SKETCHTOCHART

Swati Singh[1], Rajat Kesarwani[2], Harshit Rana [3] and Ravi Goyal[4]

[1] Assistant Professor, Computer Science & Engineering Department
ABESIT, Ghaziabad, India

[2]Computer Science & Engineering Department, ABESIT
Ghaziabad, India

[3]Computer Science & Engineering Department, ABESIT
Ghaziabad, India

[4]Computer Science & Engineering Department, ABESIT
Ghaziabad, India

Abstract: ***Charts-a very familiar and core tactic used to represent data into digital format such as web pages, research papers and presentation slides. When the cardinal data isn't available, it's obligatory to snippet data from chart to make use of the data for further audit. In this paper, we introduce SKETCHTOCHART- a new approach to alter graph from real to digital world. In the path of finding solution to the issue of real time envision, we ensemble the network of CNNs for classifying, recognizing and restraining tasks that comprises of CNNs. Our architecture comprises of three layers which will extract the insight from the image and the output will be fed to D3.js to computationally create re-drawable and re-organizable pictorial data from information collected from real world.***

Keywords: Graph Visualization, Real-time Visualization, Deep Learning, D3.js, Convolutional Neural Network

1. INTRODUCTION

An early step in creating an application is to sketch a graph on paper blocking out the representation of its type. Employees face a challenge when using tools to manually make graphs for their presentation or data representation. This can be a bit time consuming or may require basic knowledge of the tool.

Problems in the domain of turning a design into some output have been tackled before: SILK [1] turns digital drawings into application code using gestures; DENIM [2] augments drawings to add interaction; REMAUI [3] converts high fidelity screenshots into mobile apps.

Our aim is to build an application which translates wireframe sketches directly into digital form. These application considerable benefits:

- Faster Iteration: a sketch can move to a digital image.
- Accessibility: allows anyone to create digital images of graphs.

Furthermore, we have identified that deep learning methods may be applicable to this task. Deep learning has shown considerable success over classical techniques when applied to other domains, particularly in vision problems [4, 5, 6 ,7, 8]. We hypothesize that a novel application of deep learning methods to this task may increase performance over classical computer vision techniques.

This task involves major challenges:

- Building both a deep learning and classical computer vision approach which can:
  - Adjust the layout to fix the human errors from sketching.
  - Translate detected elements into digital image.
  - Display the result to user in easy manner.
- Building a dataset of different graph sketches.
- Empirically evaluating the performance.

2. BACKGROUND

In this section we describe how the process works. After that we described why we used D3.js and further move on to describe Multilayer perceptron approach vs CNN approach which we use in our method.

2.1 D3

In this section we discussed what D3 is and why we used it.

D3 stands for Data-Driven Documents. It is an open-source JavaScript library developed by Mike Bostock to create custom interactive data visualizations in the web browser using SVG, HTML and CSS. With the massive amount of data being generated today, communicating this information is getting difficult.

D3.js is also great for visually customizing familiar charts (e.g., stacked bar charts, line charts, network maps, etc.) to meet the needs of more nuanced research questions and data. Users could, for instance, add particular graphical elements layered on top of a chart to communicate something further in their data. These elements also allow interactivity. Thus users could potentially explore their data-visualization in their own way.

Our research focused on converting sketch to digital chart which can be exported in form of images or scripts that can be used in user’s desired tasks. Here D3 fulfills our requirement. D3 takes input as a type of chart with pictorial data and output a chart that can be exported in various forms (for example SVG, Script, JPEG etc).

After our CNN perform all the operation on the captured image, the insights extracted from it will be fed to D3 to get our digital chart.

2.2 Convolutional Neural Network [CNN]

In deep learning, a convolutional neural network [9] (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.
Like an MLPs they consist of 3 layers as input, output, and multiple hidden layers. The hidden layers consist of:

- Convolutional layers - which apply a convolution operation to the input passing the result to the next layer, each convolutional neuron processes data only for its receptive field.
- Pooling layers - which combine outputs of neuron clusters at one layer into a single neuron in the next layer. The pooling layer serves to progressively reduce the spatial size of the representation in order to reduce the number of parameters and amount of computation in the network, which helps control overfitting.
- Fully connected layers - Connect every neuron in one layer with every neuron in another. These are placed at the end of the network.

CNNs are particularly useful over MLPs for image processing tasks as MLPs suffer from the curse of dimensionality due to the full connectivity between neurons and therefore do not scale well for high resolution images. Furthermore, MLPs don't take into account the spatial structure of data, so pixels on opposite sides of an image would be treated the same way as pixels close together. CNNs in the image domain use raw pixel data as input. The panning of filters (you can set the stride and filter size) in CNN essentially allows parameter sharing, weight sharing so that the filter looks for a specific pattern, and is location invariant — can find the pattern anywhere in an image. This is very useful for object detection. Patterns can be discovered in more than one part of the image. Disadvantages of MLP include too many parameters because it is fully connected. Parameter number = width x depth x height. Each node is connected to another in a very dense web — resulting in redundancy and inefficiency.

2.3 Multi Layer Perceptron v/s CNN

In the section we will discuss about MLPs and CNN and which one is good for sketchtochart.
Multilayer perceptron networks are a class of feedforward ANN which can distinguish data that is not linearly distinguishable. MLPs consist of multiple neurons arranged in layers. All neurons from adjacent layers have connections between them, we say the layer is fully connected. Each connection has an associated weight. There are three types of layers:

- Input layer - the first layer in the network, no processing is done, simply feed information to the next layer which is the Hidden Layer.
- Hidden layers - There can be more than 1 hidden layers that sit between input and output layer. These perform computations and transfer information from the input layer to the output layer.
- Output layer - responsible for computation and transferring the result of these computations out of the network.
- When an MLP is training, input and output data pairs are fed into the MLP and an algorithm (back propagation [10]) updates the weights between neurons trying to model the computational transformation which transforms the input data into the output data.

2.4 Workflow

The workflow of SketchtoChart is been defined in the Figure 1. As, we can clearly see that the user first of all provide the picture which is been uploaded to the CNN1 & CNN2 as input where first CNN will determine that which type of graph it is. While second CNN detects the object in the image. The o/p of second CNN of is been uploaded to the third CNN where the characters and numbers are been detected. Now this is been given to D3 where the chart is been made according to the first and third CNN and now user can extract it.

3. Proposed Solution

In order to achieve this goal, we ensemble the network of CNNs for classifying, recognizing and localizing tasks that consists of CNNs. At the very first layer, we train CNN to classify the type of sketch into two broad divisions which are pie chart and bar graph. After this the output of first layer will serve as the base of deciding the next classifier which will segment away the numerical data from the pictorial data. The processing of input image is done to change the RGB image into B&W image so as to reduce the size of the filter of the CNN and hence thereby reducing the number of parameters that need to be estimated while training the CNN without compromising the accuracy.
Now, the second layer of the classifier will be feed with B&W image as input. This layer is a trained CNN that localize the area of the numerical data and segment it from the rest of sketch. This layer is been trained for the specific type of sketch image, that is been recognized by the first layer of the classifier.
The last layer of our classifier will be independent from the type of the sketch and only responsible for recognizing the numerical data in the pictorial/geometrical data. This layer will also be a CNN capable which recognizes handwritten, wavy as well as computer generated numerical data, thus giving the output numbers that will be further used to computationally create re-drawable and organizable pictorial data from the information obtained from the

physical world and hence acting as an interface between the physical world and the digital world.

Both the first and second layer of classifier is the CNN which is designed to classify and detect the tasks. While, the third layer of the classifier will help in recognizing the text and numbers. After, the result is been treated with D3.js to create the output where users can edit it as per their requirements and also able to download it in different-different formats.

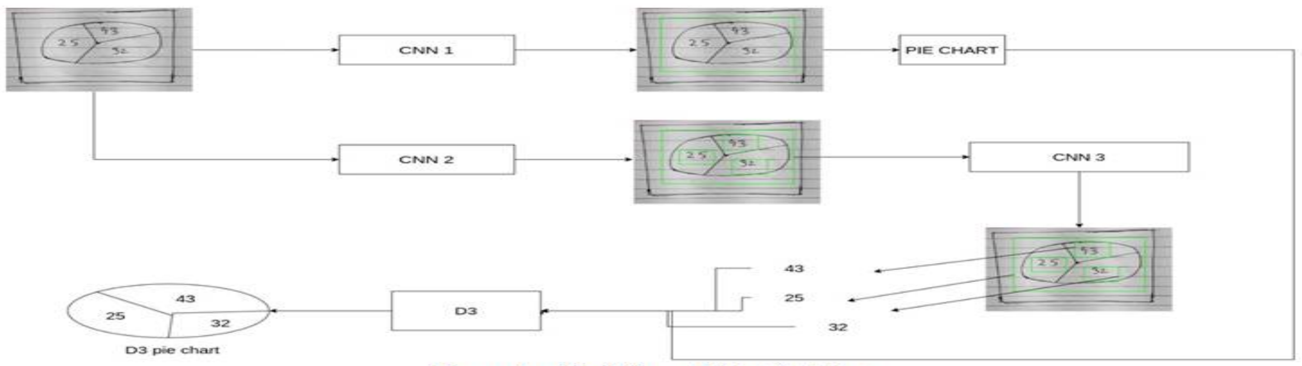

Figure 1 – Workflow of SketchtoChart

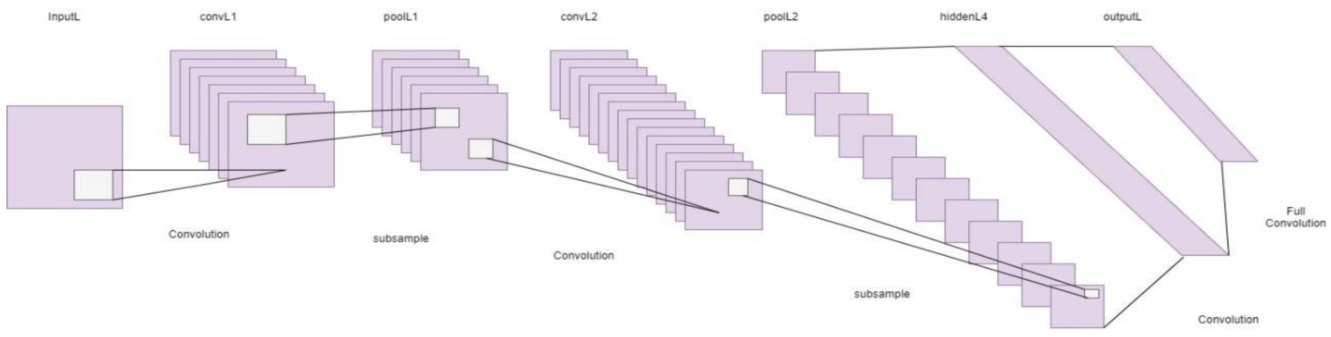

Figure 2 – First Layer [1st CNN]

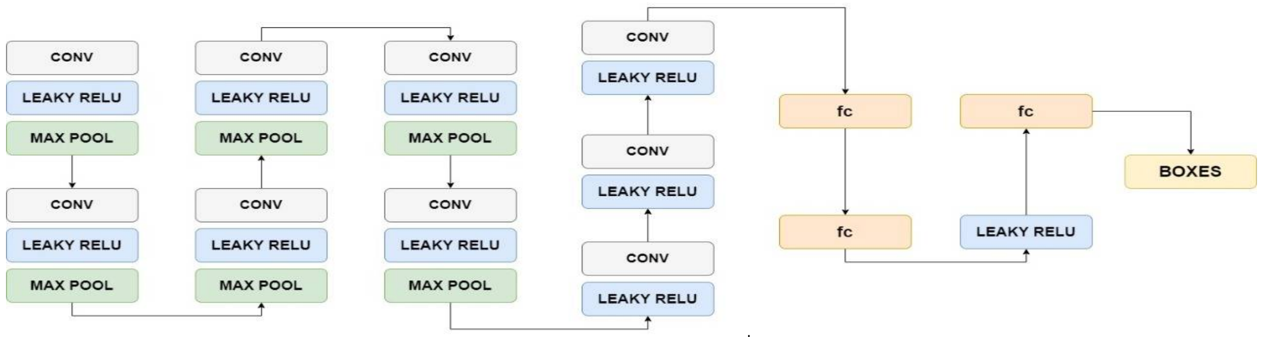

Figure 2 – Second Layer [2nd CNN]

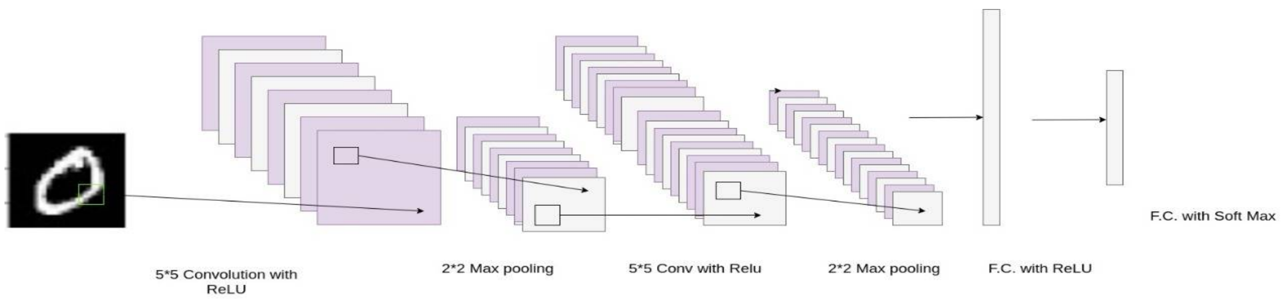

Figure 4 – Third Layer [3rd CNN]

3.1 First Layer [1st CNN]

In this CNN we have total 7 layers that is been designed for the classification of the data according to our needs:

Input:- The very first layer of network is responsible for capturing the input pattern of the image and for feedforwarding this input further into deep layers of the network.

Conv1:- Conv1 input layer performs the convolution operation into the input layer with a specified filter size.

Pool1:- pool1 layers is used to decrease dimension of image and preserve the redundant information from this.pool1 layer is a maxpool layer that performs a specific type of pooling , in which maximum value from a pixel window is captured and pass that further into the network.

Conv2:- conv2 input layer performs the convolution operation into the output of max pooling performed n pool1 layer with a same or different filter size.

pool2:- pool2 layers is used to decrease dimension of image and preserve the redundant information from this .This layer is also a maxpool layer like that of pool1 layer.

Hidden Layer: - The output of pool2 layer is flatten and feed into fully connected hidden layer.

Output layer: - The output of hidden layer is fed into softmax activation function and the output is returned.

3.2 Second Layer [2nd CNN]

In this CNN, we do object localization by using following three operational unit which are as follows:

- Conv (Convolution layer perform convolution operation by a specified filter size)
- Leaky ReLU (Output of the conv layer is then transferred to leaky relu activation function)
- Max Pool (max pooling operation is performed on the output of leaky relu activation function)

And all these three optional units performs two operations which are-

- Conv
- Leaky ReLU

The rest two layers are fully connected layers after that next layer consists of leaky relu activation function applied on output of last fully connected layer whose output is then fed into another fully connected layer with softmax activation function that gives us bounding boxes of areas of our interest.

3.3 Third Layer [3rd CNN]

In this CNN, we recognize the multi-characters in the image and also pictorial/geometrical information. This is been done by segmenting & patching the image. We also batch-up images in fully-connected layers to speed-up the process.

The dimensions of the network are similar to LeNet-5, which is an early CNN for hand-written digit recognition, though we use one less fully-connected layer. However, we use simpler yet more popular module to construct the network. For example, Leaky ReLU nonlinear function [2] is used at the output of the convolutional and fully-connected layers, which is much logical to compute than the sigmoid or the hyperbolic tangent functions and also proved to be trained faster with the same accuracy.

3.4 Oeuvre

Once, the CNN is been trained and information is been extracted from the sketch. The result is now handled with D3.js to create the output where the user can actually see it and can also download it in different formats as per their use.

4. CONCLUSION

The goal of this thesis is to create an application which will translate a sketched chart into digital image. We have presented an end to end framework which interpret delineation into digital version and produces the results on real time. This solution explains that how our framework has been developed which is easy to use. Simply, by allowing images from web or phone cameras and hosting the rendered website for collaboration. Then, with the use of common graph elements & limiting any training required.

Our evaluation shows that the approach we developed had high enough performance to be used in production environments. In particular, prior to our work we were not aware of any existing techniques to perform empirical evaluation on translating sketches into digital version while our procedure allows this. Further, we were not aware of any application of deep learning to this problem domain. We hope this dissertation as well as the release of proposed framework will stimulate further research into this domain.

5. FUTURE WORK

As based on our approach we can create an application to transform delineation into a digital graph. Approach showed significantly higher performance in element detection & classification with deep learning. Designing the network to perform detection, classification, and normalization may have decreased its performance. As such, by dividing these problems and using separate networks the performance in each of these tasks may increase. One design for this could be using a CNN such as Yolo [11] to perform detection & classification, and using another network to perform normalization. In addition to this, we can use Scene Text Detection and recognition [12] to handle the diversity and variability of text as well as complexity and interference of background and imperfect imaging condition. This can improve the application to represent the data delineate by the user on paper into a digital copy of it with much more accuracy and faster. Furthermore, models for other types of graphs can also be generated to improve flexibility and extend the limit of the application.

References

[1] J. A. Landay and B. A. Myers. "Sketching interfaces: toward more human interface design". In: Computer 34.3 (Mar. 2001), pp. 56–64. issn: 0018-9162. doi: 10.1109/2.910894

[2] James Lin et al. "DENIM: Finding a Tighter Fit Between Tools and Practice for Web Site Design". In: (Apr. 2000), pp. 510–517.

[3] T. A. Nguyen and C. Csallner. "Reverse Engineering Mobile Application User Interfaces with REMAUI (T)". In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). Nov. 2015, pp. 248–259. doi: 10.1109/ASE.2015.32

[4] Chao Dong et al. "Image Super-Resolution Using Deep Convolutional Networks". In: CoRR abs/1501.00092 (2015). arXiv: 1501.00092. url: http://arxiv.org/abs/1501.00092.

[5] Balakrishnan Varadarajan et al. "Efficient Large Scale Video Classification". In: CoRR abs/1505.06250 (2015). arXiv: 1505.06250. url: http://arxiv.org/abs/1505.06250.

[6] Oriol Vinyals et al. "Show and Tell: A Neural Image Caption Generator". In: CoRR abs/1411.4555 (2014). arXiv: 1411.4555. url: http://arxiv.org/abs/1411.4555.

[7] Andrej Karpathy and Fei-Fei Li. "Deep Visual-Semantic Alignments for Generating Image Descriptions". In: CoRR abs/1412.2306 (2014). arXiv: 1412.2306. url: http://arxiv. org/abs/1412.2306.

[8] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style". In: CoRR abs/1508.06576 (2015). arXiv: 1508.06576. url: http://arxiv.org/ abs/1508.06576.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: nature 521.7553 (2015), p. 436.

[10] Yann LeCun et al. "A theoretical framework for back-propagation". In: Proceedings of the 1988 connectionist models summer school. CMU, Pittsburgh, Pa: Morgan Kaufmann. 1988, pp. 21–28.

[11] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: CoRR abs/1506.02640 (2015). arXiv: 1506.02640. url: http://arxiv.org/abs/1506. 02640.

[12] Zhu, Yingying, Cong Yao, and Xiang Bai. "Scene text detection and recognition: Recent advances and future trends." Frontiers of Computer Science 10, no. 1 (2016): 19-36.
url: https://link.springer.com/article/10.1007/s11704-015-4488-0

AUTHORS

Swati Singh is an Assistant Professor, Computer Science & Engineering Department at ABES IT, Ghaziabad. Her primary area of interest is Mobile Computing.

Rajat Kesarwani is an under graduate Computer Science & Engineering Student pursuing B.Tech at ABES IT, Ghaziabad. His primary area of interest is Machine Learning.

Harshit Rana is an undergraduate Computer Science & Engineering Student pursuing B.Tech at ABES IT, Ghaziabad. His primary area of interest is Deep Learning.

Ravi Goyal is an undergraduate Computer Science & Engineering Student pursuing B.Tech at ABES IT, Ghaziabad. His primary area of interest is Machine Learning, Data Sciences.