End-to-end policy learning for active visual categorization

Dinesh Jayaraman and Kristen Grauman

Abstract—Visual recognition systems mounted on autonomous moving agents face the challenge of unconstrained data, but simultaneously have the opportunity to improve their performance by moving to acquire new views at test time. In this work, we first show how a recurrent neural network-based system may be trained to perform end-to-end learning of motion policies suited for this "active recognition" setting. Further, we hypothesize that active vision requires an agent to have the capacity to reason about the effects of its motions on its view of the world. To verify this hypothesis, we attempt to induce this capacity in our active recognition pipeline, by simultaneously learning to forecast the effects of the agent's motions on its internal representation of the environment conditional on all past views. Results across three challenging datasets confirm both that our end-to-end system successfully learns meaningful policies for active category recognition, and that "learning to look ahead" further boosts recognition performance.

1 Introduction

People consistently direct their senses in order to better understand their surroundings. For example, one might swivel around in an armchair to observe a person behind him, rotate a coffee mug on his desk to read an inscription, or walk to a window to observe the rain outside.

In sharp contrast to such scenarios, recent recognition research has been focused almost exclusively on static image recognition: the system takes a single snapshot as input, and produces a category label estimate as output. The ease of collecting large labeled datasets of images has enabled major advances on this task in recent years, as evident for example in the striking gains made on the ImageNet challenge [46]. Yet, despite this recent progress, recognition performance remains low for more complex, unconstrained images [37].

To illustrate the problem, Figure 1 shows some examples of Web images and images captured by a human head-mounted camera that was not explicitly controlled to capture well-framed images. Recognition systems mounted on autonomous moving agents acquire unconstrained visual input which may be difficult to recognize effectively, *one frame at a time*. However, similar to the human actor in the opening examples above, such systems have the opportunity to improve their performance by moving their camera apparatus or manipulating objects to acquire new information, as shown in Figure 2. This control of the system over its sensory input has tremendous potential to improve its recognition performance. While such mobile agent settings (such as mobile robots and autonomous vehicles) are closer to reality today than ever before, the problem of *learning to actively move* to direct the acquisition of data remains underexplored in modern visual recognition research.

The problem we are describing fits into the realm of *active vision*, which has a rich history in the literature (*e.g.*, [3], [5], [16], [20], [47], [57]). Active vision offers several technical challenges that are unaddressed in today's standard passive scenario. In order to perform active vision, a system must learn to intelligently direct the acquisition of input to be processed by its recognition pipeline. In addition, recognition in an active setting places different demands on a system than in the standard passive scenario. To take one example, "nuisance factors" in still image recognition—such as pose, lighting, and viewpoint changes—become *avoidable* factors in the active vision setting, since in principle, they can often be overcome merely by moving the agent to the right location.

This calls for a major change of approach. Rather than strive for invariance to nuisance factors as is the standard in static image recognition, an intriguing strategy is to learn to *identify when conditions are non-ideal for recognition* and to *actively select the correct agent motion* that will lead to better conditions. In addition, recognition decisions must be

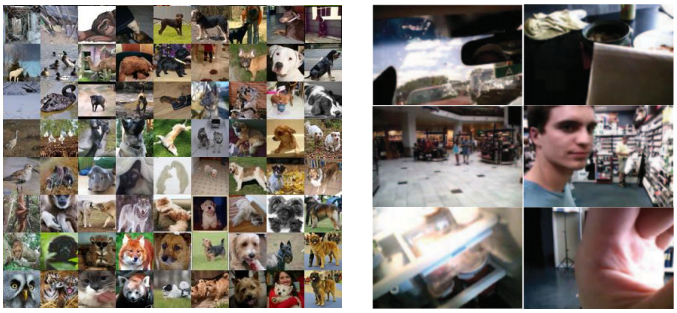

Fig. 1: Examples of Web images from ImageNet [46] (left) and randomly chosen frames from a human head-worn camera [35] (right). Cameras mounted on autonomous agents often acquire ill-framed images that can be very hard to recognize one frame at a time, compared to Web images which are human-captured and usually capture important content prominently in the foreground. However, autonomous moving visual agents can direct their cameras to acquire multiple views. Our active recognition approach employs reinforcement learning to learn policies to intelligently acquire views to facilitate scene and object category recognition.

• *D. Jayaraman was a PhD student at UT Austin at the time of this work, and is now at UC Berkeley. K. Grauman is with UT Austin.*
E-mail: see http://www.people.eecs.berkeley.edu/~dineshjayaraman/ and http://www.cs.utexas.edu/~grauman/

Manuscript received xxxxx; revised xxxxx.

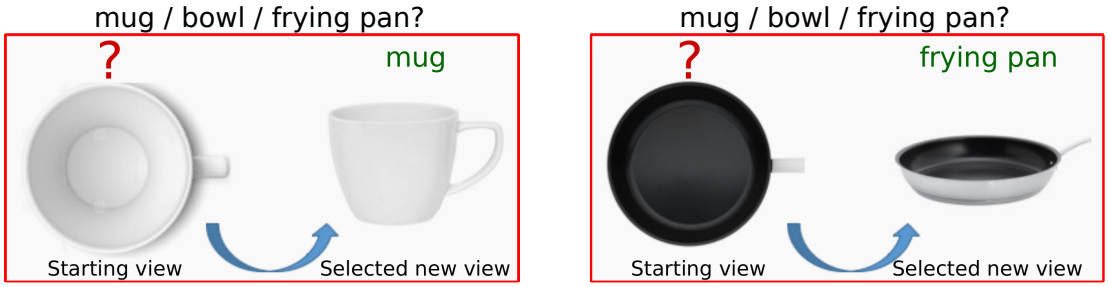

Fig. 2: A schematic illustrating the active categorization of two objects. A moving vision system may not recognize objects after just one view, but may intelligently choose to acquire new views to disambiguate amongst its competing hypotheses.

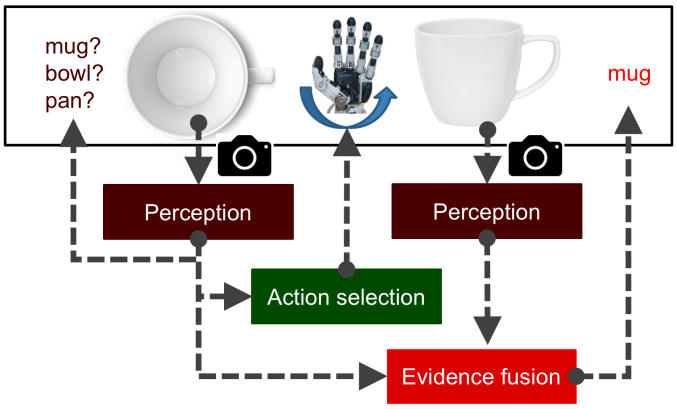

Fig. 3: A generic active recognition pipeline illustrating the three functions of an active vision system—control, per-view perception, and evidence fusion. We aim to learn all three functions jointly and end-to-end.

made based on intelligently fusing evidence from multiple observations.

Figure 3 illustrates a generic active recognition pipeline, which involves three modules—control, per-view perception, and evidence fusion. We contend that these these modules have closely intertwined functions, and must be tailored to work together. In particular, as the first contribution of this paper, we propose to learn all three modules of an active vision system simultaneously and end-to-end. We employ a stochastic neural network to learn intelligent motion policies (control), a standard neural network to process inputs at each timestep (per-view perception), and a modern recurrent neural network (RNN) to integrate evidence over time (evidence fusion). Given an initial view and a set of possible agent motions, our approach uses reinforcement learning to learn how to move in the 3D environment to produce accurate categorization results.

Additionally, we hypothesize that motion planning for active vision requires an agent to internally "look before it leaps". That is, it ought to simultaneously reason about the effect of its motions on future inputs. To demonstrate this, as a second contribution, we jointly train our active vision system to have the ability to predict *how its internal representation of its environment will evolve* conditioned on its choice of motion. As we will explain below, this may be seen as preferring equivariance, *i.e.*, predictable feature responses to pose changes, rather than invariance as is standard in passive recognition pipelines.

Through experiments on three datasets, we validate both our key ideas: (1) RNN-based end-to-end active categorization and (2) learning to forecast the effects of self-motion at the same time the agent learns how to move to solve the recognition task. We study both a scene categorization scenario, where the system chooses how to move around a previously unseen 3D scene, and an object categorization scenario, where the system chooses how to manipulate a previously unseen object that it holds. Our results establish the advantage of our end-to-end approach over both passive and traditional active methods.

2 Related Work

Active vision The idea that a subject's actions may play an important role in perception can be traced back almost 150 years [13] in the cognitive psychology literature [5]. "Active perception", the idea of exploiting *intelligent control strategies* (agent motion, object manipulation, camera parameter changes *etc.*) for *goal-directed data acquisition* to improve machine vision, was pioneered by [3], [7], [8], [57]. While most research in this area has targeted low-level vision problems such as segmentation, structure from motion, depth estimation, optical flow estimation [3], [8], [40], or the "semantic search" task of object localization [4], [15], [24], [25], [26], [33], [39], [50], [63], approaches targeting *active recognition* are most directly related to our work.

Most prior active recognition approaches attempt to identify during training those canonical/"special" views that minimize ambiguity among candidate labels [16], [19], [20], [47], [57]. At test time, such systems iteratively estimate the current pose, then select moves that take them to such pre-identified informative viewpoints. These approaches are typically applicable only to *instance* recognition problems, since broader categories can be too diverse in appearance and shape to fix "special viewpoints".

In contrast, our approach handles real world categories. To the best of our knowledge, very little prior work attempts this challenging task of active *category* recognition (as opposed to instance recognition) [11], [27], [32], [44], [59], [62]. The increased difficulty is due to the fact that with complex real world categories, it is much harder to anticipate new views conditioned on actions. Since new instances will be seen at test time, it is not sufficient to simply memorize the geometry of individual instances, as many active instance recognition methods effectively do.

Recent work uses information gain in view planning for active categorization [27], [59]. Both methods learn to *predict* the next views of *unseen test objects* conditioned on various candidate agent motions starting from the current view, either by estimating 3D models from 2.5D RGBD images [59] or by learning to predict feature responses to camera motions [27]. They then estimate the information gain on their category beliefs from each such motion, and finally greedily select the estimated most informative "next-best" move. While our idea for learning to predict action-conditional future views of novel instances is similarly motivated, we refrain from explicit greedy reasoning about the next move. Instead, our approach uses reinforcement learning (RL) in a stochastic recurrent neural network to learn optimal *sequential* movement policies over multiple timesteps. The closest methods to ours in this respect are [43] and [38], both of which employ Q-learning in feedforward neural networks to perform view selection, and target relatively simpler visual tasks compared to this work.

In addition to the above, an important novelty of our approach is in learning the entire active recognition pipeline end-to-end. Active recognition, and more generally, active vision approaches must broadly perform three separate functions: action selection, per-instant view processing, and belief updates based on the history of observed views. Previous approaches typically tackle these three tasks separately. For per-instant view processing, nearly all prior work trains a "passive" single view recognition module offline [11], [19], [20], [27], [32], [38], [44], [47], [59]. For action selection, some approaches attempt to navigate towards pre-selected discriminative viewpoints [19], [20], [47]. In addition, as described above, there are some approaches that greedily maximize information gain one step at a time [11], [27], [44], [59] and some others that employ reinforcement learning to learn policies [38], [43]. Finally, for evidence fusion, there are several manually defined heuristics in place. Some methods simply consider per-view predictions and wait for them to agree on consecutive time-steps [20], [47]. Other methods employ heuristics to allow the application of Bayes rule to update beliefs at each timestep [19], [38], [43], [44]. Recently, in [32], a system is proposed to produce predictions for each pair of observations in its history, then average over all those predictions weighting each pair with a confidence score. What is common to all these approaches is that they treat the three subproblems of active recognition as largely separate and independent components. In contrast, we train all three modules jointly for the overall active recognition objective.

Saliency and attention Visual saliency and attention are related to active vision [6], [9], [41], [48], [61]. While active vision systems aim to form policies to acquire *new* data, saliency and attention systems aim to block out "distractors" in *existing* data by identifying portions of input images/video to focus on, often as a faster alternative to sliding window-based methods. Attention systems thus sometimes take a "foveated" approach [14], [41]. In contrast, in our setting, the system never holds a snapshot of the entire environment at once. Rather, its input at each timestep is one portion of its complete physical 3D environment, and it must choose motions leading to more informative—possibly non-overlapping—viewpoints. Another difference between the two settings is that the focus of attention may move in arbitrary jumps (saccades) without continuity, whereas active vision agents may only move continuously.

Sequential attention systems using recurrent neural networks in particular have seen significant interest of late [41], with variants proving successful across several attention-based tasks [6], [48], [61]. We adopt the basic attention architecture of [41] as a starting point for our model, and develop it further to accommodate the active vision setting, instill look-ahead capabilities, and select camera motions surrounding a 3D object that will most facilitate categorization.

Predicting related features There is recent interest in "visual prediction" problems in various contexts [21], [22], [23], [27], [34], [45], [54], [55], [59], often using convolutional neural networks (CNNs). For example, one can train CNNs [29], [54] or recurrent neural networks (RNNs) to predict future video frames based on previously observed frames [45] in an entirely passive setting. These methods do not attempt to reason about *causes* of view transformations, such as camera motions. Methods for view synthesis, such as [21], [34], allow synthesis of simple synthetic images with specified factors of variation (such as pose and lighting). Given surrounding views, high quality unseen views are predicted in [23], effectively learning 3D geometry end-to-end. The methods of [1], [22], [27], [30], [42] model pixel space or feature space responses to a discrete set of observer motions. Different from all the above, we learn to predict the evolution of temporally aggregated features—computed from a complete history of seen views—as a function of observer motion choices. Furthermore, we integrate this idea with the closely tied active recognition problem.

Integrating sensors and actions Our work is also related to research in sensorimotor feature embeddings [12], [17], [18], [27], [36], [51], [56]. There the idea is to combine (possibly non-visual) sensor streams together with proprioception or other knowledge about the actions of the agent on which the sensors are mounted. Various methods learn features that transform in simple ways in response to an agent's actions [1], [2], [12], [18], [27] or reflect the geometry of an agent's environment [51]. Neural nets are trained to perform simple robotic tasks in [17], [36]. Perhaps conceptually most relevant to our work among these is [56]. Their method learns an image feature space to determine control actions easily from visual inputs, with applications to simulated control tasks. In contrast, we learn embeddings encoding information from complete *histories* of observations and agent actions, with the aim of exposing this information to an active visual recognition controller.

Finally, this manuscript builds upon our previous work published in ECCV 2016 [28]. Specifically, we further validate our method against additional baselines and on a new 3D object dataset (Section 4.2.2), propose a method to qualitatively analyze the effectiveness of motion policies (Section 4.2.3), and present additional details throughout.

3 Approach

First, we define the setting and data flow for active recognition (Section 3.1). Then we define our basic system architecture (Section 3.2). Finally, we describe our look-ahead module (Section 3.3).

3.1 Setting

We first describe our active vision setting at test time, using a 3D object category recognition scenario as a running example. Our results consider both object and scene category recognition tasks.

In the 3D object setting, the active recognition system can issue motor commands to move a camera within a viewing sphere around the 3D object X of interest. Each point on this viewing sphere is indexed by a corresponding 2D camera pose vector $\boldsymbol{p}$ indexing elevation and azimuth. An agent's manipulations of a 3D object in front of it can now be represented as a trajectory over the elevation and azimuth.

The system is allowed T timesteps to recognize every object instance X. At every timestep $t = 1, 2, \ldots T$:

- The system issues a motor command $\boldsymbol{m}_t$ *e.g.* "increase camera elevation by 20°, azimuth by 10°", from a set $\mathcal{M}$ of available camera motions. In our experiments, $\mathcal{M}$ is a discrete set consisting of small camera motions to points on an elevation-azimuth grid centered at the previous camera pose $\boldsymbol{p}_{t-1}$. At time $t = 1$, the "previous" camera pose p_0 is set to some random unknown vector, corresponding to the agent initializing its recognition episode at some arbitrary position with respect to the object.
- Next, the system is presented a new 2D view $\boldsymbol{x}_t = P(X, \boldsymbol{p}_t)$ of X captured from the new camera pose $\boldsymbol{p}_t = \boldsymbol{p}_{t-1} + \boldsymbol{m}_t$, where $P(.,.)$ is a projection function. This new evidence is now available to the system while selecting its next action $\boldsymbol{m}_{t+1}$.

At the final timestep $t = T$, the system must additionally predict a category label $\hat{y}$ for X, *e.g.*, the object category it believes is most probable. In our implementation, the number of timesteps T is fixed, and all valid motor commands have uniform cost. The system is evaluated only on the accuracy of its prediction $\hat{y}$. However, the framework generalizes to the case of variable-length episodes.

3.2 Active recognition system architecture

Our basic active recognition system is modeled on the recurrent architecture first proposed in [41] for visual attention. Our system is composed of four basic modules: ACTOR, SENSOR, AGGREGATOR and CLASSIFIER, with weights W_a, W_s, W_r, W_c respectively. At each step t, ACTOR issues a motor command $\boldsymbol{m}_t$, which updates the camera pose vector to $\boldsymbol{p}_t = \boldsymbol{p}_{t-1} + \boldsymbol{m}_t$. Next, a 2D image $\boldsymbol{x}_t$ captured from this pose is fed into SENSOR together with the motor command $\boldsymbol{m}_t$. SENSOR produces a view-specific feature vector $\boldsymbol{s}_t = \text{SENSOR}(\boldsymbol{x}_t, \boldsymbol{m}_t)$, which is then fed into AGGREGATOR to produce aggregate feature vector $\boldsymbol{a}_t = \text{AGGREGATOR}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_t)$. The cycle is completed when, at the next step $t + 1$, ACTOR processes the aggregate feature

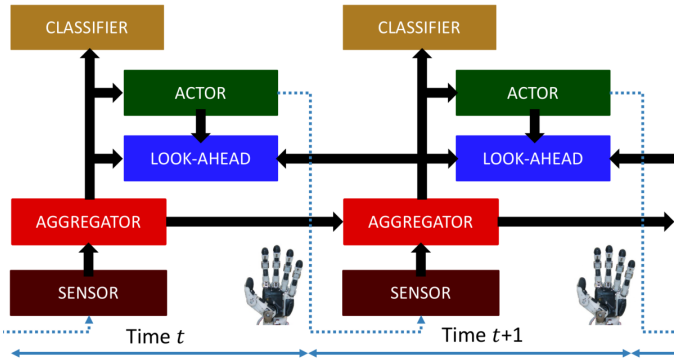

Fig. 4: A schematic of our system architecture depicting the interaction between ACTOR, SENSOR and AGGREGATOR and CLASSIFIER modules, unrolled over timesteps. This schematic depicts an unrolled version of our network architecture, where each module is repeated once for each timestep. At training time, LOOKAHEAD acts across two timesteps, learning to predict the evolution of the output of AGGREGATOR conditional on the selected motion. See Section 3.2 for details.

from the previous timestep to issue $\boldsymbol{m}_{t+1} = \text{ACTOR}(\boldsymbol{a}_t)$. Finally, after T steps, the category label beliefs are predicted as $\hat{y}(W, X) = \text{CLASSIFIER}(\boldsymbol{a}_t)$, where $W = [W_a, W_s, W_r, W_c]$ is the vector of all learnable weights in the network, and for a C-class classification problem, $\hat{y}$ is a C-dimensional multinomial probability density function representing the likelihoods of the 3D object X belonging to each of the C classes. See Figure 4 for a schematic showing how the modules are connected. Procedure block 6 lists the steps involved in the forward pass during training/inference.

In our setup, AGGREGATOR is a recurrent neural network, CLASSIFIER is a simple fully-connected hidden layer followed by a log-softmax and SENSOR separately processes the view $\boldsymbol{x}_t$ and the motor signal $\boldsymbol{m}_t$ in disjoint neural network pipelines before merging them through more layers of processing to produce the per-instance view feature $\boldsymbol{s}_t = \text{SENSOR}(\boldsymbol{x}_t, \boldsymbol{m}_t)$. ACTOR has a non-standard neural net architecture involving stochastic units: at each timestep, it internally produces an $|\mathcal{M}|$-dimensional multinomial density function $\pi(\boldsymbol{m}_t)$ over all candidate camera motions in $\mathcal{M}$, from which it samples one motion. For more details on the internal architectures of these modules, see Figure 5 (bottom).

Training

At training time, the network weights W are trained jointly to maximize classifier accuracy at time T. Following [41], training W follows a hybrid procedure involving both standard backpropagation and "connectionist reinforcement learning" [58]. The modules with standard deterministic neural network connections (CLASSIFIER, AGGREGATOR and SENSOR) can be trained directly by backpropagating gradients from a softmax classification loss, while the ACTOR module which contains stochastic units can only be trained using the REINFORCE procedure of [58].

Roughly, REINFORCE treats the ACTOR module as a Partially Observable Markov Decision Process (POMDP), with the pdf $\pi(\boldsymbol{m}_t | \boldsymbol{a}_{t-1}, W)$ representing the policy to be learned. In a reinforcement learning (RL)-style approach,

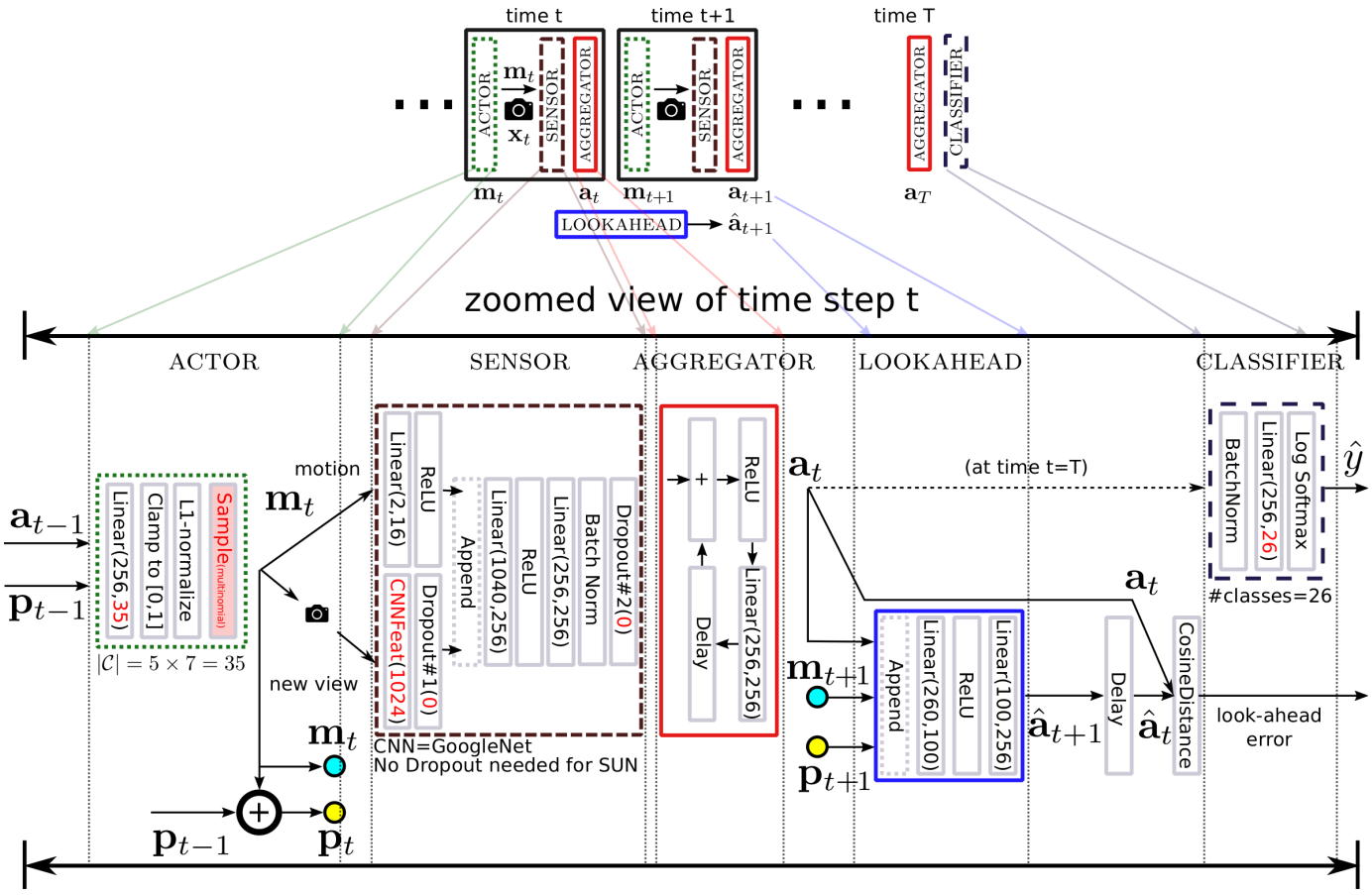

Fig. 5: (Top) A high-level schematic of our system architecture depicting the interaction between ACTOR, SENSOR, AGGREGATOR, and CLASSIFIER modules, unrolled over timesteps. Information flows from left to right. At training time, the additional LOOKAHEAD acts across two timesteps, learning to predict the evolution of the aggregate feature $\boldsymbol{a}_t$ into $\boldsymbol{a}_{t+1}$ conditional on the selected motion $\boldsymbol{m}_t$. (Bottom) A detailed schematic diagram showing the architectures of and connections amongst our active vision system modules. The small schematic at the top presents a succinct bird's-eye view of information flow within as well as between time steps, and the large schematic below zooms into the operations at some given time step t in more detail. Processing proceeds from left to right, with arrows to disambiguate where necessary. In the bottom schematic, "Linear(a,b)" denotes fully connected layers which transform a-length vector inputs to b-length vector outputs. The "Clamp" operator in ACTOR is a squashing function that sets both upper and lower limits on its inputs. The red "Sample" layer in ACTOR takes the weights of a multinomial pdf as input and samples stochastically from the distribution to produce its output (gradients cannot be backpropagated through this layer; it is trained through REINFORCE [58] instead of SGD from the classification loss). "Delay" layers store inputs internally for one time-step and output them at the next time-step. Other layer names in the schematic are self-explanatory. Input and output sizes of some layers are marked in red to denote that these are parameters derived from dataset-related choices — these are set for our SUN360 experiments in this schematic, and explanations are shown below each module. Note that AGGREGATOR is a recurrent neural network, and LOOKAHEAD may be considered a "predictive" autoencoder, that reduces its input features (appended together with the current agent motion $\boldsymbol{m}_t$) to a more compact representation in its bottleneck layer before producing its prediction of its next time-step input.

REINFORCE iteratively increases weights in the pdf $\pi(\boldsymbol{m})$ on those candidate motions $\boldsymbol{m} \in \mathcal{M}$ that have produced higher "rewards", as defined by a reward function. A simple REINFORCE reward function to promote classification accuracy could be $R_c(\hat{y}) = 1$ when the most likely label in $\hat{y}$ is correct, and 0 when not. To speed up training, we use a variance-reduced version of this loss $R(\hat{y}) = R_c(\hat{y}) - R_c(z)$, where z is set to the most commonly occuring class. Beyond the stochastic units, the REINFORCE algorithm produces gradients that may be propagated to non-stochastic units through standard backpropagation. In our hybrid training approach, these REINFORCE gradients from ACTOR are therefore added to the softmax loss gradients from CLASSIFIER before backpropagation through AGGREGATOR and SENSOR.

More formally, given a training dataset of instance-label pairs $\{(X^i, y^i) : 1 \leq i \leq N\}$, the gradient updates are as follows. Let $W_{\setminus c}$ denote $[W_a, W_s, W_r]$, *i.e.* all the weights in W except the CLASSIFIER weights W_c, and similarly, let $W_{\setminus a}$ denote $[W_c, W_r, W_s]$. Then:

$$\Delta W_{\setminus c}^{RL} \approx \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{W_{\setminus c}} \log \pi(m_t^i | \boldsymbol{a}_{t-1}^i; W_{\setminus c}) R^i, \quad (1)$$

$$\Delta W_{\setminus a}^{SM} = -\sum_{i=1}^{N} \nabla_{W_{\setminus a}} L_{\text{softmax}}(\hat{y}^i(W, X), y^i), \quad (2)$$

where indices i in the superscripts denote correspondence to the i^{th} training sample X^i. Equation (1) and (2) show

Input: 3D instance X, together with:
- projection function $P(X, \boldsymbol{p})$ denoting the view captured from camera pose $\boldsymbol{p}$
- proprioception function $f(\boldsymbol{p})$ of the current position, which is known to the active vision system, e.g., wrist position, or gravity direction.

Output: $\hat{y}$, the predicted label for instance X.

procedure FORWARDONESTEP($t, \boldsymbol{a}_{t-1}, \boldsymbol{p}_{t-1}, \hat{\boldsymbol{a}}_t$) ▷ 1 forward propagation step
  $\boldsymbol{m}_t \leftarrow$ ACTOR$(\boldsymbol{a}_{t-1}, f(\boldsymbol{p}_{t-1}))$ ▷ motor command sampled from $\mathcal{C}$ to adjust camera
  $\boldsymbol{p}_t \leftarrow \boldsymbol{p}_{t-1} + \boldsymbol{m}_t$ ▷ camera pose update
  $\boldsymbol{x}_t \leftarrow P(X, \boldsymbol{p}_t)$ ▷ capture new view
  $\boldsymbol{s}_t \leftarrow$ SENSOR$(\boldsymbol{x}_t, \boldsymbol{m}_t)$ ▷ per-view processing
  $\boldsymbol{a}_t \leftarrow$ AGGREGATOR$(\boldsymbol{a}_{t-1}, \boldsymbol{s}_t)$ ▷ evidence fusion
  if $t > 1$ **then** ▷ relevant only at training time
    $\hat{\boldsymbol{a}}_t \leftarrow$ LOOKAHEAD$(\boldsymbol{a}_{t-1}, \boldsymbol{m}_{t-1}, f(\boldsymbol{p}_t))$ ▷ look-ahead prediction of current time-step feature
    look-ahead error $\zeta_t \leftarrow d(\boldsymbol{a}_t, \hat{\boldsymbol{a}}_t)$
  end if
  return $\boldsymbol{a}_t, \boldsymbol{p}_t, \zeta_t$
end procedure

$\boldsymbol{a}_0 \leftarrow \mathbf{0}$ ▷ initialization
$\boldsymbol{p}_0 \leftarrow$ random position
for t =1,2,...T **do** ▷ move, observe, aggregate in a loop
  $\boldsymbol{a}_t, \boldsymbol{p}_t, \zeta_t \leftarrow$ FORWARDONESTEP$(t, \boldsymbol{a}_t, \boldsymbol{p}_{t-1})$
end for
$\hat{y} \leftarrow$ CLASSIFIER$(\boldsymbol{a}_t)$ ▷ final class prediction
return $\hat{y}$

Proc. 6: Pseudocode for forward propagation through our active recognition network at training/inference time

the gradients computed from the REINFORCE rewards (RL) and the softmax loss (SM) respectively, for different subsets of weights. The notation in the RHS of Equation (1) makes explicit the fact that the conditional action probability $\pi(m_t^i|\boldsymbol{a}_{t-1}^i)$ is influenced by all the parameters in the network, except classifier weights. The REINFORCE gradients ΔW^{RL} are computed using the approximation proposed in [58]. Final gradients with respect to the weights of each module used in weight updates are given by: $\Delta W_a = \Delta W_a^{RL}$, $\Delta W_s = \Delta W_s^{RL} + \Delta W_s^{SM}$, $\Delta W_r = \Delta W_r^{RL} + \Delta W_r^{SM}$, $\Delta W_c = \Delta W_c^{SM}$. Training is through standard stochastic gradient descent with early stopping based on a validation set.

3.3 Look-ahead: predicting the effects of motions

Active recognition systems select the next motion based on some expectation of the next view. Though non-trivial even in the traditional instance recognition setting [16], [20], [47], [57], with instances one can exploit the fact that pose estimation in some canonical pose space is sufficient in itself to estimate properties of future views. In other words, with enough prior experience seeing the object instance, it is largely a 3D (or implicit 3D) geometric model formation problem.

In contrast, as discussed in Section 2, this problem is much harder in active *categorization* with realistic categories—the domain we target. Predicting subsequent views in this setting is severely under-constrained, and requires reasoning about semantics and geometry together. In other words, next view planning requires some element of learning about how 3D objects *in general* change in their appearance as a function of observer motion.

Concretely, we hypothesize that the ability to predict the next view conditional on the next camera motion is closely tied to the ability to select optimal motions. Thus, rather than learn separately the model of view transitions and model of motion policies, we propose a unified approach to learn them jointly. Our idea is that knowledge transfer from a view prediction task will benefit active categorization. In this formulation, we retain the system from Section 3.2, but simultaneously learn to predict, at every timestep t, the impact on aggregate features $\boldsymbol{a}_{t+1}$ at the next timestep, given $\boldsymbol{a}_t$ and any choice of motion $\boldsymbol{m}_t \in \mathcal{M}$. In other words, we simultaneously learn how the *accumulated history* of learned features—not only the current view—will evolve as a function of our candidate motions.

For this auxiliary task, we introduce an additional module, LOOKAHEAD, with weights W_l into the setup of Section 3.2 at training time. At timestep t, LOOKAHEAD takes as input the previous timestep aggregate feature vector $\boldsymbol{a}_{t-1}$ and the last motion command issued by ACTOR $\boldsymbol{m}_t$, and produces $\hat{\boldsymbol{a}}_t$ as an estimate of the current timestep aggregate features, i.e.,$\hat{\boldsymbol{a}}_t$ = LOOKAHEAD$(\boldsymbol{a}_{t-1}, \boldsymbol{m}_t)$. This module may be thought of as a "predictive auto-encoder" in the space of aggregate features $\boldsymbol{a}_t$ output by AGGREGATOR. A look-ahead error loss is computed at every timestep between the predicted and actual aggregate features: $d(\hat{\boldsymbol{a}}_t, \boldsymbol{a}_t|\boldsymbol{a}_{t-1}, \boldsymbol{m}_t)$. We use the cosine distance to compute this error. This per-timestep look-ahead loss provides a third source of training gradients $\Delta W_{\setminus ca}^{LA}$ for the network weights, as it is backpropagated through AGGREGATOR and SENSOR:

$$\Delta W_{\setminus ca}^{LA} = \sum_{i=1}^{N} \sum_{t=2}^{T} \nabla_{W_{\setminus ca}} d(\hat{\boldsymbol{a}}_t, \boldsymbol{a}_t|\boldsymbol{a}_{t-1}, \boldsymbol{m}_t), \qquad (3)$$

where W now includes W_l and LA denotes lookahead. The LOOKAHEAD module itself is trained solely from this error, so that $\Delta W_l = \Delta W_l^{LA}$. The final gradients used to train SENSOR and AGGREGATOR change to include this new loss: $\Delta W_s = \Delta W_s^{RL} + \Delta W_s^{SM} + \lambda \Delta W_s^{LA}$, $\Delta W_r = \Delta W_r^{RL} + \Delta W_r^{SM} + \lambda \Delta W_r^{LA}$. λ is a new hyperparameter that controls how much the weights in the core network are influenced by the look-ahead error loss.

The look-ahead error loss of Equation 3 may also be interpreted as an unsupervised regularizer on the classification objective of Equation 1 and 2. This regularizer encodes the hypothesis that good features for the active recognition task must respond in learnable, systematic ways to camera motions.

This is related to the role of equivariant image features in [27], where we showed that regularizing image features to respond predictably to observer egomotions improves

performance on standard static image categorization tasks. However, this work differs from [27] in several important ways. First, we explore the utility of look-ahead for the active categorization problem, not recognition of individual static images. Second, the proposed look-ahead module is conceptually distinct. In particular, we propose to regularize the aggregate features from a sequence of activity, not simply per-view features. Whereas in [27] the effect of a discrete ego-motion on one image is estimated by linear transformations in the embedding space, the proposed look-ahead module takes as input both the history of views and the selected motion when estimating the effects of hypothetical motions.

Proprioceptive knowledge

Another useful feature of our approach is that it allows for easy modeling of proprioceptive knowledge such as the current position $\boldsymbol{p}_t$ of a robotic arm. Since the ACTOR module is trained purely through REINFORCE rewards, all other modules may access its output $\boldsymbol{m}_t$ without having to backpropagate extra gradients from the softmax loss. For instance, while the sensor module is fed $\boldsymbol{m}_t$ as input, it does not directly backpropagate any gradients to train ACTOR. Since $\boldsymbol{p}_t$ is a function solely of $(\boldsymbol{m}_1 ... \boldsymbol{m}_t)$, this knowledge is readily available for use in other components of the system without any changes to the training procedure described above. We append appropriate proprioceptive information to the inputs of ACTOR and LOOKAHEAD, detailed in experiments.

Greedy softmax classification loss

We found it beneficial at training time to inject softmax classification gradients after every timestep, rather than only at the end of T timesteps. To achieve this, the CLASSIFIER module is modified to contain a bank of T classification networks with identical architectures (but different weights, since in general, AGGREGATOR outputs $\boldsymbol{a}_t$ at different timesteps may have domain differences). Note that the REINFORCE loss is still computed only at $t = T$. Thus, given that softmax gradients do not pass through the ACTOR module, it remains free to learn non-greedy motion policies.

4 Experiments

We evaluate our approach for object and scene categorization. In both cases, the system must choose how it will move in its 3D environment such that the full sequence of its actions leads to the most accurate categorization results.

4.1 Datasets and baselines

While active vision systems have traditionally been tested on custom robotic setups [44] (or simple turn-table-style datasets [47]), we aim to test our system on realistic, off-the-shelf datasets in the interest of benchmarking and reproducibility. To this end, we work with three publicly available datasets, SUN360 [60], GERMS [38] and ModelNet10 [59].

Our **SUN360 [60]** experiments test a scenario where the agent is exploring a 3D scene and must intelligently turn to see new parts of the scene that will enable accurate scene categorization (bedroom, living room, etc.). SUN360 consists of spherical panoramas of various indoor and outdoor scenes together with scene category labels. We use the 26-category subset (8992 panoramic images) used in [60]. Each panorama by itself represents a 3D scene instance, around which an agent "moves" by rotating its head, as shown in Figure 7. For our experiments, the agent has a limited field of view (45°) at each timestep. We sample discrete views in a 12 elevations (camera pitch) $\times$ 12 azimuths (camera yaw) grid. The pitch and yaw steps are both spaced 30°apart (12×30=360), so that the entire viewing sphere is uniformly sampled on each axis. Starting from a full panorama of size 1024×2048, each 45° FOV view is represented first as a 224×224 image, from which 1024-dim. GoogleNet [53] features are extracted from the penultimate layer.[1] At each timestep, the agent can choose to move to viewpoints on a 5×7 grid centered at the current position. We set $T = 5$ timesteps.[2] Proprioceptive knowledge in the form of the current camera elevation angle is fed into ACTOR and LOOKAHEAD. We use a random 80-20 train-test split. Our use of SUN360 to simulate an active agent in a 3D scene is new and offers a realistic scenario that we can benchmark rigorously; note that previous work on the dataset does a different task, *i.e.*, recognition with the full panorama in hand at once [60], and results are therefore not comparable to our setting.

Our **GERMS [38]** experiments consider the scenario where a robot is holding an object and must decide on its next best motion relative to that object, *e.g.*, to gain access to an unseen facet of the object, so as to recognize its instance label. GERMS has 6 videos each (3 train, 3 test) of 136 objects being rotated around different fixed axes, against a television screen displaying moving indoor scenes (see Figure 8). Each video frame is annotated by the angle at which the robotic arm is holding the object. Each video provides one collection of views that our active vision system can traverse at will, for a total of $136 \times 6 = 816$ train/test instances (compared to 8992 on SUN360). While GERMS is small and targets *instance* rather than category recognition, aside from SUN360 it is the most suitable prior dataset facilitating active recognition. Each frame is represented by a 4096-dim. VGG-net feature vector [49], provided by the authors [38]. We set episode lengths to $T = 3$ steps. As proprioceptive knowledge, we feed the current position of the robotic hand into ACTOR and LOOKAHEAD. We use the train-test subsets specified by the dataset authors.

Finally, our **ModelNet10 [59]** experiments also consider an active object recognition setup, but with synthetic 3D object models. ModelNet10 contains 4899 synthetic models of 10 household furniture categories (such as "bathtub", "bed", and "chair"). We use the pre-specified train-test split (3991 training and 908 testing models). In our experimental setup, the active agent views one 2D projection of the 3D model at each timestep, and can choose to rotate the object to access neighboring 2D views, replicating the setup of [32] for direct

1. While our experiments assume pretrained features as input to the SENSOR module for speed and memory efficiency, it is also possible to treat the feature extractor as part of the SENSOR module, so that the pipeline is trained directly on pixel inputs.

2. Episode lengths were set based on learning time for efficient experimentation.

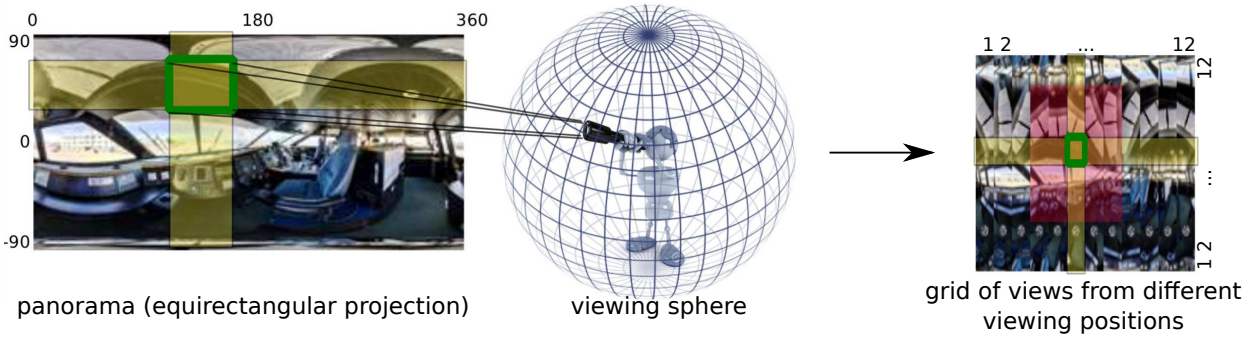

Fig. 7: (Best seen in color) An "airplane interior" class example showing how SUN360 spherical panoramas (equirectangular projection on the left) are converted into 12×12 45°FOV view grid. As an illustration, the view at grid coordinates $x = 4$, $y = 6$ outlined in green in the view grid on the right corresponds approximately to the overlap region (also outlined in green) on the left (approximate because of panorama distortions—rectangles in the panorama are not rectangles in the rectified views present in the grid). The 5×7 red shaded region in the view grid (right) shows the motions available to ACTOR when starting from the highlighted view.

Fig. 8: The GERMS active object instance recognition dataset [38] contains videos of a single-axis robotic hand rotating 136 toys against a moving background.

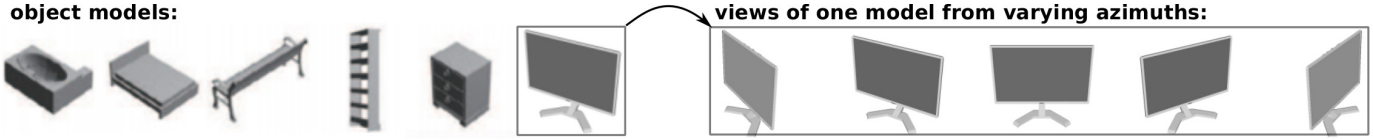

Fig. 9: Examples of synthetic 3D models from ModelNet10, used in our active object recognition experiments.

comparison. Specifically, we capture 84 views of each model (7 elevations $\times$ 12 azimuths) in a viewing grid surrounding the object. At each timestep, the agent can choose to rotate the object to access one of the eight "adjacent" views within this grid. As in SUN360, elevation is available to the agent as proprioceptive information. To represent each view, we use the publicly available ImageNet-pretrained VGG16 model [49] and finetune it on ModelNet10 single view classification, before extracting fc7 features. While this dataset is synthetic unlike SUN360 and GERMS, we use it for direct comparison against two recently proposed deep learning-based active recognition approaches [32], [59]. Some examples of 3D models from ModelNet10 are shown in Figure 9.

Of these datasets, SUN360 is selected to test our system's ability to effectively perform active recognition of complex real-world scene categories. The GERMS (real object instance recognition) and ModelNet (synthetic object category recognition) tasks reflect the current standard in active recognition evaluation methods, and thus allow comparison against prior approaches [32], [38].

Baselines

We extensively evaluate our "Look-ahead active RNN" (Section 3.3) and simpler "Active RNN" (Section 3.2) against eight baselines, including passive single-view methods, random view sampling, and traditional prior active vision approaches upgraded to be competitive in our setting, and recent work in the literature [32], [59].

- **`single view (neural net)`**: has access to only one view, like the starting view provided to the active systems. A feed-forward neural network is used for this baseline, composed from the appropriate components of the SENSOR and CLASSIFIER modules of our system. This baseline is entirely pose-agnostic, i.e., the same classifier is applied to views from all object poses.
- **`random views (average)`**: uses the same architecture as "`single view (neural net)`", but has access to T views, with successive views being related by randomly selected motions from the same motion set $\mathcal{M}$ available to the active systems. Its output class likelihood at $t = T$ is the average of its independent estimates of class likelihood for each view.
- **`random views (recurrent)`**: uses the same core architecture as our Active RNN method, except for the ACTOR module. In its place, random motions (from $\mathcal{M}$) are selected. Note that this should be a strong baseline, having nearly all aspects of the proposed approach except for the active view selection module. In particular, it has access to its selected motions in its SENSOR module, and can also learn to intelligently aggregate evidence over

views in its AGGREGATOR RNN module.

- **transinformation**: is closely based on [47], in which views are selected greedily to reduce the information-theoretic uncertainty of the category hypothesis. We make modifications for our setting, such as using 1024-D CNN features in place of the original receptive field histogram features, and using Monte Carlo sampling to approximate information gain. Each view is classified with pose-specific classifiers. When the class hypothesis is identical between consecutive views, it is emitted as output and view selection terminates. Like most prior approaches, this method relies on a canonical world coordinate space in which all object instances can be registered. Since this is infeasible in the active categorization setting, we treat each instance's coordinates as world coordinates.
- **seqDP**: is closely based on [19], and extends [47] using a sequential decision process with Bayesian aggregation of information between views. It runs to a fixed number of views.
- **transinformation + seqDP**: combines the strengths of [47] and [19]; it uses Bayesian information aggregation across views, and terminates early when the predicted class remains unchanged at consecutive timesteps.
- **Depth-ShapeNets** [59]: assumes access to depth information for all observed views, unlike our method, which only sees RGB information. It hallucinates unobserved entries in the voxel grid using 3D convolutional deep belief networks, and uses a mutual-information-based metric inspired by seqDP [19] to select next-best views. We compare against this method on ModelNet10 data using published numbers from [32].
- **RGBD-Pairwise** [32]: decomposes the sequence of observed views into pairs, classifies each pair using a CNN, and averages those pairwise classifications over the full sequence to perform information fusion. For view selection, it trains a second CNN in supervised manner to directly map the current view to the best next viewpoint. Aside from the RGB information that our method accesses, RGBD-Pairwise assumes additional access to depth information. We compare directly against this method's published results on ModelNet-10.

Hyperparameters for all methods were optimized for overall accuracy on a validation set through iterative search over random combinations [10].

4.2 Results

In Section 4.2.1, we present results for active recognition of scene categories and object instances on realistic SUN360 and GERMS data respectively. In Section 4.2.2 we replicate settings used in recent work [32], [59] for a direct comparison on ModelNet synthetic object model categorization. Finally, in Section 4.2.3, we qualitatively analyze the learned policies by studying the effect of starting position on recognition accuracy over time.

4.2.1 Active scene and object recognition

Table 1 shows the recognition accuracy results for scene categorization (SUN360) and object instance recognition (GERMS). Figure 10 and Figure 11 plot the results as a function of timesteps, comparing against ablated variants of our approach and against classic prior approaches (transinformation, seqDP and transinformation+seqDP) respectively. Both variants of our method outperform the baselines on both datasets, confirming that our active approach successfully learns intelligent view selection strategies. Passive baselines, representative of the current standard approaches to visual categorization that classify Web photos, perform uniformly poorly, highlighting the advantages of the active setting. In addition, our Look-ahead active RNN outperforms our Active RNN variant on both datasets, showing the value in simultaneously learning to predict action-conditional next views at the same time we learn the active vision policy. By "looking before leaping" our look-ahead module facilitates beneficial knowledge transfer for the active vision task.

On SUN360, even though it represents a much harder active *category recognition* problem, the margins between our method and the random view baselines are pronounced. Furthermore, while the traditional active baselines do show significant improvements from observing multiple views, they fall far short of the performance of our method despite upgrading them in order to be competitive, such as by using CNN features, as described above.

On GERMS, our method is once again easily superior to prior active methods. The margins of our gains over random-view baselines are smaller than on SUN360. Upon analysis, it becomes clear that this is due to GERMS being a relatively small dataset. Not only is (1) the number of active recognition instances small compared to SUN360 (816 vs. 8992), but (2) different views of the same object instance are naturally closer to each other than different views from a SUN360 panorama view-grid (see Figure 7 and Fig 8) so that even single view diversity is low, and (3) there is only a single degree of motion compared to two in SUN360. As a result, the number of possible reinforcement learning episodes is also much smaller.

Upon inspection, we found that these factors can lead our end-to-end network to overfit to training data (which we countered with more aggressive regularization). In particular, it is problematic if our method achieves zero training error from just single views, so that the network has no incentive to learn to aggregate information across views well. Our active results are in line with those presented as a benchmark in the paper introducing the dataset [38], and we expect more training data is necessary to move further with end-to-end learning on this challenge. This lack of data affects our prior active method baselines even more since they rely on *pose-specific* instance classifiers, so that each classifier's training set is very small. This explains their poor performance.

As an interesting upshot, we see further improvements on GERMS by averaging the CLASSIFIER modules' outputs *i.e.* class likelihoods estimated from the aggregated features at each timestep $t = 1, .., T$ ("Look-ahead active RNN + average"). Since the above factors make it difficult to learn the optimal AGGREGATOR in an end-to-end system like ours, a second tier of aggregation in the form of averaging over the outputs of our system can yield improvements. In contrast, since SUN offers much more training data, averaging over per-timestep CLASSIFIER outputs significantly *reduces* the performance of the system, compared to directly using the

Method↓/Dataset→ Performance measure→		SUN360 T=2 acc.	SUN360 T=3 acc.	SUN360 T=5 acc.	GERMS T=2 acc.	GERMS T=3 acc.
Passive approaches	Chance	14.08	14.08	14.08	0.74	0.74
	`single view (neural net)`	40.12±0.45	40.12±0.45	40.12±0.45	40.31±0.23	40.31±0.23
Random view (ablation)	`random views (average)`	45.71±0.29	50.47±0.37	54.21±0.57	45.71±0.30	46.97±0.43
	`random views (recurrent)`	47.74±0.27	51.29±0.21	55.64±0.28	44.85±0.40	44.24±0.24
Prior active approaches	`transinformation` [47]	40.69	40.69	44.86	28.83	31.02
	`seqDP` [19]	42.41	42.91	42.08	28.83	28.10
	`transinformation + seqDP`	44.69	46.91	48.19	29.93	29.56
`ours`	Active RNN	50.76±0.41	57.52±0.46	65.32±0.42	47.30±0.73	46.86±0.97
	Look-ahead active RNN	**51.72±0.29**	**58.12±0.43**	**66.01±0.34**	**48.02±0.68**	**47.99±0.79**
	Look-ahead active RNN+average	49.62±0.43	55.43±0.38	62.61±0.33	47.00±0.45	**48.31±0.72**

TABLE 1: Recognition accuracy on SUN360 and GERMS (neural net-based methods' scores are reported as mean ± standard error over 5 runs with different initializations)

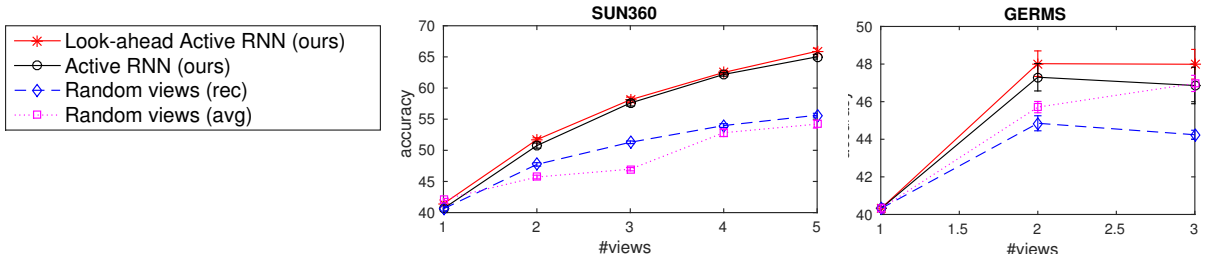

Fig. 10: Evolution of accuracy over time for various ablated variants of our method, on SUN360 (left) and GERMS (right). Our methods show steady improvement with additional views, and easily outperform the best baselines. Also see Table 1.

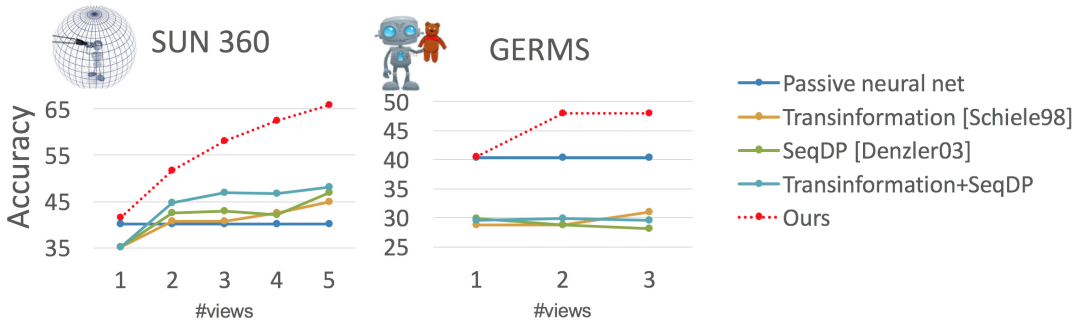

Fig. 11: Evolution of accuracy over time for our method, vs. existing active recognition methods `transinformation` [47], `seqDP` [19], and `transinformation+seqDP`. Our integrated end-to-end solution strongly outperforms these well-known and widely used classic information-theoretical approaches to active recognition implemented with state-of-the-art CNN features, identical to our method. Also see Table 1.

last timestep output. This is exactly as one would hope for a successful end-to-end training. This reasoning is further supported by the fact that "`random views (average)`" shows slightly poorer performance than "`random views (recurrent)`" on GERMS, but is much better on SUN360.

Indeed, the significant gains of "`random views (recurrent)`" over "`random views (average)`" on SUN360 points to an important advantage of treating object/scene categorization as a grounded, sequence-based decision process. The ability to intelligently fuse observations over timesteps based on both the views themselves and the camera motions relating them offers substantial rewards. In contrast, the current computer vision literature in visual categorization is largely focused on categorization strategies that process individual images outside the context of any agent motion or sequential data, much like the "Single view" or "`random views (average)`" baselines. Our empirical results show the exciting promise for future work in this space. They also suggest the need for increased efforts creating large 3D and video benchmark datasets (in the spirit of SUN360 and GERMS and beyond) to support such vision research, allowing us to systematically study these scenarios outside of robot platforms.

The result on SUN360 in particular is significant since no prior active recognition approach has been shown to successfully handle any comparably complex dataset. While active categorization is technically challenging compared to instance recognition as discussed in Section 2, datasets like SUN360 that contain complex visual data with ambiguous views may actually be most suited to showing the advantages of the active recognition paradigm.

Figure 13 and Figure 14 show some qualitative examples

Method↓/Dataset→ Performance measure→	ModelNet10 T=3 acc.	ModelNet10 T=6 acc.
Chance	11.02	11.02
`single view (neural net)`	85.65±0.09	85.65±0.09
`random views (recurrent)`	89.23±0.03	90.10±0.08
`single-action`	89.81±0.05	90.99±0.08
`single-action+`	90.25±0.06	91.77±0.04
`Depth-ShapeNets` [59]	78.7	81.0
`RGBD-Pairwise` [32]	88.8	91.6
`ours`	**91.01±0.14**	**92.50±0.07**

TABLE 2: Recognition accuracy on ModelNet, including recently published benchmarks.

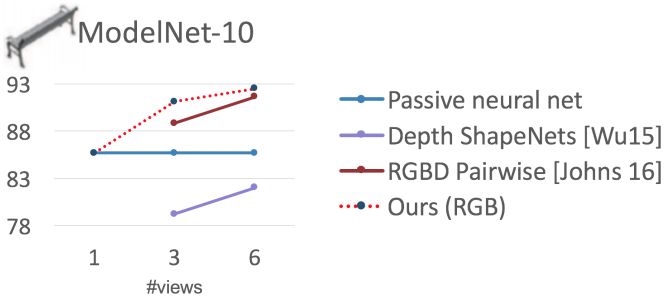

Fig. 12: Evolution of accuracy over time for our method vs. two other recently proposed deep learning-based approaches, `Depth-ShapeNets` [59] and `RGBD-Pairwise` [32], both of which use depth information. Our integrated end-to-end solution strongly outperforms both approaches with significant margins despite using only RGB information (no depth). While RGBD Pairwise is the strongest baseline, our method's performance after seeing only three views ($T = 3$) is nearly on par with the strongest baseline's performance after selecting 6 views ($T = 6$). Also see Table 2.

of the view selection behavior of our approach on SUN360 and GERMS respectively.

4.2.2 *Active object recognition with synthetic CAD models*

The above experiments establish the advantages of our method over traditional active recognition approaches in realistic settings using panoramic scenes (SUN360) and object manipulation videos (GERMS). Next, we use synthetic ModelNet10 object models in an active object recognition experiment to allow direct comparison against two recently published deep learning-based active recognition approaches: `Depth-ShapeNets` [59] and `RGBD-Pairwise` [32], both of which use the VGG-M CNN architecture, same as `ours`, to represent input images. We exactly reproduce the settings described in [32] and compare against the `Depth-ShapeNets` and `RGBD-Pairwise` results presented there. Figure 12 plots the performance of our method against these baselines. Both these methods use 2.5D information, i.e., they have additional access to depth while our method uses only 2D projections. Despite this handicap, our method significantly outperforms these baselines.

Table 2 presents the precise accuracy results and includes a comparison against other baselines: `random views (recurrent)`, `single-action`, and `single-action+`. As before, `random views (recurrent)` uses the same architecture as out method, except that it selects random motions at each timestep. We define `single-action` and `single-action+` below:

- **`single-action`**: This represents a policy that rotates the object by a fixed amount along a fixed direction at every time-step. We exhaustively test all valid rotation actions and only present results for the action that works best on the test set.
- **`single-action+`**: In our setup, since elevation is limited to $[-90°, +90°]$, a `single-action` policy that moves upwards could get stuck at the highest elevation, for instance. At this point, `single-action+` avoids this by switching directions to move downwards.

`random views (recurrent)`, `single-action`, and `single-action+` are all ablated variants of our method which replace the ACTOR module of our system with heuristic action policies while retaining the rest of our pipeline. As Table 2 shows, our learned action policy does significantly better than these heuristic policy baselines. At the same time, it is worth noting that even these ablated variants outperform the best previously published approaches [32], [59], suggesting that aside from the learned action policy, our approach's sensing and aggregation pipelines are also key to its superior performance

4.2.3 *Qualitative analysis of motion policies*

Next we analyze the motion policies learned by our method. Recall that in our setup, the first view is selected at random and presented to the agent. Some views of an object or scene are less discriminative than others, but good active recognition agents should be able to recover from poor starting views very quickly, by steering the camera or manipulating the object intelligently to reach better viewpoints.

To assess whether our approach does in fact demonstrate speedy recovery, we visualize the impact of the starting viewpoint on active recognition accuracy as a function of time on SUN360 and ModelNet10 in Figure 15. At $T = 1$, accuracy is a strong function of starting position on both SUN360 and ModelNet. On SUN360, azimuth has no discernible impact as expected since panoramas in scenes have no canonical forward direction. Interestingly, however, elevations near the horizontal (middle band in the plots of Figure 15) produce the highest accuracies, suggesting that the maximum discriminative information for scene categorization exists at approximately eye level. On ModelNet10, there is a grid-like structure to the position dependence at $T = 1$, every 90 degrees along both elevation and azimuth. Upon examination, this has an interesting explanation. ModelNet10 has several approximately cuboidal object categories like table, bed, and dresser. For such objects, at every 90°rotation, a majority of the faces of the object are occluded from view, and only one or two faces are visible, which makes recognition difficult.

There are thus strong position dependencies in both datasets for individual view discriminativeness. For both `ours` and `random views (recurrent)`, the spread of accuracies is therefore large at $T = 1$, and accuracies consistently improve over time. However, the key observation from this visualization is that for `ours`, the accuracies

Fig. 13: (Best viewed in color on pdf with zoom) Views selected using our approach on SUN360. Each row, corresponding to a scene, contains three red panels corresponding to the selected views at $t = 1, 2, 3$. Each panel shows the current view (left) and position on view grid (pink highlight is current position, yellow highlights show views reachable at next timestep). In the top row, given the first view, our method makes reasonable but wrong guesses, but corrects itself within two moves, by observing the crowd and following their gaze.

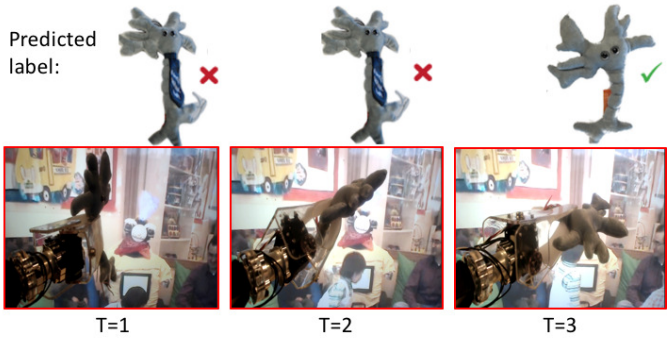

Fig. 14: Views selected using our approach for a GERMS object. The three red panels correspond to the selected views at $t = 1, 2, 3$. The predicted instance at each timestep is depicted along the top. In this case, our approach manages to manipulate from poorly lit, side-on views to a more frontal, well-lit view to correctly recognize the object instance.

converge to an approximately uniform distribution over starting position (i.e., negligible dependence on starting position) much more quickly on both datasets. Our intelligent action selection approach is better able to overcome the disadvantage of bad starting positions.

5 Conclusions

We presented a new end-to-end approach for active visual categorization. Our framework simultaneously learns (1) how an agent should move to improve its observation sequence, and (2) how its future observations are likely to change conditioned on its possible motions. We show the impact on object and scene recognition, where our active approach makes sizeable strides over single view and passively moving systems. Furthermore, we establish the advantage of treating all components of the active recognition system simultaneously. All together, the results serve as evidence that modern visual recognition algorithms can venture further into unconstrained, sequential data, moving beyond the static image snapshot labeling paradigm.

We have recently proposed an approach for learning unsupervised generic exploratory policies [31]. Trained by targeting completion of unobserved portions of a scene, these policies may then be transfered to new tasks. This points towards a method for reducing the labeled data requirement for training the active recognition policies proposed in this paper.

Our treatment of the active recognition problem leaves open several further directions for follow-up work. While our model refreshes its category beliefs after every observation, it does not have the option of terminating. Instead, we trained fixed time-budgeted models, and in our experiments, we tested accuracy as a function of time with fixed numbers of observations. An important future challenge is for the agent to decide when to observe another view and when that is unnecessary. For example, this could be accomplished by setting a confidence threshold on the agent's current beliefs at inference time. A more interesting option would be to include a termination action to learn when to terminate the observation of an object or scene. A small negative reward with each elapsed timestep could provide an agent the incentive to terminate observation early. Other situations may demand movement or energy-budgeted exploration, which could be modeled by negative rewards that are proportional to the magnitude of the selected motion.

Our experiments all assume that the object or scene to be recognized does not evolve over time. While the GERMS datset does include moving distractors, the object in the robot arm is nearly rigid. However, a natural setting might involve performing active recognition in a scene with moving objects. Since our method itself is not tied to any static world assumptions, it would be of interest in future work to study its performance in more dynamic environments.

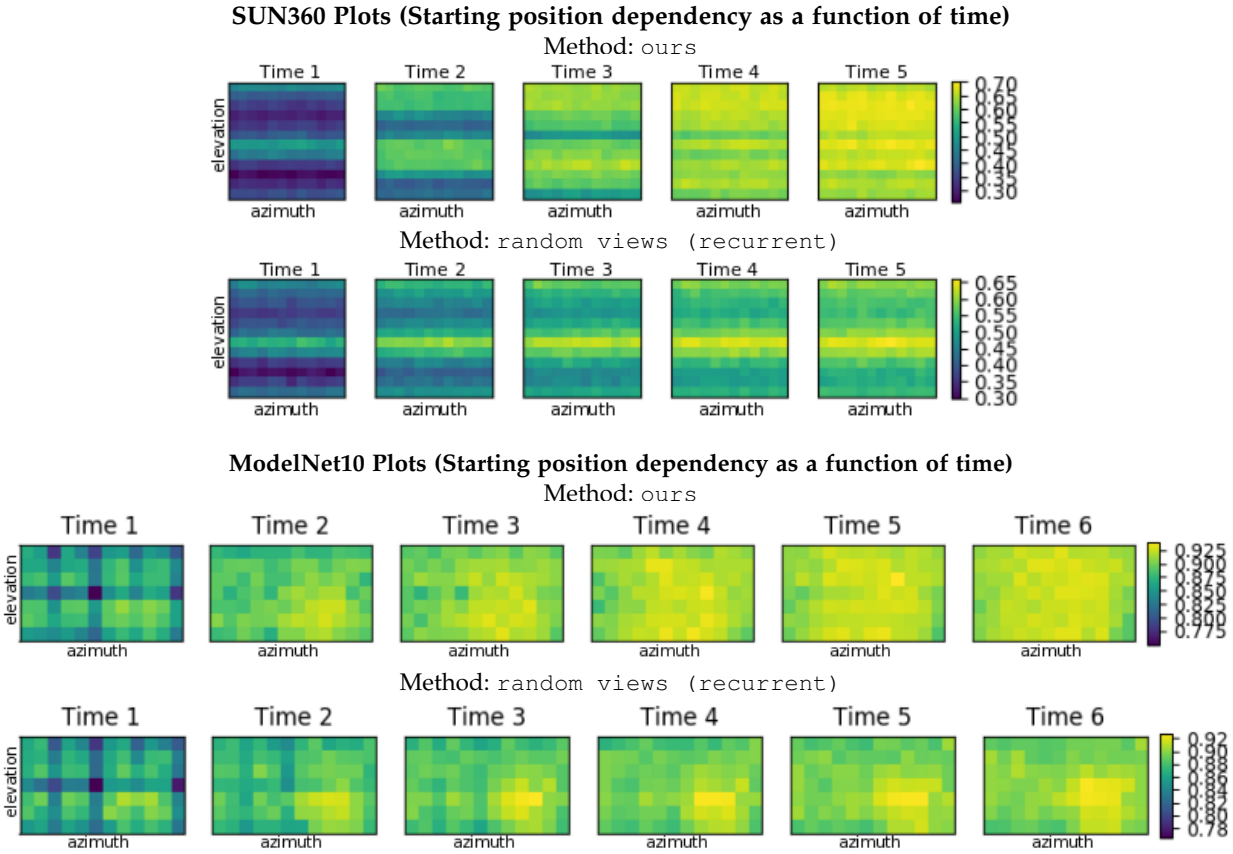

Fig. 15: (Best seen in color) Accuracy as a function of starting position, over time. For each timestep t, a color-coded viewgrid (12×12 on SUN360, 7×12 on ModelNet) presents the accuracy after t views, when starting from each position in the viewgrid. The elevation axis spans -90 to +90°, where 0°corresponds to the horizontal. The azimuth axis spans 0 to 360°. At early timesteps, accuracy is strongly dependent on starting position since some views are more discriminative than others. However, good action policies must quickly recover from poor starting views, so that active recognition accuracies become less dependent on starting position over time. On both SUN360 active scene recognition and ModelNet10 active object category recognition, our approach achieves this much more efficiently than `random views (recurrent)`.

As observed in our experiments, the look-ahead regularizer proved substantially useful only when training data was limited. This raises the question: are there better ways to utilize the look-ahead ability within an active recognition system rather than as just a regularizer? The look-ahead module is at its essence a model of state transitions within the system (considering the aggregator output as the "state"), since it models the next state, conditional on current state and action. This view of the look-ahead module brings into focus the connection to model-based reinforcement learning [52], which aims to more explicitly use models of the state transition and reward dynamics of the environment in reinforcement learning systems. This is a promising direction for potential future investigation.

Acknowledgments

This research was supported in part by ONR PECASE Award N00014-15-1-2291, a Google Faculty Research Award, and a Samsung Fellowship. We thank TACC for computing facilities, and Mohsen Malmir and Jianxiong Xiao for their assistance sharing GERMS and SUN360 data.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015.
[2] P Agrawal, AV Nair, P Abbeel, J Malik, and S Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *NIPS*, 2016.
[3] J Aloimonos, I Weiss, and A Bandyopadhyay. Active vision. In *IJCV*, 1988.
[4] A Andreopoulos and J Tsotsos. A theory of active object localization. In *ICCV*, 2009.
[5] A Andreopoulos and J Tsotsos. 50 years of object recognition: Directions forward. In *CVIU*, 2013.
[6] J Ba, V Mnih, and K Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015.
[7] R Bajcsy. Active perception. In *Proceedings of the IEEE*, 1988.
[8] D Ballard. Animate vision. In *Artificial Intelligence*, 1991.
[9] L Bazzani, H Larochelle, V Murino, J-A Ting, and N de Freitas. Learning attentional policies for tracking and recognition in video with deep networks. In *ICML*, 2011.
[10] J Bergstra and Y Bengio. Random search for hyper-parameter optimization. In *JMLR*, 2012.
[11] H Borotschnig, L Paletta, M Prantl, A Pinz, et al. Active object recognition in parametric eigenspace. In *BMVC*, 1998.
[12] M Bowling, A Ghodsi, and D Wilkinson. Action respecting embedding. In *ICML*, 2005.
[13] F Brentano. *Psychologie vom empirischen Standpunkte*. 1874.
[14] N. Butko and J. Movellan. Optimal scanning for faster object detection. In *CVPR*, 2009.
[15] J. Caicedo and S. Lazebnik. Active object localization with deep reinforcement learning. In *ICCV*, 2015.

[16] F Callari and F Ferrie. Active object recognition: Looking for differences. In *IJCV*, 2001.
[17] C Chen, A Seff, A Kornhauser, and J Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *ICCV*, 2015.
[18] TS Cohen and M Welling. Transformation properties of learned visual representations. In *arXiv preprint arXiv:1412.7659*, 2014.
[19] J Denzler and C M Brown. Information theoretic sensor data selection for active object recognition and state estimation. In *TPAMI*, 2002.
[20] S Dickinson, H Christensen, J Tsotsos, and G Olofsson. Active object recognition integrating attention and viewpoint control. In *CVIU*, 1997.
[21] W Ding and GW Taylor. Mental rotation by optimizing transforming distance. In *NIPS DL Workshop*, 2014.
[22] C Finn, I Goodfellow, and S Levine. Unsupervised learning for physical interaction through video prediction. In *NIPS*, 2016.
[23] J Flynn, I Neulander, J Philbin, and N Snavely. DeepStereo: Learning to predict new views from the world's imagery. In *CVPR*, 2016.
[24] A G Garcia, A Vezhnevets, and V Ferrari. An active search strategy for efficient object detection. In *CVPR*, 2015.
[25] S Gupta, J Davidson, S Levine, R Sukthankar, and J Malik. Cognitive mapping and planning for visual navigation. In *CVPR*, 2017.
[26] S Helmer, D Meger, P Viswanathan, S McCann, M Dockrey, P Fazli, T Southey, M Muja, M Joya, J Little, et al. Semantic robot vision challenge: Current state and future directions. In *IJCAI workshop*, 2009.
[27] D Jayaraman and K Grauman. Learning image representations tied to ego-motion. In *ICCV*, 2015.
[28] D Jayaraman and K Grauman. Look-ahead before you leap: End-to-end active recognition by forecasting the effect of motion. In *European Conference on Computer Vision*, pages 489–505. Springer, 2016.
[29] D Jayaraman and K Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *CVPR*, 2016.
[30] Dinesh Jayaraman, Ruohan Gao, and Kristen Grauman. Unsupervised learning through one-shot image-based shape reconstruction. *CoRR*, abs/1709.00505, 2017.
[31] Dinesh Jayaraman and Kristen Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. *CVPR*, 2018.
[32] E Johns, S Leutenegger, and AJ Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *CVPR*, 2016.
[33] S. Karayev, T. Baumgartner, M. Fritz, and T. Darrell. Timely object recognition. In *NIPS*, 2012.
[34] TD Kulkarni, W Whitney, P Kohli, and JB Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015.
[35] YJ Lee, J Ghosh, and K Grauman. Discovering important people and objects for egocentric video summarization. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1346–1353. IEEE, 2012.
[36] S Levine, C Finn, T Darrell, and P Abbeel. End-to-End training of deep visuomotor policies. In *ICRA*, 2015.
[37] T-Y Lin, M Maire, S Belongie, J Hays, P Perona, D Ramanan, P Dollár, and CL Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.
[38] Mohsen Malmir, Karan Sikka, Deborah Forster, Javier Movellan, and Garrison W Cottrell. Deep Q-learning for active recognition of GERMS. In *BMVC*, 2015.
[39] S. Mathe, A. Pirinen, and C. Sminchisescu. Reinforcement learning for visual object detection. In *CVPR*, 2016.
[40] A Mishra, Y Aloimonos, and C Fermuller. Active segmentation for robotics. In *IROS*, 2009.
[41] V Mnih, N Heess, A Graves, and K Kavukcuoglu. Recurrent models of visual attention. In *NIPS*, 2014.
[42] J Oh, X Guo, H Lee, RL Lewis, and S Singh. Action-conditional video prediction using deep networks in atari games. In *NIPS*, 2015.
[43] L Paletta and A Pinz. Active object recognition by view integration and reinforcement learning. In *RAS*, 2000.
[44] V Ramanathan and A Pinz. Active object categorization on a humanoid robot. In *VISAPP*, 2011.
[45] M Ranzato, A Szlam, J Bruna, M Mathieu, R Collobert, and S Chopra. Video (language) modeling: a baseline for generative models of natural videos. In *arXiv preprint arXiv:1412.6604*, 2014.
[46] O Russakovsky, J Deng, H Su, J Krause, S Satheesh, S Ma, Z Huang, A Karpathy, A Khosla, M Bernstein, AC Berg, and L Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. In *IJCV*, 2015.
[47] B Schiele and J Crowley. Transinformation for active object recognition. In *ICCV*, 1998.
[48] P Sermanet, A Frome, and E Real. Attention for fine-grained categorization. In *arXiv*, 2014.
[49] K Simonyan and A Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv*, 2014.
[50] S Soatto. Actionable information in vision. In *ICCV*, 2009.
[51] J Stober, R Miikkulainen, and B Kuipers. Learning geometry from sensorimotor experience. In *ICDL*, 2011.
[52] RS Sutton and AG Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
[53] C Szegedy, W Liu, Y Jia, P Sermanet, S Reed, D Anguelov, D Erhan, V Vanhoucke, and A Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
[54] C Vondrick, H Pirsiavash, and A Torralba. Anticipating the future by watching unlabeled video. In *CVPR*, 2016.
[55] J Walker, A Gupta, and M Hebert. Dense optical flow prediction from a static image. In *ICCV*, 2015.
[56] M Watter, JT Springenberg, J Boedecker, and M Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *NIPS*, 2015.
[57] D Wilkes and J Tsotsos. Active object recognition. In *CVPR*, 1992.
[58] R Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *JMLR*, 1992.
[59] Z Wu, S Song, A Khosla, F Yu, L Zhang, X Tang, and J Xiao. 3D ShapeNets: A deep representation for volumetric shape modeling. In *CVPR*, 2015.
[60] J Xiao, K Ehinger, A Oliva, A Torralba, et al. Recognizing scene viewpoint using panoramic place representation. In *CVPR*, 2012.
[61] K Xu, J Ba, R Kiros, K Cho, A Courville, R Salakhutdinov, R Zemel, and Y Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.
[62] X Yu, C Fermuller, CL Teo, Y Yang, and Y Aloimonos. Active scene recognition with vision and language. In *CVPR*, 2011.
[63] Y Zhu, R Mottaghi, E Kolve, JJ Lim, A Gupta, L Fei-Fei, and A Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 2017.

Dinesh Jayaraman is a postdoctoral scholar at UC Berkeley. He received his PhD from UT Austin in 2017. His research interests are broadly in visual recognition and machine learning. In the last few years, Dinesh has worked on visual learning and active recognition in embodied agents, unsupervised representation learning from unlabeled video, visual attribute prediction, and zero-shot categorization. He received the Best Application Paper Award at the Asian Conference on Computer Vision 2016 for work on automatic cinematography, the Samsung PhD Fellowship for 2016-17, a UT Austin Microelectronics and Computer Development Fellowship, and a Graduate Dean's Prestigious Fellowship Supplement Award for 2016-17.

Kristen Grauman Kristen Grauman is a Professor in the Computer Science Department at UT Austin. She received her Ph.D. from MIT in 2007. She is a recipient of the Sloan Research Fellowship, NSF CAREER and ONR Young Investigator awards, the 2013 PAMI Young Researcher Award, the 2013 IJCAI Computers and Thought Award, a 2013 Presidential Early Career Award for Scientists and Engineers (PECASE), and a 2017 Helmholtz Prize. She and her collaborators were recognized with best paper awards at CVPR 2008, ICCV 2011, and ACCV 2016. She served as a Program Chair of CVPR 2015 and currently serves as an Associate Editor in Chief of PAMI and Program Chair of NIPS 2018.