Graphics Seminar Series

A presentation on paper title :

Sketch-R2CNN: An RNN-Rasterization-CNN Architecture for Vector Sketch Recognition

Authors : Lie Li, Changqing Zou, Youyi Zheng, Qingkun Su, Hongbo Fu, Chiew-Lan Tai

Venue : TVCG 2020 (IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS)

Presented by - Gaurav Rai

Problem :

Free-Hand sketching is an easy and quick means of communication because of its simplicity and expressiveness.

While we human beings have the innate ability to interpret drawing semantics, it is still a challenging task for machines.

The goal of sketch recognition (or classification) is to identify the object category of an input sketch, which is more challenging than natural image classification largely due to the inherent ambiguities, geometric variations and lack of rich texture details in the input.

Dataset :

- There exist two large-scale sketch datasets. The first one is the TU-Berlin dataset, which contains 250 object categories with only 80 sketches per category (i.e., 20K sketches in total). Each sketch in TU-Berlin was created within 30 minutes by a participant from Amazon Mechanical Turk.

- AMT is a web-based market where requesters can offer paid “Human Intelligence Tasks” (HITs) to a pool of non-expert workers. We submit 90 × 250 = 22,500 HITs, requesting 90 sketches for each of the 250 categories. In order to ensure a diverse set of sketches within each category, we limit the number of sketches a worker could draw to one per category. In total, we receive sketches from 1,350 unique participants who spent a total of 741 hours to draw all sketches).

- The second one is QuickDraw dataset which contains 345 categories with total 50M sketches. Sketches in QuickDraw are more abstract and contain fewer strokes than those in TU-Berlin.

Dataset :

Dataset	#strokes per sketch			#points per sketch		
	Median	Mean	Stdev	Median	Mean	Stdev
TU-Berlin	13.0	17.5	16.4	179.0	203.6	113.3
QuickDraw	4.0	5.1	3.8	47.0	52.9	29.0

Fig1. Statistics of the TU-Berlin and QuickDraw datasets after preprocessing: the number of strokes and the number of points per sketch.

Convolutional Neural Network (CNN) :

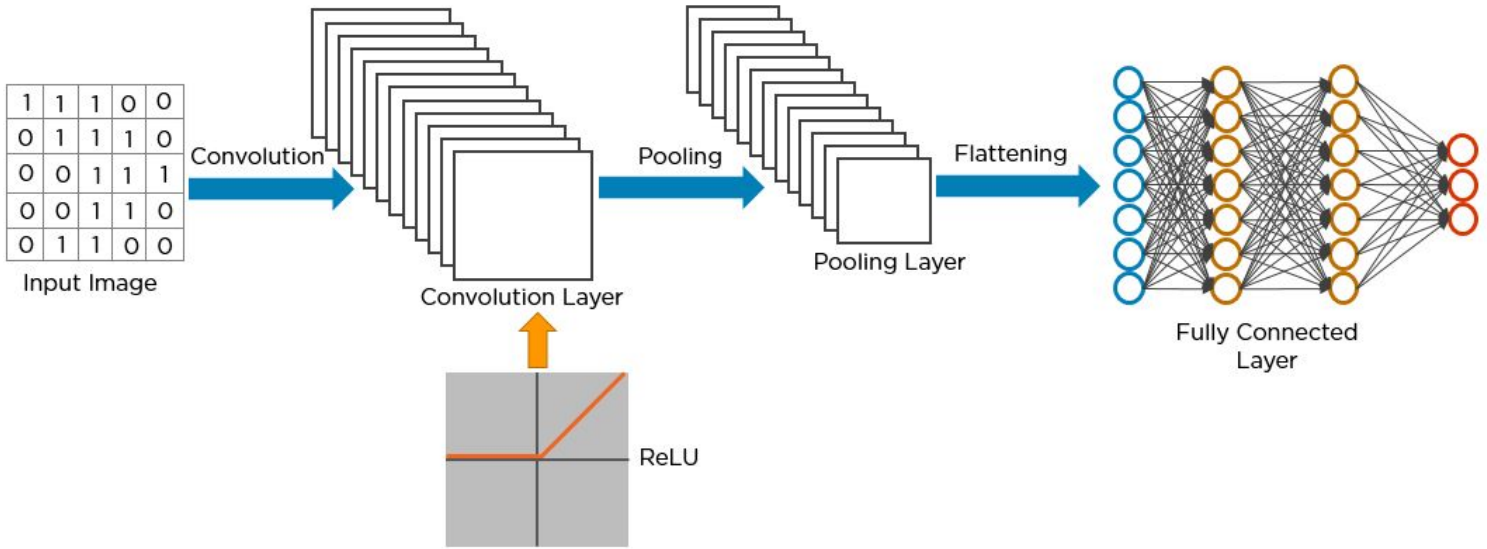

Fig2. Convolutional Neural Network (source: https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network)

Recurrent Neural Network (RNN) :

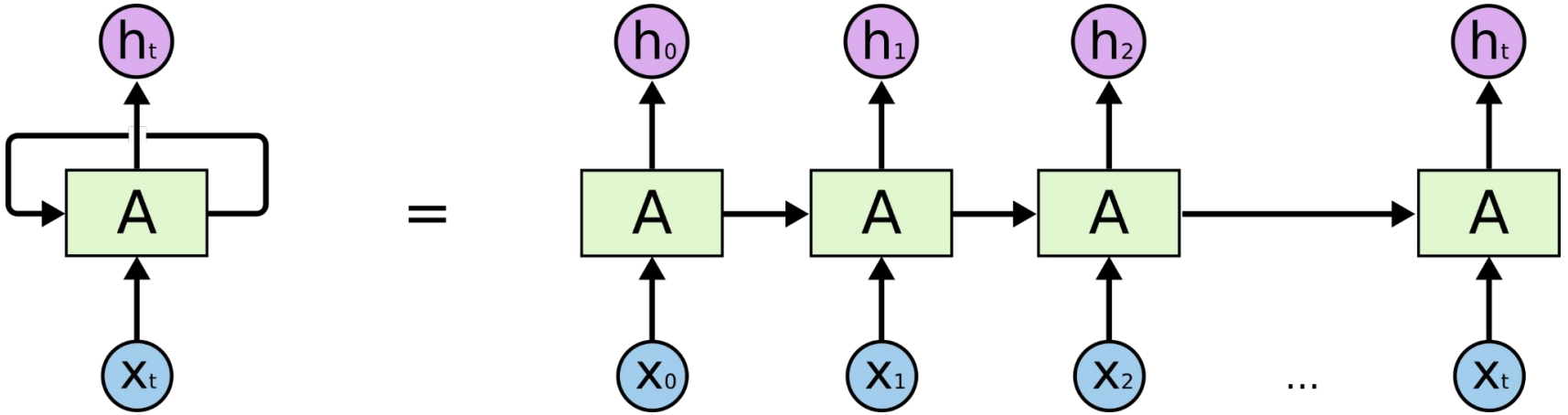

Fig3. An unrolled recurrent neural network (source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Related Work :

How Do Humans Sketch Objects? (SIGGRAPH 2012)

To recognize sketched objects, traditional methods generally take preprocessed pixel sketches as inputs. To quantify a sketch image, existing studies have tried to utilize various hand-crafted local features originally intended for photos. With the extracted features, classifiers (e.g., SVMs) are then trained to recognize unseen sketches.

The resulting recognition method is able to identify unknown sketches with 56% accuracy.

They perform a perceptual study and find that humans can correctly identify the object category of a sketch 73% of the time.

A Novel Sketch Recognition Model based on Convolutional Neural Networks :

(International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA) (pp. 1-6). IEEE. 2020)

#	Layer Name	Layer Parameters	Number of Parameters	Output Shape
0	*Input*	–	–	28x28x1
1	*Conv2D*	filters=16, kernel size=(3,3), activation=*ReLU*	320	28x28x16
2	*Conv2D*	filters=32, kernel size=(3,3), strides=(1,1), activation=*ReLU*	9,248	26x26x32
3	*Batch Norm.*	momentum=0.99	128	26x26x32
4	*MaxPooling*	pool size=(2,2), strides=(2,2)	–	13x13x32
5	*Dropout*	rate=0.4	–	13x13x32
6	*Conv2D*	filters=64, kernel size=(3,3), strides=(1,1), activation=*ReLU*	18,496	11x11x63
7	*Conv2D*	filters=128, kernel size=(3,3), strides=(1,1), activation=*ReLU*	36,928	9x9x128
8	*Batch Norm.*	momentum=0.99	256	9x9x128
9	*MaxPooling*	pool size=(2,2), strides=(2,2)	–	4x4x128
10	*Dropout*	rate=0.4	–	4x4x128
11	*Flatten*	–	–	2048
12	*Dense*	units=512, activation=*ReLU*	524,800	512
13	*Batch Norm.*	momentum=0.99	2,048	512
14	*Dropout*	rate=0.4	–	512
15	*Dense*	units=512, activation=*ReLU*	262,656	512
16	*Batch Norm.*	momentum=0.99	2,048	512
17	*Dropout*	rate=0.4	–	512
18	*Dense*	units=32, activation=*ReLU*	16,416	32
19	*Batch Norm.*	momentum=0.99	128	32
20	*Dropout*	rate=0.4	–	32
21	*Dense*	activation=*Softmax*	297	9

Fig4. The detailed information about each layer in the proposed model.

Deep Neural Networks For Sketch Recognition :

(arXiv preprint arXiv:1501.07873 1, no. 2 (2015): 3)

Ind	Type	Filter Size	Filter Num	Stride	Pad
1	Conv	16×16	128	3	0
2	ReLU	-	-	-	-
3	Maxpool	3×3	-	2	0
4	Conv	5×5	256	1	2
5	ReLU	-	-	-	-
6	Maxpool	3×3	-	2	1
7	Conv	3×3	512	1	1
8	ReLU	-	-	-	-
9	Conv	3×3	512	1	1
10	ReLU	-	-	-	-
11	Conv	3×3	256	1	0
12	ReLU	-	-	-	-
13	Maxpool	3×3	-	2	1
14	Conv	6×6	4096	1	0
15	ReLU	-	-	-	-
16	Dropout	Rate: 0.55	-	-	-
17	Conv	1×1	4096	1	0
18	ReLU	-	-	-	-
19	Dropout	Rate: 0.55	-	-	-
20	Conv	1×1	250	1	0

Fig 5. The architecture of sketch-DNN

Methodology :

To exploit the drawing cues in S, we resort to an RNN for analyzing the points of S sequentially and extracting features for each point .

To inform the CNN with the learned RNN features, we design a neural line rasterization (NLR) module that converts S with the per-point features to multi-channel point feature maps in a differentiable way.

The NLR module is the key enabler for connecting the two sub-networks that operate in completely different spaces. Compared to the pixel sketch input, the point feature maps are capable of delivering more drawing cues to the CNN.

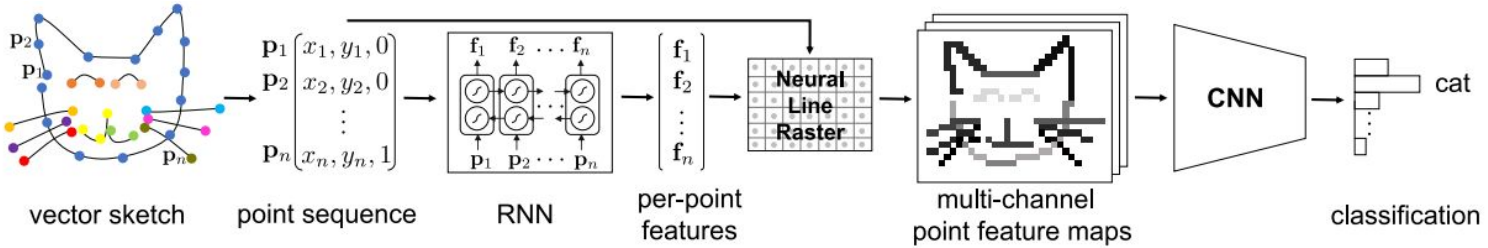

Fig 6. Illustration of our Sketch-R2CNN architecture for vector sketch recognition. A neural line rasterization (NLR) module is designed to convert per-point features, produced by the RNN with the input vector sketch, to multi-channel point feature maps, which are then consumed by an off-the-shelf CNN for recognition.

Methodology :

- Input Representation
- Network Architecture
- NLR

Experiments :

For each testing sketch from TU-Berlin or QuickDraw, recognition of its 25%, 50%, or 75% strokes (at least one stroke in a partial sketch) in the drawing order was performed.

Model	TU-Berlin			QuickDraw		
	25%	50%	75%	25%	50%	75%
SN v2 (reproduced) [2]	37.5	61.5	73.5	23.8	43.4	64.4
ResNet50 [17]	40.1	66.2	78.3	25.1	47.5	71.1
ResNet101 [17]	41.5	66.9	78.9	25.0	47.4	71.4
DenseNet161 [18]	41.8	67.6	79.5	**25.4**	48.1	71.7
Ours (w/ SN v2 reproduced)	38.6	62.8	74.7	23.7	44.8	67.4
Ours (w/ ResNet50)	42.5	68.5	80.2	24.8	48.4	73.1
Ours (w/ ResNet101)	43.2	69.1	80.3	24.8	48.2	73.2
Ours (w/ DenseNet161)	**44.1**	**69.7**	**81.2**	25.1	**48.7**	**73.5**

Fig 7. Partial sketch recognition accuracy (%) of our Sketch-R2CNN and its CNN-only counterparts on the TUBerlin and QuickDraw datasets. For a testing sketch, its 25%, 50% or 75% strokes in the drawing order (as partial sketches) were used for recognition.

Comparison Results:

- Sketch-R2CNN with ResNet-101 obtains the best performance (85.3%) on QuickDraw.
- Sketch-R2CNN with DenseNet-161 achieves the best performance (85.4%, 5.64K successes) on TU-Berlin.

Model	Accuracy	
	TU-Berlin	QuickDraw
Humans [1]	73.1	-
HOG-SVM [1]	56.0	-
Ensemble [51]	61.5	-
MKL-SVM [14]	65.8	-
FV-SP [13]	68.9	-
LeNet [52]	55.2	-
AlexNet-SVM [15]	67.1	-
AlexNet-Sketch [15]	68.6	-
SN v1 [21]	74.9	-
SN v2 [2]	77.95	-
SN v2 (reproduced) [2]	77.5	74.8
ResNet50 [17]	83.4	82.5
ResNet101 [17]	83.7	83.1
DenseNet161 [18]	84.2	83.0
Ours (w/ SN v2 reproduced)	79.4	78.4
Ours (w/ ResNet50)	84.5	84.8
Ours (w/ ResNet101)	85.0	**85.3**
Ours (w/ DenseNet161)	**85.4**	85.2

Fig 8. Recognition accuracy (%) of different methods on the TU-Berlin and QuickDraw datasets.

Limitations :

- Due to the abstract and textureless nature of sketches, RNNs may fail to extract descriptive point features to guide CNNs, leading to recognition failures (e.g., the crab).

- Sketches with seemingly ambiguous categories (e.g., the toaster or the pig) may also pose challenges to our method. It is expected that human would make similar mistakes on such cases.

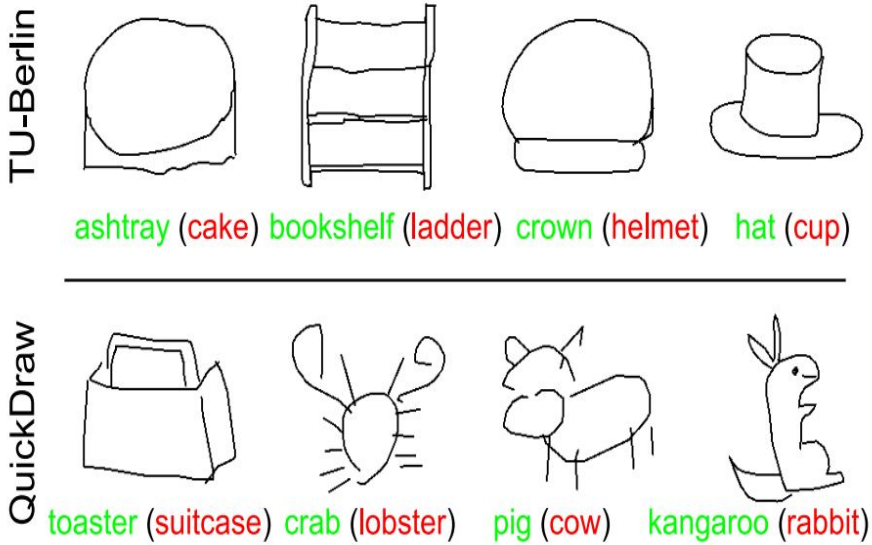

Fig 9. Recognition failures of our method. The green labels are correct predictions and the red labels are wrong predictions.[2]

Conclusion :

- Sketch-R2CNN brings consistent improvement over CNN baselines, especially on the million-scale QuickDraw dataset.

- RNN-Rasterization-CNN design allows CNNs to leverage the per-point features in vector sketches at early stages, which is enabled by a NLR module.

- The idea of in-network vector-to-pixel sketch conversion with NLR can be beneficial to other sketch-related tasks like sketch retrieval, sketch synthesis or sketch simplification.

References :

1. Li, L., Zou, C., Zheng, Y., Su, Q., Fu, H., & Tai, C. L. (2020). Sketch-R2CNN: An RNN-Rasterization-CNN Architecture for Vector Sketch Recognition. *IEEE transactions on visualization and computer graphics*.
2. Eitz, M., Hays, J., & Alexa, M. (2012). How do humans sketch objects?. *ACM Transactions on graphics (TOG)*, *31*(4), 1-10.
3. https://github.com/googlecreativelab/quickdraw-dataset

Thank you