920$.

We stacked three LSTM blocks, where each of which has three RNN or LSTM layers as shown in Fig. 2. Each LSTM block consists of minimal RNN, LSTM, and normalized LSTM layers. The number of hidden units for each layer are 400, 400, and 800, respectively. We also attached skip-connections across the input to the output of each LSTM block, which has L2 regularized dense layer to match shapes between LSTM block output and skip-connection output before addition. After the last LSTM block output is followed by two Fully-connected layers and attention module to build final 1,920 outputs. Each layer is regularized with a dropout rate = 0.5. We also add the additional dropout layer at the input layer, which keeps the ratio to be 0.8. Finally, we apply regularization loss for the Fully-connected layers.

Table 2: Pooling filter size and strides for Sequence pooling, where these provide different level of details by summarizing the distant past and the near past time sequence values. The 'min. before' indicates the time range of the sequence prior to the current time.

Sequence (min. before)	Sequence length	Pooling-size	Pooling stride	Compressed length
L1 (360~180)	181	15	8	21
L2 (179~68)	112	9	5	21
L3 (67~23)	45	3	2	22
L4 (22~1)	22	(no pooling)	(no pooling)	22
Total	360	-	-	86

4.3 Our Skip-dense-CNN (S-CNN) model

As shown in Fig. 9, our Skip-dense-CNN has four branches as follows: 1) Temperature branch, 2) Molten iron drainage branch, 3) Tuyere branch, and 4) Operational data branch. Features from these four branches are concatenated, then fed into 10 Skip-dense layers + a Fully-connected layer to generate four outputs for the remaining time left to close $(Y - t)$ for each tap-hole. We propose the novel Skip-dense structure to add more depth for better prediction, where the Skip-dense layer is fully connected layers with skip-connections at every layer as shown in Fig. 3. We now describe the details of four branches and Skip-dense layers and provide the intuition on how we designed each branch.

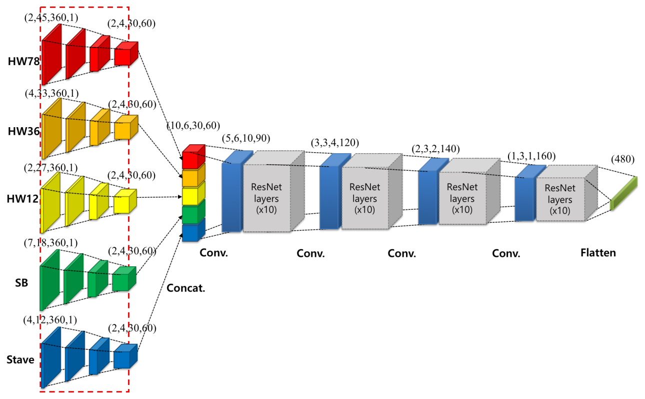

Figure 4: Temperature branch, where temperature sensor data are obtained from the hearth (HW12, HW36, and HW78), bosh (Stave), and shaft (SB). 'Conv.' means a convolutional layer and 'Concat.' indicates concatenation. Stem layers are 3 for each branch, and each 'ResNet' (gray box) has 20 convolutional layers. Therefore, we have $3 + (1 + 20) \times 4 = 87$ convolutional layers in Temperature branch.

Temperature branch. The blast furnace is shaped similar to a cup as shown in Fig. 1 and temperature sensors are placed around the cup. If we cut the cup horizontally, it becomes a tube shape. Therefore, we fit temperature sensors in the standard CNNs. However, if we cut the tube at 0 degree angle, which is in the middle of tap-hole 1 and tap-hole 4, then we lose some locality information near the cutting area. Though 359 degree angle and 0 degree angle is originally very close, they would be the farthest point after cutting. To address this issue, we apply data padding by copying and pasting half of the data (0 to 180 degree angle) on the other side of the tube.

Since temperature values are measured from three different areas of a blast furnace, we divide the CNN layers to the following three different groups: hearth, bosh, and shaft. Then, we further divide the hearth temperature into the following three areas based on the sensor location: HW12 (bottom of the hearth), HW36 (middle of the hearth), and HW78 (top of the hearth). Therefore, we have a total 5 groups as shown in Fig. 4 but each group has different 2D shapes including data padding as follows: 7×18, 4×12, 2×27, 4×33, and 2×45, respectively. Therefore, we have to design 5 different stem layers as shown in Fig. 4, where they are the leftmost three layers in Fig. 4. All convolutional layers in those stems have strides to reduce the size of the feature map. Though each HW78, HW36, HW12, SB, and Stave have different input dimensions, after

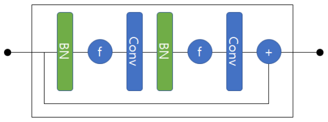

Figure 5: ResNet module, where BN indicates batch normalization, f is an activation function, Conv is a convolutional layer, and '+' represents the addition of feature map. This module is repeated for 10 times in the gray box of Temperature branch (Fig. 4), Molten iron drainage branch (Fig. 6) and Tuyere branch (Fig. 7).

passing those stem layers, they are reduced to the same feature map size, (2, 6, 30, 60). Finally, a feature map from every stem layer is concatenated along the axis of blast furnaces at a concatenation module ('concat' in Fig. 4). Then, we obtain the final concatenated feature map size (10 ,6, 30, 60). Then, in order to make it deeper, we employed the ResNet structures, where we designed it to be a total of 87 convolutional layers. The detailed ResNet structure is described in Fig. 5. After ResNet, we flatten the output of ResNet and obtain the 480 sized feature vector.

As shown in Fig. 4, Temperature branch is the largest branch in our model, which has 292 fields in total among 411 fields, as discussed in Table 1. Because temperature sensor array has 2D shape and time adds one more dimension, the Temperature branch is built with 3D CNNs. In Fig. 4, numbers separated by commas above feature maps denote their dimension. For example, (3, 3, 4, 120) indicates its dimension is $3 \times 3 \times 4 \times 120$.

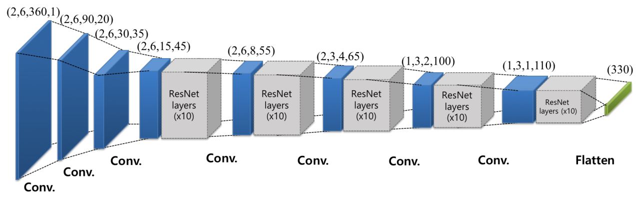

Figure 6: Molten iron drainage branch, where inputs are 1) molten iron casting speed and 2) received the molten iron amount for four tap-holes (2×4). With data padding, it becomes 2×6. Because each 'ResNet' block has 20 convolutional layers, we have a total of 107 layers in this branch.

Molten iron drainage branch. We provide the detailed structure of the Molten iron drainage branch in Fig. 6, where molten iron casting speed and molten iron received amount are used as inputs to this model. They are $2 \times 4 = 8$ data fields in total as shown in Table 1, which are molten iron casting speed and received molten iron amount for four tap-holes. Because 2D shape (2×4) and time (360 minutes) adds another dimension, the Molten iron drainage branch is naturally built with 3D CNNs. That is Molten iron drainage branch has 7 convolutional layers (blue boxes) + 5 ResNets (gray boxes), where each ResNet has 20 convolutional layers. Therefore, the Molten iron drainage branch has 107 convolutional layers. In addition, every convolutional layer in Fig. 6 have strides, so that the size of feature map is reduced from $2 \times 6 \times 360$ to $1 \times 3 \times 110$. After the flattening (a rightmost cyan box in Fig. 6), the shape of the generated feature map from Molten iron drainage branch is 330.

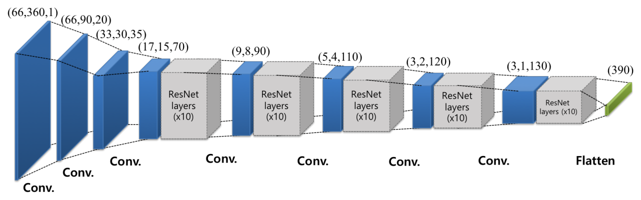

Figure 7: Tuyere branch, whose input is 44 tuyere's wind blow amount per minute data and 66 with data padding. In total, we have 107 layers in this branch.

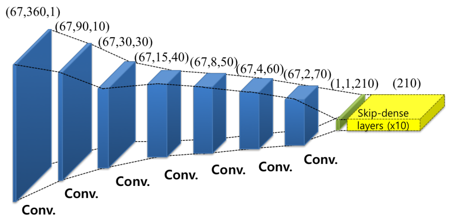

Figure 8: Operational data branch, whose input is 67 operational data. We have 6 conv1×1 layers, one convolutional layer, and 10 Skip-dense layers in this branch.

Tuyere branch. We provide the detailed structure of Tuyere branch in Fig. 7, where Tuyere branch uses a wind amount of each 44 tuyeres as described in Table 1. Because 1D shape and time (360 minutes) also add one more dimension, the Tuyere branch is constructed from 2D CNNs. In particular, the Tuyere branch has 7 convolutional layers (blue boxes) and 5 ResNet modules (gray boxes), which have 20 convolutional layers each. Therefore, the Tuyere branch has 107 layers in total. The size of the feature is reduced from 44×360 to 3×130 by strides in convolutional layers. After the flattening, the shape of the generated feature map becomes 390.

Operational data branch. The structure of the Operational data branch is depicted in Fig. 8. In our Operational data branch, we varied the different number of operational log fields, as shown in Table 1. After trying a different number of operational data fields, we chose 67 data fields based on empirical measurements, which yielded the best evaluation performance.

Though operational data is fed into the single branch, its elements are a set of independent scalar values, so that each element has only temporal locality and should be processed independently. Therefore, the Operational data branch is built with 1D CNNs (conv1×1). During 6 of Conv1×1 layers, it is processed only for temporal domain for each element. Then, it is fed into one convolutional layer, whose filter size is (67×2) so that the convolutional filter can cover the whole 67 elements. Hence, its operations are identical to a Fully-connected layer. The size of feature map is reduced from 67×360 to 1×210. Then, the feature map is fed into 10 Skip-dense layers (yellow box in Fig. 8) to add more depth to the

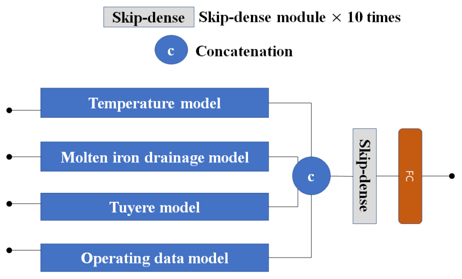

Figure 9: Combined model, which concatenate feature map from four branches (Figure 4, 6, 7, 8). The concatenated feature map is fed to 10 Skip-dense layers and 1 Fully-connected layer to produce 4 output logits. This is a large DNN model, which has four branches, 107 convolutional layers, 10 Skip-dense layers, and one Fully-connected layer.

Operational data branch and the size of feature map is not changed in Skip-dense layers.

Combined final model. Finally, features from the previous all four branches – 1) Temperature branch, 2) Molten iron drainage branch, 3) Tuyere branch, and 4) Operational data branch – are concatenated together as shown in Fig. 9. Therefore, the concatenated feature vector becomes 480 + 330 + 390 + 210 = 1,410 as an one dimensional vector. Then, the resulting 1,410 dimensional vector is fed into 10 × Skip-dense layers + one dense layer. And, we finally generate 4 logits for 4 each tap-hole, which are the final outputs to predict the remaining time left to close for 4 tap-holes.

5 EXPERIMENTAL SETUP

We build our deep learning models using Tensorflow. We train our models using data from September 2016 to August 2017 from POSCO and use data from September 2017 to December 2017 for testing.

For the baseline LSTM-3 model, we used the momentum optimizer. Activation functions are *tanh* for LSTM layers and ReLU for MLP layers. LSTM-3 model is trained about 40,000~130,000 iterations to obtain the best result, which is about 3 epochs. Because the sequence length of each input is 360 minutes, and each input is randomly sampled, 130K iterations are already covered all training set. It takes about 3 days using i7-6850K and GTX 1080ti to train and the batch size is 17. We used dropout for both input and output MLP classifiers, where the dropout rate of input is 0.2, and the output is 0.5. To improve performance of LSTM-3 model, we boosted data from 0 to 75 minutes near closing event during training. LSTM-3 model's loss function is the sum of two loss functions L_t and L_R. L_R is L_2 regularization loss for the average of parameters from each layer to prevent overfitting. L_t is described in Eq. 1, where l_n^t is the cross entropy loss for each tap-hole, and F_n is the validity indicator for each tap-hole. The validity is 1, when the tap-hole is opened, 0 otherwise.

$$L_t = \sum_{n=1}^{4} l_n^t \times F_n \qquad (1)$$

For the Skip-dense-CNN model, we used the ADAM optimizer and activation function for all layers is ReLU. We train the Skip-dense-CNN model around 100 to 200 epochs to obtain the best result. It took 5 to 6 days using Xeon E5-2660 and GTX 1080ti. The batch size was 100. Each ResNet layer and Skip-dense layers have batch-normalization. Because the Skip-dense classifier already has L_2 regularization term to stop overfitting, we did not use dropout. Similar to LSTM-3, we use sum of the two loss functions L_t and L_R but in this case, L_R is L_2 regularization loss only for Skip-dense layers, and l_n^t in Eq. 1 is the root mean square error (RMSE) instead of cross entropy.

6 RESULT

Our models predict the remaining time left to close tap holes at every minute. Figure 10 is the predicted result and label for the remaining time left to close the tap-hole, where the X-axis is the time, and the Y-axis is the remaining time, which is predicted by our model. The blue line is the correct answer, and the orange line is predicted as the result in Fig. 10. The remaining time decreases linearly every minute (blue line). Prediction (orange line) is performed every minute. As we can observe from Fig. 10, our predicted results closely follow the label. Therefore, we can examine that our model works properly.

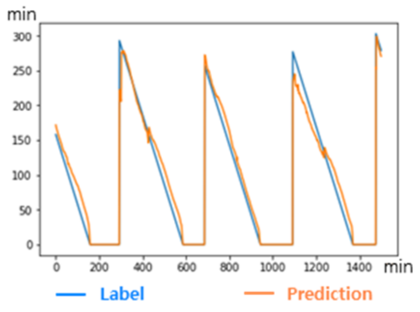

Figure 10: Predictions and labels of the remaining time left to close a tap-hole, where the X-axis is time, and the Y-axis is the remaining time. The blue line is the label, and the orange line is the prediction result. The label is calculated from the actual closing event, decided by operators.

To evaluate our model quantitatively, we use the following three different time periods: 1) when 40~60 minutes left, 2) when 20~40 minutes left, and 3) when 10~20 minutes left to close the tap-hole. , where these time periods are defined by POSCO. POSCO remarked that predicting 40~60, and 20~40 minutes earlier is helpful to operate TLCs and to predict 10~20 minutes sooner is useful for blast furnace opening and closing operations.

In addition, we use the following 5 different types of error tolerance ranges: ±5 minutes, ±10 minutes, ±15 minutes, ±20 minutes, and ±30 minutes. Above are the important tolerance ranges for blast furnace operations, because the closing tap-hole has to be prepared by operators from 15 to 30 minutes, and TLC has to be prepared about one hour before tap-holes closing. Then, we calculate the accuracy for the aforementioned each time period and error tolerance range. For example, if 38 minutes are left from the closing and our model predicts 26 minutes left. Then, it is counted as an incorrect prediction for ±5 minutes and ±10 minutes error tolerance ranges. However, it is considered as a correct prediction for ±15 min, ±20 min, and ±30 minutes error tolerance ranges. We provide the prediction accuracy of LSTM-3 baseline vs. our Skip-dense-CNN models in Table 3.

As shown in Table 3, LSTM-3 baseline's accuracy was 72.3%, when 40~60 min. left with ±30 min. tolerance, and 43.0% when 20~40 min. left with ±15 min. tolerance. Its average RMSE was 23.1 and 22.2 min, when 40~60 min. left and 20~40 min. left, respectively. On the other hand, Skip-dense-CNN model's accuracy was 90.5%, when 40~60 min. left with ±30 min. tolerance, and 70.6% when 20~40 min. left with ±15 min. tolerance as shown in Table 3. Its average RMSE was 18.3 minutes and 16.2 minutes when 40~60 minutes left and 20~40 minutes left, respectively.

In summary, the Skip-dense-CNN model is outperformed over LSTM-3 baseline in all cases.

We believe that higher performance is achieved because CNN's independent predictions with a certain time window are safer from overfitting than LSTM's predictions with sequences, especially, when we have a small amount of data. Although we have about 700,000 data records in our research, they are still hundreds of times in terms of opening and closing activities. Therefore, CNN could capture data dependency better. Our result is also consistent with recent findings and results on multivariate time series [6, 10], which shows that CNN is effective for predicting multivariate time series.

Also, we learned that accuracy is even more accurate than human operation error. Though it is not managed quantitatively, POSCO believes that human operator error exceeds 30 minutes on average and up to more than an hour. Therefore, we decide to deploy our Skip-dense-CNN model for the actual monitoring system at Gwangyang POSCO Steel works to help blast furnace operation.

7 DEPLOYMENT RESULTS AND DISCUSSION

Real-world deployment and results. Because the Skip-dense-CNN model outperformed over the LSTM-3 baseline, we actually deployed the Skip-dense-CNN model to POSCO's blast furnace monitoring system. We deployed our model from January 2018 to July 2018, processing real-time sensor data for prediction. We build a TCP socket based server-client communication module and receive sensor data values in real time from the POSCO data server. When we obtain the data, Redis publishes to alarm and then it is stored at MySQL database. At every minute, we load the data and calculate average values for each data field, then save it to MySQL DB. At the same time, we load the average data for the past 360 minutes, normalize them, and predict the upcoming closing time. Finally, we provide the prediction result back to POSCO data server, and it is displayed on the monitoring screen for operators in POSCO's Gwang-yang Steel Works. Figure 11 is a screenshot of our blast furnace monitoring system deployed at POSCO. On the screen, there are four figures of the remaining time to close for each tap-hole at the right half of the screen. The same graph in Fig. 10 is displayed in the figure, when each tap-hole is opened.

We present the results in Fig. 12. After one tap-hole is closed by operators, we check prediction result of our S-CNN model in the past. We find out that the prediction performance from February to April is lower than offline evaluation performance. We found that POSCO used wet coke a lot in those periods, where wet coke is moister than normal coke requiring operators to intervene more to stabilize blast furnace. Therefore, the accuracy was lower than 90% temporarily. From May to July - after the wet coke period is finished, S-CNN produced highly accurate prediction performance for the blast furnace. From April 2018, the prediction accuracy reaches 91% when 40~60 minutes left. It is even better than the evaluation period performance. When 20~40 minutes left, the accuracy is 67.0~70.8%. It is similar to the offline evaluation performance. Despite unforeseen changes and events in real environments, overall our Skip-dense-CNN algorithm successfully achieved the expected performance and generalized the results across different months, so that it minimized the error, even in the wet coke period.

In Table 3, we can observe that it is not easy to reduce the error tolerance range even if the remaining time is short. Through discussion with POSCO, we learned that a large part of this error could possibly stem from inaccuracy of labels. Because the label is the time that operators press a button, not the real tap-hole closing time. Some operators press the button before they close a tap-hole, while other operators press the button after all closing jobs are completed. The time difference could be up to more than tens of minutes. Even a mud gun could fail to close taps sometimes, then the whole closing process has to be re-initiated, and the time gap will be larger than normal. Currently, POSCO is further investigating an approach to improve the accuracy of labels.

Summary and Comparison to other models. Figure 3 shows the structure of the Skip-dense layer, where it is a fully connected layer with skip-connection. We can stack Skip-dense up to 10 layers, which can model complex function. Although fully connected layers are difficult to train because of its enormous number of parameters, we can make it still trainable using skip-connection and regularization. Skip-dense is different with ResNet [9], which has skip-connection every two convolutional layers. It is also different from DenseNet [11], which does not add feature maps, but we stack feature maps. Finally, all these models are concatenated, fed into Skip-dense layers.

8 CONCLUSION

We built a deep learning model to predict the remaining time left to close each tap-hole of a blast furnace in POSCO Gwangyang Steel Works. Our model is trained and evaluated with 700,000 data records, which are collected every minute for 16 months. Although predicting the remaining time left to close tap-holes is an extremely challenging problem, we successfully predicted with over 90% accuracy in the offline evaluation as well as in a real environment. Our model is successfully deployed to the real-time monitoring system for the blast furnace in POSCO, and it is currently assisting human operators to decide when to close each tap-hole. POSCO expects that our model can save about \$1.1 million operation cost per year by preventing a temperature drop of TLC by 14.5℃ on average. In addition, we believe that less skilled operators can receive significant support from our model as well as reducing risks.

Table 3: Prediction accuracy results and RMSE averaged for all tap holes, where LSTM-3 is our baseline model and Skip-dense-CNN (S-CNN) is our proposed model.

Time-left \ Tolerance		±5 min	±10 min	±15 min	±20 min	±30 min	RMSE
40~60 min	LSTM-3 (baseline)	17.7%	32.0%	43.8%	54.1%	72.3%	23.1 min
	S-CNN (ours)	22.9%	44.8%	62.7%	76.2%	**90.5%**	**18.3 min**
20~40 min	LSTM-3 (baseline)	16.8%	30.5%	43.0%			22.2 min
	S-CNN (ours)	26.1%	50.4%	**70.6%**			**16.2 min**
10~20 min	LSTM-3 (baseline)	30.7%					20.2 min
	S-CNN (ours)	37.7%					14.9 min

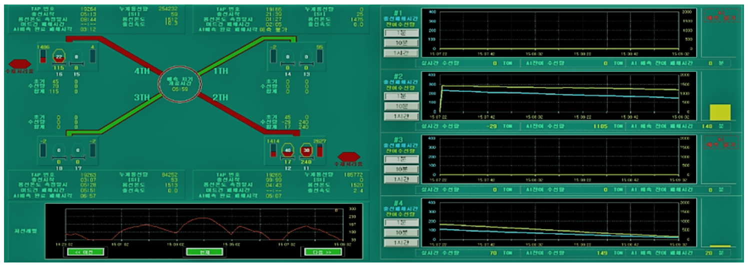

Figure 11: Blast furnace monitoring system screen: Right side on the screen shows time - $(Y - t)$ plots (prediction for the remaining time to close tap-holes as described in Fig. 10).

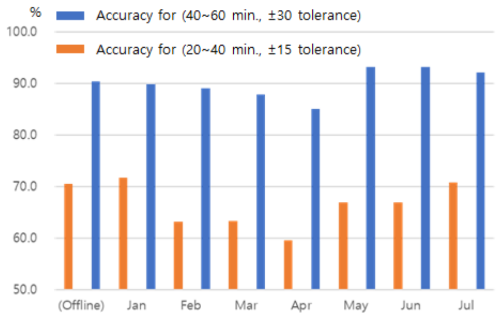

Figure 12: Accuracy of predictions when deployed from January to July 2018: we applied our model to real-time data and have achieved equivalent performance to offline evaluation. Especially, accuracy for 40 – 60 exceeds over 90% in normal operation state (May to July).

9 ACKNOWLEDGMENTS

We would like to thank Xianshu Jin, Sangyup Lee, Youjin Shin, and Shahroz Tariq for their valuable feedback. This research was supported in part by POSCO Gwangyang Steel Works grant and funded by the MSIT, South Korea, through the ICT CC Program under Grant IITP-2019-2011-1-00783, and by NRF of Korea by the MSIT (NRF-2017R1C1B5076474 and NRF-2019M3F2A1072217).

REFERENCES

[1] 2016. Posco's world's largest steel mill in Gwangyang boosts capacity by 10% after renovation. https://pulsenews.co.kr/view.php?year=2016&no=410215
[2] 2017. Equipment for steel makers. http://www.wabicorp.com/
[3] 2017. World Steel in Figures 2017. https://www.worldsteel.org/media-centre/press-releases/2017/world-steel-in-figures-2017.html
[4] Ashish Agrawal, Swapnil C. Kor, Utpal Nandy, Abhik R. Choudhary, and Vineet R. Tripathi. 2016. Real-time blast furnace hearth liquid level monitoring system. *Ironmaking & Steelmaking* 43, 7 (2016), 550–558. https://doi.org/10.1080/03019233.2015.1127451 arXiv:https://doi.org/10.1080/03019233.2015.1127451
[5] Jian Chen. 2001. A predictive system for blast furnaces by integrating a neural network with qualitative analysis. *Engineering Applications of Artificial Intelligence* 14, 1 (2001), 77–85.
[6] Zhicheng Cui, Wenlin Chen, and Yixin Chen. 2016. Multi-Scale Convolutional Neural Networks for Time Series Classification. (03 2016).
[7] Flavio S. V. Gomes, Klaus F. Côco, and José Leandro Felix Salles. 2017. Multistep Forecasting Models of the Liquid Level in a Blast Furnace Hearth. *IEEE Trans. Automation Science and Engineering* 14, 2 (2017), 1286–1296. https://doi.org/10.1109/TASE.2016.2538560
[8] F. S. V. Gomes, J. L. F. Salles, and L. A. Wasem. 2011. A new prediction model for liquid level in blast furnaces based on time series analysis. In *2011 9th IEEE International Conference on Control and Automation (ICCA)*. 772–777. https://doi.org/10.1109/ICCA.2011.6137926
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[10] En-Yu Hsu, Chien-Liang Liu, and Vincent S. Tseng. 2019. *Multivariate Time Series Early Classification with Interpretability Using Deep Learning and Attention Mechanism*. 541–553. https://doi.org/10.1007/978-3-030-16142-2_42
[11] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2016. Densely Connected Convolutional Networks. *CoRR* abs/1608.06993 (2016). arXiv:1608.06993 http://arxiv.org/abs/1608.06993
[12] Linxia Liao and Hyung il Ahn. 2016. Combining Deep Learning and Survival Analysis for Asset Health Management. *IJPHM* 7 (2016), 7. https://www.phmsociety.org/node/2192
[13] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. 2015. Long short term memory networks for anomaly detection in time series. In *Proceedings*. Presses universitaires de Louvain, 89.
[14] Nijat Mehdiyev, Johannes Lahann, Andreas Emrich, David Enke, Peter Fettke, and Peter Loos. 2017. Time Series Classification Using Deep Learning for Process Planning. *Procedia Comput. Sci.* 114, C (Nov. 2017), 242–249. https://doi.org/10.1016/j.procs.2017.09.066
[15] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. 2017. DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 351–360. https://doi.org/10.1145/3038912.3052577