Supplementary Materials for SketchXAI: A First Look at Explainability for Human Sketches

Zhiyu Qu[1,3] Yulia Gryaditskaya[1] Ke Li[1,2] Kaiyue Pang[1] Tao Xiang[1,3] Yi-Zhe Song[1,3]

[1]SketchX, CVSSP, University of Surrey [2]Beijing University of Posts and Telecommunications
[3]iFlyTek-Surrey Joint Research Centre on Artificial Intelligence

{z.qu, y.gryaditskaya, kaiyue.pang, t.xiang, y.song}@surrey.ac.uk
{like1990}@bupt.edu.cn

1. Existing explainable methods applied to sketches

Fig. 1 visualizes several explainable methods applied to images and sketches. It can be observed that image-based methods are more helpful on images than on sketches: Sketches are sparse and to some extent already represent salient features of a depicted object, therefore the image-based explainable methods tend to highlight most of the sketch.

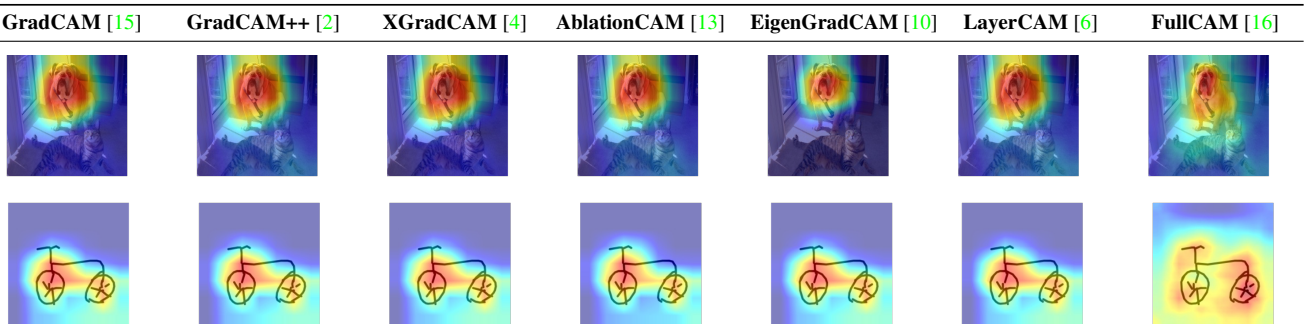


Figure 1. **Comparison of image-based explainable methods applied to images and sketches, which is indicated above each column.**

Fig. 2 adopts text-based explainable methods to sketch inputs. Fig. 1 (left) for the text is informative. In Fig. 1 (right), we plot attention maps, where the color of each point indicates its impact on the final classification decision. However, it might be challenging to make any conclusions from these visualizations, as they look rather noisy. For instance, observe how attention for points on the left and right sides in symmetric objects (church, castle) differ. We think that the main reason is that in this case the explainable component is based on points rather than strokes. In contrast, in our work we operate on strokes and their descriptors: shape, location and order. This allows us to design better visualizations for the explainability of sketch classification tasks.

2. Sketch/strokes representation

There are two popular formats that are used in deep-learning sketch literate [5, 8, 9, 11, 14] to represent vector sketches: "Stroke-3" and "Stroke-5". "Stroke-3" format represents each point as $(\delta x, \delta y, p)$, where $\delta x, \delta y$ store point's relative position to the previous point, and p is a binary pen state (0 means drawing and 1 means end of a stroke). "Stroke-5" format $(\delta x, \delta y, p_1, p_2, p_3)$ has three binary pen states $p_1 - draw$, $p_2 - lift$, $p_3 - end$, which are mutually exclusive. The important difference with our design is that we separate stroke location from its shape, as described in the main paper. Namely, in our case $\delta x, \delta y$ store relative position to the first point of each stroke. Then, we use the second pen state to indicate padding points to account for the fact that all strokes have different number of points. This new representation is key to our SLI settings. We provide a pseudocode in Alg. 1.

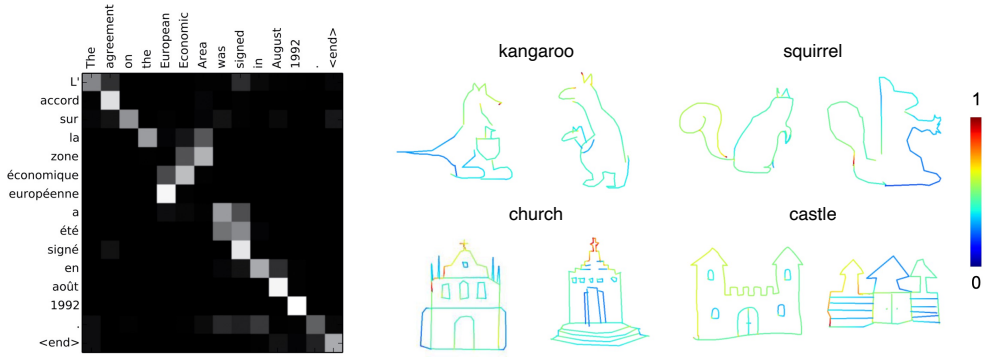


Figure 2. **Comparison of some text-based explainable methods applied to text and sketches.** (**Left**) The classical visualization of alignment between words in Neural Machine Translation [1]. (**Right**) Color-coded attention maps produced by Sketch-R2CNN [7].

Algorithm 1 Pseudocode comparison of the two sketch representation approaches

```
# Previous approach
for sketch in sketch_list:
   vector = sketch.vector_data

   # Calculate offset between all points in each sketch
   new_vector[1:] = vector[1:] - vector[0:-1]
   new_vector[0] = vector[0]
   input(new_vector)

# Our approach
for sketch in sketch_list:
   for order, stroke in enumerate(sketch):
      vector = stroke.vector_data

      # Calculate offset between all points in each stroke
      new_vector[1:] = vector[1:] - vector[:-1]

      # Save the starting point of each stroke (locations -> global information)
      locations_list.append(vector[0])

      # Reset the starting point
      new_vector[0] = [0, 0]

      # After removing the global information, the rest is shape (shapes -> local information)
      shapes_list.append(new_vector)
      order_list.append(order)

   input(order_list, shapes_list, locations_list)
```

3. Experimental setup and hyperparameters

For all ViT experiments, we use the pretrained models from ViT of HuggingFace. Our SLI task for each sketch involves an optimization process, where the only learnable parameters are strokes' locations. As on optimization is performed *per sketch*, the best learning rate is different for each sample. To adapt for the diversity of sketches, we use the `CosineAnnealingLR` scheduler with a maximum learning rate of 10 and a minimum value of 0.00001. In addition, we use `gradient clip` [12] to force a maximum horizontal and vertical stroke movement of 0.5, *i.e.*, a quarter of the canvas size.

The 30 categories that we randomly selected for our experiments are `[airplane]`, `[apple]`, `[baseball_bat]`, `[bed]`, `[bicycle]`, `[book]`, `[bread]`, `[broom]`, `[camera]`, `[car]`, `[cell_phone]`, `[chair]`, `[clock]`, `[cloud]`, `[eye]`, `[eyeglasses]`, `[face]`, `[flower]`, `[headphones]`, `[hot_dog]`, `[laptop]`, `[pants]`, `[shorts]`, `[smiley_face]`, `[snake]`, `[spider]`, `[star]`, `[sun]`, `[table]` and `[tree]`.

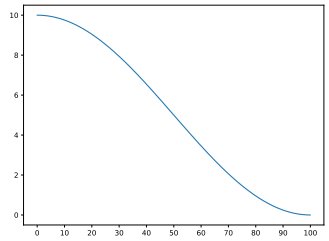


Figure 3. The attenuation curve of learning rate over 100 iterations of `CosineAnnealingLR` scheduler.

4. Analysis of stroke order embeddings

To shed light on the role of stroke order in the decision process of our classifier, we plot stroke order embeddings similarity map, in style of positional embeddings similarity maps of ViT [3].

Fig. 4 right shows the statistic on the number of sketches with a certain number of strokes in the training set. It can be seen, that over ninety percent of sketches have no more than 10 strokes in the QuickDraw dataset. Therefore, we plot similarity maps only for the embeddings of the stroke order numbers from 1 to 16. We break the 16×16 similarity map into 4×4 squares for the visualization clarity (Fig. 4 left). While the pattern is not too pronounced, it can be seen that the earlier strokes are mostly distinctive from each other, while the further you go the more similar the order embeddings become. This can be explained by the fact that as different sketches have different number of strokes, the strokes with larger order numbers participate less frequently in model training. Therefore, the model learns that earlier strokes are more important and their embeddings are more distinguishable from each other.

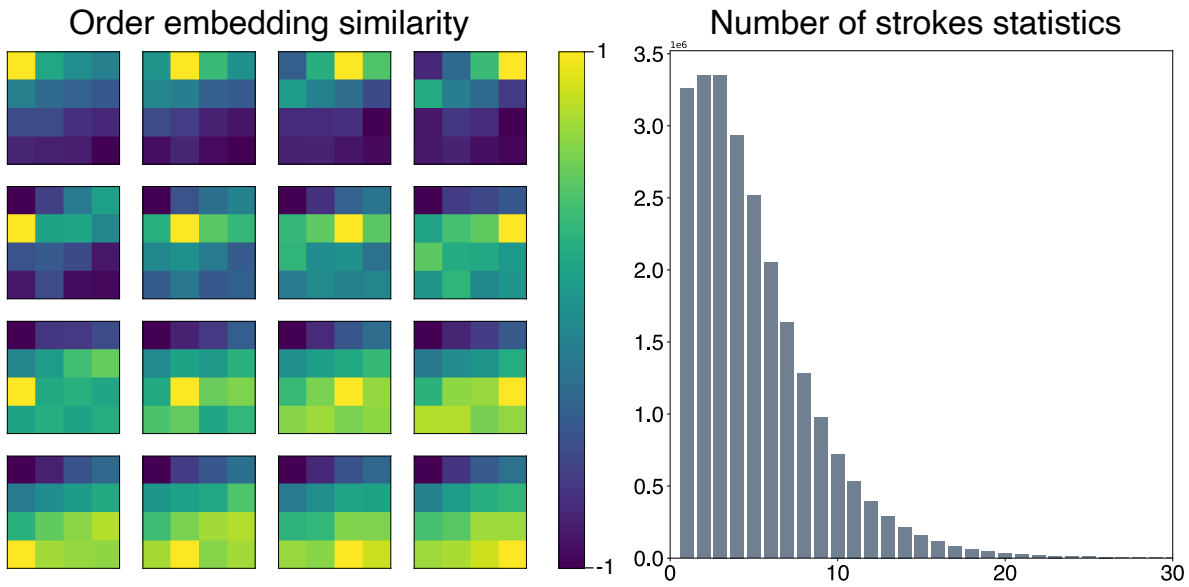


Figure 4. Similarity map of order embeddings and statistics on the number of strokes for all sketches of the training set.

5. Visualizations of shape embeddings

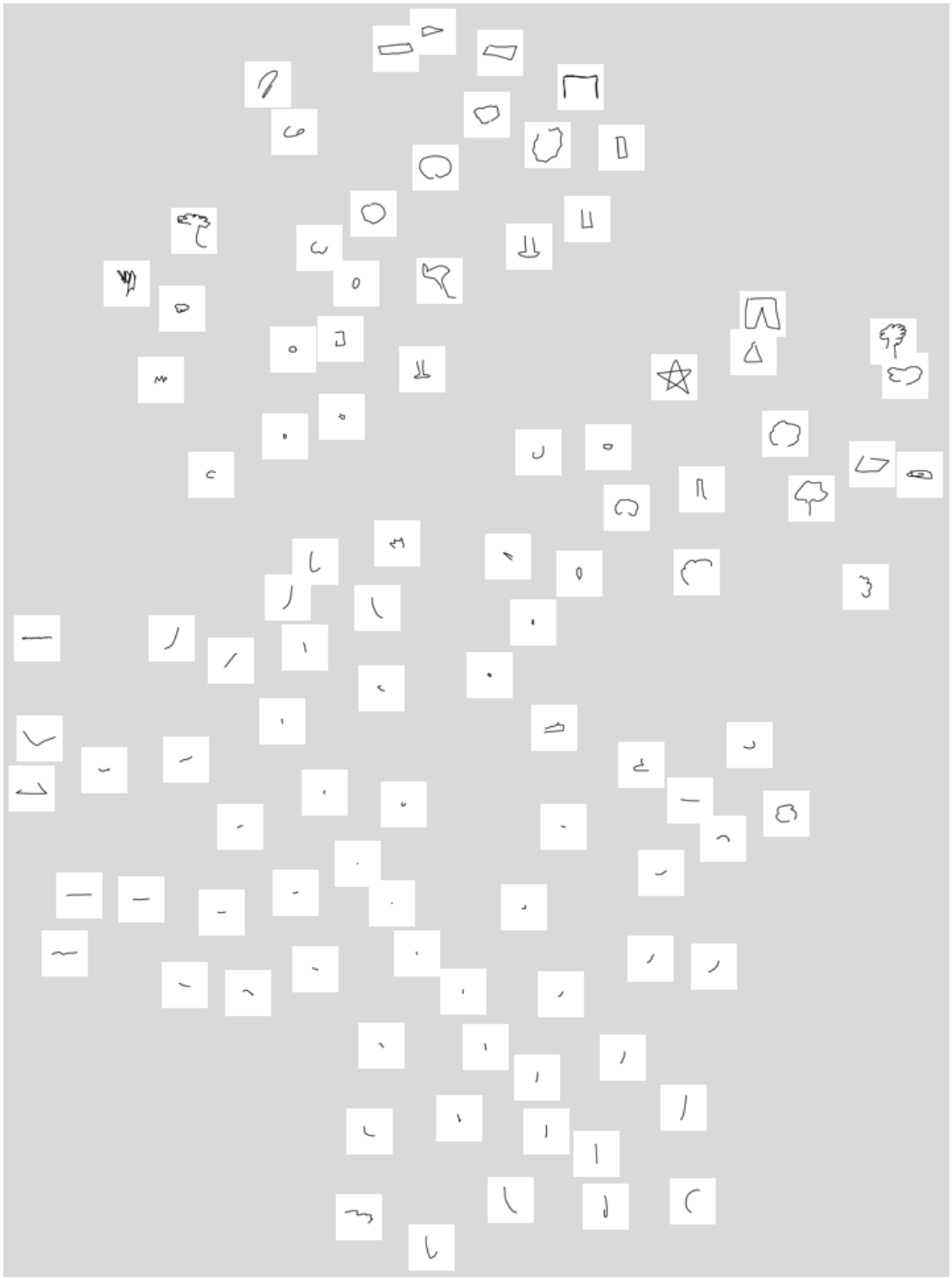
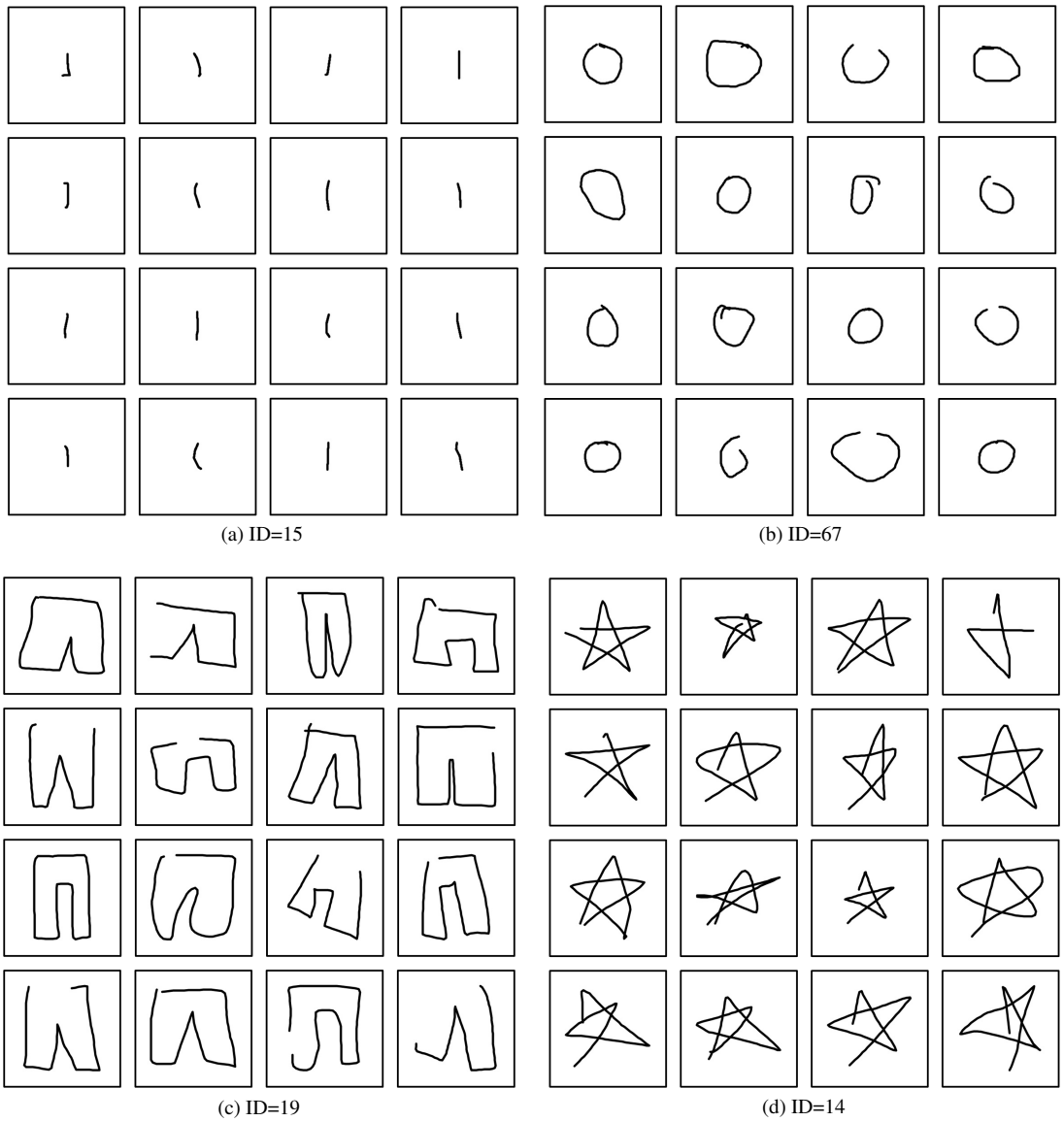


Figure 5. **t-SNE of strokes belonging to 100 clustering centres.** Each of the clustering centres contains hundreds of strokes. In this figure, we select only the nearest stroke to each clustering centre as representative.



(a) ID=15 (b) ID=67

(c) ID=19 (d) ID=14

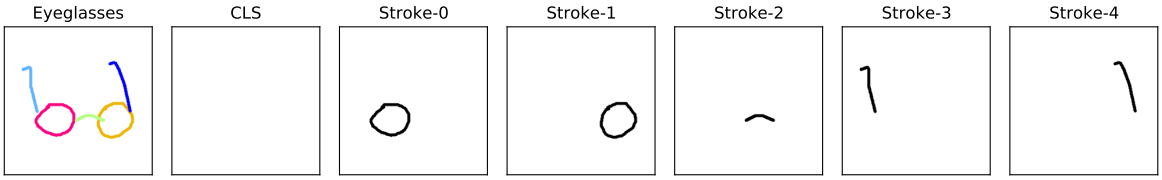
Figure 6. **Visualizations of K-means clustering of shapes.** We randomly select four indices of 100 clustering centres to present in this supplementary material. From the above results we can find that our shape embeddings are learned to group meaningful and consistent semantics of stroke shapes into different indices. Therefore, we can perform stroke replacement tasks based on this property.

6. Effect of different location initialization on explanations

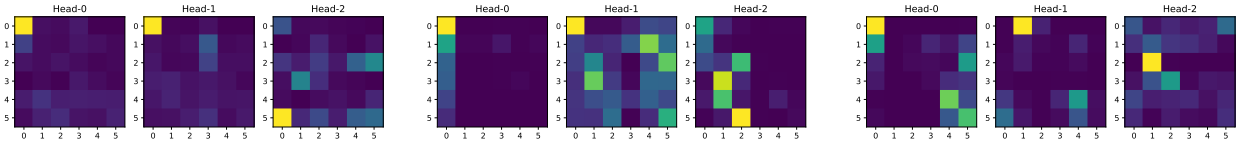
To explore the effect of different initialization locations on explanations, we design the following comparative experiments: (i) Reset location information of all strokes to 0, *i.e.*, all starting points are placed in the centre of the canvas. (ii) Randomly place strokes according to a normal distribution. In Fig. 7, we find that different initialization locations will affect the entire recovery process and generate different sketches. However, since the shapes of strokes do not change, the final structures of sketches are similar.

Figure 7. Here we show the visualizations of the 100 optimizations of the model inversion. For each sketch, the first row represents the optimization process where the locations are initialized at the centre of the canvas. While the second row represents the optimization process with random initialization of locations. The number in the top-left corner indicates the confidence that the model predicts the current sketch belongs to the original label.

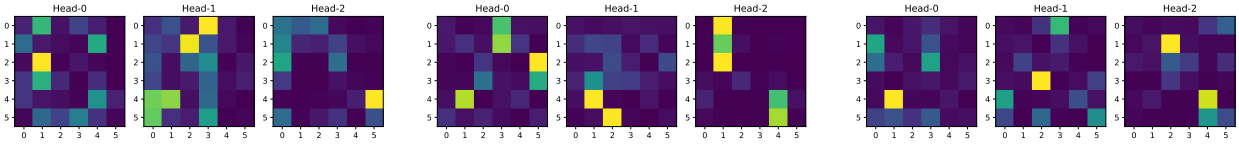
7. Additional visualizations



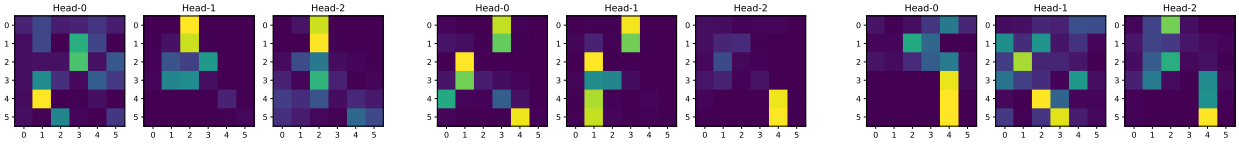
(a) A sketch sample of eyeglasses and the 6 tokens it contains (1 CLS token and 5 stroke tokens).



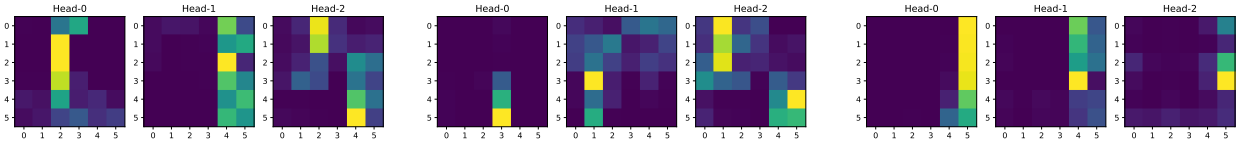
(b) Layer-0 (c) Layer-1 (d) Layer-2



(e) Layer-3 (f) Layer-4 (g) Layer-5



(h) Layer-6 (i) Layer-7 (j) Layer-8



(k) Layer-9 (l) Layer-10 (m) Layer-11

Figure 8. Visualisation of attentions in different layers of Transformer Encoder between strokes.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. 2

[2] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *WACV*, 2018. 1

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 3

[4] Ruigang Fu, Qingyong Hu, Xiaohu Dong, Yulan Guo, Yinghui Gao, and Biao Li. Axiom-based grad-cam: Towards accurate visualization and explanation of cnns. In *BMVC*, 2020. 1

[5] David Ha and Douglas Eck. A neural representation of sketch drawings. In *ICLR*, 2018. 1

[6] Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei. Layercam: Exploring hierarchical class activation maps for localization. *IEEE Transactions on Image Processing*, 2021. 1

[7] Lei Li, Changqing Zou, Youyi Zheng, Qingkun Su, Hongbo Fu, and Chiew-Lan Tai. Sketch-r2cnn: An attentive network for vector sketch recognition. *IEEE Transactions on Visualization and Computer Graphics*, 2020. 2

[8] Hangyu Lin, Yanwei Fu, Xiangyang Xue, and Yu-Gang Jiang. Sketch-bert: Learning sketch bidirectional encoder representation from transformers by self-supervised learning of sketch gestalt. In *CVPR*, 2020. 1

[9] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. In *CVPR*, 2019. 1

[10] Mohammed Bany Muhammad and Mohammed Yeasin. Eigen-cam: Class activation map using principal components. In *IJCNN*, 2020. 1

[11] Umar Riaz Muhammad, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Learning deep sketch abstraction. In *CVPR*, 2018. 1

[12] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *ICML*, 2013. 2

[13] Harish Guruprasad Ramaswamy et al. Ablation-cam: Visual explanations for deep convolutional network via gradient-free localization. In *WACV*, 2020. 1

[14] Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. Sketchformer: Transformer-based representation for sketched structure. In *CVPR*, 2020. 1

[15] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017. 1

[16] Suraj Srinivas and Francois Fleuret. Full-gradient representation for neural network visualization. In *NeurIPS*, 2019. 1