Edinburgh Research Explorer

Goal-Driven Sequential Data Abstraction

OPEN ACCESS

Goal-Driven Sequential Data Abstraction

Umar Riaz Muhammad[1]   Yongxin Yang[1]   Timothy M. Hospedales[2]   Tao Xiang[1]   Yi-Zhe Song[1]

[1]University of Surrey   [2]University of Edinburgh

{u.muhammad, yongxin.yang, t.xiang, y.song}@surrey.ac.uk, t.hospedales@ed.ac.uk

Abstract

Automatic data abstraction is an important capability for both benchmarking machine intelligence and supporting summarization applications. In the former one asks whether a machine can 'understand' enough about the meaning of input data to produce a meaningful but more compact abstraction. In the latter this capability is exploited for saving space or human time by summarizing the essence of input data. In this paper we study a general reinforcement learning based framework for learning to abstract sequential data in a goal-driven way. The ability to define different abstraction goals uniquely allows different aspects of the input data to be preserved according to the ultimate purpose of the abstraction. Our reinforcement learning objective does not require human-defined examples of ideal abstraction. Importantly our model processes the input sequence holistically without being constrained by the original input order. Our framework is also domain agnostic – we demonstrate applications to sketch, video and text data and achieve promising results in all domains.

1. Introduction

Abstraction is generally defined in the context of specific applications [5, 20, 39, 7, 23, 27]. In most cases it refers to elimination of redundant elements, and preservation of the most salient and important aspects of the data. It is an important capability for various reasons: compression [12] and saving human time in viewing the data [29]; but also improving downstream data analysis tasks such as information retrieval [2], and synthesis [14, 27].

We present a novel goal-driven abstraction task for sequential data (see Fig. 1). Sequential refers to data with temporal order – we consider video, sequentially drawn sketches, and text. Goal-driven refers to preservation of a certain aspect of the input according to a specific abstraction objective or goal. The same input may lead to different abstracted outputs depending on the abstraction goal. For example, prioritizing preserving sentiment vs. helpfulness in product review text could lead to different sum-
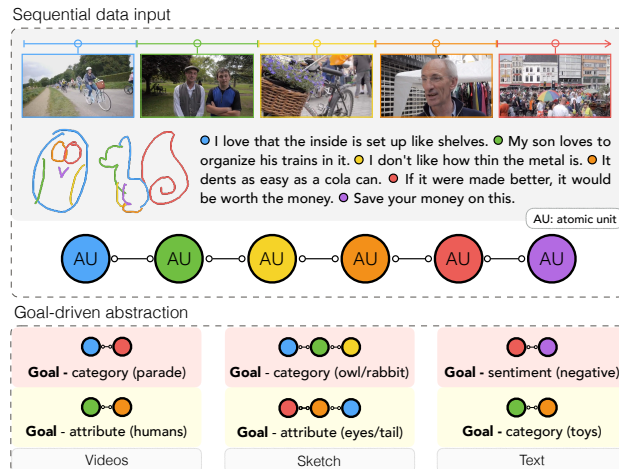

Figure 1. An illustration of our goal-driven abstraction task. Each input (video, sketch or text) consists of a sequence of atomic units (AUs) corresponding to video-segments, strokes and sentences for the three input domains, respectively. The AUs are color coded. Depending on the abstraction goal, different AUs are preserved in each abstracted output.

maries. It is important not to confuse our new goal-driven abstraction setting with traditional video/text summarization [15, 16, 42, 51, 37, 6, 28, 30, 46]. The aims are different: the latter produces a *single* compact but diverse and representative summary, often guided by human annotations, while ours yields *various* goal-conditional compact summaries. Our problem setting is also more amenable to training without ground-truth labels (i.e., manual gold-standard but subjective summaries) commonly required by contemporary video/text summarization methods.

To tackle this novel problem, new approaches are needed. To this end, we propose a goal-driven sequential data abstraction model with the following key properties: (1) It processes the input sequence holistically rather than being constrained by the original input order. (2) It is trained by reinforcement learning (RL), rather than supervised learning. This means that expensively annotated data in the form of target abstractions are not required. (3) Different goals are introduced via RL reward functions. Besides eliminating the annotation requirement, this enables

preserving different aspects of the input according to the purpose of the abstraction. (4) Finally, the RL-based approach also allows abstracted outputs of any desired length to be composed by varying the abstraction budget.

We demonstrate the generality of our approach through three very different sequential data domains: free-hand sketches, videos and text. Video and text are sequential data domains widely studied in the past. While sketch may not seem obviously sequential, touchscreen technology means that all prominent sketch datasets now record vectorized stroke sequences. For instance, QuickDraw [17], the largest sketch dataset to date, provides vectorized sequence data in the form of (x, y) pen coordinates and state p (touching or lifting). For sketch and video, we train two reward functions based on *category* and *attribute* recognition models. These drive our abstraction model to abstract an input sketch/video into a shorter sequence, while selectively preserving either category or attribute related information. For text, we train three reward functions on product reviews, based on *sentiment*, *product-category* and *helpfulness* recognition models. These drive our model to summarize an input document into a shorter paragraph that preserves sentiment/category/helpfulness information respectively.

The main contributions of our work are: (1) Defining a novel goal-driven abstraction problem, (2) a sequential data abstraction model trained by RL, that processes the input holistically without being constrained by the original input order, and (3) demonstrating flexibility of this model to diverse sequential data domains including sketch, video and text.

2. Related work

Video/text summarization Existing models are either supervised or unsupervised. Unsupervised summarization models in video [9, 31, 40, 41, 43, 50, 52, 54, 55] and text [10, 26, 25, 3] domains aim to identify a small subset of key units (video-segments/sentences) that preserve the global content of the input, e.g., using criteria like diversity and representativeness. In contrast, supervised video [13, 15, 16, 42, 49, 51] and text [37, 6, 28, 30, 46] summarization methods solve the same problem by employing ground-truth summaries as training targets. Both types of models are not driven by specific goals and are evaluated on human annotated ground-truth summaries – how humans summarize a given video/text is subjective and often ambiguous. Neither of these models thus address our new goal-driven abstraction setting.

A recent work [53] trains video summarization model in a weakly supervised RL setting using category level video labels. The aim is to produce summaries with the added criteria of category level recognizability along with the usual criteria of diversity and representativeness. The core mech-

anism is to process video-segments in sequence and make binary decisions (keep or remove) for each segment, following the above criteria. In this work we introduce a goal-driven approach to explicitly preserve any quantifiable property, whether category information (as partially done in [53]), attributes, or potentially other quantities such as interestingness [11]. We show that our model is superior to [53] thanks to the holistic modeling of the sequential input without restriction by its original order (see Sec. 4.2).

Sketch abstraction Comparing to video and text, much less prior work on sketch abstraction exists. This problem was first studied in [4] where a data-driven approach was used to study abstraction in professionally drawn facial portraits. Sketches at varying abstraction levels were collected by limiting the time (from four and a half minutes to fifteen seconds) given to an artist to sketch a reference photo. In recent work [27], automatic abstraction was studied explicitly for the first time in free-hand amateur sketches. The abstraction process was defined as a trade-off between recognizability and brevity/compactness of the sketch. The abstraction model, also based on RL, processed stroke-segments in sequence and made binary decisions (keep or remove) for each segment, but otherwise output strokes in the same order as they were drawn. In this work we also optimize the trade-off between recognizability and compactness (if the goal is recognizability). However, crucially, our method benefits from processing the input holistically rather than in its original order, and learns an optimal stroke sequencing strategy. We show that our approach clearly outperforms [27] (see Sec. 4.1). Further, we demonstrate application to diverse domains of sketch, video and text, and uniquely explore the ability to use multiple goal functions to obtain different abstractions.

Sketch recognition Early sketch recognition methods were developed to deal with professionally drawn sketches as in CAD or artistic drawings [18, 22, 36]. The more challenging task of free-hand sketch recognition was first tackled in [8] along with the release of the first large-scale dataset of amateur sketches. Since then the task has been well-studied using both classic vision [34, 21] as well as deep learning approaches [48]. Recent successful deep learning approaches have spanned both primarily non-sequential CNN [48, 47] and sequential RNN [19, 33] recognizers. We use both CNN and RNN-based multi-class classifiers to provide rewards for our RL based sketch abstraction framework.

3. Methodology

Our aim is to input a data sequence and output a shorter sequence that preserves a specific type of information according to a goal function. To this end, a goal-driven sequence abstraction (GDSA) model is proposed. GDSA processes the input sequence data holistically by first decom-
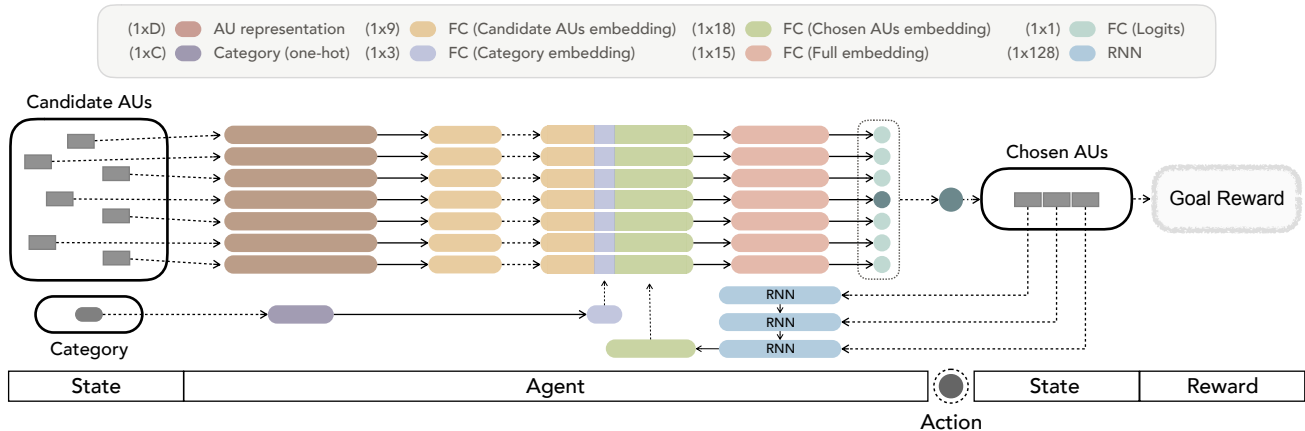
Figure 2. A schematic illustration of the proposed GDSA agent. The agent iteratively chooses AUs from the candidate pool so as to maximize the recognizability goal of the abstracted sketch/text/video. Solid arrows represent trainable weights.

posing it into a set of atomic-units (AUs), which form a pool of candidates for selection. GDSA is trained by RL to produce abstractions by picking a sequence of AUs from the pool. The output sequence should be shorter than the input (controlled by the budget) while preserving its information content (controlled by the RL reward/goal function).

3.1. Goal-driven sequence abstraction (GDSA)

The sequential data abstraction task is formalized as a Markov decision process. At each step, our GDSA agent moves one atomic unit (AU) from a pool of candidate AUs to a list of chosen AUs, and it stops when the number of chosen AUs is larger than a fixed budget. The agent is trained via RL [38] using a reward scheme which encourages it to outperform the efficiency of the original input order in terms of preserving goal-related information in the sequence given a limited length budget. Concretely, we have two data-structures: the candidate AU pool and the chosen AU list. The chosen AUs list starts empty, and the candidate AUs pool contains the full input. AUs are then picked by the agent from the candidate pool, one at a time, to be appended to the current chosen AU list.

A schematic of the GDSA agent is shown in Fig. 2. The core idea is to evaluate the choice of each candidate AU in the context of all previously chosen AUs and the category to which the input sequence belongs. We do this by learning embeddings for candidate AUs, chosen AUs, and the input sequence category label respectively. Based on these embeddings, the GDSA agent will iteratively pick the next best AU to output given those chosen so far.

Candidate AU embedding   At each iteration, every AU in the candidate pool is considered by GDSA as a candidate for the next output. To this end, first each AU is: (1) Encoded as a fixed-length vector. Note that each AU may itself contain sequential sub-structure (sketch strokes formed by segments, video segments formed by frames, or sentences formed by words), so we use a domain-specific pre-trained

RNN to embed each AU. The hidden RNN cell state corresponding to the last sub-entry of the AU is extracted and used to represent the AU as a fixed-length vector. (2) Assigned a time-stamp from 1 to 10 based on the relative position in the original input sequence w.r.t the total number of AUs. This is introduced so that during training our model can leverage information from the input sequence order. This one-hot time-stamp vector is then concatenated with the fixed-length RNN encoding vector above, and these are fed into a fully-connected (FC) layer to get the candidate AU embedding.

Chosen AU embedding   To represent the output sequence so far, all the AUs of the chosen AU list are fed sequentially to an RNN. Each AU corresponds to a time-step in the RNN. The output of the last time-step is then fed into a FC layer to get the chosen-AU list embedding. At the first time step the list is empty, represented by a zero vector.

Category embedding   There are often multiple related abstraction tasks within a domain, e.g., the object/document category in sketch/text abstraction. We could train an independent GDSA model per category, or aggregate the training data across all categories. These suffer respectively from less training data, and a mixing of category/domain-specific nuances. As a compromise we embed a category identifier to allow the model to exploit some information sharing, while also providing guidance about category differences [44].

Action selection   At each iteration, our agent takes an action (pick an AU from the candidate pool) given the category and chosen AUs so far. To this end, it considers each candidate AU in turn and concatenates it with the other two embeddings, before feeding the result into a FC layer to get a complete state-action embedding. This is then fed into a FC layer with 1 neuron (i.e. scalar output) to produce the final logit. Once all candidate AUs are processed, their corresponding logit values are concatenated and form a multinomial distribution via softmax. During training, we

sample this multinomial, and during testing the largest logit is always chosen. The picked AU is then removed from the candidate pool and appended to the list of chosen AUs. This process is repeated until a budget is exhausted.

Domain specific details   We apply our framework to sketch, video and text data. Each sketch is composed of a sequence of AUs corresponding to strokes. For video, each input is a video-clip and segments in the clip are AUs. For text, each input is a document containing a product review, and sentences are AUs. Another domain specific property is on how to present the agent-picked AUs as the final output of the abstraction. In the case of video and text, the selected AUs are kept in the same order as in the original input order, to maintain the coherence of output sequence. While for sketch, we keep the order in which AUs are picked, since the model can potentially learn a better sequencing strategy than the natural human input.

3.2. Goal-driven reward function

The objective of our GDSA agent is to choose AUs that maximally preserve the goal information. In particular, we leverage the natural input sequence, along with the random AU selection, to define a novel reward function r_t as

$$r_t = (a_t - (\delta\, h_t + (1-\delta)\, g_t))b, \qquad (1)$$

where t is the time step[1], a_t is agent performance (at goal information preservation), h_t is the performance obtained by picking the AUs following the original input order, and g_t is the performance of random order policy. The performance is evaluated according to the recognizability of the goal information to be preserved after adding the selected AU to the chosen AUs list. δ is an annealing parameter that balances comparison against human and random-policy performance. It is initialised to 0, so the agent receives positive reward as long as it beats the random policy. During training δ is increased towards 1, thus defining a curriculum that progressively requires the agent to perform better in order to obtain a reward. In detail, δ is increased from 0 to 1 linearly in K steps, where K is the total number of episodes used during training. So by the end of training, the agent's selection has to beat the original input sequence to obtain positive reward. Finally, b is a reward scaling factor. For example, given a 100-stroke sketch and budget of 10%, GDSA has to pick 10 strokes more informative than a random selection to obtain reward at the start, and more informative than the first 10 strokes of the input to obtain reward at the end.

Goals   For sketches, one abstraction goal is the recognizability of the output sequential object sketch. We quantify this by the resulting classification accuracy under a multiclass classifier (thus defining a_t, h_t and g_t in the reward

---

[1]A time step means a pass through the candidate AU pool, leading to the selection of a chosen AU.

function). To demonstrate driving abstraction by different goals, we explore rewarding preservation of other information about the sketch. Specifically, we train a sketch attribute detector to define an attribute preservation reward. For videos, the main target information to be preserved is the recognizability of the video category. To guide training we employ a multi-class classifier, which is plugged into the reward function to compute a_t, h_t and g_t values at each time step. We also consider another abstraction goal of preserving attributes in videos by employing an attribute detector to define the reward. For text, the main goal is sentiment preservation in product reviews, and the reward is given by probability of the review summary being correctly classified by a binary sentiment classifier. As different abstraction goals, we also explore the preservation of product-category and helpfulness information by training separate classifiers for these goals.

3.3. Training procedure

Variable action space   In a conventional reinforcement learning (RL) framework, the observation and action space dimensions are both fixed. In our framework, because the number of candidate AUs is decremented at each step, the action space shrinks over time. In contrast, the number of chosen AUs increases over time, but their embedding dimension is fixed, due to the use of RNN embedding. Our RL framework deals with these dynamics by rebuilding the action space at every time-step. This can be efficiently implemented by convolution over available actions (i.e., the candidate AU pool).

Objective   The objective of RL is to find the policy π that produces a trajectory τ of states and actions leading to the maximum expected reward. In our context the trajectory is the sequence of extracted AUs. The policy is realized by a neural network parametrized by θ, i.e., π_θ, where θ is the set of parameters for all modules mentioned in Sec. 3.1. The optimization can be written as

$$\theta^\star = \underset{\theta}{\mathrm{argmax}}\ \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\sum_t r(s_t, a_t)\right], \qquad (2)$$

where $r(s_t, a_t)$ is the reward for taking action a_t in state s_t, and $\tau = [s_1, a_1, s_2, a_2, \ldots, s_T, a_T]$. We employ policy gradient for optimization using the following gradient:

$$\frac{1}{N}\sum_{i,t=1}^{N,T}(\nabla_\theta \log(\pi_\theta(a_{i,t}|s_{i,t}))(\sum_{t'=t}^{T}\gamma^{t'-t}r(s_{i,t'}, a_{i,t'}))),$$
$$(3)$$

with a discount factor $\gamma = 0.9$, and $N = 2$. We summarize the pseudo code for RL training of the GDSA agent in Algorithm 1.

A synthetic illustration   We apply our method to a synthetic example for illustration. We introduce a simple 3×3

Algorithm 1 Training GDSA agent

1: **Input**: $\mathcal{D} = [(x_1, y_1), (x_2, y_2), \dots]$
2: **Initialise model parameters**: θ
3: **for** *epoch_index* **in** $[1, 2, \dots, \text{num\_epochs}]$ **do**
4:     Gradients: $\mathcal{G} = [\ ]$
5:     Sample a random input x_* with its label y_*
6:     Split x_* into AUs $[x_*^{(1)}, x_*^{(2)}, \dots]$
7:     Get AU rep.: $f = [[g(x_*^{(1)}), \tau_1], [g(x_*^{(2)}), \tau_2], \dots]$
8:     Get the category embedding: $\psi_\theta(y_*)$
9:     **for** *play_index* **in** $[1, 2, \dots, N]$ **do**
10:         Candidate-AU: $L_a = f$
11:         Chosen-AU: $L_b = [\ ]$
12:         Gradient-buff: $G = [\ ]$
13:         Reward-buff: $R = [\ ]$
14:         **for** *pass_index* **in** $[1,2,\dots,T]$ **do**
15:             Get chosen AU embed. using RNN: $\omega_\theta(L_b)$
16:             Concatenate feats: $[[\omega, L_a^{(1)}, \psi], \dots]$
17:             $\pi = \text{softmax}(\phi_\theta([\omega, L_a^{(1)}, \psi]), \dots])$
18:             Draw an AU a from discrete distribution π
19:             Move the a-th AU from L_a to L_b
20:             Calculate gradient $\nabla_\theta \log \pi[a]$ & add it to G
21:             Calculate reward using Eq. 1 & add it to R
22:         **end for**
23:         Calculate reweighed gradients using Eq. 3
24:         Add the *sum* of reweighed gradients to $\mathcal{G}$
25:     **end for**
26:     Do gradient ascent using the *average* of $\mathcal{G}$
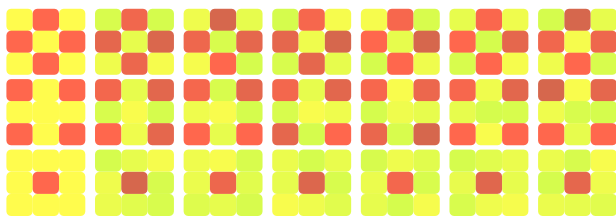27: **end for**
28: **Output**: θ

Figure 3. Samples in a synthetic dataset. The first column is the class prototype, and the remainders are observed samples.
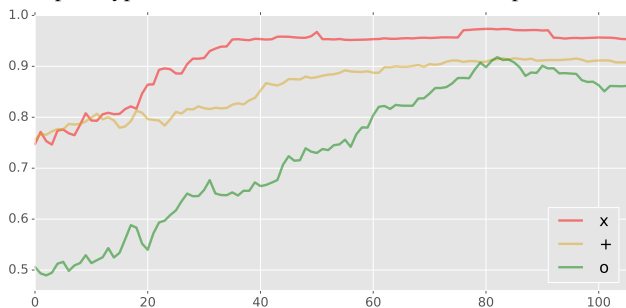

Figure 4. Synthetic data GDSA agent. X-axis: train iterations. Y-axis: test accuracy (recognition of 2-pixel image sequences).

image format, generated as a sequence of 9 binary samples

in raster scan sequential order. Each AU is one pixel, and there are 2^9 unique image categories. We choose 3 classes, corresponding to the first column in Fig. 3, denoted as '×', '+', and 'o' respectively. To introduce intra-class variability, observed samples are perturbed by Gaussian noise. A key observation in Fig. 3 is that, to recognize a category, not all AUs (pixels) are necessary. For example, in a sequence of only two AUs, if one is the corner and other is the centre, then it must be the '×' category. This creates room for simplification and re-ordering of the AU sequence to produce a shorter but information-preserving sequence.

Training the RL agent for this problem, we expect it to pick few AUs that maximize recognizability. We limit the AU selection budget to 2 (i.e., two pixel output images). As shown in Fig. 4, the agent produces output sequences with $\sim 90\%$ probability of being correctly classified by a linear classifier. This is is significantly better than its randomly initialized state (a policy that picks two strokes at random), for which the performance is about $50\% \sim 70\%$.

4. Experiments

Generic implementation details   Our model is implemented in Tensorflow [1]. The RNN used in the GDSA framework to process the chosen AU sequence is implemented with single layer gated recurrent units (GRU) with 128 hidden cells. The GRU output of dimension 1×128 is fed to a fully connected layer to get the chosen-strokes embedding of dimension 1×18. The candidate AU embedding, obtained by feeding the AU representation (fixed length feature vector concatenated with time-stamp) into a fully connected layer, is of dimension 1×9. The class embedding is of dimension 1×3. The complete embedding, obtained by concatenating the three previous embeddings and feeding into a fully connected layer, is sized 1×15. Both the code and trained models will be made public.

Setting discussion   As mentioned earlier, we proposed a new problem setting and associated solution for abstraction learning. While contemporary learning for summarization requires annotated target summaries [4, 6, 29, 35, 49], we require instead a goal function. The goal function is itself learned from metadata that is often already available or easier to obtain than expensive gold-standard summaries (e.g., sentiment label for text). Since the goal (task-specific vs. generic summaries) and data requirements (weak. vs strongly annotated) of our method are totally different, we cannot compare to conventional summarization methods.

4.1. Sketch abstraction

Dataset   We train our GDSA for sketch using Quick-Draw [17], the largest free-hand sketch dataset. As in [27], we choose 9 QuickDraw categories namely: {cat, chair, face, fire-truck, mosquito, owl, pig, purse and shoe}; using 70,000 sketches per category for training and 5,000 for
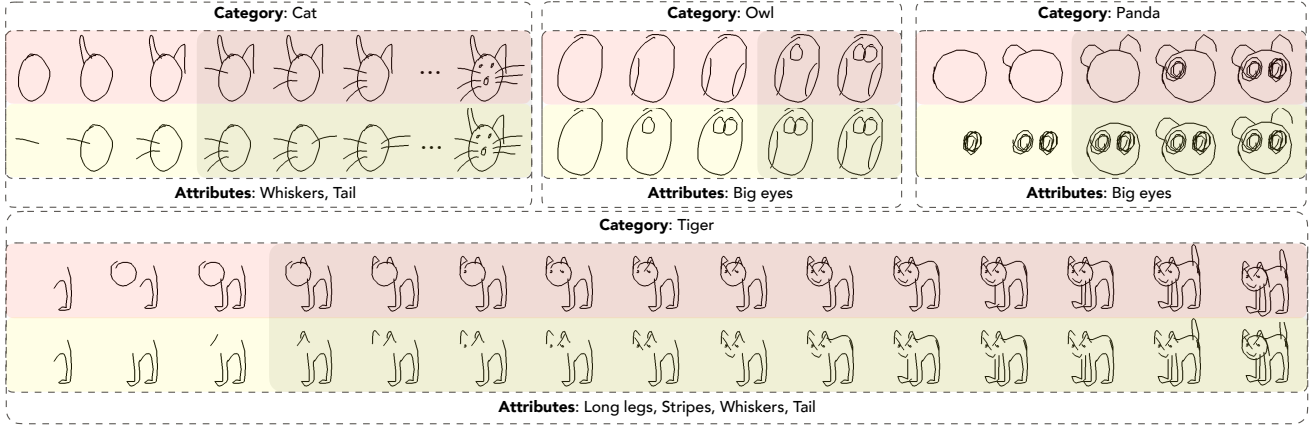
Figure 5. Qualitative comparison of goal-driven stroke sequencing strategies for QuickDraw sketches using the budget size of 25% (light background). In each section: the top row depicts the stroke sequence obtained using GDSA with category-goal, and the bottom row depicts the stroke sequence obtained using GDSA with attribute-goal.

testing. The average number of strokes in the 9 chosen categories are {9.8, 4.9, 6.4, 8.3, 7.2, 9.1, 9.5, 3.6, 3.0}.

Implementation details We train our agent with $K = 50,000$ episodes, reward scaling factor $b = 100$ and learning rate $\eta = 0.0001$. We set the budget B to 25% and 50% of the (rounded) average number of strokes per category. For reward computation, at each step the list of strokes chosen by our agent is fed into a classifier to determine the probability of the ground truth class. The same is done for the strokes chosen following original input and random order. To this end, we employ two different classifiers: (1) A state-of-the-art convolutional neural network (CNN) - Sketch-a-Net 2.0 [47], fine-tuned on the 9 QuickDraw categories. The stroke data is rendered as an image before CNN classification. (2) A three-layer LSTM with 256 hidden cells at each layer, employed in [27]. It takes an input list of (x, y, p) (coordinates and pen-state) and feeds its last time-step output to a fully-connected layer with softmax activation which provides the probability distribution for the prediction of the sketch class. After training, this RNN is also used to extract the 256 dimensional feature vector for each stroke in the candidate stroke pool which is concatenated with one-hot time-stamp vector to get the final AU representation of dimension $D = 266$. Note that we do not use Sketch-a-Net for this purpose, due to the sparsity of rendered single strokes images, with which the CNN cannot generate a meaningful representation.

Results We evaluate the performance of our GDSA model by sketch recognition accuracy when using a budget B of 25% and 50% of the average number of strokes for each category. This evaluation is performed on the testing set of 45,000 sketches. Sketch recognition is achieved using two different classifiers (RNN [27] and Sketch-a-Net [47]) described previously. We compare our abstraction model with: (1) First B strokes in the original human drawing order. This is a strong baseline, as the data in QuickDraw is

Budget	25%		50%	
Method	RNN	Sketch-a-Net	RNN	Sketch-a-Net
Human	36.66	62.08	66.73	75.90
Random	22.67	41.06	45.65	65.47
DSA [27]	38.36	65.05	67.89	81.50
DQSN [53]	38.11	64.58	67.50	80.31
GDSA	**50.50**	**71.92**	**71.75**	**86.15**
Upper bound	87.77	91.99	87.77	91.99

Table 1. Category recognition (acc. %) of the abstracted sketches.

obtained by challenging the *player* to draw the object (abstractly) in a time-limited setting - the first few strokes are thus those deemed important for recognizablity by humans. (2) Random selection of B strokes. (3) DSA [27], the state-of-the-art deep sketch abstraction model. Note that to make a fair comparison we adapt [27] to perform abstraction at stroke level, as the original paper dealt with stroke-segments (five consecutive (x, y, p) elements). (4) DQSN [53], an abstraction model originally proposed for videos. We adapt this model to our setting by plugging in stroke AU representations instead of video-frame features. We also report the performance of the full input sequence without abstraction, which represents the upper bound. The results in Table 1 show that our GDSA agent outperforms all the other methods. The improvement in performance is most evident for the harder $B = 25\%$ budget, confirming the ability of our GDSA model to learn an efficient selection policy. In particular, both DSA and DQSN are restricted by the original input AU order with a fixed 2-state action space, resulting in sub-optimal selection.

Abstraction with a different goal A key feature of our approach is the ability to select different input properties that should be preserved during the abstraction. In this experiment, we demonstrate this capability by contrasting attribute preservation with the category-preservation. We do this by selecting 9 animal categories from QuickDraw (cat, mouse, owl, panda, pig, rabbit, squirrel, tiger and zebra) and defining 5 animal attributes: whiskers (cat, mouse,

Figure 6. Qualitative comparison of goal-driven stroke sequencing strategies for TVSum videos using the budget size of 25%. Grey: Full video clip. Pink: GDSA with category preservation goal. Yellow: GDSA with attribute preservation goal.

rabbit, tiger), tail (cat, mouse, pig, rabbit, squirrel, tiger, zebra), stripes (tiger, zebra), long-legs (tiger, zebra), big-eyes (owl, panda). We train two separate Sketch-a-Net 2.0 models to recognize the above mentioned categories and attributes. These are then plugged in the reward generator to train GDSA, with budget $B = 25\%$. Qualitative comparison of the results of category vs attribute preservation are shown in Fig. 5. We can see clearly that changing the goal has a direct impact on the abstraction strategy. E.g., preserving the salient cat category cue (ears) vs. the requested attribute (whiskers).

4.2. Video abstraction

Dataset We train GDSA for video using the TVSum dataset [35], with the primary objective of preserving video category information. This dataset contains 10 categories: {changing vehicle tire, getting vehicle unstuck, groom animal, making sandwich, parkour, parade, flash mob gathering, bee keeping, bike tricks, and dog show}. We use 40 video samples for training, and 10 for testing. The video length vary from 2 to 10 minutes. Following the common practice [49, 53] we down-sample videos to 1 fps, and then use shot-change data to merge 5 consecutive shots to form coarse segments. After this, the average number of segments per video in each category are {13.5, 8.9, 10.4, 9.3, 7.8, 11.0, 10.4, 10.5, 12.1, 9.9} respectively.

Implementation details We train our agent with $K = 100$ episodes, reward scaling factor $b = 100$ and learning rate $\eta = 0.0001$. We set the budget B to 25% and 50% of the average number of segments for each category. Additionally, we test a budget of one segment to find the single most relevant segment in each video. For reward computation we use a multi-class bidrectional GRU classifier, originally proposed in [53]. This classifier, after training, is also used to extract the fixed dimension (512) feature vector which is concatenated with the time-stamp vector to obtain the final AU representation ($D = 522$) for each segment in the candidate pool.

Results The performance of GDSA is evaluated by category recognition accuracy, at three budget values of 1 segment, 25% and 50% of the average number of segments per category. Following [53] this evaluation is performed by

Budget	1 segment	25%	50%
Method	RNN	RNN	RNN
Original	28.0	28.0	28.0
Random	28.0	32.0	34.0
DSA [27]	42.0	62.0	72.0
DQSN [53]	44.0	64.0	72.0
GDSA model	**68.0**	**74.0**	**76.0**
Upper bound	78.0	78.0	78.0

Table 2. Category recognition (accuracy %) of video samples.

doing 5-fold cross-validation. Category recognition is performed using the aforementioned classifier. We compare with: (1) First B segments in the original order. (2) Random B segments. (3) DSA [27], adapted to videos by substituting stroke AU vector with video-segment AU. (4) The state-of-the-art DQSN [53], which is adapted to be trained with a category-recognition based reward for fair comparison. We also compute the upper bound of the input video without abstraction. The results in Table 2 show that our GDSA agent outperforms all competitors by significant margins.

Abstraction with a different goal In order to demonstrate the goal-driven abstraction capability of our model, we first define 5 category level attributes: animals (dog show, grooming animals, bee keeping), humans (parkour, flash mob gathering, parade), vehicles (changing vehicle tire, getting vehicle unstuck), food (making sandwich), bicycle (attempting bike tricks)}. Using the same classifier architecture used for category classification, we train the attribute classifier. This is then plugged in the reward function to guide training, with $B = 25\%$. Some qualitative results are shown in Fig. 6. We can clearly observe that abstracted output varies according to the goal function. E.g., preserving parade related segments (category) vs. segments depicting humans (attribute).

4.3. Text abstraction

Dataset We train our GDSA model for text using the Amazon Review dataset [24]. We aim to preserve positive/negative review sentiment (1-2 stars as negative, 4-5 stars as positive). We choose 9 categories: {apparel, books, dvd, electronics, kitchen and houseware, music, sports and outdoors, toys and games, and video}, based on the availability of equal number of positive and negative reviews. The average number of sentences per category are {3.5, 8.2,

Category - toys&games. **Sentiment -** negative. **Budget -** 2 sentences

I love that the inside is set up like "shelves". My son loves to organize his trains in it. I don't like how thin the metal is. It dents as easy as a cola can. If it were made better, it would be worth the money. Save your money on this one and find something better made.

If it were made better, it would be worth the money. Save your money on this one and find something better made.

My son loves to organize his trains in it. It dents as easy as a cola can.

I love that the inside is set up like 'shelves'. It dents as easy as cola can.

Category - toys&games. **Sentiment -** negative. **Budget -** 2 sentences

We went to check it out after my 4 year old saw it on tv and liked it because the bear had glasses just like him. The product seemed decent in the store, so we bought it. He has only played with it once or twice since christmas. The only time he had fun with it was when I got mad and yelled at the thing because it wouldn't do what it is supposed to. Save your money and don't buy this junk.

The only time he had fun with it was when I got mad and yelled at the thing because it wouldn't do what it is supposed to. Save your money and don't buy this junk.

We went to check it out after my 4 year old saw it on tv and liked it because the bear had glasses just like him. The product seemed decent in the store, so we bought it.

The only time he had fun with it was when I got mad and yelled at the thing because it wouldn't do what it is supposed to. Save your money and don't buy this junk.

Category - dvd. **Sentiment -** negative. **Budget -** 3 sentences

Once again, in a 90's hollywood horror flick, it is dissappointing. The actors within the movie are actually pointless and show no effect,but, there not to blame. The blame is the poor script writing for this movie. The movie has you going until about half way through, when they introduce the ghost to you. Now, unlike any other ghost in hollywood history, this ghost is literally a black blob. It contains some invisible bodies and haunts people with its face, which in turn, reveals the face of whom it kills. But, literally, when you put a 100 foot ghost into a movie, you better make sure its in the right one. This movie should probably been called the blob, not house on haunted hill. It is a horrible re-make and a waste of about 2 hours. Please, for your own sake, rent some good genre horror, like halloween or when a stranger calls. Not some crappy remake that wants to make you leave the room your watching it in. Better luck next time to the director, producer, co-producer, and writer of this film, because of all the people that took part, probably just wanted to make some cheap money, so they made a movie. Better luck next time.

The actors within the movie are actually pointless and show no effect,but, there not to blame. The blame is the poor script writing for this movie. It is a horrible re-make and a waste of about 2 hours.

This movie should probably been called the blob, not house on haunted hill. Once again, in a 90's hollywood horror flick, it is dissappointing. It is a horrible re-make and a waste of about 2 hours.

The blame is the poor script writing for this movie. It is a horrible re-make and a waste of about 2 hours. Please, for your own sake, rent some good genre horror, like halloween or when a stranger calls.

Category - music. **Sentiment -** positive. **Budget -** 2 sentences

I also had an expensive japanese cd but all this would not prepare me for what i discover here. Over these cds one discovers the original sessions from which the record was made. This is simply wonderful. Ok sometimes it takes quite a bit of patience to sit trhough all this but it is worth the time.

This is simply wonderful. Ok sometimes it takes quite a bit of patience to sit trhough all this but it is worth the time.

Over these cds one discovers the original sessions from which the record was made. This is simply wonderful.

Over these cds one discovers the original sessions from which the record was made. This is simply wonderful.

Category - apparel. **Sentiment -** positive. **Budget -** 1 sentence

It has a nice feel to it. The bathrobe is made of japanese yarn and the yarn is made of cotton so the bathrobe is made of 100% cotton. Cotton of course strinks in the wash but the strinkage is tolerable. I purchased the expresso color and after a few washes the color faded but that is to expected. Overall i am satisfied with this purchase and would purchase it over again if i had the choice to do so.

Overall i am satisfied with this purchase and would purchase it over again if i had the choice to do so.

The bathrobe is made of japanese yarn and the yarn is made of cotton so the bathrobe is made of 100% cotton.

Overall i am satisfied with this purchase and would purchase it over again if i had the choice to do so.

Figure 7. Qualitative comparison of goal-driven summarization for Amazon product reviews with budget 25%. Grey: Full review. Pink: GDSA for sentiment. Yellow: GDSA for category. Green: GDSA for helpfulness.

8.9, 5.6, 4.8, 6.8, 5.4, 4.9, 7.7} respectively. We use 1400 reviews per category for training and 600 for testing.

Implementation details We train our agent with $K = 10000$ episodes, reward scaling factor $b = 100$ and learning rate $\eta = 0.0001$. We set the budget B to 25% and 50% of the average number of sentences for each category. Additionally, we have a budget of one sentence to find the single most relevant sentence in each review. We use two different sentiment classifiers, both using Glove embedding [32] to represent each word as fixed dimension vector: (1) A state-of-the-art hierarchical attention network (HAN) [45] for text classification, trained for binary sentiment analysis on the 9 review categories. (2) A RNN built with a single layer LSTM of 64 hidden cells. It takes as input the list of word embeddings and feeds its last time-step output to a fully-connected layer with softmax activation to predict the sentiment. These classifiers, once trained, are also used to extract a fixed dimension (256/64) feature which is concatenated with the time-stamp vector to get the final AU representation ($D = 266/74$) for each sentence in the candidate sentence pool, for the respective GDSA models.

Results We evaluate the performance of our GDSA model by sentiment recognition accuracy with three budgets of 1 sentence, 25% and 50% of the average number of sentences per category. This evaluation is performed on the testing set of 5,400 reviews. Sentiment recognition uses the two classifiers (RNN and HAN [45]) described above. We compare with: (1) First B sentences in the original order. (2) Random B sentences. (3) DSA [27] and (4) DQSN [53], both adapted to text by plugging in the sentence AU representation instead of stroke and frame AU representation. Upper bound represents the performance of the full review without abstraction. The results in Table 3 show that

Budget	1 sentence		25%		50%	
Method	RNN	HAN	RNN	HAN	RNN	HAN
Original	59.70	67.47	66.57	76.06	70.73	80.27
Random	61.16	69.04	66.44	77.14	70.98	81.57
DSA [27]	66.37	72.42	71.58	80.02	73.36	83.47
DQSN [53]	65.70	71.40	71.93	80.25	73.20	83.77
GDSA model	**70.64**	**83.77**	**73.39**	**86.08**	**74.11**	**86.12**
Upper bound	76.41	86.66	76.41	86.66	76.41	86.66

Table 3. Sentiment recognition (accuracy %) of review summaries.

our GDSA agent again outperforms all competitors.

Abstraction with different goals We next demonstrate the GDSA model's goal-driven summarization capability by training instead to preserve (1) product-category (multi-class), and (2) helpfulness (binary) data. HAN classifier and $B = 25\%$ are used. Some qualitative results are shown in Fig. 7. We can observe that depending on the abstraction goal, the output varies to preserve the information relevant to the goal.

5. Conclusion

We have introduced a new problem setting and effective framework for goal-driven sequential data abstraction. It is driven by a goal-function, rather than needing expensively annotated ground-truth labels, and also uniquely allows selection of the information to be preserved rather than producing a single general-purpose summary. Our GDSA model provides improved performance in this novel abstraction task compared to several alternatives. Our reduced data requirements, and new goal-conditional abstraction ability enable different practical summarization applications compared to those common today.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org, 2015.

[2] Ramiz M Aliguliyev. A new sentence similarity measure and sentence based extractive technique for automatic text summarization. *Expert Systems with Applications*, 36:7764–7772, 2009.

[3] Elena Baralis, Luca Cagliero, Naeem Mahoto, and Alessandro Fiori. Graphsum: Discovering correlations among multiple terms for graph-based summarization. *Information Sciences*, 249:96–109, 2013.

[4] Itamar Berger, Ariel Shamir, Moshe Mahler, Elizabeth Carter, and Jessica Hodgins. Style and abstraction in portrait sketching. *TOG*, 32(4):55, 2013.

[5] John D Bransford and Jeffery J Franks. The abstraction of linguistic ideas. *Cognitive psychology*, 1971.

[6] Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. *ACL*, 2016.

[7] Ming-Ming Cheng, Jonathan Warrell, Wen-Yan Lin, Shuai Zheng, Vibhav Vineet, and Nigel Crook. Efficient salient region detection with soft image abstraction. In *ICCV*, 2013.

[8] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *TOG*, 31(4):44–1, 2012.

[9] Ehsan Elhamifar, Guillermo Sapiro, and Rene Vidal. See all by looking at a few: Sparse modeling for finding representative objects. In *CVPR*, 2012.

[10] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479, 2004.

[11] Yanwei Fu, Timothy M. Hospedales, Tao Xiang, Shaogang Gong, and Yuan Yao. Interestingness prediction by robust learning to rank. In *ECCV*, 2014.

[12] Amir Gandomi and Murtaza Haider. Beyond the hype: Big data concepts, methods, and analytics. *IJIM*, 35:137–144, 2015.

[13] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. In *NIPS*, 2014.

[14] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[15] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. Creating summaries from user videos. In *ECCV*, 2014.

[16] Michael Gygli, Helmut Grabner, and Luc Van Gool. Video summarization by learning submodular mixtures of objectives. In *CVPR*, 2015.

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

[18] Mohamad Faizal Ab Jabal, Mohd Shafry Mohd Rahim, Nur Zuraifah Syazrah Othman, and Zahabidin Jupri. A comparative study on extraction and recognition method of CAD data from CAD drawings. In *ICIME*, 2009.

[19] Qi Jia, Meiyu Yu, Xin Fan, and Haojie Li. Sequential dual deep learning with shape and texture features for sketch recognition. *CoRR*, 2017.

[20] Henry Kang, Seungyong Lee, and Charles K Chui. Flow-based image abstraction. *TVCG*, 15(1):62–76, 2009.

[21] Yi Li, Timothy M Hospedales, Yi-Zhe Song, and Shaogang Gong. Free-hand sketch recognition by multi-kernel feature learning. *CVIU*, 137:1–11, 2015.

[22] Tong Lu, Chiew-Lan Tai, Feng Su, and Shijie Cai. A new recognition model for electronic architectural drawings. *CAD*, 37:1053–1069, 2005.

[23] Inderjeet Mani. *Advances in automatic text summarization*. MIT press, 1999.

[24] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, 2015.

[25] Ryan McDonald. A study of global inference algorithms in multi-document summarization. In *ECIR*, 2007.

[26] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *EMNLP*, 2004.

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

[28] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, 2017.

[29] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Ranking sentences for extractive summarization with reinforcement learning. *NAACL*, 2018.

[30] Shashi Narayan, Nikos Papasarantopoulos, Mirella Lapata, and Shay B. Cohen. Neural extractive summarization with side information. *CoRR*, abs/1704.04530, 2017.

[31] Mayu Otani, Yuta Nakashima, Esa Rahtu, Janne Heikkilä, and Naokazu Yokoya. Video summarization using deep semantic features. In *ACCV*, 2016.

[32] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[33] Ravi Kiran Sarvadevabhatla, Shiv Surya, Trisha Mittal, and R. Venkatesh Babu. Game of sketches: Deep recurrent models of pictionary-style word guessing. *CoRR*, 2018.

[34] Rosália G Schneider and Tinne Tuytelaars. Sketch classification and classification-driven analysis using fisher vectors. *TOG*, 33:174, 2014.

[35] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. Tvsum: Summarizing web videos using titles. In *CVPR*, 2015.

[36] Pedro Sousa and Manuel J Fonseca. Geometric matching for clip-art drawing retrieval. *VCIR*, 20:71–83, 2009.

[37] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[38] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

[39] Ba Tu Truong and Svetha Venkatesh. Video abstraction: A systematic review and classification. *TOMM*, 2007.

[40] Jingya Wang, Xiatian Zhu, and Shaogang Gong. Video semantic clustering with sparse and incomplete tags. In *AAAI*, 2016.

[41] Jingya Wang, Xiatian Zhu, and Shaogang Gong. Discovering visual concept structure with sparse and incomplete tags. *Artificial Intelligence*, 250:16–36, 2017.

[42] Huawei Wei, Bingbing Ni, Yichao Yan, Huanyu Yu, Xiaokang Yang, and Chen Yao. Video summarization via semantic attended networks. In *AAAI*, 2018.

[43] Huan Yang, Baoyuan Wang, Stephen Lin, David Wipf, Minyi Guo, and Baining Guo. Unsupervised extraction of video highlights via robust recurrent auto-encoders. In *ICCV*, 2015.

[44] Yongxin Yang and Timothy Hospedales. A unified perspective on multi-domain and multi-task learning. In *ICLR*, 2015.

[45] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *NAACL HLT*, 2016.

[46] Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. Graph-based neural multi-document summarization. *CoNLL*, 2017.

[47] Qian Yu, Yongxin Yang, Feng Liu, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Sketch-a-net: A deep neural network that beats humans. *IJCV*, 122(3):411–425, 2017.

[48] Qian Yu, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales. Sketch-a-net that beats humans. In *BMVC*, 2015.

[49] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. Video summarization with long short-term memory. In *ECCV*, 2016.

[50] Yujia Zhang, Michael Kampffmeyer, Xiaodan Liang, Dingwen Zhang, Min Tan, and Eric P. Xing. Dtr-gan: Dilated temporal relational adversarial network for video summarization. *CoRR*, abs/1804.11228, 2018.

[51] Bin Zhao, Xuelong Li, and Xiaoqiang Lu. Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization. In *CVPR*, 2018.

[52] Bin Zhao and Eric P Xing. Quasi real-time summarization for consumer videos. In *CVPR*, 2014.

[53] Kaiyang Zhou, Tao Xiang, and Andrea Cavallaro. Video summarisation by classification with deep reinforcement learning. In *BMVC*, 2018.

[54] Xiatian Zhu, Chen Change Loy, and Shaogang Gong. Video synopsis by heterogeneous multi-source correlation. In *ICCV*, 2013.

[55] Xiatian Zhu, Chen Change Loy, and Shaogang Gong. Learning from multiple sources for video summarisation. *IJCV*, 117(3):247–268, 2016.