Primal-Optimal-Binding LPNet: Deep Learning Architecture to Predict Optimal Binding Constraints of a Linear Programming Problem

Natdanai Kafakthong, Krung Sinapiromsaran
Department of Mathematics and Computer Science
Chulalongkorn University
Bangkok, Thailand, 10330

***Abstract*—Identifying an optimal basis for a linear programming problem is a challenging learning task. Traditionally, an optimal basis is obtained via the iterative simplex method which improves from the current basic feasible solution to the adjacent one until it reaches optimal. The obtained result is the value of the optimal solution and the corresponding optimal basis. Even though learning the optimal value is hard but learning the optimal basis is possible via deep learning. This paper presents the primal-optimal-binding LPNet that learns from massive linear programming problems of various sizes casting as all-unit-row-except-first-unit-column matrices. During the training step, these matrices are fed to the special row-column convolutional layer followed by the state-of-the-art deep learning architecture and sent to two fully connected layers. The result is the probability vector of non-negativity constraints and the original linear programming constraints at the optimal basis. The experiment shows that this LPNet achieves 99% accuracy of predicting a single binding optimal constraint on unseen test problems and Netlib problems. It identifies correctly 80% LP problems having all optimal binding constraints and faster than cplex solution time.**

Keywords*—*Deep learning; convolution neural network; linear programming; basic feasible solution; optimization

I. Introduction

Traditionally solving a linear programming (LP) problem. The authors in [17] requires an iterative simplex method [1] or an iterative interior point method [2]. The simplex method starts from an initial basic feasible solution (BFS) and pivots to the adjacent BFS with the objective improvement. It guarantees to reach the optimal BFS for nondegenerate and bounded LP with a nonempty feasible solution. While the interior point method starts at an interior point inside the feasible region and moves along an improved direction until it reaches the solution close to the optimal one.

The simplex method requires the feasibility of the current basic feasible solution. Since a general linear programming problem may not be feasible, it will need to be reformulated. Two classical artificial-variable techniques have been developed which are the two-phase simplex method [18] and the big-M method [17], [3]. Both techniques increase the dimension of the original problem by adding an artificial variable to each constraint causing more computational solution time.

In 2014, Boonperm, et al. [4] presented a non-acute constraint relaxation technique that eliminates the need for artificial variables and reduces the start-up time to solve the initial relaxation problem. The algorithm reinserts the non-acute constraints back into the relaxation problem to guarantee the optimal solution or infeasibility or unboundedness of a linear programming problem. Moreover, there is a use case of angle measurement of LP constraints to control a jump direction in a metaheuristic algorithm to solve an LP problem that is introduced by Visuthirattanam [5]. In this paper, angle measurement between constraints is also considered as a part of preprocessing input of a deep learning model.

The deep learning concept mimics the computation from the biology of human brain cells to learn a task directly from numerical source data. It consists of multiple layers of interconnected nodes and arcs with weights and biases. The learning process adjusts weights that work together to recognize accurately and classify objects from data. An alternative method for solving an LP problem has been studied for centuries. Recently, the use of machine learning models such as a deep learning model is investigated. Instead of solving an LP problem every time a new problem is posted. Researchers investigate whether the machine learning model can be used to identify the optimal solution by learning from thousands of solved LP problems. With the appropriate form of the LP problem and the special deep learning architecture, the optimal basis of any linear programming problem can be obtained.

A. Contributions

This paper uses a deep learning model to solve a linear programming problem using only objective coefficients, the right-hand-sided values and the constraint coefficients avoid any iterative procedure that has been used for centuries. The concept is to use million solved linear programming problems as the training data with the appropriate matrix format as the all-unit-row-except-first-unit-column matrix and an additional row-column convolutional layer, which enables the deep learning model to identify the optimal binding constraints from linear programming problems of varying sizes.

The objective of this paper is the LPNet deep learning architecture that can solve a linear programming problem. It has the all-unit-row-except-first-unit column matrix as the input fed to the row-column convolutional layer. Then the output is sent to state-of-the-art deep learning model submitting to two fully connected layers before the last output layer.

In order to achieve the objective, this paper introduces a unit-vector normalization that simplifies the training process compared to the traditional normalization method. This method incorporates two key concepts - reordering constraints and scaling LP problems - to enhance the model's ability to learn effectively.

Finally, the LPNet architecture integrates all these processes together as illustrated in Fig. 1, and the trained model outperforms the benchmark LP datasets when compared to CLPEX [31], the state-of-the-art optimization LP solver.

The remainder of this paper is organized as follows. Section II provides a literature review. Section III covers the methodology of this paper. Section IV provides the analysis and results of the effective deep learning architecture. Section V is the conclusion of this paper, and the last section provides a discussion of this work.

II. Literature Review

Various deep-learning architectures have been proposed to learn different tasks via spatial relationships within the input data. A deep convolution neural network (DCNN) is one of the deep learning architectures which is suitable for computer vision tasks. Many state of the art architectures demonstrate high performance for an image classification task such as Resnet [20], MobileNet [6], EfficientNet [21], XceptionNet [22] and InceptionNet [23].

The study of Khade S. et al. [10], [12], [13] developed multiple DCNNs to identify iris liveness detection based on RestNet50 and EfficientNet for binary classifications. Moreover, the convolutional deep extreme learning machine method [14] can well recognize the pattern of a diabetic retinopathy image using DCNN architectures, that is ResNet, DenseNet [15], and GoogleNet [16]. This paper also developed DCNN with these architectures for multi-label classifications. Multi-label classifications [11] are more complex than binary classifications.

CNN with recurrent neural network (CNN-RNN) technique is utilized to detect and classify sarcasm [28]. In order to boost the detection outcomes of the CNN+RNN technique, a hyperparameter tuning process utilizing a teaching and learning-based optimization (TLBO) algorithm is employed in such a way that the classification performance gets increased.

The integration of a DCNN model to a multiparametric programming problem [24], [7] was demonstrated by Justin et al. [8]. Solving the parametric 0–1 LP problem by a DCNN model was proposed in 2022 [9]. The DCNN model for 0-1 LP learns from a histogram-like image representation.

Effati et al. [27] proposed two recurrent neural network models for solving linear and quadratic programming problems. The first model is derived from an unconstraint minimization reformulation of the program, while the second model is directly obtained from the optimality condition for an optimization problem. The paper compares the convergence of these models using the energy function and the duality gap. The paper also explores the existence and convergence of the trajectory and stability properties for the neural network models.

The use of a neural network as a solution bundle for solving ordinary differential equations (ODEs) for various initial states and system parameters is presented by Flamant et al. [26]. In 2023, Dawen Wu et al. [25] proposed a deep learning approach in the form of feedforward neural networks to solve linear programming (LP) problems. The approach models the LP problem by an ordinary differential equations (ODE) system, the state solution of which globally converges to the optimal solution of the LP problem. A neural network model is constructed as an approximate state solution to the ODE system, such that the neural network model contains the prediction of the LP problem.

III. Methodology

This paper proposed the deep convolutional neural network model, called LPNet, to learn from linear programming coefficients directly putting in the form of a general all-unit-row-except-first-unit-column matrix. This matrix contains the objective coefficients in the first row, normalized to one, the right-hand-side vector as the first column, and the rest are coefficients from constraints normalized to one. It then passes to the special row-column convolutional layer that is carefully designed with convolution filters for extracting the row-column components of the LP problems. After this layer, the result will pass to the state-of-the-art architectures. Then it will pass to two fully connected layers before connecting to the output layer. Each constraint will be marked as either it is one of the optimal basis or it is not in the output layer, see Fig. 1.

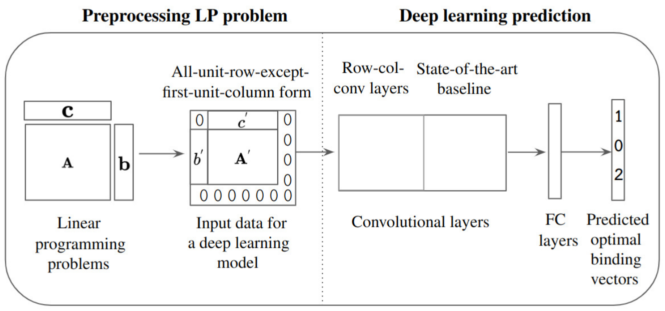

Preprocessing LP problem
Deep learning prediction
All-unit-row-except-first-unit-column form
Row-col-conv layers
State-of-the-art baseline
c
A
b
0
c'
b'
A'
0
0
0
0
0 0 0 0 0 0 0
Linear programming problems
Input data for a deep learning model
Convolutional layers
FC layers
Predicted optimal binding vectors
1
0
2

Fig. 1. Overview of an optimal binding prediction.

A. A Linear Programming Model

A linear programming model is an optimization model with a linear real-valued objective function ($\mathbf{c}^{\mathrm{T}}\tilde{\mathbf{x}}$) and linear constraints ($\mathbf{A}\tilde{\mathbf{x}} \geq \mathbf{b}, \tilde{\mathbf{x}} \geq \mathbf{0}$). The optimal solution of this LP model is the feasible point that gives the smallest objective value for the minimization or the largest objective value for the maximization. It can be expressed in the following mathematical form.

$$\begin{array}{ll} \text{Min} & \mathbf{c}^{\mathrm{T}}\tilde{\mathbf{x}} \\ \text{s.t.} & \mathbf{A}\tilde{\mathbf{x}} \geq \mathbf{b} \\ & \tilde{\mathbf{x}} \geq \mathbf{0} \end{array} \tag{1}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$ and $\tilde{\mathbf{x}} \in \mathbb{R}^n$. The optimal solution of the LP model can be solved using various iterative algorithms such as the two-phase simplex algorithm, the big-M algorithm, and the dual-simplex algorithm. Alternatively, the optimal solution could be obtained if all optimal binding constraints are identified correctly. So learning to solve a linear programming problem can be cast as learning to identify the optimal binding constraint from a million linear programming problems.

For every LP problem, there is a corresponding dual LP problem. The original LP problem is called the primal LP problem. If one of these LP problems obtains an optimal solution then both problems will possess the optimal solution having the same optimal value and satisfying the complementary slackness. The non-zero value of any basic variable will cause the corresponding dual constraint to be binding. From equation (1), the dual LP problem is

$$\begin{array}{ll} \text{Max} & \mathbf{b}^{\mathrm{T}}\mathbf{y} \\ \text{s.t.} & \mathbf{A}^{\mathrm{T}}\mathbf{y} \leq \mathbf{c} \\ & \mathbf{y} \geq \mathbf{0} \end{array} \tag{2}$$

Note that the coefficients of the decision variable from a primal LP problem correspond to the row constraint for the dual LP problem. From the complementary slackness properties, [17] of the optimal conditions, the non-zero value of the dual basic variable will give rise to the corresponding primal constraint to be binding, i.e., if $y_i > 0$, for some $i \in \{1, ..., m\}$ then the corresponding constraint in the LP primal problem will be binding. Similarly, the non-zero basic variable of the LP primal problem will give rise to the corresponding binding dual constraint. If $x_j > 0$, for some $j \in \{1, ..., n\}$ then the corresponding constraint in the dual constraints will be binding. These optimal bindings can be predicted by LPNet which will be explained in detail in Section IV. The next section will cover the input design for this deep learning model from the coefficients of the primal LP problem.

B. Basic Feasible Solution

The LP problem from (1) can be converted to a standard form by subtracting non-negative surplus decision variables $\mathbf{x}'$ as

$$\begin{array}{ll} \text{Min} & \mathbf{c}^{\mathrm{T}}\tilde{\mathbf{x}} + \mathbf{0}^{\mathrm{T}}\mathbf{x}' \\ \text{s.t.} & \mathbf{A}\tilde{\mathbf{x}} - \mathbf{I}\mathbf{x}' = \mathbf{b} \\ & \tilde{\mathbf{x}}, \mathbf{x}' \geq \mathbf{0} \end{array} \tag{3}$$

where $\mathbf{I}$ is a $m \times m$ identity matrix, and $\mathbf{x}'$ is $\{x_{n+1}, ..., x_{m+n}\}$. So a new variable vector of the LP problem (3) is $\mathbf{x} = [x_1, x_2, ..., x_{m+n}]^T$.

Definition 1: [19] For any nonsingular $m \times m$ submatrix $\mathbf{A}_B$ of $\mathbf{A}$ where $B \subset \{1, 2, ..., m + n\}$ is the set of basic indices, $\mathbf{x} = [\mathbf{x}_B, \mathbf{0}]^T$ is called a basic solution with respect to the basis ($\mathbf{A}_B$), where $\mathbf{0}$ in $\mathbf{x}$ is the zero vector of all leftover components of $\mathbf{x}$ associated with the $n-m$ non-basic variables of $\mathbf{A}$. From the constraint of the standard LP problem $\mathbf{Ax} = \mathbf{b}$, it can be rewritten as $\mathbf{A}_B\mathbf{x}_B = \mathbf{b}$. $\mathbf{x}_B$ is called a vector of basic variables or basic variables in short.

Definition 2: [19] A vector $\mathbf{x}$ satisfying the system $\mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ is said to be a feasible solution for the system. A feasible solution with a known basis is called a basic feasible solution.

Theorem 3.1: [19] Given a standard LP problem (3) where $\mathbf{A}$ is an $m \times n$ matrix of rank m,
1. if the feasible region of the LP problem is nonempty and bounded, then there is a basic feasible solution.
2. if there is an optimal feasible solution, there is an optimal basic feasible solution.

This paper proposes a way to predict the optimal basic feasible solution from binding constraints of the LP problem using the target vector $\mathbf{Y}$ corresponding to each constraint. If the element of $\mathbf{Y}$, Y_i is zero, then the constraint i will be binding and if the element of $\mathbf{Y}$, Y_j is one, then the constraint j is not binding and if the element of $\mathbf{Y}$, Y_k is two, then the constraint k does not exist from the original problem. It is just the padding constraint during the learning step of this deep learning architecture.

The next subsection will cover a theorem of scaling an LP problem which guarantees to have the same optimal basic $\mathbf{A}_B$. This theorem is designed to support an input form of LPNet in Section III.

C. Scaling LP Problems

Some LP problems may come from the same one with different multipliers to their constraints. Normalization of rows of LP problems helps identify duplicate samples during the training phase of LPNet. The following theorem shows that the scaled LP2 problem still has the same basic feasible solution as the LP1 problem. This implies that only one LP problem from all scaled LP problems is enough to be included in the training phase. This will help LPNet focus on one version of LP problems.

Theorem 3.2: Given LP1 has the optimal solution. For any positive scale α, a set of basic variables of LP1 will be the same as the basic variables of LP2(α). Moreover, the optimal basic indices in LP1 will be the same as the optimal basic indices in LP2(α)

$$\begin{array}{lll} \mathbf{LP1} & \text{Min} & \mathbf{c}^T\mathbf{x} \\ & \text{s.t.} & \mathbf{Ax} \geq \mathbf{b} \\ & & \mathbf{x} \geq \mathbf{0} \end{array} \qquad \begin{array}{lll} \mathbf{LP2}(\alpha) & \text{Min} & \mathbf{c}^T\mathbf{x} \\ & \text{s.t.} & \mathbf{Ax} \geq \frac{\mathbf{b}}{\alpha} \\ & & \mathbf{x} \geq \mathbf{0} \end{array}$$

Proof: Assume LP1 has the optimal solution. Assume that the current basis is A_B with the set of basic indices B and let the set of the non-basic indices be N. It is easy to see that the optimal objective value

$z^* = \mathbf{c}^T\mathbf{x}^* = \mathbf{c}_B^T\mathbf{x}_B^* + \mathbf{c}_N^T\mathbf{x}_N^*$ and $\mathbf{A}_B\mathbf{x}_\mathbf{B}^* + \mathbf{A}_N\mathbf{x}_N^* = \mathbf{b}$. Therefore $\mathbf{x}_B^* = \mathbf{A}_B^{-1}\mathbf{b} - \mathbf{A}_B^{-1}\mathbf{A}_N\mathbf{x}_N^*$ and

$$\begin{aligned} z^* &= \mathbf{c}_B^T(\mathbf{A}_B^{-1}\mathbf{b} - \mathbf{A}_B^{-1}\mathbf{A}_N\mathbf{x}_N^*) + \mathbf{c}_N^T\mathbf{x}_N^* \\ &= \mathbf{c}_B^T\mathbf{A}_B^{-1}\mathbf{b} + (\mathbf{c}_N^T - \mathbf{c}_B^T\mathbf{A}_B^{-1}\mathbf{A}_N)\mathbf{x}_N^*. \end{aligned}$$

Since z^* is the optimal objective value then

$$\mathbf{c}_N^T - \mathbf{c}_B^T\mathbf{A}_B^{-1}\mathbf{A}_N \geq \mathbf{0} \quad \text{and} \quad \mathbf{x}_N^* = \mathbf{0}.$$

Let $\alpha \in \mathbb{R}^+$ be such that

$$\begin{aligned}\mathbf{x}_B^{'} &= \frac{\mathbf{x}_B^*}{\alpha}\\ &= \mathbf{A}_B^{-1}\frac{\mathbf{b}}{\alpha} - \mathbf{A}_B^{-1}\mathbf{A}_N\frac{\mathbf{x}_N^*}{\alpha}\\ &= \mathbf{A}_B^{-1}\frac{\mathbf{b}}{\alpha} - \mathbf{A}_B^{-1}\mathbf{A}_N\mathbf{x}_N^{'}\\ &\geq 0\end{aligned}$$

and

$$\begin{aligned}z^{'} &= \frac{z^*}{\alpha}\\ &= \mathbf{c}_B^T\mathbf{A}_B^{-1}\frac{\mathbf{b}}{\alpha} + (\mathbf{c}_N^T - \mathbf{c}_B^T\mathbf{A}_B^{-1}\mathbf{A}_N)\frac{\mathbf{x}_N^*}{\alpha}\\ &= \mathbf{c}_B^T\mathbf{A}_B^{-1}\frac{\mathbf{b}}{\alpha} + (\mathbf{c}_N^T - \mathbf{c}_B^T\mathbf{A}_B^{-1}\mathbf{A}_N)\mathbf{x}_N^{'}.\end{aligned}$$

That is $\mathbf{x}_N^{'} = \frac{\mathbf{x}_N^*}{\alpha} = 0, \mathbf{x}_B^{'} \geq \mathbf{0}$ and $\mathbf{c}_N^T - \mathbf{c}_B^T\mathbf{A}_B^{-1}\mathbf{A}_N \geq \mathbf{0}$. Therefore, the set of indices corresponding to $\mathbf{x}^{'}$ is the current basic feasible solution of LP2 with the same basis B. Similarly, if LP2 has an optimal basic feasible solution then LP1 has the same basic feasible solution.

In addition, the optimal basic indices in LP1 will correspond to the optimal basic indices in LP2(α). ■

D. All-Unit-Row-Except-First-Unit-Column Matrix

In the context of LPNet training, the normalization of inputs plays a crucial role in facilitating deep learning optimization to discover optimal parameters for learning optimal binding constraints. The process of normalization specifically addresses the numerical scaling of the rows within the constraint matrix of LP problems. To address this, the present research introduces a two-step data preprocessing approach for any LP problem prior to its utilization in the learning process.

The first step involves the reordering of all constraints based on the angles formed between the sum vector ($\mathbf{1}$) and the coefficient of each constraint equation. This reordering aims to enhance the structure of the input data for improved learning outcomes.

In the subsequent step, each constraint is individually rescaled to attain a unit norm. This normalization further aids in aligning the constraints on a consistent scale, enabling more effective learning and optimization processes.

By implementing these two preprocessing steps, this research enhances the overall effectiveness of LPNet training by ensuring that the input data is appropriately structured and scaled to facilitate the discovery of optimal binding constraints.

E. LP Constraint Ordering

The research findings demonstrate that the optimal binding constraints remain unchanged even when each row is interchanged with another in LP problems. In order to establish a suitable input format for deep learning, it becomes necessary to rearrange all constraints based on the angle between the gradient vector of primal LP constraints and the sum vector ($\mathbf{1}$), which represents a vector of ones with the appropriate size.

By rearranging the constraints in this manner, the input data is properly structured to align with the requirements of deep learning algorithms. This ensures that the crucial information captured by the gradient vector and the sum vector is effectively utilized, leading to more accurate and meaningful training results. Consequently, this approach enhances the overall effectiveness of the deep learning process in tackling LP problems and discovering optimal binding constraints.

Let $\mathbf{a}_{i:}$ be a gradient vector of the i^{th} constraint from $\mathbf{A}$. The angle between $\mathbf{a}_{i:}$ and $\mathbf{1}$ of size n is defined as

$$\theta_i^{row} = \arccos(\frac{\mathbf{a}_{i:}^\top\mathbf{1}}{\|\mathbf{a}_{i:}\|\|\mathbf{1}\|}). \quad (4)$$

The primal constraints will be ordered from the smallest angle to the largest angle.

Similarly, the dual constraints corresponding to the column of $\mathbf{A}$ will be rearranged by sorting the angle between the $\mathbf{a}_{:j}$ and $\mathbf{1}$ of size m,

$$\theta_j^{column} = \arccos(\frac{\mathbf{a}_{:j}^\top\mathbf{1}}{\|\mathbf{a}_{:j}\|\|\mathbf{1}\|}). \quad (5)$$

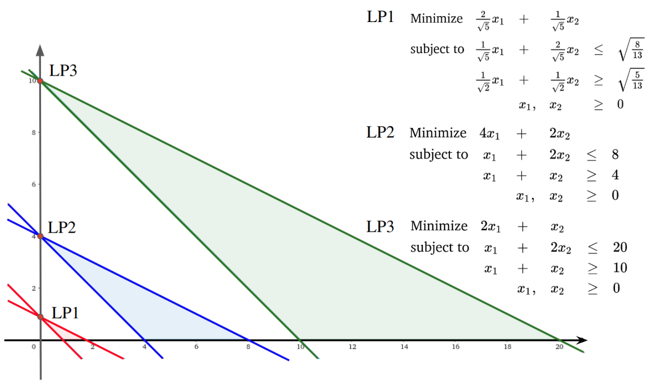
The next step is to rescale all constraints of the LP problem.

F. LP Scaling

The elimination of duplicate LP problems holds significant importance in the training of LPNet. In Figure 2, three distinct LP problems, namely LP1, LP2, and LP3, are depicted, all of which share the same basic feasible solution while differing only in scaling. In this study, LP1 serves as the normalized version for LP2 and LP3. By identifying the optimal binding constraints of LP1, it becomes possible to determine the optimal binding constraints for LP2 and LP3, as demonstrated in Theorem 2.

This approach of addressing duplicate LP problems streamlines the training process of LPNet. It leverages the knowledge gained from the normalized version (LP1) to efficiently identify the optimal binding constraints for the related LP problems (LP2 and LP3). By avoiding redundant computations, this methodology enhances the overall training efficiency and contributes to the improved performance of LPNet.

From Equation (1), coefficient vector $\mathbf{c}$, right-hand side vectors $\mathbf{b}$ and matrix $\mathbf{A}$ of the LP problem will be normalized into the unit vector format corresponding to $\mathbf{c}^{'}, \mathbf{b}^{'}$ and $\mathbf{A}^{'}$ respectively. The coefficients of the objective function will be converted to $c_j^{'} = \frac{c_j}{||\mathbf{c}||}$ for all $j = 1, 2, ..., n$, the coefficients of the constraint function $a_{ij}^{'} = \frac{a_{ij}}{||\mathbf{a_i}||}$ for $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$ and $b_i^{'} = \frac{b_i}{||\mathbf{a_i}||}$ and $b_i^{''} = \frac{b_i^{'}}{||\mathbf{b}^{'}||}$ for $i = 1, 2, ..., m$. These normalized vectors will be stacked together to form the input matrix for the training and testing phases of LPNet. The label vector $\mathbf{Y}$ will contain either 0 for the corresponding binding constraint, 1 for the corresponding non-binding constraint, and 2 for the padding constraint.



$$\text{LP1}\quad \begin{array}{llcll} \text{Minimize} & \frac{2}{\sqrt{5}}x_1 & + & \frac{1}{\sqrt{5}}x_2 & \\ \text{subject to} & \frac{1}{\sqrt{5}}x_1 & + & \frac{2}{\sqrt{5}}x_2 & \leq \sqrt{\frac{8}{13}} \\ & \frac{1}{\sqrt{2}}x_1 & + & \frac{1}{\sqrt{2}}x_2 & \geq \sqrt{\frac{5}{13}} \\ & & x_1, & x_2 & \geq 0 \end{array}$$

$$\text{LP2}\quad \begin{array}{llcll} \text{Minimize} & 4x_1 & + & 2x_2 & \\ \text{subject to} & x_1 & + & 2x_2 & \leq 8 \\ & x_1 & + & x_2 & \geq 4 \\ & & x_1, & x_2 & \geq 0 \end{array}$$

$$\text{LP3}\quad \begin{array}{llcll} \text{Minimize} & 2x_1 & + & x_2 & \\ \text{subject to} & x_1 & + & 2x_2 & \leq 20 \\ & x_1 & + & x_2 & \geq 10 \\ & & x_1, & x_2 & \geq 0 \end{array}$$

Fig. 2. An example of scaling LP problem of $\mathbf{x} \in \mathbb{R}^2$. Three corresponding LP problems, LP1 is the unit-vector normalization of both LP2 and LP3. These three LP problems are not the same problems but they have the same optimal basic variables. If the optimal binding constraints of LP 1 exist then other LP scales are immediately found. Then any LP sizes $\mathbf{x} \in \mathbb{R}^n$ will be scaled by the unit-vector normalization before entering to LPNet model.

$$\mathbf{X} = \begin{bmatrix} 0 & c_1^{'} & c_2^{'} & \cdots & c_n^{'} \\ b_1^{''} & a_{11}^{'} & a_{12}^{'} & \cdots & a_{1n}^{'} \\ b_2^{''} & a_{21}^{'} & a_{22}^{'} & \cdots & a_{2n}^{'} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ b_m^{''} & a_{m1}^{'} & a_{m2}^{'} & \cdots & a_{mn}^{'} \end{bmatrix} = \begin{bmatrix} 0 & \frac{c_1}{||\mathbf{c}||} & \frac{c_2}{||\mathbf{c}||} & \cdots & \frac{c_n}{||\mathbf{c}||} \\ \frac{b_1}{||\mathbf{a}_1||||\mathbf{b}^{'}||} & \frac{a_{11}}{||\mathbf{a}_1||} & \frac{a_{12}}{||\mathbf{a}_1||} & \cdots & \frac{a_{1n}}{||\mathbf{a}_1||} \\ \frac{b_2}{||\mathbf{a}_2||||\mathbf{b}^{'}||} & \frac{a_{21}}{||\mathbf{a}_2||} & \frac{a_{22}}{||\mathbf{a}_2||} & \cdots & \frac{a_{2n}}{||\mathbf{a}_2||} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{b_m}{||\mathbf{a}_m||||\mathbf{b}^{'}||} & \frac{a_{m1}}{||\mathbf{a}_m||} & \frac{a_{m2}}{||\mathbf{a}_m||} & \cdots & \frac{a_{mn}}{||\mathbf{a}_m||} \end{bmatrix} \tag{6}$$

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \\ y_{m+1} \\ \vdots \\ y_{m+n} \end{bmatrix} \tag{7}$$

To train LPNet with different LP problem sizes without recreating the specific input size of the deep learning model these inputs need to be embedded into the maximum matrix format. Assume that the maximum LP size is N constraints and N variables so $\mathbf{Y}$ has $2N$ components. To set up a (m, n) input LP sample into the matrix of the maximum LP size (N, N) where $m \leq N$ and $n \leq N$, the small-sized (m, n) LP input matrix are padded with zero rows from the $m + 1$ row to the N row and zero columns from the $n + 1$ column to the N column. The corresponding label vector is assigned to 2, see the matrix below for the padding concept. An example of a (2,2) LP sample is added into (N, N) matrix $\mathbf{X}$ with padding 2 in the $\mathbf{Y}$.

$$\mathbf{X} = \begin{bmatrix} 0 & c_1^{'} & c_2^{'} & 0 & \ldots & 0 \\ b_1^{''} & a_{11}^{'} & a_{12}^{'} & 0 & \ldots & 0 \\ b_2^{''} & a_{21}^{'} & a_{22}^{'} & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix} = \begin{bmatrix} 0 & \frac{c_1}{||\mathbf{c}||} & \frac{c_2}{||\mathbf{c}||} & 0 & \ldots & 0 \\ \frac{b_1}{||a_1||||\mathbf{b}^{'}||} & \frac{a_{11}}{||\mathbf{a}_1||} & \frac{a_{12}}{||\mathbf{a}_1||} & 0 & \ldots & 0 \\ \frac{b_2}{||a_2||||\mathbf{b}^{'}||} & \frac{a_{21}}{||\mathbf{a}_2||} & \frac{a_{22}}{||\mathbf{a}_2||} & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix}, \tag{8}$$

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ 2 \\ \vdots \\ 2 \\ y_{m+1} \\ y_{m+2} \\ 2 \\ \vdots \\ 2 \end{bmatrix} \tag{9}$$

The y_1 and y_2 in $\mathbf{Y}$ are represented by the binding constraint status (0 or 1) of the constraints from non-negative variables $x_1, x_2 \geq 0$ respectively. The $N + 1$ and $N + 2$ components of $\mathbf{Y}$ are $y_{N+1} = y_{m+1}$ and $y_{N+2} = y_{m+2}$ denoted by binding constraint status of the first and second constraints. By this concept, the LP model can be trained by many LP problem sizes simultaneously without recreating the CNN model for a specific LP size.

All LP problems will be converted to this all-unit-row-except-first-unit-column matrix as LPNet samples for the training and testing phases.

G. LPNet Architecture

The deep Learning model is an automated learning model that mimics the functioning of human neural networks in machine learning. It uses several layers of neurons arranged in sequential order. The target concept from training data will be learned by the weights and biases of all neurons.

Convolutional neural networks (CNNs) are specialized neural network structures capable of classifying image data much better than conventional neural networks. The main idea of CNN is to use a special type of layer called a convolutional layer that extracts parts of an image such as the borders of objects which is the spatial relation and sends to a pooling layer to extract only the information components from the multi-dimensional array. The proposed LPNet architecture needs a special convolutional layer to extract basic elements from the all-unit-row-except-first unit-column matrix of the LP problem in order to make the state-of-the-art CNN model (baseline model) efficiently be trained. The details of LPNet baseline model are shown in Table I which consists of three stages.

The first stage includes RCConv layers that extract row and column features and transform them into spatial relations. To capture the spatial information of an LP problem, the special row-column convolutional layer is introduced, as explained in the following subsection. The second stage of the model is the baseline model, which incorporates a state-of-the-art CNN baseline model. Our results show that the best-performing baseline model for LPNet is MobileNet. The final stage of the model is the estimator, which includes fully connected layers. The last layers consist of 600 nodes, representing the optimal binding constraints status of LP problems with up to 300 constraints and 300 variables. Overall, the LPNet model is designed to effectively extract and utilize features from LP problems to identify optimal binding constraints and provide high-quality solutions.

TABLE I. LPNET WITH MOBILENET BASELINE

Stage	Type/Stride	Filter Shape	Input Size
RCConv	Row Conv /s1	1 × 301 × 32	301 × 301 × 32
	Concat 301 cols		301 × 1 × 32
	Row Conv /s1	301 × 1 × 32	301 × 301 × 32
	Concat 301 cols		301 × 1 × 32
	Concat depth		301 × 301 × 32
			301 × 301 × 32
Baseline	Conv /s2	3 × 3 × 3 ×32	301 × 301 × 64
	Conv dw /s1	3 × 3 × 32 dw	151 × 151 × 32
	Conv /s1	1 × 1 × 32 ×64	151 × 151 × 32
	Conv dw /s2	3 × 3 × 64 dw	151 × 151 × 64
	Conv /s1	1 × 1 × 64 ×128	75 × 75 × 64
	Conv dw /s1	3 × 3 × 3 ×128 dw	75 × 75 × 128
	Conv /s1	1 × 1 × 128 ×128	75 × 75 × 128
	Conv dw /s2	3 × 3 × 128 dw	75 × 75 × 128
	Conv /s1	1 × 1 × 128 × 256	37 × 37 × 128
	Conv dw /s1	3 × 3 × 256 dw	37 × 37 × 256
	Conv /s1	1 × 1 × 256 × 256	37 × 37 × 256
	Conv dw /s2	3 × 3 × 256 dw	37 × 37 × 256
	Conv /s1	1 × 1 × 256 × 512	18 × 18 × 256
	5 × Conv dw / s1	3 × 3 × 3 ×32	3 × 3 × 3 ×32
	5 × Conv / s1	3 × 3 × 3 ×32	3 × 3 × 3 ×32
	Conv dw /s2	3 × 3 × 1024 dw	18 × 18 × 512
	Conv /s1	1 × 1 × 512 × 1024	9 × 9 × 512
	Conv dw /s2	3 × 3 ×1024 dw	9 × 9 × 1024
	Conv /s1	1 × 1 ×1024×1024	9 × 9 × 1024
	Avg Pool /s1	Pool 7 × 7	9 × 9 × 1024
Estimator	FC1	1024 × 512	1024
	FC2	512 × 512	512
	Output	512 × 600	512

H. Row-Column Convolutional Neural Network

In this research, CNN is used to extract features from the relationship between primal constraints and dual constraints using non-square kernels. A normalized input sample is convoluted over the column and the row to obtain row feature maps and column feature maps, respectively, see Fig. 3. Notice that, these row and column convolutional kernels will transform the row-column (primal-dual) information to spatial feature maps before passing to a state-of-the-art CNN architecture.

Primal constraints (rows of $\mathbf{X}$) will be convoluted by $(1, n)$ kernel size with k kernels for generating $(m, 1, k)$ column feature maps. Similarly, the dual constraints (columns of $\mathbf{X}$) will be convoluted by $(m, 1)$ row convolution with k kernels in order to create $(1, n, k)$ row feature maps. These feature maps will be duplicated m times for row feature maps and n times for columns feature maps. The duplicated row and column

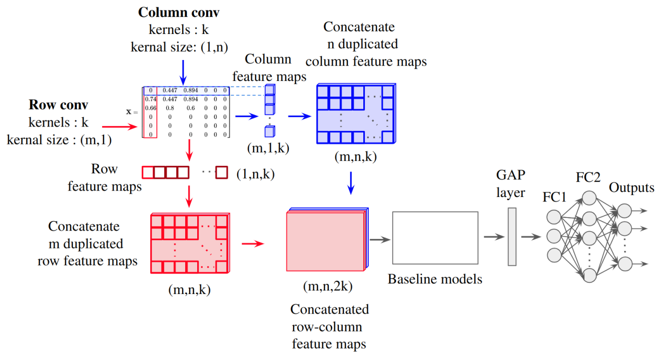


Fig. 3. The LPNet architecture.

feature maps will be concatenated to generate two $m \times n \times k$ spatial feature maps, called RCConv. Finally, these features will be concatenated by depth to be an $(m, n, 2k)$ input of a baseline convolution architecture.

I. Baseline Convolution Architectures

CNN combines a convolution layer with other types of layers, such as a pooling layer, and then stacks such layer groups on top of each other. Some hyperparameters, such as the size of the kernel, the number of strides, and the number of padding are combined to become the architecture of CNN. There are many state-of-the-art CNN architectures such as ResNet50, MobileNetv1, EfficentNetB1, Xception, and Inceptionv1 which are famous deep models that are successfully learned to classify images. Resnet50 was the first to introduce a residual block concept that makes the model learn some residual features from the earlier layers. Later, the concept of residual blocks is improved using a depthwise convolution in order to reduce model parameters which are called mobilenet blocks. The depthwise separable convolution in the mobilenet blocks makes MobileNetv1 lighter than other baseline models. The mobilenet block concept is continually improved by searching for the best size of the height, width, and depth of convolutional filters from another deep learning model, and then it becomes EfficeintNetB0-B7. This paper focuses on EfficientNetB1 which is suitable for LP samples. The idea of using multiple filters of different sizes on the same level is applied in the inception model. Then the inception model instead of having deep layers also has parallel layers thus making the model wider rather than making it deeper. A baseline model will be an engine for extracting features of LPNet after the LP sample is passed into the row-column convolutional layer. The details of the LPNet architecture with the mobilenet baseline convolution model are shown in the next section.

J. Global Average Pooling Layer and Fully Connected Layers

The feature maps from the baseline convolutional layers will be transformed into the spatial dimensions of feature maps by averaging the spatial features into vector features using a global average pooling layer. These vector features are fed to the first fully connected hidden layer. The first two hidden layers consist of 512 hidden nodes with a relu activation function following the batch normalization layer. Since this paper aims to predict the maximum optimal binding constraints which are 300 constraints and 300 non-negative constraints of decision variables. Then the total number of constraints is about 600. So the last output layer contains 600 nodes with linear activation function in order to estimate the target vector in $\{0, 1, 2\}$, which is representing the optimal binding constraints. The next section will show the experimental results of LPNet models.

It is straightforward to train LPNet in supervised learning. Training LP samples will be transformed to all-unit-row-except-first-unit-column matrices ($\mathbf{X}$) from Eq. (8) and the optimal binding vectors ($\mathbf{Y}$) from Eq.(9). The transformed matrices allow the LPNet model to be trained in many LP problem sizes at the same time without reconstructing the individual input sizes. After the trained LPNet is completed then the next subsection shows the inference algorithm of the LPNet model.

K. LPNet Inference

The inference algorithm of LPNet is summarized as follows:

1) Compute angles between the gradient vectors of the primal constraints from Eq. (4).
2) Rearrange the order of primal constraints ($\mathbf{a}_{i:}$ and $\mathbf{b}_i$) by descending angles from 1).
3) Compute angles between the gradient vectors of the dual constraints from 2) using Eq. (5).
4) Rearrange the order of coefficient variables ($\mathbf{a}_{:j}$ and $\mathbf{c}_j$) from 2) by descending angles from 3).
5) Scale down the LP problem (c_j, b_i, a_{ij}) from 4) into $(c_j^{'}, b_i^{''}, a_{ij}^{'})$ for $i \in \{1, 2, .., m\}$ and $j \in \{1, 2, .., n\}$.
6) If the LP problem size from 5) is the maximum size $(m, n) = (N, N)$ then the input matrix for the LPNet model will be defined by Eq. (6) and then go to 8). Otherwise, go to 7).
7) The LP problem will be padded zeros by using Eq. (8) and then go to 8).
8) Take the input matrix from 6) or 7) into the trained LPNet model and the result will be a predicted optimal binding vector $\mathbf{Y}$.

The experimental results of the row-column convolution layer plus given a baseline model followed by two fully connected layers and the output layer are presented in this section. About one million LP problems are randomly generated and solved for the training dataset and 500,000 LP problems for the testing dataset. Any small size LP problem is padded to have a matrix size of 300x300. All LP problems guarantee to have the optimal solution. The experiments are performed on Intel(R) Xeon(R) CPU @ 2.20GHz 25GB RAM GPU Tesla P100-PCIE on Ubuntu 18.04.5 LTS. It is implemented by Python programming language based on Tensorflow 2.8.0. [29].

IV. Analysis and Results

A. Performance Measurement

Most deep learning models use the mean squared logarithmic error (MSLE) as the evaluation measure. It is often used in regression tasks for predicting a continuous value. The MSLE loss function is defined as the mean of the squared logarithmic errors between the predicted values and the true values:

$$L(\mathbf{Y}, \hat{\mathbf{Y}}) = \tfrac{1}{m+n} \textstyle\sum_{i=1}^{m+n} (\log(y_i + 1) - \log(\hat{y}_i + 1))^2,$$

where $\hat{y}_i$ is the predicted value, y_i is the true value, and $n+m$ is the number of samples. The logarithmic transformation helps reduce the impact of large errors, which can be useful when working with skewed data or when there are a few extremely large errors that could dominate the loss. Notice that vector $\mathbf{Y}$ contains a large number of 2's, an imbalance problem occurs during model training, and the error value for 2 is larger than the error from 0 and 1, so the trained model will predict 2 more accurately than 0 or 1. This situation is not reasonable for training the deep learning model for predicting binding and non-binding constraints. Therefore, it is necessary to change the scale of the data to a log scale.

Accuracy is a common metric used to evaluate the performance of a machine-learning model which is defined as the percentage of correct predictions made by the model on a dataset. It is often used to evaluate classification models, where the goal is to predict a class label for a given input. In this case, the model's predictions are compared to the true class labels for the inputs, and the percentage of correct predictions is calculated. This paper proposes two types of accuracy that are 0-1 accuracy and 0-1-2 accuracy. The 0-1-2 accuracy is the accuracy of all optimal binding statuses of $\mathbf{Y}$. $Y_i = 0$ represents an optimal binding constraint i that is $a_{ij}^T x^* = b_i$. $Y_i = 1$ indicates an optimal nonbinding constraint. $Y_i = 2$ shows an auxiliary constraint for padding zeros rows to an (m,n) LP problem as shown in Eq. (8).

$$Acc_{0-1-2}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{\sum_{i=1}^{N} \beta_i}{N}$$

where

$$\beta_i = \begin{cases} 1 & \text{if } Y_i = \hat{Y}_i, \\ 0 & \text{if } Y_i \neq \hat{Y}_i. \end{cases}$$

The 0-1 accuracy is the accuracy without padding status $Y_i = 2$ of a (m, n) LP problem.

$$Acc_{0-1}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{\sum_{i=1}^{n} \beta_i + \sum_{i=N+1}^{N+m} \beta_i}{m+n}$$

The following subsection will show the result of using the different normalizations with LPNet.

B. Normalization Experiment

All normalizations are applied to the row-column-conv MobileNet baseline model with the row-column arrangement of input samples. From Fig. 4, after 25000 iterations the unit-vector normalization can reduce loss values faster than other normalization methods.

As for the error value of the testing dataset shown in Fig. 5, both the min-max method and the standardization method fluctuated significantly more than the unit-vector method because the normalized input samples did not reduce the variation of the input LP samples. For the unit-vector method which

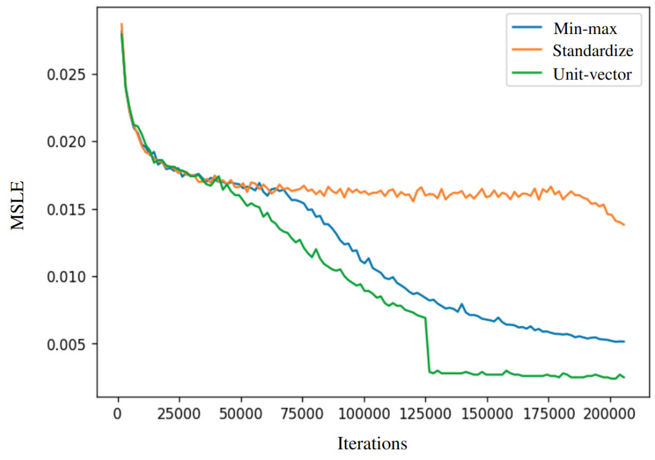


Fig. 4. Error of normalization methods of training dataset.

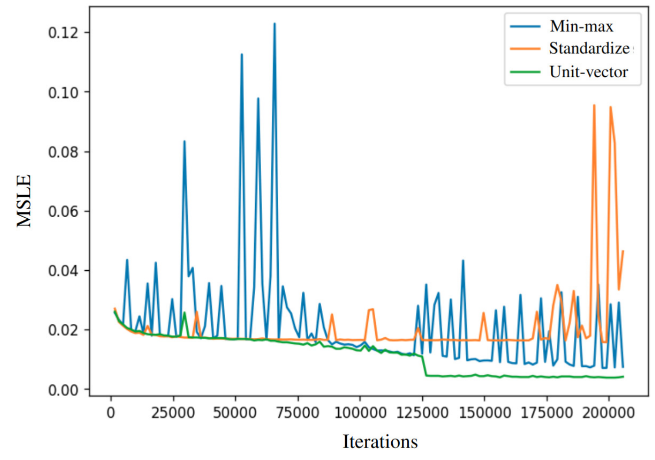


Fig. 5. Error of normalization methods of the testing dataset.

adjusts the problem to have the same scale, the loss value is more stable. Fig. 5 shows the lowest loss value of these normalization methods.

TABLE II. Normalization Methods

Normalization	MSLE	0-1-2 acc	0-1 acc
Standardize	0.0078	0.983	0.957
Min-max	0.0068	0.987	0.967
Unit-vector	**0.0034**	**0.996**	**0.990**

C. Row-column Order Experiment

The lowest loss values of the three normalization methods are shown in Table II. The unit-vector normalization achieves the lowest loss value and gains the highest accuracy both the 0-1-2 accuracy and the 0-1 accuracy.

Table III shows MSLE of a different order of the all-unit-row-except-first-unit-column matrix. The original order column represents the order of rows and columns according to the given LP problem. The rearranged row order represents the order of all primal LP constraints according to the angle between the row-coefficient vector and the sum vector. The rearranged row-column order represents the order of all primal LP constraints and dual LP constraints. Observe that the arrangement of rows and columns of the input samples shows superior performance over the original order and rearrange of row order only.

TABLE III. The Average MSLEs from the Row-column Convolution Plus Three State-of-the-art CNNs

Models	MSLE		
	Original order	Rearrange row order	Rearrange row-column order
RCConv ResNet50	0.0168	0.0213	**0.0055**
RCConv MobileNetV1	0.0169	0.0061	**0.0034**
RCConv EfficientNetB1	0.0164	0.0164	**0.0151**

D. Baseline CNN Architecture Experiment

The validation dataset loss values of LPNet with different baseline models are displayed in Fig. 6. Our results indicate that the MobileNet model exhibits the most stable convergence compared to other baseline models. This suggests that the MobileNet model is a suitable choice for training LPNet and achieving consistent results.

TABLE IV. The performance of state-of-the-art CNN architectures

Model	Parameters	MSLE	0-1-2 Acc	0-1 Acc
ResNet50	25,205,080	0.0072	0.986	0.966
MobileNetV1	4,327,640	0.0114	0.974	0.935
EfficientNetB0	5,279,415	0.0092	0.978	0.944
EfficientNetB1	7,805,083	0.0138	0.959	0.896
Xception	22,484,544	0.0070	0.990	0.976
InceptionV3	23,425,848	0.0136	0.962	0.902
RCConv ResNet50	25,422,232	0.0055	0.992	0.981
RCConv MobileNetV1	4,365,368	0.0034	0.996	0.990
RCConv EfficientNetB0	5,317,269	0.0049	0.993	0.983
RCConv EfficientNetB1	7,842,937	0.0151	0.954	0.882
RCConv Xception	22,522,272	0.0049	0.992	0.982
RCConv InceptionV3	23,463,576	0.0088	0.979	0.947

All experiments use the row-column arrangement of input samples. MSLEs of all architectures with RCConv is smaller than the one without. Especially, the MobileNet baseline with RCConv obtains the lowest loss and the best 0-1-2 accuracy as shown in Table IV.

The results of RCConv MobilenetV1 in Fig. 7 show the average of all correctly optimal binding predictions (1.0 acc) with respect to many LP problem sizes. LP size ratios (d) are defined as the following conditions:

$$d(m,n) = \begin{cases} \frac{n-m}{n} & \text{if } n > m, \\ 0 & \text{if } n = m. \end{cases} \tag{10}$$

Fig. 7 shows the number of LP models that LPNet can predict all optimal constraints of different ratios of d. When d

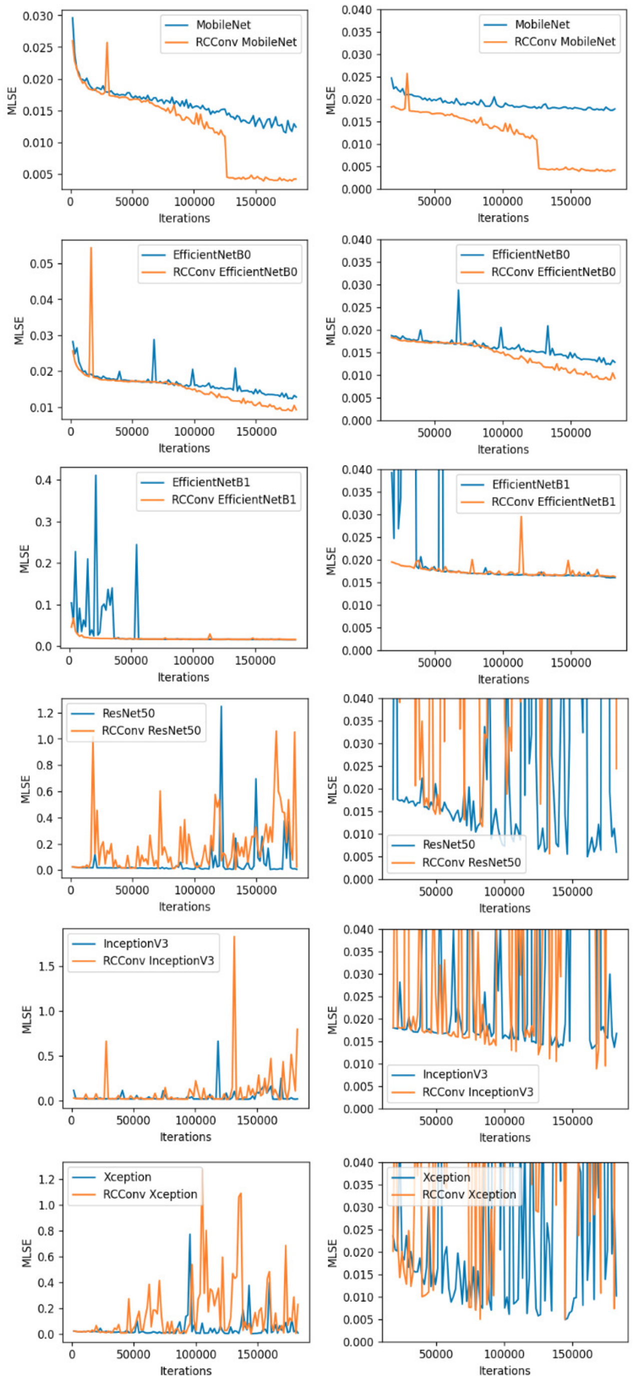


Fig. 6. Mean log square error of CNN baselines and RCConv CNN baselines of the validate dataset.

= 0, it means the number of constraints is equal to the number of decision variables. LPNet can predict all optimal binding constraints of LP problems over 80% when $0.2 \leq d \leq 0.3$. However, this model also suffers from an imbalance problem and many LP problems are infeasible if $m > m$ or $d > 0.8$. The next subsection shows the speed gain using LPNet to obtain the optimal solutions for LP problems.

From the above results, an optimal solution can be directly

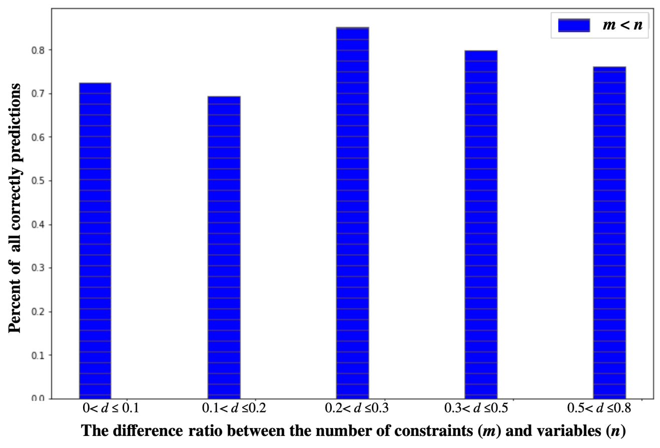


Fig. 7. The bar chart shows the average number of predicted optimal binding constraints that are correctly all components of $\mathbf{Y}$ depending on the LP problem ratios d.

obtained by solving the system of linear equations. There are n predicted constraints from $x_j \geq 0$ for $j \in \{1, ..., n\}$ and $a_{ij}x_j \geq b_i$ for $i \in \{1, ..., m\}$ that can be selected to solve the exact optimal solution. Fig. 8 shows the average total time for solving the LP problems where the number of constraints and variables are the same. The proposed LPNet algorithm saves a lot of time compared with the commercial solver Cplex [31]. The lowest solution time of the biggest LP size is 0.076 sec on GPU (Tesla-T4) using numpy.linalg solver. However, results are slightly longer on CPU (Xeon(R) 2.20GHz). Fig. 9 shows the average total time where $m < n$ and $n = 300$. LPNet GPU is also faster than LPNet CPU and the Cplex solver. LPNet GPU achieves 7.5 times faster than the cplex solution time.

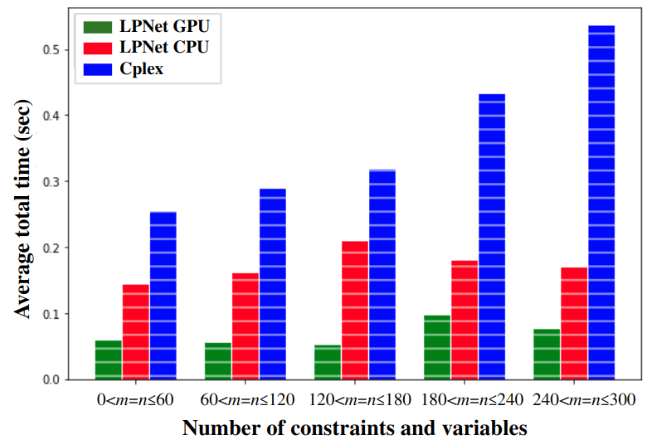


Fig. 8. Total LP solution time for the number of constraints equal to variables.

E. Netlib Dataset

Netlib is a collection of mathematical software, algorithms, and databases that were widely used in the scientific and engineering communities during the 1980s and 1990s. The dataset contains software packages for optimization, linear algebra,

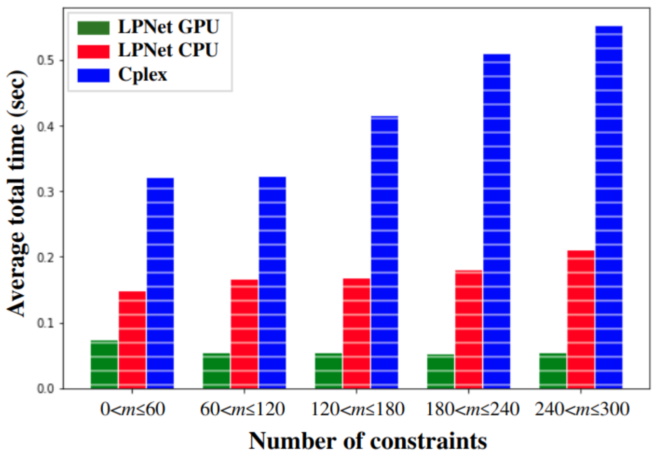


Fig. 9. Total LP solution time for the number of constraints with 300 variables.

differential equations, and other areas of scientific computing. It also includes benchmark datasets, including the Netlib LP dataset, which is a collection of linear programming problems commonly used to evaluate and compare the performance of optimization algorithms. Today, Netlib serves as an archive of legacy software and algorithms and is still widely used in academic research and education as a reference resource.

TABLE V. The Performance of LPNet Mobilenet Baseline on Netlib Data

LP problem	Optimal	m	n	0-1-2 Acc	0-1 Acc
ADLITTLE	225494.967	56	97	1.0	1.0
AFIRO	-464.753	28	32	1.0	1.0
BEACONFD	33592.485	173	262	0.998	0.997
BLEND	-30.812	74	83	0.996	0.987
BRANDY	1518.509	220	249	1.0	1.0
E226	-18.751	223	282	1.0	1.0
ISRAEL	-896644.827	174	142	1.0	1.0
SC50A	-64.575	50	48	1.0	1.0
SC50B	-70.0	50	48	1.0	1.0
SC105	-52.202	106	103	1.0	1.0
SC205	-52.202	205	203	1.0	1.0
SCAGR7	-2331389.816	129	140	1.0	1.0
SHARE1B	-76589.318	117	225	1.0	1.0
SHARE2B	-415.732	96	79	1.0	1.0
STOCFOR1	-41131.976	117	111	1.0	1.0

Table V lists selected LP problems from netlib having m and n less than 300. All of them can be directly sent to LPNet to identify the optimal solution.

F. Convolution Analysis

A visual explanation for explaining the relation between a target class and outputs of a CNN layer is introduced by Grad-CAM [30]. The main concept is to choose an interested target class in $\mathbf{Y}$ and take a partial derivative from the output-predicted class with respect to output feature maps of the interested CNN layer for measuring a gradient size corresponding to the sensitive area of the CNN feature maps. The output feature maps of each CNN layer can be localized maps highlighting important regions related to the target class.

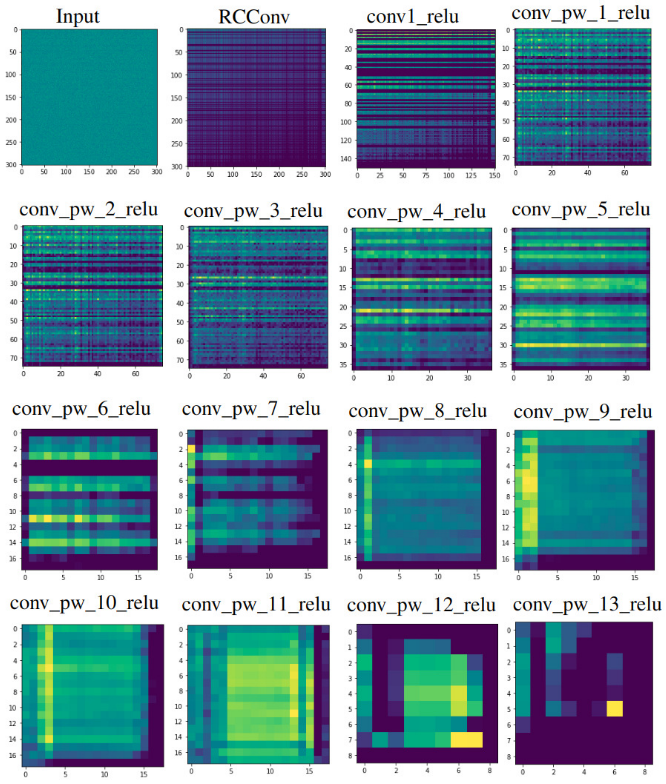


Fig. 10. Output feature maps of convolutional layers of the RCConv MobileNet baseline.

Fig. 10 shows the gradCAM heatmaps of each layer of one of the test problems of size 292×292 from LPNet with the MobileNet baseline. The all-unit-row-except-first-unit-column matrix of the test problem is plotted at the top left of Fig. 10. The second sub-figure to the right comes from the row-column convolutional layers (RCConv) which combine spatial information of rows and columns. By the concept of padding 0, the pixels over the 292x292 rows and columns respectively generate the dark color. The conv1_relu layer will be convoluted and reduced dimensions by half to get a general concept that is related to the prediction $\mathbf{Y}_i = 0$ for $i \in \{0, ..., 600\}$. MobileNet architecture has a lot of depthwise and pointwise convolutional layers for reducing many parameters compared with regular convolutional layers. Until the last pointwise CNN layer (conv_pw_13_relu), there are some feature maps that respond to the optimal binding prediction. Notice on the highlighted heatmap RCConv to conv_pw_5 layer, LPNet is extracting the sensitive area in terms of highlighted rows which corresponds to the optimal binding constraints. The higher layers will be convoluted and reduced dimensions in order to embed the important features as shown in the conv_pw_13_relu heatmap.

Further, three feature maps of all CNN layers can be localized by gradCAM for nonbinding constraints and padding constraints status in $\mathbf{Y}$ as shown in Fig. 11. For the output layer of LPNet with the MobileNet baseline, the prediction gives three possible prediction statuses which are 0, 1, or 2.

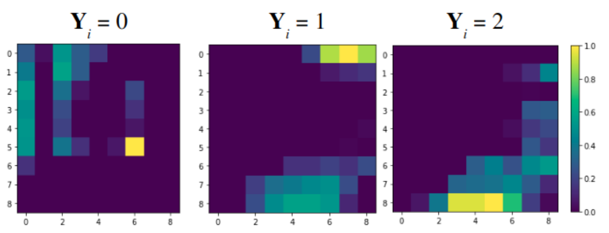


Fig. 11. GradCAM heat maps of the last CNN layer (conv_pw_13_relu) depend on the specific classes in $\mathbf{Y}$.

The important area of the specific status related to the last convolution (Conv_pw_relu) is shown in Fig. 11. In order to investigate where are the optimal binding constraints, the heatmaps from the gradCAM technique will only consider the prediction at $\mathbf{Y}_i = 0$ for $i \in \{0, ..., 600\}$ as shown in the left Fig. 11. The middle figure shows the heatmaps of nonbinding constraints that are $\mathbf{Y}_i = 1$ for $i \in \{0, ..., 600\}$. The heatmaps of the right figure are represented by the padding status $\mathbf{Y}_i = 2$ for $i \in \{0, ..., 600\}$. Moreover, this heatmap shows the separated area of binding constraints, non-binding constraints, and padding.

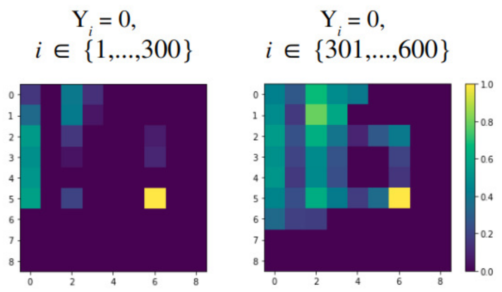


Fig. 12. Both heatmaps show the area highlight of the optimal binding constraints status $(0 \in \mathbf{Y})$ from non-negative variables and regular constraints, respectively.

Fig. 12 shows the localized heatmaps that are related to the non-negative constraints of variables from 1 to 300 and the original constraints from 301 to 600. It indicated that LPNet tried to capture the dual constraints see the left figure with highlight columns that relate to the non-negativity constraints and the right figure shows the highlight of heatmaps that affect the original constraints.

V. Discussion

This LPNet model has a limitation that it works for any linear programming problem having sizes less than 300 rows and 300 columns due to the hardware limitation. To solve a linear programming problem of larger sizes, new training data must be synthesized and solved and weights of the state-of-the-art deep learning model must be retrained on large system resources.

For the perfect prediction of the optimal binding constraints, LPNet determines the optimal solution of a linear programming problem algebraically, without the need of an iterative step. However, if the prediction is not 100%, some constraints are not the optimal binding constraints. The real optimal constraints must be reidentified. So this work can be extended using the iterative procedure after the optimal binding prediction. A complete linear programming solver utilizing the predicted optimal binding constraints could be created.

In 2023, [25] introduced a novel deep learning approach using feedforward neural networks to solve the LP problem. The approach models the LP problem by an ordinary differential equations (ODE) system, the state solution of which globally converges to the optimal solution of the LP problem. A neural network model is constructed as an approximate state solution to the ODE system, such that the neural network model contains the prediction of the LP problem. The neural network is extended by taking the parameter of LP problems as an input variable so that one neural network can solve multiple LP instances in a one-shot manner. However, it is important to note that the proposed method's performance has only been tested on a specific collection of small LP examples. Its efficacy on more complex or diverse LP problems remains uncertain.

Overall, LPNet provides a promising approach for identifying optimal binding constraints in LP problems, which can greatly reduce the computation time required for traditional iterative solvers. With further development of the algorithm, LPNet has the potential to become a powerful tool for solving complex LP problems in various fields.

VI. Conclusions

This paper presents the deep learning architecture for identifying the optimal binding constraints called LPNet. A linear programming problem must be cast as the all-unit-row-except-first-unit column matrix with row-column rearrangement before sending it to LPNet. LPNet is composed of the row-column convolutional layer followed by the state-of-the-art convolutional neural network models and two fully connected layers of neural networks and ends with the output layer. With RCConv + MobileNetV1 + two fully connected layers + output layer, LPNet achieves 99.6% 0-1-2 accuracy from a million synthesized linear programming problems with finite optimal solutions.

The form of the all-unit-row-except-first-unit column has been studied to show the performance obtained for scaling to unit-vector over the max-min normalization and the standardization of rows. Moreover, the arrangement of rows and columns also helps reduce training loss.

LPNet is able to predict 80% of linear programming problems with all optimal binding constraints from generated linear programming problems and it correctly predicts 86% benchmark netlib problems of size smaller than 300 variables and 300 constraints. It can achieve the optimal solution faster than the cplex solver more than 6 times.

In general, any LP problem may not have an optimal solution. It will be very useful to design deep learning to categorize LP problems whether they are infeasible, unbounded optimal, or have a finite optimal solution. Moreover, LPNet weights can be used to accelerate the solution time of any commercial LP solver.

Acknowledgment

This research is supported by the Science Achievement Scholarship of Thailand and the Applied Mathematics and Computational Science Program in the Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Thailand.

References

[1] M. R. Hussain, and M. E. Hussain, "Simplex method to Optimize Mathematical manipulation," International Journal of Recent Technology and Engineering (IJRTE), vol. 7, January 2019.

[2] A. P. Florian, and J. W. Stephen, "Interior-point methods," Journal of Computational and Applied Mathematics, vol. 124, pp. 281–302, 2000.

[3] M. Soleimani-damaneh, "Modified big-m method to recognize the infeasibility of linear programming models," Knowledge-Based Systems, vol. 21, pp. 377–382, 2008.

[4] A. Boonperm, and K. Sinapiromsaran, "Artificial-free simplex algorithm based on the non-acute constraint relaxation," Appl. Math. Comput., vol. 234, pp. 385–401, May 2014.

[5] R. Visuthirattanamanee, K. Sinapiromsaran, and Boonperm, "A. Self-Regulating Artificial-Free Linear Programming Solver Using a Jump and Simplex Method," Mathematics, vol. 8, 2020.

[6] G. H. Andrew, Z. Menglong, C. Bo, K. Dmitry, W. Weijun, W. Tobias, A. Marco, and A. Hartwig, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv, 2017.

[7] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," Automatica, vol. 38, pp. 3–20, January 2002

[8] K. Justin, P. Iosif, A. Styliani, and N.P. Efstratios, "Integrating deep learning models and multiparametric programming," Computers & Chemical Engineering, vol. 136, May 2020.

[9] Z. Ling, R. Liu, Y. Zhang, and X. Chen, "Can Deep Learning Solve Parametric Mathematical Programming? An Application to 0–1 Linear Programming Through Image Representation," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 52, pp. 5656–5667, September 2022.

[10] S. Khade, S. Gite, B. Pradhan, "Iris Liveness Detection Using Multiple Deep Convolution Networks," Big Data Cogn. Comput., vol. 6, June 2022.

[11] Y. Yang, Z.-Y. Fu, D.-C. Zhan, Z.-B. Liu, and Y. Jiang, "Semi-Supervised Multi-Modal Multi-Instance Multi-Label Deep Network with Optimal Transport", IEEE Transactions on Knowledge and Data Engineering, PP. 1-1., August 2021.

[12] T. T. Nguyen, T. T. T. Nguyen, A. V. Luong, Q. V. H. Nguyen, A. W.-C. Liew, and B. Stantic, "Multi-label classification via label correlation and first order feature dependance in a data stream," Pattern recognition, vol. 90, pp. 35–51, June 2019.

[13] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in CVPR, 2019.

[14] D. C. R. Novitasari, F. Fatmawati, R. Hendradi, H. Rohayani, R. Nariswari, A. Arnita, M.I. Hadi, R. A. Saputra, and A. Primadewi, "Image Fundus Classification System for Diabetic Retinopathy Stage Detection Using Hybrid CNN-DELM," Big Data Cogn. Comput., vol. 6, December 2022.

[15] G. Huang, Z. Liu, L. Van der maaten, and K.Q. Weinberger, "Densely Connected Convolutional Networks," arXiv, 2016.

[16] S. Christian, L. Wei, J. Yangqing, S. Pierre, E. R. Scott, A. Dragomir, E. Dumitru, V. Vincent, and R. Andrew, "Going Deeper with Convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, pp. 1–9, October 2015.

[17] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, Linear Programming and Network Flows, 3rd ed., John, W, New York, 2005.

[18] G. B. Dantzig, Linear Programming and Extensions, Princeton Univ. Press: Princeton, NJ, 1963.

[19] D. G. Luengberger, Linear and Nonlinear Programming, 2nd ed., Springer, New York, 2005.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016.

[21] M. Tan, and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," International Conference on Machine Learning, PMLR, pp. 6105–6114, 2019.

[22] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," Proceedings of the IEEE CVPR, pp. 1251–1258, 2017.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, pp. 1–9, 2015.

[24] C. N. Jones, and M. Morrari, "Multiparametric linear complementarity problems," Proceedings of the 45th IEEE Conference on Decision and Control, pp. 5687–5692, 2006.

[25] D. Wu, and A. Lisser, "A deep learning approach for solving linear programming problems," Neurocomputing, vol. 520, pp. 15–24, 2023.

[26] C. Flamant, P. Protopapas, and D. Sondak, "Solving differential equations using neural network solution bundles," arXiv, 2020.

[27] S. Effati and A. R. Nazemi, "Neural network models and its application for solving linear and quadratic programming problems," Applied Mathematics and Computation, vol. 172, pp. 305–33, 2006.

[28] K. Kavitha and Suneetha Chittineni, "An Intelligent Metaheuristic Optimization with Deep Convolutional Recurrent Neural Network Enabled Sarcasm Detection and Classification Model," International Journal of Advanced Computer Science and Applications(IJACSA), vol. 13, 2022.

[29] Large-Scale Machine Learning on Heterogeneous Distributed Systems. Available online: https://www.tensorflow.org

[30] R. S. Ramprasaath, C. Michael, D. Abhishek, V. Ramakrishna, P. Devi, and B. Dhruv, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," International Journal of Computer Vision, vol. 128, pp. 336–359, 2019.

[31] Cplex, I. I. (2009). V12. 1: User's Manual for CPLEX. International Business Machines Corporation, 46(53), 157.