Supplementary Material: A Probabilistic Hard Attention Model For Sequentially Observed Scenes

Samrudhdhi B. Rangrej
samrudhdhi.rangrej@mail.mcgill.ca
James J. Clark
james.j.clark@mcgill.ca

Centre for Intelligent Machines (CIM)
McGill University
Montreal, Quebec, Canada

1 Architecture

Table 1 shows the architecture of various components of our model. Note that all modules use a small number of layers. F_g uses the least possible layers to achieve the effective receptive field equal to the area of a single glimpse. The decoder D uses the smallest number of *ConvTranspose* and *Conv* layers to generate the feature maps of required spatial dimensions and refine them based on the global context. The encoder S uses flow layers according to the complexity of the dataset. All other modules use a single linear layer. We implement linear layers using 1×1 convolution layers. The dimensionality of features f, hidden state h and latent representation z for various datasets are mentioned in Table 2.

2 Optimization Details

We train our model in three stages until convergence. In the first stage we pre-train F, R and C. In the second stage, we pre-train S and D while keeping F, R, and C frozen. In the third stage, we finetune all modules end-to-end. Unless stated otherwise, we use the same setting for all three stages. We trained our models on a single Tesla P100 GPU with 12GB of memory or a single Tesla V100 GPU with 16GB of memory.

Data Preparation. We augment training images using random crop, scale, horizontal flip and color jitter transformations, and map pixel values in range $[-1, 1]$. We use a batch-size of 64. We use the same scheme for all datasets.

Loss Function. We compute an average loss across batch and time. We set the hyperparameters α and β of the loss function as follows. The hyperparameter α is set to the inverse of dimensionality of the latent representation z. The hyperparameter β is set to 32, 16, 16, 8 and 8 for SVHN, CINIC-10, CIFAR-10, CIFAR-100 and TinyImageNet respectively.

	Module	Architecture
Recurrent feature aggregator	$\mathcal{F}_l$	$Conv_{k=1}(\cdot)$
	$\mathcal{F}_g$	$n_g \times \{BN(LeakyReLU(Conv_{k=3}(\cdot))\}$ $Conv_{k=2}(\cdot)$
	$F(g,l)$	$\mathcal{F}_g(g)+\mathcal{F}_l(l)$
	$\mathcal{F}_h$	$Conv_{k=1}(\cdot)$
	$\mathcal{F}_f$	$Conv_{k=1}(BN(LeakyReLU(\cdot)))$
	$R(h,f)$	$LN(LeakyReLU(\mathcal{F}_h(h)+\mathcal{F}_f(f)))$
Classifier	C	$Softmax(Conv_{k=1}(Dropout_{p=0.5}(\cdot)))$
Partial VAE	S	$n_s \times (ActNorm(Flip(NSF(\cdot))))$
	D	$3 \times \{LN(LeakyReLU(ConvTranspose_{k=3}(\cdot)\}$ $5 \times \{LN(LeakyReLU(Conv^{p=1}_{k=3}(\cdot)\}$ $Conv^{p=1}_{k=3}(\cdot)$

Table 1: Architecture of our model. $k = kernel\_size$, $p = padding$. n_g is set to 3 for SVHN, CINIC-10, CIFAR-10 and CIFAR-100, and set to 7 for TinyImageNet. n_s is set to 4 for SVHN, CINIC-10 and CIFAR-10, and set to 6 for CIFAR-100 and TinyImageNet. BN = Batch Normalization [5]. LN = Layer Normalization [1]. NSF = Neural Spline Flows [3]. ActNorm layer is presented in [7].

	f	h	z
SVHN	128	512	256
CINIC-10	128	512	256
CIFAR-10	128	512	256
CIFAR-100	512	2048	1024
TinyImageNet	512	2048	1024

Table 2: Dimensionality of features f, hidden state h and latent representation z.

Optimizer. We use Adam optimizer [6] with the default setting of $(\beta_1, \beta_2) = (0.9, 0.999)$. In the first training stage, we use a learning rate of 0.001 for all datasets. In the second and third training stages, we use a learning rate of 0.001 for SVHN, CIFAR-10, and CINIC-10, and a learning rate of 0.0001 for CIFAR-100 and TinyImageNet. We divide the learning rate by 0.5 at a plateau.

3 Additional Experiments

3.1 Accuracy as a function of area observed in an image

At a time, a hard attention model observes only a small region of the input image through a glimpse. As time progresses, the model observes more area in the image through multiple glimpses. We compare the accuracy of various models as a function of the area observed in an image (see Figure 1). RAM+ consistently outperforms RAM on various datasets. We observe that the Random baseline performs better than the RAM and RAM+ on comparatively more challenging datasets. As discussed in the main paper, the Random baseline dedicates an entire latent space to the classifier. Our model, RAM+, and RAM use a common latent space for a classifier and a Partial VAE or a controller. Finally, for any given value of the

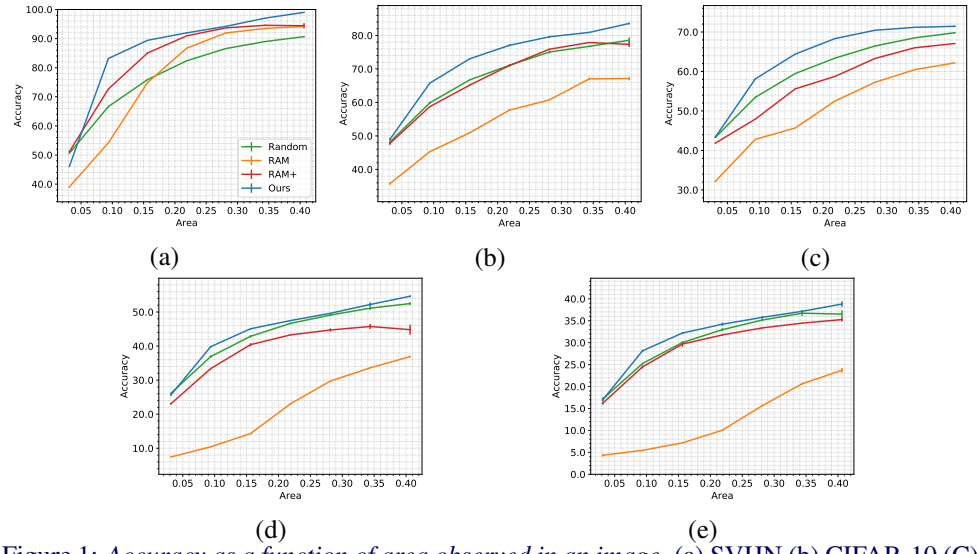


Figure 1: *Accuracy as a function of area observed in an image*. (a) SVHN (b) CIFAR-10 (C) CINIC-10 (d) CIFAR-100 (e) TinyImageNet. Hard attention models observe only a fraction of area in the image through multiple glimpses. Accuracy of the models increase as they observe more area. Our model achieves the highest accuracy for any given value of the observed area. Results are averaged over three independent runs.

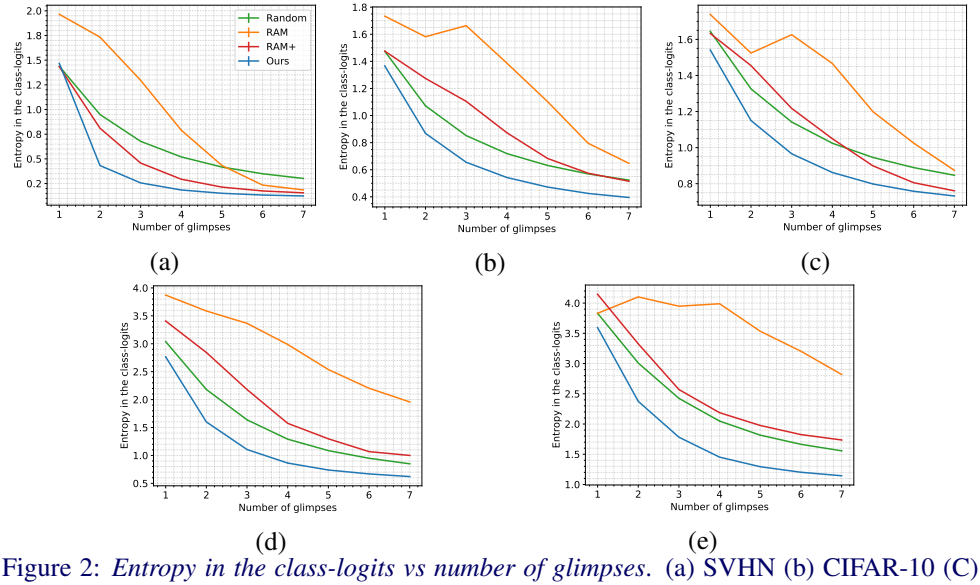


Figure 2: *Entropy in the class-logits vs number of glimpses*. (a) SVHN (b) CIFAR-10 (C) CINIC-10 (d) CIFAR-100 (e) TinyImageNet. Entropy in the class-logits decreases as the hard attention models acquire more glimpses. Our model achieves the lowest entropy in the predictions at all times. Results are averaged over three independent runs.

observed area, our model achieves the highest accuracy. The result suggests that our model captures glimpses on the regions that provide more useful information about the class label

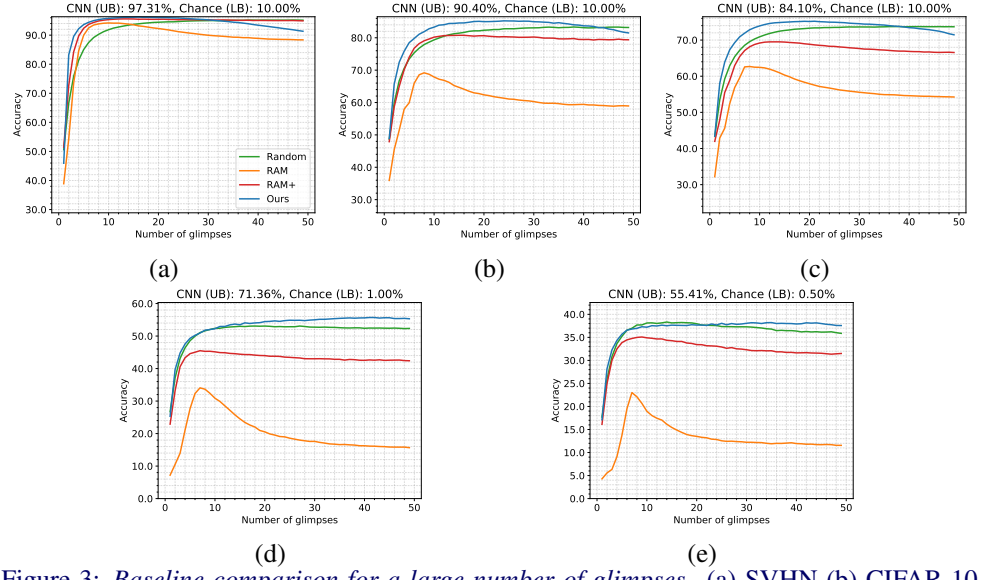


Figure 3: *Baseline comparison for a large number of glimpses.* (a) SVHN (b) CIFAR-10 (C) CINIC-10 (d) CIFAR-100 (e) TinyImageNet. Results are averaged over three runs. Our model generalizes to a large number of glimpses and achieves the highest accuracy having seen an optimal number of glimpses.

than the baseline methods.

3.2 Entropy in the class-logits as a function of number of glimpses

An efficient hard attention model should become more confident in predicting a class label with small number of glimpses. We measure confidence using entropy in the predicted class-logits; lower entropy indicates higher confidence. Figure 2 shows entropy in the class-logits predicted by various models with time. Irrespective of the glimpse acquisition strategy, all models reduce entropy in their predictions as they acquire more glimpses. In case of RAM, the trend is inconsistent for initial time steps. Recall that RAM optimizes loss only at the last time step, unlike other models that optimize losses for all time steps. Consistent with the previous analyses, RAM+ outperforms RAM on all datasets, and the Random baseline outperforms RAM and RAM+ on complex datasets. Our model achieves lower entropy with smaller number of glimpses compared to the baseline methods. The result indicates that our model acquires glimpses that help the most in reducing the uncertainty in the class-label prediction.

3.3 Generalization to a large number of glimpses

Recall that all models are trained for seven glimpses. In Figure 3, we assess their inference-time generalizability for a large number of glimpses by testing the models for T=49. Note that our method would have observed an entire image by then. We notice that, during the inference-time, RAM does not generalize well to a sequence of glimpses that are longer than the training time; [4, 9, 10] also make a similar observation. Our method and RAM+ show

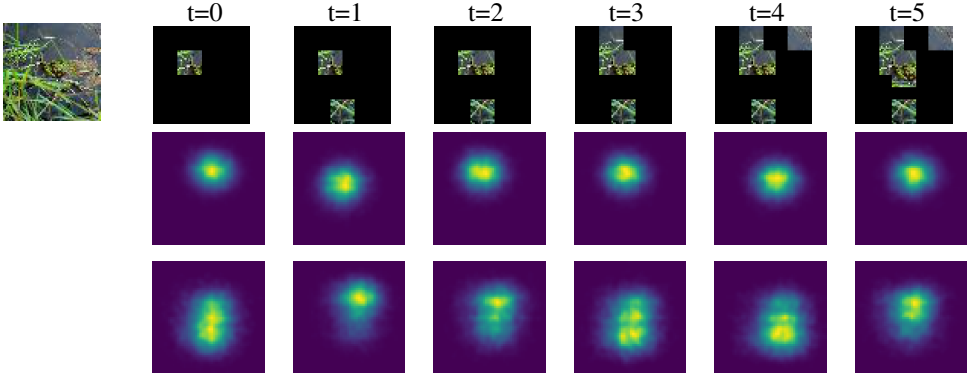
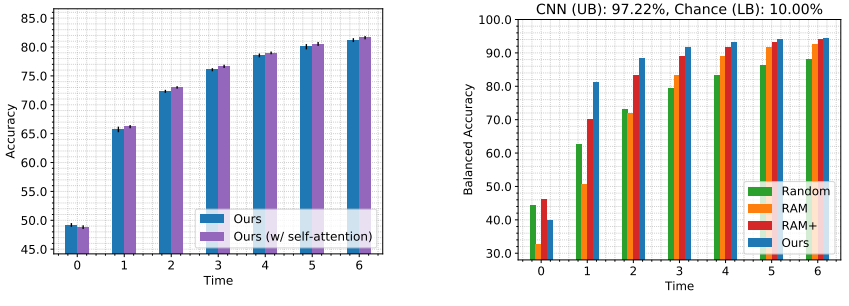


Figure 4: *TSNE projection of $q(z|h_t)$ estimated with and without normalizing flows for an example image from TinyImageNet dataset.* A complete image is shown in the first column for reference. Our model never observes a complete image. In columns two to seven: (top) glimpses observed by the models to compute h_t (middle) TSNE projection of $q(z|h_t)$ estimated without using normalizing flows. (bottom) TSNE projection of $q(z|h_t)$ estimated using normalizing flows. Normalizing flows capture a complex multimodal posterior.



(a) (b)

Figure 5: (a) *Accuracy with and without self-attention in recurrent network R for CIFAR-10.* Results are averaged over ten runs. (b) Baseline comparison using Balanced accuracy for SVHN. We observe the same trend as the one observed in section 4.1 in main paper.

greater generalizability than RAM. While the Random baseline is the most generalizable, our method outperforms it with an optimal number of glimpses. The accuracy achieved by our model with an optimal number of glimpses is lower than the CNN; however, the CNN is trained exclusively on complete images, and our model is not. Note that the primary motivation for the attention mechanism is to achieve higher accuracy using limited time and constrained resources. If time and resources are available, one can instead collect a large number of random glimpses and use a non-attentive CNN.

3.4 Visualization of $q(z|h_t)$ estimated with and without normalizing flows

Synthesizing a feature map of a complete image using only partial observations is an ill-posed problem with many solutions. We use normalizing flows in the encoder S to capture a complex multimodal posterior $q(z|h_t)$ that helps the decoder predict multiple plausible fea-

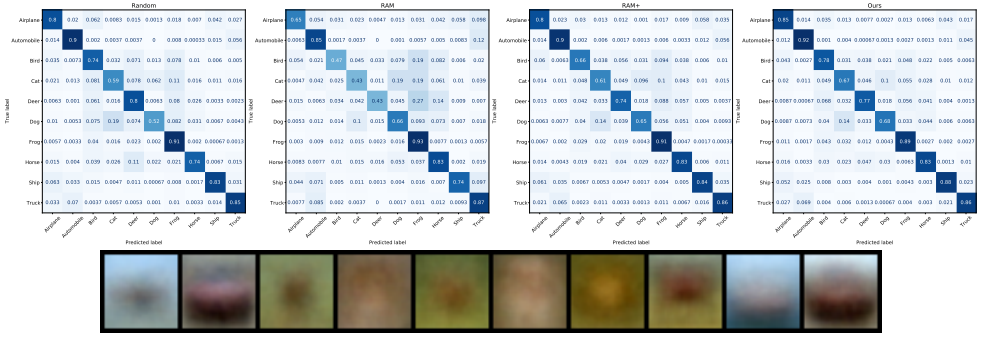


Figure 6: (Top) Confusion matrix at $t = 6$ and (Bottom) average image per class from CIFAR-10.

ture maps for a complete image. We perform an ablation study to analyze the importance of using normalizing flows. In Figure 4, we present TSNE [11] projections of $q(z|h_t)$ estimated with and without the use of normalizing flows. We can observe that the normalizing flows capture a complex multimodal posterior. Capturing multiple modes leads to a more accurate estimation of EIG and consequently higher performance.

3.5 Self-attention in R

Here, we experiment with self-attention in R. We adapt the method proposed in [8] for our problem. Figure 5(a) compares the performance of our model with and without self-attention. We observe only a marginal performance improvement due to self-attention, perhaps because the sequence of seven glimpses is relatively short. However, self-attention may achieve higher accuracy if the model is trained for longer sequences. Nevertheless, the results of this preliminary experiment are favorable, suggesting that exploring self-attention in R is a promising direction for future works.

3.6 Confusion matrix

In Figure 6, we display the confusion matrix of various methods for the CIFAR-10 dataset at t=6. We also show the average image for each class. We observe that the models are able to discern classes with similar color schemes, suggesting that they rely on complex high level features instead of simple low level features such as pixel color. Furthermore, the Random baseline, the RAM, and the RAM+ over-represent category 'frog', confusing it with many other categories. Our method does not suffer from this phenomenon.

3.7 Additional visualization

In Figure 7-9, we visualize the *EIG* maps and the glimpses observed by our model on CIFAR-10 images. The top rows in each plot show the entire image and the *EIG* maps for $t = 1$ to 6. The bottom rows in each plot show glimpses attended by our model. The model observes the first glimpse at a random location. Our model observes a glimpse of size 8×8. The glimpses overlap with the stride of 4, resulting in a 7×7 grid of glimpses. The

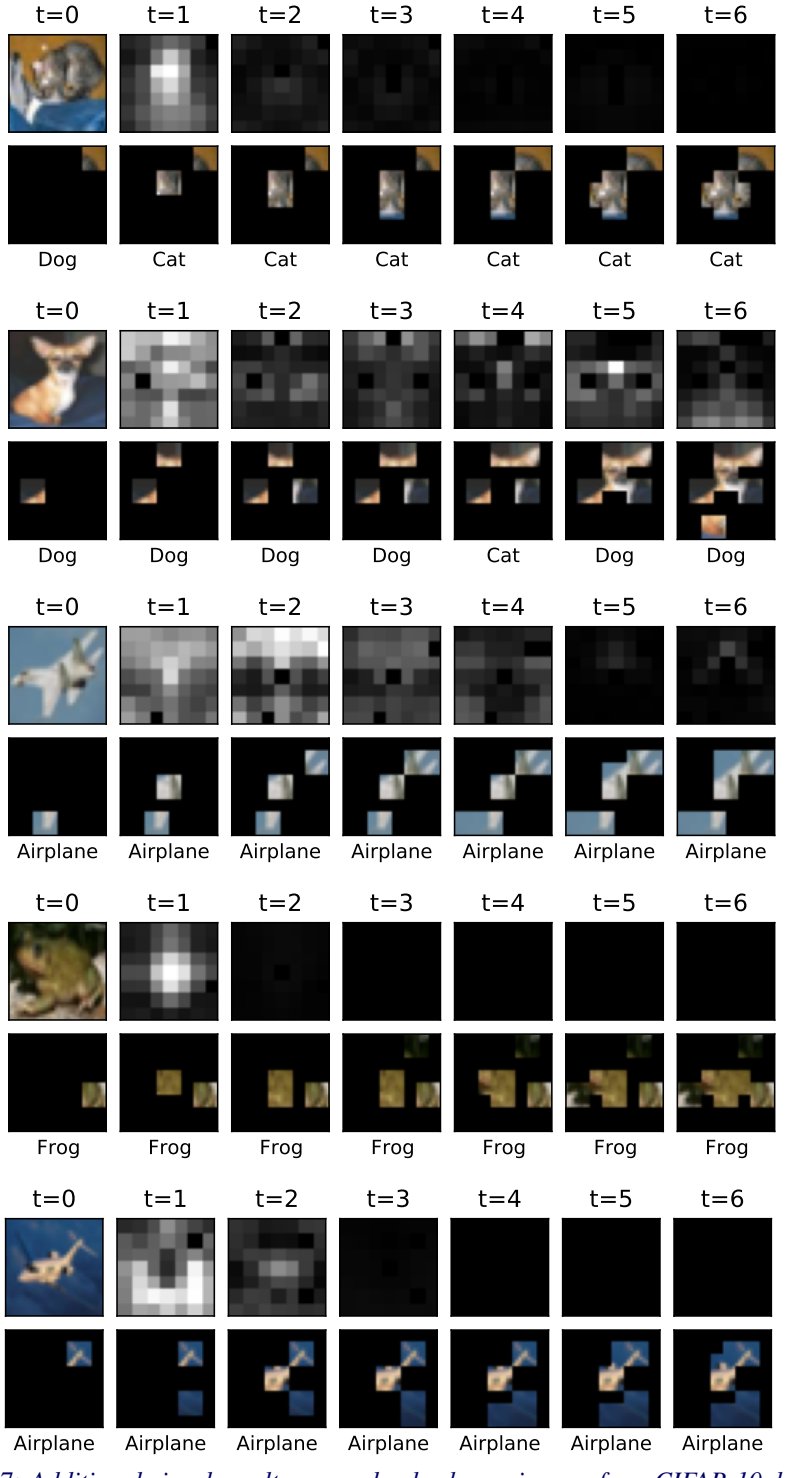


Figure 7: *Additional visual results on randomly chosen images from CIFAR-10 dataset.*

Figure 8: *Additional visual results on randomly chosen images from CIFAR-10 dataset.*

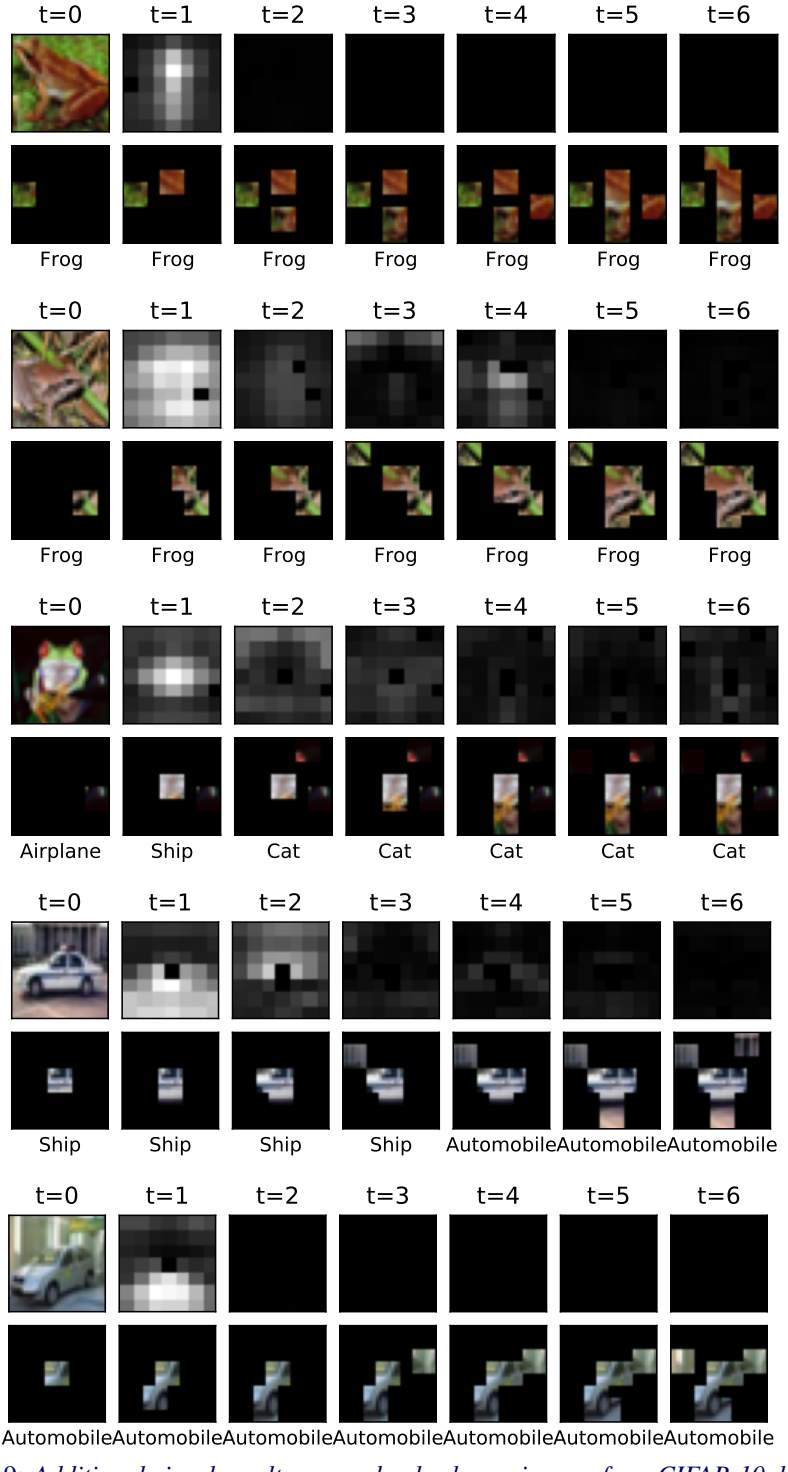


Figure 9: *Additional visual results on randomly chosen images from CIFAR-10 dataset.*

Figure 10: Visualization of glimpses observed by different methods on images from (top) SVHN (bottom) CIFAR-10. Refer to section 3.7 for explanation.

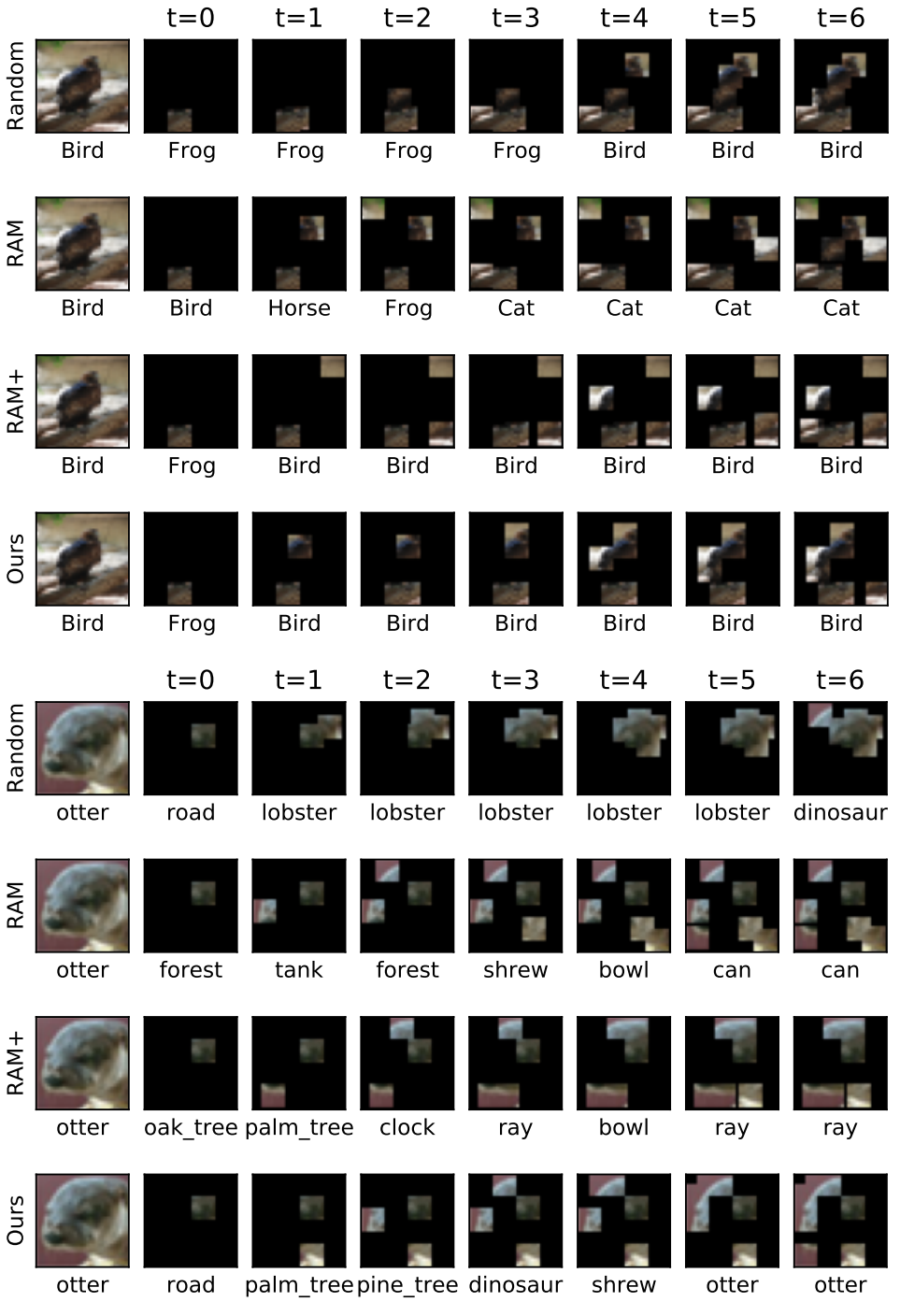


Figure 11: Visualization of glimpses observed by different methods on images from (top) CINIC-10 (bottom) CIFAR-100. Refer to section 3.7 for explanation.

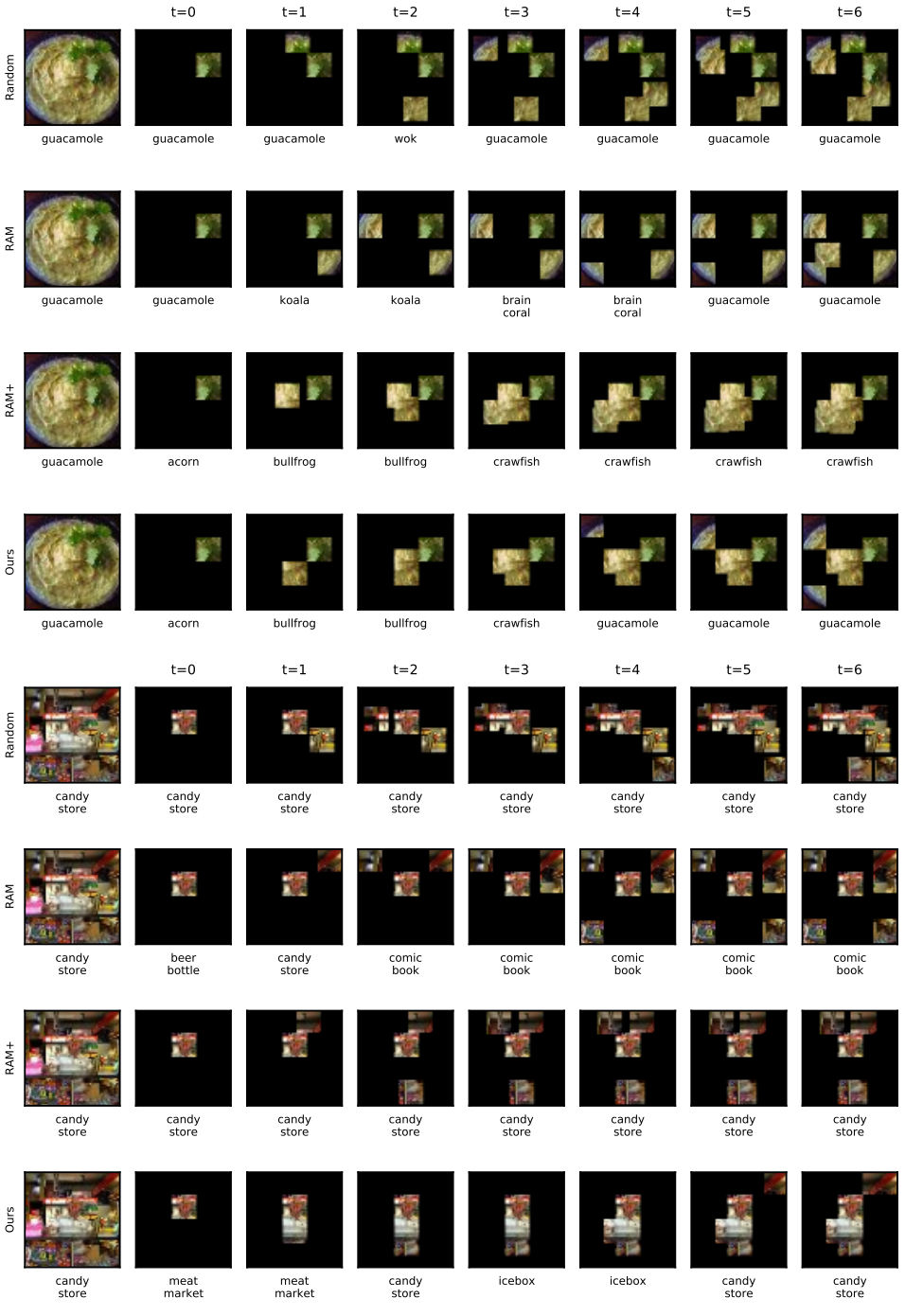


Figure 12: Visualization of glimpses observed by different methods on two images from TinyImageNet. Refer to section 3.7 for explanation.

EIG maps are of size 7×7 and are upsampled for the display. We display the entire image for reference; our model never observes the whole image. Additionally, in Figure 10-12, we visualize a series of glimpses selected by various models for one image from all datasets. A complete image is shown for reference. Compared methods do not observe the complete image. Ground Truth label is shown in the first column, below the complete image. Labels predicted after observing various glimpses are shown in the columns marked with time t.

4 Normalizing Flows

Normalizing Flows map a Gaussian distribution to a complex multi-modal distribution using a series of differentiable and invertible functions. We use conditional normalizing flows that map samples from $q(z_0|h_t)$ (Gaussian distribution) to $q(z_N|h_t)$ (a complex distribution) using a series of invertible functions $\{f_0, f_1, \ldots, f_{N-1}\}$ conditioned on h_t.

$$z_N = f_{N-1} \circ \cdots \circ f_1 \circ f_0(z_0); \text{ where } z_0 \sim q(z_0|h_t) \tag{1}$$

The relation between $q(z_0|h_t)$ and $q(z_N|h_t)$ is established using a change of variable formula.

$$q(z_N|h_t) = q(z_0|h_t) \prod_{n=0}^{N-1} |det(J_{f_n})|^{-1} \tag{2}$$

Where J_f is a Jacobian of f. We use $N = 3 \times n_s$, where values of n_s are given in Table 1. For all $n \in \{0, 3, \ldots, N-3\}$, we define f_n f_{n+1} and f_{n+2} using ActNorm [7], Flip [2] and Neural Spline Flows [3], respectively. Below, we provide a brief introduction on these three functions. To reduce clutter, we refer to z_n as z and f_n as f.

ActNorm [7]. An ActNorm layer performs an element-wise scaling and shifting of z.

$$f(z) = s \odot z + b \tag{3}$$

The scale parameter s and the shift parameter b are predicted by a neural network using h_t. We adapt a data-dependent initialization scheme for this network [7]. Specifically, we initialize the above neural network such that the predicted s and b yield $f(z)$ with unit variance and zero mean for the first batch. We can compute $|det(J_f)| = \prod_i |s(i)|$.

Flip [2]. A Flip layer simply reverses elements of z, i.e. $f(z) = reversed(z)$. The Jacobian determinant $|det(J_f)| = 1$.

Neural Spline Flows (NSF) [3]. An NSF performs element-wise transformations on z. Specifically, it transforms an element $z(i)$ using a monotonic piece-wise spline function f_i that is defined using parameters $\{\{w^{(k)}\}_{k=0}^{K}, \{v^{(k)}\}_{k=0}^{K}, \{\delta^{(k)}\}_{k=1}^{K-1}\}$. Refer to [3] for more details on these parameters. In auto-regressive NSF, a neural network predicts the parameters of f_i from the elements $z(j < i)$ and h_t. Then, we find a specific k for which $w^{(k)} < z(i) <$

$w^{(k+1)}$ and transform $z(i)$ as follows.

$$s = \frac{v^{(k+1)} - v^{(k)}}{w^{(k+1)} - w^{(k)}} \tag{4}$$

$$\xi(z(i)) = \frac{z(i) - w^{(k)}}{w^{(k+1)} - w^{(k)}} \tag{5}$$

$$f_i(\xi) = v^{(k)} + \frac{(v^{(k+1)} - v^{(k)})[s\xi^2 + \delta^{(k)}\xi(1-\xi)]}{s + [\delta^{(k+1)} + \delta^{(k)} - 2s]\xi(1-\xi)} \tag{6}$$

The derivative of f_i is defined as follows.

$$\frac{df_i}{dz(i)} = \frac{s^2[\delta^{(k+1)}\xi^2 + 2s\xi(1-\xi) + \delta^{(k)}(1-\xi)^2]}{[s + [\delta^{(k+1)} + \delta^{(k)} - 2s]\xi(1-\xi)]^2} \tag{7}$$

As NSF applies element-wise monotonic functions, $|det(J_f)| = \prod_i |\frac{df_i}{dz(i)}|$.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[3] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, pages 7511–7522, 2019.

[4] Gamaleldin Elsayed, Simon Kornblith, and Quoc V Le. Saccader: improving accuracy of hard attention models for vision. In *Advances in Neural Information Processing Systems*, pages 702–714, 2019.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018.

[8] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *International Conference on Learning Representations*, 2017.

[9] Jialin Lu. Revisit recurrent attention model from an active sampling perspective. *NeuroAI workshop, NeurIPS*, 2019.

[10] Pierre Sermanet, Andrea Frome, and Esteban Real. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054*, 2014.

[11] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.