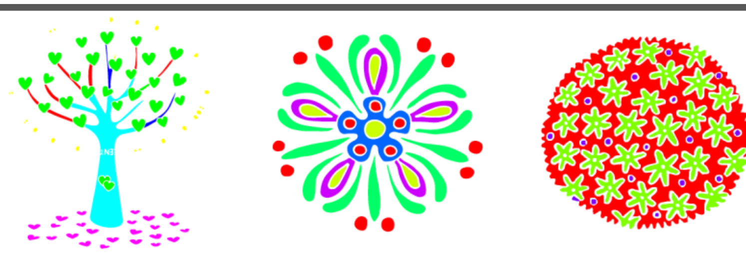
Learning to Group Discrete Graphical Patterns

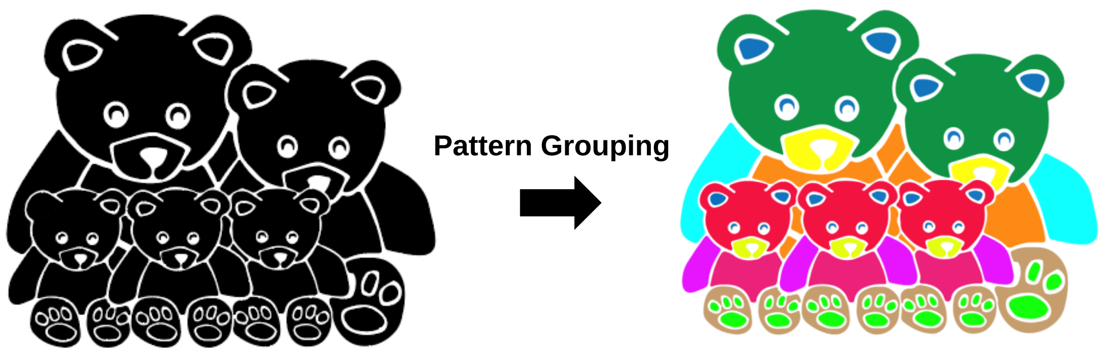


Zhaoliang Lun*[a] **Changqing Zou***[b] Haibin Huang [a] Evangelos Kalogerakis [a]
Ping Tan [b] Marie-Paule Cani [c] Hao Zhang [b]

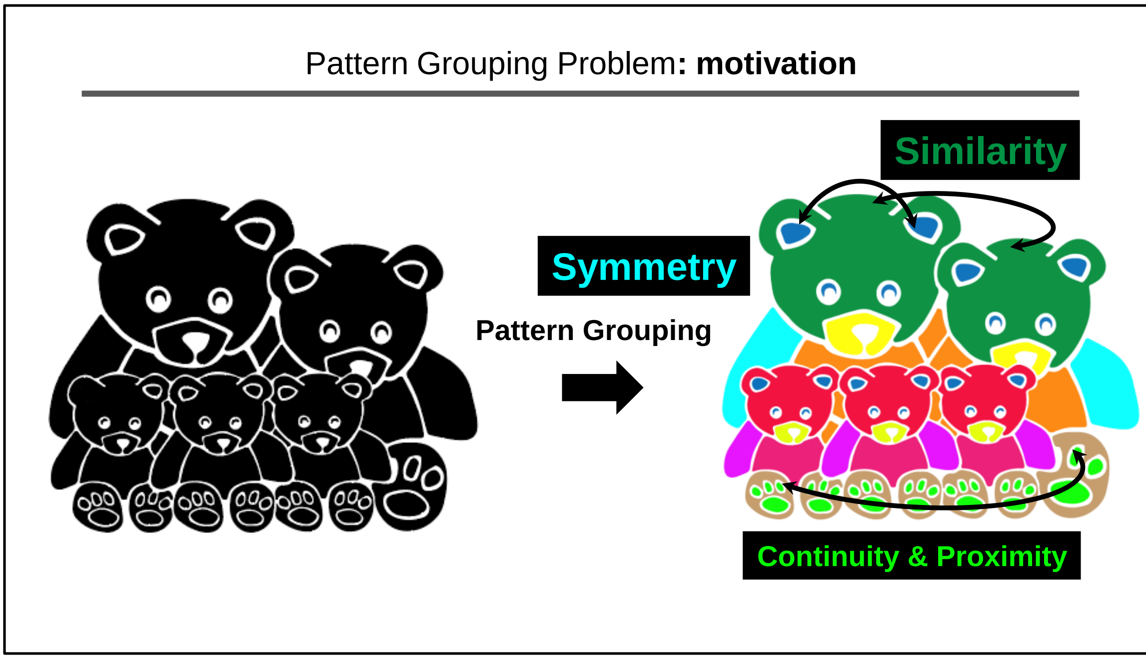
[a] UMASS Amherst [b] Simon Fraser University [c] Ecole Polytechnique

Thanks for the introduction. Good morning everyone, My name is Changqing Zou, In this talk, I will present our work on Grouping Discrete Graphical Patterns.

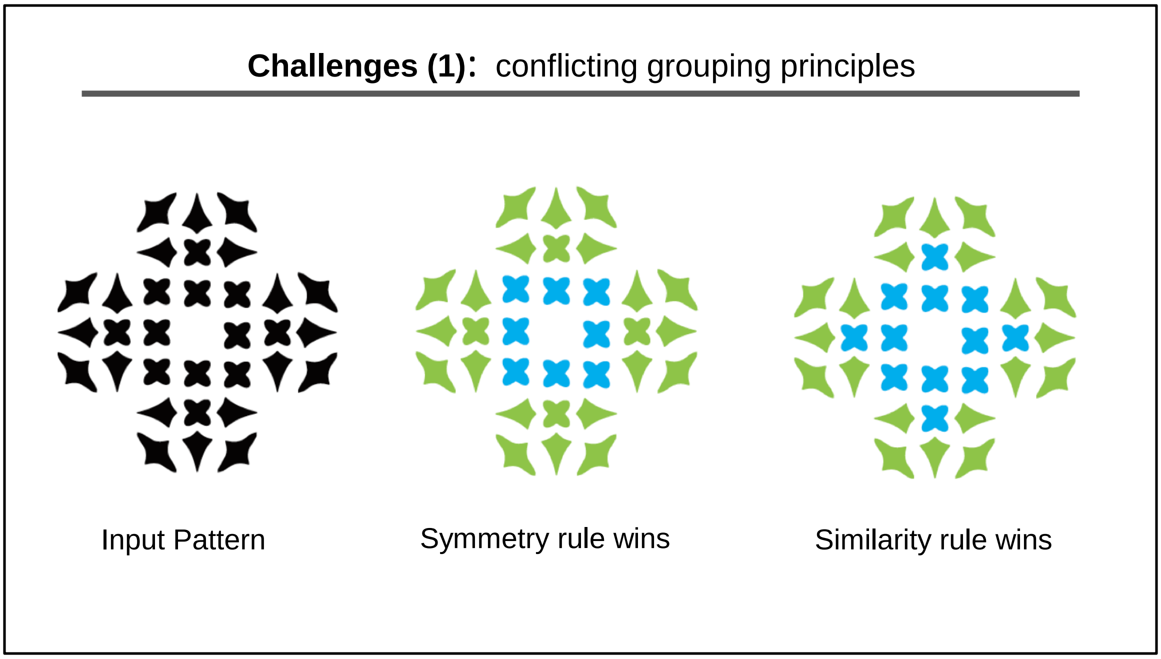
Pattern Grouping Problem: **motivation**



Given a set of pattern elements, we seek a grouping based on



criteria such as symmetry, similarity, continuity and proximity. in many cases these criteria are mixed and it is unclear how to select the most appropriate or how to properly weight their importance.

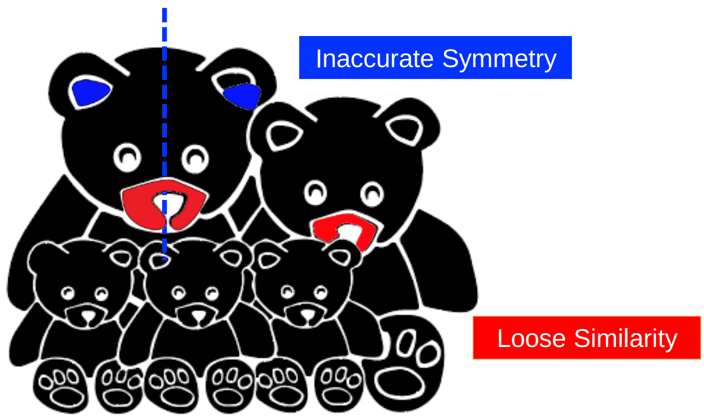


This problem is not easy, even in the case of 2D patterns, there are many challenging scenarios. Take this simple and regular pattern as an example, different perceptual grouping principles would lead to conflicting grouping results. It is unclear which grouping principles should take precedence.

Challenges (2): various noises

In real data, patterns are usually neither regular nor simple, often having various degree of noises. For examples,

Challenges (2): various noises



This pair of bear ears are not accurately symmetrical. And this pair of mouths are only roughly similar.

Challenges (3): Rich **Variations** and **Complexity**

There is also a challenge in the variations and complexity of the real data. People always say, no two leaves are exactly the same. This also happens to the real world cases we are looking at. However, we can still identify the symmetry patterns. We would like our algorithm to do the same thing. Despite its challenging, this problem is very useful. It can be used in many pattern related applications

Applications of Pattern Grouping

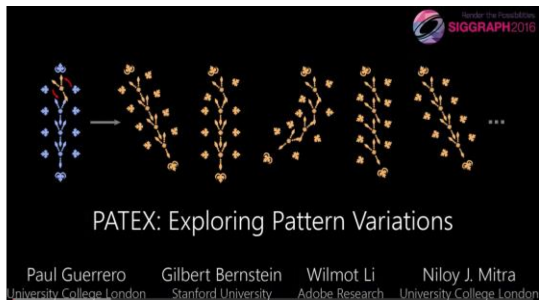
- **Pattern Editing**

Inverse Procedural Modeling by Automatic Generation of L-systems. O. Stava, et al. 2010

such as Pattern Editing, Pattern Exploration, and Layout Optimization

Applications of Pattern Grouping

- ❑ **Pattern Editing**
- ❑ **Pattern Exploration**

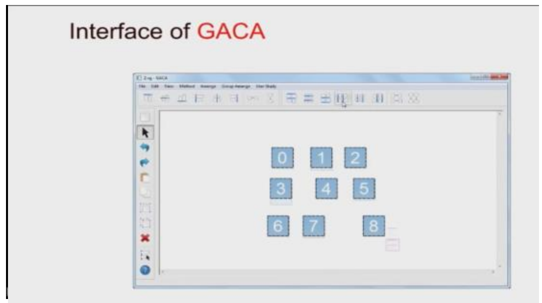


PATEX: Exploring Pattern Variations. P. Guerrero, et al. 2016

Pattern Exploration, and Layout Optimization where automatic pattern grouping will significantly lessen user's interactions

Applications of Pattern Grouping

- ❑ **Pattern Editing**
- ❑ **Pattern Exploration**
- ❑ **Layout Optimization**



GACA: Group-Aware Command-based Arrangement of Graphic Elements. P. Xu, et al. 2015

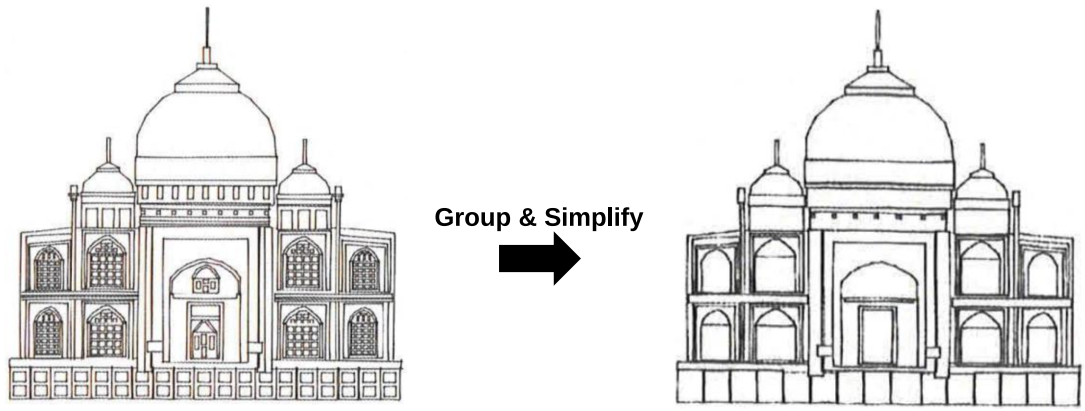
and Layout Optimization. We are not the first to propose this very useful problem

Related Work: **Model** & **Rule** Driven

- **Gestalt-based pattern grouping**
  - Conjoining Gestalt Rules for Abstraction of Architectural Drawings. Nan et al. TOG, 2011.
  - Perceptual grouping by selection of a logically minimal model, Feldman, ICCV, 2003.
  - The whole is equal to the sum of its parts: A probabilistic model of grouping by proximity and similarity in regular patterns, Kubovy & Berg. Psychological Review, 2008.
- **Symmetry-based pattern grouping**
  - Folding meshes: hierarchical mesh segmentation based on planar symmetry. Simari et al. SGP, 2006.
  - Co-Hierarchical Analysis of Shape Structures. O. Kaick et al. TOG, 2013.
  - Symmetry Hierarchy of Man-Made Objects. Wang et al. Computer Graphics Forum, CGF, 2011.
  - Layered Analysis of Irregular Facades via Symmetry Maximization. Zhang et al. TOG, 2013.

Actually, there are two major lines of work on this topic. One direction is to apply Gestalt rules, Another direction is to detect symmetries between elements.

Related Work: Gestalt-Based Pattern Grouping



Conjoining gestalt rules for abstraction of architectural drawings,
Nan et al. TOG, 2011.

The most relevant work is Nan’s SIGGRAPH project in 2011. which tries to quantify Gestalt rules in an energy-based optimization approach. This approach works well in grouping building façade patterns which have lots of perfect symmetries and regularities. But the problem we are trying to tackle in this paper has more noisy inputs.

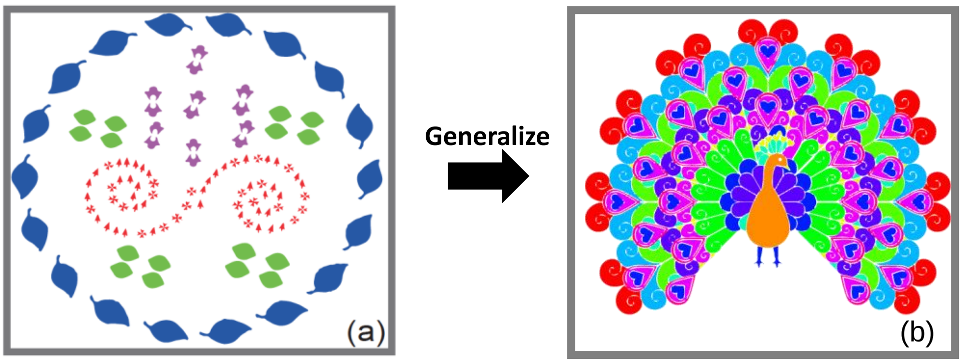
Nan's Strategy: model-driven

- **Hand-engineering** rules to quantify grouping models
- **Hand-tuning** relative importance of rules

The main characteristics of previous work focus on two aspects: coming up **hand-engineering** rules for the task, and **hand-tuning** the relative importance of rules. Unfortunately, this strategy is not robust to the noise. Taking this case for example, we will expect the elements forming the outer square being grouped together. But a direct use of Nan's strategy fails to achieve the goal.

Our Strategy

- Learning to group discrete graphical patterns from human annotations
- Loosely consider Gestalt principles
- Learn relative importance of features, without hand-engineer rules
- Robust noise handling thanks to learning approach



Instead, we propose a data-driven strategy. We don’t **hand-design** features and we don’t **hand-tune** the feature weights. We let the machine “see” many synthetic patterns with ground-truth grouping information. We expect a grouping strategy can be discovered automatically and can be generalized to real data.

Our Solution: **learn features for clustering**

- **Learned feature descriptor for each elements**
- **Clustering in learned feature space**

In a nutshell, we are trying to learn a feature descriptor for each element such that this feature descriptor is suitable for grouping. As long as we have established a feature space for the elements, any clustering strategy can be applied for the grouping task.

Our Solution: **learn features for clustering**

- **Learned feature descriptor for each elements**
- **Clustering in learned feature space**
- **Not optimize the clustering algorithm itself**
- **Learn a feature space suitable for clustering**

Again, I would like to emphasize that we are not optimizing the clustering algorithm itself. Our goal is trying to learn a feature descriptor for each element, or in other words we are trying to learn a representation space for those elements such that doing clustering in this space can yield better grouping results

Feature Learning: **how do human group**

Before teaching machine to learn grouping. Let's first **manually** do the grouping on this teddy bear. Then we can see if human experience could migrate to machine. Through this, we hope to find which learning model is most suitable for capturing each principle.

Feature Learning: **how do human group**

Similar & close-by

We can see these little toes in this teddy bear are similar and close to each other. It’s intuitive to group them together.

Feature Learning: **how do human group**

Horizontal Alignment

Also for the bodies of those 3 little bears, they are forming a horizontal alignment. Thus it also makes sense to make them a group.

Feature Learning: **how do human group**

How can we **migrate** human experience into machine learning?

For the two arms of the two big bears, although they are pretty far away, they show some kind of symmetry and can be group together. That is how we human will think about grouping. How can we migrate this human experience into machine learning?

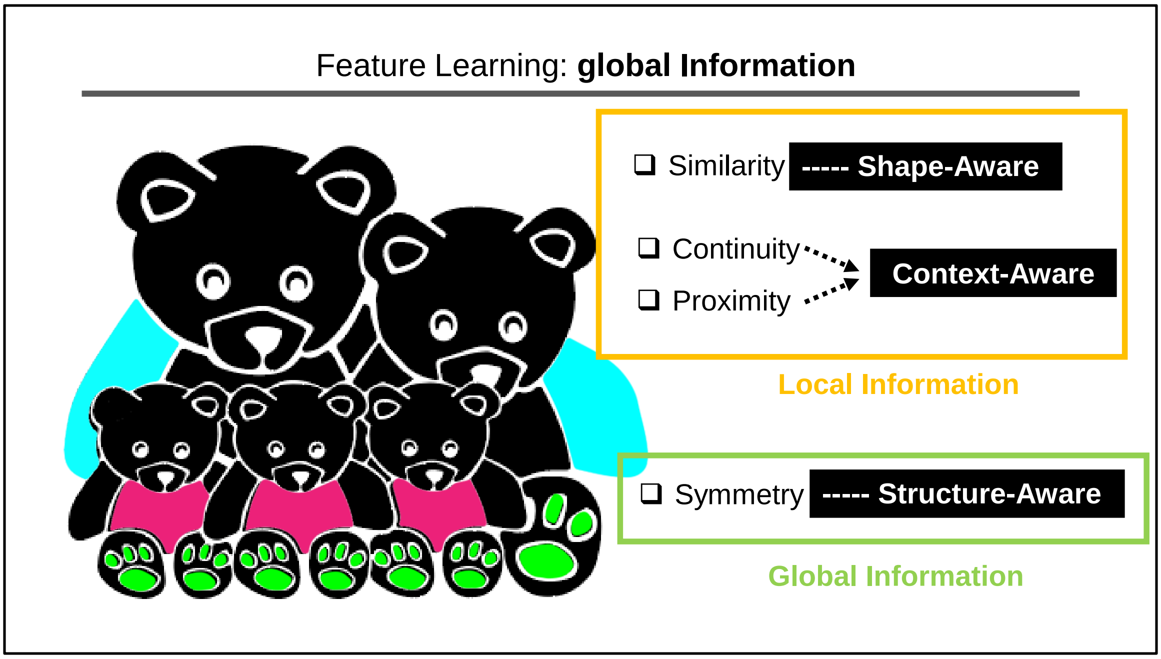
Feature Learning: **local Information**

- ❑ Similarity **----- Shape-Aware**
- ❑ Continuity
- ❑ Proximity

Context-Aware

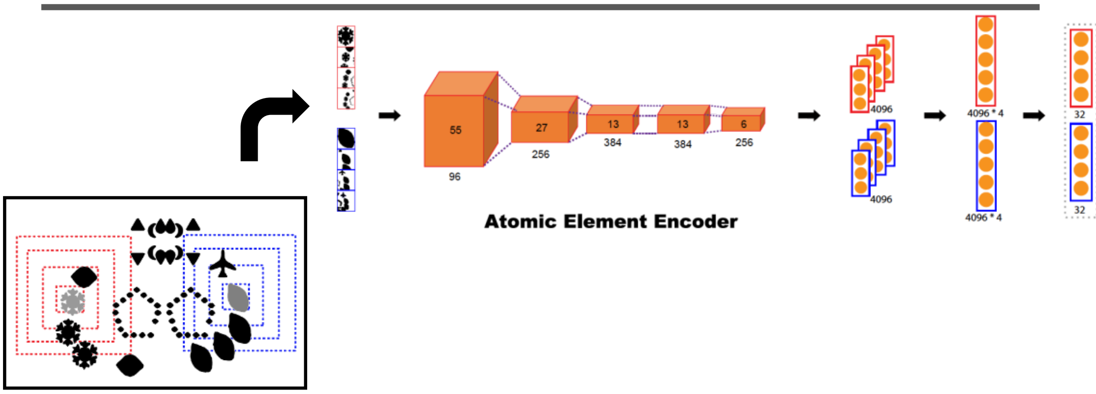
Local Information

We can see that the 'similarity' principle is related to element's shape while the proximity and continuity principle is related to element's context. They only capture the local information on individual elements so here we only need a model that can learn from local information



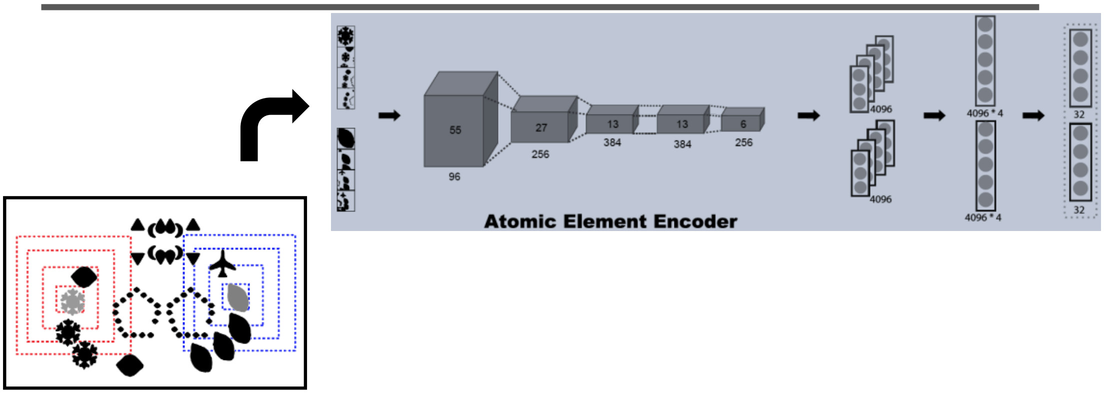
On the other hand, the symmetry principle is related to the overall structure so here we will need another learning model that can integrate the global information. Therefore we tackle this grouping problem using two different neural network models simultaneously.

Local feature: **Atomic Element Encoder**

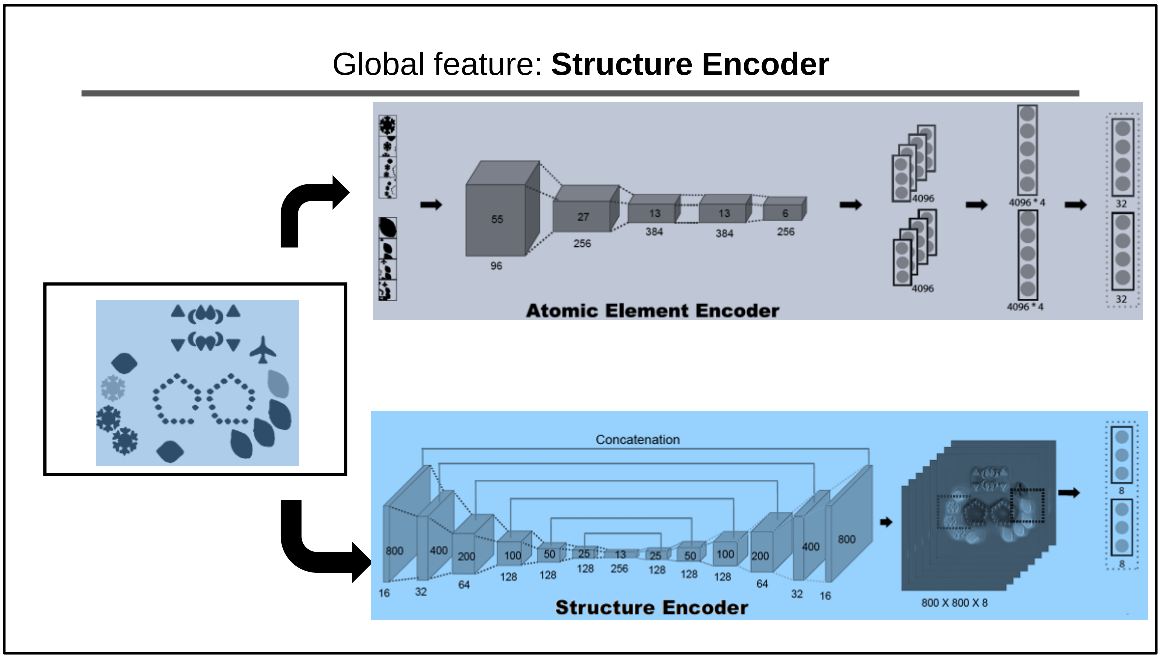


The first network is called Atomic Element Encoder which captures the local context of the elements. The input to this network contains 4 different scale of contexts around the element we are looking at. The network has a structure following Alex-Net.

Local feature: **Atomic Element Encoder**



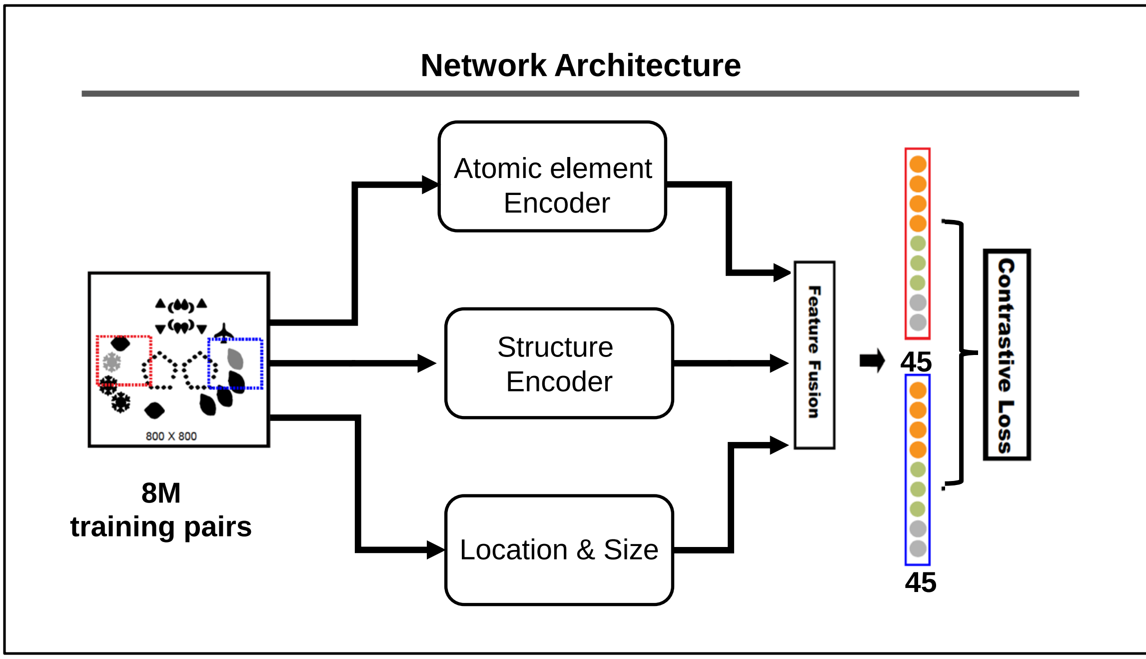
The first network is called Atomic Element Encoder which captures the local context of the elements. The input to this network contains 4 different scale of contexts around the element we are looking at. The network has a structure following Alex-Net.



The other network is called Structure Encoder which captures the global information. It feeds the entire image to the network. The network has an encoder-decoder structure with U-Net connection. The network outputs feature maps that have the same resolution as the input. These feature maps are able to integrate global information.

Network Architecture

The entire network architecture is like this. Besides the features extracted from the Atomic Element Encoder network and the Structure Encoder Network, we also fuse the location and size information directly into the final feature vector.

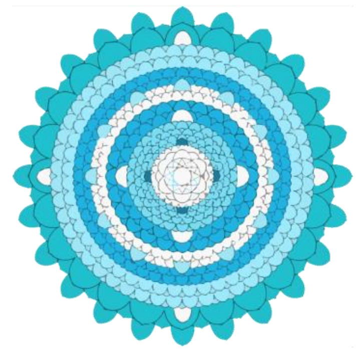


To train this network, we use a contrastive loss: it tries to pull two elements having the same group labels closer in the feature space, and push away two elements with different group labels.
The next question is how do we gather training data.

Data Collection: Lack of suitable patterns on the web

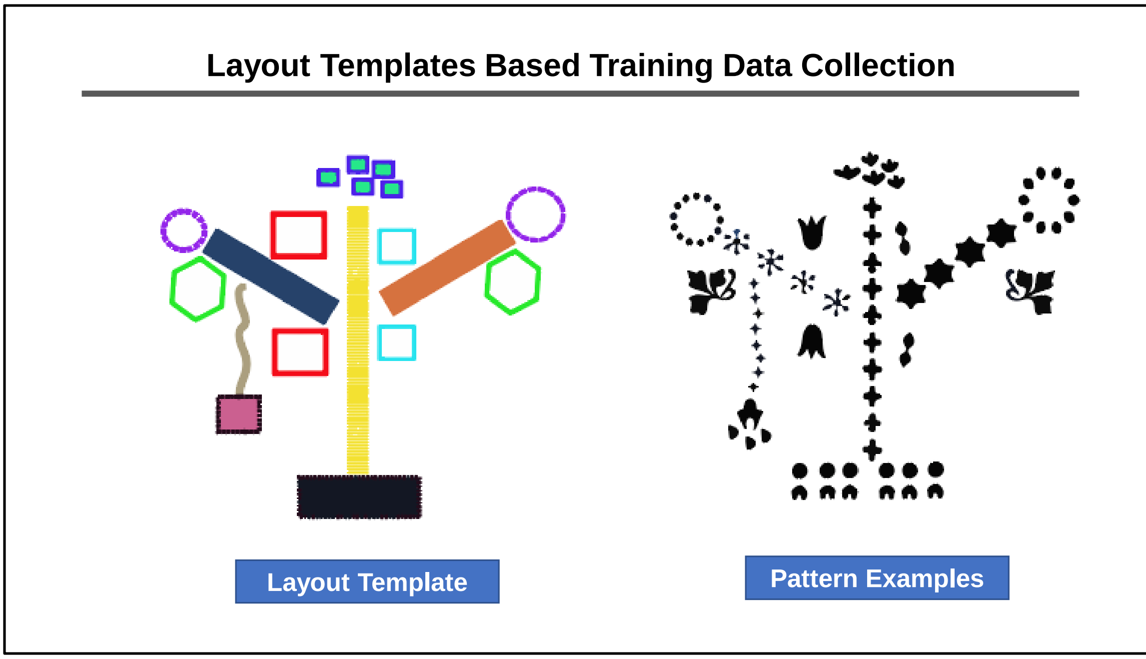
Black & White

Color Gradient

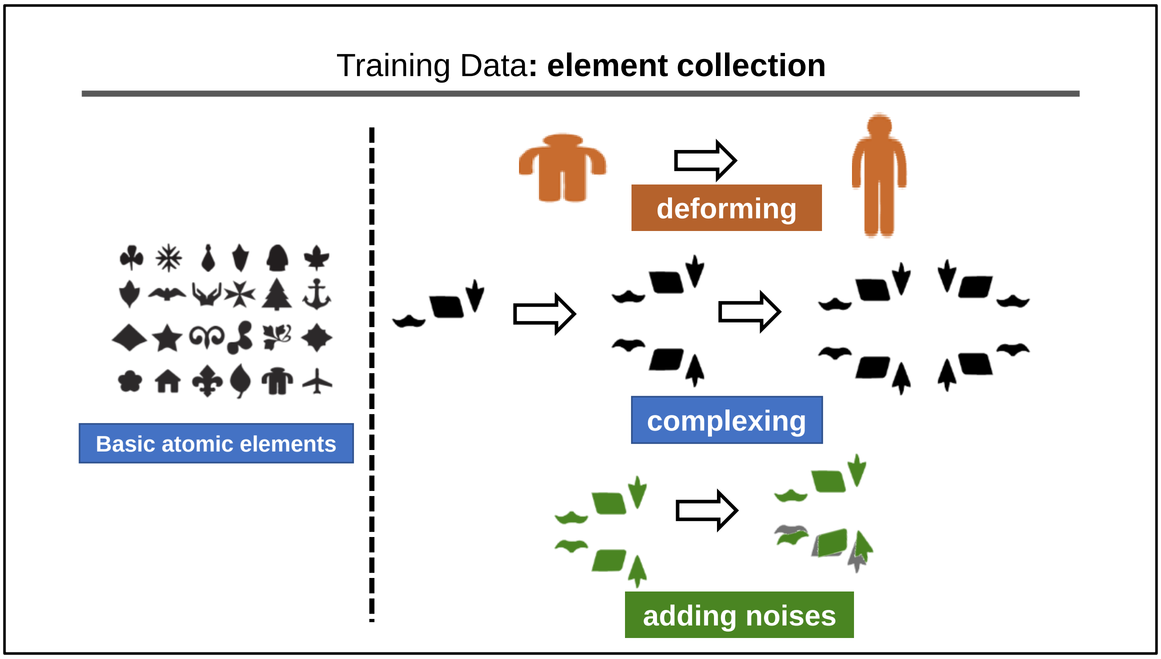


Lack Structural Variety

This task is not trivial. First, on the web, a great proportion of graphical patterns are binary images. Second, some patterns on the web have no flat colorized regions. In other word, in some regions, there is color gradient. It is hard to get the GT from these kinds of data. Lastly, although we can obtain the GT from some colorized patterns like this mondala from the web without much effort . However, this kind of data usually only have a small range of structural variety.



Manually Creating patterns with GT is impractical. We turn to a semi-automatic way. We first manually created pattern layout templates , and then generated pattern examples from these layout templates by inserting various elements

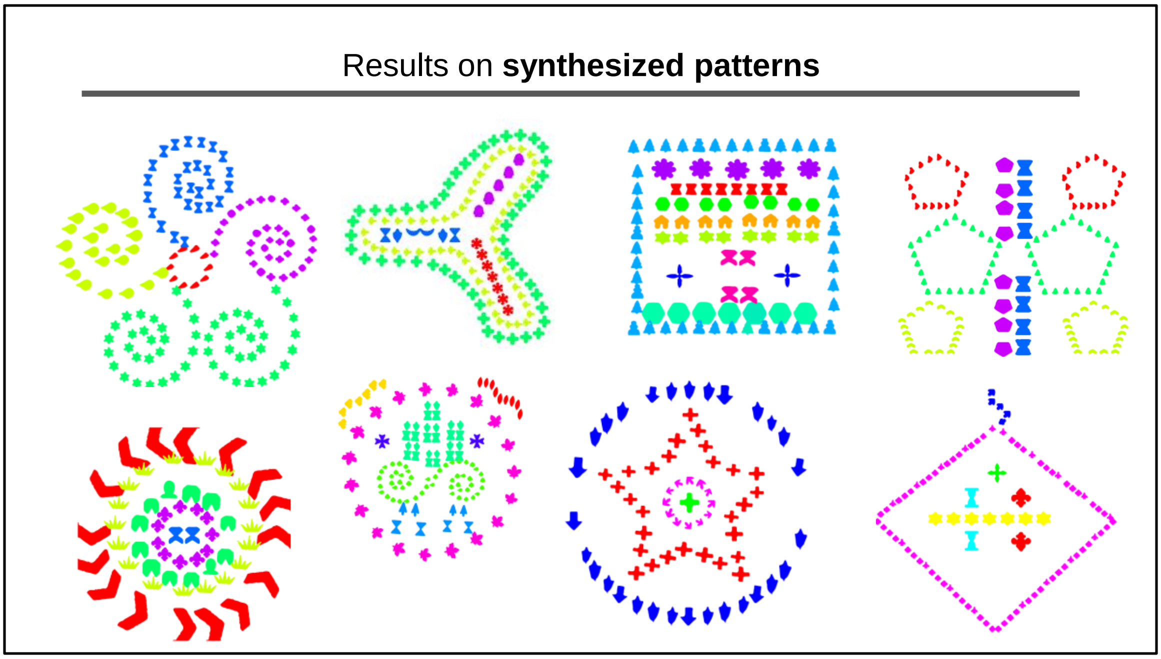


We collected 86 basic atomic elements. To augment the elements, we produced new atomic elements by deforming these basic atomic elements.
We also turned the Atomic elements into more complex elements procedurally.
Moreover, we also introduced various noises into the synthesized patterns.

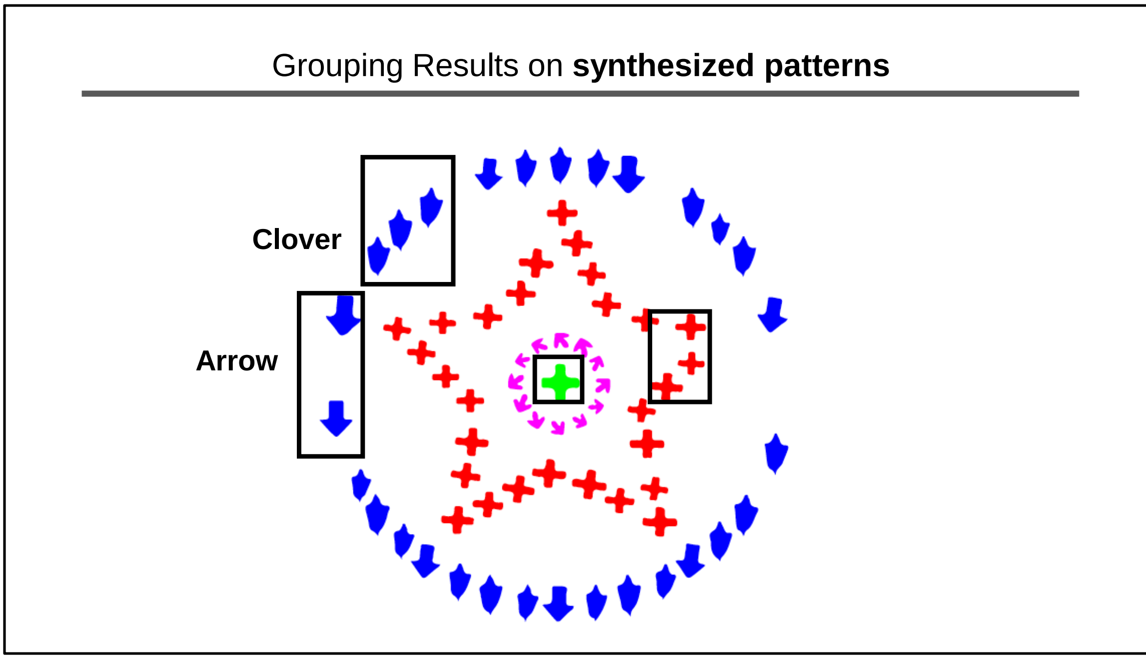
Training Data Collection

- ~800 pattern layout templates
- ~8K pattern images
- 500 positive and 500 negative pairs of elements
- ~**8M** training pairs

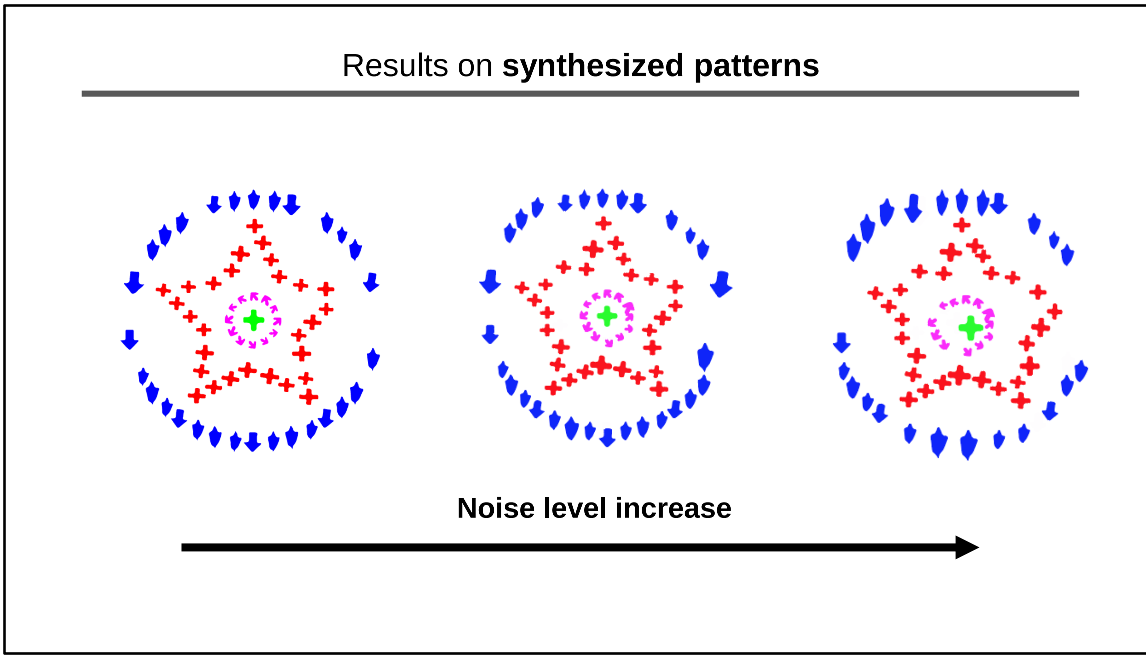
We finally collected ~800 pattern layout templates, almost 8K pattern images, We totally collected nearly 8M training pairs. Next we show results of our work.



We first tested our method on the synthesized data. We got good results on most of those examples.

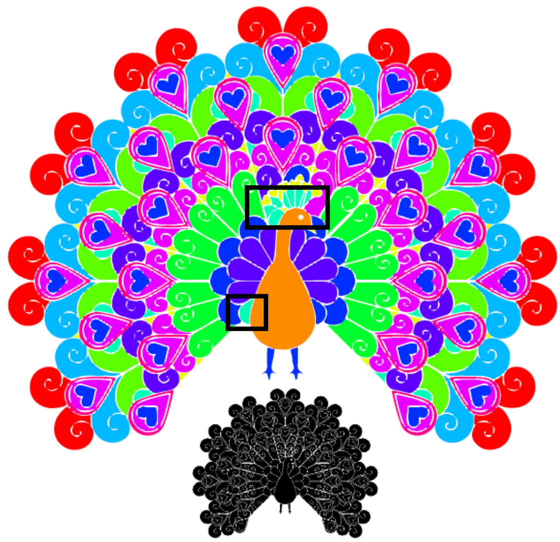


Let us see the result of this pattern, Our method can group these arrows with Clovers even a long path of the circle has no element.
Although the cross in the middle of this pattern is very similar with these crosses on the pentagram path. Our model separates them into two groups, which is consistent with human conception

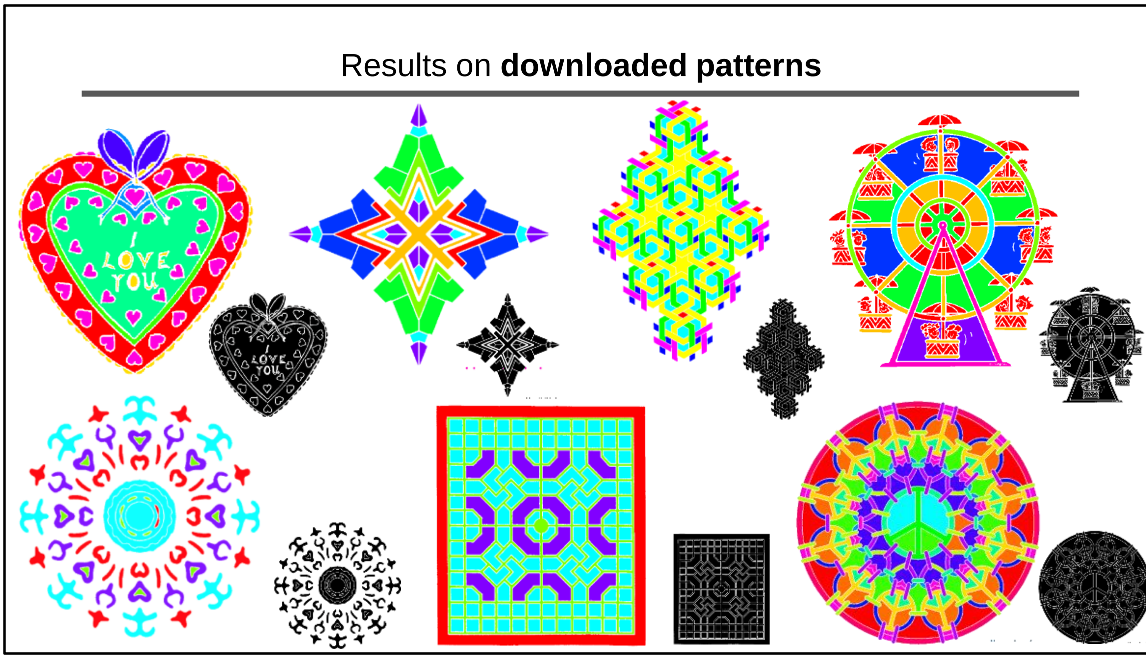


Even we increased the noise level. The results were still good.

Results on **downloaded patterns**

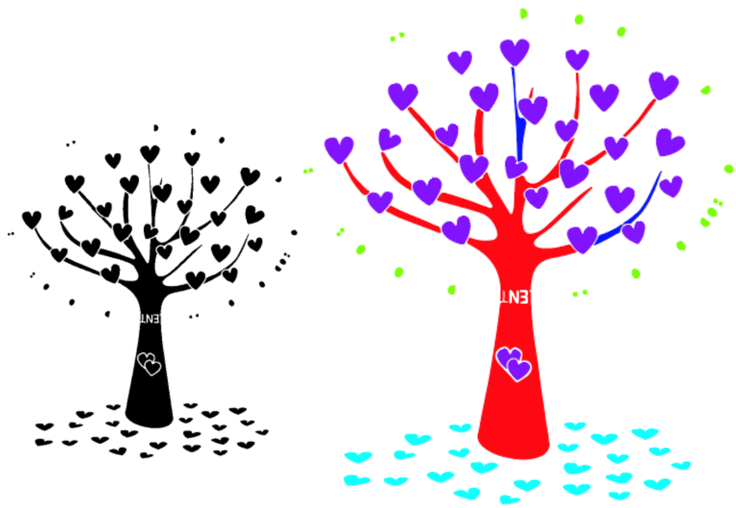


We also tested our modal on about 200 binary image patterns searched from the google and bing image by the keyword “coloring pages”. See this peacock, our model can get reasonable groupings on most elements. There are also several grouping errors like this small, occluded, piece of feather. It was grouped with other small piece of feather.

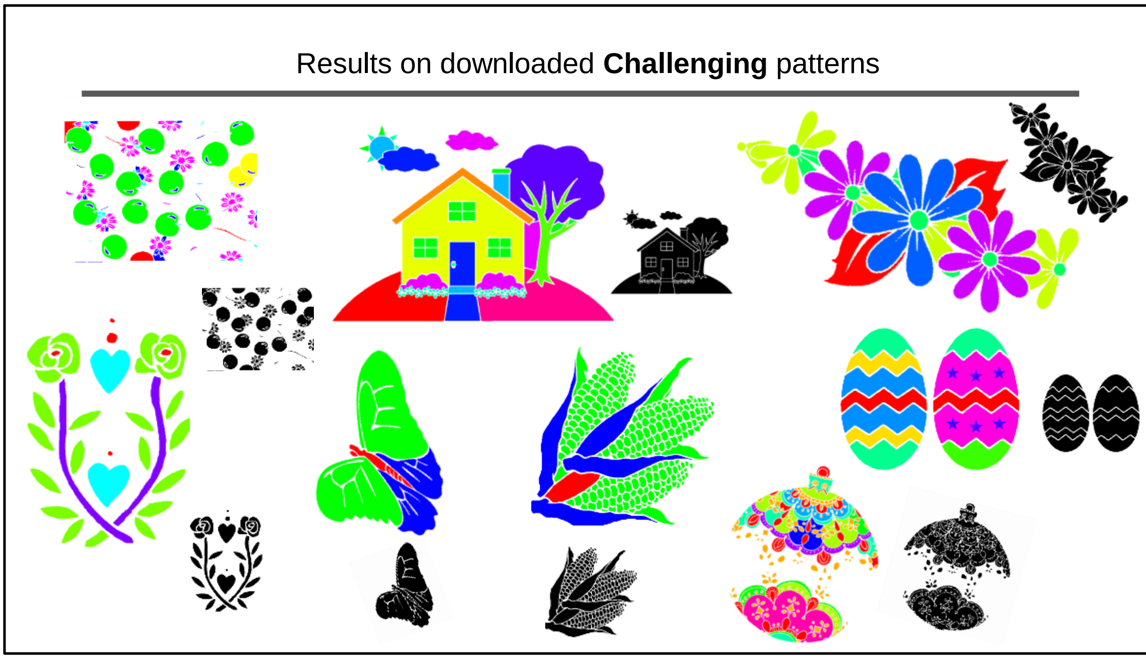


Here are results of some other relatively regular patterns . Most the results are reasonable.

Results on downloaded **Challenging** patterns



Some of our test patterns like this tree pattern are very Organic, having strong noises. Here we can see the out grouping results of this tree pattern are consistent with human perception on most elements. But there are still some failed cases. Our method did not group these two small twigs with the trunk of the tree.



Here are grouping results of some challenging examples

Quantitative Results with various **measures and alternatives**

	preset #group		auto #group	
	Rand index	purity	Rand index	purity
geometry distance	77.62%	73.13%	76.47%	76.64%
AlexNet	76.60%	70.96%	74.41%	71.16%
fine-tuned AlexNet	77.72%	73.12%	77.59%	80.21%
element encoder	78.34%	74.59%	77.72%	78.82%
structure encoder	78.79%	73.24%	77.35%	74.68%
element+structure enc.	80.28%	75.75%	80.03%	81.84%
our full measure	**83.05%**	**80.24%**	**83.58%**	**85.76%**

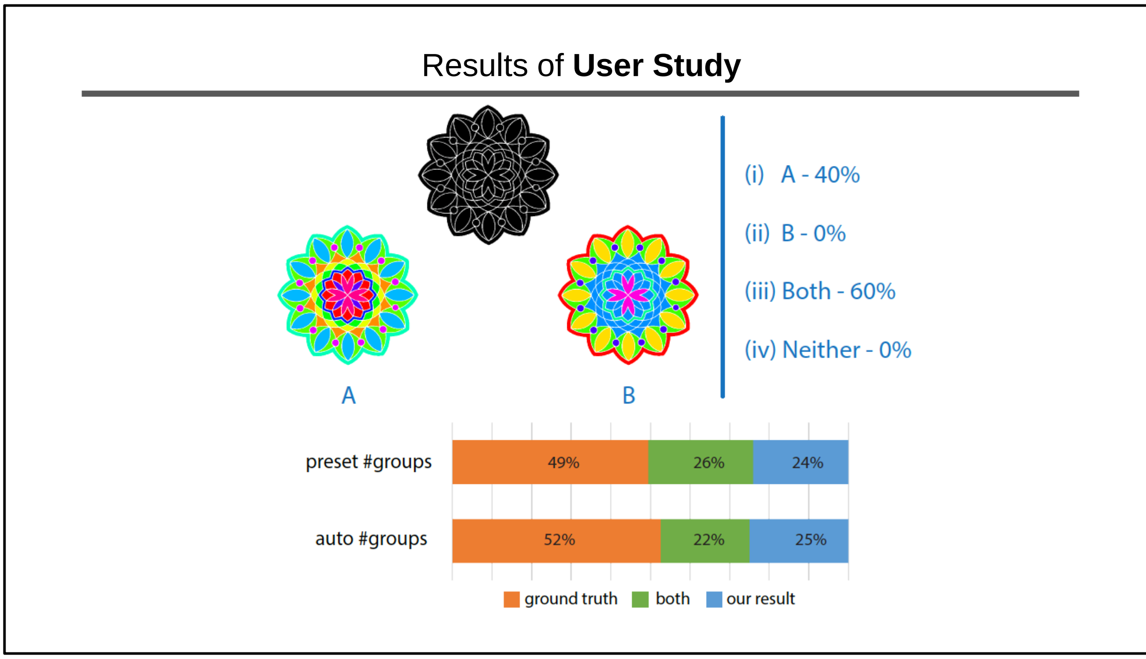
Greater score mean better grouping results

Here we show the evaluation results on different variations of network architectures. It's not surprising that our full network performs the best. For more details, please read the paper.

Quantitative Results with various **Clustering** Strategies

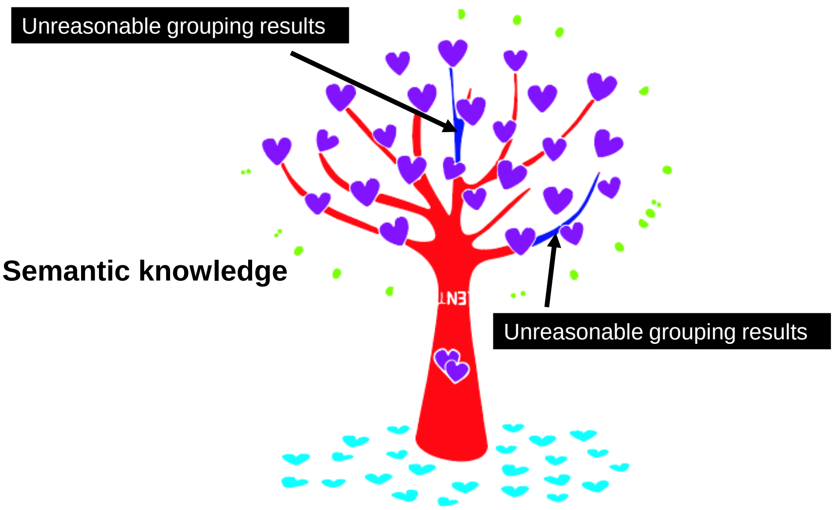
	Rand index	purity
affinity propagation	**83.05%**	**80.24%**
agglomerative (average linkage)	75.93%	71.13%
agglomerative (single linkage)	71.11%	68.38%
agglomerative (complete linkage)	76.76%	71.79%
k-means	80.85%	75.58%
Gaussian Mixture Models	80.92%	74.91%
normalized cuts	77.48%	71.72%
Tagger [Greff et al. 2016]	66.54%	55.21%

Here is a comparison of different clustering algorithms. The best performance goes to the affinity propagation algorithm. Please read our paper for more details.



We have also conducted a user study comparing our grouping results with the ground-truth labeling. We showed those two groupings side by side, randomly flipped the order, and asked users which grouping is better. The statistics show that our grouping result is almost as good as the ground-truth labeling result. Next, we discuss the limitation of our method.

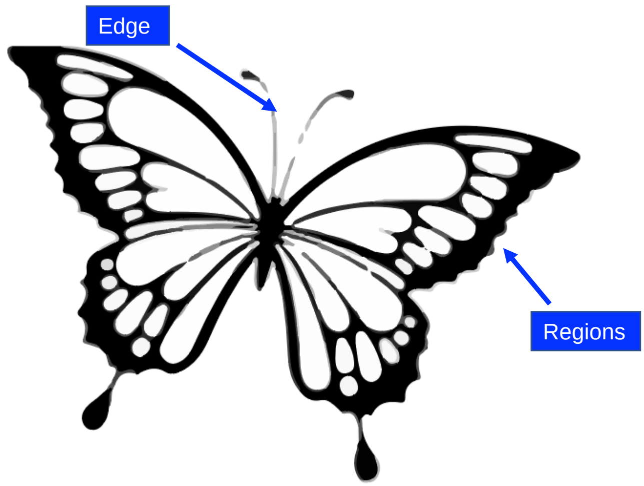
Limitation on model



No Semantic knowledge

The major limitation is related to our model. Currently, our model does not incorporate “semantic” knowledge, that is why these twigs have not been grouped with the trunk.

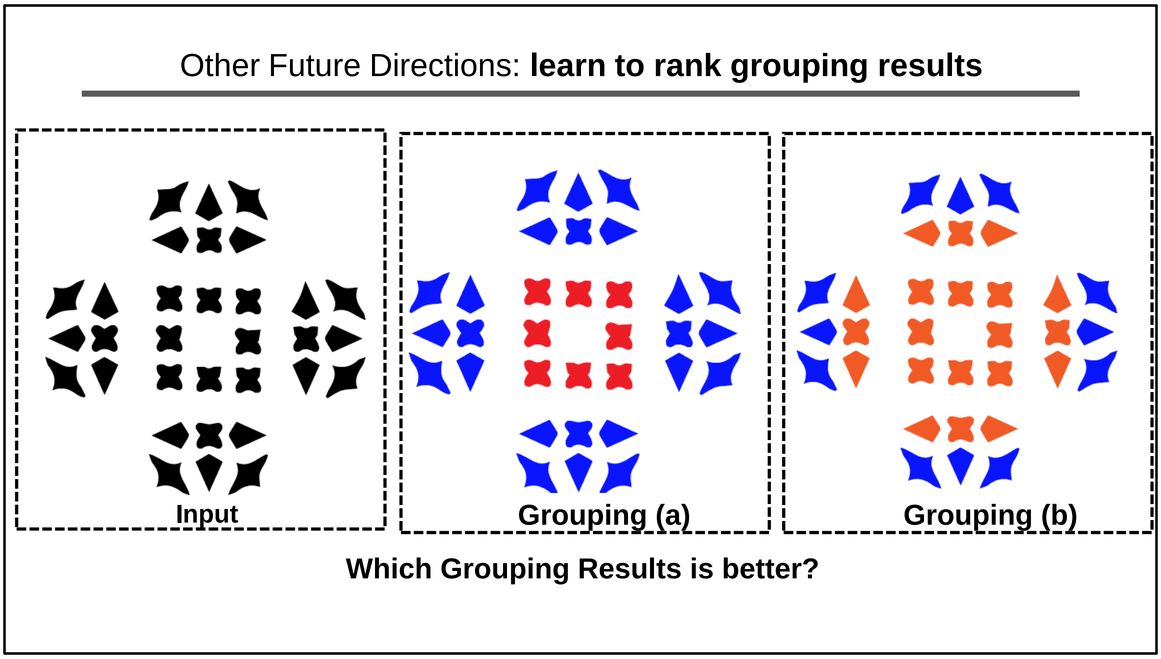
Limitation on input data



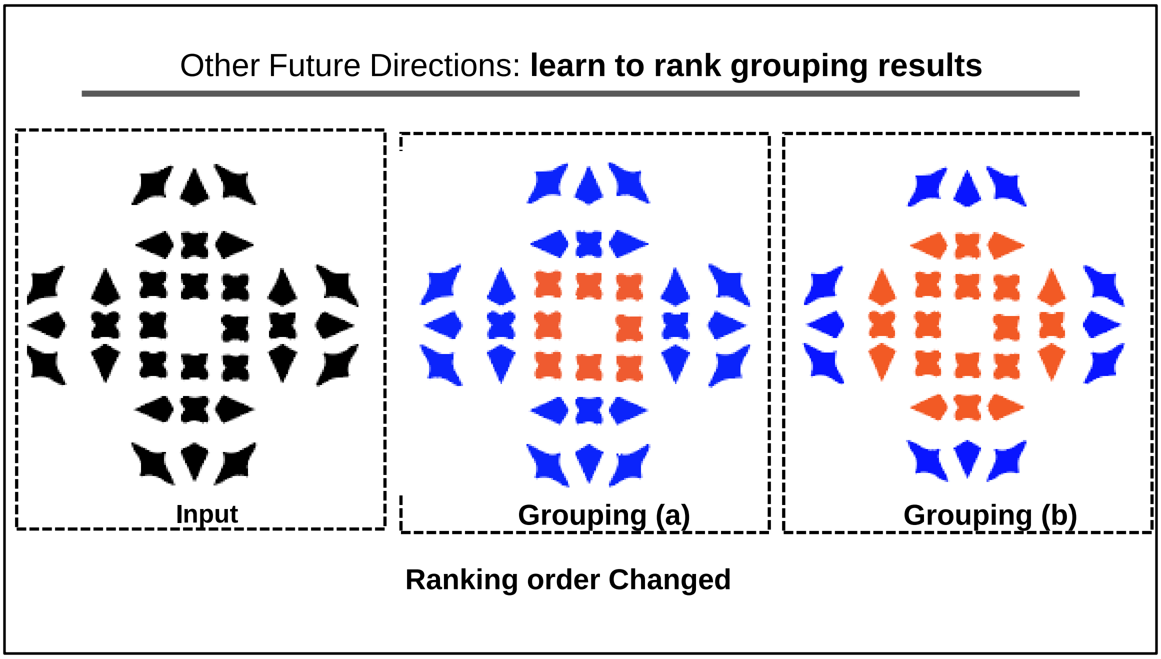
1. Another major limitation is about the input data range. Our current method only handle region-based elements. However, in real data. Graphical patterns often have various element types. For examples, this butterfly has edges of different thicknesses. Some are thick regions， while other are thin edges.

Future work: Unified Framework for Various types of Input Data

We hope our future method can well handle the input pattern with edges of various thicknesses.



Apart from the above future direcitons, learning to rank grouping results is another potential direction. For example, like this regular pattern, there are many reasonable grouping
Results like these two: a) and (b). I believe most human would prefer (a) to (b).



However, just move some elements' position a little, most human would prefer grouping (b) to (a). How to model a grouping ranking preference consistent with the statistics of human perception will be another interesting and potential problem.

Conclusion

- **First (data-driven + deep CNN)** for **discrete** 2D patterns**.**
- **Learned** shape-, context-, and structure-aware descriptors for graphical elements.
- A large, annotated dataset is provided **online.**
  http://people.cs.umass.edu/~zlun/papers/PatternGrouping/

Let's sum up our paper, Our work is the first data-driven method trained via a deep CNN for perceptual grouping of discrete graphical patterns.

Conclusion

- First (data-driven + deep CNN) for discrete 2D patterns.
- Learned **shape-, context-, and structure-aware descriptors for graphical elements.**
- A large, annotated dataset is provided **online.**
  http://people.cs.umass.edu/~zlun/papers/PatternGrouping/

Our work proposed a model to Learn shape-, context-, and structure-aware descriptors encoding graphical elements.

Conclusion

- First (data-driven + deep CNN) for discrete 2D patterns.
- Learned shape-, context-, and structure-aware descriptors for graphical elements.
- **A large, annotated dataset is provided online.**
  http://people.cs.umass.edu/~zlun/papers/PatternGrouping/

Moreover, Our work contributes A large, annotated dataset of pattern which should benefit
future research on pattern analysis and processing. All the source code and data is open on the project page.

Acknowledgements

- **Dr. Ke Li** for the help on **experimental data preparation.**
- The Science and Technology Plan Project of Hunan Province**.**
- The Massachusetts Technology Collaborative grant for funding the UMASS GPU cluster.
- NSERC Canada.
- Gift funds from Adobe Research.

We akownledge all the helps, comments, and fundings for this project.

Thanks!

Q&A

http://people.cs.umass.edu/~zlun/papers/PatternGrouping/