Two-Phase Model: Description, Ablations and Performance

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

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

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 1: 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]).

1 Introduction

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 design 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 1).

2 Two-Phase Baseline Model

Typically, a guess sequence contains two distinct phases. In the first phase, no guesses are provided by the subject since the accumulated strokes provide insufficient evidence. At a later stage, the subject feels confident enough to provide the first guess. Thus, the location of this first guess (within the overall sequence) is the starting point for the second phase. The first phase (i.e. no guesses) offers no usable guess-words. Therefore, rather than tackling both the phases within a single model, we adopt a divide-and-conquer approach. We design this baseline to first predict the phase transition location (i.e. where the first guess occurs). Conditioned on this location, the model predicts guess-word representations for rest of the sequence (see Figure 1).

In the two-phase model and the model described in the main paper, the guess-word generator is a common component. The guess-word generation model is already described in the main paper. For the remainder of the section, we focus on the first phase of the two-phase baseline.

Consider a typical guess sequence $\mathbb{G}_\mathbb{I}$. Suppose the first phase ('no guesses') corresponds to an initial sub-sequence of length k. The second phase then corresponds to the remainder sub-sequence of length $(N-k)$. Denoting 'no guess' as 0 and a guess-word as 1, $\mathbb{G}_\mathbb{I}$ is transformed to a binary sequence $\mathbb{B}_\mathbb{I} = [(0, 0 \ldots k \text{ times })(1, 1 \ldots (N-k) \text{ times})]$. Therefore, the objective for the Phase I model is to correctly predict the transition index i.e. $(k+1)$.

2.1 Phase I model (Transition prediction)

Two possibilities exist for Phase-I model. The first possibility is to train a CNN model using sequence members from $\mathbb{I}, \mathbb{B}_\mathbb{I}$ pairs for binary (Guess/No Guess) classification and during inference, repeatedly apply the CNN model on successive time-steps, stopping when the CNN model outputs 1 (indicating the beginning of guessing phase). The second possibility is to train an RNN and during inference, stop unrolling when a 1 is encountered. We describe the setup for CNN model first.

2.1.1 CNN model

For the CNN model, we fine-tune VGG-16 object classification model [1] using Sketchy [2] as in the proposed model. The fine-tuned model is used to initialize another VGG-16 model, but with a 256-dimensional bottleneck layer introduced after the penultimate (fc_7) layer. Let us denote this model as Q_1.

2.1.2 Sketch representation

As feature representations, we consider two possibilities:app [a] Q_1 is fine-tuned for 2-way classification (Guess/No Guess). The 256-dimensional output from final fully-connected layer forms the feature representation. [b] The architecture in option [a] is modified by having 160-way class prediction as an additional, auxiliary task. This choice is motivated by the possibility of encoding category-specific transition location statistics within the 256-dimensional feature representation (see Figure 2). The two losses corresponding to the two outputs (2-way and 160-way classification) of the modified architecture are weighted equally during training.

Loss weighting for imbalanced label distributions: When training the feature extraction CNN (Q_1) in Phase-I, we encounter imbalance in the distribution of no-guesses (0s) and guesses (1s). To mitigate this, we employ class-based loss weighting [3] for the binary classification task. Suppose the

Manuscript received January, 2018

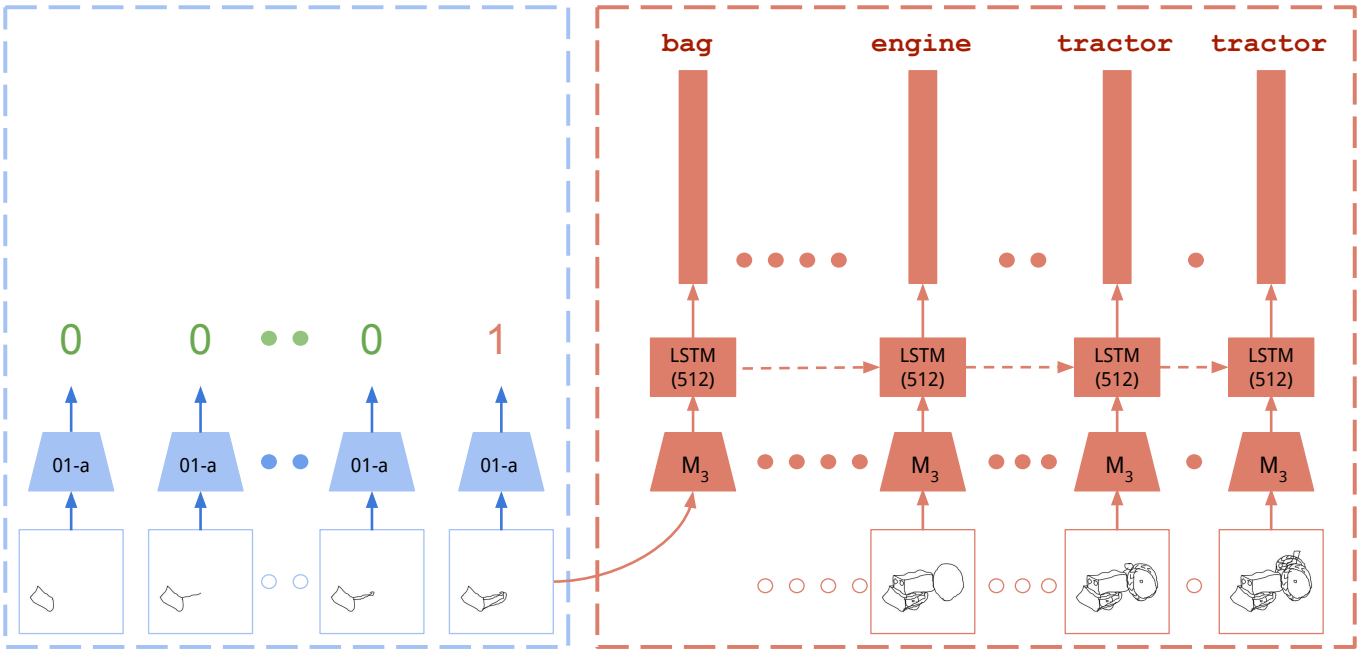

Fig. 1: Architecture for the two-phase baseline. The first phase (blue dotted line) is used to predict location of the transition to the word-guessing phase (output 1). Starting from transition location, the second-phase (red dotted line) sequentially outputs word-embedding predictions until the end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

CNN model	LSTM	Loss	Window width		
			1	3	5
01	–	CCE	17.37	36.57	49.67
01-a	–	CCE	**20.45**	41.22	**54.91**
01-a	64	Seq	17.30	38.40	52.75
01-a	128	Seq	18.94	39.25	53.47
01-a	256	Seq	18.68	40.04	53.41
01-a	512	Seq	18.22	39.45	54.78
01-a	128	wSeq	19.20	**41.48**	**55.64**
01-a	128	mRnk	18.87	37.88	52.23

TABLE 2: The transition location prediction accuracies for various Phase I architectures are shown. 01 refers to the binary output CNN model pre-trained for feature extraction. 01-a refers to the 01 CNN model with 160-way auxiliary classification. The last two rows correspond to test set accuracies of the best CNN and LSTM configurations. For the 'Loss' column, CCE = Categorical-cross entropy, Seq = Average sequence loss, wSeq = Weighted sequence loss, mRnk = modified Ranking Loss. The results are shown for 'Window width' sized windows centered on ground-truth transition location. The rows below dotted line show performance of best CNN and LSTM models on test sequences.

number of no-guess samples is n and the number of guess samples is g. Let $\mu = \frac{n+g}{2}$. The weights for the classes are computed as $w_0 = \frac{\mu}{f_0}$ where $f_0 = \frac{n}{(n+g)}$ and $w_1 = \frac{\mu}{f_1}$ where $f_1 = \frac{g}{(n+g)}$. The binary cross-entropy loss is then computed as:

$$\mathcal{L}(P,G) = \sum_{x \in \mathcal{X}_{train}} -w_x[g_x log(p_x) + (1-g_x) log(1-p_x)] \quad (1)$$

where g_x, p_x stand for ground-truth and prediction respectively and $w_x = w_0$ when x is a no-guess sample and $w_x = w_1$ otherwise. For our data, $w_0 = 1.475$ and $w_1 = 0.765$, thus appropriately accounting for the relatively smaller number of no-guess samples in our training data.

A similar procedure is also used for weighting losses when the 160-way auxiliary classifier variant of Q_1 is trained. In this case, the weights are determined by the per-object category distribution of the training sequences. Experimentally, Q_1 with auxiliary task shows better performance – see first two rows of Table 2.

2.1.3 LSTM setup

We use the 256-dimensional output of the Q_1-auxiliary CNN as the per-timestep sketch representation fed to the LSTM model. To capture the temporal evolution of the binary sequences, we configure the LSTM to output a binary label $B_t \in \{0, 1\}$ for each timestep t. For the LSTM, we explored variations in number of hidden units $(64, 128, 256, 512)$. The weight matrices are initialized as orthogonal matrices with a gain factor of 1.1 [4] and the forget gate bias is set to 1. For training the LSTMs, we use the average sequence loss, computed as the average of the per-time-step binary cross-entropy losses. The loss is regularized by a standard

CNN model	LSTM	α	Window width		
			1	3	5
01-a	128	5	19.00	41.55	55.44
01-a	128	7	19.20	**41.48**	**54.85**
01-a	128	10	18.48	40.10	54.06

TABLE 3: Weighted loss performance for various values of α.

L_2-weight norm weight-decay parameter ($\alpha = 0.0005$). For optimization, we use Adagrad with a learning rate of 5×10^{-5} and the momentum term set to 0.9. The gradients are clipped to 5.0 during training. For all LSTM experiments, we use a mini-batch size of 1.

2.1.4 LSTM Loss function variants

The default sequence loss formulation treats all time-steps of the sequence equally. Since we are interested in accurate localization of transition point, we explored the following modifications of the default loss for LSTM:

Transition weighted loss: To encourage correct prediction at the transition location, we explored a weighted version of the default sequence-level loss. Beginning at the transition location, the per-timestep losses on either side of the transition are weighted by an exponentially decaying factor $e^{-\alpha(1-[t/(k+1)]^s)}$ where $s = 1$ for time-steps $[1, k]$, $s = -1$ for $[k+2, N]$. Essentially, the loss at the transition location is weighted the most while the losses for other locations are downscaled by weights less than 1 – the larger the distance from transition location, the smaller the weight. We tried various values for α. The localization accuracy can be viewed in Table 3. Note that the weighted loss is added to the original sequence loss during actual training.

Modified ranking loss: We want the model to prevent occurrence of premature or multiple transitions. To incorporate this notion, we use the ranking loss formulation proposed by Ma et al. [5]. Let us denote the loss at time step t as $\mathcal{L}_c^t$ and the softmax score for the ground truth label y_t as $p_t^{y_t}$. We shall refer to this as detection score. In our case, for the Phase-I model, $\mathcal{L}_c^t$ corresponds to the binary cross-entropy loss. The overall loss at time step t is modified as:

$$\mathcal{L}^t = \lambda_s \mathcal{L}_c^t + \lambda_r \mathcal{L}_r^t \tag{2}$$

We want the Phase-I model to produce monotonically non-decreasing softmax values for no-guesses and guesses as it progresses more into the sub-sequence. In other words, if there is no transition at time t, i.e. $y_t = y_{t-1}$, then we want the current detection score to be no less than any previous detection score. Therefore, for this situation, the ranking loss is computed as:

$$\mathcal{L}_r^t = max(0, p_t^{*y_t} - p_t^{y_t}) \tag{3}$$

where

$$p_t^{*y_t} = \max_{t' \in [t_s, t-1]} p_{t'}^{y_t} \tag{4}$$

where t_s corresponds to time step 1 when $y_t = 0$ (No Guesses) or $t_s = t_p$ (starting location of Guessing).
If time-step t corresponds to a transition, i.e. $y_t \neq y_{t-1}$, we want the detection score of previous phase ('No Guess') to be as small as possible (ideally 0). Therefore, we compute the ranking loss as:

$$\mathcal{L}_r^t = p_t^{y_{t-1}} \tag{5}$$

During training, we use a convex combination of sequence loss and the ranking loss with the loss weighting determined by grid search over λ_{s-r} (see Table 4). From our experiments, we found the transition weighted loss to provide the best performance (Table 2).

2.1.5 Evaluation

At inference time, the accumulated stroke sequence is processed sequentially by Phase-I model until it outputs a 1 which marks the beginning of Phase-II. Suppose the predicted transition index is p and ground-truth index is g. The prediction is deemed correct if $p \in [g-\delta, g+\delta]$ where δ denotes half-width of a window centered on p. For our experiments, we used $\delta \in \{0, 1, 2\}$. The results (Table 2) indicate that the Q_1-auxiliary CNN model outperforms the best LSTM model by a very small margin. The addition of weighted sequence loss to the default version plays a crucial role in the latter (LSTM model) since the default version does not explicitly optimize for the transition location. Overall, the large variation in sequence lengths and transition locations explains the low performance for exact ($k = 1$) localization. Note, however, that the performance improves considerably when just one to two nearby locations are considered for evaluation ($k = 3, 5$).

During inference, the location predicted by Phase-I model is used as the starting point for Phase-II (word guessing). The Phase-II model is virtually identical in design as the main Unified model described in the main paper.

2.2 Overall Results

To determine overall performance, we utilize the best architectural settings as determined by validation set performance. We then merge validation and training sets, re-train the best models and report their performance on the test set. As the overall performance measure, we report two items on the test set – [a] *P-II*: the fraction of correct matches with respect to the subsequence corresponding to ground-truth word guesses. In other words, we assume 100% accurate localization during Phase I and perform Phase II inference beginning from the ground-truth location of the first guess. [b] *Full:* We use Phase-I model to determine transition location. Note that depending on predicted location, it is possible that we obtain word-embedding predictions when the ground-truth at the corresponding time-step corresponds to 'no guess'. Regarding such predictions as mismatches, we compute the fraction of correct matches for the full sequence. As a baseline model (first row of Table 5), we use outputs of the best performing per-frame CNNs from Phase I and Phase II.

The results (Table 5) show that the Unified model outperforms Two-Phased model by a significant margin. For

CNN model	LSTM	λ_s, λ_r	Window width		
			1	3	5
01-a	128	$0.5, 1.0$	17.43	35.26	48.23
01-a	128	$1, 1$	18.41	39.45	53.08
01-a	128	$1, 0.5$	18.87	37.88	52.23

TABLE 4: Ranking loss performance for various weighting of sequence loss and rank loss.

P-I	P-II	Average sequence-level accuracy					
		$k = 1$		$k = 3$		$k = 5$	
		P-II only	Full	P-II only	Full	P-II only	Full
01-a	M_3	54.06	43.61	64.11	51.54	66.85	54.18
Unified	Unified	46.35	**62.04**	56.45	**69.35**	59.30	**71.11**
01-a	$R2_5$	**57.05**	46.33	**64.76**	52.08	**67.19**	54.46

TABLE 5: Overall average sequence-level accuracy on test set are shown for guessing models (CNNs only baseline [first row], Unified [second], Two Phased [third]). $R2_5$ corresponds to best Phase-II LSTM model.

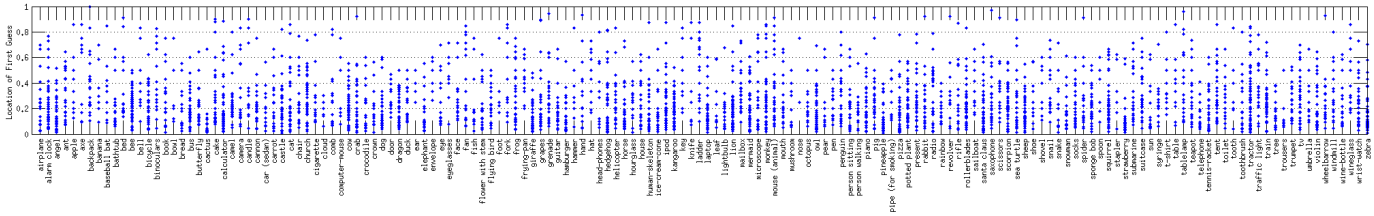

Fig. 2: The distribution of first guess locations normalized ($[0, 1]$) over sequence lengths (y-axis) across categories (x-axis).

Phase-II model, the objective for CNN (whose features are used as sketch representation) and LSTM are the same. This is not the case for Phase-I model. The reduction in long-range temporal contextual information, caused by splitting the original sequence into two disjoint sub-sequences, is possibly another reason for lower performance of the Two-Phased model.

REFERENCES

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

[2] 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.

[3] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2650–2658.

[4] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *CoRR*, vol. abs/1312.6120, 2013. [Online]. Available: http://arxiv.org/abs/1312.6120

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