Practical Approach to Evacuation Planning Via Network Flow and Deep Learning

Akira Tanaka*, Nozomi Hata*, Nariaki Tateiwa*, and Katsuki Fujisawa†
*Graduate School of Mathematics, Kyushu University, Fukuoka, Japan
†Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan
{ma216030@math, ma216009@math, ma217003@math, fujisawa@imi}.kyushu-u.ac.jp

***Abstract*—In this paper, we propose a practical approach to evacuation planning by utilizing network flow and deep learning algorithms. In recent years, large amounts of data are rapidly being stored in the cloud system, and effective data utilization for solving real-world problems is required more than ever. Hierarchical Data Analysis and Optimization System (HDAOS) enables us to select appropriate algorithms according to the degree of difficulty in solving problems and a given time for the decision-making process, and such selection helps address real-world problems. In the field of emergency evacuation planning, however, the Lexicographically Quickest Flow (LQF) algorithm has an extremely long computation time on a large-scale network, and is therefore not a practical solution. For Osaka city, which is the second-largest city in Japan, we must solve the maximum flow problems on a large-scale network with over 8.3M nodes and 32.8M arcs for obtaining an optimal plan. Consequently, we can feed back nothing to make an evacuation plan. To solve the problem, we utilize the optimal solution as training data of a deep Convolutional Neural Network (CNN). We train a CNN by using the results of the LQF algorithm in normal time, and in emergencies predict the evacuation completion time (ECT) immediately by the well-learned CNN. Our approach provides almost precise ECT, achieving an average regression error of about 2%. We provide several techniques for combining LQF with CNN and addressing numerous movements as CNN's input, which has rarely been considered in previous studies. Hodge decomposition also demonstrates that LQF is efficient from the standpoint of the total distance traveled by all evacuees, which reinforces the validity of the method of utilizing the LQF algorithm for deep learning.**

***Keywords*-universally quickest flow; deep learning; tsunami evacuation planning; graph analysis**

I. Introduction

In recent years, catastrophic disasters caused by massive earthquakes have been increasing, and effective evacuation plans are deemed essential for our society more than ever. For creating evacuation plans, we need to not only minimize Θ^*, that is, the evacuation completion time (ECT), but also maximize the cumulative number of evacuees who have already completed the evacuation at an arbitrary time until Θ^*. The evacuation planning problem with this criterion can be formalized by the so-called Universally Quickest Flow (UQF) [8]. However, UQF does not always exist if capacity constraints exist as regards refuges. Takizawa et al. [9] pointed this out with a counterexample and presented an algorithm for finding the Lexicographically Quickest Flow (LQF), which can be regarded as a practical extension of UQF. In this case, not only are the capacity constraints considered, but LQF equals UQF if there exists UQF. The practicality of LQF is reinforced with respect to the congestion of evacuees, or the total pedestrian flow of all evacuees to the refuges. In Section IV, we analyze the efficiency of LQF by decomposing LQF to cyclic and cycle-free flows.

An LQF algorithm provides an effective and practical evacuation plan from many perspectives as indicated above; however, we need more time to compute the LQF algorithm on large-scale networks despite LQF having a polynomial-time algorithm. For example, we have already computed the evacuation plan for Osaka city by utilizing LQF. In this case, we finally compute the maximum flow algorithm for LQF on a large-scale network that has over 8.3M nodes and 32.8M arcs, and note that the total computation time is over seven hours even though we use a high-performance computing server, which prevents using LQF in emergencies. In such a situation, we must choose an appropriate algorithm according to both the computation time needed to solve problems, and the given time for the decision-making process. We can regard the Hierarchical Data Analysis and Optimization System (HDAOS) explained in Section II as a very useful framework when solving large-scale and complex problems in the real world.

We finally propose a simple but powerful way to predict ECT via LQF and the Convolutional Neural Network (CNN) on HDAOS. We take inspiration from the significant progress on CNN, including deep learning. Inspired by Hubel and Wiesel's study on the neurobiological signal processing in cats' visual cortex [11], hierarchical neural network CNN is devised and grow rapidly with the development of the computation performance of computers. The early success of CNN was in handwriting recognition [12], which was applied to various computer vision tasks, particularly detection and classification [13], [14], [15]. It has also been very successful in capturing a movement like optical flow estimation [16], [17] and synthesizing character movements regarding the manifold of human motion [18]. These two researches have similarities in addressing movements; however, numerous movements of evacuees must be converted into CNN's input in our research.

We use two techniques for creating input data from many movements: resolution of vector and geometry modification.

978-1-5386-2715-0/17/$31.00 ©2017 IEEE

Resolution of vector is very useful for representing movements when evacuees move in opposite directions and cancellations of vectors occur. It also compresses the numerous movements into small dimensions without large amounts of information loss. We also treat geometric information delicately, especially for movements near the boundaries of areas. Kevin et al. [19] show the advantage of using Local Contrast Normalization Layer for object recognition. As evident by the layer name, they locally perform Gaussian normalization with weights following Gaussian distribution from the center point, under the constraint that the sum of weights equals one. We use a similar method, but not Gaussian distribution. Experiments show that a proposed model achieves high performance that the average error is quite small (2%), and even in the worst case, the regression error is limited to an acceptable range.

II. Hierarchical Data Analysis and Optimization System

We commenced our research project for developing the Urban Operating system (OS) for a large-scale city, in 2013 [1]. The Urban OS, which is regarded as one of the emerging applications of the cyber-physical system (CPS), gathers big data sets of people and transportation movements by utilizing different sensor technologies and storing them in the cloud storage system. As mentioned in our previous papers [3], [4], we have another research project whose objective is to develop advanced computing and optimization of infrastructures for extremely large-scale graphs on post peta-scale supercomputers. For example, our project team was the winner at the eighth, and 10th to 14th Graph500 benchmark [2] [5]. The Urban OS employs the graph analysis system developed by this research project and provides a feedback to a predicting and controlling center to optimize many social systems and services.

Here, we focus on the HDAOS based on CPS, which are illustrated in Figure 1. First, we gather a variety of data sets on a physical space and generate mathematical models for analyzing the social mobility of real worlds. In the next step, we apply optimization and simulation techniques to solve them and check the validity of solutions obtained on the cyber space. We finally feed these solutions into the real world.

Figure 2 shows three analysis layers, and we can choose the appropriate one according to a given time for the decision-making process. Figure 2 shows the algorithmic specifications of each layer of HDAOS. We classify many optimization algorithms into three layers according to both the computation time needed to solve problems, and the data size of the optimization problem. We have developed parallel software packages for many optimization problems

[1]http://coi.kyushu-u.ac.jp/en/

[2]http://www.graph500.org/

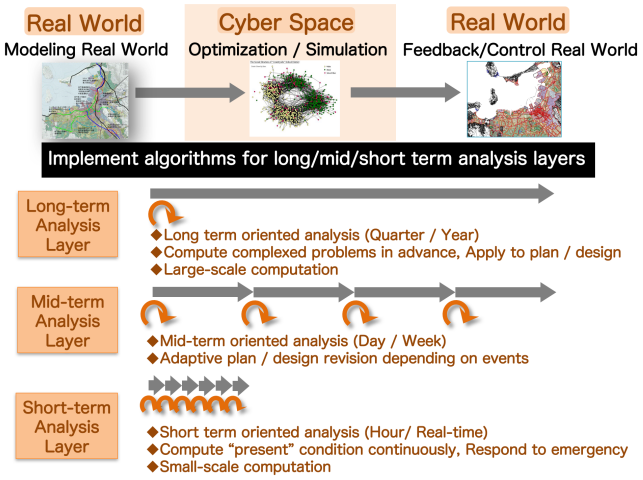

Figure 1: Hierarchical Data Analysis and Optimization System (HDAOS)

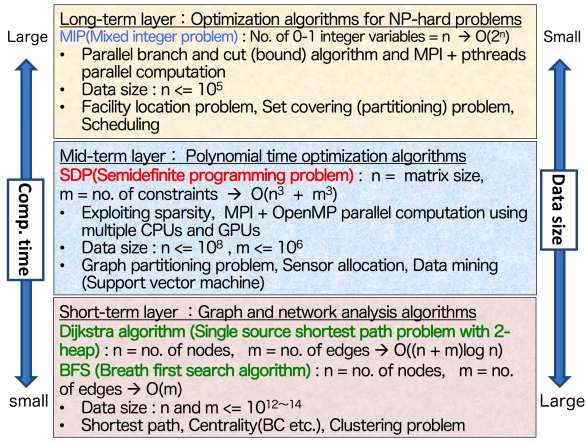

Figure 2: Specification of Each Layer of HDAOS

categorized into these three algorithmic layers. The long-term analysis layer contains optimization algorithms for NP-hard problems. The most typical and important optimization algorithm in this layer is a branch-and-cut algorithm for the mixed integer problem(MIP). We have collaborated with ZIB (Zuse Institute Berlin) [3] in developing and evaluating parallel (MPI + pthread) software packages for solving MIPs. The mid-term and short-term analysis layers contain SDP problems [6], [7] and BFS for graph analysis [5], respectively.

In this paper, we have utilized LQF and supervised machine learning with CNN. The time complexity of LQF is $\mathcal{O}(nm\log(n^2/m) \times \mathcal{O}(\log\theta)$, where n, m, and θ represent the number of nodes, number of arcs, and the ECT, respectively. The computation of the algorithm that can predict the ECT based on supervised machine learning with CNN is

[3]http://www.zib.de/

basically fast; therefore, it belongs to the short-term analysis layer, whereas LQF and the supervised machine learning with CNN belong to the mid-term analysis layer. This indicates that we can utilize LQF and supervised machine learning with CNN in normal time and predict the ECT very quickly in emergencies.

Table I: Graph Data of Yodogawa Area in Osaka City

#node	#arcs	#evacuees	#the total amount of finite sinks	#sinks with
2,933	8,924	50,000–86,000	36,549	finite capacity: 36 infinite capacity: 50

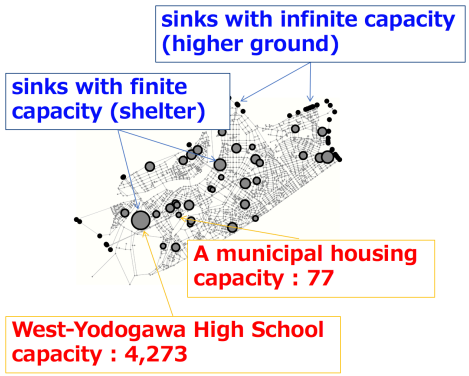

Figure 3: Evacuation Map of Yodogawa Area in Osaka City

III. Lexicographically Quickest Flow

In this section, we define UQF and LQF, and show how to apply this model to the emergency evacuation planning. Let $N = (D = (V, A), c, S, b)$ be a static network, where D is a graph consisted of a node set V and an arc set A, $c : A \to \mathbb{R}_+$ is a capacity function, $S = \{x_1, x_2, \ldots, x_k\} \subset V$ is an ordered sink set, and $b : V \to \mathbb{R}_+$ is a supply function with $b(s) = 0$ for each $s \in S$. Let $LFV_{\max}(S')(S' \subset S)$ denote the maximum flow value of a feasible flow f that enters S'. Let $S(i) = \{x_1, x_2, \ldots, x_i\}(i = 1, 2, \ldots, k)$. Let $\tau : A \to \mathbb{Z}_+$ be the transit time function. A lexicographic maximum flow is a maximum flow whose flow value is $LFV_{\max}(S(i))$ for each $i = 1, 2, \ldots, k$. UQF is defined as a lexicographic maximum flow on $N(\Theta^*)$, a time-expanded graph with the ECT Θ^*. A time-expanded graph is a static network in which for each $v \in V$ and each time step $\theta \in \{0, \ldots, \Theta^*\}$, there exists a node $v(\theta)$, and for each arc $e \in A$ from $u \in V$ to $v \in V$ and $\theta \in \{0, \ldots, \Theta - \tau(e)\}$, there exists an arc from $u(\theta)$ to $v(\theta + \tau(uv))$ with capacity $c(e)$, and for each $v \in V$ and each $\theta \in \{0, \ldots, \Theta^*\}$ there exists a holdover arc from $v(\theta)$ to $v(\theta + 1)$ with infinite capacity. For each $v \in V$, the supply of $v(\theta)$ is set to $b(v)$ if $\theta = 0$, and 0 otherwise. The sink at time θ $\{s(\theta) \mid s \in S\}$ is aggregated into a super sink node $s^*(\theta)$ as well as an arc with infinite capacity. The lexicographic maximum flow considering the ordered set of sinks $\{s^*(0), s^*(1), \ldots, s^*(\Theta^*)\}$ is UQF. If some of the capacities of the refuges are finite, then UQF does not always exist. Therefore, we compute LQF instead, which can be obtained by repeatedly computing the maximum flow with a single sink $s^*(\theta)$ until all evacuees complete evacuation.

Table II: Time-Expanded Graph of Yodogawa Area

Θ	1	2	3	.	.	2,824
#node	5,953	8,886	11.8K	.	.	8.3M
#arcs	5,210	8,151	11.1K	.	.	32.8M

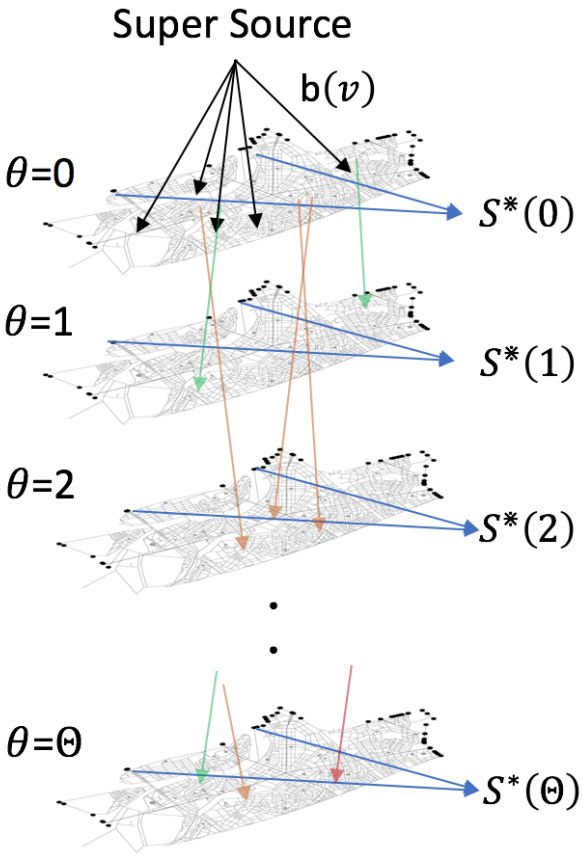

Figure 4: Visualization of Time-Expanded Graph of Yodogawa Area

We applied the LQF algorithm to the Yodogawa area of Osaka in Japan, as explained in Table I and Figure 3. The graph of Yodogawa area is generated from the GIS data as follows: the fundamental map information [4] to generate the graph, the housing map(ZmapTown II 2005) for refuges, and the population census [5] for refugees. We chose higher grounds, which are expected to avoid tsunamis, and some strong and tall buildings as the sinks for the predicted tsunami. The capacities of former sinks are set to infinity

[4]http://www.drm.jp/english/drm/e_index.htm
[5]http://www.stat.go.jp/english/data/kokusei/2010/summary.htm

and those of the latter are set to finite values depending on the size of the buildings. The time-expanded graph is shown in Figure 4, and we show that its nodes and arcs are extremely huge when time is expanded largely in Table II. We also note that the LQF algorithm requires computing maximum flow problems on the time-expanded graphs with time steps $\theta = 1, \cdots, \Theta^*$ repeatedly.

IV. Practical Property of LQF

In this section, we define how to decompose the LQF and show an efficiency of LQF with respect to the total movements of the evacuees. Let $D = (V, A)$ be an undirected graph, $\mathcal{F} \subset \mathbb{R}^{\mathcal{A}}$ be the set of all flows on D, and $\mathcal{F}_c \subset \mathcal{F}$ be the set of all cyclic flows on D. $\mathcal{F}$ and $\mathcal{F}_c$ can be regarded as linear subspaces of $\mathbb{R}^A$. We have several methods to decompose a flow $f \in \mathcal{F}$ into a cyclic flow f_c and cycle-free flow f_d. First, we can obtain a cycle-free component of f by solving a minimum-cost flow problem whose capacity function is f, and wherein each cost is set to one. Second, we can decompose f by applying the Hodge decomposition Algorithm. Finally, we developed an algorithm to decompose flow, inspired by the first and second methods. The first method is well known: given $f \in \mathcal{F}$, the formulation as a mathematical optimization problem is:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in A} |f(e) - f_c(e)| \\ \text{subject to} \quad & f_c \in \mathcal{F}_c \\ & 0 \le f_c(e) \le f(e) \qquad (\forall e \in A \text{ s.t. } f(e) \ge 0) \\ & f(e) \le f_c(e) \le 0 \qquad (\forall e \in A \text{ s.t. } f(e) \le 0). \end{aligned}$$

The optimal solution also minimizes $\sum_{e \in A} f_d(e)$, because it satisfies the last two constraints.

The second method is the Hodge decomposition Algorithm, which is an analogy of the Helmoltz-Hodge decomposition in the area of vector calculus [22]: for any differentiable vector field F on a bounded domain V, there exists a scalar potential Φ and vector potential A, and $F = \mathrm{grad}\Phi + \mathrm{curl}A$. The Hodge Decomposition Algorithm decomposes a flow f into a cyclic flow f_c and a flow f_d with a scalar potential u: $f_d(v, w) = u(w) - u(v)$. The formulation is

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in A} (f(e) - f_c(e))^2 \\ \text{subject to} \quad & f_c \in \mathcal{F}_c \end{aligned}$$

To solve this quadratic optimization problem, we need to obtain a basis of $\mathcal{F}_c$ first. To obtain this, we regard a graph as a simplicial complex and applied the CHomP [6] software to compute a basis of $\mathcal{F}_c$. Let $K := V \cup A$ be an abstract simplicial complex. Let C_i be the i-th chain group of K and H_i be the i-th homology group of K. Let $\langle c \rangle$ denote the equivalence class of $c \in C_1$ in H_1. Since $\langle e_1 + \ldots + e_k \rangle \in H_1$ for any cyclic path $\{e_1, \ldots, e_k\} \subset A$, any cyclic flow $f_c \in \mathcal{F}_c$ can be represented as an element in H_1. Given $f \in \mathcal{F}$, the Hodge decomposition algorithm decomposes f into a flow in $\mathcal{F}_c$ and a flow in $\mathcal{F}_c^{\perp}$ through orthogonal projection.

The third method is formulated by adding constraints to the second method. The formulation is

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in A} (f(e) - f_c(e))^2 \\ \text{subject to} \quad & f_c \in \mathcal{F}_c \\ & 0 \le f_c(e) \le f(e) \qquad (\forall e \in A \text{ s.t. } f(e) \ge 0) \\ & f(e) \le f_c(e) \le 0 \qquad (\forall e \in A \text{ s.t. } f(e) \le 0) \end{aligned}$$

Table III: Results of Decompositions

	Objective value	$l2$ norm of f_c
1st method	5.9×10^6	28
2nd method	3.2×10^8	2.8×10^8
3rd method	6.0×10^8	128

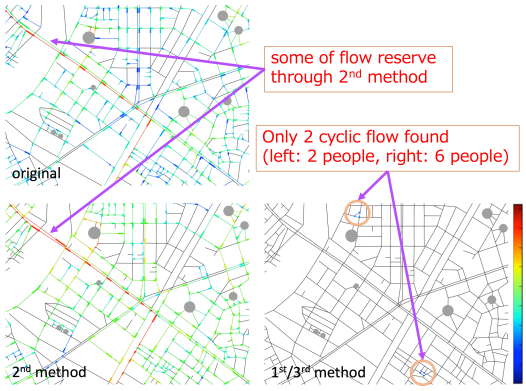

Figure 5: Results of LQF and Hodge Decomposition

Figure 5 and Table III show the result of each decomposition. LQF is originally a dynamic flow, thus we decomposed a flow f whose value of $e \in A$ is the total amount of evacuees that passed e in LQF. The upper left picture shows LQF, whose color represents the total number of evacuees passed. The red edges represent that many evacuees have passed and the blue edges indicate that only a few evacuees have passed. The presence of a sink with unlimited capacity at the upper left of this figure indicates that many evacuees walked to the upper left side. The lower right of the figure shows the results of the first and third methods, which are the same. We observe only two cycles in LQF. This suggests that LQF has few unnecessary movings. Meanwhile, LQF is

[6]CHomP: Computational Homology Project, http://chomp.rutgers.edu/

not quite a "simple" as that explained by a flow with a scalar potential.

V. Predicting Evacuation Completion Time

A. Practical Model

For emergency evacuation planning, we obtain the optimal evacuation plan after serious damage is caused because of the long running time of the LQF algorithm. To address such a problem, we usually implement the LQF algorithm on various kinds of population distributions, and the optimal evacuation plans for each situation are gradually stored. These results are utilized as training data for a proposed CNN after applying preprocessing techniques (see subsection V-B). The fact that this step is time consuming is not really relevant because we have plenty of time in normal time. As a result, we obtain a well-learned network for predicting the ECT of the best case. It is worth noticing that we build the short-term layer model by utilizing the mid-term layer algorithm, which has several optimal properties. The overview of a proposed model is illustrated in Figure 6, and a comparison between the previous and proposed models is shown in Figure 7. Few reports are available for supervised learning by utilizing optimization algorithms, and a proposed model is one of the successful cases according to the framework.

Phase	Operation	Computation Time
Normal	Calculating LQF Training a CNN	7 hour 4 min
Emergency	Prediction with a learned CNN	3.71×10^{-4} sec

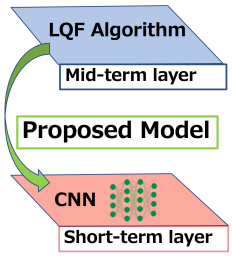

Figure 6: Proposed Model and Numerical Results

B. Novel Preprocessing Techniques For CNN

A graph topology and movements of evacuees are obviously essential for predicting accurate ECT. Meanwhile, it is well known that a CNN extracts various kind of features. Therefore, we introduce a method to convert movements of evacuees into CNN's input, and the graph topology is expected to be implicitly embedded in our network through proper representation of movements. Not much is reported for making use of numerous movements as CNN's input, hence our preprocessing techniques for a CNN are novel approaches, and experiments show that our approach is highly effective in practical use (see subsubsection V-D3, V-D4).

An LQF algorithm decides the movements of all evacuees at any arbitrary time until the ECT Θ^*. Hence, the aim of a proposed model is to predict Θ^* based on the movements of all evacuees from $\theta - 1$ to θ, where θ is a certain time. Since spacial-contextual features are needed to be represented as input, we first take a training patch according to the minimum rectangle box, which covers all nodes and arcs of the graph. Next, we divide the patch into H by W regions, where H and W are the height and width of the patch, respectively. We also define D basic directions as $0, 2\pi/D, 2 \cdot 2\pi/D, \ldots, (D-1) \cdot 2\pi/D$, and each region possesses the number of evacuees walking into each basic direction by using the following techniques. Each region also possesses the number of pausing evacuees in the region; hence, each region has $D+1$ channels in total.

We use two techniques for creating input data: resolution of vector and geometry modification. We first express one person movement as an original vector whose direction and magnitude equal the direction of the movement and 1, respectively. Next, we split the original vector into two basic vectors so that the angle between the original vector and each basic vector is less than or equal to $2\pi/D$. We note that basic vectors make the counterclockwise angle $0, 2\pi/D, \ldots, (D-2) \cdot 2\pi/D$, or $(D-1) \cdot 2\pi/D$, respectively, from the positive x-axis. Finally, these resolved two basic vectors are normalized by multiplying the same coefficient so that the sum of the length of these two vectors equals one. We refer to these two vectors as normalized basic vectors, and this procedure is explained in Figure 8. If we use only the vector sum for each region, cancellations of vectors occur when opposite vectors are added, which leads to ignoring not only the evacuees moving in opposite directions but also the effect of these evacuees on other regions. We also note that we can compact the movements of evacuees without reducing the amount of information, which is a highly desirable feature for creating input data, especially for large cities.

Subsequently, we propose a geometry modification technique. We first consider a rectangle R whose size is the same as one region defined above, and the center point is the same as the tail of the original vector. All regions that overlap R are denoted by $R_1, R_2, \ldots, R_n$. For each R_i, we calculate the intersection area of R and R_i $(R \cap R_i)$, then divided by the size of R. Two normalized basic vectors are multiplied by the value, and the magnitude of each vector is added to the corresponding channel of the region. This method is explained in Figure 9, and the pausing evacuees are managed in the same way. By applying this method, we reflect the geometric information delicately, especially for evacuees near the boundaries of areas. Two input data when D equals four are illustrated in Figure 10. The LQF_T and the arrow on each figure represent the remaining time for ECT and the corresponding channel, respectively. For example, the upper left figure shows the number of evacuees in each region who move toward the right direction. The color indicates the data value, and red means a large number while blue implies a small one.

C. Network Architecture and Optimizer

We use a famous CNN with a slight modification to suit our problem. Yi et al. [1] propose three-level carefully

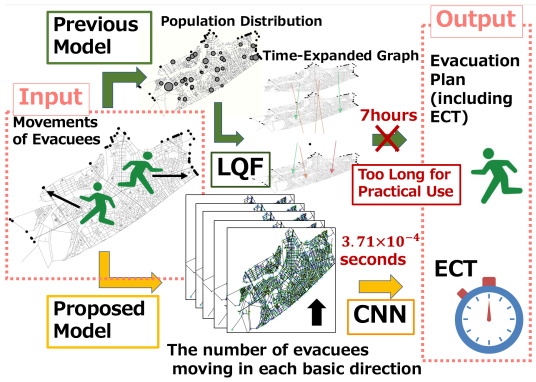

Figure 7: Comparison Between Previous and Proposed Models

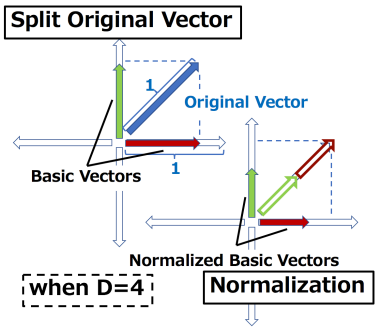

Figure 8: Resolution of Vector

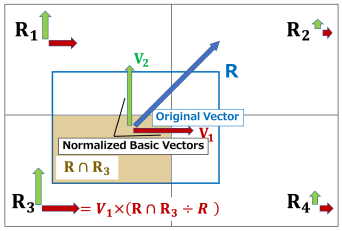

Figure 9: Geometry Modification

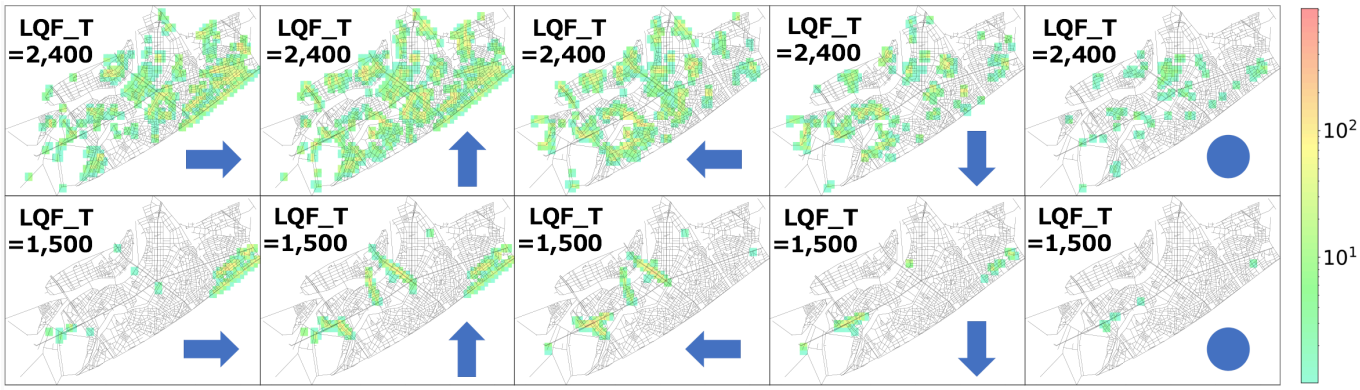

Figure 10: Input Data with Proposed Preprocessing Technique

designed convolutional networks for estimating the facial keypoints positions, achieving robust and accurate estimation. This method builds a regression model and succeeds to extract global high-level features over the whole input data, which is similar to what we want to do.

They use three-level cascaded CNNs. The networks at the second or third level are trained to locally refine the first-level outputs, and their inputs are limited to small regions around the initial predictions. However, only local information is not useful for predicting ECT because the movements of evacuees are determined by the current population distribution and the capacities of refuges. Therefore, we use their first-level regression network, which shows the robust and accurate estimation in their work. They employed locally sharing weights of neurons on the CNNs; however, it is not appropriate for our work. We want to mainly extract the congestion of evacuees and its effect on ECT, and for that it is not necessary to extract many features varying from area to area. Moreover, weight-sharing on a CNN helps prevent gradient diffusion when back-propagating works through many layers since the gradients of shared weights are aggregated, which makes supervised learning on deep layers easier. Therefore, we use a traditional CNN that shares the weights of all neurons on the same map.

Next, we explain the purpose of employing a deep CNN on a proposed model. Neurons in the lower layers are local due to local receptive fields of CNN; however, by combining spatially nearby features extracted at lower layers, neurons at higher layers can extract global features. Moreover, nonlinear activation functions after the layers make it possible to represent the complex relationship between the input and output successfully. We expect to obtain the congestion of small regions at the lower layers, and as the layers become deeper we obtain the congestion of larger regions and its effect on the ECT, which is quite useful for highly accurate prediction.

Finally, we select an appropriate optimizer $Adam$ [2], an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. This method combines the advantages of two popular methods: $AdaGrad$ [20], which works well with sparse gradients, and $RMSProp$ [21], which works well in on-line and non-stationary settings. This method also typically requires little tuning. On the other hand, our

input data is often sparse because there are few evacuees in the final stage of evacuation, and there are no nodes and arcs in some regions. Therefore, *Adam* is suitable for our input data. The structure of a proposed CNN is illustrated in Figure 11, and the details of our network and optimizer are mentioned in subsection V-D2.

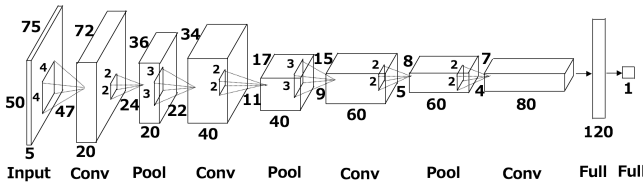

Figure 11: Structure of the Proposed CNN
Size of input, convolution (Conv), max pooling (Pool), and fully connected (Full) layers are illustrated by cuboids whose length, width, and height denote the number of maps, and the size of each map. Local receptive fields of neurons in different layers are illustrated by small squares in the cuboids.

D. Experiments

1) Input Data: We applied a proposed model for predicting ECT to Osaka city. We make 20 population distributions on the Yodogawa area of Osaka city, each of which is log-normal distributed, then apply an LQF algorithm to them separately. As a result, we obtain 20 different flows as the outputs of the algorithm and divide these 20 flows into 10 groups. Each group has two flows, and there is no overlap with each other. Subsequently, we make 10 data sets, each of which has training, test, and validation data sets. The train data set has eight groups, and both test and validation data sets have one group each. Note that the training, test, and validation data sets are not overlapped. We make input data from each flow per second by utilizing our preprocessing technique, assuming that the desired output is the ECT given by the LQF algorithm. The ECT of each flow is about 2,700 seconds under these circumstances, and a training data set has eight groups, each having two flows. As a result, the number of training data for each set is about 2,700·8·2=43,200. Similarly, the numbers of test and validation data are about 5,400 each. We also note that the difference among the sets is in the allocation of flows to training, test, and validation data. Figure 12 and Table IV show the overview of input data.

Table IV: Number of Data and Explanation of "set"

Type (Number of Data)	Set 1	Set 2	...	Set 10
Training $(43,200)$	Groups 1–8	Groups 2–9	...	Groups 1–7,10
Test $(5,400)$	Group 9	Group 10	...	Group 8
Validation $(5,400)$	Group 10	Group 1	...	Group 9

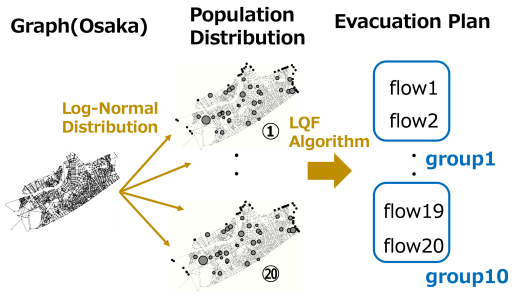

Figure 12: Overview of Creating Input Data

2) Implementation: The input layer is denoted by $I(h, w, d+1)$, where h, w, and $d+1$ are the height, width, and channel of the input, respectively. We divide each value of input data by the maximum value of the corresponding channel (map) to limit the input value. It is worth noticing that the ratio of same channel values (the number of evacuees moving toward a specific direction in each region) are very important and we do not need to normalize (subtract mean, divide by standard deviation). The convolutional layer is denoted by $CR(k, s, n)$ if the rectified linear unit (ReLU) is used, otherwise $C(k, s, n)$. k is the side length of the square convolutional kernels (or filters). s is the stride and n is the number of maps in the convolutional layer. The pooling layer is denoted by $P(s)$. s is the side length of the square pooling regions. Max pooling is used and the pooling regions are not overlapped. The fully connected layer is denoted by $FR(n)$ if ReLU is used, otherwise $F(n)$. n is the number of neurons in the current layer. We use a famous CNN with slight modifications (see subsection V-C), and the CNN is denoted by a proposed CNN in this paper. Details are presented in Table V.

The learnable network parameters including the weight and bias are initialized by small random numbers, and *Adam* [2] is used with a mini batch size of 64. A learning rate 0.001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ are used, and the proposed CNN is trained for up to 20,625 iterations when the error plateaus. In testing, we adapt the Euclidean loss $1/2N \sum_{i=1}^{N} \left\| x_i^1 - x_i^2 \right\|_2^2$, where x_i^1 and x_i^2 are the label and output of the CNN, respectively. The summary of the implementation and specifications of experimental platforms are explained in Tables VI and VII, respectively.

3) Evaluation of proposed preprocessing technique: We make another model whose input is an image that expresses the movements of evacuees with an HSV color model for a proper evaluation of a proposed preprocessing technique. The hue and saturation stands for the direction and speed of movement, and the value represents the congestion on the arcs, which implies the proportion of the number of evacuees to the capacity of the arc. Two types of images with or without arcs are prepared, and some examples are illustrated

[7] http://caffe.berkeleyvision.org

Table V: Summary of Network Structures

	layer 0	layer 1	layer 3	layer 4	layer 5	layer 6	layer 7	layer 8	layer 9	layer 10
Proposed CNN	I(50,75,5)	CR(4,0,20)	P(2)	CR(3,0,40)	P(2)	CR(3,0,60)	P(2)	CR(2,0,80)	FR(120)	FR(1)
HSV CNN	I(256,256,3)	C(5,1,20)	P(2)	C(5,1,50)	P(2)	FR(500)	F(100)	F(45)		

Table VI: Summary of Implementation

	Layer on HDAOS	Timing of Computation	Computation Time	Framework
LQF	Mid-term	Normal time	7–8 hours	
Training a proposed CNN	Mid-term	Normal time	4 minutes	Caffe [7]+ cuDNN
Prediction with a proposed CNN	Short-term	Emergency	3.71×10^{-4}seconds	Caffe + cuDNN

Table VII: Specifications of Experimental Platforms

Platform	CPU model(uArch.)	CPU freq.	GPU model	Memory	Calculation Opportunity
Huawei	E7-4890v2 (Ivy Bridge)	2.8GHz	NVIDIA Tesla K40m (not used for LQF)	2TB	LQF
Pascal	E5-2630v4 (Broadwell)	2.2GHz	NVIDIA Tesla P100 (used for CNN)	512GB	CNN (Training and Prediction)

in Figures 13 and 15. The top left LQF_T on each figure denotes the remaining time for the evacuation completion by seconds, which is given by the LQF algorithm, and the color wheel represents the directions, that is, the movement of evacuees in the right direction is illustrated by a red point.

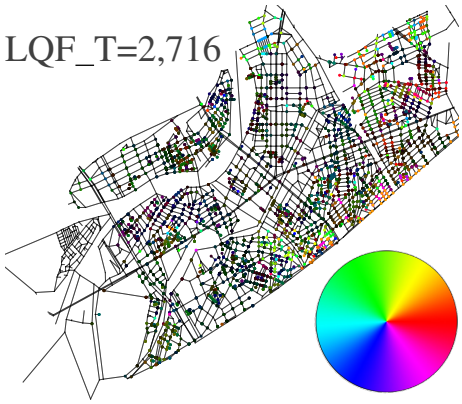

Figure 13: HSV Image on Initial Stage

To achieve high performance for image data, we use the classification model with a CNN, which have led a series of breakthroughs [13], [14]. We use a LeNet network [10] with slight modifications. This network is known to work well on digit classification tasks, and in our research the network predicts ECT by the minute as a classification task. We refer to this network as the HSV CNN. Details of the HSV CNN structure are shown in Table V. We use Stochastic Gradient Descent with a mini batch size of 64 and a learning rate of 0.000272 adjusted to deal with HSV images. We use a weight decay of 0.0005 and a momentum of 0.9, and the HSV networks are trained for up to 37,125 iterations.

The comparison of a proposed model and counterpart whose inputs are HSV images is illustrated in Figure 14.

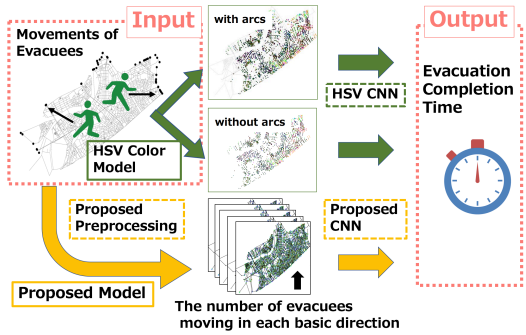

Figure 14: Comparison of HSV and Preprocessing Model

LQF_T and CNN_T denote the ECT given by the LQF algorithm and CNN, as explained in Table V. $error$ denotes the absolute difference of these, that is, $error := |LQF\_T - CNN\_T|$. If we use the percent sign for the $error$, the value is calculated by dividing $error$ by 2,751, which is the average ECT for the initial population distributions. We assume that LQF_T is correct, and CNN_T is the prediction. Some outputs of the proposed model are shown in Table VIII. The corresponding input data with the HSV color model are illustrated in Figure 15; however, we note that the real input data is the number of evacuees moving in each basic direction as shown in Figure 10.

The cumulative probabilities of $error$ with our preprocess technique and the HSV model are illustrated in Figure 16. We first analyze the prediction accuracy of flow10 showing the best performance. The probabilities that $error$ is less than or equal to one minute are 17% and 22% for using

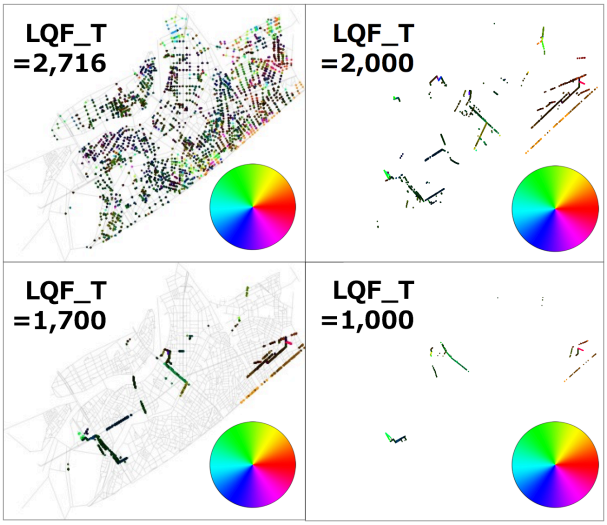

Figure 15: HSV Images With or Without Arcs in Various Stages

Table VIII: Outputs of Proposed Model for Figure 15

LQF_T	Proposed CNN	Error (%)
2,716	2,636	80 (2.9%)
2,000	2,036	36 (1.3%)
1,700	1,741	41 (1.5%)
1,000	1,038	38 (1.4%)

HSV images with or without arcs (black and yellow lines), and the probability for using our preprocessing technique is 95% (purple line). In other words, a proposed model predicts an almost accurate ECT with overwhelming probability. We also note that the model with our preprocess predicts ECT whose regression error is less than or equal to 86 seconds for all data of flow10, although the model using HSV images sometimes does not offer accurate prediction. It is very important for real application to limit the maximum regression error in an acceptable range, and the feature is observed in even the worst flow (blue line). Besides, the model with our preprocess technique predicts ECT whose regression error is less than or equal to two minutes with a high probability of 92% on average (red line).

4) Evaluation of prediction for several stages and flows: The left of Figure 17 shows the average error in each stage of evacuation when we utilize our preprocess technique. The prediction for the early stage of evacuation (2,400–3,000) is better than that for the final stage (0–600), which is desirable for real application because we can utilize the result for the evacuation plan more effectively if we know an almost accurate prediction for the early stage. We also note that the overall average error is only 55 seconds (all).

The right part of Figure 17 shows the average error for each flow when we utilize our preprocess technique. In the best case scenario, the average error is surprisingly

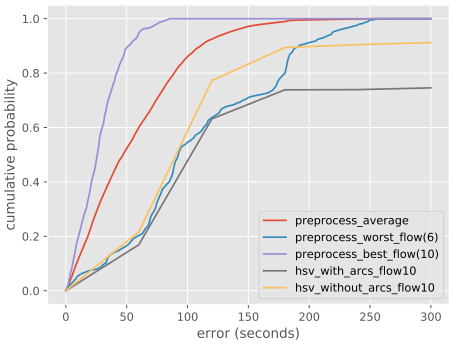

Figure 16: Cumulative Probabilities of Error With Our Preprocess Technique and HSV Model

small (flow10, 25 seconds); however, it is heavily dependent on the flows. The difference seems to be caused by the similarities between training and validation flows. Hence, we may prevent it by training the proposed CNN with wide varieties of flows.

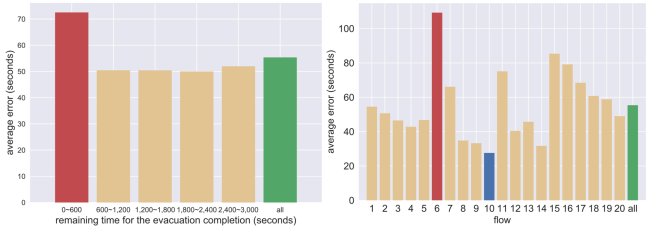

Figure 17:
Left: Average Error in Several Stages of Evacuation
Right: Average Error for Several Flows

VI. Conclusion

We propose a practical approach to evacuation planning with LQF and CNN according to HDAOS. We usually train a deep CNN by utilizing the results of the LQF algorithm, and the learned network predicts ECT instantly and accurately in emergencies, which avoids the extremely long computation time of the LQF algorithm. The efficiency and complexity of LQF is also found by applying the Hodge decomposition.

As ours is the first known effort to build such a model, we need to provide an effective evacuation planning in the near future. The Graph Convolutional Network (GCN) model [23], [24] is expected to be utilized for tackling the problem. We train a network with GCNs whose input and output are the population distribution and the number of evacuees moving on each arc, respectively. Consequently, we decide the effective movements of evacuees from the population distribution. We believe our efforts will help accelerate the pace of research on emergency evacuation planning.

ACKNOWLEDGMENT

This research was supported by the Panasonic Corporation, Japan Science and Technology Agency (JST), the Core Research of Evolutionary Science and Technology (CREST), the Center of Innovation (COI) Program, and JSPS Grant-in-Aid for Scientific Research (A) 16H01707.

REFERENCES

[1] Y. Sun, X. Wang, and X. Tang, *Deep Convolutional Network Cascade for Facial Point Detection*, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 23–28, 2013.

[2] D. P. Kingma and J. L. Ba, *A Method for Stochastic Optimization*, in The 3rd International Conference on Learning Representatations (ICLR), 2015.

[3] K. Fujisawa, T. Suzumura, H. Sato, K. Ueno, Y. Yasui, K. Iwabuchi, and T. Endo, *Advanced Computing & Optimization Infrastructure for Extremely Large-Scale Graphs on Post Peta-Scale Supercomputers*, Proceedings of the Optimization in the Real World –Toward Solving Real-World Optimization Problems–, Series of Mathematics for Industry, Springer, pp. 1–13, 2015.

[4] K. Fujisawa, T. Endo, and Y. Yasui, *Advanced Computing & Optimization Infrastructure for Extremely Large-Scale Graphs on Post Peta-Scale Supercomputers*, Proceedings of Mathematical Software, ICMS 2016, 5th International Conference Berlin, Germany, July 11–14, 2016, Lecture Notes in Computer Science 9725, Springer, pp. 265–274, 2016.

[5] K. Ueno, T. Suzumura, N. Maruyama, K. Fujisawa, and S. Matsuoka, *Efficient Breadth-First Search on Massively Parallel and Distributed Memory Machines*, Data Science and Engineering, Springer, Volume 2, Issue 1, pp. 22–35, 2017.

[6] K. Fujisawa, T. Endo, Y. Yasui, H. Sato, N. Matsuzawa, S. Matsuoka, and H. Waki, *Peta-scale General Solver for Semidefinite Programming Problems with Over Two Million Constraints*, The 28th IEEE International Parallel & Distributed Processing Symposium (IPDPS 2014), pp. 1171–1180, 2014.

[7] K. Fujisawa, T. Endo, H. Sato, M. Yamashita, S. Matsuoka, and M. Nakata, *High-Performance General Solver for Extremely Large-Scale Semidefinite Programming Problems*, The proceedings of the 2012 ACM/IEEE conference on Supercomputing, SC '12, 2012.

[8] E. Minieka, *Maximal, Lexicographic, and Dynamic Network Flows*, Operations Research, Volume 21, Issue 2 pp. 517–527, 1973.

[9] A. Takizawa, M. Inoue, and N. Katoh, *An Emergency Evacuation Planning Model Using the Universally Quickest Flow*, The 10th International Symposium on Operations Research and its Applications in engineering, technology and management, August 28–31, pp. 115–125, 2011.

[10] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner, *Gradient-Based Learning Applied to Document Recognition*, proceedings of the IEEE, Volume 86, Issue 11, November 1998

[11] D. Hubel and T. Wiesel, *Receptive Fields and Functional Architecture of Monkey Striate Cortex*. In: Journal of Physiology, Volume 195, Issue 1, pp. 215–243, 1968

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, *Backpropagation Applied to Handwritten Zip Code Recognition*, Neural Computation, Volume 1, Issue 4, December, 1989.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, The 25th International Conference on Neural Information Processing Systems, December 3–6, 2012.

[14] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 27–30, 2016.

[15] S. Ren, K. He, R. Girshick, and J. Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*, IEEE Transactions on Pattern Analysis and Machine intelligence, Volume 39, Issue 6, June 1, 2017.

[16] P. Fischer, A. Dosovitskiy, E. Ilg, et al.,*FlowNet: Learning Optical Flow with Convolutional Networks*, in The IEEE Conference on Computer Vision (ICCV), December 7–13, 2015.

[17] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha, *Unsupervised Deep Learning for Optical Flow Estimation*, proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017.

[18] D. Holden, J. Saito, and T. Komura, *A Deep Learning Framework for Character Motion Synthesis and Editing*, ACM Transactions on Graphics (TOG), volume 35, Issue 4, July, 2016.

[19] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, *What is the Best Multi-Stage Architecture for Object Recognition*, The 12th IEEE International Conference on Computer Vision, September 29– October 2, 2009.

[20] J. Duchi, E. Hazan, and Y. Singer, *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*, The Journal of Machine Learning Research, 12, 2121–2159, 2011.

[21] T. Tieleman and G. Hinton, *Lecture 6.5 - RMSProp*, COURSERA: Neural Networks for Machine Learning. Technical Report, 2012.

[22] X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye. *Statistical Ranking and Combinatorial Hodge Theory*, Math. Program., Ser. B, 127, 203–244, 2011.

[23] T.N. Kipf and M. Welling, *Semi-Supervised Classification with Graph Convolutional Networks*, The 5th International Conference for Learning Representations (ICIR), April 24–26, 2017.

[24] M. Defferrard, X. Bresson, and P. Vandergheynst, *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*, Advances in Neural Information Processing Systems 29, 2016.