Pictionary-style Word Guessing on Hand-drawn Object Sketches: Dataset, Analysis and Deep Network Models

Ravi Kiran Sarvadevabhatla, *Member, IEEE,* Shiv Surya, Trisha Mittal and R. Venkatesh Babu *Senior Member, IEEE*

Abstract—The ability of intelligent agents to play games in human-like fashion is popularly considered a benchmark of progress in Artificial Intelligence. In our work, we introduce the first computational model aimed at Pictionary, the popular word-guessing social game. We first introduce Sketch-QA, a guessing task. Styled after Pictionary, Sketch-QA uses incrementally accumulated sketch stroke sequences as visual data. Sketch-QA involves asking a fixed question ("What object is being drawn?") and gathering open-ended guess-words from human guessers. We analyze the resulting dataset and present many interesting findings therein. To mimic Pictionary-style guessing, we propose a deep neural model which generates guess-words in response to temporally evolving human-drawn object sketches. Our model even makes human-like mistakes while guessing, thus amplifying the human mimicry factor. We evaluate our model on the large-scale guess-word dataset generated via Sketch-QA task and compare with various baselines. We also conduct a Visual Turing Test to obtain human impressions of the guess-words generated by humans and our model. Experimental results demonstrate the promise of our approach for Pictionary and similarly themed games.

Index Terms—Deep Learning, Pictionary, Games, Sketch, Visual Question Answering

1 Introduction

In the history of AI, computer-based modelling of human player games such as Backgammon, Chess and Go has been an important research area. The accomplishments of well-known game engines (e.g. TD-Gammon [2], DeepBlue [3], AlphaGo [4]) and their ability to mimic human-like game moves has been a well-accepted proxy for gauging progress in AI. Meanwhile, progress in visuo-lingual problems such as visual captioning [5], [6], [7] and visual question answering [8], [9], [10] is increasingly serving a similar purpose for computer vision community. With these developments as backdrop, we explore the popular social game Pictionary^{TM}.

The game of Pictionary brings together predominantly the visual and linguistic modalities. The game uses a shuffled deck of cards with guess-words printed on them. The participants first group themselves into teams and each team takes turns. For a given turn, a team's member selects a card. The member then attempts to draw a sketch corresponding to the word printed on the card in such a way that the team-mates can guess the word correctly. The rules of the game forbid any verbal communication between the drawer and team-mates. Thus, the drawer conveys the intended guess-word primarily via the sketching process.

Consider the scenario depicted in Figure 1. A group of people are playing Pictionary. New to the game, a 'social' robot is watching people play. Passively, its sensors record the strokes being drawn on the sketching board, guess-words uttered by the drawer's team members and finally, whether the last guess is correct or not. Having observed multiple such game rounds, the robot learns computational models which mimic human guesses and enable it to participate in the game.

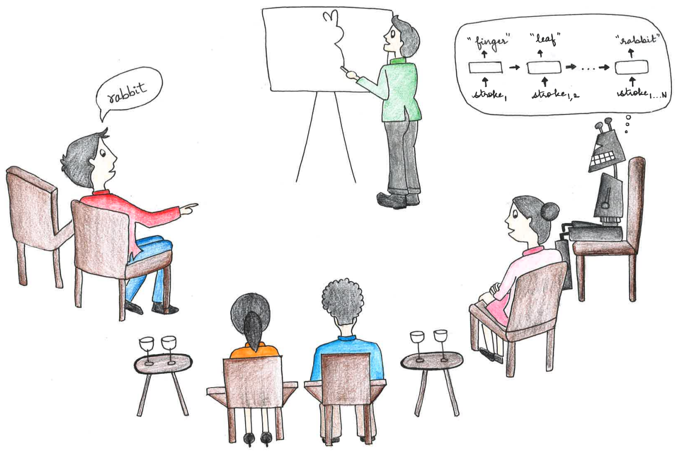

Fig. 1: We propose a deep recurrent model of Pictionary-style word guessing. Such models can enable social robots to participate in real-life game scenarios as shown above. Picture credit:Trisha Mittal.

As a step towards building such computational models, we first collect guess-word data via Sketch Question Answering (Sketch-QA), a novel, Pictionary-style guessing task. We employ a large-scale crowdsourced dataset of hand-drawn object sketches whose temporal stroke information is available [11]. Starting with a blank canvas, we successively add strokes of an object sketch and display this process to human subjects (see Figure 2). Every time a stroke is added,

Manuscript received January, 2018. An earlier brief version of this work was presented in AAAI-18 [1]. The authors are with the Video Analytics Lab, Department of Computational and Data Sciences, Indian Institute of Science, Bangalore, India, 560012. E-mail: ravika@gmail.com, shivparat@gmail.com, mittaltrisha22@gmail.com, venky@iisc.ac.in

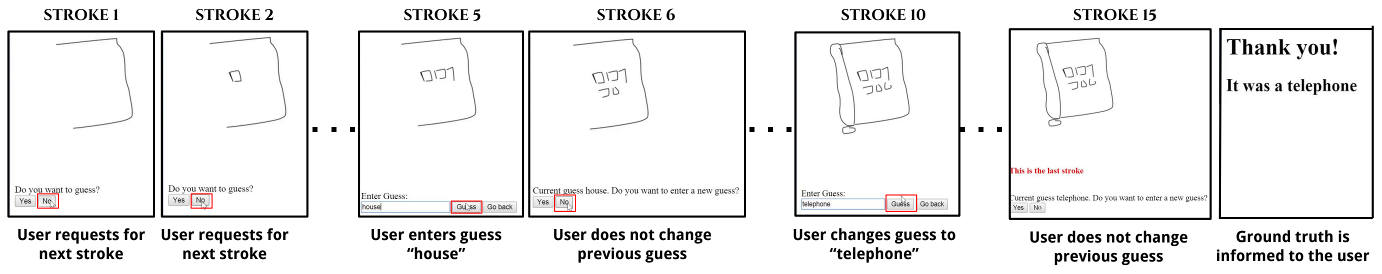
STROKE 1
STROKE 2
STROKE 5
STROKE 6
STROKE 10
STROKE 15
Do you want to guess?
Yes No
Enter Guess:
house
Guess Go back
Current guess house. Do you want to enter a new guess?
telephone
This is the last stroke
Current guess telephone. Do you want to enter a new guess?
Thank you!
It was a telephone
User requests for next stroke
User requests for next stroke
User enters guess "house"
User does not change previous guess
User changes guess to "telephone"
User does not change previous guess
Ground truth is informed to the user

Fig. 2: The time-line of a typical Sketch-QA guessing session: Every time a stroke is added, the subject either inputs a best-guess word of the object being drawn (stroke #5, 10). In case existing strokes do not offer enough clues, he/she requests the next stroke be drawn. After the final stroke (#15), the subject is informed the object's ground-truth category.

the subject provides a best-guess of the object being drawn. In case existing strokes do not offer enough clues for a confident guess, the subject requests the next stroke be drawn. After the final stroke, the subject is informed the object category.

The Sketch-QA setting has a number of important aspects to consider: [a] the visual content consists of sparsely detailed hand-drawn depictions [b] the visual content necessarily accumulates over time [c] the answers (guess-words) are open-ended (i.e. not 1-of-K choices) [d] for a while, until sufficient sketch strokes accumulate, there may not be 'an answer'. In addition, factors such as extremely sparse visual detail, inaccuracies in object depiction arising from varying drawing skills of humans and open-ended nature of answers — pose unique challenges that need to be addressed in order to build viable machine-based guessing models in Pictionary-style scenarios.

A significant motivation for developing computational models of human guessing is their potential for characterizing realistic, possibly sub-optimal human drawings and responses which arise in Pictionary-themed games. In addition, the performance of these models relative to a standardized dataset enables us to quantify the extent to which these models can encode non-trivial human behavior and mimic human responses. Now, the guess data collected via Sketch-QA can contain 'wrong' object category guesses, either due to the quality of drawing or due to human error. We deliberately do not filter out such guesses. This design choice keeps our data realistic and ensures that our computational model has the opportunity to mimic the human players *in-toto* and characterize *both* the 'success' and 'failure' scenarios of Pictionary. In addition, it also makes our work different from instances of AI agents learning human behavior for games, captioning etc. where models are trained to learn best 'moves'.

Concretely, we make the following contributions:

- We introduce a novel task called Sketch-QA to serve as a proxy for the game of Pictionary (Section 2.2).
- Via Sketch-QA, we introduce a new crowdsourced dataset of paired guess-word and sketch-strokes, dubbed WORDGUESS-160, collected from 16,624 guess sequences of 1,108 subjects across 160 sketch object categories (Section 2.1).
- We perform comparative analysis of human guessers and a machine-based sketch classifier via the task of sketch recognition (Section 4). In the process, we also propose an approach for quantifying the suitability of matching criteria used for characterizing the recognition accuracy of human guessers (Section 4.2).
- We introduce a novel computational model for open-world word guessing (Section 6). Using WORDGUESS-160 data, we analyze the performance of the model for Pictionary-style on-line guessing and conduct a Visual Turing Test to gather human assessments of generated guess-words (Section 7).

Please visit https://github.com/val-iisc/sketchguess for code, dataset and supplementary material related to the work presented in this paper. To begin with, we shall look at the procedural details involved in the creation of WORDGUESS-160 dataset.

2 CREATING THE WORDGUESS-160 DATASET

2.1 Sketch object dataset

As a starting point, we use hand-sketched line drawings of single objects from the large-scale TU-Berlin sketch dataset [11]. This dataset contains 20,000 sketches[1] uniformly spread across 250 object categories (i.e. 80 sketches per category). The sketches were obtained in a crowd-sourced manner by providing only the category name (e.g. "sheep") to the sketchers. In this aspect, the dataset collection procedure used for TU-Berlin dataset aligns with the draw-using-guess-word-only paradigm of Pictionary. For each sketch object, temporal order in which the strokes were drawn is also available. A subsequent analysis of the TU-Berlin dataset by Schneider and Tuytelaars [12] led to the creation of a curated subset of sketches which were deemed visually less ambiguous by human subjects. For our experiments, we use this curated dataset containing 160 object categories with an average of 56 sketches per category.

2.2 Data collection methodology

To collect guess-word data for Sketch-QA, we used a web-accessible crowdsourcing portal. Registered participants were initially shown a screen displaying the first stroke of a randomly selected sketch object from a randomly chosen category (see Figure 2). A GUI menu with options 'Yes','No' was provided. If the participants felt more strokes were

1. Following Eitz et al. [11], we adopt the definition of a sketch as "an abstract pictograph drawn by a non-expert". The sketch is generated as a temporal accumulation of a sequence of strokes. In turn, a stroke is a thin (typically 1-2 pixel wide) 2D curve drawn freehand on a static canvas.

Guesses	1	2	3	⩾ 4
# Sequences	12520	2643	568	279

TABLE 1: The distribution of possible number of guesses and count of number of sequences which elicited them.

needed for guessing, they clicked the 'No' button, causing the next stroke to be added. On the other hand, clicking 'Yes' would allow them to type their current best guess of the object category. If they wished to retain their current guess, they would click 'No', causing the next stroke to be added. This act (clicking 'No') also propagates the most recently typed guess-word and associates it with the strokes accumulated so far. The participant was instructed to provide guesses as early as possible and as frequently as required. After the last stroke is added, the ground-truth category was revealed to the participant. Each participant was encouraged to guess a minimum of 125 object sketches. Overall, we obtained guess data from 1,108 participants.

Given the relatively unconstrained nature of guessing, we pre-process the guess-words to eliminate artifacts as described below.

2.3 Pre-processing

Incomplete Guesses: In some instances, subjects provided guess attempts for initial strokes but entered blank guesses subsequently. For these instances, we propagated the last non-blank guess until the end of stroke sequence.

Multi-word Guesses: In some cases, subjects provided multi-word phrase-like guesses (e.g. "pot of gold at the end of the rainbow" for a sketch depicting the object category `rainbow`). Such guesses seem to be triggered by extraneous elements depicted in addition to the target object. For these instances, we used the HunPos tagger [13] to retain only the noun word(s) in the phrase.

Misspelt Guesswords: To address incorrect spellings, we used the Enchant spellcheck library [14] with its default *Words* set augmented with the 160 object category names from our base dataset [11] as the spellcheck dictionary.

Uppercase Guesses: In some cases, the guess-words exhibit non-uniform case formatting (e.g. all uppercase or a mix of both uppercase and lowercase letters). For uniformity, we formatted all words to be in lowercase.

In addition, we manually checked all of the guess-word data to remove unintelligible and inappropriate words. We also removed sequences that did not contain any guesses. Thus, we finally obtain the GUESSWORD-160 dataset comprising of guesswords distributed across 16,624 guess sequences and 160 categories. As mentioned before, it is important to note that the final or the intermediate guesses could be 'wrong', either due to the quality of drawing or due to human error. We deliberately do not filter out such guesses. This design choice keeps our data realistic and ensures that our computational model has the opportunity to characterize both the 'success' and 'failure' scenarios of Pictionary.

A video of a typical Sketch-QA session can be viewed at https://www.youtube.com/watch?v=YU3entFwhV4 . In the next section, we shall present various interesting facets of our WORDGUESS-160 dataset.

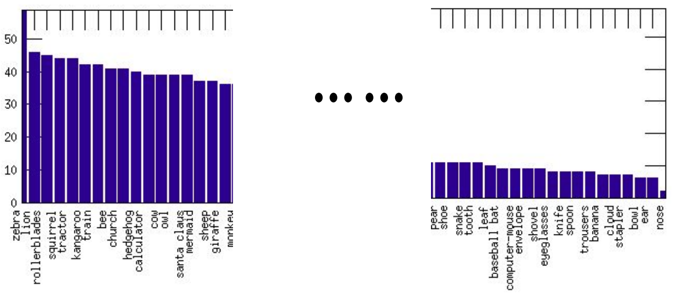

Fig. 3: Here, x-axis denotes the categories. y-axis denotes the number of sketches within the category with multiple guesses. The categories are shown sorted by the number of sketches which elicited multiple guesses.

3 GUESS SEQUENCE ANALYSIS

Given a sketch, how many guesses are typically provided by subjects? To answer this, we examine the histogram of guesses. Table 1 shows the number of sequences eliciting x guesses ($x = \{1, 2, 3, \geqslant 4\}$). Clearly, the number of guesses have a large range. This is to be expected given the large number of object categories we consider and associated diversity in depictions. A large number of subjects provide a single guess. This arises both from the inherent ambiguity of the partially rendered sketches and the confidence subjects place on their guess.

We also examined the sequences which elicited multiple guesses in terms of object categories they belong to. The categories were sorted by the number of multi-guess sequences their sketches elicited. The top-10 and bottom-10 categories according to this criteria can be viewed in Figure 3. Categories which are generally difficult to draw (e.g. `zebra`, `lion`) and categories whose intermediate stroke sequence resemble other objects (e.g. `train`, `church`) tend to elicit a large number of guesses. Sketches from simpler and iconic categories (e.g. `nose`, `ball`) exhibit the opposite trend.

Another interesting statistic is the distribution of first guess location relative to length of the sequence. Figure 4 shows the distribution of first guess index locations as a function of sequence length (normalized to 1). Thus, a value closer to 1 implies that the first guess was made late in the sketch sequence. Clearly, the guess location has a large range across the object categories. The requirement to accurately capture this range poses a considerable challenge for computational models of human guessing.

To obtain a category-level perspective, we computed the median first-guess location and corresponding deviation of first guess location on a per-category basis and sorted the categories by the median values. The resulting plot for the top-ranked and bottom-ranked categories can be viewed in Figure 5. The plots help to determine the level of detail at which categories evolve to a recognizable iconic stroke composition relative to the original, full-stroke reference sketch. Thus, categories such as `axe`, `envelope`, `ladder`, although seemingly simple, are depicted in a manner which induces doubt in the guesser, consequently delaying the induction of first guess. On the other hand, categories such as `cactus`, `strawberry`, `telephone` tend to be drawn such

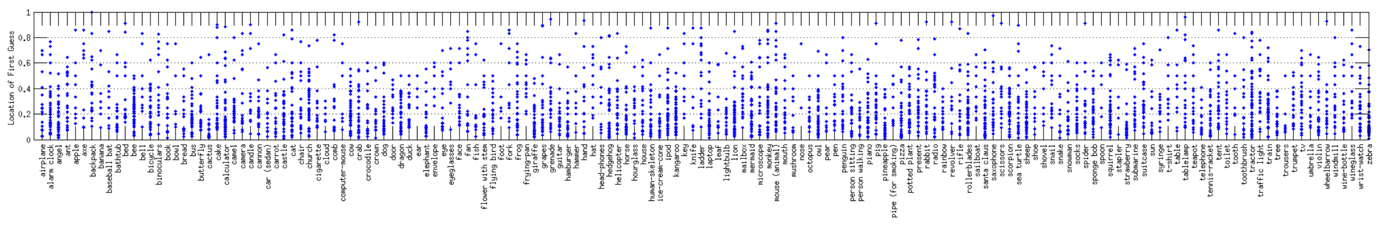

Fig. 4: The distribution of first guess locations normalized ($[0, 1]$) over sequence lengths (y-axis) across categories (x-axis).

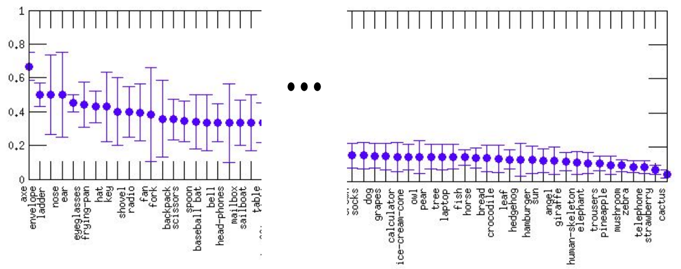

Fig. 5: Categories sorted by the median location of first guess.

that the early, initial strokes capture the iconic nature of either the underlying ground-truth category or an easily recognizable object form different from the forms typically depicted for the ground-truth category.

The above analysis focuses mostly on the overall sequence-level trends in the dataset. In the next section, we focus on the last guess for each sketch stroke sequence. Since the final guess is associated with the full sketch, it can be considered the guesser's prediction of the object underlying the sketch. Such predictions can then be compared with ground-truth labels originally provided with the sketch dataset to determine 'human guesser' accuracy (Section 4.2). Subsequently, we compare 'human guesser' accuracy with that of a machine-based sketch object recognition classifier and discuss trends therein (Section 5).

4 Final guess-word analysis

With GuessWord-160 data at hand, the first question that naturally arises is: What is the "accuracy" of humans on the final, full sketches (i.e. when all the original strokes have been included)? For a machine-based classifier, this question has a straightforward answer: Compute the fraction of sketches whose predicted category label is exactly the same as ground-truth. However, given the open-ended nature of guess-words, an 'exact matching' approach is not feasible. Even assuming the presence of a universal dictionary, such an approach is too brittle and restrictive. Therefore, we first define a series of semantic similarity criteria which progressively relax the correct classification criterion for the final sketches.

4.1 Matching criteria for correct classification

Exact Match (EM): The predicted guess-word is a literal match (letter-for-letter) with the ground-truth category.
Subset (SUB): The predicted guess-word is a subset of ground-truth or vice-versa. This criteria lets us characterize certain multi-word guesses as correct (e.g. guess: *pot of gold at the end of the rainbow*, ground-truth: *rainbow*).
Synonyms (SYN): The predicted guess-word is a synonym of ground-truth. For synonym determination, we use the WordNet [15] synsets of prediction and ground-truth.
Hypernyms (HY): The one-level up parents (hypernyms) of ground-truth and predicted guess-word are the same in the hierarchy induced by WordNet graph.
Hypernyms-Parent and Child (HY-PC): The ground-truth and prediction have a parent-child (hypernym) relationship in the WordNet graph.
Wu-Palmer Similarity (WUP) [16]: This calculates relatedness of two words using a graph-distance based method applied to the corresponding WordNet synsets. If WUP similarity between prediction and ground-truth is at least 0.9, we deem it a correct classification.

4.2 Classification Performance

To compute the average accuracy of human guesses, we progressively relax the 'correct classification' rule by combining the matching criteria (Section 4.1) in a logical-OR fashion. The average accuracy of human guesses can be viewed in Table 2. The accuracy increases depending on the extent to which each successive criterion relaxes the base 'exact match' rule. The large increase in accuracy for 'EM | SUB' (2nd column of the table) shows the pitfall of naively using the exact matching (1-hot label, fixed dictionary) rule.

At this stage, a new question arises: which of these criteria best characterizes human-level accuracy ? Ultimately, ground-truth label is a consensus agreement among humans. To obtain such consensus-driven ground-truth, we performed a human agreement study. We displayed "correctly classified" sketches (w.r.t a fixed criteria combination from Table 2) along with their labels, to human subjects. Note that the labelling chosen changes according to criteria combination. (e.g. A sketch with ground-truth `revolver` could be shown with the label `firearm` since such a prediction would be considered correct under the 'EM | SUB | SYN | HY' combination). Also, the human subjects weren't informed about the usage of criteria combination for labelling. Instead, they were told that the labellings were provided by other humans. Each subject was asked to provide their assessment of the labelling on a scale of -2 ('Strongly Disagree with labelling') to 2 ('Strongly Agree with labelling'). We randomly chose 200 sketches correctly classified under each criteria combination. For each sketch, we collected 5 agreement ratings and computed the weighted average of the agreement score. Finally, we computed the average of these weighted scores. The ratings (Table 3) indicate that 'EM | SUB | SYN' is the criteria combination most agreed upon by human subjects for characterizing human-level accuracy. Having determined the criteria for a correct match, we can also

Criteria Combination	EM	EM | SUB	EM | SUB | SYN	EM | SUB | SYN | HY	EM | SUB | SYN | HY | HY-PC	EM | SUB | SYN | HY | HY-PC | WUP
Accuracy	67.27	75.49	77.97	80.08	82.09	83.33

TABLE 2: Accuracy of human guesses for various matching criteria (Section 4.1). The | indicates that the matching criteria are combined in a logical-OR fashion to determine whether the predicted guess-word matches the ground-truth or not.

Criteria Combination	EM | SUB	EM | SUB | SYN	EM | SUB | SYN | HY	EM | SUB | SYN | HY | HY-PC	EM | SUB | SYN | HY | HY-PC | WUP
Avg. rating	1.01	**1.93**	0.95	1.1	0.21

TABLE 3: Quantifying the suitability of matching criteria combination for characterizing human-level sketch object recognition accuracy. The larger the human rating score, more suitable the criteria. See Section 4.2 for details.

contrast human-classification performance with a machine-based state-of-the-art sketch classifier.

5 Comparing human classification performance with a machine-based classifier

We contrast the human-level performance ('EM | SUB | SYN' criteria) with a state-of-the-art sketch classifier [17]. To ensure fair comparison, we consider only the 1204 sketches which overlap with the test set used to evaluate the machine classifier. Table 5 summarizes the prediction combinations (e.g. Human classification is correct, Machine classification is incorrect) between the classifiers. While the results seem to suggest that machine classifier 'wins' over human classifier, the underlying reason is the open-ended nature of human guesses and the closed-world setting in which the machine classifier has been trained.

To determine whether the difference between human and machine classifiers is statistically significant, we use the Cohen's d test. Essentially, Cohen's d is an effect size used to indicate the standardised difference between two means and ranges between 0 and 1. Suppose, for a given category c, the mean accuracy w.r.t human classification criteria is μ_h^c and the corresponding variance is V_h^c. Similarly, let the corresponding quantities for the machine classifier be μ_m^c and V_m^c. Cohen's d for category c is calculated as :

$$d_c = \frac{\mu_m^c - \mu_h^c}{s} \tag{1}$$

where s is the pooled standard deviation, defined as:

$$s = \sqrt{\frac{V_m^c + V_h^c}{2}} \tag{2}$$

We calculated Cohen's d for all categories as indicated above and computed the average of resulting scores. The average value is 0.57 which indicates significant differences in the classifiers according to the signficance reference tables commonly used to determine Cohen's d significance. In general, though, there are categories where one classifier outperforms the other. The list of top-10 categories where one classifier outperforms the other (in terms of Cohen's d) is given in Table 4.

Some examples of misclassifications (and the ground-truth category labels) can be seen in Figure 6. Although the guesses and ground-truth categories are lexically distant, the guesses are sensible when conditioned on visual stroke data.

Machines outperform humans	Humans outperform machines
scorpion (0.84)	dragon (0.79)
rollerblades (0.82)	owl (0.75)
person walking (0.82)	mouse (0.72)
revolver (0.81)	horse (0.72)
sponge bob (0.81)	flower with stem (0.71)
rainbow (0.80)	wine-bottle (0.65)
person sitting (0.79)	lightbulb (0.65)
sailboat (0.79)	snake (0.63)
suitcase (0.75)	leaf (0.63)

TABLE 4: Category level performance of human and machine classifiers. The numbers alongside category names correspond to Cohen's d scores.

Prediction		Relative % of test data
Human	Machine	
✔	✕	9.05
✕	✔	20.43
✔	✔	67.61
✕	✕	2.91

TABLE 5: Comparing human and machine classifiers for the possible prediction combinations – ✔ indicates correct and ✕ indicates incorrect prediction.

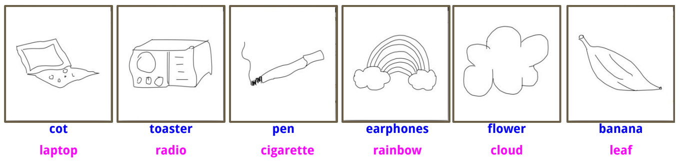

Fig. 6: Some examples of misclassifications: Human guesses are shown in blue. Ground-truth category labels are in pink.

6 Computational Models

We now describe our computational model designed to produce human-like guess-word sequences in an on-line manner. For model evaluation, we split the 16624 sequences in GuessWord-160 randomly into disjoint sets containing 60% , 25% and 15% of the data which are used during training, validation and testing phases respectively.

Data preparation: Suppose a sketch I is composed of N strokes. Let the cumulative stroke sequence of I be $\mathbb{I} = \{S_1, S_2, \ldots S_N\}$, i.e. $S_N = I$ (see Figure 2). Let the sequence of corresponding guess-words be $\mathbb{G}_\mathbb{I} = \{g_1, g_2, \ldots g_N\}$. The sketches are first resized to 224×224 and zero-centered. To ensure sufficient training data, we augment sketch data and associated guess-words. For sketches, each accumulated stroke sequence $S_t \in \mathbb{I}$ is first morphologically dilated ('thickened'). Subsequent augmentations are obtained by

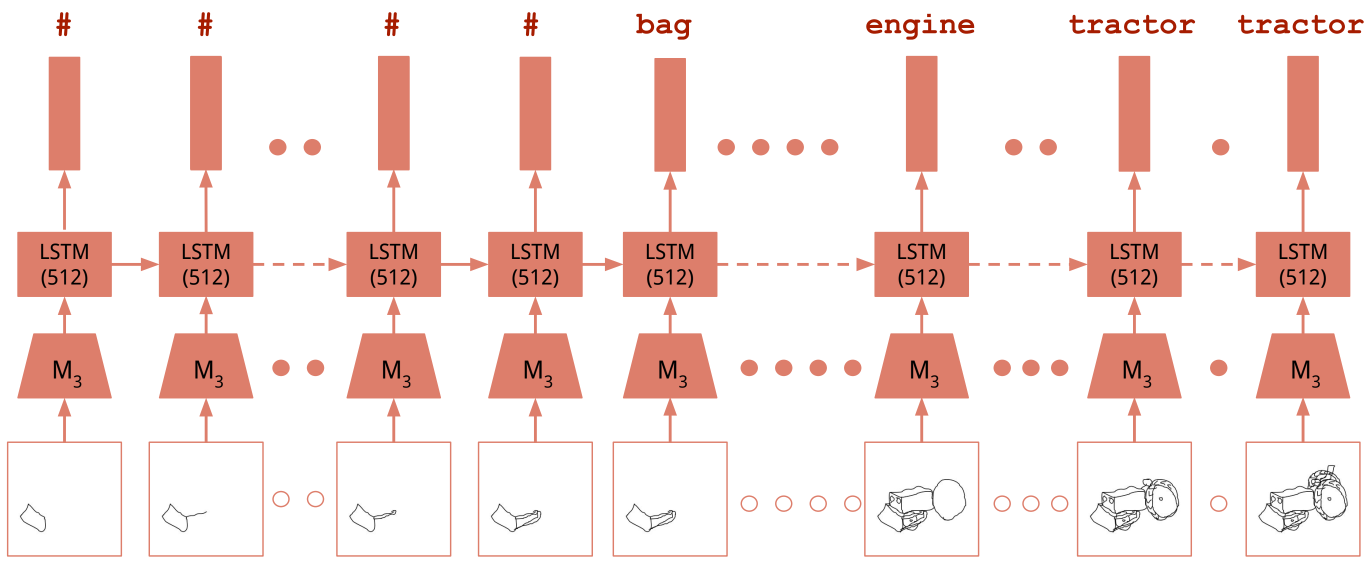

Fig. 7: The architecture for our deep neural model of word guessing. The rectangular bars correspond to guess-word embeddings. M_3 corresponds to the CNN regressor whose penultimate layer's outputs are used as input features to the LSTM model. "#" reflects our choice of modelling 'no guess' as a pre-defined non-word embedding. See Section 6 for details.

applying vertical flip and scaling (paired combinations of $-7\%, -3\%, 3\%, 7\%$ scaling of image side). We also augment guess-words by replacing each guess-word in $\mathbb{G}_{\mathbb{I}}$ with its plural form (e.g. `pant` is replaced by `pants`) and synonyms wherever appropriate.

Data representation: The penultimate fully-connected layer's outputs of CNN fine-tuned on sketches are used to represent sketch stroke sequence images [17]. The guess-words are represented using pre-trained word-embeddings. The details of CNN, related fine-tuning procedure and word-embeddings shall be presented in Section 6.1. Typically, a human-generated guess sequence contains two distinct phases. In the first phase, no guesses are provided by the subject since the accumulated strokes provide insufficient evidence. Therefore, many of the initial guesses (g_1, g_2 etc.) are empty and hence, no corresponding embeddings exist. It is reasonable to choose an embedding for a 'no-guess' instance such that it lies far away from embeddings corresponding to 'word' guesses. By construction, embeddings corresponding to non-words (e.g. special characters, numbers) can be expected to have this property. As an instance of a non-word, we arbitrarily chose the embedding of the character "#". In effect, we map each 'no guess' instance to the pre-defined non-word-embedding of the symbol "#".

Model design strategy: Our model's objective is to map the cumulative stroke sequence $\mathbb{I}$ to a target guess-word sequence $\mathbb{G}_{\mathbb{I}}$. Given our choice of data representation above, the model effectively needs to map the sequence of sketch features to a sequence of word-embeddings. To achieve this sequence-to-sequence mapping, we use a deep recurrent neural network (RNN) as the architectural template of choice (see Figure 7).

For the sequential mapping process to be effective, we need discriminative sketch representations. This ensures that the RNN can focus on modelling crucial sequential aspects such as when to initiate the word-guessing process and when to transition to a new guess-word once the guessing has begun (Section 6.2). To obtain discriminative sketch representations, we first train a CNN regressor to predict a guess-word embedding when an accumulated stroke image is presented (Section 6.1). It is important to note that we ignore the sequential nature of training data in the process. Additionally, we omit the sequence elements corresponding to 'no-guess' during regressor training and evaluation. This frees the regressor from having to additionally model the complex many-to-one mapping between strokes accumulated before the first guess and a 'no-guess'.

To arrive at the final CNN regressor, we begin by fine-tuning a pre-trained photo object CNN. To minimize the impact of the drastic change in domain (photos to sketches) and task (classification to word-embedding regression), we undertake a series of successive fine-tuning steps which we describe next.

6.1 Learning the CNN word-embedding regressor

Step-1: We fine-tune the ImageNet [18]-trained VGG-16 object classification net [19] using Sketchy [20], a large-scale sketch object dataset, for 125-way classification corresponding to the 125 categories present in the dataset. Let us denote the resulting fine-tuned net by M_1.

Step-2: M_1's weights are used to initialize a VGG-16 net which is then fine-tuned for regressing word-embeddings corresponding to the 125 category names of the Sketchy dataset. Specifically, we use the 500-dimensional word-embeddings provided by the `word2vec` model trained on 1-billion Google News words [21]. Our choice is motivated by the open-ended nature of guess-words in Sketch-QA and the consequent need to capture semantic similarity between ground-truth and guess-words rather than perform exact matching. For the loss function w.r.t predicted word embedding p and ground-truth embedding g, we consider [a] Mean Squared Loss : $\|p-g\|^2$ [b] Cosine

LSTM	Avg. sequence-level accuracy		
	1	3	5
–	52.77	63.02	66.40
128	54.13	63.11	66.25
256	55.03	63.79	66.40
512	**55.35**	**64.03**	**66.81**

TABLE 6: Sequence-level accuracies over the validation set are shown. In each sequence, only the portion with guess-words is considered for evaluation. The first row corresponds to M_3 CNN regressor. The first column shows the number of hidden units in the LSTM. The sequence level accuracies with k-nearest criteria applied to per-timestep guess predictions are shown for $k = 1, 3, 5$.

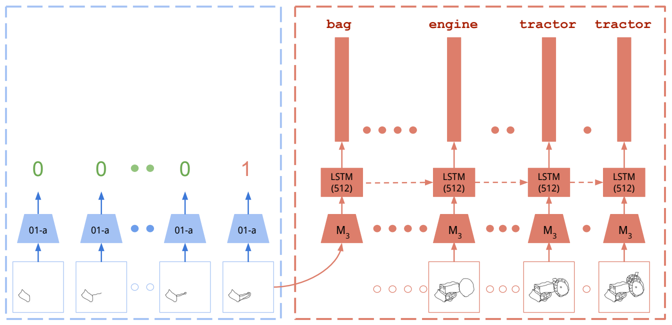

Fig. 8: Architecture for the two-phase baseline. The first phase (blue dotted line) is used to predict location of the transition to the word-guessing phase (output 1). Starting from transition location, the second-phase (red dotted line) sequentially outputs word-embedding predictions until the end of stroke sequence.

Loss [22] : 1- $cos(p, g) = 1 - (p^T g/\|p\| \, \|g\|)$ [c] Hinge-rank Loss [23] : $max[0, margin - \hat{p}^T \hat{g} + \hat{p}^T \hat{h}]$ where $\hat{p}, \hat{g}$ are length-normalized versions of p, g respectively and $\hat{h} (\neq \hat{g})$ corresponds to the normalized version of a randomly chosen category's word-embedding. The value for $margin$ is set to 0.1 [d] Convex combination of Cosine Loss (CLoss) and Hinge-rank Loss (HLoss) : CLoss + λHLoss. **Prediction:** The predicted embedding p is deemed a 'correct' match if the set of its k-nearest word-embedding neighbors contains g. Overall, we found the convex combination loss with $\lambda = 1$ (determined via grid search) to provide the best performance. Let us denote the resulting CNN regressor as M_2.

Step-3: M_2 is now fine-tuned with randomly ordered sketches from training data sequences and corresponding word-embeddings. By repeating the grid search for the convex combination loss, we found $\lambda = 1$ to once again provide the best performance on the validation set. Note that in this case, $\hat{h}$ for Hinge-rank Loss corresponds to a word-embedding randomly selected from the entire word-embedding dictionary. Let us denote the fine-tuned CNN regressor by M_3.

As mentioned earlier, we use the 4096-dimensional output from the penultimate (fc$_7$) layer of M_3 as the representation for each accumulated stroke image of sketch sequences.

Architecture	Avg. sequence-level accuracy		
	1	3	5
M_3 (CNN)	43.61	51.54	54.18
Two-phase	46.33	52.08	54.46
Proposed	**62.04**	**69.35**	**71.11**

TABLE 7: Overall average sequence-level accuracy on test set are shown for guessing models (CNNs only baseline [first row], two-phase baseline [second] and our proposed model [third]). Note that accuracy here reflects the extent to which models agree with human guesses at the sequence level.

6.2 RNN training and evaluation

RNN Training: As with the CNN regressor, we configure the RNN to predict word-embeddings. For preliminary evaluation, we use only the portion of training sequences corresponding to guess-words. For each time-step, we use the same loss (convex combination of Cosine Loss and Hinge-rank Loss) determined to be best for the CNN regressor.
RNN Implementation details: We use LSTM [24] with a 512-dimensional hidden state as the specific RNN variant. For all the experiments, we use Adagrad optimizer [25] (with a starting learning rate of 0.01) and early-stopping as the criterion for terminating optimization. Our implementation and related details can be accessed at https://github.com/val-iisc/sketchguess.
Evaluation: We use the k-nearest neighbor criteria mentioned above and examine performance for $k = 1, 2, 3$. To determine the best configuration, we compute the proportion of 'correct matches' on the subsequence of validation sequences containing guess-words. As a baseline, we also compute the sequence-level scores for the CNN regressor M_3. We average these per-sequence scores across the validation sequences. The results show that the CNN regressor performs reasonably well in spite of the overall complexity involved in regressing guess-word embeddings (see first row of Table 6). However, this performance is noticeably surpassed by LSTM net, demonstrating the need to capture temporal context in modelling guess-word transitions.

7 Overall Results

For the final model, we merge validation and training sets and re-train with the best architectural settings as determined by validation set performance (i.e. M_3 as the feature extraction CNN, LSTM with 512 hidden units as the RNN component and convex combination of Cosine Loss and Hinge-rank Loss as the optimization objective). We report performance on the test sequences.

The full-sequence scenario is considerably challenging since our model has the additional challenge of having to accurately determine when the word-guessing phase should begin. For this reason, we also report performance on a two-phase architecture as an alternate baseline. In this baseline, the first phase predicts the most likely sequential location for 'no guess'-to-first-guess transition. Conditioned on this location, the second phase predicts guess-word representations for rest of the sequence (see Figure 8)[2].

2. To retain focus, we only report performance numbers for the two-phase baseline. For a complete description of baseline architecture and related ablative experiments, please refer to our project page.

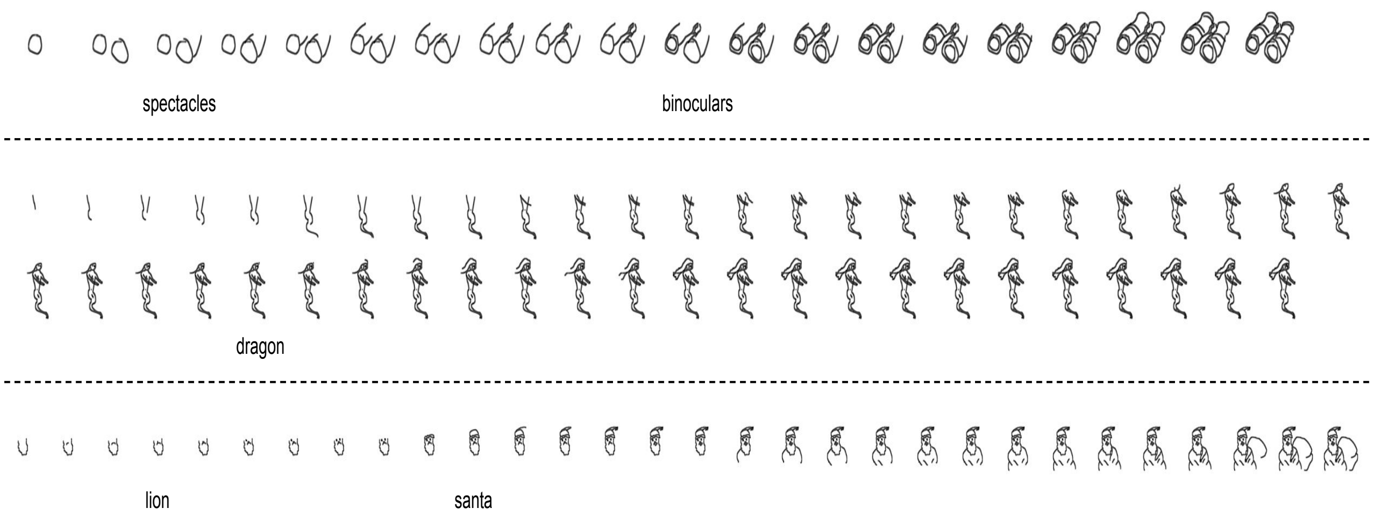

Fig. 9: Examples of guesses generated by our model on test set sequences.

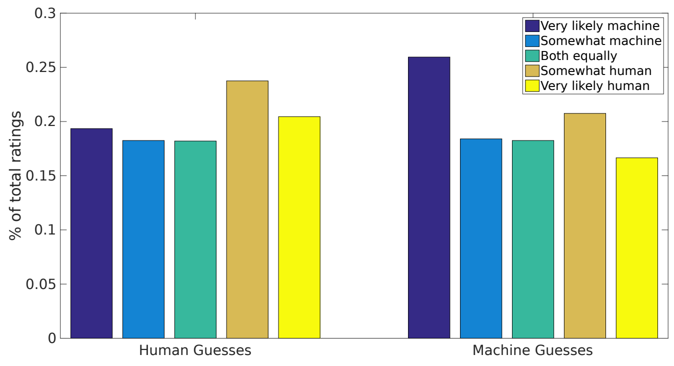

Fig. 10: Distribution of ratings for human and machine-generated guesses.

As can be observed in Table 7, our proposed word-guess model outperforms other baselines, including the two-phase baseline, by a significant margin. The reduction in long-range temporal contextual information, caused by splitting the original sequence into two disjoint sub-sequences, is possibly a reason for lower performance for the two-phase baseline. Additionally, the need to integrate sequential information is once again highlighted by the inferior performance of CNN-only baseline. We also wish to point out that 17% of guesses in the test set are out-of-vocabulary words, i.e. guesses not present in train or validation set. Inspite of this, our model achieves high sequence-level accuracy, thus making the case for open-ended word-guessing models. Examples of guesses generated by our model on test set sketch sequences can be viewed in Figure 9.

Visual Turing Test: As a subjective assessment of our model, we also conduct a Visual Turing Test. We randomly sample $K = 200$ sequences from our test-set. For each of the model predictions, we use the nearest word-embedding as the corresponding guess. We construct two kinds of paired sequences (s_i, h_i) and (s_i, m_i) where s_i corresponds to the i-th sketch stroke sequence ($1 \leqslant i \leqslant K$) and h_i, m_i correspond to human and model generated guess sequences respectively. We randomly display the stroke-and-guess-word paired sequences to 20 human judges with 10 judges for each of the two sequence types. Without revealing the origin of guesses (human or machine), each judge is prompted "Who produced these guesses?".

The judges entered their ratings on a 5-point Likert scale ('Very likely a machine', 'Either is equally likely','Very likely a human'). To minimize selection bias, the scale ordering is reversed for half the subjects [26]. For each sequence $i, 1 \leqslant i \leqslant K$, we first compute the mode ($\mu_i^{\mathcal{H}}$ (human guesses), $\mu_i^{\mathcal{M}}$ (model guesses)) of the 10 ratings by guesser type. To determine the statistical significance of the ratings, we additionally analyze the K rating pairs $((\mu_i^{\mathcal{H}}, \mu_i^{\mathcal{M}}), 1 \leqslant i \leqslant K)$ using the non-parametric Wilcoxon Signed-Rank test [27].

When we study the distribution of ratings (Figure 10), the human subject-based guesses from WORDGUESS-160 seem to be clearly identified as such – the two most frequent rating levels correspond to 'human'. The non-trivial frequency of 'machine' ratings reflects the ambiguity induced not only by sketches and associated guesses, but also by the possibility of machine being an equally viable generator. For the model-generated guesses, many could be identified as such, indicating the need for more sophisticated guessing models. This is also evident from the Wilcoxon Signed-Rank test which indicates a significant effect due to the guesser type ($p = 0.005682, Z = 2.765593$). Interestingly, the second-most preferred rating for model guesses is 'human', indicating a degree of success for the proposed model.

Visualization: The task-relevant encoding of the input-output sequence pairs is considered to be present in LSTM's hidden-state. As a visualization experiment, we examined the hidden-state values for the guess sequences. Given a sequence of sketch stroke images, we pass them to our LSTM model and record the hidden-state values for each element in the sequence. An example of a sketch stroke sequence and corresponding hidden-state values can be viewed in Figure 11. LSTM's hidden-state, as with representations learnt by

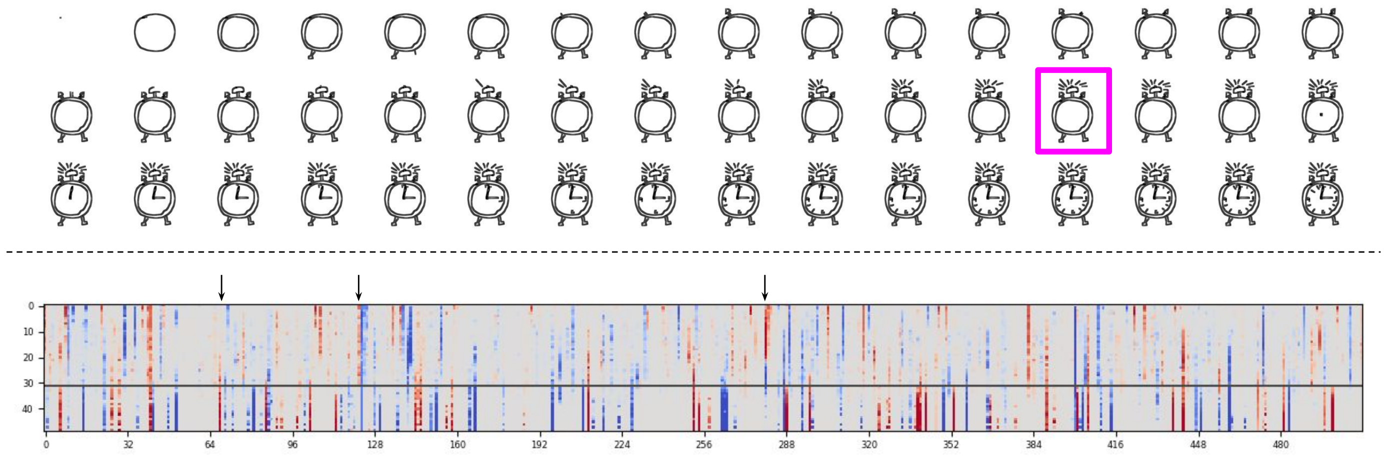

Fig. 11: The pictures at the top correspond to a sketch stroke sequence from the category `alarm clock`. The plot at the bottom corresponds to color-coded hidden state values as generated by our LSTM model. In the plot, the rows correspond to time-steps and columns correspond to dimensions of hidden state (512). The time-step corresponding to the first transition ('no-guess' to first guess-word) is shown in black, spanning all the columns. The stroke sequence member corresponding to first 'guess' is shown with a pink border (see the image sequence at top). Arrows atop selected columns of the plot correspond to hidden dimension entries which plausibly encode the first transition.

deep-learning models in general, tends to be distributed (i.e. spread across multiple dimensions). Therefore, it is difficult to isolate specific dimensions of the hidden-state for analyzing temporal structures (e.g. 'no-guess' to 'guess' transition). Nevertheless, some dimensions seem to encode the 'no-guess' to 'guess' transition in a visible manner. These dimensions are indicated by arrows in the figures. We plan to study the hidden-states and their filtered versions (cell-states of LSTM) in greater detail for future versions of our model.

8 Related Work

Beyond its obvious entertainment value, Pictionary involves a number of social [28], [29], collaborative [30], [31] and cognitive [32], [33] aspects which have been studied by researchers. In an attempt to find neural correlates of creativity, Saggar et al. [34] analyze fMRI data of participants instructed to draw sketches of Pictionary 'action' words (E.g. "Salute","Snore"). In our approach, we ask subjects to guess the word instead of drawing the sketch for a given word. Also, our sketches correspond to nouns (objects).

Human-elicited text-based responses to visual content, particularly in game-like settings, have been explored for object categorization [35], [36]. However, the visual content is static and does not accumulate sequentially, unlike our case. The work of Ullman et al. [37] on determining minimally recognizable image configurations also bears mention. Our approach is complementary to theirs in the sense that we incrementally add stroke content (bottom-up) while they incrementally reduce image content (top-down).

In recent times, deep architectures for sketch recognition [17], [38], [39], [40] have found great success. However, these models are trained to output a single, fixed label regardless of the intra-category variation. In contrast, our model, trained on actual human guesses, naturally exhibits human-like variety in its responses (e.g. a sketch can be guessed as 'aeroplane' or 'warplane' based on evolution of stroke-based appearance). Also, our model solves a much more complex temporally-conditioned, multiple word-embedding regression problem. Another important distinction is that our dataset (WordGuess-160) contains incorrect guesses which usually arise due to ambiguity in sketched depictions. Such 'errors' are normally considered undesirable, but we deliberately include them in the training phase to enable realistic mimicking. This in turn requires our model to implicitly capture the subtle, fine-grained variations in sketch quality – a situation not faced by existing approaches which simply optimize for classification accuracy.

Our dataset collection procedure is similar to the one employed by Johnson et al. [41] as part of their Pictionary-style game Stellasketch. However, we do not let the subject choose the object category. Also, our subjects only provide guesses for stroke sequences of existing sketches and not for sketches being created in real-time. Unfortunately, the Stellasketch dataset is not available publicly for further study.

It is also pertinent to compare our task and dataset with QuickDraw, a large-scale sketch collection initiative by Google (https://github.com/googlecreativelab/quickdraw-dataset). The QuickDraw task generates a dataset of object sketches. In contrast, our task SketchQA results in a dataset of human-generated guess words. In QuickDraw, a sketch is associated with a single, fixed category. In SketchQA, a sketch from an existing dataset is explicitly associated with a list of multiple guess words. In SketchQA, the freedom provided to human guessers enables sketches to have arbitrarily fine-grained labels (e.g. 'airplane', 'warplane','biplane'). However, QuickDraw's label set is fixed. Finally, our dataset (WordGuess-160) captures a rich sequence of guesses in response to accumulation of sketch strokes. Therefore, it can be used to train human-like guessing models. QuickDraw's dataset, lacking human guesses, is not suited for this purpose.

Our computational model employs the Long Short Term

Memory (LSTM) [24] variant of Recurrent Neural Networks (RNNs). LSTM-based frameworks have been utilized for tasks involving temporally evolving content such as as video captioning [6], [42] and action recognition [43], [44], [45]. Our model not only needs to produce human-like guesses in response to temporally accumulated content, but also has the additional challenge of determining how long to 'wait' before initiating the guessing process. Once the guessing phase begins, our model typically outputs multiple answers. These per-time-step answers may even be unrelated to each other. This paradigm is different from a setup wherein a single answer constitutes the output. Also, the output of RNN in aforementioned approaches is a soft-max distribution over *all* the words from a fixed dictionary. In contrast, we use a regression formulation wherein the RNN outputs a word-embedding prediction at each time-step. This ensures scalability with increase in vocabulary and better generalization since our model outputs predictions in a constant-dimension vector space. Lev et al. [46]'s work adopts a similar regression formulation to obtain improved performance for image annotation and action recognition. Das et al. [47]'s work uses a cooperative 'image guessing' game formulation to train agents for dialog-based image retrieval. In our case, the agent modelling focuses on the answers and the answering agent, rather than the drawer and the strokes employed by the mentioned work.

Since our model aims to mimic human-like guessing behavior, a subjective evaluation of generated guesses falls within the ambit of a Visual Turing Test [48], [49], [50]. However, the free-form nature of guess-words and the ambiguity arising from partial stroke information make our task uniquely more challenging.

9 Discussion and Conclusion

We have introduced a novel guessing task called Sketch-QA to crowd-source Pictionary-style open-ended guesses for object line sketches as they are drawn. The resulting dataset, dubbed GuessWord-160, contains $16{,}624$ guess sequences of $1{,}108$ subjects across 160 object categories. We have also introduced a novel computational model which produces open-ended guesses and analyzed its performance on GuessWord-160 dataset for challenging on-line Pictionary-style guessing tasks.

Our computational model stands out in terms of its focus on imitating human game play, even when the corresponding moves are suboptimal. This is in contrast with existing work involving games and questions related to visual content which typically focus on aspects such as accuracy maximization. Our work represents a preliminary but important step towards the realization of agents whose human-like, relatable responses encourage social inclusion with humans. Developing models for Pictionary-like settings could also be useful in other scenarios involving recommender systems. For example, given a question or a task, the agent program could be taught to 'wait' until enough evidence has accumulated rather than provide an answer right away.

In addition to the computational model, our dataset GuessWord-160 can serve researchers studying human perceptions of iconic object depictions [40]. Since the guess-words are paired with object depictions, our data can also aid graphic designers and civic planners in creation of meaningful logos and public signage. This is especially important since incorrectly perceived depictions often result in inconvenience, mild amusement, or in extreme cases, end up deemed offensive. Yet another potential application domain is clinical healthcare. GuessWord-160 consists of partially drawn objects and corresponding guesses across a large number of categories. Such data could be useful for neuro psychiatrists to characterize conditions such as visual agnosia: a disorder in which subjects exhibit impaired object recognition capabilities [51].

In future, we wish to also explore computational models for optimal guessing, i.e. models which aim to guess the sketch category as early and as correctly as possible. In the futuristic context mentioned at the beginning (Figure 1), such models would help the robot contribute as a productive team-player by correctly guessing its team-member's sketch as early as possible. In our dataset, each stroke sequence was shown only to a single subject and therefore, is associated with a single corresponding sequence of guesses. This shortcoming is to be mitigated in future editions of Sketch-QA. A promising approach for data collection would be to use digital whiteboards, high-quality microphones and state-of-the-art speech recognition software to collect realistic paired stroke-and-guess data from Pictionary games in home-like settings [52]. It would also be worthwhile to consider Sketch-QA beyond object names ('nouns') and include additional lexical types (e.g. action-words and abstract phrases). We believe the resulting data, coupled with improved versions of our computational models, could make the scenario from Figure 1 a reality one day.

References

[1] R. K. Sarvadevabhatla, S. Surya, T. Mittal, and R. V. Babu, "Game of sketches: Deep recurrent models of pictionary-style word guessing," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16486

[2] G. Tesauro, "TD-gammon, a self-teaching backgammon program, achieves master-level play," *Neural Computation*, vol. 6, no. 2, pp. 215–219, 1994.

[3] *Deep Blue Versus Kasparov: The Significance for Artificial Intelligence*. AAAI Press, 1997.

[4] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] X. Chen and C. Lawrence Zitnick, "Mind's eye: A recurrent visual representation for image caption generation," in *CVPR*, 2015, pp. 2422–2431.

[6] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko, "Sequence to sequence-video to text," in *CVPR*, 2015, pp. 4534–4542.

[7] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention." in *ICML*, vol. 14, 2015, pp. 77–81.

[8] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "VQA: Visual question answering," in *ICCV*, 2015, pp. 2425–2433.

[9] H. Xu and K. Saenko, "Ask, attend and answer: Exploring question-guided spatial attention for visual question answering," in *ECCV*. Springer, 2016, pp. 451–466.

[10] M. Ren, R. Kiros, and R. Zemel, "Exploring models and data for image question answering," in *NIPS*, 2015, pp. 2953–2961.

[11] M. Eitz, J. Hays, and M. Alexa, "How do humans sketch objects?" *ACM Trans. on Graphics*, vol. 31, no. 4, p. 44, 2012.

[12] R. G. Schneider and T. Tuytelaars, "Sketch classification and classification-driven analysis using fisher vectors," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 174:1–174:9, Nov. 2014.

[13] P. Halácsy, A. Kornai, and C. Oravecz, "HunPos: an open source trigram tagger," in *Proc. ACL on interactive poster and demonstration sessions*, 2007, pp. 209–212.

[14] D. Lachowicz, "Enchant spellchecker library," 2010.

[15] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[16] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *ACL*. Association for Computational Linguistics, 1994, pp. 133–138.

[17] R. K. Sarvadevabhatla, J. Kundu, and R. V. Babu, "Enabling my robot to play pictionary: Recurrent neural networks for sketch recognition," in *ACMMM*, 2016, pp. 247–251.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[20] P. Sangkloy, N. Burnell, C. Ham, and J. Hays, "The sketchy database: learning to retrieve badly drawn bunnies," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 119, 2016.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[22] T. Qin, X.-D. Zhang, M.-F. Tsai, D.-S. Wang, T.-Y. Liu, and H. Li, "Query-level loss functions for information retrieval," *Information Processing & Management*, vol. 44, no. 2, pp. 838–855, 2008.

[23] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov *et al.*, "Devise: A deep visual-semantic embedding model," in *NIPS*, 2013, pp. 2121–2129.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *JMLR*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[26] J. C. Chan, "Response-order effects in likert-type scales," *Educational and Psychological Measurement*, vol. 51, no. 3, pp. 531–540, 1991.

[27] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945. [Online]. Available: http://www.jstor.org/stable/3001968

[28] T. B. Wortham, "Adapting common popular games to a human factors/ergonomics course," in *Proc. Human Factors and Ergonomics Soc. Annual Meeting*, vol. 50. SAGE, 2006, pp. 2259–2263.

[29] F. Mäyrä, "The contextual game experience: On the socio-cultural contexts for meaning in digital play," in *Proc. DIGRA*, 2007, pp. 810–814.

[30] N. Fay, M. Arbib, and S. Garrod, "How to bootstrap a human communication system," *Cognitive science*, vol. 37, no. 7, pp. 1356–1367, 2013.

[31] M. Groen, M. Ursu, S. Michalakopoulos, M. Falelakis, and E. Gasparis, "Improving video-mediated communication with orchestration," *Computers in Human Behavior*, vol. 28, no. 5, pp. 1575 – 1579, 2012.

[32] D. M. Dake and B. Roberts, *The Visual Analysis of Visual Metaphor*. ERIC, 1995.

[33] B. Kievit-Kylar and M. N. Jones, "The semantic pictionary project," in *Proc. Annual Conf. Cog. Sci. Soc.*, 2011, pp. 2229–2234.

[34] M. Saggar, E.-M. Quintin, E. Kienitz, N. T. Bott, Z. Sun, W.-C. Hong, Y.-h. Chien, N. Liu, R. F. Dougherty, A. Royalty *et al.*, "Pictionary-based fMRI paradigm to study the neural correlates of spontaneous improvisation and figural creativity," *Scientific reports*, vol. 5, p. 10894, 2015.

[35] L. Von Ahn and L. Dabbish, "Labeling images with a computer game," in *SIGCHI*. ACM, 2004, pp. 319–326.

[36] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie, "Visual recognition with humans in the loop," in *European Conference on Computer Vision*. Springer, 2010, pp. 438–451.

[37] S. Ullman, L. Assif, E. Fetaya, and D. Harari, "Atoms of recognition in human and computer vision," *PNAS*, vol. 113, no. 10, pp. 2744–2749, 2016.

[38] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, and T. M. Hospedales, "Sketch-a-net that beats humans," *arXiv preprint arXiv:1501.07873*, 2015.

[39] O. Seddati, S. Dupont, and S. Mahmoudi, "Deepsketch: deep convolutional neural networks for sketch recognition and similarity search," in *CBMI*. IEEE, 2015, pp. 1–6.

[40] R. K. Sarvadevabhatla and R. V. Babu, "Eye of the dragon: Exploring discriminatively minimalist sketch-based abstractions for object categories," in *ACMMM*, 2015, pp. 271–280.

[41] G. Johnson and E. Y.-L. Do, "Games for sketch data collection," in *Proceedings of the 6th eurographics symposium on sketch-based interfaces and modeling*. ACM, 2009, pp. 117–123.

[42] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *CVPR*, 2015, pp. 2625–2634.

[43] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei, "Every moment counts: Dense detailed labeling of actions in complex videos," *arXiv preprint arXiv:1507.05738*, 2015.

[44] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *CVPR*, 2015, pp. 4694–4702.

[45] S. Ma, L. Sigal, and S. Sclaroff, "Learning activity progression in lstms for activity detection and early detection," in *CVPR*, 2016, pp. 1942–1950.

[46] G. Lev, G. Sadeh, B. Klein, and L. Wolf, "RNN fisher vectors for action recognition and image annotation," in *ECCV*. Springer, 2016, pp. 833–850.

[47] A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra, "Learning cooperative visual dialog agents with deep reinforcement learning," in *CVPR*, 2017, pp. 2951–2960.

[48] D. Geman, S. Geman, N. Hallonquist, and L. Younes, "Visual turing test for computer vision systems," *PNAS*, vol. 112, no. 12, pp. 3618–3623, 2015.

[49] M. Malinowski and M. Fritz, "Towards a visual turing challenge," *arXiv preprint arXiv:1410.8027*, 2014.

[50] H. Gao, J. Mao, J. Zhou, Z. Huang, L. Wang, and W. Xu, "Are you talking to a machine? dataset and methods for multilingual image question," in *NIPS*, 2015, pp. 2296–2304.

[51] L. Baugh, L. Desanghere, and J. Marotta, "Agnosia," in *Encyclopedia of Behavioral Neuroscience*. Academic Press, Elsevier Science, 2010, vol. 1, pp. 27–33.

[52] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta, "Hollywood in homes: Crowdsourcing data collection for activity understanding," in *ECCV*, 2016.

Ravi Kiran Sarvadevabhatla is an Assistant Professor at International Institute of Information Technology-Hyderabad (IIIT-H), India. He received his Ph.D. from Dept. of Computational and Data Sciences, Indian Institute of Science, Bangalore, India. He has broad-ranging research interests and likes to work on problems involving multi-modal multimedia data (e.g. images, videos, text, audio/speech, eye-tracking data) and multiple disciplines (e.g. Humanities, Graphics, Robotics, Human-Computer Interaction).

Shiv Surya is an Applied Scientist at Amazon. Before joining Amazon, he worked as a Project Staff Member at Video Analytics Lab, Indian Institute of Science (2016-2017). He obtained his MS in Electrical Engineering from University of Southern California, Los Angeles, USA in 2015. He obtained his B.Tech degree in Electrical and Electronics Engineering from Rashtreeya Vidayalaya College of Engineering, Bangalore, India in 2013. His research interests span visual understanding, domain-adaptation , dense object counting and machine learning for advertising.

Trisha Mittal is a MS student in Computer Science at University of Maryland, College Park, USA. She obtained her B. Tech. and M. Tech. (Bachelors and Masters of Technology) in Information Technology degree from International Institute of Information Technology Bangalore, India. For her M.Tech. thesis, she also explored the use of Reinforcement Learning in generative tasks.

R. Venkatesh Babu is an associate professor in Department of Computational and Data Sciences, Indian Institute of Science, Bangalore. He received his Ph.D. degree from the Dept. of Electrical Engineering, Indian Institute of Science, Bangalore. Thereafter, he held postdoctoral positions at NTNU, Norway and IRISA/INRIA, Rennes, France. Subsequently he worked as a research fellow at NTU, Singapore. He spent couple of years working in industry before joining IISc in August 2010. His research interests span signal processing, compression, machine vision, image/video processing, pattern recognition and multimedia. He is a senior member of IEEE.