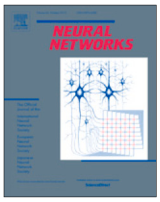
Neural Networks 144 (2021) 627–638

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet



A language modeling-like approach to sketching

Lisa Graziani [a,1], Marco Gori [b,c], Stefano Melacci [b,*]

[a] Department of Social, Political and Cognitive Sciences, University of Siena, Italy
[b] Department of Information Engineering and Mathematics, University of Siena, Italy
[c] MAASAI, Universitè Côte d'Azur, Nice, France

A R T I C L E   I N F O

Article history:
Received 15 February 2021
Received in revised form 17 September 2021
Accepted 21 September 2021
Available online 2 October 2021

Keywords:
Sketch generation
Recurrent Neural Networks
Language Modeling

A B S T R A C T

Sketching is a universal communication tool that, despite its simplicity, is able to efficiently express a large variety of concepts and, in some limited contexts, it can be even more immediate and effective than natural language. In this paper we explore the feasibility of using neural networks to approach sketching in the same way they are commonly used in Language Modeling. We propose a novel approach to what we refer to as "Sketch Modeling", in which a neural network is exploited to learn a probabilistic model that estimates the probability of sketches. We focus on simple sketches and, in particular, on the case in which sketches are represented as sequences of segments. Segments and sequences can be either given – when the sketches are originally drawn in this format – or automatically generated from the input drawing by means of a procedure that we designed to create short sequences, loosely inspired by the human behavior. A Recurrent Neural Network is used to learn the sketch model and, afterward, the network is seeded with an incomplete sketch that it is asked to complete, generating one segment at each time step. We propose a set of measures to evaluate the outcome of a Beam Search-based generation procedure, showing how they can be used to identify the most promising generations. Our experimental analysis assesses the feasibility of this way of modeling sketches, also in the case in which several different categories of sketches are considered.

1. Introduction

Sketching is a powerful and universal form of expression. Its immediateness in representing concepts and, more generally, its capability of being a simple-and-effective mean of communication among humans, has made sketching a popular tool whose origins can be traced back to the ancient times. In the last decade, due the growing ubiquity of touchscreen-based devices, sketching has even become more popular in some specific contexts, such as in communication, design, education (Forbus et al., 2018), games (Sarvadevabhatla, Surya, Mittal, & Babu, 2018), and others. During the same years, Machine Learning-based approaches to sketch-related tasks have been proposed, especially in the context of Deep Learning. For example, we briefly mention tasks of sketch recognition (Ballester & Araujo, 2016; Eitz, Hays, & Alexa, 2012), retrieval (Creswell & Bharath, 2016), segmentation (Sun, Wang, Zhang, & Zhang, 2012).

Deep Learning-based sketch generation has been the subject of recent works, that differ in the way they represent the input data and in the criterion used to generate new samples. Sketches can be represented in several different ways (Xu, 2020): in the usual image pixel space (Liu et al., 2019; Tang et al., 2019; Zhang, Su, Qi, & Yang, 2019), as sets of dynamic strokes (Cao, Yan, Shi, & Chen, 2019; Ha & Eck, 2017; Ribeiro, Bui, Collomosse, & Ponti, 2020; Sasaki & Ogata, 2018; Song, Pang, Song, Xiang, & Hospedales, 2018), or in geometrical graph spaces (Xu, Joshi, & Bresson, 2019). When considering the case in which a photorealistic image is provided to the system, that is expected to output a sketch-image (pixels) that resembles the input, sketch generation can be framed as an image-to-sketch translation problem. Supervised learning (Tang et al., 2019) or unsupervised criteria have been exploited (Zhang et al., 2019), the latter using Generative Adversarial Networks (GANs) and a cycle consistency loss. GANs can also be used to jointly complete a partial sketch and to recognize it in the context of some given categories (Liu et al., 2019).

However, it sounds natural to consider sketching as a dynamic and temporal process, in which the sketch is progressively created step-by-step. Following this intuition, many authors investigated the use of Recurrent Neural Networks (RNNs) with the purpose of handling sequences of primitives that represent sketches. In Graves (2013), Long Short-Term Memories (LSTMs)

* Corresponding author.
E-mail addresses: lisa.graziani@unifi.it (L. Graziani), marco.gori@unisi.it (M. Gori), mela@diism.unisi.it (S. Melacci).

[1] The work was carried out while being affiliated to the Department of Information Engineering, University of Florence, Italy, and to the Department of Information Engineering and Mathematics, University of Siena, Italy.

https://doi.org/10.1016/j.neunet.2021.09.020
0893-6080/© 2021 Elsevier Ltd. All rights reserved.

are used to generate complex sequences with long-range structure, by predicting one data point at a time. The authors of Ha and Eck (2017) represent a sketch as a sequence of "pen" strokes, indicating in which direction to move the pen, when to lift it up, and when to stop drawing. The model, called *sketch-rnn*, is a sequence-to-sequence Variational Autoencoder (VAE), able to complete a partial sketch or to generate similar looking sketches conditioned on the latent space. However, the limitations of this work are that it requires input data acquired in a special way in order to collect the stroke-level representations, and a limited capability of dealing with multi-class generations, since a generator for each category of sketches must be trained. In order to partially overcome these limitations, a VAE-based network has been studied, extending sketch-rnn (Cao et al., 2019). Although there are improvements in the quality of the generations, experimentation is limited to twenty classes of sketches. Some works tried to improve sketch-rnn combining stroke vectorial and pixel-based representation (Chen, Tu, Yi, & Xu, 2017; Sasaki & Ogata, 2018). The model presented in Chen et al. (2017), called *sketch-pix2seq*, enhances sketch-rnn replacing the bidirectional Recurrent Neural Network (BRNN) encoder with a Convolutional Neural Network (CNN) and removing the Kullback–Leibler divergence from the objective function of the VAE. In this approach the strokes are transformed into flat images that are fed to a CNN, creating a latent representation that is combined with the sequence of strokes, and provided to an RNN. Still following the inspiring idea of sketch-rnn, more recently, *sketchformers* were proposed (Ribeiro et al., 2020), to integrate transformer-based architectures into the sketch drawing pipeline. Another work that is related to what we propose, even if more straightforward, is the one of Sasaki and Ogata (2018), in which supervised learning is used to train an RNN that processes stroke coordinates and the pixel-level image. Finally, the authors of Song et al. (2018) are the first ones to propose a sequence-based photo–sketch translation, where sketches are sequentially produced using a recurrent decoder. The model is trained only on two categories of sketches (shoe and chair).

In this paper, we propose a novel approach to what we refer to as "Sketch Modeling", taking inspiration from Language Modeling (LM) (Mikolov, Kombrink, Burget, Černockỳ, & Khudanpur, 2011). In neural-LM, a neural network is trained to allow us to compute a probabilistic model that estimates the probability of a sentence to belong to a target language. The atomic elements of the sentence, usually words (or characters Maggini, Marra, Melacci, & Zugarini, 2019), are symbols of a given vocabulary. Similarly, we consider sketches as sequences of well-defined primitives, that, in the context of this paper, are assumed to be straight segments whose extremes belong to target "vocabulary" of quantized coordinates. We train an RNN (LSTM) to estimate the probability of predicting the next segment of the current sketch, thus allowing us to compute the probability of a sketch to belong to the set of "valid" sketches. We focus on simple sketches, in which the sequences of segments can be either given – when the sketches are originally drawn in this format – or automatically generated from the input drawing by means of a procedure that we designed to create short sequences, loosely inspired by the human behavior. The RNN is seeded with an incomplete sketch that it is asked to complete accordingly to the learned sketch model. Beam Search (Wiseman & Rush, 2016) is used to generate multiple sketch candidates, and we propose a set of measures that evaluates some properties of the sketches in order to identify the most promising generations. Differently from most of the existing works, we depart from the usual regression on the stroke coordinates. Only the recently proposed model of Ribeiro et al. (2020) considers the possibility of quantizing the input data (amongst others) even if mostly emphasizing sketch encoding for downstream applications (classification and retrieval). To the best of our knowledge, we are the first ones to specifically investigate the appropriateness of Language Modeling tools when adapted to build a model of sequences of segments. We assess the feasibility of this novel way of handling sketches using multiple datasets, reusing the same tools that are widely popular in Language Modeling with neural networks. Unlike existing works, our approach does not necessarily need to have access to data collected with specific tools or procedures, since we also propose a procedure to extract segments from raster images of simple sketches. Of course, there are no attempts to propose a model that overcomes the visual quality of some VAE-based and class-specific related works. Since we aim at handling simple sketches composed of a small number of segments, we can efficiently make use of the recurrent-units, showing that we can deal with multi-class data by letting the model implicitly learn also intra-class relationships.

This paper is organized as follows. Section 2 describes related literature in detail, while Section 3 describes the proposed model, remarking its qualities and its limitations. Experimental results are provided in Section 4 and, finally, Section 5 includes our final comments.

2. Related work

In order to better contextualize what we propose with respect to existing literature, we provide more details about those works that, for different reasons, look more related to our ideas, emphasizing the main differences.

In Ha and Eck (2017), a sketch is considered as a vector representation of strokes. The authors propose *sketch-rnn*, a recurrent neural network that is able to construct stroke-based drawings of common objects. A Sequence-to-Sequence Variational Autoencoder (VAE) is used, where the encoder consists of a bidirectional RNN, while the decoder is an autoregressive RNN. A sketch is represented as a sequence of actions controlling the virtual pen that is used for drawing. In particular, a "point" is represented as a vector of 5 elements: $(\Delta x, \Delta y, p_1, p_2, p_3)$, where the first two elements indicate in which direction to move the pen. The last three elements are binary one-hot vectors of three possible states. The first state p_1 denotes that the pen is touching the paper, and that a line will be drawn connecting the next point to the previous one. The other states p_2 and p_3 indicate when to lift the pen up, and when to stop drawing, respectively. Sketch-rnn is able to perform sketch generation, and it can also encode existing sketches into a latent vector, and generate similar looking sketches conditioned on the latent space. The authors trained one category of sketches (or a few categories) at time. Differently from what we propose, sketch-rnn is not designed to support multi-class generations, so that class-specific models are needed. Moreover, the source data is expected to be represented in a format that assumes the use of specific devices to track the drawing procedure, that might not always be available. Finally, due to the selected data encoding, sketch-rnn fails to precisely preserve the relative position between strokes.

Chen et al. (2017) proposed an improvement over sketch-rnn named *sketch-pix2seq*, which could learn and draw multiple categories of sketches. Two modifications were made to improve the sketch-rnn model: the authors replace the bidirectional RNN encoder with a Convolutional Neural Network (CNN); moreover, they reconsidered the role of Kullback–Leibler divergence in the objective function of the VAE. Stroke vectors were transformed into images and fed them to the CNN, training the model on only 6 categories of the QuickDraw database. Our approach involves a larger set of categories, since it tries to capture drawing regularities that might be shared among different sketch categories. Moreover, we show both how to directly process the original sequence of sketch primitives with or without providing to pixel-based representations of the sketch. Amongst the other works that try to overcome the limits of sketch-rnn, we mention *AI-sketcher* of Cao et al. (2019), a deep generative model (VAE) for generating high-quality multi-class sketches, using stroke vectors and pixel-based representation of the strokes (128×128, binary pixels). The strokes are represented as in Ha and Eck (2017) (quintuple). Up to 20 classes of sketches are used, with a conditional model that can differentiate sketches. A CNN is used to extract spatial features from the training sketches, in order to capture the relative positions of strokes. They generate sketches using information coming from all the hidden state values of the RNN encoder.

One of the most recent improvements over sketch-rnn is the one of Ribeiro et al. (2020). A transformer-based representation is exploited, to build a model named *sketchformer*. The model can encode free-hand sketches represented in different ways. Starting from the sketch-rnn-like quintuple, the authors explore a dictionary building approach and a quantized representation that is similar to the one we propose. Four special tokens are added the sketch sequence: a Start of Sketch (SOS) token at the beginning of every sketch, an End of Sketch (EOS) token at the end, a Stroke End Point (SEP) token to be inserted between strokes (indicating pen lifting) and a padding (PAD) token to pad the sketch to a fixed length. Sketchformer is focused on downstream tasks such as sketch classification, sketch based image retrieval, and it also briefly evaluates the reconstruction and interpolation of sketches. However, reconstruction consists in decoding a previously encoded sketch, and not the completion of a sketch that is being drawn. Differently, we focus on bridging Language Modeling and Sketch Drawing, thus studying models that learn sketch regularities from the drawing sequence, being able to eventually complete a partial sketch. Moreover, in order to support the main scope of this paper, our experimental experience is about evaluating our model with the same tools that are used in Language Modeling.

There exist several other works that are still in the context of sketch drawing, but with a different focus with respect to the ones described so far. Sasaki and Ogata (2018) proposed a supervised learning model that can be trained using examples of visuomotor sequences from drawings made by human. The authors show that their approach can associate motion to what is being depicted and it can adapt the drawing behavior for completion purposes. Their dynamical model is implemented by RNNs, exploiting images and "actions", that correspond to the trajectory of a pen and other information on the drawing process. Some works that are about generating sketches from realistic photos also exploit RNNs, in analogy to our approach, even if with a different goal. The work of Song et al. (2018) is the first example of sequential vectorized photo–sketch translation, where sketches are sequentially produced using a recurrent decoder. The model includes four mapping functions: a photo encoder (generative CNN), a sketch encoder (bidirectional LSTM), a photo decoder (conditional CNN), a sketch decoder. In Zhang et al. (2019) an unsupervised approach for image-to-sketch translation is proposed, by means of a cycle consistency loss and exploiting the edge cue from real image. These "translation" models are usually trained on a single class of sketches or on a few classes, since their goal is to capture those regularities that allow the machine to convert a photo-realistic picture into a sketch, and that is different from what we discuss in this paper.

3. Model

When thinking about the sketch drawing process, it is immediate to describe a sketch as a set of strokes with different properties (length, curvature, thickness, ...). We can reduce the degrees of freedom in such descriptions by representing sketches as a *sequence* of *primitives* of different types, such as a sequence of straight lines, circles, squares, of any other kind of primitives one might think of. Considering a *sequence* appears pretty natural, since the drawing process has a natural sequential organization due to the way the human artist works and it plays an important role in whole process. Think about primitives with different colors that partially overlap: changing the order of the drawing sequence would result in differently colored sketches. Even if we discard the color information, the drawing process has a natural sequential organization due to the way the human artist works. In this paper, a sketch s can be described as a sequence of line segments,

$$s = [q_1, q_2, \dots, q_T] \tag{1}$$

where q_t is a segment, and T is the length of the sequence. Each segment is described by the coordinates of its endpoints, $(x_1, y_1), (x_2, y_2)$, respectively, where $x_1, x_2 \in [1, h]$ and $y_1, y_2 \in [1, w]$, being $w \times h$ the size of the paper or the resolution of the media on which the sketch is drawn. While the choice of the primitives could be arbitrary, we are motivated by the fact that the most simple sketches are commonly based on line segments or on shapes that can be reasonably approximated by segments, including those based on curved lines. Differently, if we consider other primitives, such as circles or ellipses, they can more hardly be adapted to approximate lines. In Section 3.3 we will describe how to extract line segments in an automatic manner from pre-designed simple sketches that include arbitrary shapes.

In Language Modeling (Mikolov et al., 2011), if w_t is a word belonging to vocabulary V, the probability that a given sentence $w_1, w_2, \dots, w_n$ belongs to the target language is factorized as $p(w_1, \dots, w_n) = \prod_{t=0}^{n-1} p(w_{t+1}|w_1, \dots, w_t)$. In this paper we propose what we refer to as "Sketch Modeling", in which, in analogy with Language Modeling, the probability of a sketch s to belong to the set of *valid* sketches is

$$p(s) = p(q_1, \dots, q_T) = \prod_{t=0}^{T-1} p(q_{t+1}|q_1, \dots, q_t), \tag{2}$$

where a sketch is valid if it does seem to have some regularities that make it look like a representation of some concrete entities, while it is not valid if it has no apparent regularities.[2] However, the discrete representation of words in vocabulary V does not match the real-valued representation of the segment endpoints. We propose to quantize the segments by gridding the drawing area, thus generating a partitioning of the $w \times h$ paper into equally spaced cells. Without any loss of generality, in order to simplify the descriptions we assume that the drawing area is squared, and that we have g cells in each dimension, thus generating a *cell-vocabulary* G of $g \times g$ elements. It is trivial to map each segment endpoint into an element of G, so that we can define each segment as $q_t = [\alpha_t, \beta_t]$, where $\alpha_t \in G$ and $\beta_t \in G$ are the symbols associated to the endpoints. As a result, each segment is a symbol belonging to $G \times G$. Every time that we need to map back a symbol of G to real-valued coordinates, we consider the coordinates of the center of the grid cell associated to the symbol.[3] Each sketch is also augmented with a start-drawing and an end-drawing signal, that we instantiated by adding two special symbols to G, thus increasing the size of G to $|G| = g^2 + 2$. The start-drawing signal is $[g^2 + 1, g^2 + 1]$, while the end-drawing

[2] When $t = 0$ we only have $p(q_1)$, of course

[3] One could also think of building a vocabulary of segments instead of a vocabulary of endpoints. However, in that case each segment would be an independent symbol and, for example, two segments that share an endpoint would not be different from two segments that do not share any endpoints.

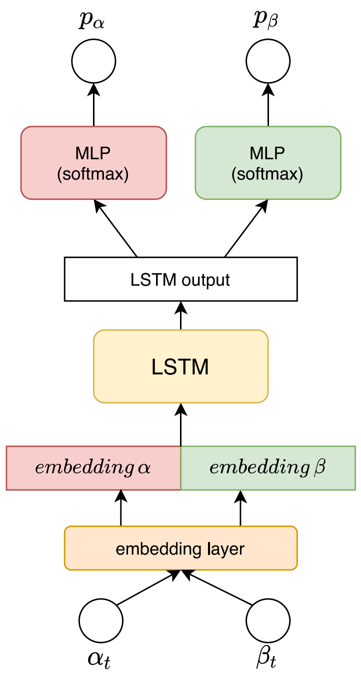


Fig. 1. Architecture of the sketch modeling network: α_t and β_t are the extremes of the tth segment while p_α and p_β are the probability distributions over the endpoint locations of the segment at time $t+1$.

signal is $[g^2+2, g^2+2]$ (one-based indexing). In this way, a sketch is represented with a matrix of dimension $(T+2)\times 2$, as formalized in the following,

$$\begin{bmatrix} g^2+1 & g^2+1 \\ \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \\ \vdots & \\ \alpha_T & \beta_T \\ g^2+2 & g^2+2 \end{bmatrix}$$

where the endpoints of T segments are reported.

We designed a neural architecture that implements $p(q_{t+1}|q_1, \ldots, q_t)$ of Eq. (2). Due to our choice of breaking the segment-level representations into the representations of the two endpoints of the segment, $q_{t+1} = [\alpha_{t+1}, \beta_{t+1}]$, we have that $p(q_{t+1}|\cdot) = p_\alpha(\alpha_{t+1}|\cdot)p_\beta(\beta_{t+1}|\cdot)$, and our network is modeled to yield two probability distributions of size $|G|$, i.e., p_α and p_β, respectively. The first one models the likelihood of each symbol in G to be the first endpoint of q_{t+1}, while the second one is about the second endpoint of q_{t+1}. Fig. 1 shows the structure of the network. The input symbols at time t, i.e., α_t and β_t, are first embedded into dense representations and concatenated, then a Long Short-Term Memory (LSTM) recurrent neural network (Hochreiter & Schmidhuber, 1997) computes in its hidden state an embedding of the sketch drawn up to time t. Finally, given such hidden representation of the sketch (named LSTM output in Fig. 1), two Multi Layer Perceptrons (MLPs) are used to compute p_α and p_β. Each MLP has $|G|$ output units with softmax activation, to ensure a probabilistic normalization of its outputs. For the sake of simplicity, we overload the notation p_α and p_β to also refer to the functions computed by the neural architecture, i.e., $p_\alpha(\alpha_t, \beta_t|E, W, W_1)$, $p_\beta(\alpha_t, \beta_t|E, W, W_2)$, where E, W, W_1, W_2 are the weights of the network, distinguishing between the weights in the embedding layer (E), the weights of the LSTM (W), and the weights of the two MLPs (W_1, W_2). We notice that Transformers (Vaswani et al., 2017) could be used as well to implement our ideas, since they have been shown to lead to large quality models of language (Brown et al., 2020; Radford, Narasimhan, Salimans, & Sutskever, 2018). We investigated the case of LSTMs mostly due to their longer tradition in the Language Modeling literature, and their wide diffusion in the Machine Learning community since several years.

Training the Sketch Modeling system can be done following the exact same schema of Language Modeling by neural networks. After having processed t segments from a ground truth sketch, the α_{t+1}th output component of the first MLP (p_α) is pushed toward a 1, and the same is done to the β_{t+1}th output unit of the second MLP (p_β), thus favoring the emission of the segment $q_{t+1} = [\alpha_{t+1}, \beta_{t+1}]$ that is coherent to the one of the ground truth. In the case of Language Modeling, words that are synonyms or that are about somehow related terms (for example, terms that are used in similar contexts) should yield similar embeddings. The strength of the term-to-term relationships are not always easily defined in advance, and are frequently learned from text corpora (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). This comment still holds, even with a lesser degree, in the case of Sketch Modeling, in which the symbols that the system emits implicitly share a strong and known-in-advance spatial relationship that is given by the position of the cells to which they correspond. In other words, cells that are nearby are expected to be embedded into similar vectors. This property can be explicitly enforced when training the system. Let us define with $e_{i,j}$ the embedding vector of the cell in position $\{i,j\}$ of the grid, while the already introduced notation E indicates the set that collects all the cell-embeddings. The following regularization term $\mathcal{R}(E)$ is zero when nearby cells share the same embeddings, and it will be enforced in a soft-manner into the learning problem,

$$\mathcal{R}(E) = \frac{1}{g^2}\sum_{i=1}^{g}\sum_{j=1}^{g}\sum_{(h,k)\in \mathtt{neig}(i,j)} \|e_{i,j} - e_{h,k}\|_2^2, \tag{3}$$

where $\|\cdot\|^2$ is the squared L_2 norm. We used the notation $\mathtt{neig}(i,j)$ to indicate the set of coordinates of the cells in the immediate neighborhood of cell at (i,j), excluding out-of-grid elements (we have 8 cells at most in each neighborhood). Of course, different configurations of the neighborhoods could be considered as well. While computing the regularizer is $O(g^2)$, a coarse grid (small g) is enough to get good results that match the expected precision of a sketch based on simple primitives such as segments. Moreover, Eq. (3) could be optimized in stochastic manner, randomly subsampling a sub-portion of the (i,j)-pairs at each step of the gradient-based optimization.

Given N training sketches collected in set $\mathcal{D}$, the loss function that must be minimized with respect to E, W, W_1, W_2 in order to train the system is composed of two cross-entropy losses, each of them indicated with $\mathcal{L}$, that measure the mismatch between p_α and the expected target and of p_β and its target, respectively, and by the spatial regularity term of Eq. (3), weighted by the scalar $\lambda_e > 0$,

$$\mathtt{loss}\,(\mathcal{D}, E, W, W_1, W_2) = \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=1}^{T^{(i)}}\left(\mathcal{L}\left(p_\alpha\left(\alpha_t^{(i)}, \beta_t^{(i)}|E, W, W_1\right), \alpha_{t+1}^{(i)}\right) + \mathcal{L}\left(p_\beta\left(\alpha_t^{(i)}, \beta_t^{(i)}|E, W, W_2\right), \beta_{t+1}^{(i)}\right)\right)\right] + \lambda_e\mathcal{R}(E), \tag{4}$$

where the superscript (i) indicates the ith training example.

A trained Sketch Model can be naturally used to generate sketches in the same way a Language Model is used to generate text (Dong et al., 2019). In the most simple case, once a partially drawn sketch is fed to the network (i.e., some initial segments),

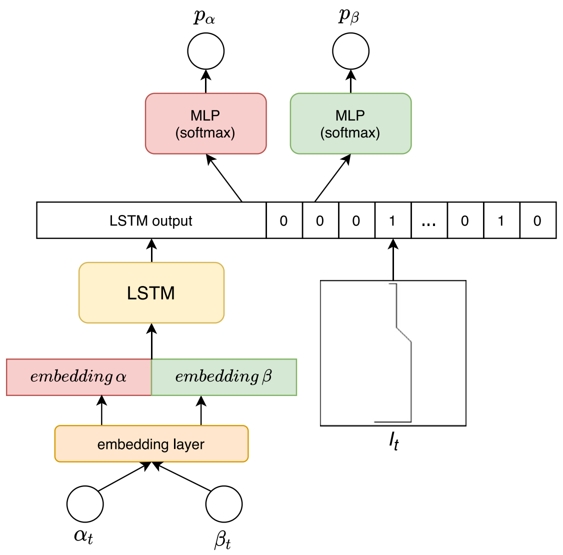


Fig. 2. Model augmented with an explicit representation of the sketched segments. The LSTM output (hidden state) is concatenated with a binary vector that represents a flattened view of a $g \times g$ image I_t with the sketch drawn up to time t. Such vector does not store any information on the order in which segments were sketched.

taking as α_{t+1} the index of the cell that yielded the highest probability in p_α and as β_{t+1} the winning index of p_β, immediately leads to the next segment to drawn, $q_{t+1} = [\alpha_{t+1}, \beta_{t+1}]$. Of course, Beam Search (Wiseman & Rush, 2016) can also be used to generate multiple sketches that are coherent with the Sketch Model. We consider the generation completed whenever the next segment has one of the endpoints associated to the stop symbol.

3.1. Explicit representation of the sketched segments

A critical issue with the proposed Sketch Modeling network is that the role of memorizing the information of the sketch drawn up to time t is only played by the hidden state of the LSTM. However, differently from the case of Language Modeling, the sequence of symbols processed by the LSTM has a clear correspondence with what is drawn in a 2D paper. For this reason, once the segments collected up to time t have been rendered on a $g \times g$ image I_t, such image represents an alternative and complete representation of the outcome of the actions executed up to time t.[4] We can exploit I_t to augment the Sketch Modeling network described so far. In particular, we flattened I_t into a binary vector with g^2 components, that is concatenated with the hidden state of the LSTM, thus providing an extended representation to the MLPs, as shown in Fig. 2.[5] This augmented representation allows the MLPs to leverage both from the LSTM representation, that kept track also of the order in which the sketch was drawn, and of the image-based input I, that has no information on the order but that compactly represents with no-loss what has been drawn so far.

[4] $g \times g$ is the minimal resolution to have a 1-to-1 correspondence between the grid cells and the pixels in I_t, that is the reason why we selected it.

[5] Convolutional Neural Networks could be used as well, avoiding to flatten the image representation. We selected the simplest solution, also considering that g is relatively small in our experiments.

3.2. Beam search-based generation with automatic selection

Once we provide a trained network a sequence $\tilde{s}$ of an incomplete sketch, the network can complete the sketch either in a static way, keeping the most promising segments at each time step or exploring multiple generations with the Beam Search algorithm (Wiseman & Rush, 2016), as already anticipated. The latter solution provides more flexibility to the generation procedure that, however, is still plagued by the crucial issue of selecting what sequences of the beam are the ones that yield a *qualitatively* better-looking sketch, that might not be the one associated with the largest probabilities — as we also experienced.

We propose two measures to be used as preference criteria in the Beam Search procedure to filter out the candidate generations and select the most promising ones. These measures are not related to the generation probabilities and they involve some specific aspects of the sketches that are not enough to guarantee a qualitatively good result, but that are certainly related to the most promising generations (small values of these measures usually imply better sketches). Realistic sketches are not a single long chain of segments, and the proposed model is general enough to handle both connected sequences of segments and fully disconnected segments, since there is no explicit constraint that enforces β_{t+1} to be equal to α_t. Of course, a sketch in which each segment is, on average, drawn far away from the previous one might be the outcome of a bad generation. Motivated by this consideration, we introduce the $\texttt{chain}(s)$ function, that returns a score that is smaller in those sketches in which, on average, consecutive segments are connected, meaning that the second endpoint of segment at time t is close to the first endpoint of segment at $t + 1$. It returns zero when all the segments yield a long chain-like pattern. The distance among endpoints, $d(\cdot, \cdot)$, is the Euclidean distance computed mapping back the discrete representations to real coordinates,

$$\texttt{chain}(s) = \frac{1}{T-1} \sum_{t=1}^{T-1} d\left((x_2, y_2)_t,\ (x_1, y_1)_{t+1}\right). \tag{5}$$

The second measure that we propose to select the most promising candidates of the Beam Search procedure considers that the drawn segments, overall, should occupy a large portion of the available “canvas”. This measure is motivated by the fact that a sketch that covers a small fraction of the canvas might be the outcome of a failed generation. When providing a paper to a human artist that is expect to create a sketch in such a paper, it is very reasonable to expect the human to use a large portion of that paper. It would be very disappointing to see a sketch that is drawn at a very small scale or in a small area close to a corner of the paper. For this reason we propose to check the value of the following function, named $\texttt{density}(s)$, to filter out not-promising candidates. In detail, after having rendered the sketch on a $g \times g$ image I, $\texttt{density}(s)$ returns an index in $[0, 1)$, that is the fraction of locations that are not covered by the sketch,

$$\texttt{density}(s) = 1 - \frac{\#[I = 0]}{g^2}, \tag{6}$$

where $\#[I = 0]$ is the number of drawn cells. The role of this measure should be also jointly analyzed with another consideration. It might sound natural to discard too short sequences, that might be due to a too early generation of the stop-drawing signal. However, this would completely ignore the case of those sketches that are indeed composed of a few segments, but they are still significant enough to be considered valid. Imagine the case of the body of a person sketched by a child, in which the body is a simple line, as well as legs and arms, while the head is a triangle, thus 8 segments in total. If it is well covering the drawing area, that it can be considered a valid sketch, independently on the short length of the sequence. Now consider the sketch composed of 8 (or many more) tiny segments depicted in a very small area of the canvas. They might be about the previous person at a too small scale or about some other good or weird sketches, that, in any case, are too small to be really appreciated. The density measure might help also in these cases.

The two proposed measures implement heuristic criteria that are not expected to hold in all the cases. Advanced principles could be also followed (Todorovic, 2008), more strongly related to the quality of the resulting sketch. The scientific literature includes existing techniques that specifically consider the quality of the curves for hand-drawn sketches (Baran, Lehtinen, & Popović, 2010), also to help in freehand sketching (Fišer, Asente, & Sỳkora, 2015). These techniques could be considered for setting up more detailed selection criteria for Beam Search. However, this would increase the complexity and computational time needed to evaluate the generated sketches, since they involve multiple primitives and geometric relations. What we propose is simpler and fast to compute, making the proposed measures very useful to filter out failed candidate generations.

3.3. Segment extraction and data preprocessing

When a sketch is drawn in a digital media, explicitly using segments as basic drawing primitive, then generating the data for the Sketch Modeling network is straightforward. However, if the sketch has been drawn on a piece of paper (i.e., on a non digital canvas) or if it has been drawn without any special limit in the type of strokes, then (1) it must be converted into a sequence of segments and (2) the order of the strokes, if not available, must be artificially defined.

(1) In this work, we followed a procedure that is based on classic image processing techniques an, in particular, on extracting the contours of the sketched data. The Canny edge detector (Canny, 1986) is exploited to identify the edges of the strokes, that are then used in the contour-detection algorithm of Suzuki et al. (1985) to retrieve several contours.[6] The algorithm detects multiple contours on the edge-based representation. We discarded contours that cover small areas, that, in most of the cases, we found to be due to mistakes in the detection procedure (in some cases this might end up in ignoring small details of the original picture). In order approximate the contours with a reduced set of segments, we exploited the Douglas–Peucker algorithm (Douglas & Peucker, 1973), that can represent open or closed contours by means of a reduced set of segments. The algorithm depends on a customizable approximation level $\epsilon > 0$, that is the maximum distance between the original curve and its approximation (i.e., largest values of ϵ yields less detailed approximations).[7]

(2) The extracted segments are not ordered. We defined a simple criterion that is inspired by what a human would do when drawing a simple sketch using straight lines (without any claims of closely approximating a human artist). The outer/larger contours are drawn first, and, more generally the sketch is drawn starting from the bigger contours and then moving toward the smallest one, i.e, the details. We ordered contours with respect to their areas, largest areas first. Then, for each contour, we heuristically assumed that the segment with the leftmost endpoint is drawn first, simulating a left-to-right sketching style. The contour is traversed in counter clockwise manner, progressively accumulating the segments that approximate the contour.

The segment data are normalized so that a virtual square "canvas" is fully occupied by the sketch, either in the horizontal or in the vertical direction (the sketch is centered in the other direction). In Fig. 3 (top) we show some examples of original sketches and the ones obtained after having applied the proposed procedure. In the bottom part of Fig. 3, we consider the case of the human-created sketch of a penguin (a) and different outcomes of the segment extraction procedure, varying ϵ (b,c,d). In case (b), a large ϵ is selected, so that the approximation quality of the curved contours is pretty low and a small number of segments (13) is exploited. In (c), ϵ is reduced, yielding a better segment-based sketch (47 segments) that more closely follows the original curves of the sketch. The result of (d) is based on an even smaller ϵ and it consists of 195 segments, even if it does not look so different from the previous case.

3.4. Limitations

The proposed approach to sketching offers a clear interpretation that can be easily connected to what happens when facing Language Modeling tasks. Reusing tools that are well-established in other research directions allows our method to be easily understood and implemented, but it also inevitably inherits some limitations. The length of the sequence that yields the sketch can become pretty large in detailed sketches, if compared to the average length of sentences that is commonly considered in most neural network-based Language Models (Mikolov et al., 2011). This is one of the reasons why, in this paper, we considered simple sketches. We decided to focus on the case of Recurrent Neural Networks (LSTMs) in order to use popular tools that are very well-known to the Machine Learning community, being them based on a largely diffused computational model that naturally copes with sequential data. On the other hand, significantly large neural models for Language Modeling or related tasks have been recently shown to perform extraordinary well also in case of longer term dependencies (Brown et al., 2020), especially when different architectures are considered, such as Transformers (Radford et al., 2018; Vaswani et al., 2017), thus opening for the application of what we propose to more advanced drawings. Another limitation of our approach comes from the choice on the type of primitive that is considered in the sketching process. In this paper we analyzed the case of segments, mostly motivated by their simplicity and their versatility. Multiple segments can be used to approximate curved contours, with different levels of quality. This implies that our segment-based model is pretty flexible in terms of what can be drawn. Of course, this choice has also some drawbacks, since a large number of segments might be needed to get very detailed approximations, that is in contrast with the sequence length limitations we just discussed. However, what we propose is general, and other primitives could be explored, in order to draw sketches with different styles. Also the case in which multiple primitives are considered can be instantiated in the proposed framework with minor adaptations, and it represents an interesting direction for future work.

[6] We used the OpenCV (Python) implementation of the contour detection algorithm of Suzuki et al. (1985) with the following parameters `cv2.findContours(edges, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE)`, where `edges` is the outcome of the edge detection procedure.

[7] We also evaluated line-based extractors that use the Probabilistic Hough Transform (Matas, Galambos, & Kittler, 2000). However, after several post-processing operations and manual tuning attempts, the resulting segments were mainly disconnected and with a large variability even in similar sketches, making it hard for the network to learn regularities. We found the contour-based extraction to be more appropriate.

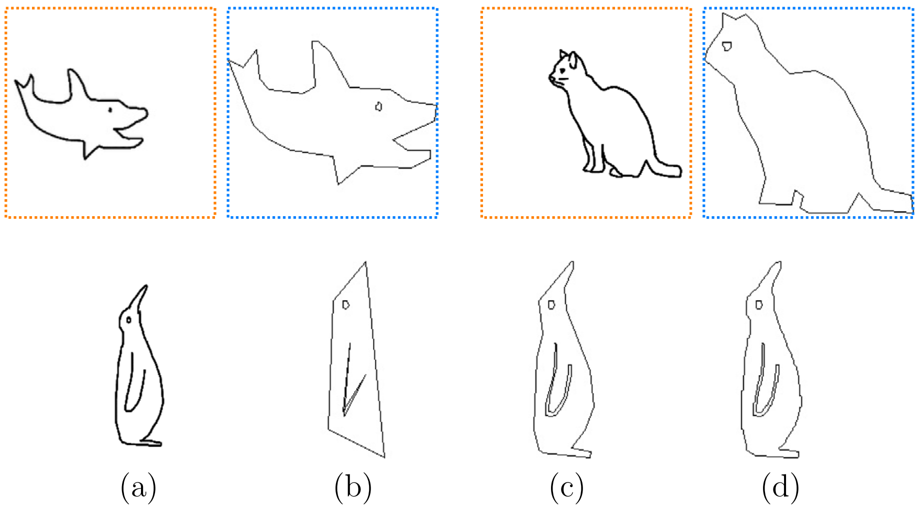


Fig. 3. Top: real-world sketches created by a human (orange boxes) and segment-based representations, automatically generated and normalized as described in Section 3.3 (blue boxes) – the input of the Sketch Modeling network. Bottom: a sketch created by a human (a) is automatically converted into segment-based representations (b,c,d), using decreasing values of ϵ (see the paper text for details).

4. Experimental results

We assessed the quality of the proposed model in Sketch Modeling (Section 4.1) and Generation (Section 4.2), exploiting popular sketch databases and an ad-hoc collection of emotion-related sketches of human faces. The goal of our experiments is to confirm the validity of the proposed idea of bridging Sketch Modeling and Language Modeling, approaching the former with the same neural architectures and tools of the latter.

4.1. Sketch modeling

We considered two datasets that are frequently used in related works: the Sketchy Database (Sangkloy, Burnell, Ham, & Hays, 2016) and the Quick Draw Dataset (Ha & Eck, 2017). The Sketchy Database consists of 125 categories of sketches. The overall number of sketches is 75 471. The Quick Draw Dataset (Ha & Eck, 2017) is a collection of 50 million drawings across 345 categories, created by players of the online game *Quick, Draw!*. Notice that drawings are about sketches that are not composed of line segments only, and they were not designed having our approach in mind, thus making our experimental setup very challenging. We automatically pre-processed the data with the procedure of Section 3.3. In order to avoid too long sequences, we also filtered out the results, keeping only sketches with less than 50 segments.

In the first set of experiments, we considered only the pre-processed Sketchy Database data, that was divided into training (12 912 examples), validation (3107 examples) and test set (3107 examples). Then, in the second set of experiments, we also added data coming from the pre-processed Quick Draw Dataset. In particular, we considered a subset of randomly selected categories, ensuring to keep 2400 sketches for each of them after the pre-processing stage. This allowed us to collect enough examples to create an augmented training set of $\approx$ 150,000 examples (2000 new examples per Quick Drawn category) and validation and test sets of $\approx$ 1500 examples (200 new examples per Quick Draw category).

We considered different configurations of the architecture of Fig. 1, varying the embedding size in $\{10, 50, 100, 150\}$, the size of the hidden state of the LSTM in $\{50, 100, 200, 500\}$, the number of hidden layers (in $\{0, 1, 2, 3\}$), hidden units per layer (in $\{100, 200, 300\}$), and activation functions (hidden units) of the MLPs (in {ReLu, sigmoid, tanh}), and different values of λ_e

Table 1
Final configurations selected for each model. T is the maximum number of segments, g the number of cells in each dimension of the grid.

Param/Model	SM net w/wo sketch image	FF net sketch image
λ_e	10^{-4}	–
T	50	50
g	20	20
Cell embedding size	50	–
LSTM hidden state size	200	–
Layers in MLP	2	6
Neurons in MLP (per layer)	300, 402	500, 600, 700, 600, 500, 402
Activ. in MLPs (per layer)	ReLu, Softmax	ReLu (×5), Softmax
Learning rate	0.001	0.001
Batch size	128	128

(Eq. (4)) in $\{10^{-6}, 10^{-4}, 10^{-2}, 0.1\}$. We evaluated a large number of randomly selected configurations inside the reported sets of values training the model in the Sketchy Database, and then we manually fine-tuned the parameter values starting from the best found configuration in order to maximize a selected accuracy measure (defined below) on the validation set, including the learning rate of the Adam optimizer (starting value was 0.001). The final configuration we selected consists of $\lambda_e = 10^{-4}$, embedding size 50, LSTM hidden state size 200, MLPs with 2 fully connected layers, with 300 hidden neurons and ReLu activations. Table 1 is a quick reference to the optimal parameters for each of the considered models.

We trained the model for up to 100 epochs. In order to reach improved performance, we defined a training data augmentation procedure that is based on a random-small perturbation of the endpoints of each segment (ensuring to keep the connection among those segments that are sharing an endpoint). We considered $g = 20$ cells per edges and, after having mapped back the cell elements to real coordinates, we added some random noise sampled from a uniform distribution in $[-0.05, 0.05)$, in order to ensure that a point is shifted by a single cell at most. We used mini-batches of size 128, doubling their size with the augmentation procedure.

When evaluating the quality of the network in Sketch Modeling, we followed the exact same setting that is used when evaluating Language Models, thus analyzing the quality of the system in predicting the next segment after having processed

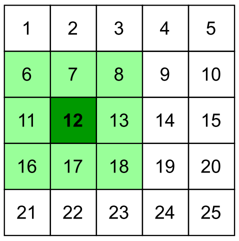


Fig. 4. Example of the principle behind the SOFT-ACC accuracy. If cell 12 is the target one that we expect the system to predict for a segment endpoint, we consider correct the prediction if it is about cell 12 or one of the 8 surrounding cells. Here we represented a 5 × 5 canvas for the sake of simplicity.

the portion of sketch that was drawn so far, i.e., the quality of $p(q_{t+1}|q_1, \dots, q_t)$, for $t \geq 2$ (see Eq. (2)), with sequences of maximum length equal to 50 (discarding the start-drawing and end-drawing signals).[8] In our work, such probability distribution is broken into two distributions p_α and p_β (Section 3).

We considered four different measures of quality. (*i.*) The first one is fully inherited by Language Modeling approaches, and it consists in the PERPLEXITY score (lower perplexity indicates a better quality in predicting the next segment). We averaged the perplexity computed in p_α and p_β. (*ii.*) The second measure is the classic accuracy in predicting the next segments. The system is enforced to take a decision on the endpoints of the next segment to predict (arg max over p_α and p_β), and we measured the accuracy (named STRICT-ACC) with respect to the ground truth endpoints. This is a very "strict" measure, that does not consider the uncertainty in the prediction (differently from the perplexity) and that gives the same score to remarkable errors and to less significant mistakes in which the prediction is close to the ground truth. (*iii.*) We considered a "relaxed" instance of the accuracy measure (named SOFT-ACC) that considers valid also a prediction in which an endpoint is in the immediate neighborhood of the ground truth cell. In other words, if the prediction corresponds with one of the 8 cells around the target cell (or less than 8, if the target cell is on the edge of the canvas) it is considered right, as shown in the example of Fig. 4. Given a sketch s with T segments, we compute SOFT-ACC as in the following,

$$\text{SOFT-ACC}(s) = \frac{1}{T}\sum_{t=1}^{T}\frac{\sigma(\alpha_t) + \sigma(\beta_t)}{2}, \tag{7}$$

where the summation, of course, does not include the start and end drawing signals and $\sigma(\alpha_t)$ is 1 if $\arg\max p_\alpha(\alpha_t, \beta_t|\cdot) = \alpha_{t+1}$ or if $\arg\max p_\alpha(\alpha_t, \beta_t|\cdot) \in \texttt{neig}(\alpha_{t+1})$, otherwise it is 0. We used the notation $\texttt{neig}(\alpha_{t+1})$ to indicate the surrounding cells to the target α_{t+1}. An analogous definition holds in the case of β. (*iv.*) Finally, we considered the Chamfer Distance (Aksan, Deselaers, Tagliasacchi, & Hilliges, 2020), which takes one point of the generated image and find the distance of a closest point of the target image. Formally, given U the set of points of a generated sketch and V the set of points of the target image, the chamfer distance between U and V is

$$\text{CD}(U, V) = \frac{1}{|U|}\sum_{u\in U}\min_{v\in V}|u - v|. \tag{8}$$

The reader can find a precise studies and in-depth description of this measure in Aksan et al. (2020) and Dantanarayana, Dissanayake, and Ranasinge (2016), together with further details.

[8] We skipped the prediction of first segment, since it would have been the same for all the examples.

Table 2
Accuracies (STRICT-ACC, SOFT-ACC), PERPLEXITY (PX), and Chamfer Distance (CD) on the Sketchy Database. The top portion of the table is about the validation set, while the bottom portion is about the test set.

		STRICT-ACC	SOFT-ACC	PX	CD
VAL	SM net	57.27%	70.24%	17.37	7.69
	SM net + sketch image	60.21%	73.57%	11.05	4.86
	FF net sketch image	8.98%	25.71%	147.05	15.76
TEST	SM net	57.19%	69.98%	17.59	7.68
	SM net + sketch image	**60.24%**	**73.44%**	**11.21**	**4.87**
	FF net sketch image	8.93%	25.40%	150.23	15.72

This measure (the lower the better) relaxes the strict correspondence between generated and target data, due to the min operation, in a more flexible way with respect to the case of SOFT-ACC. When validating the model parameters we used the perplexity score.

In Table 2 we report the results on the Sketchy Database. They include a comparison between the initially proposed Sketch Modeling network (SM Net) of Fig. 1 and the augmented version of Fig. 2 that also exploits an extra input composed of the binary image of the sketch drawn up to the considered time instant (SM Net + Sketch Image). In order to evaluate the impact of such extra input, we also evaluated a Feed-Forward neural network that processes only such input (FF Net Sketch Image), completely discarding the LSTM.[9] The results on the test data are pretty inline with the ones on the validation set, and there is coherence in promoting the SM Net + Sketch Image as the best of the compared models. On one hand, this result was expected, since we are providing a description of the whole sketch drawn up to the current time instant that helps the MLPs to correctly predict the next segment. On the other hand, the role of the LSTM is crucial, considering the results of the FF Net Sketch Image model. This suggests that the LSTM is encoding important information about the order of the drawing sequences out of which the network is able to identify regularities that are exploited to make better predictions. The plain representation of the sketch image is a sparser and less regular representation, that, alone, is not enough to reach high accuracies. Overall, the SM network (with or without the extra input) is able to reach low perplexity values on the test set (the training perplexity is 14.38 for SM Net and 7.0 for the one with the extra input). Similar considerations can be done when focussing on the Chamfer distance, that further remarks the quality of the LSTM-based models.

In order to better investigate our experimental results, in Table 3 we report category-specific results about 20 categories from the Sketchy Database (SM Net of Table 2). While there are some variations in the four measures across the considered classes, we see that such variations are not huge, confirming that the average results are a reasonable summary of the model performance. Considering that classes are composed of pretty different objects, the model seems to capture the most common regularities of the data, without learning to perform well on a few classes and pretty bad in others.

Introducing data from the Quick Draw Dataset, we covered a larger number of categories of sketches, making the learning task more challenging. Although we significantly increased the number of data samples, the results are similar to the already discussed ones, as shown in Table 4. After having inspected some outputs of the system, we found that most of the errors are about the fine-grained details, that are the ones on which there is more variability in the training data and that the LSTM has

[9] Results are about the best FF network we were able to obtain, considering up to 5 layers with up to 700 neurons each.

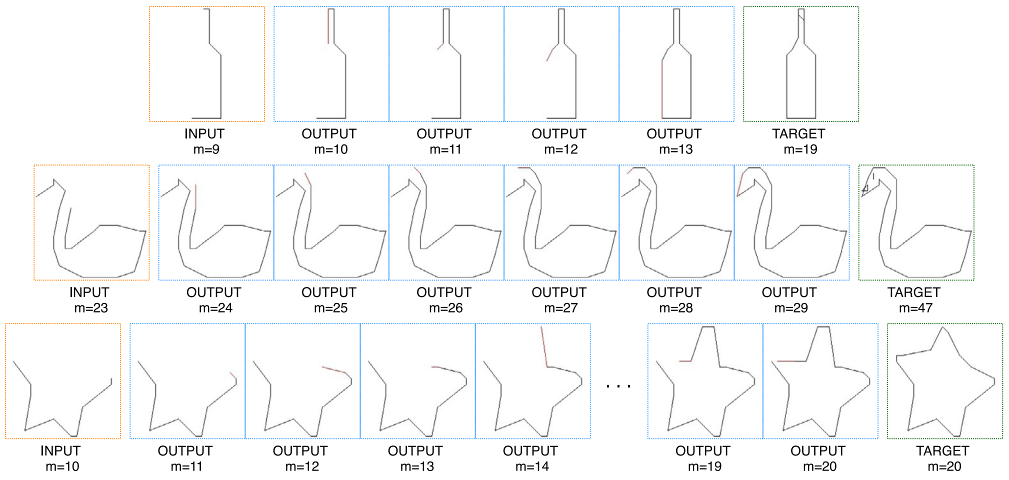


Fig. 5. Examples of sketch completion. The system receives the first half of the sketch segments (INPUT) and tries to complete the sketch (m indicates the number of segments of the sketch in each figure). We report the generated sketch (OUTPUT) at multiple time instants.

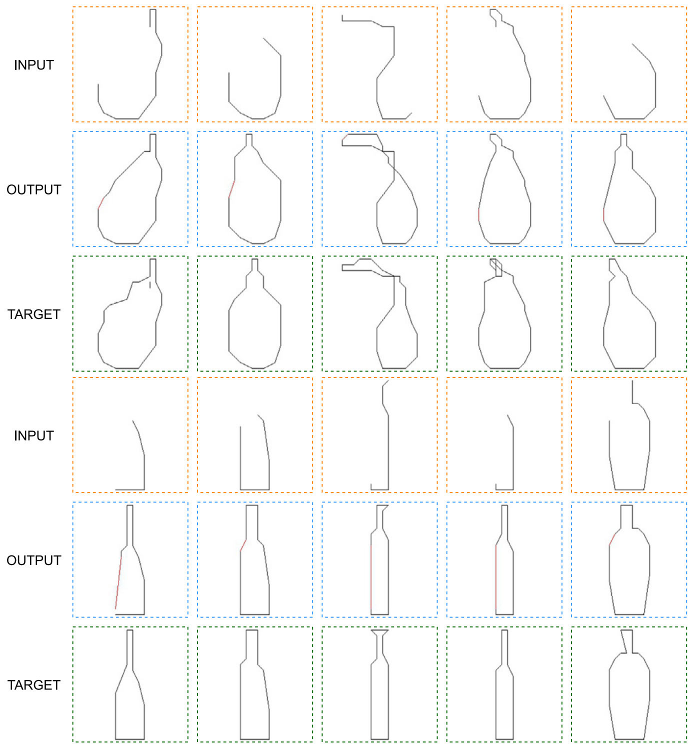


Fig. 6. Generated examples of the class pear and wine bottle (Sketchy Database). INPUT corresponds to partial sketch provided to the networks, and OUTPUT indicates to the generated sketch, to be compared with TARGET.

more difficulties in capturing/generalizing. On the flip-side, we also found that some of these errors are actually not avoidable, since also the variability of the test data is large for some sketch types, and this is expected, due to the intrinsic nature of this task.

4.2. Generation

When providing the SM Net a sequence $\tilde{s}$ composed of half of the segments of the full sketch, we generated the remaining half of the sketch, reporting some qualitative results in Fig. 5. We provide the model half segments of the original sketch and we let it go on alone in the prediction of the next segments, until it finds the stop signal. The initial examples are taken from the test set of the model trained on the Sketchy Database, so they are never-seen-before sketches (but still belonging to categories that are in the training data as well). The system is actually able to complete these sketches, missing some fine-grained details. This is coherent with the error analysis we performed in the sketch modeling task. In Fig. 6 we report further qualitative examples with multiple examples for the class "pear" and "wine bottle", respectively.

Using the same model, we investigated the effects of the Beam Search-based generation and automatic selection procedure of Section 3.2. In particular, we chose a beam size of 5 and we selected the generation for which the following condition holds true: $\texttt{chain}(s) < 1.5 \wedge \texttt{density}(s) < 0.9$. The examples reported in Fig. 7 show that the sketches that fulfill such condition are the ones that better complete the shapes of the input sketches. Some discarded sketches have more segments than the selected ones,

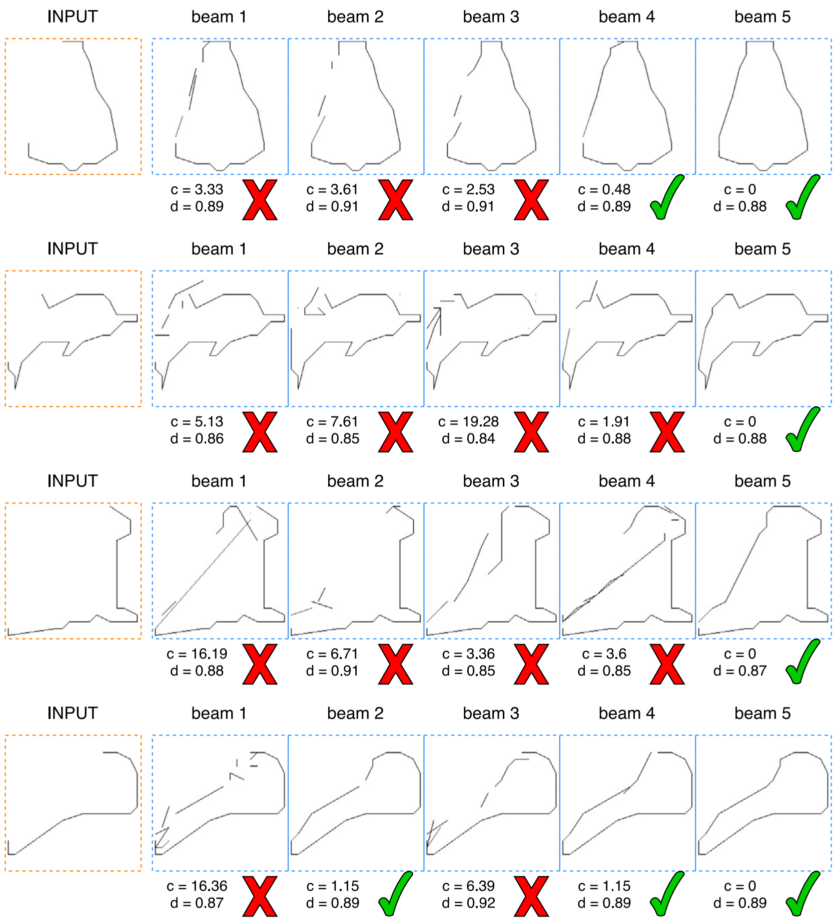


Fig. 7. Beam search results with beam size 5. Qualitative measures of Eqs. (5) and (6) are applied to choose the best outputs. c is the chain score and d the density score. The red cross indicates that the output is rejected and the green thick indicates an accepted output.

Table 3
Accuracies (STRICT-ACC, SOFT-ACC) and PERPLEXITY (PX) and Chamfer Distance (CD) on the Sketchy Database for some categories (SM net).

Category	Strict-Acc	Soft-Acc	Px	CD
Apple	60.15	72.08	10.77	6.71
Banana	57.15	67.64	18.54	7.39
Bell	58.84	70.51	12.71	7.47
Cup	55.7	65.93	18.24	7.28
Dolphin	56.7	66.51	18.92	7.23
Duck	55.66	67.67	16.15	6.62
Fish	55.75	67.51	15.24	6.82
Guitar	57.88	69.47	12.92	7.47
Hammer	56.3	67.41	19.31	7.4
Hot-air balloon	60.37	71.1	9.49	7.25
Hourglass	58.08	67.9	15.17	8.65
Knife	57.74	68.82	15.02	8.06
Mushroom	58.02	67.48	16.6	7.46
Pear	60.43	73.28	9.87	7.64
Shark	56.81	68.37	16.95	5.8
Spoon	59.75	71.84	11.18	8.06
Starfish	56.96	65.32	22.6	7.68
Swan	55.99	67.36	20.58	7.13
Umbrella	55.62	64.65	23.19	7.7
Wine bottle	61.58	70.2	8.97	9.87

Table 4
Accuracies (STRICT-ACC, SOFT-ACC) and PERPLEXITY (PX), and Chamfer Distance (CD) on the sketchy database + quick draw dataset (test split).

	Strict-Acc	Soft-Acc	Px	CD
SM net	58.4%	67.85%	17.32	7.07
SM net + sketch image	**62.02**%	**76.17**%	**7.43**	**4.49**
FF net sketch image	7.85	22.31	97.93	13.86

but they are less connected one to each other, thus violating the condition on the chain score. All the generation examples of this section are based on the aforementioned ranges for the chain and density scores, as it has been done in the case of Fig. 7.

In order to emphasize the importance of the spatial regularizer of Eq. (3), in Fig. 8 we report some sketches generated with a model trained with $\lambda_e = 10^{-4}$ and one without any regularizers ($\lambda_e = 0$). The system has more difficulty to grasp the shape of the drawing when $\lambda_e = 0$. This is mostly because a single small error in the prediction of the spatial location of an endpoint might bring the system in a state from which the following generation does not recover, while the spatial regularizer smooths the cell embeddings among nearby cells, thus introducing robustness in the cell representation.

We also evaluated the quality of the SM Net by training it on sketches that are about facial expressions. In particular, given the trained network, we considered the task of completing the sketch of a human face in which the mouth is not visible, thus asking the system to automatically draw it. The challenging feature of this task consists in generating a mouth that is coherent with the emotion that seems to be represented by the visible portion of the face. We selected $\approx$ 1000 face images labeled with the six universal emotions, leveraging a dataset we used in previous experiences (Graziani, Melacci, & Gori, 2018). We automatically extracted 68 landmarks from each face (see Graziani et al., 2018 for further details), and we collected $T = 65$ segments connecting those landmarks that belong to the same face part (jaw, right eyebrow, left eyebrow, nose, right eye, left eye, outer mouth,

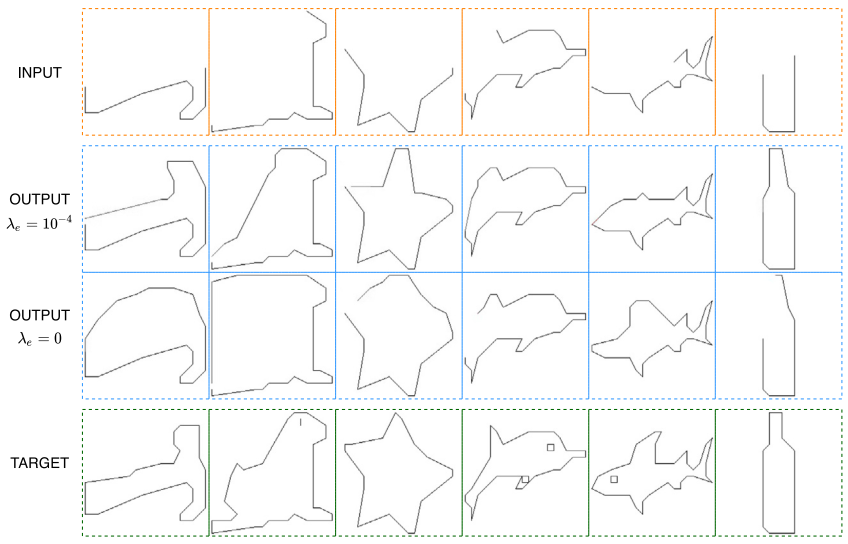


Fig. 8. Examples in generation comparing model with ($\lambda_e = 10^{-4}$) and without ($\lambda_e = 0$) spatial regularity. INPUT corresponds to half initial segments of the TARGET.

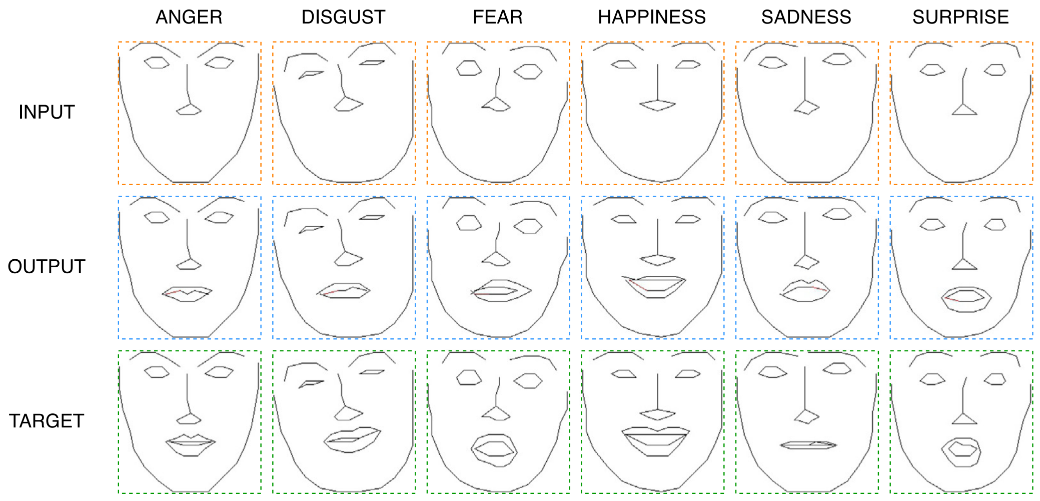


Fig. 9. Sketches of facial expressions. The INPUT of the SM Net is a partially drawn face (we report the associated expression on top of it). The system is asked to complete the sketch, showing OUTPUT results that are coherent with the trend shown in the TARGET row.

inner mouth). The generated sketches[10] of Fig. 9 are created from out-of-sample inputs, and they show that the system can capture the characteristic features of each emotion and complete the sketch in a way that looks appropriate and coherent with the expectations. For instance, in the face with surprised expression the model observed raised eyebrows, wide open eyes and dropped jaw, and it generated a wide open mouth that is coherent with such expression. For completeness, we report quantitative results of the SM Net model in modeling the face-sketching task, that are 69.96% (STRICT-ACC), 85.54% (SOFT-ACC), 4.00 (PX), 2.33 (CD). In particular, we collected 150 out-of-sample sketches to test the selected model. Notice that results are higher than our previous experiences, mostly due to the fact that here the model is exposed to segments that were extracted from well defined facial landmarks that correspond among different examples.

5. Conclusion

In this paper we proposed and assessed the feasibility of a novel approach to sketch generation, named "Sketch Modeling", that is inspired by neural Language Modeling. Sketches are represented as sequences of segments. Instead of performing a regression on the space of stroke coordinates, we treat sketches as sequences of segments whose extremes belong to a vocabulary of quantized coordinates. If needed, segments are automatically extracted and organized from the images of given sketches. Unlike existing works, our model does not need to use special devices or specific applications to collect the sketch data and its simplicity allows it to learn also from multiple classes of sketches. In future work we will focus on dealing with more detailed sketches and longer sequences.

[10] We set $g = 40$, to generate more detailed sketches.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Aksan, E., Deselaers, T., Tagliasacchi, A., & Hilliges, O. (2020). CoSE: Compositional stroke embeddings. arXiv preprint arXiv:2006.09930.

Ballester, P., & Araujo, R. M. (2016). On the performance of GoogLeNet and AlexNet applied to sketches. In *AAAI conference on artificial intelligence*.

Baran, I., Lehtinen, J., & Popović, J. (2010). Sketching clothoid splines using shortest paths. In *Computer graphics forum, Vol. 29* (pp. 655–664). Wiley Online Library.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., et al. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin (Eds.), *Advances in neural information processing systems, Vol. 33* (pp. 1877–1901). Curran Associates, Inc..

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6), 679–698.

Cao, N., Yan, X., Shi, Y., & Chen, C. (2019). Ai-sketcher: A deep generative model for producing high-quality sketches. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 33* (pp. 2564–2571).

Chen, Y., Tu, S., Yi, Y., & Xu, L. (2017). Sketch-pix2seq: a model to generate sketches of multiple categories. arXiv preprint arXiv:1709.04121.

Creswell, A., & Bharath, A. A. (2016). Adversarial training for sketch retrieval. In *European conference on computer vision* (pp. 798–809). Springer.

Dantanarayana, L., Dissanayake, G., & Ranasinge, R. (2016). C-log: A chamfer distance based algorithm for localisation in occupancy grid-maps. *CAAI Transactions on Intelligence Technology*, *1*(3), 272–284.

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., et al. (2019). Unified language model pre-training for natural language understanding and generation. In *Advances in neural information processing systems* (pp. 13063–13075).

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, *10*(2), 112–122.

Eitz, M., Hays, J., & Alexa, M. (2012). How do humans sketch objects? *ACM Transactions on Graphics*, *31*(4), 1–10.

Fišer, J., Asente, P., & Sỳkora, D. (2015). ShipShape: a drawing beautification assistant. In *Proceedings of the workshop on sketch-based interfaces and modeling* (pp. 49–57).

Forbus, K. D., Garnier, B., Tikoff, B., Marko, W., Usher, M., & McLure, M. (2018). Sketch worksheets in STEM classrooms: Two deployments. In *Thirty-second AAAI conference on artificial intelligence*.

Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850.

Graziani, L., Melacci, S., & Gori, M. (2018). The role of coherence in facial expression recognition. In *International conference of the Italian association for artificial intelligence* (pp. 320–333). Springer.

Ha, D., & Eck, D. (2017). A neural representation of sketch drawings. arXiv preprint arXiv:1704.03477.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Liu, F., Deng, X., Lai, Y.-K., Liu, Y.-J., Ma, C., & Wang, H. (2019). Sketchgan: Joint sketch completion and recognition with generative adversarial network. In *IEEE conference on CVPR* (pp. 5830–5839).

Maggini, M., Marra, G., Melacci, S., & Zugarini, A. (2019). Learning in text streams: Discovery and disambiguation of entity and relation instances. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(11), 4475–4486.

Matas, J., Galambos, C., & Kittler, J. (2000). Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, *78*(1), 119–137.

Mikolov, T., Kombrink, S., Burget, L., Černockỳ, J., & Khudanpur, S. (2011). Extensions of recurrent neural network language model. In *Int. conf. on ASSP* (pp. 5528–5531). IEEE.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in NIPS* (pp. 3111–3119).

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. https://openai.com/blog/language-unsupervised/.

Ribeiro, L. S. F., Bui, T., Collomosse, J., & Ponti, M. (2020). Sketchformer: Transformer-based representation for sketched structure. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 14153–14162).

Sangkloy, P., Burnell, N., Ham, C., & Hays, J. (2016). The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics*, *35*(4), 1–12.

Sarvadevabhatla, R. K., Surya, S., Mittal, T., & Babu, R. V. (2018). Pictionary-style word guessing on hand-drawn object sketches: Dataset, analysis and deep network models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(1), 221–231.

Sasaki, K., & Ogata, T. (2018). Adaptive drawing behavior by visuomotor learning using recurrent neural networks. *IEEE Transactions on Cognitive and Developmental Systems*, *11*(1), 119–128.

Song, J., Pang, K., Song, Y.-Z., Xiang, T., & Hospedales, T. M. (2018). Learning to sketch with shortcut cycle consistency. In *IEEE conference on CVPR* (pp. 801–810).

Sun, Z., Wang, C., Zhang, L., & Zhang, L. (2012). Free hand-drawn sketch segmentation. In *European conference on computer vision* (pp. 626–639). Springer.

Suzuki, S., et al. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, *30*(1), 32–46.

Tang, H., Chen, X., Wang, W., Xu, D., Corso, J. J., Sebe, N., et al. (2019). Attribute-guided sketch generation. In *International conference on automatic face & gesture recognition* (pp. 1–7). IEEE.

Todorovic, D. (2008). Gestalt principles. *Scholarpedia*, *3*(12), 5345.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Wiseman, S., & Rush, A. M. (2016). Sequence-to-sequence learning as beam-search optimization. arXiv preprint arXiv:1606.02960.

Xu, P. (2020). Deep learning for free-hand sketch: A survey. arXiv preprint arXiv:2001.02600.

Xu, P., Joshi, C. K., & Bresson, X. (2019). Multi-graph transformer for free-hand sketch recognition. arXiv preprint arXiv:1912.11258.

Zhang, Y., Su, G., Qi, Y., & Yang, J. (2019). Unpaired image-to-sketch translation network for sketch synthesis. In *2019 IEEE visual communications and image processing (VCIP)* (pp. 1–4). IEEE.