electronics

MDPI

Article

Line Chart Understanding with Convolutional Neural Network

Chanyoung Sohn [1], Heejong Choi [1], Kangil Kim [1,*], Jinwook Park [1] and Junhyug Noh [2]

[1] Electrical Engineering and Computer Science Department & Artificial Intelligence Graduate School, Gwangju Institute of Science and Technology (GIST), Buk-gu, Gwangju 61005, Korea; chanyoungsohn@gmail.com (C.S.); hee_jong@gm.gist.ac.kr (H.C.); jinwookpark@gist.ac.kr (J.P.)
[2] Lawrence Livermore National Laboratory, Livermore, CA 94550, USA; jhroh86@gmail.com
* Correspondence: kangil.kim.01@gmail.com

Abstract: Visual understanding of the implied knowledge in line charts is an important task affecting many downstream tasks in information retrieval. Despite common use, clearly defining the knowledge is difficult because of ambiguity, so most methods used in research implicitly learn the knowledge. When building a deep neural network, the integrated approach hides the properties of individual subtasks, which can hinder finding the optimal configurations for the understanding task in academia. In this paper, we propose a problem definition for explicitly understanding knowledge in a line chart and provide an algorithm for generating supervised data that are easy to share and scale-up. To introduce the properties of the definition and data, we set well-known and modified convolutional neural networks and evaluate their performance on real and synthetic datasets for qualitative and quantitative analyses. In the results, the knowledge is explicitly extracted and the generated synthetic data show patterns similar to human-labeled data. This work is expected to provide a separate and scalable environment to enhance research into technical document understanding.

Keywords: data generation; knowledge template; line chart understanding; neural networks

Citation: Sohn, C.; Choi, H.; Kim, K.; Park, J.; Noh, J. Line Chart Understanding with Convolutional Neural Network. *Electronics* **2021**, *10*, 749. https://doi.org/10.3390/electronics10060749

Academic Editors: George A. Tsihrintzis and Daniel Morris

Received: 23 February 2021
Accepted: 18 March 2021
Published: 22 March 2021

1. Introduction

Understanding the propositions in chart images is a basic task to understand technical documentation. For this task, a variety of problem settings and machine learning solutions have been proposed [1–5]. Because of the ambiguity in defining a standard of knowledge to extract from a chart, in most studies, the task is indirectly solved as part of a larger integrated task as image caption generation.

This end-to-end style of problem solving can hinder research in academia in finding optimally configured deep neural networks for chart understanding. For solve sequential tasks at once, many deep networks are successful, such as neural machine translation [6], compared with the conventional approach of dividing and conquering the integrated tasks [7,8]. This is not the only case observed in this specific area. Deep neural networks showed high-accuracy image classification by mitigating the drawbacks of decomposing feature extraction and abstraction [9]. Because of the impact of the end-to-end style of problem solving, many deep network researchers configure a whole architecture first and analyze its macroscopic behavior. However, if we do not sufficiently understand the properties of separate tasks, architecture configuration to find the optimal generalization, model capacity, connections, and the required input features for each layer are delayed because all the settings should be searched from scratch. The optimal settings for each task can be hidden because of the effects of merging all integrated tasks in the search.

To address this problem in this paper, we propose a problem definition for the explicit analysis of a chart image, provide an algorithm to generate supervised data, and share them (https://github.com/cy-sohn/LCUdataset_generator (accessed on 9 March 2021)). To the best of our knowledge, problem definition and shared data for understanding statements implied in a line chart have been rarely proposed for helping with microscopic

architecture design. We focused on understanding knowledge in line chart images from visual perspectives rather than text-mixed information, called line chart understanding (LCU) in this paper. In the proposed definition, we test well-known and simply tuned convolutional neural networks for image analysis [10]. They are configured for multitask learning [11,12] with various classification and regression subtasks to determine propositions and their numerical arguments. The contributions of this work are summarized as follows:

- proposing a definition of knowledge implied in a line chart;
- providing an algorithm to automatically generate input chart images with their labels;
- analyzing the properties of the task and data by applying well-known neural networks to synthetic and real datasets.

We note that the main contribution is defining LCU and providing synthetic data with an algorithm validated with human-labeled real data. The neural network configuration is just an example we use to provide easy-to-obtain performance and intuition about this task for readers.

In Section 2, we explain state-of-the-art works related to chart understanding, and in Section 3, we introduce the problem definition for specifying target chart images and the knowledge template. Section 4 describes the algorithm to generate synthetic data. Sections 5 and 6 show experiment setups and their results in the synthetic data and human-labeled real data. In Sections 7 and 8, we conclude and discuss future challenges.

2. Related Works

Deep-learning-based chart understanding has been proposed [13–16], but these works focused on estimating the positions of chart objects rather than understanding implied knowledge in a chart. References [1–4,17] introduced methods to extract data from a chart or to convert data to other forms. They correlated the recognized results to text and graphic information shown in technical chart images rather than extracting implied statements as LCU. Reference [18] introduced the object detection network for the evaluation of scientific plots. This work aimed to build a model to understand a horizontal bar graph by estimating its numerical attributes. LCU targets line charts and extracting implied propositions instead of estimating the numerical values. Chartsense [4] uses deep-learning-based classifiers to determine the chart type of a given chart image and extracts simple information from the chart image, which is an integrated task implicitly using part of the knowledge in a chart, even though there is no explicit knowledge-understanding module. Figureseer [5] recognizes texts using character recognition modules and parses them with plots together for re-designing various charts and applying them to question answering. The mainly discussion was the estimation of a regression form, rather than capturing knowledge in a logic form. Reference [3] proposed a similar method for understanding and redesigning a chart, but its targets are bar and pie charts. It omits a function to predict intents in a chart different compared with LCU. Chart image generation [19–22] may also include the chart understanding problem. Reference [19] proposed a method to generate line, bar, and pie chart images, but it is partially automatic so the scalability of data is limited for training data-driven models. PlotQA [20], FigureQA [21], and DVQA [22] provide data used for question answering. PlotQA [20] provides data using the three types of plot images: horizontal bar graph, line plot, and dot-line graph. Text appearing in the chart images consists of words in the document texts. Labels, grids, font sizes, tick labels, line styles, line colors, and legend locations are used as attributes of the chart. In our work, we set wider ranges for those attributes and used more data samples to express detailed local implications of a line. Slopes, positions, and the ranges of lines are also more expressive in LCU. LCU uses both human-labeled and synthetic data for evaluation to confirm the impact of the synthetic data as a test bed for real-world understanding. FigureQA [21] provides visual inference data consisting of more than a million pairs of questions and answers. It can express five types of plot types (line, dot-line, vertical and horizontal bar, and pie charts) and learn logic such as maximum, minimum, and smoothness. Similar

to PlotQA, these data have limited forms and attributes in line plots. Attributes such as title, label, tick, and axis label are fixed, and the shape and legend of the line are expressed differently for each plot. There are six questions fixed about the line plot in FigureQA that can be answered by yes or no. LCU has a wider variety of logic templates than FigureQA. DVQA [22] provides data for understanding bar charts. These data are not only applied to QA but also used for extracting numerical and semantic information. The targeted chart of this work is different than that of LCU.

3. Problem Definition for Line Chart Understanding

The goal of the LCU problem is to determine the propositions implied in a line chart image. Thus, an input image is given and we need to predict the most accurate labels representing the propositions and estimate their numerical arguments. In this section, we describe the targeted image conditions and propositions that compose a knowledge template.

3.1. Input: A Line Chart Image

A line chart has many diverse attributes [23]. To cover a wide range of graphic perceptions that humans understand [23–25], we set a variety of attributes as shown in Table 1. To obtain unbiased and diverse lines, we set the range of attributes as large as possible in a uniform distribution when generating a value for each attribute (the library used for generating lines: https://matplotlib.org (accessed on 1 January 2021)).

Table 1. Attributes of the frame of the targeted line chart image.

	Attribute	**Range**
title	name	up to 10 characters
	size	[5, 10] (font size)
	position	{top, down} × {left, center, right}
Axis	label	up to 10 characters
	label position	{center, none}
	label size	[5, 8] (font size)
	label color	{black, white}
	range	[0.0, 1.0]
	tick label	up to 10 characters
	tick label size	[4, 7] (font size)
	tick digit	two decimal places
	number of ticks	[3, 12]
Legend	label	up to 10 characters
	position	{upper, lower} × {right, left}
	border	{border, none}
	border color	{black, gray}
	background	4 colors
Line	type	4 types including sold and dotted type
	color	7 colors
	thickness	[1, 4] (line width)
Background	color	5 colors
	grid	{horizontal, vertical, both, none}
	plot area frame	{lower left, lower left & upper right}

In this problem, we focus on a single chart composed of at most two lines, because this is the first step to solve before we consider more complex charts. The target chart image follows these rules:

- An image has a line chart.
- A chart has at most two lines.

- All lines are continuous and have different colors.

This input setting is used to evaluate the basic functionality of understanding knowledge. It can be easily integrated with other practical downstream tasks in a multitask learning or fine-tuning manner. In addition to the rules, the target image uses a standardized chart frame as follows:

- The origin point is located at the left bottom.
- The range of each axis is [0,1] (a standardized range).

The conditions show that any statement assigned to this image is based on visual perspectives. For example, if a model predicts an optimum in this graph, it generates an X-coordinate in [0,1]. Then, the selected point is linearly transformed to the range determined by the attached numerical text labels without any additional process. This setting has the advantage of clarifying the effect of predicting combinations with tick labels and images when detecting knowledge determined purely by visual properties.

3.2. Output: A Knowledge Template

The knowledge template proposed in this paper is the set of propositions determined by classification and regression subtasks. It can also be interpreted as the set of discrete labels and related numerical arguments. The structure, labels, and ranges of labels of all the subtasks are shown in Figure 1. Depending on the objects contained in an image, the logics representing knowledge are categorized into chart, line, and partition groups. In the chart group, the superiority subtask determines which line is superior to the other line overall. If lines have a cross point, the superiority has a None label. The line group has three subtasks: number of partitions is used to recognize the number of segments in the line. We allow one to three contiguous partitions to imply different logics. The line segment in each partition can have an independent growth type label. Monotonicity is used to distinguish whether the slope of the line is positive or negative from the starting to the ending points of a line. If a clear monotonicity is not observed, the None label is assigned. The minimum and maximum are subtasks used to detect minimum and maximum real-valued XY-coordinates in a line, respectively. In the partition group, the range is used to estimate the X-coordinates used as the partition boundaries. Growth type determines the growth type of the line segment in each partition. Examples of input images for extracting the knowledge template are shown in Figure 2.

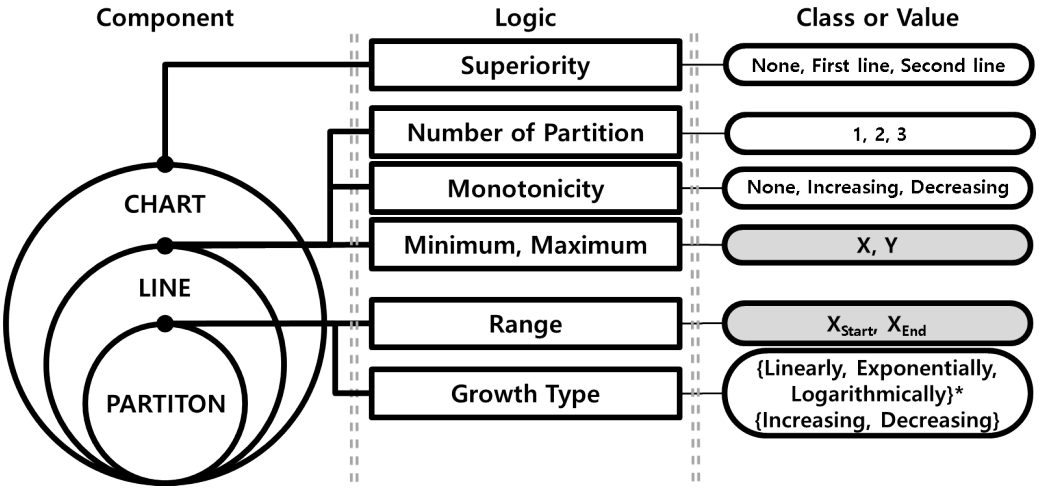


Figure 1. The structure of logic categories used as labels and their associated numerical ranges in the proposed knowledge template (white boxes in the third column are classification and grey boxes are regression subtasks).

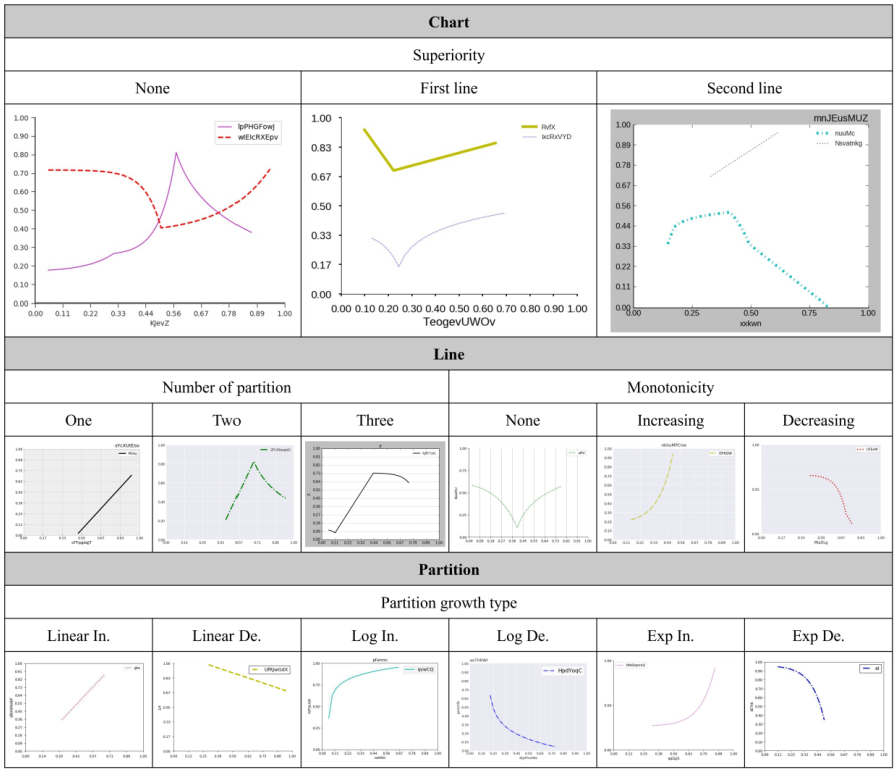


Figure 2. Examples of the generated input images and labels for classification subtasks (In., increasing; De., decreasing).

4. Data Generation

4.1. Algorithm to Generate Labeled Data

After generating the attributes for a chart image, lines are automatically generated for the selected labels of the subtasks. The whole process of generating lines and labels is shown in Figure 3 and Algorithm 1. In the overall steps, we select logics and their numerical arguments first, and randomly select data points to satisfy the selected labels.

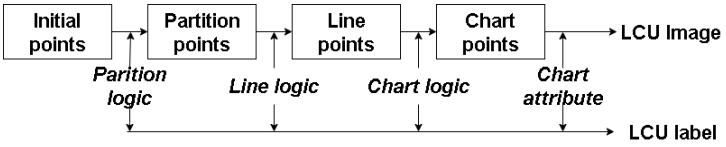


Figure 3. Flow chart of algorithm for data generation.

In the first step, the algorithm randomly generates two points used as the starting and ending point of a line. The points are in the range of 0 to 1. To determine the number of logics for a line in between the two points, the algorithms selects the number of partitions from {1,2,3} and then build partitions by randomly generating intermediate boundary points. Then, the growth type for each partition is randomly selected from the label set {linear, logarithmic, exponential}. After selecting a growth type for each partition, the form of the lines for the selected label is determined as

$$\textit{linear} \text{ label}: \quad y = \quad mx + b \tag{1}$$

$$\textit{logarithmic} \text{ label}: \quad y = \quad k \log(x - a) + b \tag{2}$$

$$\textit{exponential} \text{ label}: \quad y = \quad ke^{ax} + b \tag{3}$$

where x and y are the coordinates of a point; k, a, and b are the parameters to be tuned for drawing a line to pass all generated samples. The value for k is a randomly selected number in $[1, 5]$ for linear lines; m and b are approximated for generated data points using

the Python library. Data points are sampled at regular intervals on the X-axis. In the algorithm, the range of θ is in $[0.3, 2.9]$. In logarithmic and exponential functions, b and k are approximated to pass the initial points. The parameter a is initially fixed in $[2, 20]$ for the exponential function and $[0.85 \times X_{start}, 0.99 \times X_{start}]$ for the logarithmic function, where X_{start} is the X-coordinate of the leftmost initial points. The boundary conditions locate the lines into the first quadrant. The number of data points positioned in a partition is in the range of 10 to 50.

Algorithm 1 Generation of synthetic supervised data.

```
Randomly select the slope of line θ
Randomly select starting and ending points of a line with θ
Randomly select a label for the number of partitions
Randomly select the boundary X-coordinate of partitions
for all p do
    Randomly select a label for growth type
    Randomly select a line shape in the type
    Generate data points
    Draw a line in the range of p
end for
Determine labels for line-level subtasks
Determine labels for chart-level subtasks
Return (a chart, a set of labels) pairs
```

4.2. Detailed Settings for Label Generation

Categories and the range of outputs for each task are shown in Figure 1, which use the following specific configurations for their output. For the number of partitions, we assign the number of partitions to each line; therefore, the partition boundaries of lines are also independent. Growth type is independently assigned to each partition of each line. Superiority determines whether the first line is greater than the second line in the overall area. If a chart has only a line, this task is ignored in training. Label 1 means greater than the second in the overall area, 2 means the opposite case, and 0 means that it is too ambiguous. If the first line is greater than the second line, the minimum value of the first line is greater than or equal to the maximum value of the second line. Monotonicity determines a consistently increasing or decreasing state of a line in its all partitions. We set the label 1 for monotonic increasing, 2 for decreasing, and 0 for the inconsistent case. We set the labels by checking the sign and slope of generated lines. Minimum and maximum are regression subtasks to predict two points whose Y values are the minimum or maximum overall X-coordinates in a line, respectively. The growth type label is separately assigned to each partition of each line. Range is the subtask used to predict the meaningful partition boundaries composed of X-coordinates. In this subtask, the starting point S and ending point E on the X-axis are predicted. The total number of output variables to predict and their types are shown in Table 2. Superiority, monotonicity, growth type, and number of partitions are classification tasks and the others are regression tasks.

Table 3 shows the distribution of labels in the generated 75,000 samples.

Figure 4a,c shows the distributions of minimum and maximum points and mean X-coordinates of partitions. To visualize the distribution, 1000 images were sampled for each number of partitions, and the mean X-coordinates for the starting and ending points were plotted.

Table 2. Numbers and types of subtasks (the number of output variables to predict is doubled for two lines in all subtasks except superiority).

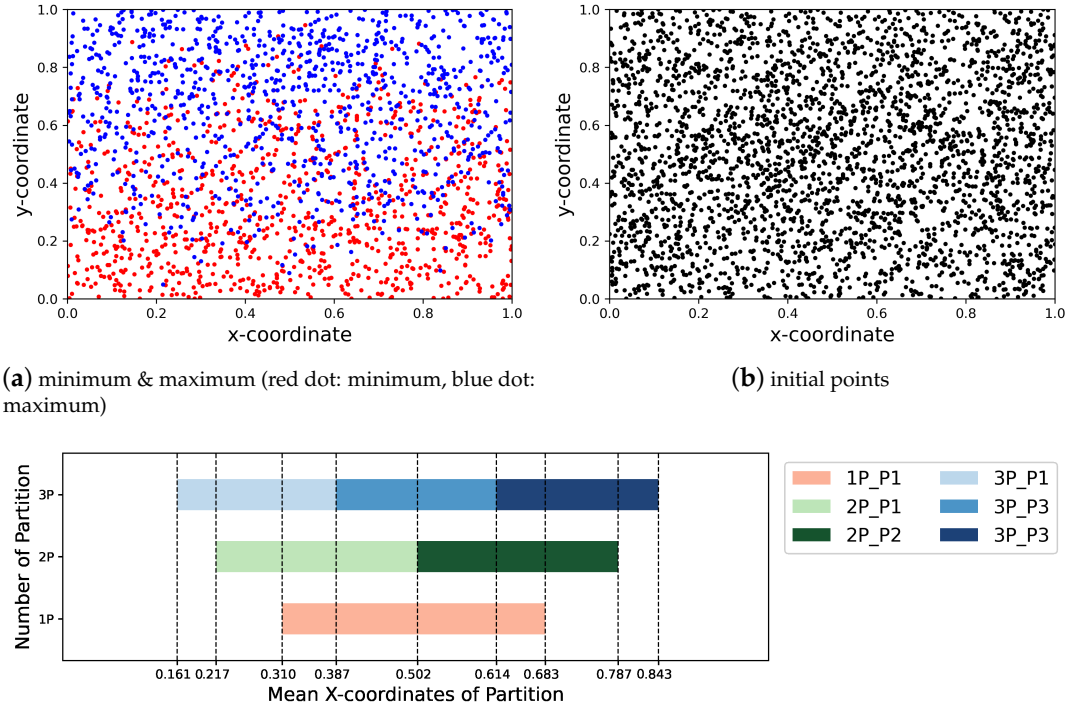
Category	Subtask	Number of Subtasks	Type
chart	Superiority	1	classification
line	Number of Partitions	2	classification
	Monotonicity	2	classification
	XY-coordinates for Minimum	4	regression
	XY-coordinates fpr Maximum	4	regression
partition	X of start & end for 1 partition case	4	regression
	X of start & end for 2 partition case	8	regression
	X of start & end for 3 partition case	12	regression
	Growth Type labels for 1 partition case	2	classification
	Growth Type labels for 2 partition case	4	classification
	Growth Type labels for 3 partition case	6	classification

Table 3. Proportion of labels in training data.

Subtask	Class	Proportion
Number of Line	1	49.92%
	2	50.08%
Number of Partitions	1	33.43%
	2	33.22%
	3	33.35%
Superiority	None	86.85%
	Line 1	6.77%
	Line 2	6.38%
Monotonicity	None	51.27%
	Increasing	24.42%
	Decreasing	24.31%
Growth Type	Linear Increasing	16.74%
	Linearly Decreasing	16.7%
	Logarithmic Increasing	16.57%
	Logarithmic Decreasing	16.45%
	Exponential Increasing	16.73%
	Exponential Decreasing	16.82%

4.3. Detailed Settings for Input Image Generation

The default resolution of a chart image is 100 dpi at a figure size of 640 × 480. The background color of the chart area is randomly selected except for black. The grid lines and the chart frame containing the axes are turned on or off. The direction of the lines is vertical, horizontal, or both. Text elements appearing on a chart can contain up to 10 uppercase or lowercase characters. This condition for text generation is equally applied to the chart title, X-axis label, Y-axis label, and line labels. The number of ticks in the chart is between 3 and 12 and represented with two decimal places.



(**a**) minimum & maximum (red dot: minimum, blue dot: maximum)

(**b**) initial points

(**c**) partitions (Each box represents a range of X-coordinate of partitions. *n*P_P*m*: X-coordinates of *m*th partition in *n* partition number case)

Figure 4. Distribution of randomly generated attributes. All these distributions show the large coverage of lines covered by the algorithm. (**a**) Minimum (blue dots and maximum (red dots) points. Lines are drawn to pass the minimum and the maximum. (**b**) The initially selected two points of a line. They were all randomly selected and the leftmost point is the starting point and the rightmost point is the ending point. (**c**) The distribution of X-coordinates of the boundaries. In each box, the X-coordinates are randomly sampled.

5. Experiments

The goal of the following experiments was to show the easy-to-obtain performance of well-known neural networks and their difference between human-labeled and synthetic test data. We note that proposing a novel and extensively optimized architecture was beyond the scope of this study.

5.1. Model Configuration

To evaluate an easy-to-obtain performance in this problem, we tested ResNet-50, Wide-ResNet-50-2, and Chart-Understanding-Spatial-Transformer-Network (CU-STN), as illustrated in Figure 5. ResNet-50 [26] and Wide ResNet-50-2 [27] were modified to leave the spatial information. The average pooling layer was replaced by the conversion layer (channel = 128, kernel = 3, and stride = 2). Their fully connected layer was also modified to fit the output size. CU-STN is a network configuration that was proposed to apply the spatial transformer network to the ResNet backbone resized for LCU. This network constructs a more robust network given the flexibility of the positions of the lines on a chart. The number of parameters for ResNet-50, Wide-ResNet-50-2, CU-STN is 26, 69, and 9 million, respectively.

5.2. Training Setting

The training loss is the sum of loss functions for 17 classification and 32 regression subtasks. We used cross-entropy for classification and average mean squared error for

regression. The problem types for each subtask are shown in Table 2. The total loss function $\mathcal{L}_{total}$ is defined as follows:

$$\mathcal{L}_{total} = \sum_{i \in S} \mathbb{L}(i)\mathbb{P}(i)\mathcal{L}_i, \tag{4}$$

$$\mathbb{L}(i) = \begin{cases} 1, & \text{if a line for the subtask } i \text{ exists} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\mathbb{P}(i) = \begin{cases} 1, & \text{if a partition for } i \text{ exists} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where S is the set of all subtasks and $\mathcal{L}_i$ is the ith subtask. Because the number of subtasks is dependent on the value of the selected line and the partition number, we used the indicator functions $\mathbb{L}$ and $\mathbb{P}$ to determine which subtasks to include in the total function. For monotonicity and superiority, ambiguity is very high and their proportion is not uniform as shown in Table 3. To remove the bias in training, we set the balancing parameters as shown in Table 4 multiplied with cross-entropy loss functions. The balancing parameter was set to the ratio of the inverse of the corresponding proportions. To investigate various behaviors with respect to the generated data size, we prepared four training data sets composed of 1000, 5000, 10,000, and 50,000 sample images. The detailed hyperparameter settings for training are listed in Table 4.

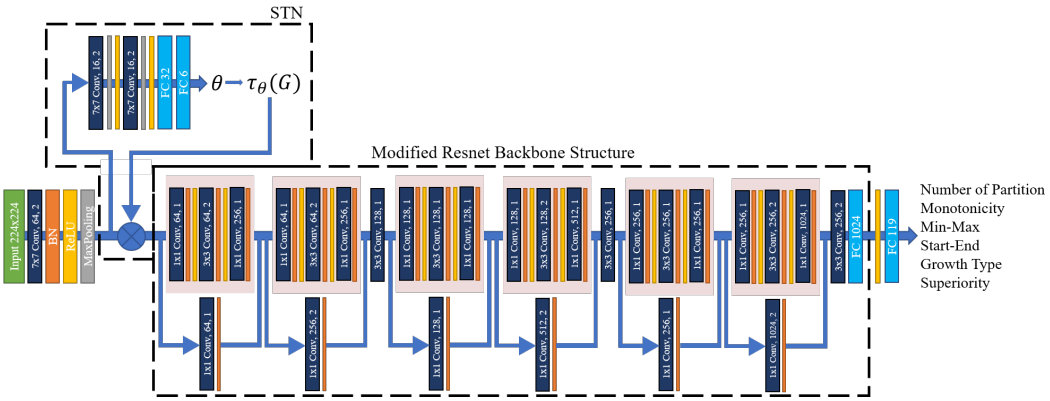


Figure 5. Architecture of CU-STN (θ: transformation parameter of STN for grid generator).

Table 4. Hyper-Parameter Settings (In.: Increasing, De.: Decreasing).

Dataset	Hyper-Parameter	Value
Common	batch size	64
	pretrained model	false
Training	validation ratio	0.5
	maximum update step	500,000
	optimizer algorithm	Adam
	optimizer hyperparameter (alpha, beta)	(0.9, 0.999)
	learning rate	0.001
	weight decay	0
	learning rate scheduler algorithm	ReduceLROnPlateau
	scheduler hyperparameter (patience)	$\min(\lfloor \frac{\text{total epochs}}{10} \rfloor, 10)$
	scheduler hyperparameter (factor)	0.1
Test	balancing parameters (monotonicity (None, In., De.)	(0.58, 1.21, 1.21)
	label weights for superiority (None, Line1, Line2)	(0.11, 1.40, 1.49)

5.3. Evaluation Setting

To evaluate performance, we prepared three test data sets composed of 500 synthetic images, 5000 synthetic images, and 500 human-labeled real images. The best validation model observed in training was used for test evaluation.

6. Result and Discussion

6.1. Quantitative Analysis

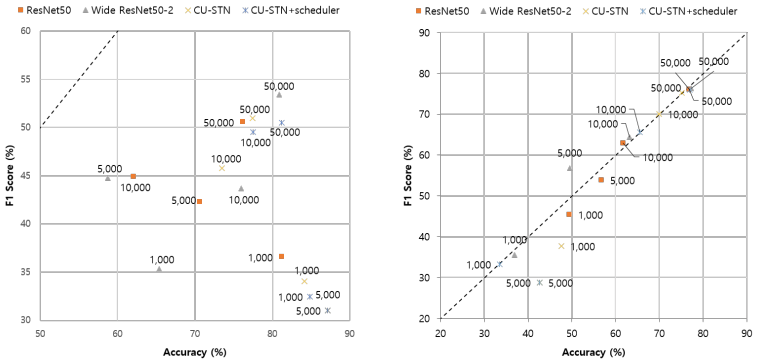
The accuracy and error results from 5000 synthetic test images are shown in Table 5. The growth type results are split to the three cases of number of partitions. The best results are displayed in bold text. Growth type per partition is more complex than the other tasks. This result may have been caused by the high ambiguity of the growth type values of short lines. The decrease in the accuracy was an expected pattern because the accuracy in each case is the percentage of the images that obtained the correct labels for all the partitions involved. Superiority is the simplest task. The estimation of partition boundary showed significant errors. Minimum and maximum estimation are more complex than the boundary estimation.

Table 5. Performance for all subtasks in synthetic test data (5000 samples; part., the number of partitions; mono., monotonicity; super., superiority; $|D_{tr}|$, size of training and validation data; MSE, mean square error; W-ResNet-50-2: Wide-ResNet-50-2).

		Classification Accuracy (%)						**Average MSE (10^{-1})**			
	$\|D_{tr}\|$	**Growth Type**			**Part.**	**Mono.**	**Super.**	**Range**			**Min & Max**
		p1	**p2**	**p3**				**p1**	**p2**	**p3**	
ResNet 50	1K	21.47	4.26	0.78	37.24	49.41	81.19	0.65	0.53	0.45	0.70
	5K	26.27	5.60	1.48	40.24	56.72	70.61	0.55	0.50	0.44	0.69
	10K	36.48	9.36	1.86	42.57	61.68	62.02	0.55	0.53	0.38	0.64
	50K	76.23	49.92	25.93	60.85	76.83	76.17	**0.24**	0.19	0.15	0.35
W-ResNet 50-2	1K	16.95	3.09	0.58	34.76	36.87	65.37	0.53	0.39	0.30	0.70
	5K	31.92	6.65	1.63	40.87	49.49	58.71	0.46	0.40	0.32	0.50
	10K	35.43	9.66	2.29	46.66	63.13	76.00	0.35	0.24	0.19	0.40
	50K	**78.85**	**52.80**	**30.75**	**64.24**	**77.25**	80.87	0.25	0.20	0.16	0.36
CU-STN	1K	20.02	4.18	0.39	34.22	47.60	84.14	0.55	0.35	0.27	0.64
	5K	17.15	3.22	0.43	32.58	42.59	**87.16**	0.48	0.32	0.22	0.63
	10K	38.01	11.96	2.68	50.99	69.88	73.51	0.38	0.24	0.19	0.39
	50K	71.71	46.28	21.04	62.15	75.13	77.47	0.27	**0.18**	**0.15**	0.36
CU-STN+ scheduler	1K	16.26	2.93	0.74	34.52	33.51	84.87	0.49	0.33	0.23	0.63
	5K	17.15	2.97	0.23	33.28	42.59	87.16	0.47	0.32	0.22	0.62
	10K	37.09	10.79	1.86	45.31	65.51	77.51	0.35	0.25	0.20	0.39
	50K	69.45	37.42	13.63	59.25	76.91	81.19	0.25	0.19	0.16	**0.34**

According to Tables 6 and 7, the results varied but overall patterns of accuracy of subtasks were not significantly different between the human-evaluated data. For the superiority and monotonicity tasks, the proportion of labels is unbalanced compared to the other subtasks maintaining uniform distribution, so we additionally evaluated F1 scores in the small synthetic dataset, as shown in Figure 6. In the case of monotonicity, F1 scores were similar to the accuracy results, which implied that average recall was close to one rather than zero. Superiority showed a significantly lower F1 score compared with the accuracy, so the average recalls were also low. This difference was observed even in the high accuracy near 90%, which implied that the dominating labels had sufficiently large

precision and recall while the others did not. Because of the high ambiguity of labeling, this task has high problem complexity.



(**a**) superiority (**b**) monotonicity

Figure 6. Comparison of F1 score and accuracy of the (**a**) superiority and (**b**) monotonicity subtasks for synthetic data.

Table 6. Performance for all subtasks in human-labeled real test data (500 samples; part., the number of partitions; mono., monotonicity).

		Classification Accuracy (%)					**Average MSE (10^{-1})**			
	$\|D_{tr}\|$	**Growth Type**			**Part.**	**Mono.**	**Range**			**Min & Max**
		p1	**p2**	**p3**			**p1**	**p2**	**p3**	
ResNet 50	1K	46.88	5.56	0.00	35.04	44.53	1.05	0.64	0.52	1.15
	5K	36.60	22.22	0.00	31.39	53.28	0.56	0.59	0.42	1.41
	10K	50.45	16.67	7.14	26.64	69.71	0.84	0.61	0.42	0.82
	50K	85.71	**69.44**	42.86	81.02	89.05	0.49	**0.15**	0.13	0.69
W-ResNet 50-2	1K	21.43	2.78	0.00	20.07	36.86	0.81	0.31	0.25	1.30
	5K	38.34	5.56	0.00	43.43	67.15	0.64	0.40	0.23	0.96
	10K	65.18	19.44	7.14	28.10	82.48	0.75	0.33	0.16	0.73
	50K	**86.61**	66.67	**50.00**	79.93	90.15	0.49	0.20	0.14	0.58
CU-STN	1K	16.52	2.78	7.14	11.68	13.14	1.21	0.45	0.24	1.24
	5K	42.86	0.00	0.00	13.14	8.76	0.86	0.31	0.17	1.33
	10K	59.38	25.00	0.00	50.00	84.31	0.60	0.29	0.16	0.62
	50K	82.14	58.33	42.86	63.14	**93.07**	0.51	0.20	0.13	**0.56**
CU-STN+ scheduler	1K	20.98	2.78	0.00	5.84	68.61	0.76	0.28	0.17	1.30
	5K	42.86	0.00	0.00	**81.75**	8.76	0.78	0.27	0.16	1.29
	10K	50.45	19.44	0.00	22.99	70.07	**0.31**	0.25	**0.09**	0.63
	50K	72.77	36.11	35.71	45.98	90.15	0.61	0.21	0.18	0.62

Table 7. Performance of all subtasks with the synthetic test data (500 samples; part., the number of partitions; mono., monotonicity; super., superiority; $|D_{tr}|$, size of training and validation data, W-ResNet-50-2: Wide-ResNet-50-2).

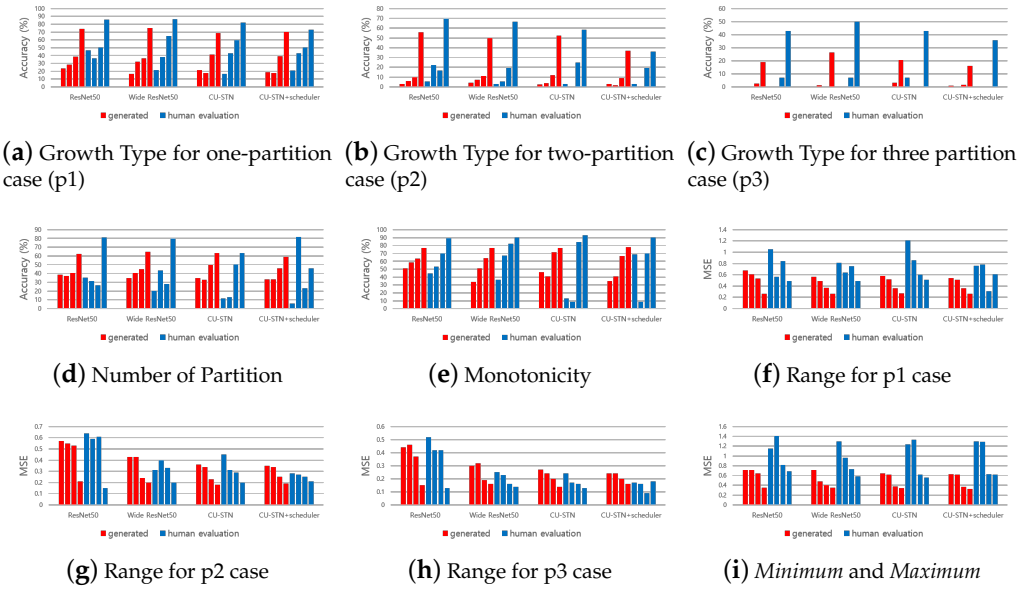
		Classification Accuracy (%)						Average MSE (10^{-1})			
	$\|D_{tr}\|$	Growth Type			Part.	Mono.	Super.	Range			Min & Max
		p1	p2	p3				p1	p2	p3	
ResNet 50	1K	23.79	3.19	0.41	38.65	51.35	83.75	0.68	0.57	0.44	0.71
	5K	28.23	5.98	0.41	37.43	58.78	68.75	0.61	0.55	0.46	0.71
	10K	38.71	9.96	2.49	40.68	63.51	61.25	0.53	0.53	0.37	0.64
	50K	74.19	**55.78**	19.09	62.43	76.76	78.75	**0.26**	0.21	0.15	0.35
W-ResNet 50-2	1K	16.53	4.38	0.00	34.86	33.92	68.75	0.56	0.43	0.30	0.71
	5K	32.26	7.17	1.24	40.68	50.95	56.67	0.49	0.43	0.32	0.48
	10K	36.29	11.16	0.41	45.00	63.92	73.75	0.37	0.24	0.19	0.40
	50K	**75.00**	49.80	**26.56**	**64.59**	77.16	80.83	0.26	0.20	0.16	0.35
CU-STN	1K	21.37	2.39	0.41	34.73	46.49	85.00	0.58	0.36	0.27	0.64
	5K	17.74	3.98	0.00	32.84	40.81	88.33	0.52	0.34	0.24	0.62
	10K	41.53	11.95	3.32	49.59	71.49	71.25	0.36	0.23	0.20	0.38
	50K	68.55	52.59	20.75	63.11	76.76	78.33	0.27	**0.18**	**0.14**	0.34
CU-STN+ scheduler	1K	18.95	3.19	0.83	33.38	35.00	85.42	0.54	0.35	0.24	0.63
	5K	17.74	1.59	0.00	33.51	40.81	**88.33**	0.51	0.34	0.24	0.62
	10K	39.11	9.16	1.66	45.68	66.49	76.25	0.36	0.25	0.20	0.37
	50K	70.16	37.05	16.18	58.92	**77.84**	80.42	0.26	0.19	0.16	**0.33**

Figure 7 shows the task-wise comparison results between human-labeled real and small synthetic data. Fluctuation patterns were similar in growth type estimation for one partition case. The two and three partition cases showed large difference, which were caused by the ambiguity shown in the quantitative analysis. The number of partitions, monotonicity, boundary estimation, and minimum and maximum regressions showed relatively similar patterns.

The validation results were also collected, as shown in Table 8.

In this setting, the ratio of validation and training samples was 1:1. The highest accuracy was recorded for growth type, partition confidence, monotonicity, and superiority. The lowest mean square error (MSE) values were recorded for range and minimum and maximum. As with the test, growth type and range were separately marked according to the number of partitions. The score was high because it was the best score recorded in each task during validation regardless of the total loss.

The overall results showed that simple CNN settings resulted in good performance on most subtasks, but a few tasks had low performance. The cause of this limitation is the ambiguity of labels in the data, because the rules for data generation with labeling were mainly based on human intuition. For example, determining linear or logarithmic in many images was challenging. Beyond the problem of ambiguous labeling, limits in machine learning perspectives remain. First, we used multitask learning framework, but learning all subtasks together may not be beneficial depending on their similarity.



(**a**) Growth Type for one-partition case (p1) (**b**) Growth Type for two-partition case (p2) (**c**) Growth Type for three partition case (p3)

(**d**) Number of Partition (**e**) Monotonicity (**f**) Range for p1 case

(**g**) Range for p2 case (**h**) Range for p3 case (**i**) *Minimum* and *Maximum*

Figure 7. Comparison of synthetic data (red) and human evaluation data (blue). Bars are the accuracy and MSE for classification and regression, respectively. Bars in the same color are the results of models trained with 1000, 5000, 10,000, and 50,000 samples from the leftmost side. Overall subtasks result show similar tendencies. Detailed numerical results of the tests are shown in Tables 6 and 7.

Table 8. Performance for all subtasks in synthetic validation data (tr., training; va., validation; part., the number of partitions; mono., monotonicity; super., superiority; $|D_{tr}|$, size of training and validation data; W-ResNet-50-2: Wide-ResNet-50-2).

		Classification Accuracy (%)						**Average MSE (10^{-1})**			
	$\|D_{tr}\|$	**Growth Type**			**Part.**	**Mono.**	**Super.**	**Range**			**Min & Max**
		p1	**p2**	**p3**				**p1**	**p2**	**p3**	
ResNet 50	1K	27.03	6.72	2.23	41.53	50.84	85.35	0.49	0.44	0.40	0.65
	5K	34.24	9.12	2.35	42.16	61.84	85.90	0.54	0.37	0.36	0.61
	10K	43.67	12.60	2.82	45.41	69.36	86.70	0.38	0.31	0.24	0.45
	50K	75.93	47.70	27.79	61.01	76.18	86.45	**0.24**	0.20	0.15	0.35
W-ResNet 50-2	1K	22.39	5.53	2.28	42.56	51.23	84.98	0.37	0.27	0.25	0.52
	5K	35.69	9.59	2.18	44.07	64.21	85.90	0.45	0.34	0.26	0.48
	10K	42.53	13.00	2.74	46.89	70.17	86.66	0.35	0.24	0.19	0.39
	50K	**77.02**	**52.49**	**32.30**	**63.07**	**76.59**	87.28	**0.24**	0.20	0.17	0.37
CU-STN	1K	24.71	4.74	2.30	39.71	52.13	84.98	0.41	0.30	0.25	0.58
	5K	18.44	4.17	0.97	34.14	42.90	85.90	0.48	0.32	0.22	0.62
	10K	477	12.24	2.70	49.25	72.41	86.85	0.38	0.25	0.20	0.38
	50K	73.37	45.74	24.13	60.75	75.64	87.63	0.26	**0.19**	**0.14**	0.35
CU-STN+ scheduler	1K	285	4.35	0.77	37.00	48.25	84.98	0.46	0.31	0.25	0.63
	5K	17.24	3.14	1.21	36.51	42.90	85.90	0.48	0.32	0.22	0.62
	10K	38.98	11.11	2.62	44.94	66.36	86.66	0.36	0.26	0.20	0.39
	50K	72.78	42.01	18.20	60.41	75.78	**88.17**	0.25	0.19	0.15	**0.34**

6.2. Qualitative Analysis

For Figure 8, we selected two sample images in the synthetic data for each number of partitions case from the test result from the synthetic data. Figure 8a,b shows the correct prediction results for growth type, and regression tasks still need improvement. In Figure 8c,d, some partitions are relatively well-predicted but the maximum and minimum values may be distant from correct points. Growth type labels are partially incorrect, but they are ambiguous even in human evaluation. In Figure 8e,f, partition and growth type values show large errors. In the accurate cases of prediction, we obtain somewhat understandable knowledge in human evaluation, but there are still errors that needs improvement in all tasks. Similarly, Figure 9 shows the prediction results on real test data. Compared with the synthetic data, we can see the natural language texts for labels, various ranges of real tick labels, and other practical attributes. The red bar and blue cross are the prediction results. The results in this data set are similar to those of the synthetic test dataset. Because the prediction is completely based on visual perspective, the prediction can be applied to practical images without the loss of generality.

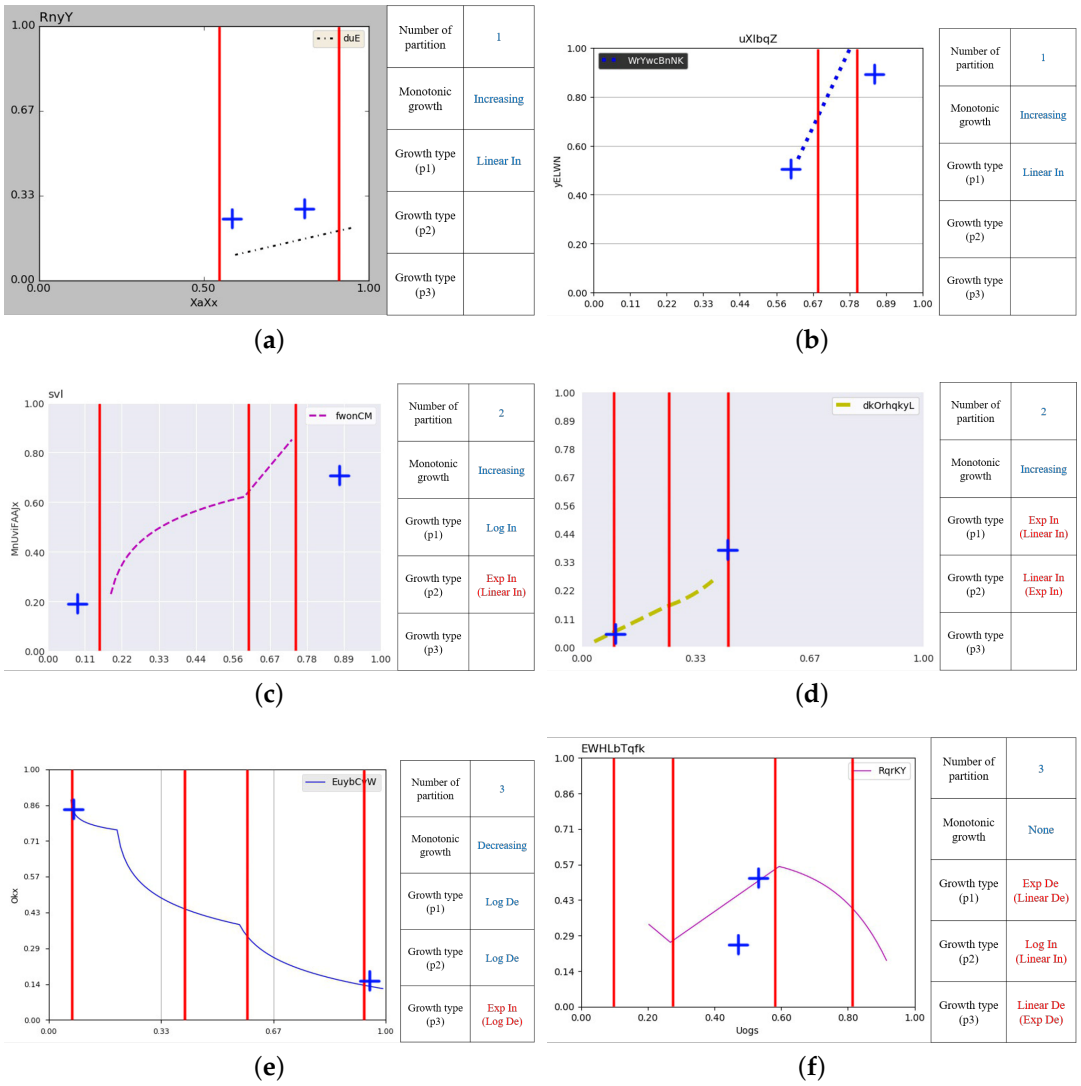


Figure 8. Example of detailed results on synthetic test dataset. Blue, correct prediction; red, wrong prediction; In, increase; De, decrease; blue cross, minimum and maximum point; red line:,partition boundary. These test data consist of one line chart, so superiority evaluation was excluded.

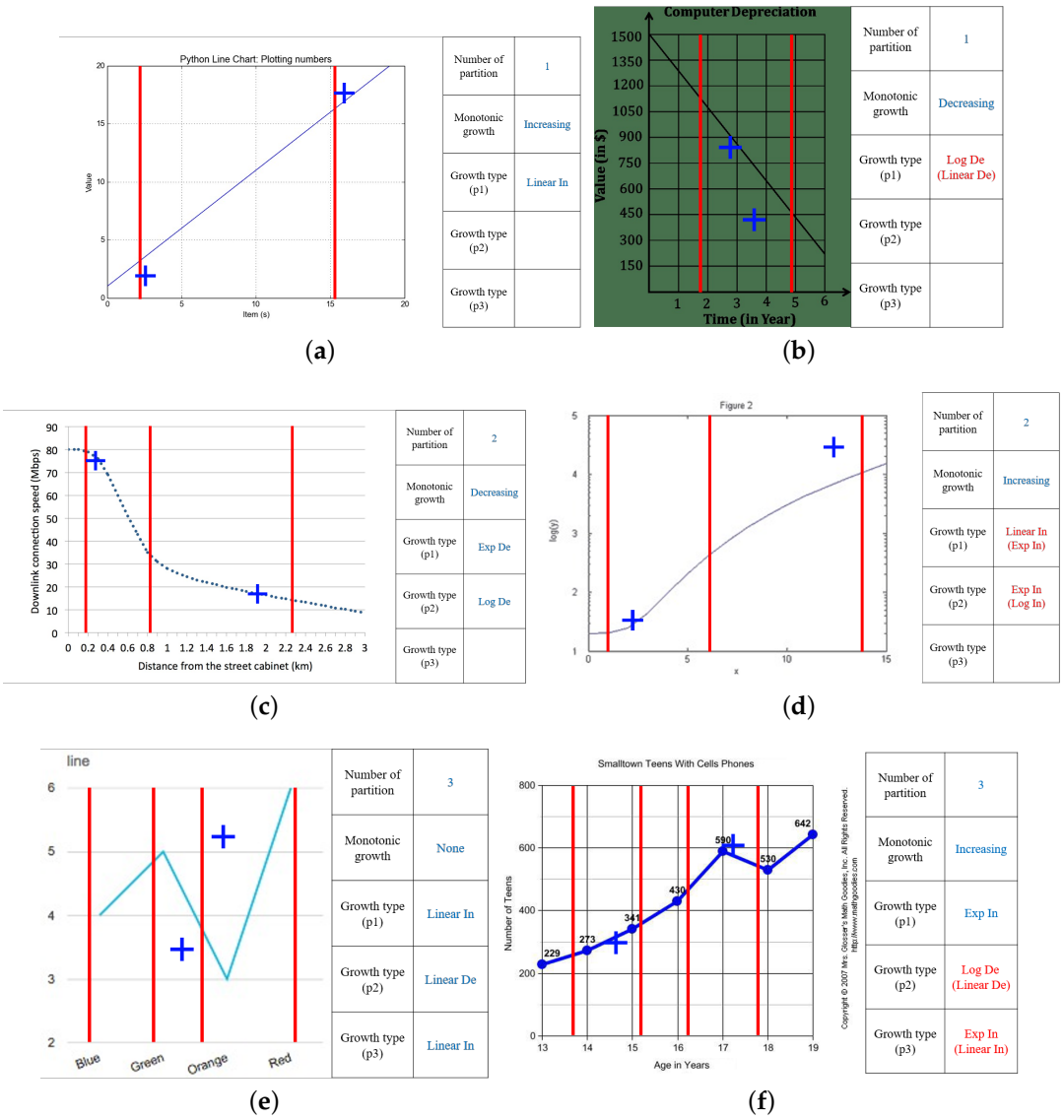


Figure 9. Example of detailed results with the human-labeled dataset. Blue, correct prediction; red, wrong prediction; In, increase; De, decrease; blue cross, minimum and maximum point; red line, partition boundary. These test data consist of one line chart, so superiority evaluation was excluded.

7. Conclusions

In technical document understanding, learning knowledge implied in a line chart is important, but it is conducted together with downstream tasks. This integration slows research on optimizing the configuration of neural networks used for understanding the knowledge. The explicit knowledge template proposed in this paper and the algorithm to automatically generate supervised data can be used as an incubating environment of models to solve the task. As an example of using the environment, we showed three configurations of convolutional neural networks and analyzed their performance and actual prediction cases. The synthetic data showed similar patterns to the human-labeled real data, showing that this environment can work for incubating models without a data-size limitation. This shared task is expected to boost research on the understanding of technical documents.

8. Future Works

In future work, the domain of applicable charts could be extended. We plan to more rigorously analyze the human evaluation results.

Author Contributions: Conceptualization, K.K. and C.S.; methodology, C.S. and H.C.; software, C.S., H.C., and J.P.; validation, C.S., H.C., and K.K.; formal analysis, C.S., H.C., and K.K.; investigation, C.S.; resources C.S. and J.P.; data curation, C.S.; writing–review and editing, K.K., J.P. and J.N.; writing–original draft preparation, C.S.; visualization, C.S. and H.C.; Supervision, K.K.; project administration, K.K.; funding acquisition, K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Global University Project (GUP) grant funded by the GIST in 2020, the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-01842, Artificial Intelligence Graduate School Program (GIST)), and the National Research Foundation of Korea (NRF) grant funded by Korean government (MSIT) (2019R1A2C109107712).

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

References

1. Balaji, A.; Ramanathan, T.; Sonathi, V. Chart-text: A fully automated chart image descriptor. *arXiv* **2018**, arXiv:1812.10636.
2. Mishchenko, A.; Vassilieva, N. Chart image understanding and numerical data extraction. In Proceedings of the 2011 Sixth International Conference on Digital Information Management, Melbourne, Australia, 26–28 September 2011; pp. 115–120.
3. Savva, M.; Kong, N.; Chhajta, A.; Fei-Fei, L.; Agrawala, M.; Heer, J. Revision: Automated classification, analysis and redesign of chart images. In Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, 16–19 October 2011; pp. 393–402.
4. Jung, D.; Kim, W.; Song, H.; Hwang, J.i.; Lee, B.; Kim, B.; Seo, J. ChartSense: Interactive data extraction from chart images. In Proceedings of the 2017 Chi Conference on Human Factors in Computing Systems, Denver, CO, USA, 6–11 May 2017; pp. 6706–6717.
5. Siegel, N.; Horvitz, Z.; Levin, R.; Divvala, S.; Farhadi, A. FigureSeer: Parsing result-figures in research papers. In Proceedings of the Computer Vision—ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016; pp. 664–680.
6. Sundermeyer, M.; Alkhouli, T.; Wuebker, J.; Ney, H. Translation modeling with bidirectional recurrent neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 14–25.
7. Hutchins, W.J.; Somers, H.L. *An Introduction to Machine Translation*; Academic Press: London, UK, 1992; Volume 362.
8. Koehn, P. *Statistical Machine Translation*; Cambridge University Press: Cambridge, UK, 2009.
9. Lee, H.; Grosse, R.; Ranganath, R.; Ng, A.Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 609–616.
10. Valueva, M.V.; Nagornov, N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Math. Comput. Simul.* **2020**, *177*, 232–243. [CrossRef]
11. Baxter, J. A model of inductive bias learning. *J. Artif. Intell. Res.* **2000**, *12*, 149–198. [CrossRef]
12. Thrun, S. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*; Morgan Kaufmann Publishers: Los Altos, CA, USA, 1996; pp. 640–646.
13. Kavasidis, I.; Pino, C.; Palazzo, S.; Rundo, F.; Giordano, D.; Messina, P.; Spampinato, C. A saliency-based convolutional neural network for table and chart detection in digitized documents. In Proceedings of the Image Analysis and Processing—ICIAP 2019, Trento, Italy, 9–13 September 2019; pp. 292–302.
14. Amara, J.; Kaur, P.; Owonibi, M.; Bouaziz, B. Convolutional Neural Network Based Chart Image Classification. In Proceedings of the 25th International conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2017), Primavera Congress Center, Plzen, Czech Republic, 29 May–2 June 2017; pp. 83–88.
15. Siddiqui, S.A.; Malik, M.I.; Agne, S.; Dengel, A.; Ahmed, S. Decnt: Deep deformable cnn for table detection. *IEEE Access* **2018**, *6*, 74151–74161. [CrossRef]
16. Saha, R.; Mondal, A.; Jawahar, C. Graphical Object Detection in Document Images. In Proceedings of the 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, Australia, 20–25 September 2019; pp. 51–58.
17. Huang, W.; Liu, R.; Tan, C.L. Extraction of vectorized graphical information from scientific chart images. In Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Curitiba, Brazil, 23–26 September 2007; Volume 1, pp. 521–525.
18. Ganguly, P.; Methani, N.; Khapra, M.M.; Kumar, P. A Systematic Evaluation of Object Detection Networks for Scientific Plots. *arXiv* **2020**, arXiv:2007.02240.

19. Huang, W.; Tan, C.L.; Zhao, J. Generating ground truthed dataset of chart images: Automatic or semi-automatic? In Proceedings of the Graphics Recognition. Recent Advances and New Opportunities, Curitiba, Brazil, 20–21 September 2007; pp. 266–277.
20. Methani, N.; Ganguly, P.; Khapra, M.M.; Kumar, P. PlotQA: Reasoning over Scientific Plots. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision, Snowmass Village, CO, USA, 1–5 March 2020; pp. 1527–1536.
21. Kahou, S.E.; Michalski, V.; Atkinson, A.; Kádár, Á.; Trischler, A.; Bengio, Y. Figureqa: An annotated figure dataset for visual reasoning. *arXiv* **2017**, arXiv:1710.07300.
22. Kafle, K.; Price, B.; Cohen, S.; Kanan, C. DVQA: Understanding data visualizations via question answering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 5648–5656.
23. Huang, W. Scientific Chart Image Recognition and Interpretation. Ph.D. Thesis, National University of Singapore, Singapore, 2008.
24. Cleveland, W.S.; McGill, R. Graphical perception: The visual decoding of quantitative information on graphical displays of data. *J. R. Stat. Soc. Ser. A* **1987**, *150*, 192–210. [CrossRef]
25. Cleveland, W.S.; McGill, R. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *J. Am. Stat. Assoc.* **1984**, *79*, 531–554. [CrossRef]
26. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
27. Zagoruyko, S.; Komodakis, N. Wide residual networks. *arXiv* **2016**, arXiv:1605.07146.