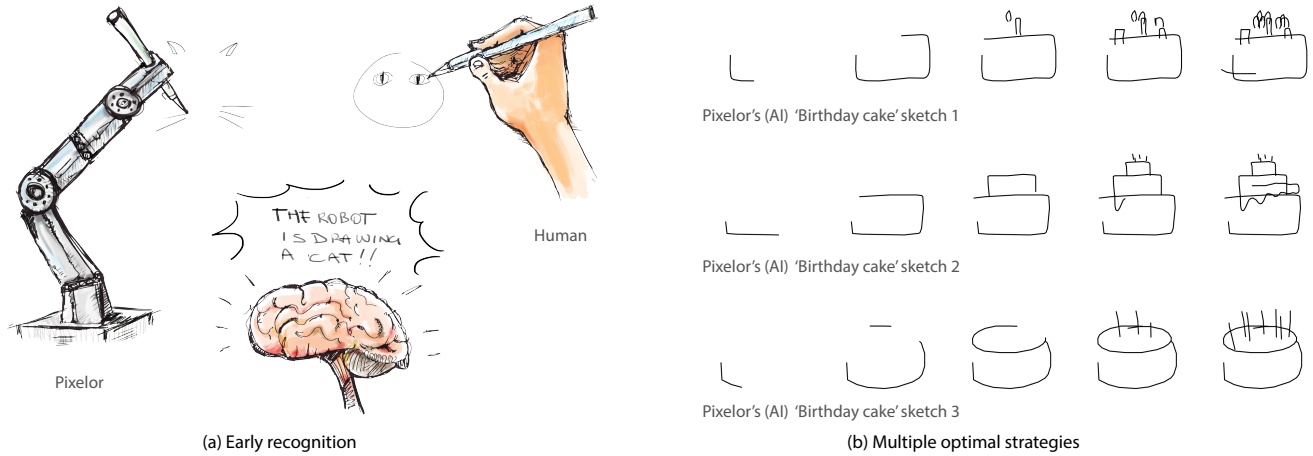
# Pixelor: A Competitive Sketching AI Agent. So you think you can sketch?

AYAN KUMAR BHUNIA*, SketchX, CVSSP, University of Surrey, UK
AYAN DAS*, SketchX, CVSSP, University of Surrey, iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, UK
UMAR MUHAMMAD*, SketchX, CVSSP, University of Surrey, UK
YONGXIN YANG, SketchX, CVSSP, University of Surrey, iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, UK
TIMOTHY M. HOSPEDALES, University of Edinburgh, SketchX, CVSSP, University of Surrey, UK
TAO XIANG, SketchX, CVSSP, University of Surrey, iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, UK
YULIA GRYADITSKAYA, SketchX, CVSSP, University of Surrey, iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, UK
YI-ZHE SONG, SketchX, CVSSP, University of Surrey, iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, UK




(a) Early recognition (b) Multiple optimal strategies

Fig. 1. Our AI sketching agent *Pixelor* learns sketching strategies that lead to early sketch recognition. (a) In a competitive scenario, *Pixelor* and a human player sketch a specified visual concept, while a judge (human or recognizer AI) attempts to recognize the concept. The competitor whose sketch is correctly recognized first is the winner. Winning requires conveying as much as possible with few pixels/little ink. (b) Trained on human sketches, *Pixelor* is able to learn *multiple* winning sketching strategies.

*These authors contributed equally.

Authors' addresses: Ayan Kumar Bhunia, a.bhunia@surrey.ac.uk, SketchX, CVSSP, University of Surrey, UK; Ayan Das, a.das@surrey.ac.uk, SketchX, CVSSP, University of Surrey, iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, UK; Umar Muhammad, u.muhammad@surrey.ac.uk, SketchX, CVSSP, University of Surrey, UK; Yongxin Yang, SketchX, CVSSP, University of Surrey, iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, UK, yongxin.yang@surrey.ac.uk; Timothy M. Hospedales, University of Edinburgh, SketchX, CVSSP, University of Surrey, UK, t.hospedales@ed.ac.uk; Tao Xiang, SketchX, CVSSP, University of Surrey, iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, UK, t.xiang@qmul.ac.uk; Yulia Gryaditskaya, SketchX, CVSSP, University of Surrey, iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, UK, yulia.gryaditskaya@gmail.com; Yi-Zhe Song, SketchX, CVSSP, University of Surrey, iFlyTek-Surrey Joint Research Centre on Artificial Intelligence, UK, y.song@surrey.ac.uk.


0730-0301/2020/12-ART166 $15.00
https://doi.org/10.1145/3414685.3417840

We present the first competitive drawing agent *Pixelor* that exhibits human-level performance at a Pictionary-like sketching game, where the participant whose sketch is recognized first is a winner. Our AI agent can autonomously sketch a given visual concept, and achieve a recognizable rendition as quickly or faster than a human competitor. The key to victory for the agent's goal is to learn the optimal stroke sequencing strategies that generate the most recognizable and distinguishable strokes first. Training *Pixelor* is done in two steps. First, we infer the stroke order that maximizes early recognizability of human training sketches. Second, this order is used to supervise the training of a sequence-to-sequence stroke generator. Our key technical contributions are a tractable search of the exponential space of orderings using neural sorting; and an improved Seq2Seq Wasserstein (S2S-WAE) generator that uses an optimal-transport loss to accommodate the multi-modal nature of the optimal stroke distribution. Our analysis shows that *Pixelor* is better than the human players of the *Quick, Draw!* game, under both AI and human judging of early recognition. To analyze the impact of human competitors' strategies, we conducted a further human study with participants being given unlimited thinking time and training in early recognizability by feedback from an AI judge. The study shows that humans do gradually improve their strategies

with training, but overall *Pixelor* still matches human performance. The code and the dataset are available at http://sketchx.ai/pixelor.

CCS Concepts: • **Computing methodologies** → *Search methodologies*; *Machine learning algorithms*; *Image manipulation*; *Modeling and simulation.*

Additional Key Words and Phrases: Sketch-generation, neural search, early recognition, AI games, recurrent neural network

**ACM Reference Format:**
Ayan Kumar Bhunia, Ayan Das, Umar Muhammad, Yongxin Yang, Timothy M. Hospedales, Tao Xiang, Yulia Gryaditskaya, and Yi-Zhe Song. 2020. Pixelor: A Competitive Sketching AI Agent. So you think you can sketch?. *ACM Trans. Graph.* 39, 6, Article 166 (December 2020), 15 pages. https://doi.org/10.1145/3414685.3417840

## 1 INTRODUCTION

The majority of sketch research to date has focused on making sense of existing human sketch data, including the problems of sketch recognition [Eitz et al. 2012; Schneider and Tuytelaars 2014; Yu et al. 2017], segmentation [Schneider and Tuytelaars 2016; Yang et al. 2020], beautification [Bessmeltsev and Solomon 2019; Simo-Serra et al. 2018] and 3D inference [Su et al. 2018; Xu et al. 2014]. Others have also explored the relationship between sketch and photo via the practical application of sketch-based image retrieval [Sangkloy et al. 2016; Yu et al. 2016]. Instead of just learning a sketch-specific feature representation, these works aim to learn a joint embedding between photo and sketch where retrieval can be conducted. Sketches are the result of a dynamic drawing process – they can be highly abstract and subject to individual drawing skill variability – are thus distinctively different to photos: photos are static and pixel-perfect visual representations.

While static sketch generation and stylization given a photo or an image [Berger et al. 2013; Li et al. 2019; Liu et al. 2020] were studied in vision and graphics communities for a long while, only recently were we able to train machines to draw novel sketches [Ha and Eck 2018]. This task is of fundamental importance to visual understanding of sketches – the AI agent needs to mimic the actual human drawing process stroke-by-stroke to produce a plausible sketch, accommodating different style and abstraction levels (Figure 1). We aim to push the envelope further by introducing an agent that not only can sketch like a human, but also does so with a stroke sequence targeted at early recognizable rendition – just as a good human Pictionary player would do!

In conventional Pictionary, a human player draws an object while being continually evaluated by another human judge. The score in the game depends on how quickly the judge guesses the player's intended object. To train an optimal drawing strategy, we need a training set that is representative of strategies used by strong players. We conduct a study on QuickDraw [Ha and Eck 2018], the largest human sketch dataset to date. We found that albeit with different styles and levels of abstraction, stroke drawing order is a dominant factor in winning the game. For sketches drawn under the time-pressured condition of QuickDraw data collection, human stroke ordering is dramatically better than random: confirming the existence and exploitation by participants of a mental model of stroke informativeness. A key discovery is that the most common ordering is not the optimal one needed to win a Pictionary game, and moreover the few good winning Pictionary players exploit *distinct* stroke orderings. That is, the distribution of optimally ordered sketches with an early recognition property is multi-modal (Figures 1b, 5 and 6).

We train *Pixelor*, our competitive sketching agent, with a two-stage framework. The first stage inputs a given set of training sketches with arbitrary sequential ordering and infers the *stroke-level* ordering that maximizes early recognition for this train set, which is typically better than the human ordering in the raw input. To infer the target stroke ordering we exploit Sketch-a-Net 2.0 [Yu et al. 2017] to score the recognizability of a partial sketch. An obvious strategy is to exhaustively evaluate all possible stroke orders, however, such exhaustive search results in a computationally intractable combinatorial search space. As a tractable optimization strategy over strokes permutations, we leverage *NeuralSort* [Grover et al. 2019], a continuous relaxation of a classical sorting algorithm that allows backward flow of straight-through (ST) gradients [Bengio et al. 2013]. Existing applications of differentiable sorting have used hand-crafted losses [Cuturi et al. 2019; Grover et al. 2019], instead we deploy it to optimize a learned *perceptual* loss [Johnson et al. 2016] in the form of Sketch-a-Net recognition accuracy. Overall, this framework side-steps combinatorial search of stroke ordering by learning a stroke scoring function, such that sorting strokes by score achieves early recognition.

In the second stage, we use the optimal training set computed above to train a Seq2Seq Wasserstein autoencoder (S2S-WAE) sketch generation agent. Classic stroke-based sketch generators [Ha and Eck 2018] are based on the standard Kullback-Leibler (KL) divergence loss widely used in variational autoencoders [Bowman et al. 2016; Kingma and Welling 2014]. Our contribution is to replace this with an optimal transport induced loss [Tolstikhin et al. 2018]. This design choice is necessitated by observations made in the aforementioned human study – the stroke ordering maximizing early recognition is inherently a multi-modal distribution that is better captured using the Wasserstein autoencoder [Tolstikhin et al. 2018]. In addition, we use a Transformer encoder [Vaswani et al. 2017] in order to capture better contextual information compared to bidirectional LSTM in our sketch generator.

To evaluate *Pixelor*, we compare its generated sketches with those from human participants in the *Quick, Draw!* game in terms of early recognizability. We find that for both AI Sketch-a-Net and human judging, *Pixelor* sketches are recognized earlier than the *Quick, Draw!* sketches that form its training data – thus confirming the impact of our stroke order learning. To understand the impact of human sketching conditions on early recognition performance, we conduct a new human study by collecting new sketches via a custom-built on-line interface. In this study humans are instructed to optimize their stroke order for early recognition and given on-the-fly feedback on recognizability in the form of accuracy of the AI classifier. Our analysis shows that with such training, humans do gradually improve their performance in terms of early sketch recognizability. Crucially, (and unlike in *Quick, Draw!*), humans are given unlimited time to think about their sketch. We compare AI and human sketches with pixel-level synchronization, without penalizing clock time or physical drawing speed. The results show

that *Pixelor* trained on old (but reordered) *Quick, Draw!* data still matches human performance under these challenging conditions.

As an application of this work, we define a competitive AI-vs-human sketching game termed Pixelary, where a human competes with a pixel-synchronized *Pixelor* agent to draw an object that can be recognized more quickly by an AI judge.

The main contributions of our work can be summarized as:

- We present *Pixelor*: The first competitive sketching agent that produces novel renditions of a given concept with early recognizability.
- We generate training data for *Pixelor* by solving for the stroke order that maximizes early recognition. We formulate this problem as an *optimization over strokes permutations* and utilize a differentiable solver to learn a stroke scoring model end-to-end. The trained model can be used to compute the optimal order of any given sketch.
- Our sketching agent *Pixelor* is implemented by a Sequence-to-Sequence Wasserstein Auto-Encoder with transformer-encoder to handle the multi-modeal distribution of optimal strategies present in the sketches of good Pictionary players.
- We conduct a comprehensive study using both automated and human evaluation to demonstrate the efficacy of our agent. The results show that *Pixelor* surpasses human performance as exhibited in the *Quick, Draw!* game, and matches the human performance exhibited in our new study with more favorable conditions and training for the human competitors.

## 2 RELATED WORK

*Sketch synthesis.* Most existing sketch synthesis models take raster image as input and follow a photo-to-sketch synthesis paradigm where the task is to produce a human sketch styled version of the input photo. Early works either rely on training a category-specific pictorial structure model [Li et al. 2017] or visual abstraction model [Berger et al. 2013] that replaces strokes from photo edge maps with those of humans. Recent attempts were mostly motivated by neural style transfer methods from the photo domain [Chen et al. 2017b; Gatys et al. 2016; Johnson et al. 2016; Ulyanov et al. 2016]. Zhang et al. [2017] integrated a residual U-net to apply painting styles to sketches with an auxiliary classifier generative adversarial network (AC-GAN) [Odena et al. 2017]. Another class of generative models exists that learns to draw by means of model-free [Ganin et al. 2018] or model-based [Huang et al. 2019] exploration. Such models exploit reinforcement learning algorithms to incorporate immediate feedback from a drawing environment. However, such approaches are known to be sample-inefficient and hence hard to train. Although, recent works [Zheng et al. 2018] tried to alleviate this drawback to some extent by learning differentiable environment models [Ha and Schmidhuber 2018].

Ha and Eck [2018] proposed SketchRNN, a sequence-to-sequence variational autoencoder (Seq2Seq-VAE [Bowman et al. 2016]) to model the sequential sketching process. In this model, the encoder is a bi-directional recurrent neural network (RNN) that takes in a vectorized human sketch and produces a Gaussian distribution over a latent vector that summarizes the sketch, and the decoder is an autoregressive RNN that samples an output sketch conditioned on a random vector drawn from that distribution. The SketchRNN model is trained to mimic the human sketching process including both style and temporal order. Our desired competitive-sketching agent however must furthermore generate sketches efficiently to achieve early recognition with few strokes. To this end it must solve additional challenges, highlighted by a comprehensive analysis on human sketch data (Section 3). First, there are several potential routes to a winning sketching strategy, so it must be able to represent multi-modal data. Second, winning strategies turn out to use more jumps between the end-point and start-point of temporally consecutive strokes than in typical human sketches. The above mentioned SketchRNN model struggles to represent such strategies and generates non-optimal sketching sequences, even if trained with optimally ordered data. To address this issue, we advance sketch generation by employing an optimal transport induced loss for the sequential generative model learning (instead of Kullback-Leibler (KL) divergence loss used in [Bowman et al. 2016; Ha and Eck 2018]).

*Sketch recognition.* Early methods for sketch recognition were developed to deal with professionally drawn sketches as in CAD or artistic drawings [Jabal et al. 2009; Lu et al. 2005; Sousa and Fonseca 2009]. A more challenging task of free-hand sketch recognition was first tackled in [Eitz et al. 2012] along with the release of the first large-scale dataset of amateur sketches. Initial attempts [Li et al. 2015; Schneider and Tuytelaars 2014] to solve this problem consisted mainly of using hand-crafted features together with classifiers such as SVM. Inspired by the success of deep convolutional neural networks (CNN) on various image related problems, the first CNN-based sketch recognition model was proposed in [Yu et al. 2015]. This outperformed previous hand-crafted features, and was followed by various subsequent CNN-based approaches [Jia et al. 2017; Sarvadevabhatla et al. 2018; Yu et al. 2017]. In this paper, we ask a more challenging question – instead of asking AI to recognize pre-drawn sketches, can we train agents to draw sketches from scratch and compete with humans at competitive sketching? However we draw on existing state of the art for recognition to help determine optimal ordering when training our model and to evaluate our results. We use an off-the-shelf recognizer [Yu et al. 2017], as our focus is on how to formulate the early recognition objective using a pre-defined classifier. Exploring alternative on-the-fly recognition methods is an interesting direction for future work.

*Optimal ordering.* Sorting of pre-defined lists is one of the most thoroughly studied algorithms in classic computer science [Cormen et al. 2009]. However, the presence of non-differentiable steps in conventional sorting approaches makes the use of sorting operators in gradient-based machine learning challenging, and use of sorting in learning has been relatively limited to some carefully designed special cases such as ranking [Burges et al. 2005; Burges 2010; Rigutini et al. 2011]. The recent development of fully differentiable sorting [Cuturi et al. 2019; Grover et al. 2019] algorithms enables new capabilities in deep learning including representation learning with ranking and $k$-NN losses, and top-$k$ operators. Existing neural sorting strategies have been demonstrated with hand-crafted losses, where the optimal sorting is the known target for learning. In this paper, we integrate differentiable sorting with a learned perceptual

loss [Johnson et al. 2016], in the form of recognizability as assessed by Sketch-a-Net [Yu et al. 2017]. We train a stroke representation and scoring function such that their sorted scores leads to the earliest possible recognition. This enables us to tractably generate the optimal training set to train *Pixelor*.

*AI Gaming.* With recent advances in artificial intelligence (AI) there have been multiple instances where AIs are able to compare, and in some cases even surpass, human intelligence, whether in simple recognition tasks [Yu et al. 2015] or a competitive game [Morris 1997; Silver et al. 2016; Tesauro 1995]. In particular, the synthesis of deep neural networks and reinforcement learning (RL) has served as a key enabler in recent successes on games such as Atari [Guo et al. 2014; Mnih et al. 2015], 3D virtual environments [Dosovitskiy and Koltun 2017; Jaderberg et al. 2017; Mnih et al. 2016] and the ancient game of Go [Silver et al. 2016, 2017]. Pushing this line of research, we explore the potential of an AI agent to play a competitive Pictionary-like game of competitive sketching. This task is particularly challenging since it involves understanding how to convey semantics efficiently through sketch. Contrary to the aforementioned games, it also uniquely involves: (i) a visual *generation* aspect, rather than solely visual *perception*, and (ii) the added complexity of competing in a game that is subjectively judged by humans. We tackle the challenge by developing a model that combines static (strokes) and dynamic (stroke order) visual information to learn a stroke sequencing strategy targeted on quickly conveying semantics through sketch. Finally, we apply the result to define a fun human-AI competitive sketching game called Pixelary (Section 8).

## 3 HUMAN SKETCH ANALYSIS

To inform the development of our *Pixelor* agent for competitive sketching, we first conduct a study on the largest human sketch dataset to date [Ha and Eck 2018]. This is in order to (i) gain better understanding on the human sketching process, and more importantly (ii) extract insights on how best to win the game. QuickDraw [Ha and Eck 2018] is the largest free-hand sketch dataset with more than 50 million drawings across 345 categories. These sketches were collected through a Pictionary-like game, where users are asked to draw a doodle given a category name, while the game's AI tries to guess the category. An important aspect of this game is the time-limit imposed on the users to draw a recognizable sketch, which makes this dataset suited for our study – we want to compare people who are able to produce a recognizable sketch early on, to those whose sketches are recognized much later on (or never).

*Revealing good and bad human Pictionary players.* In Figure 2, we show average recognition accuracy of 1000 randomly selected sketches from 20 categories from the QuickDraw dataset at different sketch completion rates. *The completion rate* is the ratio of the length of drawn strokes (in pixels) to the total number of pixels in a sketch. The cluster of sketches in the top right corresponds to complete and detail-rich sketches that are well recognized: Those of good sketchers, but weak Pictionary players. The sketches in the bottom right correspond to complete and detailed but non-recognizable

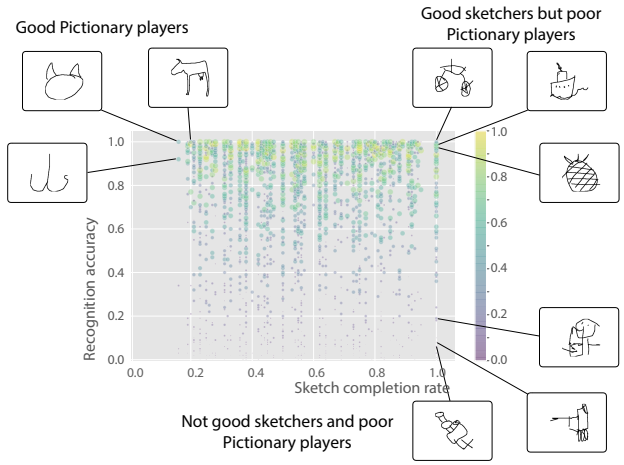



Fig. 2. Sketch recognizability versus sketch completion rate. Plot shows density, encoded both with the color and size of the points. (Top left) Sketches of good Pictionary players, (top right) sketches of good sketchers but poor Pictionary players, and (bottom left and right) sketches of bad sketchers and poor Pictionary players.


(a) Average over several categories (b) Face

Fig. 3. Recognizability scores (Sketch-a-Net 2.0 [Yu et al. 2017]) at various levels of sketch completion for a randomly selected subset of sketches from the QuickDraw dataset [Ha and Eck 2018]. Dashed lines indicates the original human sketching strategy, shaded area indicates the region between the best and worst possible sketching strategies, obtained with a coarse exhaustive search.

sketches: those of bad sketchers. The top left of the plot is of particular interest, as it reflects sketches drawn by people potentially good at a Pictionary game – *they are able to produce recognizable sketches with very few strokes.*

*The gap between good and bad stroke order.* Figure 3 shows that stroke order plays a crucial role in the early recognition of a sketch. The dashed line indicates how the recognizability (using Sketch-a-Net 2.0 [Yu et al. 2017]) of the partial sketches increase as more detail is added – when following the original human sketching order. We perform a coarse exhaustive search (described in detail in Section 4.1.1) over stroke order in each human sketch to find the best- and worst-possible stroke sequences for early recognition, representing by the top and bottom of the colored area. From the plots, it can be seen that: (1) There is a significant gap between the upper- and lower-bound, confirming that stroke order is indeed a dominating

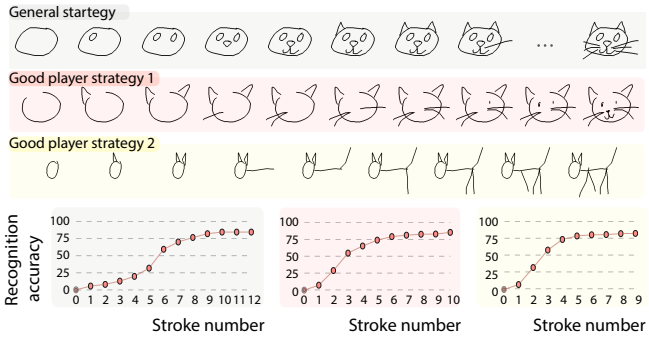



Fig. 4. Example human strokes ordering strategies. Grey: General strategy. Red and Yellow: Two alternative sketching strategies of the good Pictionary players. The graphs show the prediction probability of each sequence at each accumulated stroke. It can be seen that sketches of good players are recognized earlier.

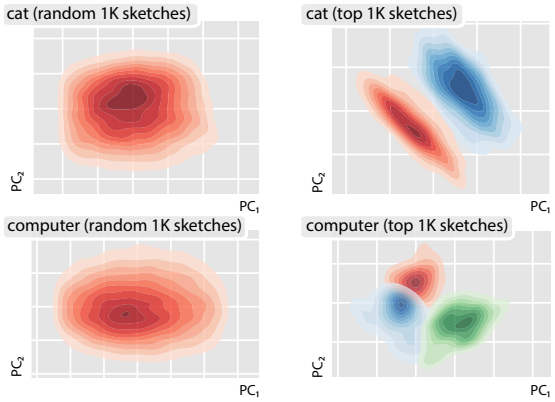



Fig. 5. Kernel density plot of 128D Seq2Seq autoencoder features projected to principal component axes. (Left) Normal human data is uni-modal while top players (right) exhibit multi-modal strategies.

factor for early sketch recognizability. (2) Human sketch order is slightly closer to the upper bound than the lower bound, meaning human participants in QuickDraw generally prioritized drawing salient strokes first (Figure 3 a). (3) In most categories, there is still a big gap between the human order and the optimal order (Figure 3 b). This suggests that – even if using the same set of stroke primitives – an AI agent has scope to compete with humans at Pixelary simply by learning a better stroke-sequencing strategy. This is the motivation behind the training setting of our AI drawing agent: it learns the sequencing strategy by imitating the optimal stroke order rather than the human one, which is sub-optimal for early recognizability purposes.

*The difference between strategies of bad and good human Pictionary players.* We further compare ordering strategies of ordinary human Pictionary players with good players who achieve a recognizable rendition early on. We make two key observations. First, there is by-and-large a common ordering strategy among most humans, while good sketchers often exploit diverse and distinct drawing strategies (Figure 4). In Figure 5 we show kernel density plot of 128D Seq2Seq autoencoder, trained separately on each category, features after PCA, from two categories: cat and computer. It can be seen that there is one dominating cluster of PCA features when we randomly sample 1K sketches from their respective categories. However, we see multi-modal distributions emerging when we choose 1K samples from the sketches of the good Pictionary players. The top 1K sketches are chosen by considering the average recognition probability of the accumulated strokes for each sketch.

Second, the optimal stroke orderings of strong human QuickDraw players exhibit large jumps in-between consecutive strokes. The average euclidean distance (in a $256 \times 256$ canvas) between the end-point and the start-point of consecutive strokes constitutes $94.62(\pm 14.74)$ for the original strokes order and $115.16(\pm 15.47)$ for the optimal stroke order. This is intuitive since humans tend to complete one semantic part of the sketch before moving to the next one (i.e., drawing all whiskers on one side of a cat face sketch before moving to the other side. However, the complete part is often not necessary for recognition, therefore the stroke sequencing strategy that leads to early recognition may move away from this part-by-part paradigm.

## 4 METHODOLOGY

Our goal is to obtain the sketching agent that produces recognizable images with few strokes, as per top Pictionary players. Unfortunately, only a small percentage of QuickDraw sketches satisfy this criteria and are not sufficient for training. Therefore, as mentioned earlier, there are two steps to training a strong Pixelary agent (Figure 6). (i) Our first step is generation of the optimal training set from the (sub-optimal) QuickDraw human sketch dataset. (ii) The optimally ordered QuickDraw dataset is used to train the generation model. Steps (i) and (ii) could be performed jointly as a single step. Nevertheless, since there is no information flow from (ii) to (i), it is simpler to first pre-compute step (i).

### 4.1 Data reordering

In QuickDraw dataset, each category contains 75,000 sketches. Each sketch has variable number of strokes. Exhaustively searching the best stroke order in each sketch means evaluating each stroke ordering permutation. A sketch with 20 strokes would have $20!(2.43e{+}18)$ permutations. Such process is computationally expensive and infeasible. QuickDraw contains approximately $220K$ feasible sketches with $N < 5$ strokes and $500K$ infeasible ones with $N \geq 8$ strokes. We, instead, explore three alternative strategies: (1) coarse level stroke reordering, (2) a greedy strategy of picking the next stroke that maximizes the accuracy and (3) a neural sorting.

*4.1.1 Coarse level stroke reordering.* To obtain coarse stroke reordering, for each sketch we first iteratively group strokes into 5 stroke groups. At each iteration we identify the shortest stroke and group it with the preceding or following neighboring stoke, depending which of them is shorter. We repeat this process till we obtain 5 stroke groups. 5 stroke groups give $5!(= 120)$ possible stroke permutations. For each stroke groups permutation we compute *Early Recognition Efficacy (ERE)* as the area under curve of correct recognition probability (using Sketch-a-Net 2.0 [Yu et al. 2017]) at each

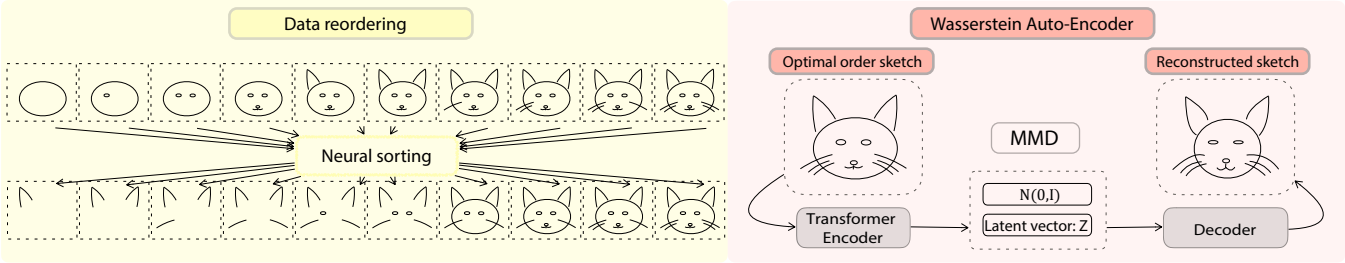



Fig. 6. A schematic illustration of the two step setting to train our stroke rendering agent – *Pixelor*. Yellow: Neural sort prediction for stroke-level order which maximizes sketch recognition for the accumulated stroke representation. Red: Supervised training of Seq2Seq-WAE. We use Maximum Mean Discrepancy (MMD) [Gretton et al. 2012] as a divergence measure between two distributions.

stroke (ERE is formally defined in Section 6.1). We search for the permutation of strokes groups with the highest ERE, and select this as the optimal ordering for each sketch.

*4.1.2 Greedy approach.* The greedy strategy is to sequentially pick a stroke that maximizes the immediate accuracy gain. The computation cost of such approach is $O(N^2)$ classifier evaluations for $N$ strokes.

*4.1.3 Neural sorting.* To obtain an optimal reordering we propose stochastic neural sorting [Grover et al. 2019] and optimize a perceptual loss [Johnson et al. 2016] in the form of recognizability as assessed by Sketch-a-Net [Yu et al. 2017]. Each sketch is first passed into an autoencoder to obtain its feature embedding. The sketch embeddings scores, which are passed to a stochastic sorting operator, are obtained through a multilayer perceptron network those parameters are optimized to obtain early recognition as judged by Sketch-a-Net [Yu et al. 2017].

*Sketch Embedding.* The first step of our training model is obtaining the embedded representation of a sketch. We define a sketch $X = \left[s_j\right]_{j=1}^{N}$ as an ordered set of $N$ strokes $s_j$ and $\mathcal{X}$ as a set of all possible sketches. The set $\mathcal{X}$ also contains partial/incomplete sketches which may not have a semantic meaning. We use an encoder $\mathcal{F}_{\phi_1^*} : \mathcal{X} \rightarrow \mathbb{R}^D$, with parameters $\phi_1^*$, as a feature extractor to extract an embedding vector representing the sketch. $\mathcal{F}_{\phi_1^*}$ produces a $D$-dimensional embedding from a rasterized version of the sketch $X$ on a fixed sized canvas. $\mathcal{F}$ is realized using the Sketch-a-Net 2.0 [Yu et al. 2017] classifier up to a penultimate layer. Thus, the sketch embedding is the $512D$ output of the penultimate layer of the Sketch-a-Net 2.0, where the parameters $\phi_1^*$ are trained through Equation (4).

*Scoring a Stroke.* The embeddings of the full rasterized sketch $X$ and its individual strokes rasterized on a fixed sized canvas together serve as inputs to the neural sorting module. Stochastic neural sorting, just like the classical sorting operator, operates on a set of scalar values: the *scores* of each element in a given set. The scores need to represent the relevance of each element for the downstream task. In our case the elements are individual strokes and the task is sketch recognition. As a scoring function $S_\theta : \mathbb{R}^{2D} \rightarrow \mathbb{R}$ we use (a 3 layer multilayer perceptron neural network MLP) that estimates the relevance score of each stroke of a sketch $X$ with respect to the full $X$. The score vector of a sketch $X$ is computed as $\mathbf{r}(X;\theta) = [r_1(X;\theta), \cdots, r_N(X;\theta)]^T$, where $N$ is a total number of strokes in the sketch $X$, and

$$r_j(X;\theta) = S_\theta\left(\left[\mathcal{F}_{\phi_1^*}\left(\{s_j\}\right); \mathcal{F}_{\phi_1^*}(X)\right]\right) \tag{1}$$

where $\{s_j\}$ is a singleton set of $j^{th}$ stroke of $X$ and [ ; ] represents the concatenation operator. $\mathcal{F}_{\phi_1^*}\left(\{s_j\}\right)$ is the embedding of the rasterized individual stroke, and $\mathcal{F}_{\phi_1^*}(X)$ is the embedding of the full rasterized sketch $X$. Thus, the score $r_j(\cdot)$ for stroke $j$ depends on the concatenated embedding of the $j$th stroke and the full sketch.

*Permuting Strokes.* A sorting of $N$ elements is defined by an $N$-D permutation. Following the notation of [Grover et al. 2019], we denote a valid permutation $\mathbf{z} = [z_1, z_2, \cdots, z_N]^T$ as a list of unique indices from $\{1, 2, \cdots, N\}$ and the corresponding permutation matrix as $P_\mathbf{z} \in \{0,1\}^{N\times N}$ whose entries are $P_\mathbf{z}[i,j] = \mathbf{1}_{j=z_i}$. The value of $z_i$ denotes the index of the $i^{th}$ most salient stroke of the given sketch. With this definition, a sketch permuted by a given permutation $\mathbf{z}$ is expressed as $X_\mathbf{z} = P_\mathbf{z} X^T$.

*Sorting Strokes by Score.* Let us now denote a potentially partial sketch

$$X_{1\rightarrow n} \triangleq \left[s_j\right]_{i=1}^{n}$$

as the set of its first $n$ strokes. By definition, a complete sketch is $X_{1\rightarrow N}$. Towards fulfilling our end goal, we would like to have an optimally permuted sketch $X_{\mathbf{z}^*}$ such that the incomplete sketch $(X_{\mathbf{z}^*})_{1\rightarrow n}$ can be recognized for as low value of $n$ as possible. Hence, we formulate our learning objective for a single sketch $X$ as a minimization of the negative log-likelihood of the correct label for every sketch $(X_\mathbf{z})_{1\rightarrow n}$:

$$J(\theta;X) = \sum_{n=1}^{N} -\log \mathbb{P}\left[y = Y | \mathcal{F}_{\phi_1^*}\left(\left(P_\mathbf{z}\cdot X^T\right)_{1\rightarrow n}\right); \phi_2^*\right] \text{ with} \tag{2}$$

$$P_\mathbf{z} = \textsc{StochasticNeuralSort}\left(\mathbf{r}\left(X;\theta\right)\right), \tag{3}$$

here $Y$ is the true label of the sketch $X$, $\textsc{StochasticNeuralSort}(\cdot)$ is the stochastic sorting operator [Grover et al. 2019] on the ordered set of $N$ scores and $\mathbb{P}[y|\cdot;\phi_2^*]$ is a Sketch-a-Net [Yu et al. 2017] classifier, through which we define a perceptual [Johnson et al. 2016] rather than hand-crafted [Grover et al. 2019] loss. The minimization is performed over the parameters $\theta$ of the scoring function $\mathbf{r}\left(X;\theta\right)$, defined in Equation 1.

The early terms in the summation (i.e., $(X_{\mathbf{z}})_{1\to 1}$, $(X_{\mathbf{z}})_{1\to 2}, \cdots$) will struggle to contribute this optimization, however, uniformly optimizing over all of them encourages an ordering that maximizes the chances of recognition as early on as possible.

*Feature Extractor and Classifier.* The parameterized feature extractor $\mathcal{F}_{\phi_1}(\cdot)$ and classifier $\mathbb{P}[y|\cdot;\phi_2]$ are pre-trained jointly for the sketch classification task as

$$\left\{\phi_1^*, \phi_2^*\right\} = \arg\min_{\{\phi_1,\phi_2\}} \sum_{X\in\widehat{\mathcal{X}}} -\log \mathbb{P}\left[y|\mathcal{F}_{\phi_1}(X);\phi_2\right], \tag{4}$$

where $\widehat{\mathcal{X}}$ is the set of only complete sketches present in the dataset. For efficiency, the representations $\phi_1$ and $\phi_2$ are learned through Eq 4, and fixed for subsequent training of the scoring model with Equation (3). The vector $\phi_1$ is the parameters of the Sketch-a-Net 2.0 [Yu et al. 2017] classifier up to a penultimate layer – the output of this layer is a sketch embedding $\mathcal{F}_{\phi_1}(\cdot)$ and $\phi_2$ is the parameters of the last layer that returns the classification score $\mathbb{P}[y|\cdot;\phi_2]$.

*Optimal ordering.* Optimizing $J(\theta;X)$ (Equation 2) with respect to $\theta$, trains a stroke scoring function in such a way that when all strokes are sorted by salience (Equation 3), the early recognition according to $\mathbb{P}$ is maximized. The optimally sorted version of Quick-Draw for training a generation agent is obtained by permuting each sketch $X$ with the estimated sorting $\mathbf{z}^*$ as $P_{\mathbf{z}^*}X^T$. The re-ordering is independent per sketch, while learning $\theta$ aggregates performance across all sketches.

*Complexity.* Our neural sorting method estimates a relevance score for each stroke in a given sketch in $O(N)$, and performs sorting in $O(N^2)$ time for $N$ strokes. In the training phase, an additional $O(N)$ classifier (loss) evaluations (Equation 2) are required. In practice, the classifier (linear) cost dominates, thus learning to re-order is fast. The complexity values of all considered methods are listed in Table 5.

*Implementation details.* We implemented the scoring function $S_\theta$ with an MLP of a structure $2\times 512 \to 512 \to 256 \to 1$, with each hidden layer followed by a ReLU activation function and the final score is activated with a $\textsc{Sigmoid}(\cdot)$ function. The only trainable parameters of the sorting module are the ones in this MLP ( $\sim 0.6M$). The Sketch-a-Net [Yu et al. 2017] is composed of a few sets of *convolution-max pooling-ReLU* blocks followed by an MLP. We set the temperature parameter of the stochastic neural search [Grover et al. 2019] $\tau(e) = \dfrac{1}{1+\sqrt{e}}$ at $e^{th}$ epoch, gradually reducing it towards zero as the training progresses, forcing the sorting model to discover the non-relaxed permutation matrix. For each sketch in the set of 70,000 training of each class, we infer the best stroke permutation.

## 4.2 Sequence-to-Sequence Wasserstein Auto-Encoder

Our *Pixelor* sketching agent is a sequence-to-sequence Wasserstein autoencoder, which is trained on the reordered data computed in the previous section. We first describe our encoder/decoder architecture and then provide a motivation for a Wasserstein autoencoder.

*Sequence-to-Sequence. Pixelor* is realized as a sequence-to-sequence encoder-decoder architecture. While the classic SketchRNN [Ha and Eck 2018] uses bidirectional LSTM and autoregressive LSTM as the encoder and decoder respectively, we replace the bidirectional LSTM encoder by a Transformer [Vaswani et al. 2017] encoder to capture better contextual information. Following SketchRNN [Ha and Eck 2018], we keep the autoregressive LSTM decoder with Gaussian Mixture Model components.

*Generative Sequence Models.* In vanilla sequence-to-sequence models such as SketchRNN, the encoder produces a mean vector and (diagonal) covariance matrix with which a Gaussian distribution is paramaterized. A random vector is then drawn from this Gaussian which forms the input for the decoder. The difference between *Pixelor* and prior VAE-based work, such as SketchRNN, lies in how the distribution of latent vectors is constrained. The distribution of latent vectors is important because we need to generate a *distribution of new sketch without any reference* such as an input sketch or photo. In SketchRNN's *variational* autoencoder the distribution of latent vectors is controlled via the famous variational lower bound:

$$\inf_{G(X|Z)\in\mathcal{G}} \inf_{Q(Z|X)\in\mathcal{Q}} \mathbb{E}_{P_X}\Big[\mathrm{KL}\left(Q(Z|X), P_Z\right) - \mathbb{E}_{Q(Z|X)}\left[\log p_G(X|Z)\right]\Big], \tag{5}$$

where $P_X$ is the real data distribution, $P_Z$ is the prior distribution over latent vectors (here it is a standard Gaussian), $Q(Z|X)$ is a probabilistic encoder, and $G(X|Z)$ is a deterministic decoder. Both $Q(\cdot)$ and $G(\cdot)$ are realized by neural networks, thus the infimum operator over sets ($\mathcal{Q}$ and $\mathcal{G}$) corresponds to the minimization over network parameters. This model struggles with matching multi-modal distributions because of the form of reverse KL divergence, which results in the requirement of training SketchRNN separately for each object category [Chen et al. 2017a]. As mentioned in Section 3, we observe a multi-model distribution of optimal strategies already within a single category. To alleviate this limitation we propose to use instead a Wasserstein autoencoder.

*Wasserstein autoencoder (WAE).* The wasserstein autoencoder (WAE) is formulated as,

$$\inf_{G(X|Z)\in\mathcal{G}} \inf_{Q(Z|X)\in\mathcal{Q}} \mathbb{E}_{P_X}\left[\mathbb{E}_{Q(Z|X)}\left[c\big(X, G(X|Z)\big)\right]\right] + \lambda\mathcal{D}(Q_Z, P_Z) \tag{6}$$

where $c(a,b)$ is an arbitrary function that measures the difference between $a$ and $b$, $\mathcal{D}$ is an arbitrary divergence between two distributions, and $Q_Z = \mathbb{E}_{P_X}[Q(Z|X)]$. The first term in Equation 6 corresponds to the second term in Equation 5 as they are both reconstruction losses.

The key difference is then how to realize the regularization term: $\mathbb{E}_{P_X}\left[\mathrm{KL}\big(Q(Z|X), P_Z\big)\right]$ (VAE) v.s. $\mathcal{D}(Q_Z, P_Z)$ (WAE). The difference between these two mechanisms matters, as VAE asks every sample's distribution to match the prior (standard Gaussian), which results in a *small intersected area* as the overlap of samples' posterior distributions. This will eventually lead to the lack of diversity/modality.

Instead, $\mathcal{D}(Q_Z, P_Z)$ employs a deterministic decoder, and matches one single posterior distribution (per mini-batch) with the prior (standard Gaussian). As a result, the Wasserstein autoencoder creates a *wider space* for latent vectors, and eventually promotes diversity. The choice of $\mathcal{D}$ is flexible as long as it is a valid distribution

divergence, and we use Maximum Mean Discrepancy (MMD) [Gretton et al. 2012] for the ease of implementation as alternative choices, e.g. Jensen–Shannon divergence, may involve adversarial training, which is less stable. We use an inverse multi-quadratic kernel, as is used in the paper that introduced the Wasserstein autoencoder [Tolstikhin et al. 2018].

*Implementation details.* We use the same data format for sketch coordinates as used by the classic SketchRNN, and the output dimension of the Transformer encoder is 512 with 4 heads and 2 stacked layers. The dimension of the feed forward network within the transformer is 2048, we use the same parameter free positional encoding layer as used by [Vaswani et al. 2017]. The value of $\lambda$ in Equation 6 is 100. Unlike LSTM, since the input and output dimension of transformer encoder are same, we use a simple linear layer, shared across the time steps, to convert 5 elements sketch representation to a 512 dimensional vector at each time step of the sketch sample. Instead of using the final time step's output to predict the distribution parameters, we perform a maxpooling operation over the time steps and use it to predict the distribution parameters.

## 5 TRAINING DATA

We train our model using QuickDraw [Ha and Eck 2018], the largest free-hand sketch dataset. We choose 20 object categories namely bicycle, binoculars, birthday cake, book, butterfly, calculator, cat, chandelier, computer, cow, cruise ship, face, flower, guitar, mosquito, piano, pineapple, sun, truck and windmill; using 70,000 sketches per category for training. In order to maintain the high quality of the generated sketches, we train one separate sorting and synthesis model for each category but share a common classifier as a judge. However, we observed that having one classifier with too many classes makes the sorting model relatively unstable to train. In our experiments, we used two separate 10 class classifiers to train our sorting model. We divide all the data into training and test splits, by keeping a quarter of all sketches as a test set.

## 6 EVALUATION ON QUICKDRAW

In this section, we evaluate neural sorting ordering algorithm against alternative ordering strategies and sketch generation models in terms of early sketch recognition and diversity of generated samples. In this section, recognition accuracy is judged by the Sketch-a-Net [Yu et al. 2017] recognizer AI throughout. Our code is implemented in PyTorch [Paszke et al. 2019].

### 6.1 Metrics

As a quantitative measure of early recognition we define *Early Recognition Efficacy* (ERE) as the area under curve of the correct class probability $\mathbb{P}[y = Y|X_t]$ at different completion rates $t \in [0, 1]$ of a sketch, where $X_{t=0}$ and $X_{t=1}$ denote empty and complete sketches, respectively. We compute an empirical estimate of this quantity as a weighted summation over $T$ discrete intervals

$$ERE \approx \sum_{p=0}^{T} \mathbb{P}\left[y = Y|\mathcal{F}_{\phi_1^*}\left(X_{p\cdot\Delta t}\right); \phi_2^*\right] \Delta t, \tag{7}$$

where $\Delta t = 1/T$.

Table 1. ANOVA results for the impact of each of the ordering strategies on the measure of early recognition efficacy (ERE), defined in Section 6.1.

| Compared ordering strategies | p-value |
|---|---|
| Human order vs Coarse exhaustive search | 5e-12 |
| Coarse exhaustive search vs Greedy | 1.2e-6 |
| **Greedy vs Neural Sort** | **5.5e-15** |

We measure quality and diversity by Frechet Inception Distance (FID) [Heusel et al. 2017]. It is computed by evaluating the distance between the activations of the second last layer of the pre-trained Sketch-a-Net 2.0 classifier for generated sketches and real sketches from the training dataset:

$$d^2 = \|\mu_1 - \mu_2\|^2 + \mathrm{Tr}(C_1 + C_2 - 2\sqrt{C_1C_2}), \tag{8}$$

where $\mu_1, C_1$ and $\mu_2, C_2$ are the mean and covariance of the activations for generated and real sketches respectively.

For the evaluation of all generation models with both metrics, we generate 10,000 sketches for each baseline.

### 6.2 Ordering Strategies

We first validate the ability of our neural sorting ordering algorithm to find a good stroke ordering. The neural sorting model for reordering the original data is compared against: (1) Using the raw data from QuickDraw, in original human provided order, (2) A coarse exhaustive search (Section 4.1.1) (3) A greedy strategy of picking the next stroke that maximizes the accuracy (Section 4.1.2).

*Neural Sort Improves Human Ordering in QuickDraw.* Figure 7 shows the correct class probability (Sketch-a-Net AI judge) versus pixel-level percentage of sketch completion. For neural sorting, we evaluated all metrics on a held-out test-set comprising a quarter of all sketches. All methods start with low recognition accuracy at low sketch completion percentage and asymptote to high accuracy. However better methods increase in accuracy quicker at a given completion rate (closer to top-left corner). The average early recognition efficacy (ERE) (Section 6.1) of sketches with original human order of strokes is $0.612 \pm .10$. Greedy stroke search (dashed) improves on the original human order (dotted), with ERE of $0.665 \pm .08$. A coarse exhaustive search (dash-dotted) with 5 groups achieves an ERE of $0.673 \pm .08$. Thus, both naive heuristics provide similar improvement on the original human data. Our neural sorter (solid) achieves the best performance, reaching ERE of $0.715 \pm .09$, while still being quite efficient to evaluate. We analysed these results for significance using ANOVA, which showed that each reordering method improves statistically significantly in early recognition compared to its predecessor in terms of the average ERE. The computed p-values are shown in Table 1.

*Neural Sort Convergence.* Figure 8 analyses the convergence of our neural sorting method during training in terms of early recognition efficacy of re-ordered human sketches (on a held-out test-set comprising a quarter of all sketches) versus epochs on a few representative categories. We can see that performance improves continually as training proceeds, and surpasses the original ordering within 10 epochs of learning. This confirms that neural sorting

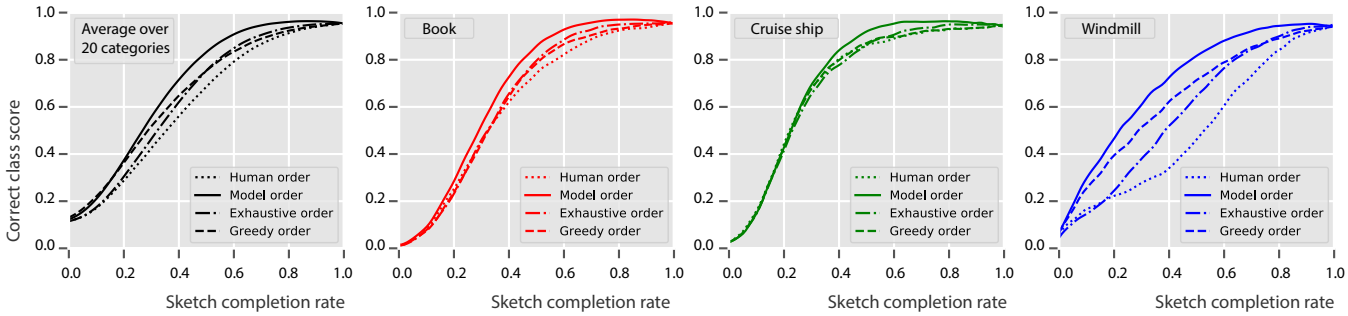



Fig. 7. Correct class probability for different orderings of QuickDraw strokes at different sketch completion rates. We compare correct class probability for the original human strokes ordering (dotted line), versus the following ordering strategies: coarse exhaustive search (dash-doted), greedy search (dashed line) and the proposed neural search (solid line). The curves are shown for the average of all 20 chosen categories and three distinct ones. The curves for all the categories individually are shown in the supplemental.

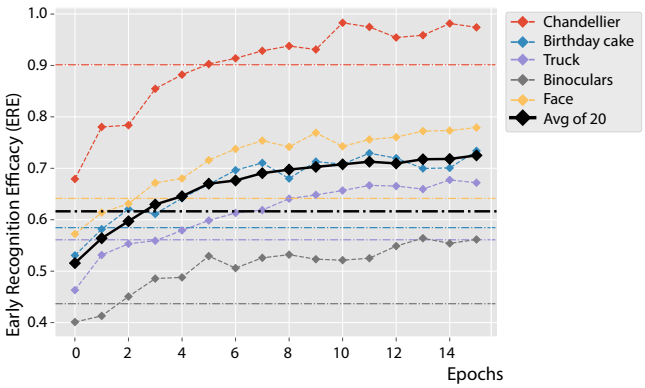



Fig. 8. Evolution of early recognition efficacy (ERE, Section 6.1) of re-ordered QuickDraw data over training epochs of or neural sort. For comparison, we show the ERE of original order as well (dashed lines).

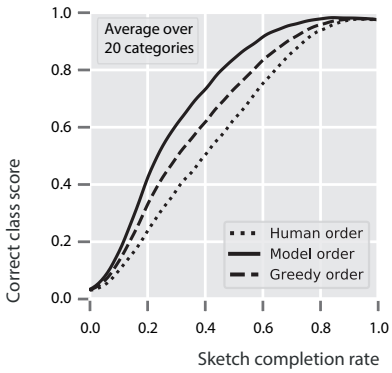



Fig. 9. Correct class probability for the sketches generated with our S2S-WAE, when trained on the data sorted with the neural sort (solid line), with greedy search (dashed line) or original QuickDraw data (dotted line). Correct class probability is an average over all 20 considered categories.

works as expected, and crucially that it is possible to learn stroke and sketch embeddings that provide perceptual informativeness scores (Equation 1) suitable for optimizing early recognition accuracy by sorting.

*Neural Sort Generalization.* As described in Section 4, learning the parameters of the sorting model is reliant on a feature extractor and a classifier already trained to be generalizable on new sketches. As a result, the sorting model can be used on unseen sketches during inference. We took advantage of it by training the sorting model on a subset of the QuickDraw dataset and reordered all sketches only by means of inference. Unlike with an exhaustive search or greedy search, we are able to reduce the computational cost for the neural sort by invoking the classifier while reordering the data. In our experiments, we used 3/4 of the data for training, and reordered the entire dataset as a part of inference. We validate the ability of Neural Sort to generalize by comparing the ERE scores across training set sketches and unseen sketches – these are almost the same, $0.72 \pm .07$ and $0.71 \pm .1$, respectively.

## 6.3 Quality and Diversity of Generation

We first provide a detailed evaluation of an impact of the network architecture choice on the diversity of the generated sketches and early sketch recognition, when trained on optimally sorted data. We then compare the performance of our optimal suggested network architecture versus SketchRNN [Ha and Eck 2018] on the original and sorted data.

*6.3.1 Impact of Model Components.* As described in Section 4.2, the original SketchRNN uses Long Short-Term Memory (LSTM) [Semeniuta et al. 2016] both as an encoder and decoder and Kullback-Leibler (KL) divergence as a distribution criteria. We evaluate the performance when an encoder or/and decoder is a Transformer network [Vaswani et al. 2017]. We as well, substitute variation lower bound (Equation 5) with Wasserstein loss (Equation 6), and with Maximum Mean Discrepancy (MMD) [Gretton et al. 2012] distribution divergence measure. Table 2 shows that the optimal performance in terms of both ERE and FID is obtained when we use Transformer as encoder, in combination with MMD loss, which both

contribute to a better performance. We refer to this S2S-WAE model as *Pixelor*.

Table 2. Ablation study of model components in terms of image generation quality measured by FID score mean and standard deviation. Training with neural-sorted data.

| Model | Encoder | Decoder | Prior | FID Score | ERE |
|---|---|---|---|---|---|
| Sketch-RNN | LSTM | LSTM | KL | 10.17(1.94) | 0.56 |
| Variant-1 | LSTM | LSTM | MMD | 9.81(1.98) | 0.59 |
| Variant-2 | Transformer | LSTM | KL | 9.63(1.68) | 0.63 |
| Variant-3 | Transformer | Transformer | KL | 10.49(1.81) | 0.61 |
| Variant-4 | Transformer | Transformer | MMD | 10.32(2.01) | 0.62 |
| S2S-WAE (Ours) | Tranformer | LSTM | MMD | **9.06(1.86)** | **0.70** |

*6.3.2 Pixelor performance.* We compared the performance of both *Pixelor*, our S2S-WAE and the classic SketchRNN [Ha and Eck 2018] when trained with each dataset: the original QuickDraw stroke data, and our neural sort optimized data.

*Diversity.* We evaluate the diversity of the generated samples in terms of FID scores ([Heusel et al. 2017] and Section (6.1)). Our S2S-WAE has lower FID scores than the conventional Sketch-RNN, both, when trained with the optimally ordered dataset or with the original datatset (Table 3), an thus have higher quality of the generated data. This is due to our ability to better handle the multi-modality of optimally ordered data. The Sketch-RNN is unable to cope with multi-modal distributions, and thus, when trained on the optimally-ordered data, produces lower quality generations. Meanwhile, our S2S-WAE is able to generate high quality images when training on this optimally ordered dataset. FID score models the distribution as a single Gaussian and compares it with the ground-truth distribution. Since in our case the optimally ordered dataset often can not be modeled with a single normal distribution well, the FID scores increase when trained on optimally ordered data. Please see the supplemental for the break down of FID scores per category.

*Early recognition.* We evaluate early recognition through ERE (Eq. 7), by averaging the predicted probability value of the ground-truth category of the sketches at 100 intervals. The results in Table 3, show that: (1) our Seq2Seq-WAE architecture improves on SketchRNN's Seq2Seq-VAE in terms of early recognizability, no matter what data is used for training. (2) Training our model with re-ordered data improves performance substantially. However, (3) only our Seq2Seq-WAE is able to exploit the re-ordered data. SketchRNN's performance drops slightly when using the re-ordered data due to its inability to exploit the multi-modality.

We further evaluate the quality of the fully generated sketches by computing the classification accuracy with pre-trained Sketch-a-Net 2.0. Overall our model generates more recognizable sketches (Table 4), even when only late (rather than early) recognition is considered. This is attributed to the better sequence to sequence architecture of S2S-WAE compared to the classic SketchRNN.

The results in Figure 9 show the early recognition performance of S2S-WAE generated sketches when trained with original human data (dashed line), greedy ordering (dotted line), and our learned neural sort ordering (solid line). We can see that our neural sort ordering outperforms the competitors by a large margin.

Table 3. Early recognition efficacy (ERE) and image generation quality, evaluated through FID scores (lower is better), of generated sketches trained with original ordered data, and order estimated by NeuralSort. Accuracy is averaged across 20 classes. The values in braces indicate standard deviations.

| | Original order | | NeuralSort order | |
|---|---|---|---|---|
| | ERE | FID | ERE | FID |
| S2S-WAE (Ours) | 0.563 | 7.52 (1.72) | 0.657 | 9.07 (1.86) |
| SketchRNN | 0.532 | 8.95 (1.73) | 0.526 | 10.18 (1.94) |

Table 4. Classification accuracy of complete sketches generated by different synthesizers.

| | Orignal order | NeuralSort order |
|---|---|---|
| S2S-WAE (Ours) | 92.6% | 94.9% |
| SketchRNN | 89.0% | 87.3% |

## 6.4 Performance and complexity of reordering

Table 5. Computational complexity and practical single GPU run-time for reordering the $2.4M$ QuickDraw sketches of our chosen 20 classes. Classifier cost $C$, scoring cost $S$, strokes $N$, stroke groups $G$, backprop. epochs $E$.

| | Complexity | Approx runtime |
|---|---|---|
| Exhaustive search | $O(CN!)$ | est. $10^7$ days |
| Coarse Exhaustive search | $O(CG!)$ | $\sim$ 60 days |
| Greedy | $O(CN^2)$ | $\sim$ 8 days |
| **Neural sorting (Ours)** | $O((SN^2 + CN)E)$ | $\sim$ 15 days |

Our optimal neural sort re-ordering strategy has comparable runtime to the greedy strategy that picks a stroke to maximize the accuracy gain. For our model, scoring is much faster than the classifier (so $S \ll C$) and the classifier evaluation (Sketch-a-Net inference $\approx 0.1ms$ on a standard GPU) dominates the cost. The alternatives have no scoring cost, but worse dependence on the classifier cost. Our Neural Sort approach scales well, it is slower than Greedy search, but provides improves early recognition of the ordered and generated data. Furthermore, once trained it can be used to re-order new data as only $O(SN^2)$ cost (Section 6.2).

## 6.5 Discussion

Overall our results show that our Neural Sorting significantly improves the ordering of QuickDraw data in a scalable manner. Furthermore, our resulting *Pixelor* agent can synthesize sketches that can be recognized more early than those in QuickDraw; and earlier and more accurately overall than those generated by SketchRNN.

# 7 HUMAN STUDY

In this section, we perform a set of human studies to evaluate our *Pixelor* agent. In particular, we re-evaluate the previous comparisons on QuickDraw data, but using a team of human judges, rather than Sketch-a-Net AI judge. More significantly, we collect a new dataset of human sketches under favorable conditions designed for early recognition termed *SlowSketch*, and compare *Pixelor*'s performance against this new data under both AI and human judging.

### 7.1 Human Study Setup

*Subset of QuickDraw sketches.* We first evaluate how Pixelor, trained on optimally-ordered QuickDraw data, competes with the human sketchers, who are limited by time constraints (all the participants of QuickDraw datatset were limited by a 20-seconds time frame) when evaluated by human judges. For this evaluation, we randomly selected 12 sketches for each of 20 categories, listed in Section 5 from both QuickDraw and *Pixelor* generated data.

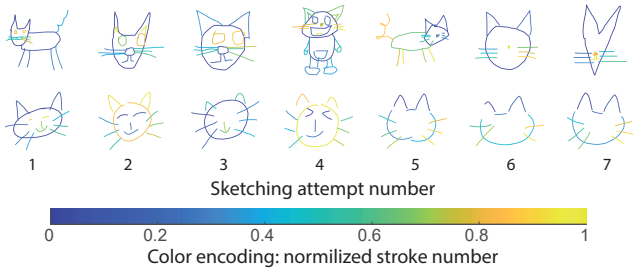



Fig. 10. Visualization of the sketches of the 'cat' category by two participants from the *SlowSketch* datatset.

*Newly collected human sketches - SlowSketch.* We next built a custom interface to collect a new set of test data, where human observers are not limited by any time constraint, and are provided with on-the-fly feedback about the confidence of recognition of their sketches. The participants were asked to make sketches of 20 given categories. To help train participants to optimize for early recognition, the sketching interface consisted of a canvas as well as a plot that shows the score given by the Sketch-a-Net AI judge for the desired category. Participants made at least 7 attempts per category (Figure 10). In this way participants had the chance to refine their sketching strategy to achieve earlier recognition by the AI judge over repeated trials. We collected ≈ 1700 sketches from 12 participants. A screenshot of the interface is provided in the supplemental.

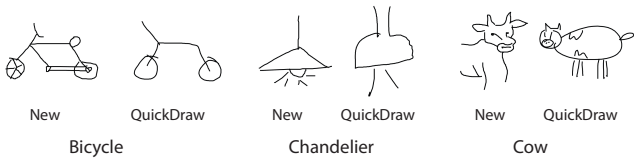



Fig. 11. Visual differences between newly collected *SlowSketch* sketches and samples from the QuickDraw dataset.

*Analysis of SlowSketch.* The new *SlowSketch* data differs from the QuickDraw data in the amount of details and diversity of sketching strategies as shown in Figure 11. We can see that the new sketches are cleaner and more precise than QuickDraw, due to the lack of (clock) time pressure. To study the affect of multiple practice trials, the ERE (AI Judge) scores of the human sketches vs trials are plotted in Figure 12. We can see that performance slowly improves with

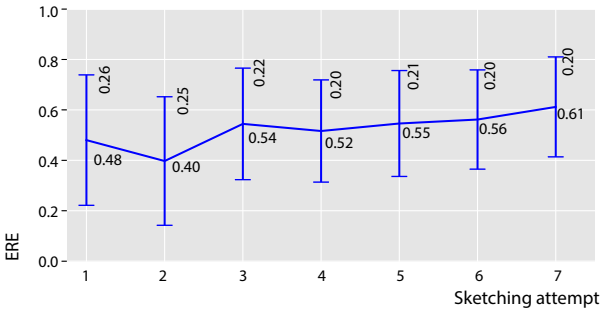



Fig. 12. Early Recognition Efficacy (ERE) of new human sketches in the new *SlowSketch* dataset at the different attempt numbers. ERE improves slightly with practice.

sketching attempts, after some exploration of extreme strategies on the second attempt.

*Human AI Competition: Data Preparation.* To organize the data for a fair comparison of sketching efficiency, we need to normalize for physical drawing speed and scale. We centered and scaled all the sketches so that their area matches a canvas size $256 \times 256$ pixels. To control for drawing speed, we then establish pixel-level synchronization. We compute the length of all sketches and select the one with the largest value. We then pad all the sketches to have the same length by repeating the last sub-stroke of a sketch. This does not visually alter the appearance of the sketch, or the early recognition time. But it does allow all sketches to be compared evenly in terms of % of pixel completion.

For newly collected human sketches, for each participants and each category we select one sketch out of 7 for subsequent *human judge* evaluation, as *the sketch with the shortest total length.* This corresponded to the 4th attempt on average, which is the middle practice trial out of seven. We later show in terms of ERE scores that the selected sketch are indeed representative of the whole set (Figure 14). Note however that the shortest sketch is likely to have the best early recognition properties for human observers among all 7 attempts.

*Metrics.* We employ different metrics for when the judge is human or AI. For AI judge, we measure the early recognizability of sketches through the previously defined ERE (Section 6.1), which is an approximation of a continuous measure. For human judge, we resort to two discrete measures. First, we compute an average (the ratio of the length of drawn strokes in pixels to the total number of pixels in a sketch) for all correct guesses made by human within a given category – smaller number indicates better early recognizability. Second, we evaluate an average amount of the correct guesses – larger number indicates better recognizability.

*Pixelor Training.* For the comparisons in this section, the same *Pixelor* as in Section 6 is used. That is, it is trained on re-ordered QuickDraw data.

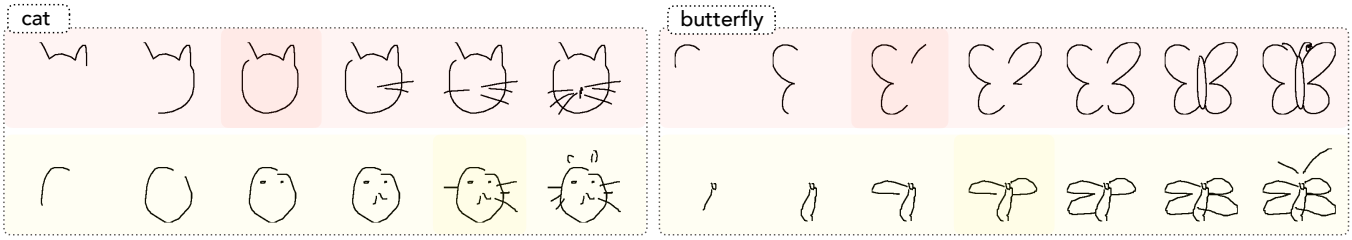



Fig. 13. Top row in each section shows the sketch generated by *Pixelor*, while the second row shows human drawing in *SlowSketch*.

## 7.2 AI judge

We first compare *Pixelor* sketch generation with those drawn in our *SlowSketch* dataset, and those of QuickDraw, when judged for early recognition by the AI classifier Sketch-a-Net 2.0 [Yu et al. 2017]. The early recognition results in Figure 14 show that *Pixelor* clearly provides the best early recognition performance. On average, the newly *SlowSketch* human sketches are correctly recognized at 33% of sketch completion while *Pixelor* ones already at 17%. (The sketch is considered to be correctly recognized when the accuracy of the true class is higher than 0.5.) Only 68% of complete human sketches are correctly recognized, compared to 85% for the randomly selected subset of *Pixelor* sketches, used in this study.

Surprisingly, the AI classifier performs much worse on the newly collected *SlowSketch* sketches than on the QuickDraw human sketches. This is due to the discrepancy between the style of QuickDraw sketches on which the recognition AI was trained, and those in the new dataset (Figure 11). Thus the human *SlowSketch* sketches here are scored unrealistically badly, and this result should not be taken very literally. We do also compare the performance on the selected sketches (previous section) with the average performance and see that the performance is similar. Thus the selected sketches can be considered representative of human performance in *SlowSketch*.

In the next section we repeat the evaluation with human judges, which we assume provides gold-standard recognition such that the dataset bias between QuickDraw and *SlowSketch* can be ignored.

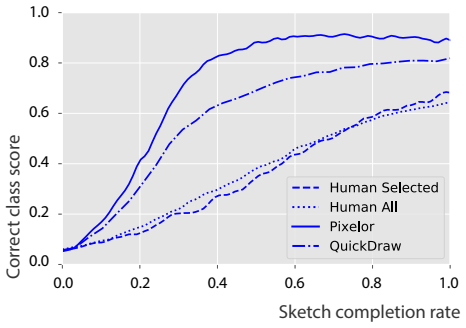



Fig. 14. Comparison of early recognition performance for *Pixelor*, human Quickdraw and human *SlowSketch* sketches (both selected and average). The shown curves are computed by averaging the correct class probabilities of all sketches at each completion rate.

## 7.3 Human judging

In this section we evaluate *Pixelor* vs QuickDraw humans and *Pixelor* vs *SlowSketch* humans under human judging in two independent studies with non-overlapping sets of participant judges.

*Interface.* We created an on-line interface, where the participant judges are shown one sketch at a time and do not know if it is a human or AI sketch. Participants are able to explore a sketch unfolding dynamically by moving a slider (only the movement forward is possible) in 5% completion intervals. Below the canvas with the sketch, we showed 20 categories and asked the judge to guess as soon as they think they recognize the shape. An screenshot of the interface is shown in the supplemental.

*Pixelor vs QuickDraw humans.* In the first study, 8 participant judges provided 1920 guesses in total. In each trial they were randomly shown either a sketch from QuickDraw dataset or a *Pixelor* sketch. We recorded the completion step at which they first correctly recognized the sketch.

In this setup, humans are able to recognize *Pixelor* sketches slightly faster than the sketches from the QuickDraw dataset. The average correct guess for the QuickDraw sketches was done at 29.5% of sketch completion, while for *Pixelor* at 28.2%. We performed an ANOVA analysis which shows that *Pixelor* wins statistically significantly on average with $p$-value of $5e-4$. Moreover, if we analyze each category individually and fix the significance threshold at $5e-3$, *Pixelor* wins statistically significantly on 6 categories and loses just on 2. The total number of correct guesses for the *Pixelor* sketching sequences is also higher than for Quickdraw: 82% versus 79%. Thus, in summary, we can see that *Pixelor* systematically outperforms the humans of QuickDraw.

*Pixelor vs SlowSketch humans.* In the second study another 8 participant judges took part, and were prompted to guess on a mix of *Pixelor* and newly collected *SlowSketch* human sketches. We again collected 1920 guesses. For each category we compute the average sketch completion rate when the correct guess was made. The results in Figure 15 show that *Pixelor* performs comparably to human performance: The average correct guess for Humans is at 31% of the sketch completion vs 33% for AI sketches. On average, significance analysis shows that *Pixelor* loss is statistically significant with $p$-value of $4e-4$. However, when we fix the significance threshold for each category at $5e-3$, we see the performance is almost the same on Human and *Pixelor* generated sketches: *Pixelor* wins on 4 and loses on 5. Please see the supplemental for category-wise $p$-values.

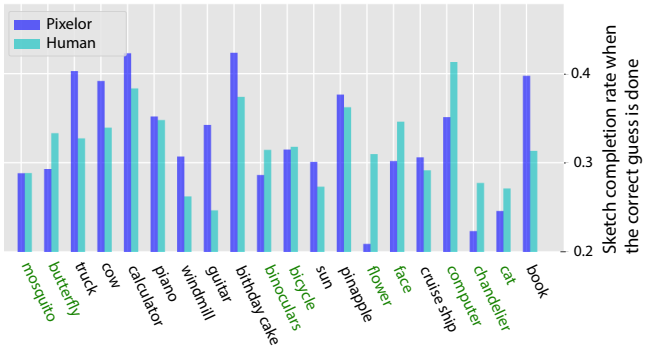



Fig. 15. Comparison of the average sketch completion rates when the correct guess is made. *Pixelor* wins on 9 categories (highlighted in green), demonstrating a competitive behavior. The lower is better.

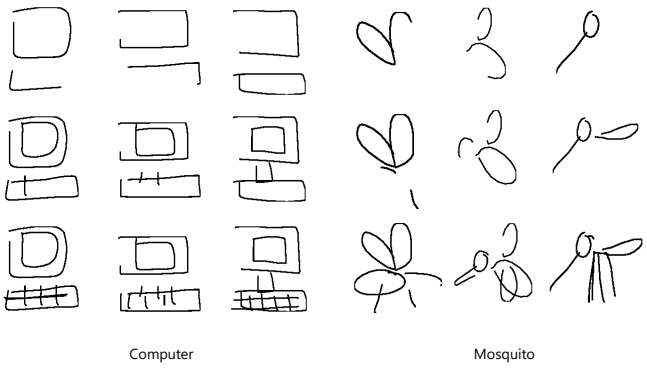



Fig. 16. Examples of the multi-modal generation behavior of Pixelor.

On average human judges made a correct guess on 78.2% of *Pixelor* sketches and 79.0% of human sketches. This similar recognition rate and win/loss ratio indicates that overall *Pixelor* can be considered to perform comparably to the favorably evaluated *SlowSketch* humans.

*Qualitative Analysis.* We provide qualitative comparison of the strategies developed by *Pixelor* versus those of human competitors in *SlowSketch*. Figure 13 shows examples where *Pixelor* beats *SlowSketch* competitors in early recognition under human judging. The dark square indicates the completion step where the human guessed correctly. Overall *Pixelor* starts the sketch with drawing iconic parts of the respective categories. For example, in the case of 'cat' it tends to avoid drawing eyes and nose at an early stage, prioritizing whiskers. For 'butterfly', *Pixelor* focuses on the iconic wing contours ahead of the body or antennae.

More winning sketches of *Pixelor* are provided in Figure 16. It shows that *Pixelor* is indeed able to generate multiple winning strategies per category – computers vary from the standard setups to all-in-one, and mosquitos depict different wing patterns and proboscis. This echoes well with the superior performance of *Pixelor* sketches in terms of FID scores, as previously reported in Section 6.3.2.

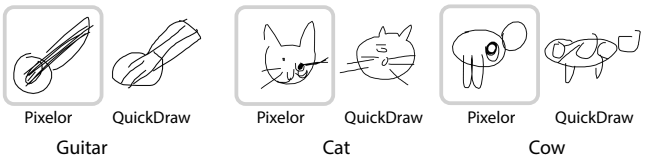



Fig. 17. Poor quality sketches generated by AI (marked with a gray frame) versus example QuickDraw sketches.

Failure cases due to poor-quality sketch generation are illustrated in Figure 17. We notice *Pixelor* would often struggle with synthesizing repetitive patterns (e.g., guitar strings, and cow body patches). This behavior resembles closely with the failure cases of SketchRNN. We attribute this to limitations imposed by the underlying RNN architecture and the difficulty behind modeling the pen jump between sketch strokes.

*Discussion.* The analysis above shows that our *Pixelor* agent outperforms human participants when judged by AI and performs competitively, when judged by a human observer. We attribute this to the overall quality differences between the real human sketches sampled from QuickDraw or the sketches from the new *SlowSketch*, and those automatically synthesized by Pixelor – the former being 100% AI-recognizable sketches, whereas the latter with an average recognition rate of 94.9% (Table 4). Cow is a good example, where the synthesizer often produces odd-looking cows (Figure 17) which directly resulted in cow performing badly in the human study (Figure 15). A better synthesizer would help (e.g., further extending the decoder to Transformer-based), though is outside the scope of this paper. Our neural sorting module is however separate and should work with any synthesizer. We, further, hypothesize that if it was possible to collect a larger amount of data of the type in *SlowSketch* for retraining the synthesize, then the quality of the generated doodles can be improved.

## 8 APPLICATIONS

*Pixelary game.* As an application of our *Pixelor* agent, we define a competitive Pictionary-like sketching game, that is fair, fun for a human user, and focuses the competition on the question of scientific interest – that of sketch generation strategy.

We take inspiration from the popular setting of a Pictionary game where one member from each team is given an object name and the competitors are required to simultaneously draw sketches corresponding to the given word in such a way that a judge can guess the word correctly. To focus the competition on the sketch *generation* rather than recognition, we use an AI judge, and let a human compete with *Pixelor* AI. The judge attempts to guess the category being sketched at each moment of drawing, and the competitor (human or AI) whose sketch's name is guessed correctly first wins the game.

Defining a Pictionary-like game for fair competition between AI and human competitors is non-trivial due to the disembodied nature of the AI. For example, a major problem of AI vs. Human competition in a conventional Pictionary game scenario is that the human is limited by physical movement speed while the AI is not.

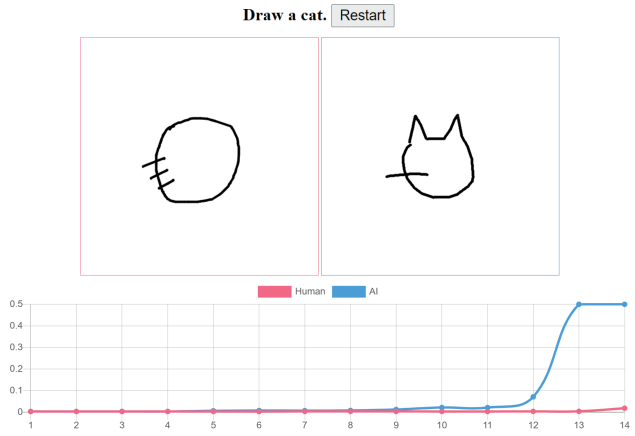



Fig. 18. In our AI vs. Human Pixelary game, a human-player and an AI-player draw simultaneously (with pixel level synchronization) the sketch of a given object on two canvases side by side. Both the sketches are observed an AI judge as they are drawn and evaluated independently. The judge makes one attempt at guessing after each time-step of drawing. The player whose sketch is recognized correctly first wins.

This aspect of unfair competition would lead to a 100% victory rate for the AI, and a boring experience for the human player. To this end, we remove the impact of movement speed in the typical Pictionary game by introducing pixel-level synchronization between the human and AI agent drawings, similarly as we use for our evaluation of the *Pixelor* agent. This means that pixels (~drops of ink) are used by the human and AI agent at exactly the same rate. The player who draws a recognizable sketch first, given the same pixels/ink usage, wins the game. We call this the Pixelary game, as illustrated in Figure 18.

Note that our focus differs from the one presented in Sarvadevabhatla et al. [2018], as we tackle the very different and much more challenging task of creating an agent that can draw sketches to compete with human players rather than developing a judge that guesses the drawn sketches of humans.

*More than a game.* The proposed game and its variations has a potential of assisting in child development and learning, and generally improving human sketching abilities. Results in Figure 12 is a good testament of that, where participants of our data collection study improved the early recognition properties of their sketches. Our agent generates sketches with diverse appearances, as shown in Figure 16, thus encouraging human participants to explore alternative sketching strategies (Figure 10), rather than learning by heart the optimal strokes sequence. The data collected under such scenarios can be used to study human cognitive processes and can encourage the development of advanced sketch recognition agents.

## 9 CONCLUSION

We proposed a novel challenge at the intersection of sketch synthesis and recognition: that of developing a sketching agent optimized for early recognition of its sketches. Our *Pixelor* sketching agent can sketch competitively with humans and often beat humans in early recognition rate in direct competitive sketching. It beats classic SketchRNN, as well as humans of the QuickDraw competition under both human and AI judging, and matches human performance under more challenging conditions of our new *SlowSketch* competition under human judging. Neural generative modeling of sketches is imperfect compared to human sketching quality, meaning that the AI starts at a disadvantage in sketch quality compared to humans. Nevertheless *Pixelor* achieves excellent human-level performance through learning optimized stroke ordering. In particular it achieves excellent performance by developing novel strategies including forsaking the human prior preferences for completeness and symmetry. The key novelty of our work is in formulating the early recognition objective using a pretrained classifier and studying early recognition in line drawing sequences.

## ACKNOWLEDGMENTS

We thank all the participants contributed to the *SlowSketch* dataset and the user study.

## REFERENCES

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

Itamar Berger, Ariel Shamir, Moshe Mahler, Elizabeth Carter, and Jessica Hodgins. 2013. Style and abstraction in portrait sketching. *ACM Transactions on Graphics (TOG)* 32, 4 (2013).

Mikhail Bessmeltsev and Justin Solomon. 2019. Vectorization of line drawings via polyvector fields. *ACM Transactions on Graphics (TOG)* 38, 1 (2019), 1–12.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. In *Conference on Natural Language Learning (CoNLL)*.

Christopher Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine learning (ICML)*.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010).

Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. 2017b. Stylebank: An explicit representation for neural image style transfer. In *Computer Vision and Pattern Recognition (CVPR)*.

Yajing Chen, Shikui Tu, Yuqi Yi, and Lei Xu. 2017a. Sketch-pix2seq: a model to generate sketches of multiple categories. *arXiv preprint arXiv:1709.04121* (2017).

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, Third Edition* (3rd ed.). The MIT Press.

Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. 2019. Differentiable Ranking and Sorting using Optimal Transport. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Alexey Dosovitskiy and Vladlen Koltun. 2017. Learning to act by predicting the future. In *International Conference on Learning Representations (ICLR)*.

Mathias Eitz, James Hays, and Marc Alexa. 2012. How do humans sketch objects? *ACM Transactions on Graphics (TOG)* 31, 4 (2012).

Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Eslami, and Oriol Vinyals. 2018. Synthesizing programs for images using reinforced adversarial learning. *arXiv preprint arXiv:1804.01118* (2018).

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *Journal of Machine Learning Research (JMLR)* 13, 1 (2012).

Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. 2019. Stochastic Optimization of Sorting Networks via Continuous Relaxations. arXiv:stat.ML/1903.08850

Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. 2014. Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning. In *Neural Information Processing Systems (NeurIPS)*.

David Ha and Douglas Eck. 2018. A Neural Representation of Sketch Drawings. In *International Conference on Learning Representations (ICLR)*.

David Ha and Jürgen Schmidhuber. 2018. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems 31*. 2451–2463.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Neural Information Processing Systems (NeurIPS)*.

Zhewei Huang, Wen Heng, and Shuchang Zhou. 2019. Learning to paint with model-based deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*. 8709–8718.

Mohamad Faizal Ab Jabal, Mohd Shafry Mohd Rahim, Nur Zuraifah Syazrah Othman, and Zahabidin Jupri. 2009. A comparative study on extraction and recognition method of CAD data from CAD drawings. In *International Conference on Information Management and Engineering (ICIME)*.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. 2017. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations (ICLR)*.

Qi Jia, Meiyu Yu, Xin Fan, and Haojie Li. 2017. Sequential Dual Deep Learning with Shape and Texture Features for Sketch Recognition. *Computing Research Repository (CoRR)* (2017).

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. *Computing Research Repository (CoRR)* (2014).

Yijun Li, Chen Fang, Aaron Hertzmann, Eli Shechtman, and Ming-Hsuan Yang. 2019. Im2Pencil: Controllable Pencil Illustration From Photographs. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1525–1534.

Yi Li, Timothy M Hospedales, Yi-Zhe Song, and Shaogang Gong. 2015. Free-hand sketch recognition by multi-kernel feature learning. *Computer Vision and Image Understanding (CVIU)* 137, C (2015).

Yi Li, Yi-Zhe Song, Timothy M Hospedales, and Shaogang Gong. 2017. Free-hand sketch synthesis with deformable stroke models. *International Journal of Computer Vision (IJCV)* 122, 1 (2017).

Difan Liu, Mohamed Nabail, Aaron Hertzmann, and Evangelos Kalogerakis. 2020. Neural Contours: Learning to Draw Lines from 3D Shapes. *arXiv* (2020), arXiv–2003.

Tong Lu, Chiew-Lan Tai, Feng Su, and Shijie Cai. 2005. A new recognition model for electronic architectural drawings. *Computer-Aided Design (CAD)* 37, 10 (2005).

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015).

Robert Morris. 1997. Deep Blue Versus Kasparov: The Significance for Artificial Intelligence. *Technical Report* (1997).

Augustus Odena, Christopher Olah, and Jonathon Shlens. 2017. Conditional Image Synthesis with Auxiliary Classifier GANs. In *International Conference on Machine Learning (ICML)*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems(NeurIPS)*.

Leonardo Rigutini, Tiziano Papini, Marco Maggini, and Franco Scarselli. 2011. SortNet: Learning to rank by a neural preference function. *IEEE Transactions on Neural Networks* 22, 9 (2011).

Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. 2016. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)* 35, 4 (2016).

Ravi Kiran Sarvadevabhatla, Shiv Surya, Trisha Mittal, and R. Venkatesh Babu. 2018. Game of Sketches: Deep Recurrent Models of Pictionary-Style Word Guessing. *Computing Research Repository (CoRR)* (2018).

Rosália G Schneider and Tinne Tuytelaars. 2014. Sketch classification and classification-driven analysis using fisher vectors. *ACM Transactions on Graphics (TOG)* 33, 6 (2014).

Rosália G Schneider and Tinne Tuytelaars. 2016. Example-based sketch segmentation and labeling using crfs. *ACM Transactions on Graphics (TOG)* 35, 5 (2016).

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2016. Recurrent dropout without memory loss. *arXiv preprint arXiv:1603.05118* (2016).

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016).

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017).

Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. 2018. Mastering sketching: adversarial augmentation for structured prediction. *ACM Transactions on Graphics (TOG)* 37, 1 (2018), 1–13.

Pedro Sousa and Manuel J Fonseca. 2009. Geometric matching for clip-art drawing retrieval. *Visual Communication and Image Representation (VCIR)* 20, 2 (2009).

Wanchao Su, Dong Du, Xin Yang, Shizhe Zhou, and Hongbo Fu. 2018. Interactive sketch-based normal map generation with deep neural networks. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 1 (2018), 1–17.

Gerald Tesauro. 1995. Td-gammon: A self-teaching backgammon program. In *Applications of Neural Networks*.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Scholkopf. 2018. Wasserstein Auto-Encoders. In *International Conference on Learning Representations (ICLR)*.

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. 2016. Texture Networks: Feed-forward Synthesis of Textures and Stylized Images.. In *International Conference on Machine Learning (ICML)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems (NeurIPS)*.

Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2Form: 3D curve networks from 2D sketches via selective regularization. (2014).

Lumin Yang, Jiajie Zhuang, Hongbo Fu, Kun Zhou, and Youyi Zheng. 2020. SketchGCN: Semantic Sketch Segmentation with Graph Convolutional Networks. *arXiv preprint arXiv:2003.00678* (2020).

Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy Hospedales, and Chen Change Loy. 2016. Sketch Me That Shoe. In *Computer Vision and Pattern Recognition (CVPR)*.

Qian Yu, Yongxin Yang, Feng Liu, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. 2017. Sketch-a-net: A deep neural network that beats humans. *International Journal of Computer Vision (IJCV)* 122, 3 (2017).

Qian Yu, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales. 2015. Sketch-a-net that beats humans. In *British Machine Vision Conference (BMVC)*.

Lvmin Zhang, Yi Ji, and Xin Lin. 2017. Style Transfer for Anime Sketches with Enhanced Residual U-net and Auxiliary Classifier GAN. *Computing Research Repository (CoRR)* (2017).

Ningyuan Zheng, Yifan Jiang, and Dingjiang Huang. 2018. Strokenet: A neural painting environment. In *International Conference on Learning Representations*.