SSR-GNNs: STROKE-BASED SKETCH REPRESENTATION WITH GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing end-to-end visual recognition models do not possess innate spatial invariance and are thus vulnerable to out-of-training attacks. This suggests the need of a better representation design. This paper follows existing cognitive studies to investigate a sketch representation that specify stroke information on vertices and inter-stroke information on edges. The resultant representation, combined with a graph neural networks (dubbed as SSR-GNNs), achieve both high classification accuracy and high robustness against translation, rotation, and stroke-wise parametric and topological attacks thanks to the use of spatially invariant stroke features and GNN architecture. While prior studies demonstrated similar sketch representations for classification and generation, these attempts heavily relied on run-time statistical inference rather than more efficient bottom-up computation via GNN. The presented stroke-based sketch representation poses good structured expression capability as it enables generation of sketches semantically different from the training dataset. Lastly, we show SSR-GNNs are able to accomplish all tasks (classification, robust feature learning, and novel pattern generation), which shows that the representation is task-agnostic.

1 INTRODUCTION

Unlike human vision system, it is well acknowledged that end-to-end deep learning methods lack intermediate representations that enable innately invariance to spatial translation and rotation (Hinton, 2021; 1979; Snead, 2017; Baggaley, 2017; Levesque et al., 2011; Clark, 2015; Marcus, 2013; 2017). While such transformation invarianace can potentially be achieved through expensive robust training, it is believed that invariance (1) should be an innate property rather than an external model constraint, and (2) should not trade off recognition accuracy.

A commonly sought-after solution is to identify a part-whole structure (Hinton, 2021), following the insights that the human vision system parses scenes into atomic parts for recognition and generation, while both parts and the topologies of parts are invariant to spatial transformations. The part-whole structure is also supported by the Gestalt principles (Desolneux et al., 2007) and cognitive science (Lake et al., 2017).

Building on top of existing work and within the context of computer vision, we propose a part-whole representation where strokes, as parts, are connected as a graph to form a sketch. The focus on sketches draws inspiration from studies in biology and cognitive science. (Landau et al., 1988) shows that human vision relies more on shapes than on textures or colors. Studies also show that successful CNNs learn shape representations from natural images (Geirhos et al., 2019; Kriegeskorte, 2015). In this paper, we assume that non-sketch inputs, e.g., textures, can be ignored, and it is feasible to convert images into sketches, e.g., through (Geirhos et al., 2019; Lamb et al., 2020).

An example of the proposed representation is shown in Fig. 1: From an input image of "R", we use unsupervised image processing to identify fork points that separate strokes (Lake et al., 2015), and estimate control points of these strokes to form an undirected graph representation where each vertex contains stroke information and edges specify interactions between strokes. Specifically, to achieve spatial invariance, each vertex encodes pairwise spatial distances between each pair of control points for the corresponding stroke (see v_2 in Fig. 1), which is a $n \times n$ matrix. If two strokes (each with n control points) are connected, we form an edge in the graph. The edge encodes the pairwise distance between each control point from one stroke to each of the other (see $e_{2,3}$ in Fig. 1).

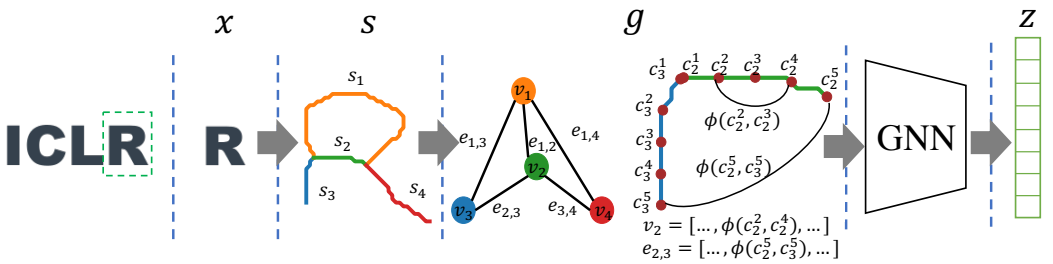


Figure 1: An overview of SSR-GNNs. We take "R" for example. The image "R" is composed of 4 strokes $s = \{s_1, s_2, s_3, s_4\}$. Each stroke is composed of 5 control points. $s_2 = \{c_2^1, c_2^2, c_2^3, c_2^4, c_2^5\}$. The 4 strokes are associates with 4 vertices $v = \{v_1, v_2, v_3, v_4\}$ in graph g. The value of vertex is the pairwise distance between control points. $v_2 = \{\phi(c_2^p, x_2^q)\}_{p=1,q=1}^{p=5,q=5}$. The value of edge is the pairwise distance between two connecting control point. $e_{2,3} = \{\phi(c_2^p, c_3^q)\}_{p=1,q=1}^{p=5,q=5}$. After passing to a learnable graph neural network(GNN), the z is the high level representation of the image x.

Since distances between control points are invariant to spatial transformations, our graph design is innately spatially-invariant. To use the graph representation for classification, we adopt a Graph Neural Network (GNN) for its good generalization capability (Scarselli et al., 2008), flexibility at managing variable input graph topologies, and perseverance of spatial invariance.

In summary, we present a Stroke-based Sketch Representation for GNNs (dubbed as SSR-GNNs) to learn spatial-invariant classifiers. We claim the following new contributions:

- Through extensive experiments on both MNIST and two subsets of the Quickdraw dataset (Ha & Eck, 2017), we show that SSR-GNNs are innately robust to rotations and translations, while maintaining high classification accuracy.
- In addition, we show that SSR-GNNs are robust to parametric and topological attacks without robust training, which suggests that stroke-based graphs are robust features for perception.
- Lastly, we show that SSR-GNNs can generate novel sketch patterns distinguishable from the training set. E.g., by learning to classify decimal digits, the model can then be used to generate hypothetical "A"s to "F"s for a hexadecimal system. This shows that SSR-GNN has strong structured expression capability.

Task-agnostic is another desired property of representations, meaning that while the representation should facilitate down-stream tasks such as classification and generation, its learning is not dependent on any specific tasks. Being able to accomplish all tasks (classification, robust feature learning, and novel pattern generation) shows that the proposed stroke-based representation is task-agnostic.

2 METHODS

The procedure of obtaining the high level stroke representation Z from the image X can be summarized as: 1) decomposing the source image X to a stack of expressive stroke representation S, 2) associating the S to the graph representation G and 3) learning the mapping G to the high-level representation Z through supervised training. Our paper aims to link the strokes of an image to a graph and so that the produced representation inferred from the graph can be further transferred to series of down-stream tasks. Fig. 1 depicts the overall, and we break it down to sub-steps and describe each step's details in the following sections.

2.1 ACQUIRING STROKES FROM AN IMAGE

We first decompose a source image $x \in X$ into a set of strokes, $s = \{s_i\}^{|s|} \in S$, where the number of strokes $|s|$ varies for each image. We follow the pre-process procedures from (Lake et al., 2015), which include thinning the image (Lam et al., 1992), detecting the fork points (Liu et al., 1999) and finally merging the noisy & redundant fork points by the maximum circle criterion (Liao & Huang, 1990). The order of the strokes is then formed by a random walk between fork points. Fig. 1 showcases an example of strokes acquired from the image "*R*".

2.2 STROKE-BASED GRAPH REPRESENTATION

Taking strokes as input, we associate them with a graph g, denoted as sets of vertices and edges (v, e), where each vertex $v_i \in v$ corresponds to an independent stroke s_i, and e denotes the set of edges $e = \{e_{i,j}\}_{s_i, s_j \text{are connected}}$ connecting two vertices v_i, v_j. Here, we consider the geometrical connection between two strokes s_i, s_j (see Fig. 1). V, E denote the collections of v, e from X.

Building upon (Lake et al., 2015)'s stroke representation, s_i is modeled as an uniform cubic b-spline and decomposed into three variables, namely, the *control point*, the *scale* and the *mean*. In practice, we fit the stroke with varying sizes using b-spline and then re-fit back to a new stroke which contains n fixed number of points. Then we represent the set of control points for one stroke and we denote them as $s_i = \{c_i^p\}_{p=1}^n$.

For a spatially-robust representation, in this paper, we present a novel stroke-based graph formation other than directly assigning the control point as a vertex and point-wise relationship as an edge. Instead, we represent the vertices-edges by the bipartite distance between each two control points. In particular, for each vertex $v_i = \{\phi(c_i^p, c_i^q)\}_{p=1,q=1}^{p=n,q=n}$ and $v_i \in \mathbb{R}^{n \times n}$, where ϕ is the Euclidean distance and c_i^p, c_i^q are control points in stroke s_i. Similarly, each single edge is represented as $e_{i,j} = \{\phi(c_i^p, c_j^q)\}_{p=1,q=1}^{p=n,q=n}$, where c_i^p, c_i^q are control points from strokes s_i and s_j respectively.

2.3 LEARNING WITH THE GRAPH NEURAL NETWORKS

To address the issue of having dynamic number of vertices and interaction between vertices, we apply a message passing neural networks(MPNNs) (Gilmer et al., 2017; Duvenaud et al., 2015) (a sub-category of GNNs) over the formerly constructed stroke-based graphs.

Specifically, the MPNN contains three sub-steps, namely: message passing, update and readout (Battaglia et al., 2018). Formally, the calculation of the three sub-step are:

$$m_{v_i}^{(t+1)} = \sum_{s_i, s_j \text{are connected}} M_t(v_i^{(t)}, v_j^{(t)}, e_{i,j}^{(t)}), \tag{1}$$

$$v_i^{(t+1)} = U_t(v_i^{(t)}, m_{v_i}^{(t+1)}), \tag{2}$$

$$z = R(v_i^{(T)} | v_i^{(T)} \in v). \tag{3}$$

The message passing and update phase executes for T times. The message at phase $t+1$, $m_{v_i}^{(t+1)}$ is encoded by vertex information $v_i^{(t)}$, adjacent vertex $v_j^{(t)}$ and edge information $e_{i,j}^{(t)}$ at step t. The new vertex information $v_i^{(t+1)}$ is updated by current vertex information $v_i^{(t)}$ and message information $m_{v_i}^{(t+1)}$. After T steps, the high level representation z is a feature vector, combining all vertices information by a readout function at step T. Both of M_t, U_t, R are parameterized by a neural architecture with learnable parameters. After the readout, a linear classifier with softmax is used for the linear classification on the feature vector z. To simplify the notation, we define the MPNN as $\mathcal{F} : g = (v, e) \to z$ and linear classifier as $\mathcal{F}_\theta : z \to y$.

Our model can be optimized by any typical supervised training objectives and the learned representations can be used for other downstream tasks, showing SSR-GNNs is task-agnostic. In this paper, we adopt specific objective for each empirical task. Briefly, we describe each objective in the following sections one by one.

2.4 OPTIMIZATION OBJECTIVES

We have two objectives for three experimental tasks in Sec. 3. For classification and spatial-robustness task in Sec. 3.1, we use the Eq. 4. For robust feature learning in Sec. 3.2, we use the Eq. 5. For the novel shape/sketch generation in Sec. 3.3, we adopt both objectives and reconcile them. Note that since the objective will be used from different subsets of the whole set X later, we define the $\chi \subseteq X$. The notation of S, V, E on subset χ are referred to S_χ, V_χ, E_χ. For the whole framework, we define the process mapping the s to v, e as a deterministic function $\mathcal{P} : s \to v, e$. The function $\mathcal{P}$ can be applied to the set S_χ, mapping $S_\chi \to V_\chi, E_\chi$ and S, mapping $S \to V, E$, in the same manner.

2.4.1 CLASSIFICATION OBJECTIVE

The objective for classification is the typical cross-entropy loss. Given the label set Y_χ for χ, the objective function is:

$$\min_{\mathcal{F},\mathcal{F}_\theta} \mathcal{L}(\mathcal{F}, \mathcal{F}_\theta, S_\chi, Y_\chi) = \max \mathbf{E}[Y_\chi \log(\mathcal{F}_\theta(\mathcal{F}(V_\chi, E_\chi)))], \text{where } V_\chi, E_\chi = \mathcal{P}(S_\chi). \tag{4}$$

2.4.2 OBJECTIVE OF GENERATING/MODIFYING THE STROKES

The objective to generate or modifying the strokes is to optimize the stroke s on the classification loss. If labeling the stroke graph as y, we fix the feature extractor $\mathcal{F}$ and linear classifier $\mathcal{F}_\theta$, and minimize the objective function over s:

$$\min_{s} \mathcal{L}(\mathcal{P}, \mathcal{F}, \mathcal{F}_\theta, s, y) = \max_{s} \mathbf{E}[y \log(\mathcal{F}_\theta(\mathcal{F}(v, e)))], \text{where } v, e = \mathcal{P}(s). \tag{5}$$

The objective function only guarantees generating s to fit label y. To generate more natural looking strokes, additional constraints on s are furthered needed and we list them below.

Boundary Constraint: Preventing the generated strokes to be out of image boundary, we supplement a boundary constraint on s. It is accomplished by clipping s to the image size.

Distance Constraint: To avoid having strokes with abnormal length, we design constraints on the pairwise distance between control points. There are two types of constraints. Taking V_χ for example, the hard constraint is by measuring the ℓ_1 difference between the mean of V_χ and v as $\mathcal{L}_{hard} = |\mathbf{E}(V_\chi) - v|$. The soft constraint is measuring how well the log-likelihood of v follows the distribution formed by the set V_χ. The objective function is:

$$\mathcal{L}_{soft} = -\mathbf{E}[\log p(v|V_\chi)] - \mathbf{E}[\log p(e|E_\chi)], \text{where } v, e = \mathcal{P}(s). \tag{6}$$

Radius Constraint: Except the distance constrain, we can also pose constraint on the radius of the stroke. The radius r_i of a stroke s_i is defined as:

$$r_i = \{\frac{(c_i^j - c_i^{j+1}) \cdot (c_i^j - c_i^{j-1})}{\|c_i^j - c_i^{j+1}\| \|c_i^j - c_i^{j-1}\|}\}_{j=2}^{j=n-1}. \tag{7}$$

Here, we further extend the definition of $\mathcal{P}$ to $\mathcal{P} : s \to v, e, r$ inducing radius measure. The way to formulate constraint on the radius is similar to the distance constraint.

3 EXPERIMENTS

We empirically evaluate the validity of our SSR-GNNs framework from the aspects of spatial robustness, performance on classification task, and the structured expression capability. In particular, we first conduct the hand-written digit classification experiments using the stroke representation under a wide range of spatial transformations to test whether our representation achieve high spatial robustness while maintaining high accuracy. Spatial robustness can also be implicitly leveraged by exploring the robust feature without the robust training, compared with the robust model by adversarial training (Tsipras et al., 2019). To show our representation is with high structured expression capability. Similar to the generation experiment from (Goodfellow et al., 2014) showing generative neural networks is able to create new samples based on known distribution, we demonstrate SSR-GNNs is capable of generating novel categories of sketch patterns that are categorically distinguishable from the existing set (from the existing decimal digits to generate digits A (10) to F (16) in a hexadecimal system).

3.1 CLASSIFICATION AND SPATIAL ROBUSTNESS

3.1.1 DATASET AND PRE-PROCESSING

We exploit two datasets as our testbeds: MNIST and Google Quickdraw datasets (Ha & Eck, 2017). MNIST is a hand-written digit dataset containing the numerical digits from 0-9. It contains $60,000$ training samples and $10,000$ testing samples. Google Quickdraw is a human hand-drawn sketch dataset with 345 different categories, ranging from The Great Wall, airplane, to hands, square and

dogs. They have two types of input. The first type is the sketch image, shifted to top-left corner and normalized the scale to 224×224. CNNs and our method use this type of input. The second type is the simplified stroke key points in temporal order (Xu et al., 2021; 2018), computed by the Ramer-Douglas-Peucker algorithm (Ramer, 1972). The Graph transformer (Xu et al., 2021) and Recurrent Neural Network (Xu et al., 2018) uses this type of input. Due to the abstractness of some categories in Quickdraw, we just adopt two sub-parts of the dataset for our experiments. The first sub-part is relatively simple, containing all shape categories including *circle, hexagon, line, octagon, square, triangle, zigzag* (see Fig. 2(b)). We then test the model on a more complex sub-part, containing all body categories including *arm, ear, elbow, face, finger, foot, hand, nose, toe, tooth* (see Fig. 2(c)). For both parts, we select 1000, 100, 100 samples per category for training, validation and testing. To avoid the spatial transformation moving strokes out of image, we zero-pad 40 pixels at each side of the image. In the pre-processing step acquiring strokes from sketch image, we follow procedure (Lake et al., 2015) in Sec. 2.1. Since the strokes in the sketch image are mostly short ones, we set $n = 10$ for all experiments. Due to the existence of noisy fork points, the maximum circle criterion cannot be used on sketch image. We dilate the sketch with 4 pixels which allows maximum circle criterion to remove the noisy fork points.

3.1.2 Network Architecture and Training Details

We adopt the MPNN (from Sec. 2.3) following the same architecture as the gated graph neural networks(GG-NNs) (Li et al., 2015). The message passing function is $M_t(v_i^{(t)}, v_j^{(t)}, e_{i,j}^{(t)}) = \Phi_1(e_{i,j}^{(t)}) \cdot v_j^{(t)}$. The update function is a Gated recurrent unit (Cho et al., 2014), where $U_t = \text{GRU}(v_i^{(t)}, m_{v_i}^{(t+1)})$. The readout function is $R = \sum_{v_i^{(T)} \in v} \sigma(\Phi_2(v_i^{(T)}, v_i^{(0)})) \odot (\Phi_3(v_i^{(T)}))$. For MNIST, Φ_1, Φ_2, Φ_3 each is a linear four layers networks, with the intermediate feature size as 128, 256, and 128. The message passing iterations T is set to 1 and the final feature vector size is set to 10. We use the batch size 128, with an initial learning rate $= 1e^{-4}$. To handle the complex images in Quickdraw, we increase the depth of our architecture to 8 linear layers and the dimension of the intermediate features are 128, 256, 512, 2048, 521, 256, 128. The message passing iterations is set to 3 and the dimension of the final feature vector is 1024. We set the the batch size as eight with an initial learning rate $= 2e^{-4}$ using SGD optimizer. The objective function is a typical cross-entropy loss, as depicted in Sec. 2.4.1.

3.1.3 Baselines

Convolutional Neural Network (CNN) is one of the important baseline in our comparisons. On the MNIST dataset, we build the CNN with two convolutional layers and two linear layers maintaining comparable learnable parameters as our models for fair comparisons. On Quickdraw dataset, we choose the Inception network (Szegedy et al., 2015) as the baseline.

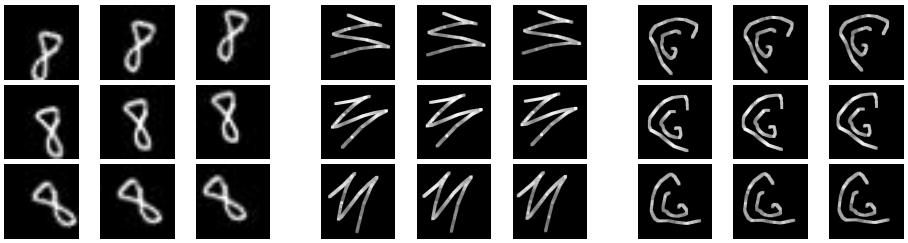
Graph Transformer (Xu et al., 2021) can also be applied as encoding the sketch image as a graph. However, the encoded graph differentiates from our method in such a way that each of the vertex in it is an independent stroke key point represented by its coordinates. Such representation is therefore not invariant to spatial transformation. In addition, Graph Transformer based method needs flag bits, indicating the start or end of a stroke, and temporal information which tracks sequence of the pen states. These information is not needed for the presented method.

3.1.4 Evaluation

To evaluate the spatial robustness of the model, we apply the rotation θ and translation (δ_x, δ_y) attacks on the input images (Engstrom et al., 2019). It moves the pixel at (x, y) to (x', y'):

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}, \tag{8}$$

For MNIST dataset, we rotate at most $30°$ and translates with at most 3 pixels. For Quickdraw dataset, since the image size increases, we increase the maximum translation to 10 pixels. To generate the transformed images, we discretize the parameters to grids of rotations and translations (as shown in Fig. 2). We sample 5 values per translation direction and 31 values for rotations. Together, the procedure yields 775 transformed samples per image. If one of the transformed sample fails a model, the model is then considered not robust against spatial transformation w.r.t. the input image.



(a) MNIST (b) Quickdraw(Shape) (c) Quickdraw(Body)

Figure 2: Spatial transformation on MNIST, Quickdraw(shape), Quickdraw(body). The grid of each row is the rotation of $-\theta, 0, \theta$. The grid of each column is the translation of $(-\delta_x, -\delta_y), (0, 0), (\delta_x, \delta_y)$. For MNIST, $\theta = 30^\circ, \delta_x = \delta_y = 3$px. For Quickdraw, $\theta = 30^\circ, \delta_x = \delta_y = 10$px.

Method	Evaluation	MNIST	Quickdraw(Shape)	Quickdraw(Body)
CNNs	Accuracy	98.98%	87.14%	80.10%
	Spatial Robustness	0.01%	21.90%	31.10%
	Parameter Size	600,810	25,315,474	25,315,474
Graph Transformer	Accuracy	-	80.71%	75.4%
	Spatial Robustness	-	0.00%	0.00%
	Parameter Size	-	39,984,729	39,984,729
Ours (SSR-GNNs)	Accuracy	93.01%	73.00%	64.20%
	Spatial Robustness	93.01%	73.00%	64.20%
	Parameter Size	546,634	8,707,868	8,707,868

Table 1: The accuracy and spatial robustness on three dataset(MNIST, Quickdraw(Shape), Quickdraw(Body)). We compare our method(SSR-GNNs) with CNNs(Inception-V3 for Quickdraw) and graph transformer. Our evaluation metrics are accuracy, spatial robustness and the parameter size.

3.1.5 Classification and Spatial Robustness Result

Table 1 summarizes the experimental results. 1) For accuracy, CNNs achieve the best result. Our method achieves comparable accuracy. Specifically comparing to CNNs, our method's accuracy is 5.97%, 14.14%, and 15.90% lower on MNIST, Quickdraw(Shape) and Quickdraw(Body). The gaps between ours with graph transformer's are even smaller (MNIST cannot be processed to the graph transformer). 2) For spatial robustness, since our stroke representation is invariant to spatial transformations, SRR-GNNs' performance before and after spatial transformation keep the same. We can see that SSR-GNNs outperforms all other methods with a significant margin taking spatially transformed inputs. The spatial robustness of CNNs drops to 0.01%, 21.90%, 31.10% on three datasets. And spatial robustness of graph transformer is essentially 0% on the two subsets of Quickdraw. 3) Our model is also parameter-size-wise efficient. We only compare the number of parameters on the Quickdraw dataset because we design the CNN containing similar number of parameters to ours on MNIST for a fair comparison. W.r.t. Inception-V3 and Graph Transformer, SSR-GNNs only need 34% and 21% of parameters. The results validate that our SSR-GNNs achieve significantly higher spatial robustness while maintaining comparably accuracy with a less amount of model parameters.

3.2 Robust feature

A robust model has the capability to explore the robust feature (Ilyas et al., 2019; Tsipras et al., 2019). We show SSR-GNNs are spatially-robust by directly exploring the robust feature without robust training, unlike current studies that mainly analyze the correlation between image pixels and label on a robust model by adversarial training (Tsipras et al., 2019). If features have positive correlation with the label, they are robust features. We follow the setting, with two major differences: 1) we do not need adversarial training since our model presents high spatial robustness shown in Sec. 3.1. 2) we can explore the robust feature at stroke level. To seek correlation between the stroke-based representation and the label, we launch adversarial attack on our stroke-based graph.If either altering the graph geometry or altering graph vertex and edge values can alter the model prediction, the corresponding alterations are the robust features. We adopt the same classification model (SSR-GNNs) from Sec. 3.1.

3.2.1 ALTERING THE GRAPH GEOMETRY

In our experiment, we consider adding/deleting vertices to alter the graph geometry. We conduct the experiment by adding one stroke s_2 on digit 1 with single stroke s_1 and targeting the stroke graph to be categorized as digit 7. Since our representation is spatially invariant, the starting point of s_2 is fixed and connected to either side of s_1 and all other trainable $n-1$ control points of s_2 are initialized with the same value. The procedure to get s_2 follows the setting described in Sec. 2.4.2. We apply hard constraint on radius and soft constraint on distance, where the χ in Eq. 6 is the set of all two strokes 7.

We visualize in Fig. 3(a) to show how the added stroke evolves. Part of the new stroke evolves towards being flat at step 100, while the final newly-added stroke becomes flat after more than 1000 steps. Noting that the confidence score of the sample being classified as 7 increases. We conduct the same experiment on other sample for adding one stroke to 1, targeting 7 in Fig. 3(b) and adding one stroke to 7, targeting 2 in Fig. 3(c). Here, we show that the newly generated stroke onto our graph representation is a kind of robust feature.

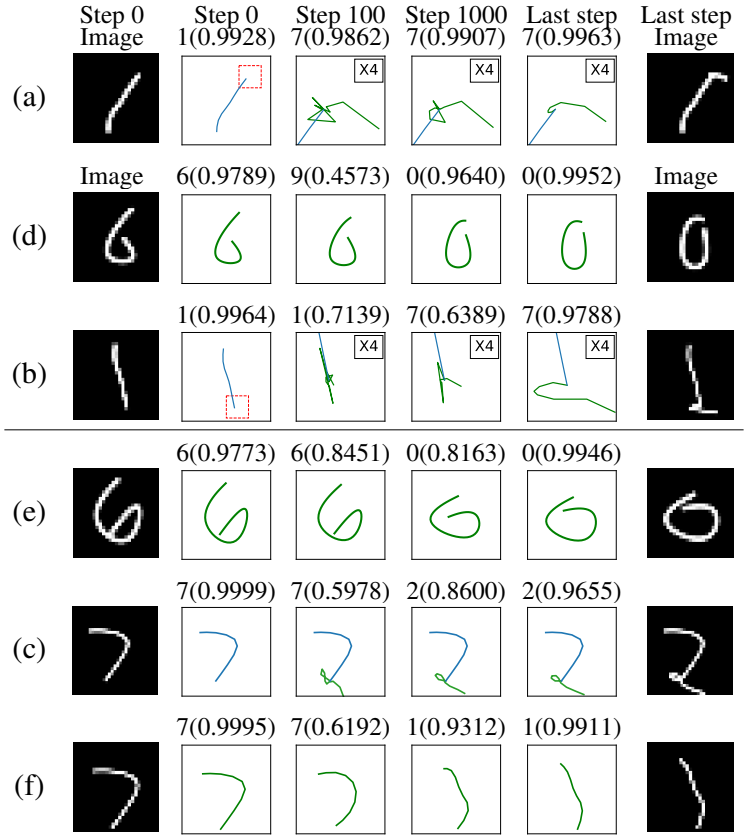


Figure 3: Exploring the robust feature by altering graph geometry and modifying the vertices/edges value. (a)(b)(c) The robust feature by modifying the graph geometry. The first image is the original image. The last step image is the image after adding one stroke. The middle ones are intermediate steps. (a)(b) from a single-stroke digit 1 to a two-strokes digit 7. (c) from a single-stroke 7 to a two-strokes 2. In (a)(b), the red block on the step 0 image indicates the ZOOM in windows. (d)(e)(f) show the robust feature by modifying the graph value. (c)(d) from a single-stroke 6 to a single-stroke 0. (f) from a single-stroke 7 to a single-stroke 1.

3.2.2 MODIFYING VERTICES AND EDGES VALUE

Other than modifying the graph geometry, we further show that altering vertices/edges value (namely the stroke's control points) will affect the classification output as well. Here, we alter the control point values influencing an image of 6 towards an image as 0. The control points of stroke s are initialized from one sample image of 6. The process is the same as Sec. 2.4.2. We adopt the hard

radius constraint and the soft distance constraint following Eq. 6, where χ in Eq. 6 is the set of all 0 images with only single stroke.

Fig. 3(d) shows the intermediate and final results. At step 0, the image is recognized as 6 with a high confidence score 0.9789. While altering the control points, at step 1000, it is recognized as digit 9, with a confidence score 0.4573, while 0.2552 as 0 and 0.2710 as 6. For the final step, the model recognizes the altered image as 0, with a high confidence score 0.9952. We show experimental results on other input samples in Fig. 3(e) and (f). By altering the control point values affects the classification outputs, thus they are the robust features.

3.3 New digits generation

Here we design and present a novel experimental setting: generating new categorical patterns visually distinguishable from existing categories. The underlying rationale is that if our stroke-based graph representation is with a strong structured expression capability, the underlying space spanned by it after supervised training on existing categories, could guide a generation process to come up with new categorical patterns. In practice, we present a generation process following an iterative manner.

Algorithm 1 The generation of new digits

Initial: Function: $\mathcal{P}, \mathcal{F}, \mathcal{F}_\theta$, Existing sets: V_D, E_D, R_D, starting stroke (parameterized with 10 control points): t and variance σ, Hyper-parameters: $\alpha, \beta_1, \beta_2, \beta_3, \gamma$;
Output: t, σ;
while not converged **do**
  $T \sim \mathcal{N}(t, \sigma)$;
  while not converged **do**
    $V_T, E_T, R_T \leftarrow \mathcal{P}(T)$;
    $\mathcal{L} = -\mathbf{E}[\log(\mathcal{F}_\theta(\mathcal{F}(V_T, E_T))) + \log(1 - \mathcal{F}_\theta(\mathcal{F}(V_D, E_D)))]$;
    $\theta \leftarrow \theta - \gamma \frac{\partial \mathcal{L}}{\partial \theta}$.
  end while
  while not converged **do**
    $v_t, e_t, r_t \leftarrow \mathcal{P}(t)$;
    $T \sim t + \sigma\mathcal{N}(0, 1)$;
    $V_T, E_T, R_T \leftarrow \mathcal{P}(T)$;
    $\mathcal{L} = -\mathbf{E}[\log \mathcal{F}_\theta(\mathcal{F}(V_T, E_T))] - \beta_1 \mathbf{E}[\log p(v_t|V_D)] - \beta_2 \mathbf{E}[\log p(e_t|E_D)] - \beta_3 \mathbf{E}[\log p(r_t|R_D)]$;
    $t \leftarrow t - \alpha \frac{\partial \mathcal{L}}{\partial t}$;
    $\sigma \leftarrow \sigma - \alpha \frac{\partial \mathcal{L}}{\partial \sigma}$.
  end while
end while

For initialization, we form the first set of new digits $T \sim \mathcal{N}(t, \sigma) = t + \sigma\mathcal{N}(0, 1)$, drawing from a Gaussian distribution, where the mean t is the starting stroke (parameterized with 10 control points), and σ is the scale of the Normal distribution noise (we set it to 4 here initially). We consider one stroke case, thus the a graph geometry is fixed. D denotes the set of all existing patterns. Algorithm 1 depicts the iterative procedure, with two alternating steps of optimization.

The first optimization is over a binary classifier $\mathcal{F}_\theta$, separating the set D (labeled as 0) and set T (labeled as 1). For the MPNN network $\mathcal{F}$ (defined in Sec. 2.3), we freeze all parameters. The binary classification objective function is given as:

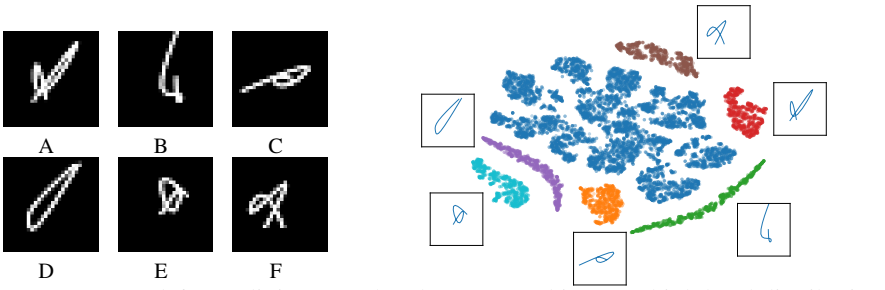
$$\min_{\mathcal{F}_\theta} -\mathbf{E}[\log(\mathcal{F}_\theta(\mathcal{F}(V_T, E_T))) + \log(1 - \mathcal{F}_\theta(\mathcal{F}(V_D, E_D)))]. \tag{9}$$

The second optimization is altering the control points of the starting stroke t and its variance σ, targeting label 1 (following Eq. 5). We adopt the soft distance constraint and the radius constraint given in Sec. 2.4.2. The constraint set χ here is the existing set D. The objective function thus becomes:

$$\min_{t,\sigma} -\mathbf{E}[\log(\mathcal{F}_\theta(\mathcal{F}(V_T, E_T)))] - \mathbf{E}[\log p(v_t|V_D)] - \mathbf{E}[\log p(e_t|E_D)] - \mathbf{E}[\log p(r_t|R_D)]. \tag{10}$$

where $v_t, e_t, r_t = \mathcal{P}(t), T \sim \mathcal{N}(t, \sigma), V_T, E_T, R_T = P(T)$.

In our experiment, we generate a sequence of novel digits with single stroke (hypothetical A-F in a hexadecimal system) from the existing decimal digits, illustrated in Fig. 4(a). It is worth noting that the newly generated digits share a similar visual style of the original MNIST hand-written 0-9s, at the same time visually distinguishable from them. Fig. 4(b) further confirms our claim, as we can see that on the space formed by the final MPNN network $\mathcal{F}$, all novel digits are separable from each other, and are distinguishable from the original set. The new digits generation experiment validates that our SSR-GNNs have a strong structured expression capability.



(a) Generated 6 new digits (b) The generated image's high level distribution

Figure 4: The generation of new sketch images. (a) The new generated digits replacing A-F in hexadecimal system. (b) Projecting each digit's distribution to a 2-dimensional space for visualization

4 Related work

Part-whole visual representation: Recent studies on cognitive science show that the human vision system (HVS) parse visual input into part representation, which is invariant to spatial transformation and viewpoints (Hinton, 2021; Sabour et al., 2018; Singh & Hoffman, 2001). In the literature of Computer Vision and Signal Processing, there are some notable work compositing the part representation. The structural description models (Biederman, 1987; Hummel & Biederman, 1992; van den Hengel et al., 2015; Kodratoff et al., 1984) is one such method, which combines the description of the part components. Particularly for stroke as the part representation, one sketch image can be parsed into parts and sub-parts as strokes (Lake et al., 2015). Both take strokes as the part representation, (Lake et al., 2015) adopts a Bayesian program learning framework, which is an iterative optimization process searching an optimal parse, while our method yields an end-to-end trainable framework.

Graph Neural network: The graph neural network(GNNs) (Duvenaud et al., 2015; Bruna et al., 2013) generalizes the neural network on a graph structured input. The vertex information on the graph is represented by neighborhood aggregation through the graph topology. Message passing neural networks (MPNNs) (Gilmer et al., 2017; Li et al., 2015) is a sub-category of the GNNs. Its convolutional operator directly applies on the spatial graph topology. Since our representation is a stroke-based sketch graph with a flexible number of vertices, we adopt MPNNs as our learning backbone.

5 Conclusion and future work

We present a novel stroke-based sketch representation with graph neural networks (SSR-GNNs). We show that SSR-GNNs are spatially-robust (through robust classification, and robust feature exploration experiments on MNIST and QuickDraw) with a strong structured expression capability (through novel digits generation experiments) and it is a task-agnostic learning framework. The promising properties of SSR-GNNs paves the way for a series of exciting future research, including but not limited to 1) a stroke-based representation learning in an unsupervised manner (a SSR auto encoder); 2) augmenting SSR-GNNs' generalization capability by forming analogies between the graph representations. 3) forming representation of a complicated visual pattern using hierarchical graphs, further improving the structured expression capability.

REFERENCES

Kate Baggaley. There are two kinds of AI, and the difference is important: Most of today's AI is designed to solve specific problems. *Popular Science*, 2017. URL https://www.popsci.com/narrow-and-general-ai. 1

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. 3

Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987. 9

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 9

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 5

Peter Clark. Elementary school science and math tests as a driver for AI: take the ARISTO challenge. In *AAAI*, pp. 4019–4021, 2015. 1

Agnes Desolneux, Lionel Moisan, and Jean-Michel Morel. *From gestalt theory to image analysis: a probabilistic approach*, volume 34. Springer Science & Business Media, 2007. 1

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *arXiv preprint arXiv:1509.09292*, 2015. 3, 9

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pp. 1802–1811. PMLR, 2019. 5

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019. 1

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017. 3, 9

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 4

David Ha and Douglas Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017. 2, 4

Geoffrey Hinton. Some demonstrations of the effects of structural descriptions in mental imagery. *Cognitive Science*, 3(3):231–250, 1979. 1

Geoffrey Hinton. How to represent part-whole hierarchies in a neural network. *arXiv preprint arXiv:2102.12627*, 2021. 1, 9

John E Hummel and Irving Biederman. Dynamic binding in a neural network for shape recognition. *Psychological review*, 99(3):480, 1992. 9

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 6

Yves Kodratoff et al. Learning complex structural descriptions from examples. *Computer vision, graphics, and image processing*, 27(3):266–290, 1984. 9

Nikolaus Kriegeskorte. Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual review of vision science*, 1:417–446, 2015. 1

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 1, 2, 3, 5, 9

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017. 1

Louisa Lam, Seong-Whan Lee, Ching Y Suen, et al. Thinning methodologies-a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 14(9):869–885, 1992. 2

Alex Lamb, Sherjil Ozair, Vikas Verma, and David Ha. Sketchtransfer: A new dataset for exploring detail-invariance and the abstractions learned by deep networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 963–972, 2020. 1

Barbara Landau, Linda B Smith, and Susan S Jones. The importance of shape in early lexical learning. *Cognitive development*, 3(3):299–321, 1988. 1

Hector J Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 2011. 1

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015. 5, 9

Chia-Wei Liao and Jun S Huang. Stroke segmentation by bernstein-bezier curve fitting. *Pattern Recognition*, 23(5):475–484, 1990. 2

Ke Liu, Yea S. Huang, and Ching Y. Suen. Identification of fork points on the skeletons of handwritten chinese characters. *IEEE transactions on pattern analysis and machine intelligence*, 21 (10):1095–1100, 1999. 2

Gary Marcus. Why Can't my computer understand me. *The New Yorker*, 2013. URL https://www.newyorker.com/tech/elements/why-cant-my-computer-understand-me. 1

Gary Marcus. Artificial Intelligence Is Stuck. Here's How to Move It Forward. *New York Times*, 2017. URL https://www.nytimes.com/2017/07/29/opinion/sunday/artificial-intelligence-is-stuck-heres-how-to-move-it-forward.html. 1

Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256, 1972. 5

Sara Sabour, Nicholas Frosst, and Geoffrey Hinton. Matrix capsules with em routing. In *6th international conference on learning representations, ICLR*, volume 115, 2018. 9

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008. 2

Manish Singh and Donald D Hoffman. Part-based representations of visual shape and implications for visual cognition. In *Advances in psychology*, volume 130, pp. 401–459. Elsevier, 2001. 9

Sam Snead. Facebook's AI boss: 'In terms of general intelligence, we're not even close to a rat'. *Business Insider*, 2017. URL https://bit.ly/3vuou7G. 1

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015. 5

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019. 4, 6

Anton van den Hengel, Chris Russell, Anthony Dick, John Bastian, Daniel Pooley, Lachlan Fleming, and Lourdes Agapito. Part-based modelling of compound scenes from images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 878–886, 2015. 9

Peng Xu, Yongye Huang, Tongtong Yuan, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, Zhanyu Ma, and Jun Guo. Sketchmate: Deep hashing for million-scale human sketch retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8090–8098, 2018. 5

Peng Xu, Chaitanya K Joshi, and Xavier Bresson. Multigraph transformer for free-hand sketch recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 5